Predrag Sidjanin

FOR AN URBAN ENVIRONMENT DESIGN TOOL

TR 3699 S

PROPOSITIONS

Attached to the thesis

A COGNITIVE FRAMEWORK FOR AN URBAN ENVIRONMENT DESIGN TOOL

Predrag SIDJANIN

1) Throughout history, love and caprice have been stimuli for changing the world. A stimulus in art and science is curiosity together with a lot of patience.

2) Art and science may differ in their means, methods and goals. They may also differ in their products and in the value of their products. But they have one thing in common – pleasure in the creative process.

3) The task of art and science is to push back the borders of existing knowledge, a little at least. However, this happens very rarely.

4) Creativity in art and in science is not only a matter of technique or investigation. Sometimes it is based on a moment of 'vision' in which a technique is employed to provide a logical expression of the path to the successful result.

5) The product of a research project is not related exclusively to its established goal. It might be used to satisfy unpredictable goals.

6) It the early 1960s, Kevin Lynch developed the theory of 'urban form' and the concept of 'cognitive mapping'. At that time it was not clear how

these theories could be applied. Today, developments in computer science demonstrate the practicality of Lynch's ideas.
Creative theory should be encouraged even if it is not clear at that moment how it can be applied.

7) Recent computer applications [GIS, GPS] can assist people to navigate through space, to locate moving objects in space and to memorize all components of an object in a space. Soon computers will be able to analyze the cognitive maps of people and through them guide behavior and criticize space arrangements.

8) Object-oriented database systems can represent the 'universe of discourse' very precisely. Visualization techniques, such as computer simulation, are becoming more realistic than the surrounding reality [only on screen]. Together, they will provide a very powerful tool in the near future, which will improve different aspects of human activities.

9) A poem can be written in nine minutes. A painting can be painted in nine days. A baby can definitely be born in nine months. In science, we need years before we can say that something is 'born'.

STELLINGEN

Behorend bij het proefschrift

A COGNITIVE FRAMEWORK FOR AN URBAN ENVIRONMENT DESIGN TOOL

Predrag SIDJANIN

1) In de loop van de geschiedenis waren liefde en eigenzinnigheid prikkels die de wereld veranderden. De prikkel in kunst en wetenschap is nieuwsgierigheid gepaard met veel geduld.

2) Kunst en wetenschap kunnen verschillen in hun middelen, methodes en doelstellingen. Zij kunnen ook verschillen wat betreft hun producten en de waarde van hun producten. Maar zij hebben een ding gemeen – plezier in het creatieve proces.

3) De taak van kunst en wetenschap is het – tenminste iets - verleggen van de grenzen van het bestaande weten. Maar dit gebeurt hoogst zelden.

4) Creativiteit in kunst en wetenschap is niet slechts techniek of onderzoek. Soms is zij gebaseerd op het moment van "visioen" waarin de techniek wordt gebruikt om een logische expressie te vormen van het traject naar een succesvol resultaat.

5) Het product van een onderzoeksproject is niet enkel en alleen met het gestelde doel verbonden. Het kan worden gebruikt om onvoorspelbare doelen te bevredigen.

6) In de vroege jaren 60 ontwikkelde Kevin Lynch een theorie van "urban form" en van "cognitive mapping". Toentertijd was niet duidelijk hoe deze theorieën zouden kunnen worden toegepast. Heden tonen de ontwikkelingen in de computer-wetenschap de bruikbaarheid van de ideeën van Lynch.
Creatieve theorie zou moeten worden aangemoedigd zelfs als op het moment niet duidelijk is hoe zij kan worden toegepast.

7) De nieuwe computertoepassingen [GIS, GPS] kan mensen erbij helpen om door de ruimte te navigeren, om bewegende objecten in de ruimte te vinden en om alle componenten van een object in de ruimte te onthouden. Binnenkort zullen computers in staat zijn om de "cognitive maps" van mensen te analyseren en op die manier hun gedrag te leiden en inrichting van de ruimte te bekritiseren.

8) De object-georienteerde database-systemen kunnen de "discourse van het universum" zeer nauwkeurig weergeven. De visualiseringstechnieken zoals computer-simulatie, worden realistischer dan de realiteit die hen omgeeft [slechts op het scherm]. Samen zullen zij in toekomst een zeer krachtig instrument opleveren dat verscheidene aspecten van menselijke activiteiten zal verbeteren.

9) Er verschijnt een gedicht dat in negen minuten geschreven kan zijn. Een schilderij kan in negen dagen geschilderd worden. Een kind kan zeer zeker in negen maanden ter wereld komen. In de wetenschap hebben wij jaren nodig eer wij kunnen zeggen dat "iets is geboren".

TR 3699

3205
[illegible]
30[illegible]

A COGNITIVE FRAMEWORK FOR AN URBAN ENVIRONMENT DESIGN TOOL

PROESCHRIFT

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus prof. ir. K.F. Wakker,
voorzitter van het College voor Promoties,
in het openbaar te verdedigen op dinsdag 5 juni 2001 om 13.30 uur

door Predrag SIDJANIN
diplomirani inženjer arhitektute, Master of Technological Design
geboren te Novi Sad, Joegoslavië.

CIP-GEGEVENS KONINKLIJKE BIBLIOTHEEK, DEN HAAG

Predrag, Sidjanin

A cognitive framework for an urban environment Design Tool / Predrag Sidjanin. - Delft: Predrag Sidjanin - Ill.
Doctoral Dissertation Delft University of Technology. - With ill., tab., index, ref. - With summary in Dutch.
ISBN 90-90148620
Keywords: Architectural Theory, Urban Design, Design Tool, Information Systems, Object-oriented Database Systems |OODBS|

To the memory of my father,
to the love of my mother and my daughters,
and to the respect of my partner.

PREFACE AND ACKNOWLEDGEMENT

The idea for this research was the product of a discussion I had with Prof. Tzonis right after I finished my Master's thesis (in 1995). In my Master's thesis I produced an integration of Geographical Information Systems [GIS] and virtual reality [VR]. From several innovative things discussed in my thesis Prof. Tzonis *saw* 'cognitive mapping' as a new potential for further research. Cognitive mapping, which was something I was not even very familiar with at that time, became our focus and resulted in this research. Everything started with cognitive mapping and Lynch's theory behind it, and the need to implement and improve this nice idea in design practice. Later, during the investigation for the research, the idea of a new tool for urban designers to bridge an existing gap in the profession became clearer. I understood that it would be very difficult to integrate this somewhat psychological aspect of the urban environment and its characteristics into a computational tool. From this point the idea was that the research should be a kind of experiment in which we could see whether or not it is possible to integrate cognitive mapping and computational techniques. It was a challenging process, which gained new impulses after hearing a lecture given by Prof. Gerhard to the DKS group in 1997, on the possibilities of recently developed database systems. After this lecture, regular contact was maintained with Prof. Gerhard and the discussion on the integration of cognitive mapping and computational techniques found powerful support in object-oriented database systems. Professor Gerhard became my second supervisor.

It has been very pleasant and inspirational for me, as a professional architect and artist, to discuss, work and collaborate with my supervisors Prof. Gerhard and Prof. Tzonis. I have learned a lot from them and they have influenced me for the rest of my life, not just for this research. They both raised my research to higher standards. They were good leaders and guides and they became friends with whom I can discuss many other things too.

I would also like to express my gratitude to my examination committee: Prof. M. Jenks, Prof. Prof. P. van Oosterom, Prof. J. Rosemann, Prof. P. Drewe and Prof. E. Jansen. I also wish to thank Prof. W. L. Porter and Prof. S. Anderson, both from MIT, with whom I had a good discussions about my research, especially on Lynch and his work. I would also like to thank Prof. S. J. Doorman for his lectures on related phylosophical issues, which he gave over the last couple of years to our DKS group. These lectures were very informative and inspirational for me.

I wish to express my deep gratitude to the memory of Prof. Donald Schön, who left us so early. It was a real privilege for me, as was the case with other fellows of the DKS group, to discuss my research with this great man and outstanding scientist.

My research also benefited from the very important influence and help of Mattie Sim, Natalio Simon and Peter Kist from Bitbybit Information Systems, the company which developed Perspective-DB technology, which I used in my research with their permission. Many thanks to all of you!

Thanks also to all my old and recent colleagues at the Design Knowledge Research Center, with whom I had friendly discussions on various topics during our regular seminars: John Heintz, Peter Donker, Sinan Inanç, Karina Moraes Zarzar, Asaf Friedman, Joo-Hwa Bay and Liane Lefaivre. Many thanks to our secretary Janneke Arkenstein for help with all the administrative and travel procedures.

I would especially like to thank my English editor Mr. Marcus Richardson, without whose help this thesis would not be before us. Marcus will also be my *paranymphus* at my defense. Many thanks to Mrs. Eva Bodor for Dutch translations. Many thanks also to Mr. Klaus-dieter Michel for his creative support with the cover design.

There were also many other people, mostly professionals and academics, whom I met during visits to different conferences who gave me useful comments.

Finally I would like to thank Alexandra Tisma, my friend, partner in life and colleague, for her help, understanding and patience.

Predrag Sidjanin
The Hague, March 2001

CONTENTS

Part one - **General introduction**

CHAPTER 1

INTRODUCTION

A Design Tool for urban design and planning

The goal of this research is to develop a *conceptual model* of a Design Tool that will help in analyzing, evaluating and controlling the visual quality of the urban environment in urban design practice. The research will involve the integration of knowledge from design theory, cognitive science and the use of computational techniques. With the support of a theoretical platform, this integration will lead to a *conceptual model* of the Design Tool. The *conceptual model* of the Design Tool will be based on the idea of cognitive mapping. The computational aspect of it will be provided by an object-oriented database system (OODB).

In current architectural, urban design and planning practice, the role of new information technology has become important. The use of computer models has been going on in architectural design and urban planning since the 1950's. Early architectural drafting was transformed through computer - aided design (CAD) packages, such as the now widely used *AutoCAD*. In urban planning, computation began with municipal information systems and landuse transportation modeling. In the last decade there have been major developments in tools for visualization and representation, particularly through the development of geographical information systems (GIS). With the development of GIS information technology, it has become possible to link up with available 2D or 3D spatial data through new desktop packages such as *ArcInfo, ArcView* or *MapInfo*, which have become standard. On the other hand, urban design was for many years untouched by computational developments. Later on we will explain the main reasons for this.

General goal of the research and definition of the terms

The general goal of this multidisciplinary research is to improve design methods and theory by developing new knowledge about designing as a cognitive process embedded in social and cultural practice. This is also the main policy of the Design Knowledge System Research Center[1]. The Center, to which this research is affiliated, emphasizes the development of new design tools to improve the quality and creativity of design practice, based on a deeper scientific knowledge of design thinking. To achieve these goals, this research combines investigations into design theories, cognitive science and computational science within a unique framework.

According to these main policies of the DKS Research Center, the goal of this research is to develop a *conceptual model* of a Design Tool. A *conceptual model* for a Design Tool provides knowledge that supports the Design Tool.

To avoid possible misuses in terminology, we need to explain in what sense we will use some common terms in this research. First we will explain the terms 'Design Tool' and 'system', and then we will present the meanings of the terms 'urban design', 'tool design' and 'database design', and at the end we will explain the difference between a 'conceptual model' and a 'functional model'. It is important to specify the distinction between these terms because this research works with different disciplines. All of the working disciplines

[1] *Design Knowledge Systems Research Center 1985-2000*, Technische Universiteit Delft, Faculty of Architecture, Delft 2000

use the same terms with different meanings. Here we will explain the meaning of these terms in our research as well as in these disciplines.

The term 'Design Tool' represents a software application that can help users in their design practice, and it will be guided by the results of this research. The proposed 'Design Tool' will consist of several 'sub-tools', such as a tool for object manipulation, a tool for decision-making, a tool for modeling, and so forth. All 'sub-tools' will be integrated in the Design Tool, without strictly defined boundaries between them and they will not always be visible in their particularity to the users. In this sense, the 'Design Tool' will be an integration of different sub-tools, which will work in a certain order or simultaneously together and will be at the disposal of users as one integrated software application.

The term 'system' can be defined simply as organized or established ideas, principles or procedures, and as such is usually intended to explain a working arrangement; it will also be used in the field of our research disciplines. In urban design practice the term 'system' is used for a street system, traffic system, city network system, and so forth. In the field of cognitive science the term 'system' is used in the sense of perceptive system, nerve system, memory system, behavioral system, etc. In computational terminology 'system' is used to define different processes, which usually work as coherent integrated processes, such as a database system, code checking system, self-learning system, etc.

Some urban design systems, such as the *system of hierarchy of urban elements*, cognitive systems like the *shape recognition system* and computational systems like the *object-oriented database system*, will be integrated in the *conceptual model* of our tool. In this respect, the proposed tool will also be a system in itself. This system will be called the 'Design Tool', in order to avoid all possible misunderstandings and confusions. All implemented 'sub-tools' and 'systems' will comprise one integrated tool called simply the 'Design Tool'[2]. 'Design Tool' is also a kind of *name* or *title* for the tool that can be used by designers to improve their practice.

We will therefore use several *design* terms in our research and here we will explain the most common ones, such as 'urban design', 'tool design' and 'database design'.

The term 'urban design' will be used in this research to mean the finished process in designing urban environments. 'Urban design' is the urban environment in itself. 'Urban design' is not the process of designing, it is the final result of it. We will use the term 'urban design' as a generic term or

[2] *'Design Tool'* is defined here as a tool which can help in process of design

synonym for the urban environment, or for a physical part of it. 'Urban design' will be also used in the research as 'urban scene', with the meaning of one particular 'urban design'.

The next term that will be used in this research is 'tool design'. 'Tool design' does not bear the same meaning as the terms 'design tool' or even 'Design Tool'. In our research 'tool design' will have double meanings. The first meaning concerns the *process* of developing the proposed Design Tool. Second meaning of 'tool design' will be about the final *product* of developing the tool, called the *conceptual model* of the Design Tool. 'Tool design' will be a kind of overview of the structure of the *conceptual model* of the Design Tool.

The last term related to design that will be used in this research is 'database design'. 'Database design' is a term which will also have double meanings. The first meaning will be about the *process* of developing the structure of a database system for the proposed Design Tool. The second meaning of the 'database design' will be concerned with the final *product* of developing the structure of the database system, also called the *database model* of the Design Tool. The *database model* will be the final 'database design' from which the database system of the Design Tool can be programmed.

Now we will explain the terms 'conceptual model' and 'functional model'.

'Conceptual model' will be used as a term for the high-level description of the logic and structure of the Design Tool. The 'conceptual model' is the final developed structure of the Design Tool. The 'conceptual model ' will also be a 'tool design', which will schematically represent specific modules of the Design Tool. The 'conceptual model' will be a collection composed of the different logical and functional segments of the proposed Design Tool.

'Functional model' will be used as a term for the high-level description of the functional interaction of the object-oriented database system and the 'conceptual model'. The 'functional model' will represent the role that the object-oriented database system plays inside the 'conceptual model' of the 'Design Tool'. The 'functional model' will be based on and will follow the structure of the 'conceptual model'.

We have thus presented and explained the key terms that will be used in this research. There are also a number of other terms that will be used, but we do not need to explain all of them as we assume that their meanings and usage are sufficiently well known.

The computer-based design tool in urban design

Urban design is the discipline located on the boundary between architecture and urban planning, a discipline that deals with architectural objects as well as with building blocks, streets or landscapes. Urban design is the operational arrangement in the city planning process which gives 'physical design direction to urban growth, conservation and change' (Barnett, 1982). Urban design is a part of urban planning activities and arranges relationships properly between objects, infrastructure, the landscape and so on. In urban design, all the technical, economic, social and aesthetic issues involved are highly harmonized in a synthesis of all of them.

According to Batty (Batty et al., 1998), urban design has been almost untouched by computerization in the field. Since the late 1950's, both architecture and urban planning have been highly influenced by software developments, which followed the advances in the field of the electronics industry – not only computers but also satellites. CAD systems changed everyday practice and organization in the architectural office tremendously. Architectural practice has become more efficient and the new possibilities in the domain of 3D realistic and real-time visualization, such as immersive virtual reality (VR), have changed the perspective in the architectural profession. GIS since the 1960's has completely changed the field of urban planning through the incorporation of spatial satellite data which can precisely build a large scale environment, even the whole world, with 2D or 3D components to satisfy the wide interests and needs of professionals.

Software development in the domain of urban design is far behind, and does not even exist in appropriate forms, such as in architecture and urban planning. There are several reasons for this *software vacuum* (Batty et al., 1998): mostly because the urban design process is too diverse and too participatory with a high political influence to be borne by software. One possible reason might also be the difficulties in synthesizing urban design methods with the field of visualization techniques, which has nowadays become possible in links between CAD and GIS in the field of visualization. There are several new research developments in urban design from a GIS perspective for sketch planning, visualization and analysis which incorporate GIS functionality and multimedia visualization (Danahy, 1988; Dekker, 1992; Shiffer, 1992; Ferreira and Wiggins, 1993, Dave and Schmitt, 1994, and others). Most of these researches concern the support of the design process in different parts of it rather than implementing urban design. Some decision support systems have been developed to support planning for specific tasks, such as facility locations, retailing and the like (Harris, 1989; Shiffer, 1992; Densham, 1996; Klosterman, 1997).

The creative aspect of the urban design process cannot be fully implemented in the development of computer applications, but some basic support tool kits can be developed with the presence of CAD, GIS and the Internet.

Urban design problems

There are many possible questions that indicate some of the problems related to urban design. Among the main questions are: how to efficiently represent functional models of the urban environment, and how to test solutions and resolve them; how to collect feedback from an interested group of citizens; how to make analyses of the possible impacts and consequences; how to integrate all of the information in the new functional and acceptable model; how to make decisions based on the democratic vote of experts and non-experts; and how to implement the chosen model. Translating these questions into computer-based technology, they should look like this: is it possible to create a computer-based tool/s to integrate analysis, shaping, re-shaping, testing and deciding in urban design, all based on the knowledge of experts and existing technology, such as CAD, GIS and the Internet, which can be applicable for both professionals and citizens? This single question can be subdivided into hundreds of related questions, but the answers may be only a few. In the state of the art of this research (Chapter 2), a couple of answers to these questions will be presented, by showing some recently developed tools. We should also try to give some answers through the execution of this research, while appreciating that it is only a small contribution in this 'vacuum' field of urban design software.

Urban design manipulates the small-scale and is less abstract than urban planning which deals with large scales. People can have a clear view of how design might be carried out. The potential of urban design is great in involving the support of new technologies on the neighborhood scale, not only for experts but also for ordinary people. A large amount of different data, presented in a visual and acceptable form, for all interested groups, should have a useful feedback impact on urban design. To be more effective and collaborative, designers can make frameworks of the project in which the users of the environment will participate via the Internet. The results of the public participation will be analysed and integrated in the process of designing. This *reflective design* is the approach that can probably solve design problems in generating suitable plans of the urban system (Schön, 1983).

Representation of the urban system

The urban system, at the level of urban design, consists of four types of representation (Batty et al., 1998):

(1) *socio-economic* information is traditionally a spatial, macro kind of information usually pertaining to area,

(2) *functional* information that is addressed to relationships between spatial elements of the system,

(3) *behavioral* information which usually reflects more micro information, such as the way users respond to the imageability of the urban environment, and

(4) *physical* information is geometric (vector) in contrast to geographic (raster) information.

All these types of information are stored in different databases, mostly in GIS's. Probably the best way of integrating of the representation of an urban system and the design process is through the relevancy of the GIS. Urban design involves small-scale location problems, which demands 2D or 3D considerations. Urban design also involves a wide range of issues from socio-economic, to functional, aesthetics, constructional and others. The designer has to have control over and responsibility for all of them. In such circumstances a good supporting design tool, based on recently developed technology, is necessary.

Here the research presented will accept and use all four types of representation, but it will be mostly focused on *behavioral* representation. *Behavioral* representation will be based on cognitive aspects of the urban environment, and it will be one of the tasks of this research.

Cognitive representation in urban design

How people understand and experience the environment around them defines the idea of cognitive representation. Cognitive representation in urban design, here the proposed *conceptual model* of the Design Tool, will be based on the concept of cognitive mapping, way-finding and navigation. The relationship between the physical environment and the psychological pattern of stored information on it is called cognitive mapping. The Design Tool, which can represent a cognitive approach in urban design, will be based on an urban design theory and computer cognition theory, both presented later in the research. The urban design theory, as well as the concept of cognitive mapping, is adopted from Kevin Lynch (1960). Way-finding and navigation will be other aspects of the integration of cognitive representation in the proposed Design Tool.

The cognitive representation of urban design is assumed to be realistic, both from cognitive and implementational viewpoints.

Computational implementation in urban design

Batty (Batty et al., 1998) suggested a framework for developing a computer technique that can be implemented in urban design. The framework illustrates the stages through which new computer technologies can be implemented in the process of urban design. The framework consists of three main parts:

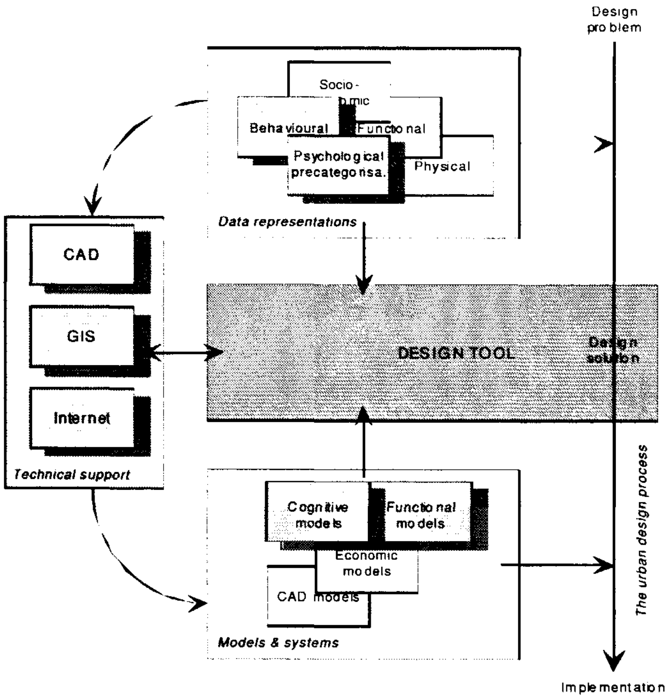
(1) *data representations* show the four types of representations of the urban system (socio-economic, functional, behavioral and physical),

(2) *models and systems* are presented by CAD models, functional models and economic models, and

(3) *the urban design process* presented with different steps in the process from problem definition to the choice of the best plan (implementation).

'Data representation' and 'models and systems' are connected with a 'GIS tool-kit' that is also connected to the *network, photo, CAD, syntax, maps, draw edits* and *2D to 3D models.* This schematical framework should be universal following certain criteria. For the approach of cognitive representation we implemented in the framework the 'Design Tool' [by itself], 'psychological precategorization', 'technical support' and 'cognitive model'. Here Batty's proposed framework is adopted, changed and minimized to satisfy the cognitive approach of our research (see Figure 1.1).

Figure 1-1
The framework of the implementation of computer technology in the urban design process according to a cognitive approach
(based on Batty at al., 1998)



The proposed framework for the implementation of computer technology in the urban design process, according to a cognitive approach, consists of five parts:

(1) *data representation* presents the relevant types of representation in which 'behavioral' and 'psychological precategorization' (a new type) are dominant,
(2) *models and systems* shows the relevant models such as 'functional models' and 'cognitive models' (the present model) in the framework,
(3) *technical support* presents the framework according to computational techniques which could be involved in urban design like 'CAD', 'GIS' and the 'Internet',
(4) *Design Tool* or design tools, are the new software application/s that can improve urban design practice, and
(5) *urban design process* schematically presents the process from the indication of the design problem, via implementation of the Design Tool with the production of design solutions to implementation of the chosen solution.

The position of the Design Tool is central in the framework. All parts of the framework belong to each other directly or indirectly and as such they interact with each other.

Research assumptions

The general research assumption – which will be examined by this research – related to the Design Tool implementation is:

The quality and efficiency of the urban design process can be improved by providing a new Design Tool for analysis and control of the visual quality of the urban environment through the image scene based on the cognitive mapping approach.

The general assumption's claim is that the new Design Tool for urban design will be instrumentally good. The assumption explicitly indicates the general idea of the tool and the field of its implementation, as well as the main approach of the research.

There are another two assumptions related to the Design Tool development, which claim:

(1) *the applicability of the Design Tool is high according to the measure of the 'performance usability' criteria, where*

applicability and performance adequacy scores are determined in reference to the conceptual model, and

(2) *the Design Tool's advantage is that it is better than some available alternatives, measured according to 'performance usability' criteria.*

The claims made here are that the new Design Tool will be applicable and adequate and that it will perform in given circumstances better than some other alternative tools. The testing assumptions will be examined in Chapter 10, where the testing model with the set of 'performance usability' criteria will be developed. The applicability of the tool will also be examined through test simulations of the objects of the cases.

Methods of the research

This research will examine certain levels of the urban design process such as analysis of the urban scene through cognitive mapping, navigation, way-finding and others. An interdisciplinary research method will be proposed to integrate some old developed theories of the urban environment and design methodology and employ recently developed theories in cognitive science, artificial intelligence, information systems and database design. The basic characteristic of this research will be integration of various knowledge for the task of developing the *conceptual model* of the new Design Tool for urban design. The research will propose a cognitive approach that can be integrated in computer processing.

The proposed research methods can help in the creation of the *conceptual model* of the urban Design Tool in the stage of checking and analyzing, defining and generating urban design models presented through the image scene. The proposed methods will go further towards improving specific design problems that will be generated through the urban Design Tool. This research has to develop the guidance for the urban Design Tool. The development of the tool can be seen as a generator of new knowledge (Schön, 1983). In order to do this a *conceptual model* of the Design Tool will be developed, which can adequately represent the phenomena we are interested in. Therefore, this research will work out the *conceptual model* that will be used as a basis for the tool development.

As we see, two main lines of research methods will be pursued. The first line will be an examination of the theoretical background of the research field of interest, and the second line will be providing different computational applied techniques that can support the *conceptual model* of the Design Tool. Once the *conceptual model* of the Design Tool is developed, it has to be tested. For this task we will also develop a new testing method. The testing will be conducted through the simulation of case study objects. The results of the testing will demonstrate the acceptability of the proposed testing model as well as the *conceptual model* of the Design Tool.

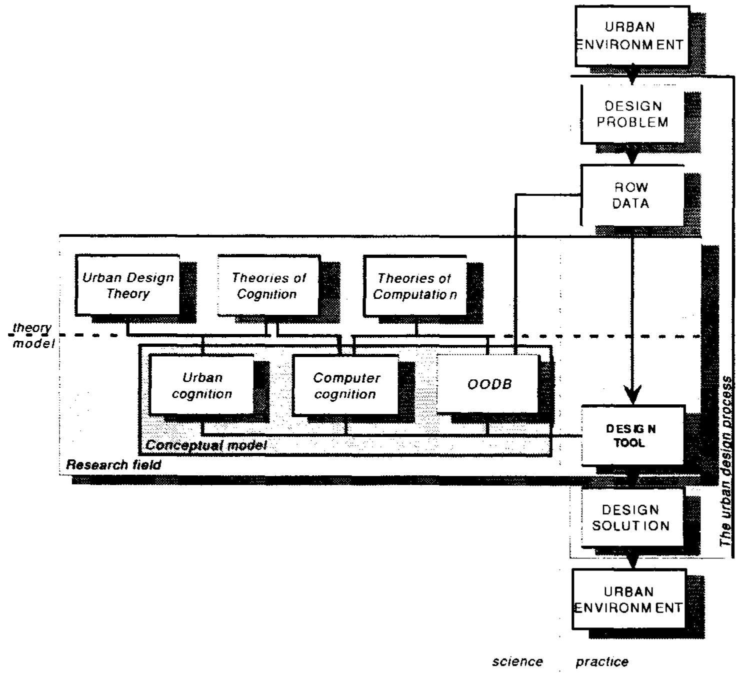
Framework of the research

The framework of the research is presented in Figure 1.2. There are two basic objectives of this research that can establish relevance criteria. The first one is in the field of the research [based on computerized visual urban quality] and the second one is the user process [the urban design process].

The 'research field' consists of two levels: *theoretical* and *model*. The *theoretical* level applies 'urban design theory', 'theories of cognition' and 'theories of computation'. 'Urban design theory' will be based on Kevin Lynch's theory of 'urban form', and we have called it the starting point of the research. 'Theories of cognition' will be based on 'computer cognition' and 'high-level vision'. Both theories will apply a cognitive science approach to the computation. The last element is 'theories of computation', and we will use object-oriented database systems (OODB) because of their functional flexibility and applicability. The *model* level is deduced from the *theoretical* level. The *model* level presents the *conceptual model* of the Design Tool, which consists of three main models: 'urban cognition', computer cognition' and 'OODB'. The *conceptual model* will be the guidance for making a computer application that can be applied in urban design practice as a Design Tool.

The 'urban design process' is only schematically indicated. The 'urban design process' is much more complex than the schema presented, but we do not need to analyze the whole process. We want to indicate two things: the position of our Design Tool inside the urban design process and some other parts of the process, which relate to the tool. First of all, we indicate the 'designing problem' which usually comes from the urban environment and which could be equivalent to the 'problem definition' from Batty's

Figure 1-2
The research framework



framework (Batty at all., 1998). Then the position of the 'row data' is indicated, which could apply 'data collection' and 'data analysis'. The 'row data' is also connected to the OODB model. And the last step in the process that we indicate, after the computational processing is finished through the Design Tool, is 'design solution'. 'Design solution' could correspond to the 'choice of best plan' from Batty's framework (Batty et al., 1998).

The proposed framework will be used consistently during the execution of the research assumptions, the development of the *conceptual model* of the Design Tool and the testing of the model.

Scope of the research

The starting point of the research is Lynch's theory of 'urban form' and his concept of *cognitive mapping*. The whole research is based on them and the general scope of the research is in this domain. The general scope will be extended through the common interests of the research into the domain of cognitive science and computational techniques.

The proposed urban Design Tool will generally operate within an open information system in which the 'input' influences the system. This process will include several phases of operations and the complexity has to be controlled. For this reason it will be necessary to clearly define the operational scope of the research. There are three main strategies in defining the scope of the research in terms of:

(1) the general *assumptions* related to the development of the *conceptual model* of the Design Tool,

(2) the generic tool *requirements*,

(3) the *constraints* of the research.

The *development* of the *conceptual model* of the Design Tool will be what this research sets out to do. Urban design practice needs such a tool to help rectify the existing *software vacuum*, which was briefly explained earlier. The general definition of the development of *conceptual model* of the Design Tool will be based on a *cognitive mapping* approach and the integration of 'urban theory', 'cognitive science' and 'computational techniques'. Each of them will be examined in the course of the research.

Assumptions

There are several basic *assumptions* related to the development of the *conceptual model* of the Design Tool:

(1) The *conceptual model* of the Design Tool can be developed on the main tasks, which are the analysis and control of the visual quality of the urban environment based on a cognitive mapping approach. The urban environment can be represented within an image scene by image, photo, map, CAD model or drawing. The second main task will be way-finding and navigation through the image scene.

(2) The *conceptual model* of the Design Tool can be developed on the needs of urban designers. The professional 'rules' taken from practice can be followed during the development of the *conceptual model* of the Design Tool.

(3) The *conceptual model* of the Design Tool can be developed not only on the needs of urban designers but also on the needs of 'individual design logic'. 'Individual design logic' is logic that cannot be predicted by designing 'rules'.
(4) The *conceptual model* of the Design Tool can provide techniques and methods to analyze images of the urban environment and to extract shape elements and features and to represent and store them in an object-oriented database.
(5) The *conceptual model* of the Design Tool can be developed as an open platform for extending needs to related areas of the urban design field, such as landscape, transportation or water, all following the cognitive mapping approach.
(6) The *conceptual model* of the Design Tool can be developed to be accepted by urban design professionals by testing the tool. The testing of the tool can be the simulation of the objects of the case study through the participation of urban design professionals.
(7) The *conceptual model* of the Design Tool can be re-designed, based on comments of urban design professionals after the tool testing, before making application.

Requirements

The generic tool *requirements* are based mostly on urban design needs and the proposed assumptions of the *conceptual model* of the Design Tool development. Some specific tool requirements related to the three main theoretical frames will be developed through the course of the research.

These are some of the generic Design Tool *requirements*, related to usage and operational criteria:
(1) the Design Tool can generate different kinds of visual data input file formats, from scanned pictures, drawings, photos and the like to CAD models or various GIS maps; input data can be static or dynamic (e.g. digital video),
(2) the Design Tool can check, analyze and store visual input data in the memory through a code library,
(3) the Design Tool can handle visual data efficiently; each element must be coded and then summarized as a figural pattern,
(4) the Design Tool can apply various useful sub-tools as plug-in applications, such as sub-tools for computer image recognition, graphical manipulations and the like.

(5) the Design Tool can prompt easy access to different remote databases such as general statistical data, geo-spatial data etc., all via the Internet,

(6) the Design Tool can process all active data in order or simultaneously depending on the given task,

(7) the Design Tool can implement a method of transferring a raster image [satellite data] into a vector image [geometry of the objects],

(8) the Design Tool can systematically analyze a single object from the image scene as well as the whole visual pattern,

(9) the Design Tool can offer direct access to all the relevant available data of each of the objects from the image scene, by directly selecting the object,

(10) the Design Tool can offer the possibility that each of the changed values of the objects from the image scene will be registered and saved by the database and it will automatically have a new code number,

(11) the Design Tool can allow the user to implement his/her sets of criteria in a direct and easy way,

(12) the Design Tool can allow the user to implement a new design (such as a CAD model, or GIS map) in the tool in a direct and easy way,

(13) the Design Tool can allow the user to interact directly with the tool by changing, implementing and deciding the new design solution,

(14) the Design Tool can help the user in decision-making with the 'computer based knowledge', which is built up in the tool through different criteria and norms,

(15) the Design Tool can save the new chosen solution and automatically give a new code to it,

(16) the Design Tool can retrieve all coding and recognize elements from the memory and storage in a very short time,

(17) the Design Tool will be 'situated' in a complex of information, structured in terms of prototypical conditions for the tasks of future analysis, and therefore the tool will be a 'self-learning' and 'knowledge-based' tool; the tool will be a problem-solving tool by memorizing all needs and knowledge as a *look-back* tool,

(18) the Design Tool can, on the first level, edit visual data, on the second level analyze the visual data and on the third level the tool

is capable of 1) describing, 2) explaining and 3) modifying the visual data that was analysed,

(19) the Design Tool can compose an output image in one of the possible graphical file formats, depending on the user's needs,

(20) the Design Tool can retrieve not only codes and elements but also the 'computer based knowledge' from the storage memory [for example, the decision pattern protocol, or self-learning protocol], and

(21) the Design Tool can work with other computer based tools, such as GIS applications (*ArcInfo*, *ArcView*, etc), CAD programs (like *AutoCAD*), spreadsheet programs, graphical programs or even text processors.

Constraints

The *constraints* of the research and Design Tool are based on the empirical knowledge related to the general *assumptions* and the tool *requirements.*

These are some of the main *constraints* of the research as well as of the Design Tool:

(1) this research is not in the domain of the pure theory of design and pure cognitive science,

(2) this research is not in the domain of improving existing CAD, GIS or Internet applications,

(3) this research is not in the direct domain of artificial intelligence and user interface design,

(4) this research cannot produce a final Design Tool, i.e. an application for the market,

(5) the Design Tool cannot be used for the tasks of pure architecture and urban planning,

(6) the Design Tool cannot be a universal design tool in the domain of urban design,

(7) this research cannot solve all aspects of the conceptual model such as how to calculate the visual quality in terms of Lynch's theory,

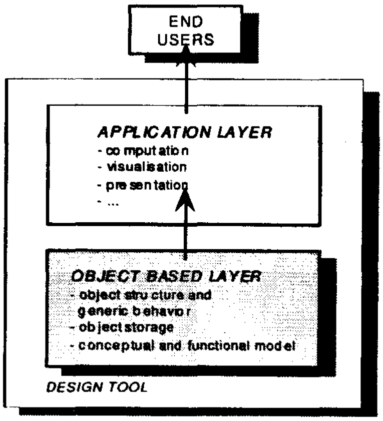
(8) the Design Tool is not directly aimed at non-professional users, such as different social groups, or neighborhood communities, etc.

(9) this research cannot be tested with whole sets of application tests (a whole usability test, interface design tests, etc.).

Tool development

The Design Tool development consists of the two phases presented in Figure 1.3. The first phase is the 'object-based layer', which integrates the *conceptual and functional model, object structure and generic behavior* and *object storage.* All these parts of the 'object-based layer' will be developed in this research and they will represent only instrumental basis on which others will be able to calculate measures of visual quality of the urban environment, after second phase.

Figure 1-3
The Design Tool development phases



The second phase is the 'application layer' and it will integrate *computation, visualization* and *presentation.* The second phase is not the task of this research and it should be developed in the future. The computational problem, one of several, which has to be solve in the second phase are the algorithms which will be able to calculate a quantitative assessment of the visual and functional quality of the urban environment. This phase will complete the Design Tool as an application ready for end users.

The Design Tool development will be based on the examination of the theoretical platform, from which the tool will differ by its applicability. To develop the Design Tool, the *conceptual model* of the tool must first be composed. The *conceptual model* of the Design Tool will describe all aspects of the phenomena of the research interests that are necessary for the tool development. The *conceptual model* that needs to be developed has to correspond to the cognitive mapping approach and to the urban environment.

First we will examine the phenomena of cognitive mapping and we will try to implement this knowledge in the *conceptual model* of the Design Tool. Second, we will examine the theory of 'computer vision' and we will then adopt the method of 'high-level vision' as a plug-in application for the *conceptual model* of the Design Tool. Generally 'high-level vision' is a computer-based system for object and shape recognition. And thirdly, we will evaluate object-oriented database systems and chose one from which we will create a object structure and generic behavior and object storage for the *conceptual model* of the Design Tool.

The *conceptual model* of the Design Tool will indicate for the second phase of the development the visible interfaces that carry out the user's interaction with the tool and the invisible protocols that will govern the interaction between different modules. The general manager, as a part of the database system, will play a crucial role in the communication between these two important and distinct items of the *conceptual model* of the Design Tool.

Once developed, the *conceptual model* of the Design Tool has to be tested by simulation of the objects of the cases. The testing of the *conceptual model* will be performed using the developed testing model and certain criteria of performance usability. The tested *conceptual model* of the Design Tool will show whether it is applicable and acceptable in the context of urban design practice and in which areas it has to be corrected and changed before the final Design Tool can be produced.

The proposed Design Tool will assist and support urban designers in their everyday practice. The Design Tool will be an independent software application that consists of several modules. The Design Tool can use specific protocols to handle communication between modules in order to accomplish the urban design task. Each module will represent the corresponding functionality, all working together and helping in the final decision-making. The decision will not be intended to replace the authority of the designer, rather it will exploit the user's knowledge by providing new structural knowledge based on the various criteria and norms.

The chosen new urban design 'solution' will be composed of existing visual elements with the support of all available related data, and it will be shown as the Design Tool *output*. The acceptable and implementable *output* will be the final product of the Design Tool, which will be realized in the second phase as the integration of *the computational model, visualization model* and *presentational model*.

Outline of the research

This research will be composed of four parts: *general introduction, theoretical background, conceptual tool design* and *evaluation*. During the examination of the research through these four parts we will slowly move from the purely theoretical to applied research. The gap between theory and practice will be bridged as the chapters of this dissertation develop.

The following is a brief outline of the research:

Chapter 1: Introduction

At the beginning of this chapter we presented a definition of the terms which will be used in this research. This chapter introduced the background of the research by explaining recent urban design problems and the position of computerising technology in the field. The chapter also formulated the general assumptions of the research and explained the methodology, framework and scope of the research.

Chapter 2: Review of present urban visual quality approaches and tools

In the beginning of this chapter we will provide an overview of Nasar's 'evaluative approach'. Then we will examine computer applications concerning Lynch and Nasar's approaches. These applications, 'WayMaker' and the 'web-based mapping and survey tool', will be compared with our proposed Design Tool. At the end of this chapter we will examine some approaches related to aesthetics and control of the urban environment.

These first two chapters will provide the general introduction for the research and will lead into the theoretical background, which can be examined in the next three chapters.

Chapter 3: Urban design theory - cognitive maps

This chapter will investigate the theoretical background focused on urban design theory, primarily on Lynch's theory of urban form. Lynch's sign system and some limitations of Lynch's theory will also be introduced here. Then the general concept of cognitive maps will be set out with Lynch's view on it. The next section of this chapter will be reserved for post-Lynchian thinkers on the topic. In the last two sections of this chapter we will generalize cognitive maps and make some remarks on the urban environment.

The main introduction of this chapter will be the concept of cognitive maps, which leads into the main approach of the research.

Chapter 4: Theories of cognition - computer vision

In this chapter we will examine the literature concerning computer vision and object recognition. The theoretical investigation will be focused on the 'representational theory of mind' produced by Marr and the 'theory of high-level vision' introduced by Ullman. The 'theory of high-level vision' and its technique of computer image recognition will be adopted as a plug-in application for the Design Tool.

Chapter 5: Theories of database systems - object-oriented

This chapter will present the wide field of database theories and systems. The research interest will be the study of object-oriented database concepts, models, structure and technologies. In this chapter we will also introduce and adopt Perspective-DB technology, from which an object-oriented database model will be developed.

The conclusions of the last three chapters (Chapters 3, 4 and 5) lead the research to the further development of the conceptual tool design, described in the next four chapters.

Chapter 6: Applying Lynch's theory to the Design Tool

At the beginning of this chapter we will set out a framework for applying Lynch's theory to the Design Tool. Then we will deeply investigate the urban elements derived from Lynch's theory for the Design Tool. Here we will analyze psychological elements that build the visual quality of the urban environment and then the physical elements [objects] of the urban environment. In the last two sections of this chapter we will set out an answer for applying Lynch's theory to the Design Tool and the sets of different tool requirements derived from this research.

Chapter 7: The computer cognition model of the Design Tool

This chapter will introduce the new computer cognition model based on the knowledge of computer modeling in cognitive science. The chapter will also introduce the concept of filtering by 'microunits' and the concept of synthesis by 'global units'. In this chapter new sets of tool requirements related to the cognitive model will be presented.

Chapter 8: The object-oriented database model for the Design Tool

This chapter will show the advantage of applying object-oriented technology and Perspective-DB, the object-oriented database technology that we chose for the Design Tool. The chapter will present the generic design of the database structure and its behavior. The general concept of the communication between data models and the Design Tool via the 'general manager' will also be presented in the chapter.

The above three chapters presented (Chapters 6, 7 and 8) develop some specific parts of the *conceptual model* of the Design Tool and lead the research to the further development of the *conceptual model*, whic h will be explained in the next chapter.

Chapter 9: Tool design - the conceptual model

In this chapter the theoretical framework and system framework will be provided. The *conceptual model* will then be introduced through its structure and structural analys is. At the end of the chapter the *functional model* of the Design Tool will be presented with an analysis of the position and relations of the database system in the Design Tool.

The next two chapters will demonstrate the evaluation of the *conceptual model* of the Design Tool and research conclusions.

Chapter 10: Testing and evaluation of the Design Tool

A new testing model will be developed for the testing of the *conceptual model* of the Design Tool, which will be used for demonstrating performance usability. The testing of the *conceptual model* will be implemented through the simulation of a few of the objects of the cases. A further testing procedure will also be described.

Chapter 11: Conclusions

After testing and evaluation, we will provide general reflections on the research and make final conclusions on the acceptability of the *conceptual model* of the tool. In this last chapter we will also make some suggestions for further research.

CHAPTER 2

REVIEW OF PRESENT URBAN VISUAL QUALITY APPROACHES AND TOOLS

***Task of the chapter**: a review of the theoretical approaches in the field of visual quality of the urban environment and its aesthetics and related computer applications; this is important knowledge which will be helpful in the further development of the proposed Design Tool.*

Introduction

The assumptions and methods of the research given in the last chapter indicate the general direction of the research. Before we start examining the research it is necessary to take a look at what is happening in the field of our research interest.

The goal of this chapter is to review present theories in the field of our research interest with respect to cognitive aspects and the visual quality of the urban environment. The focal point of our research is a theory developed by Kevin Lynch, which will be elaborated in Chapters 3 and 6.

Here we will present a short review of the main concept of Nasar's 'evaluative approach of the city'. This will be followed by two recently developed computer applications related to the approaches of Lynch and Nasar. The first application is called 'WayMaker', and the second one is 'web based mapping and survey tools'. Both applications are interesting in the way they use, interpret and implement the theories in computer-based tools. Later, we will compare these two applications with the proposed Design Tool.

There are a couple of other researches (applications) which also experimented with Lynch's theory. One of the researches (Darken and Sibert, 1996) focused on determining whether people use physical-world wayfinding strategies in a large virtual environment. They used Lynch's (1960) and Passini's (1984) studies to base the role of environmental designers' thoughts on wayfinding tasks. Some other researches (Ingram and Benford, 1995) used Lynch's theory in visualizing large information systems. In these researches, the role of Lynch's theory is marginal and only inspirational for the authors, and we do not need to examine them here.

At the end of this chapter we will review some other research in the domain of 'urban aesthetics' and 'controlling the urban environment'; both fields are in the wider interest of our research.

In the following chapter the second par t of this research will start, where we will examine the theoretical background. First, in the following chapter, we will discuss urban design theory in general terms, with a focus on cognitive mapping. Then Lynch's theory will be explained and post-Lynchian approaches will be reviewed. These are all important sources for this research and for the Design Tool development.

Nasar's 'evaluative approach'

This work of Kevin Lynch has become influential for many researchers and their theories in subsequent years. One of these, which builds directly upon Lynch's influential idea, is the work of Jack Nasar presented in the book "The evaluative image of the city" (1998). Nasar extends Lynch's work in the domain of 'cognitive maps', and here we will present the main idea of his 'evaluative approach'.

Nasar argues that it is not appropriate to place city shape in the realm of art. People can choose whether or not to experience literature, art or music, but such a choice is not afforded for city design. In their daily activities people must pass through and experience different public areas of the city. Urban designers and planners can derive valuable information for their job by studying what kind of evaluative image people have of their own city form. Nasar also argues that public preferences of urban form are often quite different from what professionals believe. Nasar concludes that the values of urban planners and designers are not representative and people cannot build a knowledge base for guiding community appearance upon their ideas and preferences alone. Instead, professionals should develop 'evaluative maps' of cities that are based on actual surveys of the public's preferences. These maps should act as guidelines for decisions about city design.

In contrast to Lynch, Nasar argues that a knowledge of identity and structure (imageability) is not enough. His work focuses on Lynch's third category of 'meaning', which refers to inferences about the quality and character of the city and its users. Human feelings and meanings define Nasar's 'evaluative images of the city'. Evaluative images of the city refer to how people react to their environment and how they evaluate it. A city whose imageable elements are liked by the majority of people will have a positive evaluative image. If, on the other hand, people dislike these elements, this suggests a need for changes in the city's appearance.

Beauty in the urban environment is less qualitative and subjective than many people think. Meanings may vary with sociocultural conditions and individuals may have idiosyncratic preferences. But citizens generally agree that certain components make for desirable urban form. Nasar's research presents strong consistencies in what people like and dislike in an urban environment. Studies of the evaluative images and the meanings that people attach to an urban environment can provide useful information for designing a desirable urban environment. Nasar suggests that it is possible to learn the public's preferences by empirically measuring them. It is possible to measure preferences to determine the degree to which people like or dislike various areas of a city. Nasar's method focuses on one aspect of the evaluative image and visual quality: the 'likability' of the cityscape. "*Likability* refers to the probability that an environment will evoke a strong and favorable evaluative response among the groups or the public experiencing it." (p. 3). He agrees that many factors other than the evaluative image make for a successful city, but the evaluative image or likability has a fundamental value.

The evaluative image arises from a person's mental activities related to ongoing interaction within the urban environment. A person observes many attributes of the environment and evaluates what he or she sees. It may involve feelings related directly to the structure and content meaning of the urban form. Nasar requires this mental activity to: (1) recognize the content, (2) draw inferences about it and place it into a mental framework, and (3) evaluate it. To draw a universal principle out of the uniqueness of human experiences, Nasar indicates several visual features as relating to social meanings and preferences. Findings on evaluative images of cities reflect five features: naturalness, upkeep, openness, order and historical significance. In contrast to cognitive (mental) maps, which are incomplete and disordered with simplified information, evaluative maps suggest associations with city structure and experience. They indicate likability associated with five features. The map shows the identity, location and likability of visual features and explains the basis for the evaluation.

According to Nasar, an evaluative image contains two important variables: visual aspects of the urban environment and human evaluative responses. Visual features are independent variables and human responses are dependent variables. People also have a degree of 'aesthetic' response in their evaluative response. People also experience connotative meaning - feelings about places and activities. The cognitive process represents important mediating variables in people's evaluative responses. Nasar's model suggests two broad components of evaluative response: perceptual and cognitive, and two kinds of environmental variables: formal and symbolic.

To sum up, if ordinary people evaluate an urban environment as favorable, this means that evaluative images have notable features for urban designers and planners. Nasar's approach indicates that professionals can benefit from attending to and using the evaluative images of people.

Computer applications concerning the approaches of Lynch and Nasar

Theoretical followers of Kevin Lynch are numerous. One of them, as we have already explained, is Nasar. It is a pity that these useful theories have almost no effective computational support in urban designers' practice. Here we will present two pioneering attempts in the domain of Lynch and Nasar's theoretical contributions. These works will also be interesting for this research and will be compared with the proposed Design Tool.

"WayMaker"

"WayMaker" is an application produced as a result of the ongoing research (Strohecker and Barros, 1998; Strohecker, 1999) of MERL[1]. "WayMaker" is a tool for non-professionals to create digital layouts for large-scale graphical virtual environments. The tool is based on Lynch's (1960) elements of city images. The tool is supports Lynch's value of participatory design, while enabling an extension of his efforts to understand how people 'image' and 'think' about the city. From a verbal and pictorial account, Lynch derived five basic elements of the city images: districts, paths, edges, nodes and landmarks. "WayMaker" uses Lynch's elements to address a current problem in virtual environment design, which is becoming commonplace.

"WayMaker" provides a basic set of Lynch's elements as symbols that correspond to aspects of city form. There are five basic symbols representing five of Lynch's basic elements: districts, paths, nodes, edges and landmarks. Using the symbols, users create maps reflecting their conceptions of space. Users create their imaginary or real maps, which they can modify with specific details. "WayMaker"'s maps and scenes depict only the large-scale structural features of a space. The system transforms the user's map design into a street-level scene representing a walk through the domain. The initial prototype works with street-scenes based on Paul Cezanne's reproductions. Paintings comprise studies of the relationship of the built to the natural environment. The prototype is only suggestive, not a rendering of the real world. The task is to help users understand the connection between 'cognitive maps' and the virtual space that they represent.

Working with "WayMaker" is done by simply dragging symbols into the working area. Symbols of urban elements can be stretched and curved to take on various shapes. All symbols can be placed anywhere in the working area, and some of the symbols are in two sizes (large and small). The user can create maps with certain details to indicate a sensual quality about the activity associated with nodes. The map cannot represent detailed characteristics of the environment in the way that, for example, GIS maps

[1] MERL - Mitsubishi Electronic Research Laboratory, Cambridge, MA.

can. The sensory details in the walkthrough depend on the image and sound database that the system accesses in composing scenes in a virtual world. In the prototype, the image database consists of Cezanne's paintings. These paintings are interpreted according to a classic perspective in which side planes are defined by diagonals that converge on a vanishing point along the horizon. Each painting is comprised of side planes, ground, back and sky. The paintings belong to different districts on the virtual map composition and they vary within a single district. Such variations ensure richness in the depiction of the virtual world. By specifying details of the layout, the user can indicate the walkway route through the virtual world. Then the system can loop through a series of functions: select framework that corresponds to district – varies the framework for the next scene, and so on. "WayMaker" is premised on an explicit version of Lynch's assumption: way-finding is translated as a set of creative actions. The result of this creation is a series of scenes that vary subtly, frame by frame, and distinctly, district by district. The view changes from map to street level. The transformation is simultaneously visible through displays of the map and the walkthrough scenes. The display indicates each scene's location within the map diagram, as the walkway sequence plays out.

Any number of prepared multimedia (visual and sound) databases could support "WayMaker" maps. "WayMaker" could easily use an alternative 2D database for the walkthrough scene, from real images to very abstract ones. The current depiction of scenes in "WayMaker"'s virtual space is independent of the question of whether or how Lynch's formulations may be broadly useful as a basis for design and construction tools for the virtual world.

The users who design cognitive maps can understand the transformation from abstract symbols to realistic imagery. They can create their own maps to reflect memories of some particular places or to create some abstract imaginary world. The activity is conducive to thinking and learning about spaces and spatial relationships. The user can change the combination of elements, and even a map and walkthrough gives some sense of the "dynamic nature of perception" and underscores the principle that designing and building are ongoing processes (Lynch, 1984, p. 157).

As we mentioned, "WayMaker" is an ongoing project and the authors planned to place the system within the social context of different neighborhood groups. "WayMaker" might be used in social, participatory contexts in the task of developing an understanding of spatial relations within an urban environment.

"Web-based mapping and survey tools"

Al-Kodmany (2000) and urban planners and designers from the University of Illinois at Chicago (UIC) were invited to be part of a participatory community-planning process in the Pilsen community (Chicago). The team developed a planning method based on the work of Jack Nasar (1998), by using new interactive GIS technology on the World Wide Web (WWW). A GIS system for visualizing the Pilsen neighborhood was developed, along with a system for enabling real-time interaction with residents over the Internet. This case study utilizes a Web-based survey as a tool to discover public preferences that can be used as a guide in shaping and reshaping neighborhood design.

The initial idea for designing the tool came from Nasar's conclusion that the values of professional designers are not representative, and thus we cannot build a knowledge base for guiding community appearance upon their ideas and preferences alone. Nasar insists that urban planners should develop evaluative maps of cities that are based on an actual survey of the public's preferences. These maps can then function as guidelines for decisions about city design. Nasar developed the methods for 'evaluative maps' based on traditional survey methods and manual mapping techniques. Al-Kodmany's approach is to use Nasar's knowledge represented by state-of-the-art computer technology. The goal was to combine real-time interaction over the Internet with GIS visualization capabilities.

The Internet will become the dominant mode of communication in the coming years. In 1999, it was estimated that almost 50 percent of all U.S. and Canadian households had access to the Internet, and this number is still rapidly growing. The Internet is also the only mass medium that is interactive and can facilitate communication between planners and the people. In addition, many GIS systems have recently appeared on the Web. Peng (1999) argues that the integration of GIS wit h the Internet is inevitable and will result in making GIS technology and spatial information easily accessible to the public. Sarjakoski (1998) suggests that the most difficult issues with community planing on the Internet are institutional rather than technical. Presently there are only small numbers of actual Web-based surveys based on forms and text. And there are no Web-based surveys that consist of visualization incorporated into a public survey.

Al-Kodmany's team decided to create a Web-based survey to show how people would respond to 'urban likability and dislikability' (ULD) for the Pilsen community. Participants simply logged on to the Website where they could view a map of Pilsen. They were asked to identify and point out areas of their community on the map that they most liked and disliked by clicking on the appropriate square of the grid, and to provide reasons for their responses. Green pointers indicated 'liked' areas and red pointers indicated 'disliked' areas. The community map was replaced with a high-resolution aerial photograph of the area and the green and red colors were opaque on the grid square. The aerial photo also provides zoom-in capabilities to assist the user in selecting community areas. There are also several features that assist navigation through the aerial photo, such as still photos, movies and panoramic views from the main nodes. All images corresponding to the areas under the highlighted arrow are loaded up and displayed in another window. This allows the user to see the visual appearance of the area while at the same time seeing its location on the map. These 'point and click capabilities' provide a dynamic way of experiencing the spatial configuration of the street.

Using a Web-based survey made mapping the result easy. The team decided to use an Oracle database linked to a GIS application. This connection provides a capability for automatically plotting all available information from users to the map. The database system can also group the associated comments of users. The survey GIS map was then created from this data. Here we see an improvement of Nasar's original method of compiling individual maps manually. The tool can immediately create composite maps that show the public's evaluation image of Pilsen, with each Internet access log. GIS maps are interactive: by clicking on a map's grid, a new window will appear with text that lists the user's stated reasons for liking or disliking this particular area. The goal of the system was that the community members would be able to simultaneously view maps, images and text that describe the evaluative image of the community.

The big advantage of the Web-based method is that it saves both time and money, because the costs of publishing a survey on the Web are low compared with the costs associated with traditional methods; there is less paper, postage and interview time. A web survey eliminates data entry since the respondent's data is automatically stored in a Web access log and then transferred into a GIS program. Another advantage is clear privacy: no names, no time pressure - users take as much time as they like, and respond openly to the survey at any hour, day or night, with no influence by an interviewer. In terms of Nasar's original method, using GIS for the likability/dislikability survey enables the automatic processing of respondents' data and the immediate creation of composite maps. GIS also displays related data about the social and physical conditions in the selected area. The Web provides participants with contextual information and visualization that was not possible in Nasar's method. It is provided in still and motion pictures that help participants offer a more precise assessment of their community.

The main disadvantage is that not everyone at present has access to the Internet and many people lack technical knowledge. Another significant disadvantage is simple bias; participants have a particular interest in the topic. And the final disadvantage is that the survey must rely upon participants' honesty. It is difficult to ensure, by using one password per person, that the person did not fill in the survey multiple times to stress a certain view.

This pioneering project uses Web-based technology to support participatory planning and design. This tool has the potential to connect planners and researches that are separated geographically, even from different countries. Also, the survey could be adapted to deal with other environmental aspects to be used, for example, in agriculture, transportation and so forth. The most significant impact of the tool is in the field of transforming public participation efforts. Donald Schön (1996) has suggested several initiatives for the beneficial use of new communications technology on the part of low-income communities. In the Pilsen case study, the designer's team provides access to the survey Website through computers located in the offices of local organizations and the main library. Computers are available for low-income people at several locations in the Pilsen community to enable them to participate actively in community development. The designer's team is still engaged in adding new functional features to their design tool.

Comparison with the proposed Design Tool

The two computer applications presented here have some similarity to the proposed Design Tool, but first we must present the basic concept and field of interest of the Design Tool.

Like the two applications explained above, the Design Tool also has its starting point in Lynch's theory. Lynch's theory of 'urban form' and its hierarchical structure of main urban elements will be applied, together with his concept of 'cognitive mapping', to a conceptual model of the Design

Tool. The conceptual model of the Design Tool will be supported by an Object-Oriented Database system (OODB) and a GIS database, which will constitute a functional model of the tool. The Design Tool will be aimed at urban planners and designers with the task of analyzing an existing or planned urban environment. The Design Tool will be supportive for analyses and tests of middle-scale urban environments in very early, or very late phases of the design process. The Design Tool can test the visual quality of the urban environment by using multimedia data as input, as well as output, of the system. The Design Tool will be a robust, stand-alone application offering the possibility of accessing different remote databases via the Internet.

This brief outline of the Design Tool indicates the major differences from the applications explained previously. "WayMaker" and the "Web-based mapping and survey tool" are different in their approaches as well as in technological support.

The proposed Design Tool will not be able to create a 'layout for large-scale graphical virtual environment' like "WayMaker", but both use the same sources from Lynch's theory, his five elements, i.e. district, node, edge, path and landmark. The Design Tool will deal with concrete images of an urban environment, or maps, and will analyze them from different aspects depending on the designer's task. "WayMaker" deals with some abstract graphical representation of an imaginable environment by simulating it through precise rules, and it seems to be more of an experiment than a useful application for professionals.

The proposed Design Tool will not be a surveying tool at all, like Al-Kodmany's very useful tool. Also, the proposed Design Tool will not be created to be used directly through the World Wide Web and to support a designer's interaction with community groups. The Design Tool will support the process of analyzing, testing and decision-making in the overall process of shaping or reshaping an urban environment.

There are some similarities between the two applications presented here and the proposed Design Tool. The first and main similarity is that they use the same theoretical resources: Lynch and Nasar's theories. The second similarity is the same field of interests of the tools: an urban environment and its visualization. The final possible similarity could be their use of a methodology that results in 'research applications', which could be almost directly useful in practice. All three of the presented design tools are still termed 'ongoing projects'.

Aesthetics of urban design

The aesthetics of urban design has become a wide field of research interest in recent decades. A lot of interesting work has been produced and here we will briefly indicate some of it that is most interesting for our research topic.

An aesthetic response results from an ongoing interaction between active humans and their environment. Cognition also affects emotions (Lazarus, 1984; Zajonc, 1984) and the term 'aesthetic response' arises from this. The pursuit of enjoyable surroundings does not imply uniform design criteria to make all buildings and places pleasant. Evaluative response has been found to consist of three components: pleasantness, excitement and calmness (Ward & Rusell, 1981; Rusell, 1988; Nasar, 1988).

Formal aesthetics is the study of the structure of forms, which include the following attributes: shape, proportion, rhythm, scale, complexity, color, illumination, shadowing, order, hierarchy, spatial relations, incongruity, ambiguity, surprise and novelty (Wohlwill, 1976; Lang, 1988; Groat & Despers, 1990).

Symbolic aesthetics is the study of human responses to the content of form (Lang, 1988). Humans experience a building or environment through mediating content variables, which are not defined solely by physical attributes. This content variable reflects the individual's internal associated representation of the building or environment by *denotative* meaning (e.g., building classification by type or by style) or by *connotative* meaning (e.g., one building looking friendlier than another). Content variables can have both denotative and connotative meanings.

Formal and symbolic variables interact. The connotative meaning of and aesthetic response to a style depend on the viewer seeing a style or 'characteristic formal organization', and on the match of the style to the viewer's expectations in relation to the building type (Norberg-Schultz, 1965).

Several *formal* variables emerge as prominent in a human's experience with their physical surroundings:

(1) enclosure (openness, spaciousness, density, mystery),

(2) complexity (diversity, visual richness, ornamentation, information rate), and

(3) order (unity, order, clarity).

People prefer open space to wide-open space or highly enclosed spaces (Ulrich, 1983; Kaplan & Kaplan, 1989). Complexity, diversity (Wohlwill,

1976) or visual richness involve a comparison in which more independent elements produce greater complexity. Because of the importance of wayfinding in a real environment, individuals prefer legibility or coherence to, for example, mystery. Most studies confirm a preference for organizing variables such as legibility, identifibility and coherence (Kaplan & Kaplan, 1989). For interest or excitement, a design review might try to encourage high diversity and low coherence.

Through the content of *symbolic* categories, in interacting with the environment and developing knowledge structures, individuals from different places, cultures and subcultures will develop different meanings and preferences. Several content variables have emerged as salient in humans' experience of their environment:

(1) naturalness,
(2) upkeep,
(3) intensity of use, and
(4) style.

Many studies have found artificial content (such as poles, wires, signs, vehicles, dilapidation and intense uses such as industry) to depress preference (Cooper, 1972; Appleyard, 1981; Nasar, 1983, 1987, 1990).

Movement through the environment can also introduce surprise. For example, for an observer moving by car, judgments of complexity would be affected by large-scale elements (such as full buildings). For another observer moving at walking speed, the relevant information may come from smaller-scale details, such as an ornament or street board. All studies have related responses to static stimuli and none related to moving stimuli.

Control of urban design

Design control is conceived to be primarily concerned with aesthetics (external appearance), and hence the term *'aesthetic control'* is favored. Aesthetic control impedes the quality of building or urban environment designs. Quality design can only be achieved by the use of good and experienced designers, able to exercise choice within given, reasonable, limited and measurable restraints. Quality design consists of more than just buildings; it is the underlying pattern of buildings rather than buildings themselves that stands the test of time. Quality design largely fails to move beyond the physical and perceptual aspects of urban design, although it does explicitly recognize its temporal, spatial, morphological and contextual dimensions (Carmona, 1996).

Urban design is a complex phenomenon which defies simple definitions or explanation, but which encompasses *"the design, creation and management of 'good' urban spaces and places"* as both an approach and response to the process of urban change and development (Rowley, 1994, p. 195). It is an ongoing evolutionary process of urban change concerned with satisfying social and emotional needs, as well as with other requirements of a convenient, safe, healthy and efficient public realm. Four key principles of permeability, variety, legibility and robustness are also important for the social perception of design. Visual appropriateness, richness and personalization encompass the more visual aspects of urban form.

Urban design operates at many different spatial scales and over different time frames. Urban design needs to be seen as part of an ongoing process, rather than as a one-stop answer to a specific problem. There are four stages in this process: analysis, synthesis, appraisal and decision-making (Moughtin, 1992). These stages are neither independent nor necessarily sequential, but are instead part of an integrated, cyclical and iterative process in which the nature of urban design as activity is reflected.

Some of the *spatial* elements for controlling urban design are: open space, road hierarchy, settlement pattern, town cramming, capital web, compact form, districts, neighborhoods, public transport and topography.

Some of the *visual* elements for controlling urban design are: bulk, appearance, development size, local style, massing, amenity, scale, detailed design, materials, style, balance, color, corners, focal points, form, harmony, landmarks, proportion, rhythm, richness, roofscape, skyline, solid vs. void, texture, townscape and vertical vs. horizontal.

Some of the *perceptual* elements for controlling urban design are defensibility, distinctiveness, enclosure, place, variety, appropriateness, getaways, human scale, identity, image, legibility and sensual experience.

All developed categories and elements guiding the control of urban design are based on an understanding of the place's distinctive characteristics.

Conclusions

In this chapter we reviewed some of the theoretical approaches in the field of visual quality of the urban environment and its aesthetics and related computer applications.

The main theory that is to be adopted in this research was developed in the early sixties – Kevin Lynch's theory of 'urban form' and his concept of 'cognitive mapping'. Later, some other researchers developed Lynch's ideas further in different fields of scientific interest, such as Nasar's 'evaluative image of the city', which was explained above. As we showed earlier in the chapter, Lynch's *five elements* of the urban environment, which build the imageability of the city, played an important role in creating the "WayMaker" computer application. "WayMaker", and the second application reviewed, the "web-based mapping and survey tool", gave important information about what is going on in computational design tools in the field of our research interest. Both 'research applications' are also important as extended applied knowledge in urban cognition.

This chapter also presented studies in the domain of aesthetics and controlling urban design, which are additional interests of the research topic. The most important things missing in the review of literature and 'research applications' are computer applications that would be able to analyze the visual quality of the urban environment by cognitive mapping precedents. Nothing was found relating to this particular subject. The wide literature investigation showed that the main topic of this research is in many aspects new in its approach.

In the next chapter the second part of the research will start, in which we will discuss the theoretical background concerning the research topics. The following chapter will present the fundamentals of Lynch's theory and the different theoretical approaches of Lynch's followers. We will introduce the concept of cognitive maps and some other knowledge that will be helpful in the further development of the proposed Design Tool.

Part two - Theoretical background

CHAPTER 3

URBAN DESIGN THEORY - COGNITIVE MAPS

***Task of the chapter**: Lynch's theory and others related to it, the concept of 'cognitive maps', and a personal human knowledge of a familiar urban environment, constitute the main theoretical platform for the further development of the proposed Design Tool.*

Introduction

In the previous chapter we reviewed theoretical approaches in the field of the visual quality of the urban environment and its aesthetics and related computer applications. The importance of Kevin Lynch's theory of 'urban form' for this research was indicated.

The field of spatial cognition has grown steadily since the sixties. Architects, urban planners, geographers, anthropologists and psychologists have attempted, from their different points of view, to characterize those attributes of spatial environments that constitute for them a great source of experience. These are highly multidisciplinary experiences dealing with the interaction between humans and their environment at many levels. Lynch introduced the importance of human vision and our analysis has the task of elucidating this importance.

The task of this chapter is to give an overview of Kevin Lynch's theory of 'urban form', as set out in the book 'Image of the City' (1960), which constitutes a central component of design theory in this research. This book is crucial among all the literature about the *look of cities*. It is an important study, following on from which several new theories have been developed. More important for our research however, is the theory of cognitive perception of the urban environment and more specifically - *cognitive mapping*; in the domain between space representation and cognitive psychology.

In this chapter we will explain concepts concerning cognitive maps, and we will provide some comments on them. The knowledge presented in this chapter will be the first theoretical platform for designing the *conceptual model* of the Design Tool.

The theoretical background of the research will be examined in this and the next two chapters. The next chapter will examine the second theoretical background of this research concerning theories of cognition. We will also review Marr's 'computer vision' theory and Ullman's theory of 'high-level vision'.

Lynch's theory of *urban form*

Kevin Lynch (1918 - 1984) was the first author who focused his work on visual elements and cognitive concepts of the urban environment. In his books – we will mentioned here only a few well-known works suc h as *The Image of the City* (1960), *The View from the Road* (together with Appleyard and Mayer, 1964), *What Time is the Place* (1972), *Managing the Sense of Region* (1976) and *Good City Form* (1981) –a rich, innovative way of conceiving of the urban environment was presented with a deep design knowledge that changed the attitudes of both professionals and scholars. His books integrated a theory of *urban form* that consists of physical and psychological elements, which was a new approach distinct from the urban theories of the time.

In our research we will focus on Lynch's book 'Image of the City' (1960) on account of the theory of *urban form* which was introduced in it. An urban environment is a complex system of interactions between people [users] and various surrounding objects. Lynch describes the user as a *citizen* who "has had long associations with some part of his city, and his image is soaked in memories and meanings."(ibid. p.1). He then explains users as "moving elements in a city, and in particular the people and their activities, are as important as the stationary physical parts." (ibid. p.1). Objects, the physical elements of the environment, represent the perceptual form of the city in this interaction with users. Lynch described two things important for a subsequent explanation of the whole theory: first, physical elements of the city and second, the psychological, mental image of the city.

Lynch distinguished physical elements into natural and man-made elements. Natural elements, such as air, sky, rivers, lakes, ponds and hills are all elements that exist in nature that man uses and interpolates in his 'built elements'. Built elements are: infrastructure, objects, vehicles, airplanes, and so on, i.e. all objects that man makes by his intention and that physically exist in the perceived environment, as static or dynamic objects. All together, natural and man made elements are characterized by common characteristics, such as color, smell, noise, warmth, and so on, which build a perceptual form of the urban environment.

The visual quality of the urban environment, in Lynch's theory, relates to the physical elements of the environment and the mental image of its users. Users perceive an urban environment in its fragmentation into elements and patterns. All perceptions are different and special, and are related to users' knowledge, experience or familiarity with an urban site. Almost every sense is in action all of the time. The visual qualities of some elements and features are used as generalities in the process of navigating in the urban environment. Lynch considers the visual quality of the city "by studying the mental image of that city which is held by its citizens."(ibid. p. 2). The visual quality of the city is concentrated in four elements in Lynch's theory: *legibility, building the image* [*image*], *structure and identity* [*identity*], and *imageability*.

Legibility is defined as elements whose parts can be recognized and organized in a coherent pattern or symbols. "A legible city would be one whose districts or landmarks or pathways are easily identifiable and are easily grouped into an over-all pattern." (ibid. p. 2). Legiability is very important in city settings and especially in the task of rebuilding a city.

Image or *building the image* is defined thus: "environmental images are the result of a two-way process between the observer and his environment." (ibid. p.6). The image of a given urban environment may vary between different observers [users] and it is an individual mental image. "The common mental pictures carried by large number of city's inhabitants" Lynch called public images. (ibid. p. 7).

Identity or *structure and identity* were defined by Lynch as an environmental image that can be "analysed into three components: identity, structure and meaning." (ibid. p. 8). These components in reality always appear together.

Imageability is defined as the "quality in a physical object which gives it a high probability of evoking a strong image in any given observer" (ibid. p. 9). Imageability "does not necessarily connote something fixed, limited, precise, unified or regularly ordered, although it may sometimes have these qualities." (ibid. p. 10).

Lynch also analyzed the effects of physical, perceptible objects, and from this the *five elements* of the urban environment were derived. Lynch –as in the case in our research – does not explain all other influences of an urban environment on imageability, such as social meaning, functionality, tradition, names, and so on. The five elements derived from the analysis of urban objects in Lynch's theory are: *paths, edges, districts, nodes* and *landmarks*.

Paths "Are the channels along which the observer customarily, occasionally, or potentially moves. They may be streets, walkways, transit lines, canals, railroads." (ibid. p. 47).

Edges "Edges are the linear elements not used or considered as paths by the observer. They are the boundaries between two paths, linear breaks in continuity: shores, railroad cuts, edges of development, and walls." (ibid. p. 47).

"Such edges may be barriers, more or less penetrable, which close one region off from another; or they may be seams, lines along which two regions are related and joined together." (ibid. p. 47).

Districts "Districts are the medium-to-large sections of the city, conceived of as having two-dimensional extent, which the observer mentally enters 'inside of', and which are recognizable as having some common, identifying characters. Always identifiable from the inside, they are also used for exterior reference if visible from the outside." (ibid. p. 47).

Nodes "Nodes are points, the strategic spots in a city into which an observer can enter, and which are the intensive foci to and from which he is travelling. They may be primarily junctions, places of a break in transportation, a crossing or convergence of paths, moments of shift from one structure to another. Or the nodes may be simply concentrations, which gain their importance from being the condensation of some use of physical character, such as a street-corner hangout or an enclosed square." (ibid. p. 47).

Landmarks "Landmarks are another type of point-reference, but in this case the observer does not enter within them, they are external. They are usually a rather simply defined physical object: building, sign, store, or mountain." (ibid. p. 48). "Some landmarks are distant ones, typically seen from many angles and distances, over the tops of smaller elements, and used as radial references." (ibid. p. 48).

The main elements of Lynch's theory, the four elements of the visual quality of the city and the five elements of an urban environment presented here will be examined in more detail in Chapter 6. These elements are the main structural parts which will be integrated in the database system of the Design Tool.

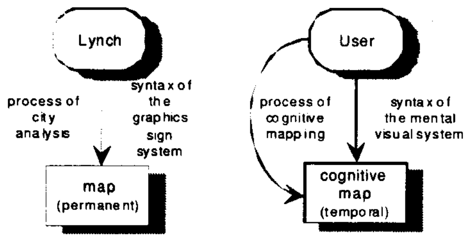
The graphic sign system in Lynch's theory

After describing the basic directions of his theory, Lynch carried out an analysis of three American cities [Boston, Jersey City and Los Angeles]. In this analysis he transcribed the urban information into elements of a unique graphic sign system. Figure 3.1 presents Lynch's process of describing syntax and relations in the visual communication process in his theory. Figure 3.1 also schematically presents the syntax process of the mental visual system in the process of *cognitive mapping* by a user of the urban environment.

Figure 3.1 also indicates two categories of maps: *permanent* and *temporal*.

A *permanent map*, in the sense of Lynch's theory, is a map created by graphic representation for different, mostly permanent tasks. This map is traditionally presented in a two-dimensional medium (usually on paper), such as a topographical or city map. This map can also be a graphical representation of the process of cognitive mapping.

Figure 3.1
Syntax in the visual communication process in Lynch's theory


Lynch
process of city analysis
syntax of the graphics sign system
map (permanent)
User
process of cognitive mapping
syntax of the mental visual system
cognitive map (temporal)

A *temporal map* is the mental interpretation of images [objects] of a particular space by a process of cognitive mapping. This *map* is always temporal, memorized as a mental vision of a particular space in the human brain. This map is not a traditional map but it can be graphically interpreted in a traditional way. We will explain more about cognitive maps and cognitive mapping later in this chapter.

The basic concept of the graphic sign system is that the plane, the two-dimensional medium (originally on paper) on which maps and other graphics are presented is considered to be continuous, homogenous and consisting of two dimensions. The implementation or representation of basic graphic elements (points, lines, areas or different signs) and their combination determines the graphic image on the map. Each map consists of more or less conventional graphical elements. In cartography, for example, it is the geographical component that uses the two dimensions of the plane. This component is continuous and uses the x and y axes of the plane to express its geographical organization of space.

In his analysis of three American cities, Lynch presented his new graphic sign system [notation] for the first time. This system was correlated with the five elements of the urban environment, as described above, and became the main graphical *notation* in Lynch's theory. This *notation* is categorized in two groups of elements, based on their degree of importance in the urban environment, as *major* and *minor* elements. Table 3.1 gives a schematical categorization of the main graphical *notation* that Lynch used in the process of urban analysis. The first product of the 'five elements' notations' is the study of Boston, presented in the book "Image of the City" on page 19. This map very clearly presents Lynch's graphical notations of elements of a particular urban environment.

Table 3.1
Lynch's graphic sign system *(notation)*

URBAN ENVIRONMENT	Major element	Minor element
Paths	━━━━	━ ━
Edges	▥	
Districts	●	○
Nodes	■	□
Landmarks	✡	▲

In his book, Lynch's use of graphical notation is not consistent with this main usage; there are no strict categorizations or reasons for this. This type of graphical notation is more intuitive and resulted from Lynch's momentary creativity and was related to specific analyses of space.

How did Lynch create his maps during his analyses of the three American cities? He used an existing map of the selected city, and over the top of it, as a new layer, he created a new map arising from his analysis of this city. He adopted the geographical orientation and notation of scale from the original map. All other map elements were created, according to his new notation system, by tracing the shapes and characteristics of the city from the original, underlying map.

Some limitations of Lynch's theory

We do not need to present a general criticism of Lynch's theory – which in any case was done by colleagues and professionals after the study was published – because this would be senseless for the function of our research. However, through the analysis of Lynch's theory in this chapter, we have found some aspects that are problematic. We will now focus on these points and provide some criticism. This will permit us to extract from Lynch's theory those aspects that will be helpful for developing our Design Tool.

Here we will illustrate two important critical points that can be related to the main tasks of our research.

The first critical point concerning Lynch's theory is the fact that it is based on analyses of three American cities: Boston, Jersey City and Los Angeles. By analyzing these three cities, Lynch derived some general conclusions.

The fact that Lynch produced a theory based only on American cities necessarily provokes several questions, such as:

- What about cities different to these three American ones?
- What about cities that were built in a rich historical context, such as Prague?
- What about cities located in a specific natural environment, like Venice?

At the same time, some questions related to the theoretical model arise:

- Is his theoretical model also a universal model?
- Can this model be useful for other, more complex cities such as Paris, Jerusalem or Saint Petersburg?
- Could this theory be useful for well-known and lesser-known cities, metropolitan areas and regions?
- Is this theory also useful for smaller urban areas, like the campus of TU Delft?

In our research, with the goal of analyzing possible universal categories and criteria from Lynch's theory, we can try to add a couple of new elements that can be applied to some specific urban area (like canals).

A second critical point relates to the way in which Lynch explains the idea of cognitive mapping. In his study, Lynch explains this idea in a few very short points, latently, implied between the written words. He does not explain the concept in a readily comprehensible and direct way. He does not provide a sharp structure for the concept, but merely touches on it here and there. He does not think about its characteristics. Nor does he think about the process of making a mental map in a human brain. For Lynch, cognitive mapping was important, but not so important as to require explanation, for example, in a whole chapter of a study. He only announced this process as important. One is left with the feeling that he had touched on something that should have been seen as important, but at the time when he worked on the study, he did not realize its full importance. But cognitive mapping was to become very important. Many other, later theoreticians of cognitive science have seen cognitive mapping as a crucial part of Lynch's theory. Later in this chapter we will explained more on cognitive maps and Lynch's view on them.

Some other critical points can also be raised, but they are not essential to our research. Modifications to Lynch's theory, according to the criticism presented above, have to be incorporated during the process of creating the Design Tool. Such modifications should be aimed at adding new characteristics and criteria to enable the further analysis of almost any urban environment.

The general concept of *cognitive maps*

The term 'cognitive map' was first used by Tolman (1948), who used it to account for the behavior of rats in a maze that escaped and ran across the top directly to the food source. The rats had obviously integrated information about travel through the maze into a representation of its layout. It was previously thought that a maze is learnt only in terms of left and right turns at particular points.

Since Tolamn's experiment, many studies on the nature of cognitive representations of space have been carried out. Here are a few theoretical models of cognitive representation of space derived mainly from different works on visual imagery:

(1) *literal map in the head:* space is represented in a spatial way in the brain,

(2) *functional map in the head:* space is represented non-spatially (as propositional knowledge), but the representation functions identically to a map,

(3) *unlike a map:* space is represented non-spatially, but also does not really function as a map at all (spatial events directly influence our actions in space), and

(4) *hybrid representation:* different forms of representation are used for different areas, different scales, different functions (procedural representation on a large scale).

The perceptual quality of the environment reflects on the human mind, in many senses. Humans *recognize* an environment [space] by the reflection of shapes, light and depth. Humans *orient* themselves in an environment by identifying the environment by its *elements* and *patterns*. Humans make a *mental interpretation* of an environment by memorizing and retrieving elements of the environment and patterns in their brain. This process is defined as cognitive maps.

A cognitive map is an association of the cognition of environmental structure and properties of subjective meanings of the environment. In this research we will deal with environmental perception. The main characteristics of environmental perception are:

(1) *size* and *complexity:* the environment cannot be perceived all at once and is generally so complex that it takes some time to gain full experience of it,

(2) *surrounding:* because the environment surrounds us, we experience it from within, we move around, in and through the scene and are in a sense part of it, and

(3) *purposive connection:* we generally interact with the environment with a specific goal or plan in mind.

Environmental perception is different from perception of an object. Humans perceive an object in totality - all at once with all complexity of it. Humans can experience an object with its surroundings and with purposive connections. Humans even move objects from one place to another and change the context of objects.

Comprehending patterns in space places essentially identical requirements on cognitive structure. Humans must be able to anticipate which places come next. It is the continuity issue that makes the difference between a collection of isolated representations and a coherent structure, a cognitive map of space or time. Human experience tends to support this emphasis on continuity. Intuitively, our cognitive maps seem as continuous as the environments they represent. On functional grounds, the requirements are not for continuity but merely for connectedness. It is not necessary for one to know all the possible points between two places. It is only necessary for one to know that these two points are connected, that one can generally go from one point to the other. This is the mechanism of *association*: a representation is assumed to be *associated* with the other representations that have tended to follow it in an individual's experience (Kaplan, 1973). Connectedness transforms what would be a mere collection of representations into a cognitive map. Connectedness does not arise as isolated experiences; this association is built as the result of many experiences. Representations will not generally have unique associations. What makes this integrated and compact storage possible is the pattern recognition capability of the representation. Different experiences with any given place are overlaid on top of each other. These sequences of experiences are stored as overlapping patterns, yielding a whole network of associations.

Cognitive maps of a highly familiar environment are often packed with salient information. Thus it may be that a cognitive map intuitively feels continuous even though it is actually made up of connected, but discrete, representations. This view of the cognitive map places great emphasis on representations and hence on landmarks in the environment that representations stand for. Landmarks have a substantial influence on the way people remember and think of the environment and their experiences:

(1) landmarks tend to move to become the orientation decision point,
(2) landmarks will be recalled as being more prominent than they actually are,
(3) landmarks may be remembered as being in line of sight from the prior landmark, when actually they are only the next memorable thing after the prior landmark, and
(4) landmarks provide information about distance.

Cognitive maps provide a satisfactory basis for decision even when a great deal of information is missing.

It is unlikely that there is one single cognitive map that contains all of the information an individual has. Rather, each of us must have many different cognitive maps that are distinct but at the same time somehow interrelated. This suggests that there might be some sort of hierarchical relationship among the different cognitive maps that reside in a single head. One of the related problems is *scale*. Some maps deal with small areas with many details, others with vast areas where details are necessarily much more limited. A scale shift in the cognitive map is comparable to a shift to a totally different cartographic map. It is still meaningful to speak of representations making up the cognitive maps in different scales. Thus there are many circumstances where a cognitive map is in some sense part of another cognitive map. The high-level cognitive map is one that is more abstract or on a larger scale. Such *generic maps* can apply to many different situations, even to some that have not yet been encountered. Generic maps are no more limited to the physical environment than are any other cognitive maps. A generic map makes it possible to place different individuals in different spatial locations.

Lynch's view of cognitive maps

Environmental cognition has been approached from many perspectives. There are different ideas about how a cognitive map works and even about what constitutes a cognitive map. The most stimulating and influential - certainly for our research - is the view that Kevin Lynch presented in his book *The Image of the City* (1960). Tzonis and Lefaivre (1994) comment in their study of Lynch's *cognitive theory of the city*: "His concept of 'cognitive mapping', which is both verbal and pictorial, has become a key methodology found concurrently in the writings of postmodernist Marxists and cognitive scientists." (p. 23).

Lynch was an urban planner who carried out pioneering work on people's urban cognitive maps. He got his first ideas from Georgy Kepes, in the fifties, as Tzonis and Lefaivre (1994) describe: "Kepes is important because he provided a link between the purely American, pragmatic and empiricist environment in which Lynch existed, and prewar European aesthetic modernism in art as well as city planning." (ibid. p. 24). On the other hand, Lynch formulated some of his ideas from experience gained during his time spent in the military in the Second World War. He was an aerial photography analyst for years, and had everyday experience of analyzing the structure of bombed European cities.

Tzonis and Lefaivre (1994) comment: "Lynch tried instead to discover an empirically grounded methodology to represent the urban environment as its *users* capture it from within, to reconstruct their cognitive maps of the city." (ibid. p.25). Therefore, Lynch was mainly interested in how people structure their image of their environment, so as to design city layouts that would accord with the ways in which we perceive and understand our environments. For Lynch, being able to orient oneself in one's environment is a fundamental existential necessity for humans. In the age of modern cities, we need this sense in order to navigate between the numerous locations where we carry out our everyday activities. Lynch points out the fear that we associate with becoming disorientated in our surroundings: "The very word 'lost' in our language means more than simple geographic uncertainty; it carries overtones of utter disaster" (Lynch, 1960. p.4).

As a planner, Lynch was interested in analyzing the urban form, and in particular he identified the criterion of the 'legibility' of a cityscape which he defined as "the ease with which its parts can be recognized and can be organized into a coherent pattern" (ibid. p.2). A legible city would thus be one "whose districts or landmarks or pathways are easily identifiable and are easily grouped into an over-all pattern" (ibid. p.3). His method involved externalizing the 'mental images' that city-dwellers have of their cities, through interviews and sketch-mapping exercises. Because of his focus on identifying ways to improve the physical structure of cities, he was less interested in individual differences in mental images than in the aggregate image of inhabitants of a particular city. This 'public image' was used to identify aspects of good and bad structure in cities.

Tzonis and Lefaivre also explained that in Lynch's theory "the 'well-formedness' of a place was the result of a happy relationship between the cognitive structure of the inhabitant's mind and that of the 'lived-in' environment that lead to successful recognition, memorization and navigation of the city and its parts." (Tzonis and Lefaivre, 1994. p. 25).

Lynch proposed a set of elements that he felt to be fundamental to the structure of the urban environment and thus expected to be manifest in people's mental structuring of the environment. Lynch's theory will be elaborated in Chapter 6. Lynch made the definition that city elements *operate together, in context* (Lynch, 1960. p.84). The recognition of an object is as much dependent on a context as on the form of the object itself. He also defined all urban elements as arranged and interrelated in four structural stages (ibid. pp.88-89):

(1) the various elements are free; there is no structure or interrelations between parts,
(2) the structure becomes positional; the parts are roughly related to each other,
(3) most often, the structure is flexible, and
(4) as connections multiply, the structure tends to become rigid.

This characteristic of structure might be applied in different ways to different levels of *imageability*.

Lynch also proposed a method which was developed on the basis of research conducted in two ways: first, urban residents were interviewed about their city and asked to draw sketch maps (cognitive maps by the process of cognitive mapping); second, trained field workers were sent out on foot to make detailed plans of the city with the five elements and their interrelationships in mind. All the data were analyzed to determine:

(1) what were the distinctive features of each city and which areas were more or less legible, and
(2) how well the field maps compared with the aggregate maps of the interviewees.

He also found that different parts of the city were differentiated in terms of their legibility, defined as the strength of imageability of elements and of the structural interplay between elements. From these maps and from the rich detail included in the interviews, Lynch went on to outline a set of criteria for improving the legibility of elements and structure.

Tzonis and Lefaivre (1994) nicely concluded for us that "during the 1980s the research on spatial thinking was joined with developments in artificial intelligence and machine-based simulation of pattern recognition, spatial memory, problem solving and 'navigation', and Lynch's notion of cognitive mapping became a frequent reference in this developing literature. As these studies find their way today to a cognitive theory of design, the significance of Lynch's contribution becomes even more indisputable than ever. We are just beginning to suspect the possibilities that his pioneering method offers." (Tzonis and Lefaivre, 1994. p. 26).

Prior research on the topic - post-Lynchians

A scientist searches for information related to things that can be used primarily in urban perception problems, which can help individuals locate themselves in urban areas and act as pointers for building images of the urban environment. An individual's conceptual structure of the urban environment will in some way conform to the physical models that have been derived from objectively analyzing the locations of segments of the urban environment.

Research into the cognitive structure of the urban environment can be subdivided into three parts:

(1) micro-level studies: the aim of this approach is to find out how well people can locate specific points in urban areas (Golledge, Briggs & Demko, 1966; Lee, 1970; Briggs, 1971; Passini, 1984, 1990);

(2) the problem of determining what metric is used to measure distances in urban areas. It relies on a variety of scaling techniques to extract information from data sets. The first is exemplified in the attempt to define perceptually small areas of

the city - generally described using the term 'neighborhoods' (Sarrinen, 1964; Zannaras, 1969); the second is an attempt to reconstruct a set of urban features which are observed on journeys through urban areas (Appleyard, Lynch & Meyer, 1964; Carr & Schissler, 1969; Kosslin, 1994); and

(3) a macro-level emphasis attempting to reconstruct maps of urban areas from the knowledge that individuals have about those places (Lynch, 1960, 1976; Kaplan & Kaplan, 1982; and others).

The difficulty is in exploring how an urban environment is represented in people's minds and memories, or more specifically, how mental maps are correlated to cognitive perception. Ittelson (1970) defined the characteristics of environments in a few categories:

(1) the quality of surrounding and defining property - forces the observer to become a participant,
(2) environments are always multimodal,
(3) the peripheral, as well as central, information is always present; peripheral in the mechanical sense - the area behind one is no less a part of the environment than that in front - and peripheral in the sense of being outside the focus of attention,
(4) the environment always provides more information than can possibly be processed - the environment always represents, simultaneously, instances of redundant information, of inadequate and ambiguous information, and of conflicting and contradictory information,
(5) environment perception always involves action - the environment is not and cannot be passively observed; it provides the arena for action,
(6) the environment possesses the property of providing symbolic meanings and motivational messages which themselves may affect what direction the action takes - meanings and motivational messages are a necessary part of the content of environmental perception,
(7) the environment always has an ambience and atmosphere, and
(8) the first four characteristics deal roughly with stimulus properties, the other three explain the role of the action and task limited and caused by the environment.

Regardless of individual differences, people seem to organize perceptual responses to the environment and interrelated levels of analysis. These are effect, orientation, categorization, systematization and manipulation. All these processes are involved in environment perception and they do not function sequentially, rather they interact with each other continuously. The environment acting on humans, or humans acting on the environment. The environment is thus seen as a total, active, continuous process involving the participation of all aspects. In any concrete situation, the environment has no fixed boundaries in either space or time.

People as observers will build their cognitive images from existing environmental features by selecting and organizing those which are meaningful to them. When moving through the city toward some specific goal or location, humans use objective maps when their cognitive map lacks coherence.

The study of cognitive maps and 'urban images' has been one of the major areas of interest in urban perception. The concern is with the maps or pictures of a city that people have in their minds; how they develop; what affects their form; how the maps differ from individual to individual; and how they influence behavior. Carr (1966) argued that there would be great individual differences in the images and the needs of the citizen, based on the relative social value that particular districts have for them. Stea (1969) concluded that people order their conceptions of the environment in terms of hierarchies, boundaries, and relationships between points and connectedness (paths). Since then, many studies have been carried out replicating Lynch's technique (Alexander, 1969; Milgram, 1970; Downs & Stea, 1973; Kaplan & Kaplan, 1982; Passini, 1984, 1990; and others). An overview of the recent literature on 'cognitive maps of spatial arrangements' is given in Forman and Gillet (1997, 1998).[1]

Cognitive maps are a process comprised of a series of psychological transformations by which an individual acquires, codes, stores, recalls and decodes information about the relative locations and attributes of phenomena in his everyday spatial environment (Downs and Stea, 1973). Downs and Stea also emphasize that cognitive maps comprise not only spatial knowledge, but also all other attributes of a place/location. A cognitive map is functionally equivalent to a cartographic map. Cognitive maps are complex, highly selective, abstract, generalized representations in various forms. A simple framework of the abstraction process in mapping, presented in Table 3.2, defines mapping as the transformation of an *object set* into an *image set* via a *mapping function*. Spatial attribute sets of the mapping function include shape, size, relative distance and direction.

[1] See also at: http://www.psypress.com/BKFILES/0863777996.htm

Table 3.2
Conceptual *frameworks* of the abstraction process in mapping [from: Downs & Stea]

ENVIRONMENT	OBJECT SET		IMAGE SET		MAPPING FUNCTION
	urban environment	→	cartography	→	city map
	map	→	cognitive perception	→	cognitive map

Planners, architects, sociologists and psychologists have been interested for various reasons in the perception and cognition of the larger environment. Perception cannot be understood in isolation from values or behavior. Looking at the city as a product of different group perceptions, three theoretical models of urban perception appear (Appleyard, 1976):

(1) operational: the environment is seen as a setting for personal action and behavior; more often elements such as traffic islands, signs, entrances etc.,

(2) responsive to the configuration of the environment; J. J. Gibson (1979) gives the term for this kind of perception 'literal'; and

(3) inferential and probabilistic in nature, a generalizable coding system of environmental categories, concepts and relationships: our personal urban model.

The representation of urban perception, direct and indirect, stands between man and the city in an array of media. Newspapers, radio, TV, the electronic highway all take part in urban information. The influence of the media on our comprehension of the spatial city has, no doubt, been profound (McLuhan, 1965). Modes of walking or travel can cause wide variations in the form and extent of a person's comprehension of the city. Each person's mode of representing the city has differences according to the extent of his knowledge, and the accuracy of his spatial perception. The use of maps, aerial photographs and other known media that directly facilitate objective understanding of the spatial city is confined to professionals, although public maps are available to help orientation. Maps must be learned and matched, as abstract graphic symbols of reality, with direct experience.

No formal survey of planners' and designers' perceptions was made; they habitually drew both formal and sketchy maps of the city. These maps described the distribution of uses in the broadest categories and also more specifically. This 'plan' world was less complex but more extensive than people's knowledge. Only a few functional types were selected for their significance. The planners' knowledge of the city was not restricted to the language of their maps, but their maps were good indications of their modes of thinking. Language directs thought (Vygotsky, 1962), and many decisions were made on the basis of map information. The view of urban designers and architects is based on the media and language they use to simulate the city.

Cognitive mapping is a sort of accumulation of the experience and knowledge that a person has about a familiar environment. It is knowledge for a task, knowledge with a function. It guides the behavior of the 'owner' of the map, to help the individual in whose head the map resides to be effective in the given environment. Cognitive maps store information about the environment, so that a person knows what to expect and what to do under various circumstances. A consideration of the requirements for functioning leads to broad criteria:

(1) generality; thus the information we have about the environment must in some way be general rather than particular;

(2) economy; the explanations one finds for the way in which the environment is coded in the mind must come to terms with the economy of that storage system; and

(3) connectedness; a need for connections between known points, otherwise getting from place to place would generally be impossible.

To appreciate how impressive this internal model is, it might be useful to consider the approaches taken by some storage/retrieval mechanisms, such as computers. Computers handle the problem of storing and retrieving information all the time. As has been shown, the vital element in an individual's model of the environment is the representation. Computer devices could help people to understand cognitive maps with a representation of information processing of spatial data.

A representation is a formal system for making explicit certain entities or types of information, together with a specification of how the system achieves this (Marr & Nishihara, 1978). To describe a representation as a collection of salient features weighted in terms of their importance, is essentially the same as calling it an average, or summary, of the experience that led to it (Kaplan & Kaplan, 1982). Representation also means an internal summary of a class of stimulus patterns. Its role is to take the place of some object in the real world (i.e. urban elements), and to represent it.

The transformation from incoming information to the turning on of the appropriate representation involves at least four different aspects that leave their mark on perception:

(1) *simplicity:* information is discarded - there is far less information in a representation than in the many experiences that led to it - this does not mean that our experience is restricted to the information contained in the
representation,

(2) *essence:* the loss of information is not random, but highly systematic - what is retained is what tends to be reliable and characteristic,

(3) *discreteness:* a representation involves the separation of experience into distinct categories,

(4) *unity:* the representation tends to behave as a thing.

Together, these properties color the human experience of the environment. They shape not only what people perceive but what they do as well. Since an urban element has some structural relationship to what it represents, it could certainly be considered analog. The urban element is based on representations (which are necessarily generic) and associations (which constitute the paradigm relation). Hence urban elements must be analog and generic and relational.

The modeling of the material and social world and of human behavior is a complex and ambitious task. It demands first of all a clear distinction between the concepts of *model* and *simulation*. Modeling and simulation are two complementary ways of representing a process, which can be either physical or cognitive. A *model* is essentially a theoretical description of a process or system, based on a number of hypotheses and simplifying principles, which can be formulated as analytic or lexicographic expressions (the model language). A *simulation* is a concrete expression or instantiation of the model in a form that is controllable or executable, for example, for a practical application or for computation (Hollnagel, Cacciabue & Hoc, 1995).

There are many possible ways to store information, to structure people's knowledge. It is exciting because the actions people take, the decisions they make, their hopes, their fears, their aspirations are all based on their conceptions, on their models of the world. Forrester (1971) captured the impact people's models have on their choices and actions very effec tively: *"A mental image is a model. All of our decisions are taken on the basis of models. All of our laws are passed on the basis of models. All executive actions are taken on the basis of models. The question is not to use or ignore models. The question is only a choice among alternative models."* (p. 17).

A conceptual model is a kind of system model that combines the properties and functionality of both a mental model and a real operating model. Models of human cognition have been developed in order to predict human behavior through computer simulation (Kjaer-Hansen, 1995).

Today computers can help us to solve difficult, real-world problems, to create new opportunities in many areas, such as in our research, to help in analyses of the visual quality of the urban environment according to cognitive mapping.

Generalizing cognitive maps

As we have seen, a cognitive map is a highly general cognitive structure of the information on the physical world that humans perceive, process and store. We have just introduced things that are tied directly to ways of understanding environmental cognition. People representing a variety of disciplines and referring to a variety of subject matter have used the idea of cognitive maps. Its lends itself to discussion of the future, to the structure of organizations and to conceptual structures of all kinds.

The physical world is Euclidean, being defined by three dimensions. Location within that space is absolute. Being in space with no semantic, absolute position has no meaning. Looking at it another way, location can be given semantics by defining space through the use of labeled dimensions. If the world were to become a confusing place in which everything constantly changes location, absolute distance and direction would not be available as orientational cues and consequently cognitive maps would become redundant.

People cannot and do not know the world in its every detail. Rather, they tend to remember the essentials and how these are related. They can preserve the essential relationship that exists in reality. Such an arrangement of ideas is often referred to as a model. A mental image is a model. A cognitive map is a model. Each model is a compact, orderly collection of knowledge. It contains more information than we can generally perceive at once. It permits one to anticipate, to react, and to consider possible forthcoming events. Having a more concrete conception of future possibilities tends to simplify one's cognitive processing, and to contribute substantially to one's confidence.

Our representations of environment constitute more than featureless sets of relationships; we bring a rich set (*attribute* set) of expectations about how a place should look. *Descriptive* attributes represent sensory features that identify a place, while *evaluative* attributes are the evaluative tags we attach to our models of places.

Different sources of spatial information exist, and it is possible to distinguish between *direct* and *vicarious* sources.

Direct sources refer to experiences at 'ground level' which are generally reliable sources of information about the environment. This source is experiencing the world at first hand, or by 'learning by doing'.

Vicarious sources are 'second-hand' descriptions of places, such as verbal reports, maps, brochures, television, etc. The main characteristic of these types of sources is a view of the environment through another's eyes. Information has been filtered.

Both sources of spatial information help people to make mental images of mental maps of familiar as well as unknown environments. These mental images will increase people's knowledge in everyday way-finding and navigation through different environments, familiar as well as unknown [for example, in discovering new cities].

Some remarks about the urban environment

Cognition is intimately tied to motivation because of the 'goods' and 'bads' of our experience. The process of using a cognitive mechanism has motivational implications. When we recognize something or when we make a good prediction, or when we make a good navigational decision, it feels good. This cognition dealing with emotionally laden material is called 'hot' cognition (Abelson, 1963). The connection between cognition (what we can do) and motivation (what we want to do) expresses itself not only in pains and pleasures per se, but also in the environment we seek and prefer. Since we have a need to be cognitively effective, we tend to prefer environments that help make this possible.

As we have seen, the preference framework has been concerned with two basic informational needs: (1) making sense, and (2) involvement, and in a time dimension those focuses on the immediate and the longer-term possibilities. The scene of the environment is not merely something to perceive, but something to enter into. Designers must consider how one would function if one were to enter into the space and move around it. Thus the longer-range, or more future, aspect of preference depends upon the analysis of the inferred three-dimensional space. The more immediate aspect of preference seems to involve the two-dimensional qualities of the scene, or what we previously referred to as the 'picture plane'. The main elements and relationships of the scene are:

(1) *coherence* is the 'fittingness' of an element in a setting,

(2) *redundancy* is the possibility of treating a whole area of the scene as the same; repeating elements is another aspect of redundancy,

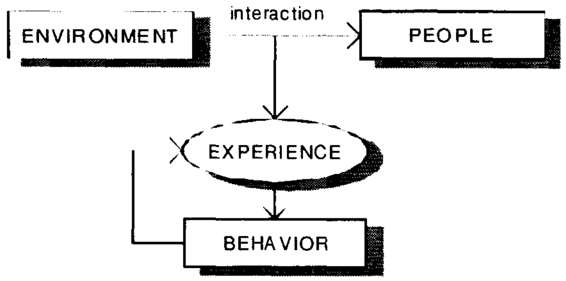
(3) *complexity* is based on the number of different countable things, the visual 'richness', or diversity of the scene; it is all too easy to have complexity at the expense of coherence,

(4) *mystery* provides partial information concerning what might lie ahead; new information is not present, it is only suggested or implied, and

(5) *legibility* is characteristic of an environment that looks as if one could explore it extensively *without* getting lost.

People prefer environments in which they can make sense of what they perceive and in which they can learn more. Environments with which they will easily interact. When a person becomes an information recipient, those preferences are pertinent. People are motivated to achieve clarity; they tend to evaluate information in terms of its contribution to this process. People use learned experience in their interactions with the environment. An elementary presentation of how people interact with their environment is shown in Figure 3.2.

Figure 3.2
Interaction between people and environment [based on Lynch]



Conclusions

The proposed Design Tool for analyzing and controlling the visual quality of an urban environment can be developed based on different areas of theoretical knowledge. One area of theoretical knowledge relates to urban design. We need to examine urban design theory from two aspects: the hierarchical structure of urban elements and the concept of cognitive maps. These two aspects of urban design theory were developed by Kevin Lynch in his 'theory of urban form'.

This chapter briefly described Lynch's theory of urban form and some other related theories and the concept of 'cognitive maps', which constitute the main theoretical platform for the further development of the proposed Design Tool.

First we explained Lynch's theory of *urban form* with its main characteristics and elements. We explained four elements that belong to the visual quality of the city: *legibility, building the image* [*image*], *structure and identity* [*identity*], and *imageability*. We then briefly explained the five elements of the urban environment: *paths, edges, districts, nodes* and *landmarks*. These five elements of the urban environment will be examined in detail in Chapter 6. We then presented the graphical notational system which Lynch proposed in his theory. At the end of this section we set out some criticisms of Lynch's theory and made some suggestions for modification during the development of the Design Tool.

In the next section of the chapter we explained the general concept of cognitive maps, which can play a crucial role in the Design Tool development. We also gave Lynch's view on cognitive maps, which relate to individuals and their capacity for memorizing and retrieving information about a familiar environment. Some of the generic aspects and characteristics of cognitive maps presented in this chapter will be implemented in the Design Tool development.

One section of the chapter was dedicated to post-Lynchian thinking and prior research on the topic. There we elaborated different theoretical approaches related to cognition, spatial representation, cognitive mapping, modeling and some other theories, all appearing after the publication of Lynch's theory. In the last two sections of the chapter we generalized cognitive maps and made some global remarks about urban environments. This chapter represents the results of an examination of theories that can be useful for the Design Tool development.

For developing the proposed Design Tool we also have to explore theories that relate to cognition, with a focus on computer vision and image recognition. The knowledge that will be set out in the next chapter, deriving from cognitive science and computation, will constitute the second important theoretical background for developing the proposed Design Tool.

CHAPTER 4

THEORIES OF COGNITION - COMPUTER VISION

***Task of the chapter**: 'computer vision', and 'high-level vision' in particular, will be computational theories and methods which we will adopt to help us in the process of object recognition for objects in the urban environment in our tool development.*

Introduction

In the previous chapter we investigated theoretical knowledge related to urban design from two perspectives: the hierarchical structure of urban elements and the concept of cognitive maps, which Kevin Lynch developed in his 'theory of urban form'. We explained four elements that belong to the visual quality of the city: legibility, building the image [image], structure and identity [identity], and imageability and the five elements of the urban environment: paths, edges, districts, nodes and landmarks. The graphic notational system, which Lynch proposed in his theory, is also explained in the chapter. We also reviewed post-Lynchian thinking and prior research on the topic related to cognition, spatial representation, cognitive mapping, modeling and some other theories. To understand the knowledge of cognitive aspects of the urban environment we need to introduce theories of cognition vision. In this chapter we will elaborate recent theories in the field, which can be helpful for the development of our Design Tool.

In the process of architectural and urban design, perception of the three-dimensional world constitutes crucial knowledge and experience about space and its characteristics. Knowledge about how people perceive the environment and how they reproduce visual information is the basis for understanding the process of cognitive mapping. It is therefore important for our research and the development of our cognitive mapping computer-based Design Tool for evaluation and control of the visual quality of the urban environment.

Three-dimensional spatial perception is concerned with how humans perceive the space around them through vision. In our research, we will focus on conceptual aspects related to spatial vision leaving aside neuro-cognitive aspects. We will briefly discuss some theoretical points concerning human vision and we will examine aspects of the theory of computer vision. The theory focuses on information processing, different ways in which motion may be used to recover structure from a sequence of images and how the results of such processes can be represented efficiently for recognition.

We will also introduce the high-level vision or object recognition drawing from the work of Shimon Ullman and his ideas about how the brain 'sees'. We will discuss the processes of classification, scene segmentation, visual routines, saliency and neural mechanisms of the visual cortex that will help us to understand how the visual system functions.

Vision

The human vision system comprises a receptor system – the eyes, and a processing system – the brain. The functioning of the human visual system has been discussed in depth in general studies of vision in Marr (1982) and Bruce & Green (1985).

Vision is an information processing task that provides information about the shape and spatial relations of physical objects. "Vision is a process that produces from images of the world a description that is useful to the viewer and not cluttered with irrelevant information." (Marr, 1976; Marr &

Nishihara, 1978). In studies, two main approaches to vision can be found: Gibson, 1979 and Marr, 1982. The main point of discussion about these two approaches and their authors concentrates on the problem of how physically measurable energy is converted into meaningful information (to perceive a three-dimensional object, for example). The approaches can be distinguished as 'direct' and 'indirect' (Overbeek & Stratman, 1988).

Gibson's 'direct' approach (1979) assumes that the light which falls on the retina is structured and unambiguous. This structured light, called the optical array, contains all essential information about the environment. According to this approach the image on the retina does not need to go through a reconstruction process for a three-dimensional impression of the environment to be obtained, because of the non-ambiguous characteristics of the light. This is also the reason why this approach is called 'direct'.

In Maar's 'indirect' approach (1982), a classical theory, it is assumed that the light projected on the retina is ambiguous and can be interpreted in more than one way. A process of reconstruction of the two-dimensional retinal images is necessary to come to a three-dimensional impression of the environment. This can be done using the cues available in the retinal images.

Common to both theories is the fact that they acknowledge the existence of depth information.

Here we will introduce some important details of Marr's (1982) theory of computer vision and Ullman's (1996) theory of object recognition and visual cognition. The reason for choosing these two theories is that they strongly support theoretical vision in the computational process needed for this research. Marr's theory will give us some preliminary pointers for developing a new model and Ullman's theory should give us particular methods for solving the main problems of object recognition and classification.

Computer vision

"Vision is a process of discovering from images what is present in the world and where it is." (Marr, 1982 p. 3). Vision is the predominant mode of human perception. Vision is fast and accurate, reliable and non-inintrusive, simultaneously providing detail and an overview. Vision provides the picture that is worth a thousand words. Vision is not a sound, not a touch, smell or taste. It is a kind of process of representation of information in our brain. It entails recovering the structure and properties of the visual world such as color, forms, beauty, motion and details.

But what is computer vision and why we are interested in it?

Computer vision, or image understanding, is the automatic/computerized deduction of the structure and properties of a dynamic 3D world from either a single image or multiple 2D images of the world. These images may be monochromatic or colored, they may be captured by one or more cameras, and each of them can be stationary or mobile. The structure and properties of the 3D world that we seek to deduce in computer vision include geometric properties, such as: shapes, sizes and locations of objects, and material properties, such as: lightness or darkness of surfaces, colors, textures and their material composition.

We need computer vision to help us to recognize, predict, navigate, manipulate, reorganize, and so forth, information from the real 3D world through the task of preimaging this information for a simulation-scale of 2D representation. In our case, computer vision is the second theoretical platform for developing a conceptual model of the Design Tool. Here we will now present a brief overview of the theory set out by Marr (1982).

Marr's Representational Theories of the Mind

The focus of the research here is on the science of computer vision, the primary goal being the elucidation of the fundamentals of Marr's approach.

The philosophy and the approach

A representation implies the notion that someone can capture some aspect of reality by making a description of it using symbols. According to Marr's (ibid. p. 20) definition, "representation is a formal system for making explicit certain entities or types of information, together with a specification of how the system does this." In computation terminology we are dealing with information-processing which uses symbols to represent things. Marr called the set of symbols for putting computational rules together the 'formal schema' of representation. The importance of this issue is noticed: "how information is represented can affect how easy it is to do different things with it" (ibid. p. 21).

Marr divided important features in the computational process[1] into a couple of levels. The first level is that it contains separate arguments about what is computed and why, and the second level is that the resulting operation is defined uniquely by the constraints it has to satisfy and specify how. "In the theory of visual processing, the underlying task is to reliably derive properties of the world from images of it" (ibid. p 23).

The first level of running the computational process is to make a description of the entities that the process manipulates. The second level of the analysis of the process "involves two things: (1) a representation for the input and output of the process, and (2) an algorithm by which the transformation may be accomplished" (ibid. p.23).

The first level specifies what and why, the second level specifies how. There is usually a wide choice of representation. The choice of algorithm often depends on the particular representation that is employed. "For a given fixed representation there are often several possible algorithms for carrying out the same process" (ibid. p. 23).

This brings us to the third level - hardware implementation, or the decision concerning which devices the process is to be realized on physically. Table 4.1 illustrates the three different levels at which an information-processing device must be understood.

Some phenomena can be explained at just one or two of the levels. The correct explanation must be formulated at the appropriate level. The level of algorithm and representation is more directly related to psychophysics. Psychophysics has also helped to determine the nature of representation. A general view of the differences between a brain and a computer would be that the brain is parallel and the computer is serial, this distinction being on the level of algorithm. Anything programmed in parallel can be rewritten serially but not necessarily vice versa. "The brain operates so differently to the computer that the computer could not be programmed to perform the same tasks" (ibid. p. 27).

[1] The *computational process* can be defined as "meanings associated with machines that are carrying out information-processing task" (Marr, 1982 p.22)

Table 4.1.
The three levels of an information-processing device must be understood [from: David Marr, 1982 p. 25]

Computational theory	Representation and algorithm	Hardware implementation
What is the goal of the computation, why is it appropriate, and what is the logic of the strategy by which it can be carried out?	How can this computational theory be implemented? In particular, what is the representation for the input and output, and what is the algorithm for the transformation?	How can the representation and algorithm be realized physically?

Representing the Image

In Marr's theory (ibid. p. 41), there are four main factors responsible for the intensity values of the image:

(1) the geometry,
(2) the reflectance of the visible surface,
(3) the illumination of the scene, and
(4) the viewpoint.

The function of the early visual processing is to sort out which changes are due to which factors and hence to create representations in which the four factors are separated.

Suitable representations of the changes and structures in the image are obtained. The result of this first stage is a representation called the primal sketch[2].

In the second stage, the primal sketch is manipulated to derive a retinocentric representation of the geometry of the visible surface, called the 2½-dimensional (2½D) sketch[3].

[2] *Primal sketch* is defined as "a representation of the two-dimensional image that makes explicit the amount and disposition of the intensity changes there. The representation is hierarchical, the primitives at the lowest level representing raw intensity changes and their local geometrical structure, and those at the higher levels capturing groupings and alignments occurring among the lower items." (Marr, 1982 p. 366)

[3] *21/2-D sketch* is defined as "a viewer-centered representation of the depth and orientation of the visible surface, including contours of discontinuity in these parameters." (Ibid. p. 367)

"The purpose of this representation is to provide useful descriptions of aspects of the real 3D world and their structure which play important roles in determining the nature of representations and processes that derive and maintain them" (ibid. p. 43).

According to Marr (ibid. p. 52), the general critical ideas of the primal sketch are the following:

(1) the primal sketch consists of primitives of the same general kind in different scales; but primitives can be defined from an image from very concrete to very abstract in a variety of ways,

(2) the primitives are built up by analyzing and representing the intensity changes and forming tokens directly from them, then by adding representations of the local geometrical structure of their arrangement, then by processing things with active selection and a grouping process to form larger-scale tokens that reflect the larger-scale structure in the image, and so forth,

(3) projected orientations in the image change if the surface orientation changes.

The three main stages in the process that derives the primal sketch are:

(1) the detection of zero-crossing,

(2) the formation of the raw primal sketch, and

(3) the creation of the full primal sketch.

Zero-crossing[4] concerns the detection of intensity change. The intensity changes in different scales in an image. Zero-crossing is the point at which the value of a function passes from positive to negative. Zero-crossing provides a natural way of moving from an analog or continuous representation, such as 2D image intensity values, to a discrete, symbolic representation. A fascinating thing about this transformation is that it probably incurs no loss of information.

The raw primal sketch is a rich description of an image. Its primitives are edges, bars, blobs and terminations, and these have attributes of orientation, contrast, length, width and position. This is important because "it is the first representation derived from an image whose primitives have a high probability of directly reflecting physical reality directly" (ibid. p. 71).

Each zero-crossing and each descriptive element of the raw primal sketch has a coordinate in the image that determinates its position there. Spatial relationships, such as density, collinearity, or local parallelism, are all implicit in the positions of each item. Marr defines (ibid. pp. 80-81) that the spatial relations are important for decoding surface geometry:

[4] *Zero crossing* is defined as a "point where a function's value change the sign." (Ibid. p. 368)

(1) average local intensity (changes in average intensity can be caused by changes in illumination, in depth or in surface orientation or reflection),
(2) average size of items on a surface (size includes a concept of length and width),
(3) local density of the items,
(4) local orientation if such exists,
(5) local distances associated with the spatial arrangement of similar items, the distance between neighboring pairs of similar items, and
(6) local orientation associated with the spatial arrangement of similar items, the orientation of the line joining neighboring pairs of similar items.

Another important account of spatial aspects of the image and visible surface is light source and transparency. Marr adopted Ullman's (1976) discussion "of light-source detection methods based on the highest intensity in a field, high absolute intensity, high intensity compared with average intensity in the field, high contrast, and some other parameters" (ibid. p. 86). None of these factors defined the necessary conditions for the perception of the light source.

There are, in general terms, two main goals in the spatial organization of images: firstly, to make tokens, and secondly, to find boundaries. Generally, the approach is to build up descriptive primitives in an almost recursive manner. The raw material from which everything starts is the primitive description obtained from the image that we called the raw primal sketch. To make tokens or primitives in each scale that capture the spatial structure in that scale, it is necessary for roughly similar elements to be initially selected from it and to group them and cluster them together, forming lines, curves, blobs, groups and small patches. This is the process of building up the full primal sketch. When these primitives have been constructed, they can tell us about the geometry of the visible surface. This is an important issue in the case of analyzing objects from an image scene [of an urban environment] with possibility of directly taking corresponding data from the database structure in our Design Tool.

There are two ways for boundaries to be detected: one is to find sets of tokens that owe their existence to the physical discontinuity and are therefore organized geometrically along it. The importance of adding boundaries to the representation of the image is that they may provide significant evidence about the location of surface discontinuities. Marr called this "the hypothesis of geometrical origin for perceptual texture boundaries" (ibid. p. 94). There are a couple of different criteria for the detection of different types of boundaries created from a change in visual texture. Here we will give an example, taken from a snap-shot from a video of a VR simulation of an urban environment, to illustrate surface and light, image organization, boundaries and perceptual texture as discussed here (see Figure 4.1 below).

Figure 4.1
Simulation of an urban environment illustrating surface and light, image organization, boundaries and perceptual texture
[Image taken from: Sidjanin, 1995]

From images to surfaces

The two eyes, as we know, form slightly different images of the world. The relative difference in the position of objects in the two images is called disparity. Our brains are capable of measuring disparity and of using it to estimate the relative distance of the object from the viewer. The subjective distance to the object as perceived by the viewer is called depth.
Why might something like that work in a computational process?
First, we need some further explanation to help us to answer this question. The constraints we need are: first, a given point on a physical surface has a unique position in space at any one time, and second, meter is cohesive and the surface of the object is generally smooth, or there are other sharp differences that can be attributed to changes in distance from the viewer. The three matching constraints are: compatibility, uniqueness and continuity. Marr directed these three restrictions to the task of making "the fundamental assumption of stereopsis: If a correspondence is established

between physically meaningful primitives extracted from the left and right images of a scene that contains a sufficient amount of detail, and if the correspondence satisfies the three matching constraints, then that correspondence is physically correct" (ibid. p.115). Marr also gave two algorithms for stereo matching, which will not be explained here, since they are not fundamental to our research.

Visual motion is a process of studying how information about the organization of movement in an image can be used to make inferences about the structure and movement of the 3D real world. The main aspect of the motion is the analysis of time, not only because moving things can be harmful, but also because old descriptions of the state of a moving body soon become useless. Two types of information can be gleaned from motion: first, noticing a movement and finding its position in the visual field, and second, determining its 2D shape. Generally the issue of visual motion could be important for developing the conceptual model of the Design Tool in the field of analysis of video images or computer animations. See also Figure 4.1 for an illustration.

"The movement of an object against its background can be used to delineate the object's boundaries" (ibid. p. 175). The human visual system is used to explore this effect. The complete velocity field (speed and direction) is not directly available from measurements of small oriented elements. An additional stage is necessary for detecting discontinuities in the velocity field. Movement is a process that usually produces smooth changes in an image. That is important for the perceptual analysis of an image. The fundamental approach is to contrast the positions of the items at one time in the image with their positions at a sufficiently later time for the differences to be measured reliably. The differences can be used to make calculations about the underlying shapes and movements. Marr analyzed Ullman's concept of the correspondence process of motion. In Ullman's (1979) framework, the goal of the correspondence process is to establish a relationship between successive frames that allows measurements of the changes that have taken place. These measurements can provide the input for a subsequent process that can recover the structure and its motions. Most structures in the visual world are rigid, or at least almost so. Ullman (1979) defined the rigid assumption as: "any set of elements undergoing a two-dimensional transformation that has a unique interpretation as a rigid body moving in space, is caused by such a body in motion, and hence should be interpreted as such" (Ullman, 1979 p. 419).

According to Marr (1982, p. 215) there are four basic ways in which contours can arise in an image:

(1) discontinuities in distance from the viewer,

(2) discontinuities in surface orientation,

(3) changes in surface reflectance, and

(4) illumination effects such as shadows, light sources and highlights.

Contours in an image are two-dimensional, yet we often see them in three-dimensions. Shape contours are all two-dimensional contours that yield information about three-dimensional shapes. There are three types of contours:

(1) contours that occur at discontinuities in the distance of the surface from the viewer (occluding contours),

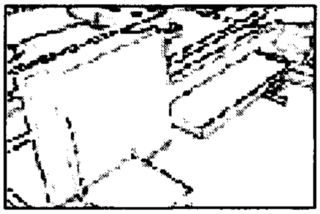
(2) contours that follow discontinuities in surface orientation, and

(3) contours that lie physically on the surface.

The third type is a shadow line, for example. An occluding contour is a contour that marks a discontinuity in depth and this corresponds to the silhouette of an object as seen in 2D projection. Marr (1977; 1982 pp. 219-222) described three assumptions related to contours. The first is that each line of sight from the viewer to the object should graze the object's surface at exactly one point. The second is that nearby points on the contour in an image arise from nearby points on the contour generator on the viewed object. A contour generator is a piece of wire bent in 3D space. That gives a third assumption: the contour generator is planar. The contours will play an important role in our development of the conceptual model of the Design Tool. Figure 4.2 illustrates the row contour map or shape contours of objects from the image scene, taken from the example of a video simulation of an urban environment presented in Figure 4.1. During the process of analysis of an image through the testing simulation of the proposed conceptual model of the Design Tool, the image will be presented as contours and explored in several steps; all this will be briefly discussed in Chapter 10.

The generalized cone is the idea that the surface is created by moving a cross-section along an axis. If the convexities and concavities of a bounding contour in an image are actual properties of a surface, then that surface is a generalized cone. A natural link exists between generalized cones and the imaging process itself.

Figure 4.2
A row contour map [shape contours] of an image scene of the urban environment



Surface orientation contours mark the loci of discontinuities in surface orientation. The question about a contour is whether it corresponds to a convexity or a concavity on the surface. One cannot have two concave contours and one convex contour meeting at a point. Surface contours arise for various reasons in the image of smooth surfaces and they yield information about the three-dimensional shape of the surface. The recovery of surface orientation from surface contours remains an intriguing and unsolved problem.

Gibson (1950) formulated a hypothesis that texture is a mathematically and psychologically sufficient stimulus for surface perception. There is sufficient information in the monocular image of a textured surface to specify the distance to points on the surface and to specify the local surface orientation. Surface orientation itself is split into two components, one is slant which is the angle by which the surface dips away from the frontal plane, and the second is tilt which is the direction in which the dip takes place.

Marr also introduced the phenomena of shading, brightness, lightness and color. He explained them briefly and only psychophysically, by comparing different theories already in existence. He gave ways in which surface information is encoded in images and explored possibilities of how such information may actually be recovered.

The Immediate Representation of Visible Surface

The pivotal point in the theory of vision is the construction of the 2½-D sketch (Marr and Nishihara, 1978). It told us what the goals of early vision were, and it related them to the notion of an internal representation of objective physical reality that preceded the decomposition of the scene into 'objects'. It hinted at the limits of 'pure perception', the recovery of surface information by a purely data-driven process.

Image segmentation is the process of dividing the image into regions that are meaningful for the task in hand or for their correspondence to physical objects or their parts. An object can be composed of parts and/or it can be a part of a larger object. The structure of the image might be so complicated that it is impossible to recover the desired region by using only grouping criteria, based on local similarity or other purely visual cues. Regions that have 'semantic' importance do not always have any particular visual distinction. The notion of segmentation continued to be investigated with increasingly complex techniques. The main question is what information we should try to recover from an image, and then our task is to design a representation for expressing it. This information is the intensity values in an image. We need to preview the general classification of shape representation, information provided by psychophysics and to look at the computational aspect of the problem. According to Marr (ibid. p. 298), the general nature of shape representation has three characteristics:

(1) the type of coordinate system the representation uses,
(2) the nature of the shape primitives used by representation, and
(3) the information in a description.

Vision provides several sources of information about shape. The most direct are stereopsis and motion. Stereopsis and motion are capable of delivering depth information directly. Surface shading and contours provide more direct information about surface orientation. Occlusion and brightness and size clues can deliver information about discontinuities in depth. The main function is not only to provide information about depth, local surface orientation and discontinuities in the quantities, but also to create and maintain a global representation of depth that is consistent with the local cues that these sources provide.

Depth may be represented in a 2½-D sketch by a scalar quantity, the distance of a point on the surface from the viewer. Surface discontinuity may be represented by the oriented line elements. Surface orientation may be represented as a vector. The 2½-D sketch is useful because it makes explicit information about the image in a form that is closely matched to what the early visual process can deliver. The 2½-D sketch itself deals only with discovering the properties of surface in an image.

All the processes we have discussed are naturally retinocentric. There are several ways to represent surface orientation in a retinocentric coordinate frame. Information from different sources is probably checked for consistency and combined in some kind of retinocentric frame. Some conversion of the coordinate frame takes place at this point in order to express information from the different processes in a standard form. Such a conversion would:

(1) facilitate the computation of predicates like flat, convex or concave,

(2) allow easy comparison of the orientation of surfaces in different parts of the visual field, and

(3) prepare the way for the business of allowing for eye movements.

From a computational point of view, two problems need to be understood: the notion of discontinuity and possibilities for interpolation. Grimson (1979) explained in a dictum that "places of no information are actually places of information. In other words, one cannot hide discontinuities, and conversely, if the image provides no evidence at all about the presence of a discontinuity, not even an edge fragment anywhere along where one is a expected, then such a discontinuity may not be assumed" (ibid. p. 290). There are three principles of interpolation methods:

(1) linear interpolation in depth,

(2) linear interpolation in surface orientation, and

(3) 'fair surface' interpolation.

One final point that might be considered puzzling: stereopsis definitely assigns all edges to one plane, the figure should be seen in two dimensions and not in three. It is best to regard all contours in the 2½-D sketch as aiming for a three-dimensional interpretation.

Representing Shapes for Recognition

The knowledge which can derived from the representation of shapes for the recognition process could be very important for developing the conceptual model of our Design Tool. The main idea of the Design Tool is to perform different analyses – mostly from a cognitive mapping approach – of the input image by employing the possibility of directly taking data behind each object from the image scene. To develop such an aim on the conceptual level, all knowledge that arises from shape and object recognition will be welcome and useful.

A representation of shape is a formal scheme for describing shape or some aspect of shape together with rules that specify how the scheme is applied to any particular shape. Marr calls the result of using a representation to describe a given shape a description of shape in that representation. Formulating a completely general classification of shape is difficult. If the representation is to be used for recognition, the shape description must be unique. To be useful for recognition, the similarity between two shapes must be reflected in their descriptions, but at the same time even subtle differences must be expressible.

Marr considered three aspects of a representation's design (ibid. p. 298):

(1) the representation's coordinate system; in the case of our Design Tool, the geometry of objects from an input image scene will be directly linked to the global geographical coordinative system - via GIS data,

(2) its primitives; in the case of our Design Tool the objects from the input image will be treated as primitives during the process of analysis, and

(3) the organization that the representation imposes on the information in its descriptions; in our case, for each object of the input image the data behind will be directly available.

The most important aspect of the coordinate system is the way it is defined. If location is specified relative to the viewer, we say the representation uses a viewer-centered coordinate system. If location is specified in a coordinate system defined by the viewed object, the representation uses an object-centered coordinate system. There are several versions of each type. The coordinate system has to be identified from the image before the description is constructed.

The primitives of a representation are the most elementary units of shape information available in the representation, and are types of information that the representation receives from earlier visual processes. Two aspects of the representation's primitives are separate: the type of shape information they carry and their size. There are also two principal classes of shape primitives: surface-based (two-dimensional) and volumetric (three-dimensional). Volumetric primitives carry information about the spatial distribution of a shape. The third design dimension is the way shape information is organized by a representation. In the simplest case, no organization is imposed by the representation and all elements in a description have the same status. See also Figure 4.1 for an illustration of volumetric [tree-dimensional] primitives.

The spatial relations specified in a 3D model description are always local to one of its sub-models and should be given in a frame of reference determined by the model for the same reasons that we prefer an object-centered system over a viewer-centered one. The important characteristics of this model are (ibid. p. 306):

(1) each 3D model is a self-contained unit of shape information and has a limited complexity; in the case of our Design Tool each image scene contains several primitives with limited complexity,

(2) information appears in shape contexts appropriate for recognition; in the case of our Design Tool each primitive will be recognizable as an object with corresponding data, and

(3) the representation can be manipulated flexibly; in the case of our Design Tool each primitive will be a vector object with several possibilities of manipulation.

Each 3D model specifies the following (ibid. p. 305):

(1) a model axis is a primitive of the representation and provides information about characteristics such as size and orientation,

(2) optionally, the relative spatial arrangement and sizes of major component axes, and

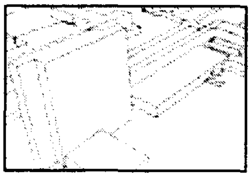
(3) the name of 3D models for the shape components associated with the

component axes.

The beginning of the examination of the 3D model representation is associated with identifying the model's coordinate system and its component axes and then transforming the viewer-centered axis specifications into the model's coordinate system. Then we treat the task of recognizing this description as a question of indexing into a catalog model of stored 3D model descriptions. Finally, we consider the interaction between the process that derives a 3D model description and the recognition process. To construct a 3D model, which is illustrated in Figure 4.3, the model's coordinate system and component axes must be identified from an image, and the arrangement of the component axes in that coordinate system must be specified. In the 3D model representation all axes' dispositions are specified by adjunct relations, so a mechanism is required for computing an adjunct relation from the specification of two axes in a viewer-centered coordination system. This mechanism is called an image space processor.

Recognition involves two things: a collection of stored 3D model descriptions and various indexes on the collection of them that allow a newly derived description to be associated with a description in the collection. This collection of the indexing is called the catalog of 3D models. Three access paths into the catalog appear: the specificity index, the adjunct index and the parent index. 3D models can be classified

Figure 4.3
The contour 3D model illustrates the result of the 'image space processor'



hierarchically according to the precision of the information they carry, and an index that can be based on this classification is called the specificity index.

Once a 3D model for a shape has been selected from the catalog, its adjunct relations provide access to 3D models for its components based on their locations, orientations and relative sizes. This gives us another access path to the models in the catalog, which we call the adjunct index. The adjunct index provides useful defaults for the shapes of the components of a shape prior to the derivation of 3D models for them from the image.

After a cataloged model is selected by using one of the three indexes, we then want to use it to improve the analysis of the image. There are two phases to this: first, the component axes from the image must be paired with the adjunct relations supplied by the catalog, and second, the image-space processor must be employed to combine the constraints available from the image with those provided by the model to produce a new set of derived adjunct relations that are more specific than those from the catalog model. This last phase involves an analysis of constraints that must be satisfied by adjunct relations consistent with both the image and the information from the catalog. The general idea of using a stored model of a shape to assist in the interpretation of an image was first used by Roberts (1965).

Correspondence between image and model increases as the derivation-recognition process proceeds. Initially, positional information along the principal axis of the stick figure has priority, since it is the least distorted by the perspective projection. Other clues available initially include (Marr, 1982 p. 322):

(1) the relative thickness of the shapes about the component axes,
(2) possible decomposition of component axes,
(3) symmetry or repetition, and
(4) large differences in the adjunct relations.

Once a homology has been established between a 3D model and the image, we want to use the information that it makes available to constrain the possible slant angles for the axes. The basic idea is that there are often only a few combinations of the slant specifications for the projected axes in the image for which the adjunct relations derived from the image would be consistent with those supplied by the catalog model. The sharpness of these constraints depends on the particular viewing angle and on the particular adjunct relations in the 3D model. Generally, the constraints are strongest when the component axes have very different orientations and when the principal axis does not lie in the image plane.

The overall recognition process may be summarized thus: first, we select the model from the catalog based on the distribution of components along the length of the principal axis. This model then provides relative orientation constraints that help to determine the absolute orientat ion of the component axes in the image, and with this information the image-space processor can be used to compute the relative lengths of the component axes. This new information can then be used to disambiguate shapes at the next level of the specificity index.

Marr's approach rested heavily on the principle of modularity (Marr, 1976), which states that any large computation should be split up into a collection of small, nearly independent, specialized sub-processes. Here, the analysis of vision was based on evidence from psychophysics and from everyday experience about what the modules were likely to be. The underlying argument is that if visual information processing is not organized in a modular way, incremental changes in its design - presumably an essential requirement for its evolutionary development - would be unable to improve one aspect of visual performance without simultaneously degrading the operation of many others.

It is clear that the brain must construct three-dimensional representations of objects and the space they occupy. Sutherland (1979) proposed at least two good reasons for this: firstly, in order to manipulate objects and avoid bumping into them, organisms must be able to perceive and represent the disposition of objects' surfaces in space, and secondly, in order to recognize an object by its shape, some kind of three-dimensional representation must be built from the image and matched in some way to a stored three-dimensional representation with which other knowledge is already associated. The two processes of construction and matching cannot be rigorously separated because a natural aspect of constructing a three-dimensional representation may include the continual consultation of an increasingly specific catalog of stored shapes.

High-level vision

As we proposed in the introduction to this chapter, here we will discuss high-level vision of object recognition drawing from the work of Ullman. We will also outline the importance of high-level vision for developing the conceptual model of our Design Tool. To start with we will quote Ullman's definitions of high-level vision and low-level vision in order to introduce the main terms, and then we will discuss his theory.

According to Ullman (1996 p. xii), high-level vision is concerned with the preliminary processing of the image, such as "extraction of shape properties and spatial relations, and with object recognition and classification.". It is concerned with the interpretation and use of information in the image and its use in the visual process.

By contrast, "low-level vision is usually associated with the extraction of certain physical properties of the visible environment, such as depth, three-dimensional (3-D) shape, object boundaries, or surface material properties" (ibid. p. xi). Low-level vision is an interpretation of what is seen in the image.

Ullman's theory of high-level vision

Vision is a highly complex process. Many different components, not always in interaction with each other, are involved. High-level vision deals wth some of the object components, such as shape, color, texture, depth, location or motion. Some of this visual information is used for processes of object recognition, locomotion or manipulation. Ullman's theory uses only the aspect of recognition in the area of visual cognition. This has to do with the perception of shape properties and their spatial relations as well as with object recognition and classification. This concentration of the information in the image, rather than the direct recovery of physical properties of the image is defined, as we mentioned above, as high-level vision. In the task of our research and Design Tool development the process of recognition can play an important role. We need a technique that can be relevant in the process of recognizing each single object from an image scene for different purposes, but mostly for manipulation, controlling and analysis of the visual quality of the urban environment by taking data 'behind' recognized objects directly from an input image. Another important purpose is the possibility of interaction between recognized objects with their surroundings in the case of comparison of all available objects and their data. To satisfy these purposes we first need to apply some computer-based object recognition techniques to our conceptual model of the Design Tool. In this case we need to discuss and apply Ullman's theory.

Object recognition

The process of object recognition and classification in the human brain, as well as in neurological and biological vision, is a spontaneous, natural activity. This process is crucial in human visual perception. The way in which the human brain perceives an object is already known, but the difference between brain and computer perception needs more explanation and is still not completely understood.

The main difference between the human brain and the computer is in their capacity in any form of performing. The brain never needs firm rules and instructions like a computer, the brain works automatically and spontaneously. On the other hand, a computer must be programmed to recognize some forms or objects with fixed and precisely defined shapes, positions and steps of instructions in processing. These are the big limitations of the machine, although they can be partly overcome.

Object recognition is the process of visual perception and "naming an object in sight" (Ullman, p.3). Naming is an invisible action, but it helps in the structuring process of identifying. Another similar action called recognition is also a process of simultaneous identification of an object as a member of certain classes, types or categories. An image is usually built up of many objects and each may belong to different classes. Each of the objects may contain numerous recognizable parts. To understand and analyze better the complex image of, for example, the urban environment, it is useful to focus the theoretical investigation on individual simple objects. But the process of recognition of a simple individual object is also a difficult task. This process is based on different sources of information. High-level vision deals with some of this information located in the object itself. The recognition of an object is based on its characteristic shapes, color and texture. The location of an object in relation to other objects in the image is also primarily characteristic of object recognition. Motion, with its characteristics, is another of the possibilities for recognition of an object. Some of the nonvisual characteristics, such as knowledge, expectation, temporal continuity or reasoning, can also indicate recognition of an object with its specific characteristics. Some objects could not even be recognized without their context and expectations.

In the process of image recognition, the human brain tries to match an image it sees to an image that it has already seen in the past. This associative memory is used as stored information of objects and a new image would be automatically processed as closely as possible to the image that resembles it. The problem of this approach is that simple image comparison is insufficient by itself to cope with a large variety of images from memory. According to Ullman (ibid. p. 6), the direct approach is insufficient for several reasons, but the main reason is that the image to be recognized will often not be sufficiently similar to stored images. There are several main causes for this: viewing position (direction and distances, such as different scales) is not similar; photometric effects (distribution of light sources in the object scene); object setting (object in isolation versus object as a part of a complex scene) and changing shape (a change of object shape causes a change in the expression of the object scene).

The idea of the direct approach is to compare the real object with the stored object but under different viewing conditions. To be able to compare two different but similar images, it is necessary to define a measure of their similarity. That can be done through the computation of some local distortion between images and compensation for changes in overall intensity level and linear intensity gradients; the measurement of differences is insensitive to global parameters.

The process of optimization of differences between two images to be compared is also important. Using various forms of linear filtering of real images is another type of difference measurement. Filtering takes place in the artificial image processing system. However, the object recognition system deals with these variations and it is not possible to use simple, straightforward object comparison.

Ullman (ibid. p. 14) distinguished the main general classes of object recognition, classified on the basis of the regularity problem, as: invariant properties methods, part and structural decomposition methods and alignment methods. These general classes could be implemented in the conceptual model of the Design Tool with their full functions or with some adaptation, which we will indicate in the review.

Invariant properties methods assume that objects have certain invariant properties that are common to all of their views. The invariant properties must prescribe effective procedures for extracting these properties. A broader use of 'invariants' in recognition includes computations that depend on both the input image and stored internal models, or more than a single input image.

Part and structural decomposition methods rely on the decomposition of an object into parts. Many objects are composed of their natural parts which can be recognized on their own. These parts as generic components of the object can be found first and then the main object can be composed of the identified parts. Generic components must also be stable so that the recognition process locates them as parts and classifies them into the different types of generic components and then describes the main object in terms of its constituent parts. This is many-to-one mapping object recognition which begins at the parts level. This approach has different 'feature hierarchies'. The structural decomposition invariant properties approach is defined using relations among parts. This method is usually described in a hierarchical graph structure, which can be easily matched with similar graph structures stored in memory. This method has become popular in recent years for visual object recognition.

The basic idea of the alignment approach is to compensate for the transformations separating the viewed object and the corresponding model stored in memory, and then to compare them. In this approach, transformations are explicitly applied to the incoming images or to the stored model.

The alignment approach

The alignment approach uses the fact that the sets of transformations of a rigid object are lawful and restricted. Transformations can be determined uniquely on the basis of very restricted information. First, the image of the input object establishes correspondence with the image of the stored model. The small number of features (points and lines) are identified as matching features in the object image and in the model from memory. Based on corresponding features, the transformation separating the stored model from the object image is uniquely determined. The recovered transformation is then applied to the stored model. The image generated (new image model) by the transformed stored model is then compared with the viewed object. This new candidate image model, depending on the degree of common matches, is accepted or rejected. To be accepted, the new image model must be sufficiently close to and better than the stored model.

The alignment method deals with 3D rigid object models and relies on corresponding features. For more complex objects, it deals with 2D object models and relies less on feature correspondence. The alignment of 3D rigid objects is more complicated than that of the 2D model. There are two methods for the alignment of 3D objects: the first is to maintain a single 3D model for each object and use the 3D transformations recovered in the alignment stage to transform the model into alignment with the viewed object. The second is to store a number of models and alignment keys associated with them representing the object from different viewing positions. In the first method, the model is object-centered and viewer independent, in the second the representation is viewer dependent since a number of different objects from different viewing positions will be used. Each of the methods consists of a set of contours and 3D coordinates which have to do with both recognition and alignment applied to 3D solids, rigid objects.

There are several methods for alignment recognition of solids that can be used. One of the methods is aligning pictorial descriptions; this uses an internal model for each object with their structural descriptions and contains a number of parts with their associated shape descriptions together with symbolic descriptions of the spatial relations among the parts. A replica of the object ('rooster' object) can be used as an internal model to align the model and the viewed object as precisely as possible.

In the case of our conceptual model of the Design Tool, we can propose an alignment method in which the image [image after the process of *recognition*] will correspond to the stored generic typology of 'microunits'. The generic typology of 'microunits' will have functionality equal to the image of the stored model explained above. The features of the image will be identified as matching features from the stored generic topology of 'microunits'. The reason for this new concept lies in the almost infinite number of different urban environment images with which the proposed Design Tool will work. It will be very difficult to predict, collect and store all images related to urban environments, which can build the stored model images. Instead we propose a *generic* solution for this. The concept of the generic typology of 'microunits' will be elaborated in Chapter 9, where we will develop the *conceptual model* of the Design Tool.

The curvature method is the next one that can be used in predicting the new appearance of an object with smooth surfaces. This method uses the 3D surface curvature along the object contours. The method is good for predicting the object's new appearances for a rotation about any given axis with a single parameter and magnitude of the curvature vectors. Three pictures are sufficient and five can be used to estimate the components independently. To cover all possible views of a given object, the object may be represented by a number of models, each covering a range of potential viewpoints.

Recognition by combining images

The main idea of this method, discussed by Ullman, is to use new schema stores and manipulate 2D views instead of explicit 3D models. The linear combination scheme assumes that the same object points are visible in different views. The variety of views depicting the same object under different transformations can often be expressed as linear combinations of small numbers of views. Two stored views of an object taken from different viewing directions, and a new, novel view will be given from some intermediate viewing direction. This novel object should be closer to each of the objects and it can be between the two views. Modeling a novel object depends on the object's shape and it is different for objects with sharp edges and objects with smooth boundary contours.

From the point of view of developing of our Design Tool, an important issue for the recognition process is to identify all objects from an image scene. Sometimes an input image is of very bad quality or the image is not complete, while in some cases only part of the object is visible, such as a tree or a figure standing in front of an object; so many other possibilities are evident in practice. To fully identify and recognize all objects from an image scene, and to solve such problems, the approach of multi-views recognition can be relevant.

The model of an object with sharp edges consists of a number of 2D views taken from different viewing positions. The views are in correspondence with points. Two views are sufficient to represent general linear transformations of the object and three views are enough to represent rigid transformations in 3D space. To represent an object from all possible viewing directions, a number of different models of this type will be required.

In the case of an object with smooth boundaries, translation, scaling and image rotation is similar to that of the case of an object with sharp edges. The difference is when an object rotates in 3D space. A method for predicting the appearance of an object following 3D rotations is called the curvature method. The model is represented in a set of 2D contours and each point along the contours is labeled with its depth value [z] and curvature value [r]. This method requires images to represent rotations around the fixed axis, and five images for general rotations in 3D space. Here the x and y coordinates follow rotation in space.

The linear combination of images uses feature correspondence to determine alignment coefficients applied to images of the 3D object. The general schema is the comparison of a viewed object with a single object model. The candidate model is selected from many objects stored in memory. The established correspondence between the viewed object and the stored model is called linear mapping. The term comes from the fact that the scheme is a map of views and it is linear in the mathematical sense (the 2D image views as input produce a new 2D image view as output). Any view of the image can be mapped.

The nonlinear combination of images is called the radial basis function (RBF) method. The method uses nonlinear interpolation between 2D images for the task of recognizing 3D objects by the superposition of basic functions centered on the known data points. This model is similar to the linear model, in that it relies on a limited number of 2D image views and recognizes novel views as a combination of stored patterns. The difference between the two methods is that the linear combination method produces a compensated or predicted internal model which is not generated by the RBF method.

The modeling of an object in combination with a stored object is not limited to rigid objects. Different non-rigid objects such as articulation and non-rigid stretching can be modeled this way. Recognition of these objects is obtained mainly through non-shape methods, including the use of color, texture and some abstract labels of the types.

The approach that an object is represented for the task of recognition by a number of its views is sometimes described as recognition by multiple views. In this approach, a number of object views are used collectively in the recognition process. This method also includes a known correspondence between individual views. The object views are not limited to simple images of the object. This method emphasizes the role of image-to-model correspondence as part of the recognition process, when the model is stored in pictorial form. Three corresponding views of the same object can be used to reconstruct the object's 3D shape. During the process of model construction, there is an opportunity to gather 3D information from multiple views and to use it in the recognition process.

Classification and identification

According to Ullman, the recognition of objects is considered at different levels of specificity and generalization, such as streets, canals, railroads, and parks. As the classification of the object is considered with specific shapes and transformations, the changes are easier to specify and compensate for. Recognition systems are usually more suitable for identification than for classification. It is easier to recognize known shapes than to capture the common characteristics of the class of objects.

A biological system of visual classification is a natural and spontaneous task. "For biological systems, classification is easier than identification, while the opposite holds true for artificial systems." (Ullman, 1996 p.160). Recognizing an object at the specific level is more informative than merely recognizing it. Object classification at different levels of generalization is useful in its own right, especially in dealing with novel objects and for the intermediate stage on the way to specific identification. Classification has an important function in the properties of both known and novel objects, based on the properties common to other objects in the same class. It is useful to be able to classify a novel object even if we have not seen it before.

Classification can aid identification in a number of ways. In the process from classification to identification, the system can use specific information associated with the class in question to aid the recognition process. In identification, the class-based process can play a useful role. Classification can often precede more specific identification in the chain of processing where a view of a familiar object is identified immediately without the intermediacy of the classification process. Classification can aid identification by reducing the amount of processing that needs to be performed when comparing an object with multiple internal models. In this prototype-based model, objects are grouped into classes. Each group of similar objects is represented by a single class prototype. The first step in recognizing a new input object consists of comparing it by class prototypes, rather than against the individual objects. The input object is matched against all the images stored in the database.

Some objects can be recognized on their own, or they can form parts of larger and more complex objects. Some objects are parts of complex context scenes, some are individually isolated. In dealing with a large number of objects in the models stored in memory, examination on a priority basis will increase the efficiency of the recognition process. The results indicate that the recognition of a first object help to recognize the next one. This recognition memory makes it possible to use the recognition of one object in a complex scene to help the recognition of related objects in the same scene. This process has an important role in the recognition of spatial scenes, such as an urban environment.

Urban environmental scenes are mostly complex, composed of several different objects. Some of the objects are simple, while some are complex, as in the example presented in Figure 4.1. In the case of complexity, classification and identification play significant roles. In our *conceptual model* of the Design Tool, each object or part of an object will be classified in a hierarchical structure of the urban environment. The hierarchical structure of the urban environment will be mirrored in the structuring of the object-oriented database system. We need this because of the concept of taking data directly from objects from an image scene. For this task of our Design Tool the relevance of classification and identification are evident.

Segmentation and saliency

To recognize an individual object in a complex scene, it is useful to separate the images of the object from the scene as much as possible. Segmentation is an approach that precedes the recognition process. Segmentation can isolate a sub-structure from the image that corresponds to a single object. Segmentation simplifies and reduces the recognition to single isolated objects. Segmentation includes a grouping process that starts from the image described in terms of local elements, and proceeds to identify a larger structure. In human vision, segmentation is an autonomous process.

Ullman proposed (ibid. p. 234) that there are two levels of segmentation: low-level and high-level. The low-level of segmentation is a process that performs grouping and segmentation operations on the basis of image properties such as proximity, collinearity, similarity of contrast, color, motion, texture and the like. The high-level in segmentation is a process that uses known object shapes to perform segmentation and grouping, by looking for a particular object in the image and using the identified object in the segmentation process. "The segmentation process in human vision appears to rely heavily on high-level process that employ knowledge about objects to be able to identify certain image structures as corresponding to a particular object" (ibid. p.235).

Saliency computation achieves two goals: first, it links local contour elements together to create a larger image structure - this process is called 'chunking', and second it produces a 'saliency map' of the image with local and global image properties, such as contrast, color, motion and disparity.

Visual cognition and visual routines

Ullman (ibid. p 263) defined *visual cognition* as "visual analysis of shape properties and spatial relations". Analysis of shape properties and spatial relations is not limited by the task of object recognition and classification. Visual analysis is also used in manipulating objects, planning and executing movements in the environment, selecting a path and so on. The visual extraction of spatial information is important and quite precise. 'Shape property' refers to spatial relations such as 'above', 'inside', 'longer-than', and so on. These relations are natural settings in the course of object manipulation, planning actions, reasoning about objects, and navigation. Visual aids such as diagrams, charts, sketches and maps can be used to help our reasoning and decision process. Spatial analysis of this kind does not require object recognition.

Ullman's (ibid. p. 266) idea of "visual routine assumes that the perception of shape properties and spatial relations is achieved by the application of the early visual representation." These visual routines of basic operations are 'wired into' the visual system. Routines are composed of the same set of basic operations using different sequences. The fixed set of basic operations of the visual system can assemble different routines. A visual routine is a process composed of a set of elemental operations to establish shape properties and spatial relations.

One of the basic operations is inside-outside perceiving. It assumes the identification of the bounding contour of an object, and features inside the contour belong to the object and outside features belong to the surroundings. The process underlying the perception of inside/outside is not yet well known. Shape perception and recognition is described in hierarchical 'feature detectors'. This hierarchical method of ray-intersection combines to produce higher order units lying inside or outside a closed curve. 'Coloring' is an alternative procedure that avoids some of the limitations in the ray-intersection method. Starting from a given point, the area around it in the internal representation is somehow activated. This activation spreads outward until a boundary is reached, but it is not allowed to cross the boundary. Depending on the starting point, either the inside or the outside of the curve, but not both, will be activated.

The concept of being 'inside' is abstract, because it does not refer to any particular shape, but can appear in many different forms. For a completely random set of shapes containing no regularities, simplified recognition procedures will not be possible. The minimal program required for the recognition of the set would in this case be essentially as large as the set itself. Closure is also an abstract property because it must be established by some process that makes use of general characteristics of closed curves. "The perception of abstract properties such as 'insideness' or closure would then be explained in terms of the computations employed by the visual system to capture the regularities underlying different properties and relations" (ibid. pp. 275-276).

In recognition, the process to use stored information can take different forms. Here we will explain bi-directional processing. 'Bottom-up' and 'top-down' processing are terms used in psychology of perception and in computer vision. "The distinction is that bottom-up processes are involved in the analysis of the incoming image, and top-down processes originate with stored models and information associated with them" (ibid. p. 320). In computer vision, the integration of bottom-up with top-down processing has been a major concern. In Ullman's theory, top-down processing is involved in dealing with the effects of viewing direction, facial expression and illumination. Bottom-up processing is used in the figure to handle variations due to position and scale. Bi-directional processing applied to the image is involved, for example, in compensating in differences and in scale. The search is bi-directional and a linking sequence is successfully established when the two streams of activation meet somewhere in this large network of interconnected patterns. In the *conceptual model* of the Design Tool the recognition process will be structured as an autonomous module in which bi-directional processing will play its functional role.

Recognition and visual routines are separate processes and both are important parts of high-level vision. To recognize a 3D object, we use a specialized recognition process that compares the current view with stored models. The specialized recognition method is useful for the specific task of object recognition, but is not suitable for general visual cognition tasks, such as the analysis of large-scale regional maps.

Some remarks about theories of cognition

An urban environment could be too complex. The structure of its representation in computer images is also complex and often contains too much information to be efficient in the process of recognition. The main problem is how to manage the vast amount of information in an efficient way for the task of recognition in our research and Design Tool.

As we mentioned earlier, there are three main prevailing theories of object recognition:

(1) the *invariant properties method* which leads to the notions of invariances, feature spaces, clustering and separation techniques;

(2) the *parts decomposition method* which leads to the notions of symbolic structural descriptions, feature hierarchies and syntactic pattern recognition; and

(3) the *alignment method*, the approach of which is to compensate for the transformations separating the viewed objects and the corresponding stored model and to then compare them.

The alignment method, discussed by Ullman, is a 'more correct' method that should be preferred under all circumstances. We can adopt this method in our research as the more adequate one for the task of the recognition of images taken from the urban environment. This method will be integrated in the *conceptual model* of the Design Tool in Chapter 9.

But still some questions remain. The main question in this field might be: what information should we try to recover from an image and then use to design an adequate representation expressing it? Or, do we need to use all information from an image and represent it? Or more concretely, if the objects are recognized from an image of the urban environment by analysis of their shapes and by the process of comparison, should the corresponding attribute data be added later in the computation process? We have to find adequate answers to these questions in the following chapters.

Conclusions

To establish the theoretical platform and knowledge necessary for developing the *conceptual model* of the proposed Design Tool we need to elaborate certain theoretical areas. One of these theoretical areas concerns cognitive science. We need to examine cognitive science from two aspects: computer cognition and object recognition. Marr's theory of computer vision [computer cognition] was discussed in the first part of the chapter. Ullman's object recognition theory, called high-level vision, was discussed in the second part of the chapter. Both theories give us theoretical as well as practical approaches to human and computer vision important for developing the Design Tool.

Vision entails recovering the structure and properties of the visual world. We focused here on the science of computational vision, the primary goal being the elucidation of fundamental concepts. Issues of algorithms and implementation and system architecture received little or none of our attention. Ullman's theory discusses high-level vision and provides the method for object recognition on scene recovery. Recognition is a cognitive process rather than a perceptual one. As we have seen, object recognition has important implications for our research.

We will adopt Ullman's method of high-level vision for the task of object recognition of input images of the urban environment. Effectuating recognition of all objects from an image scene will be the primarily function of the adopted method. The adopted method will be one of the first functional instances in the development of the *conceptual model* of the Design Tool.

The adopted method will also be very helpful in developing connections between the objects of an image scene with the corresponding data. If one object from an input image scene is recognized and selected through the adopted method, it will be the first step to establishing the connection to the corresponding data from the object-oriented database model. The conceptual models of the proposed processes will be developed in Chapters 7, 8 and 9.

In the next chapter we will present the main theoretical approaches to database systems. We will focus our investigation on object oriented database systems, which will be the main computational issue in developing the *conceptual model* of the Design Tool. Introducing the database systems will be the last theoretical examination in our research; all of these theoretical examinations [the theories of design, cognitive science and database systems] are necessary for understanding and developing the *conceptual model* of our Design Tool.

CHAPTER 5

THEORIES OF DATABASE SYSTEMS - OBJECT ORIENTED

***Task of the chapter**: the 'object-oriented database system' will be the main information theory and method which will support our Design Tool and which can represent a large amount of data for any urban environment.*

Introduction

In the previous two chapters we presented the theoretical platform of the research related to urban design theory and cognitive maps, and theories of cognition and computer vision. The theories presented together with the theories of database systems, which will be elaborated in this chapter, will propose the theoretical background and framework for this research and for developing the Design Tool. In this chapter we will provide an overview of different database systems and we will adopt one of the object-oriented database technologies for the task of developing our Design Tool. In this chapter we will elaborate the recent improvement in database technology, focusing the investigation on contemporary theoretical sources.

One kind of increasing progress that has enjoyed the benefits of computing technology in the last three decades has been *information system technology*. An information system is a collection of activities that regulate the sharing and distribution of information and the storage of data.

Information is an abstract commodity; you cannot touch information, but you can gain information by touching something. Database technology is concerned with one particular form of information representation within a computer system, i.e. data. Data consists of symbols written on some recording medium, such as paper, magnetic disk, or CD-ROM, to represent facts, concepts, or instructions in a formal manner so that they can be interpreted or processed by humans or machines. *Data* are known facts that can be recorded and that have implicit meaning, such as names, telephone numbers and addresses in an address book. "A *database* is a collection of related data. By data, we mean known facts that can be recorded and that have implicit meaning." (Elmasri & Navathe, 2000 p. 4). A database represents certain aspects of the real world (*miniworld*), sometimes called the *Universe of Discourse* (UoD). Changes to the UoD are always reflected in the database. A database is designed, built and populated with data for a specific task and intended group of users.

A database is made to provide an organization with the information necessary for it to carry out its activities. Database systems are used in all walks of life including industry, business, the armed forces and emergency services, education, government and scientific research. In our research, the database system is a crucial part of the tool design. The activities and organization of an urban environment do not operate at random or in isolation. They must be coordinated by a flow of information within the environment and between the environment and the outside world.

Database systems have traditionally been used to represent information as conventional data such as numbers and text. Nowadays, database systems support the representation of many different forms of information, such as sounds and images as well as numbers and text. This multimedia form of data representation might also be static or dynamic (photos or video, for example).

The first generations of data models (until *relational data*) were focused mainly on representing the meaning of data in terms of the structure of the information represented. The third generation of data models in addition represented the meaning of data in terms of the behavior associated with the information represented. (Elmasri & Navathe, 2000). In this chapter we will explain more of the mainstreams of data with a focus on *object-oriented database systems*. At the end of this chapter we will propose Perspective-DB technology, which will be adopted for developing the object-oriented data model for our Design Tool.

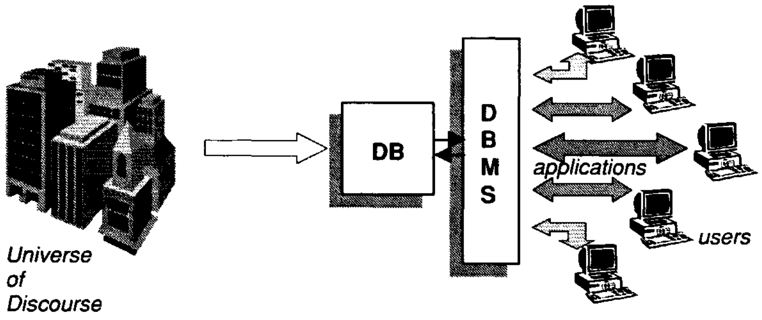
About database technology in general

Database technology plays an important role in today's increasingly complex software systems. Databases and database technology play an important role in almost all areas of our use of computers, including the fields of business, processing technology, engineering, education, civil service, law, medicine, science, and so on. They have become integrated components of our wired life and modern society.

Database and database technology help people to use and process information via complex computer applications. Data are presented in a database system [DB]. A database system functions through a database management system [DBMS]. The database management system is generic and it defines, constructs and manipulates databases for various applications. The data are mostly used for different tasks and by various groups of users [data sharing]. Figure 5.1 shows the function of the use of a database system.

Usually, databases are in use for a long period of time and for privileged access. Some databases are for wide users, especially with the launch of Internet search engine technologies, whereby data are circulated freely.

Figure 5.1
The schematical function of the use of a database system



About data

A database is a collection of related data. A database is a collection of data that represents important objects in a user's business. A database represents information on a specific part of reality and it is a *model* of this specific part of reality. A fact of reality that we would like to be recorded in the database is a piece of *data*. That fact has implicit meaning. The collection of related data from the real world with implicit meaning is thus a *database*.

As we see, one particular part of the real world presented in a database is the Universe of Discourse. Changes in the Universe of Discourse ought to be reflected in a database. A database is a model of a specific part of the real world (reality); it can be a world of ideas, notions, processes, plans, etc. These real and ideal aspects of reality, the Universe of Discourse, are explained by the following notions, which are about the Universe of Discourse:

- identified, limited elements that represent information about the Universe of Discourse,
- interrelations between these elements.
- desirable actions with these elements, and
- conditions by which these elements could be explained.

An *entity* is called a limited element due to its identity. One entity is different from other entities because it is one recognizable part of the Universe of Discourse. An entity serves as information about the Universe of Discourse and presents some of its aspects.

If the database presents a model of the Universe of Discourse, then this model must be specified and explained. This process is called the *designing* of the database.

The database has some source from which its data are derived, some degree of interaction with the real world and an audience that is actively interested in the content of the database. A random assortment of data cannot correctly be referred to as a database. A database is created, built and populated with data for a specific task and for an intended group of users. The creation of a database is usually called database designing, and this process consists of many steps. Although different database designers may use different methods, the process generally follows a basic pattern of data modeling, a description of the data to the database, and physical implementation of the database. This basic process of database designing can be broken down into five phases (see Table 5.1):

(1) planning

(2) analysis
(3) design
(4) implementation
(5) maintenance

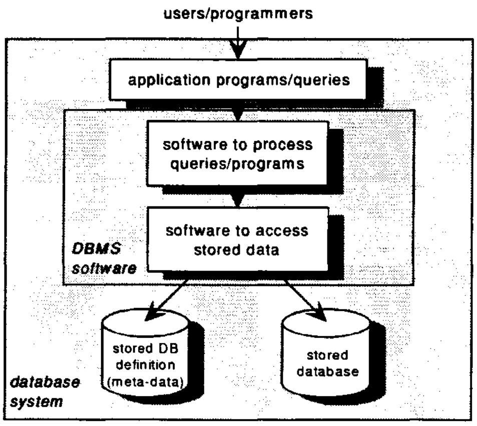
Table 5.1
Five phases of database design with their purposes and some tools and/or techniques of each

PHASE	*PURPOSE*	*TOOLS & TECHNIQUES*
PLANNING	Define what to accomplish	Focus list Issue list Goals
ANALYSIS	Define objects of importance	Business models Diagrams Logical data models
DESIGN	Define how the database is built	Tables of files Structure Constraints Physical models
IMPLEMENTATION	Define the database to the DBMS	DBMS program Application design Views, queries, reports SQL
MAINTENANCE	Manage and control of the database	Data dictionary Security functions Structure management Documentation

These phases often overlap and some techniques and tools may be used in more than one stage, especially between analysis and design. For example, although the data dictionary, which is used to indicate a more general software utility and to maintain information on the system hardware and software configurations, documentation, applications and users, is usually introduced in the analysis stage, it is often carried over into the design phase and later used by the database administrator in the maintenance stage. Database design is nondeterministic; in other words, there is no "right" design. The true goal of database design is to create a well-structured database that represents the user's perspective of the business and provides the user with a productive tool. A database can be complex and of any size. A database may be generated and maintained manually or computerized. A computerized database may be created and maintained by application programs created for a special task or by a database management system.

A database management system [DBMS] is a collection of programs that enables users to create and maintain databases. The DBMS is hence a general-task software system that facilitates the process of defining, constructing and manipulating databases for various applications. One DBMS creates possibilities for more users to simultaneously use different operations of the same data from the database (data concurrency). The DBMS can be classified as a *general-task DBMS*, which facilitates a disposable wide application domain (its functioning is not optimal in all application domains), or a *special-task DBMS*, which facilitates optimal functioning in a specific application domain (for example, CAD applications).
The database and DBMS together are called the *database system*, see Figure 5.2 (Elmasri and Navathe, 2000).

Figure 5.2
A simplified database system environment
[from: Elmasri and Navathe, 2000]


users/programmers
application programs/queries
software to process queries/programs
software to access stored data
DBMS software
stored DB definition (meta-data)
stored database
database system

Traditional database systems have most of the information stored and accessed in textual or numerical form. *Multimedia database systems* can store and access different – mostly visual – data such as images, video or sound. There are also several specific database systems, such as the *GIS* (geographical information systems) *database system*, which plays an important role in our research, and can store and access spatial data from maps and satellites.

Generic database properties/characteristics

A database should be a natural representation of information as data, with few imposed restrictions, capable of being used by all relevant applications without duplication. Information may be thought of as being about interrelated entities.

The database approach is to store information as data in such a way that the data are structured to represent the entities that are linked in the same way to represent the relationships. The database approach is centered on the data, rather than on the process applied to the data.

The database approach to computerized information activities is to treat the organisation's data, i.e., its database, as a resource that is shared by all relevant application programs. The data are made shareable by structuring it in such a way that it reflects the structure of the information it represents. Database application programs can be programmed to access the logical structures of the information that is processed, rather than the structure of the files that are used to store the data that represents it. This means that programs that require access to the same information can access the same parts of the database.

However, a database system contains not only the database itself but also a complete definition or description of the database structure and constraints. This definition is stored in the system *catalog*, which contains information such as the structure of each file, the type of storage format of each data item and different constraints of data. The information stored in the catalog is called *meta-data*, which describes the structure of the primary database (see also Figure 5.2).

The extent to which a database and the programs that use it are maintained as independent resources depends on the nature of the organization that the database is designed to serve. The database is embedded within the system and is only accessible by the operating system.

The database may also be designed to store data for a wide range of applications. The database exists as a resource in its own right. The DBMS software must work equally well with any number of data applications as long as the database definition is stored in the catalog.

In contrast to traditional file processing, each user defines and implements the files needed for a specific application as part of programming the application. Here, data definition is typically part of the application programs themselves. The programs are constrained to work with only one specific data file, whose structure is declared in the application programs.

File-processing software can access only specific data files. DBMS software can access diverse databases by extracting the database definitions from the catalog and then using these definitions. The DBMS allows multiple users to access the database at the same time and the DBMS includes concurrency control software to ensure that several users trying to update the same data do so in a controlled manner, so that the result of the updates is correct.

Data models

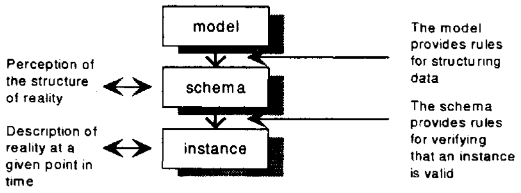
To understand and use information from the Universe of Discourse [UoD], the database approach provides a level of data abstraction by hiding details of data storage. A data model provides the necessary means to achieve this abstraction. A data model is also a collection of concepts that can be used to describe the structure, operations and conditions of the database. The structure of the database contains the data types, relationships and constraints that should apply to the data and basic operations for retrievals and updates of the database.

We categorized data models according to the types of concepts they use to describe the database structure. There are three types of data models: *conceptual* or high-level data models, *logical* data models and *physical* or low-level data models.

Conceptual data models "provide concepts that are close to the way many users perceive data." (Elmasri & Navathe, 2000 p. 24). *Logical* data models provide different classes for implementation in database management systems. *Physical* data models provide concepts that describe the details of how data are stored in the computer. The models have to be mapped to each other. See Figure 5.4 for an example.

In the design of the database, conceptual models are used first to produce a high-level description of reality, then to translate the conceptual schema into a logical schema. A *schema* is a representation of a specific portion of reality. A schema, as we explained earlier, is a static, time-invariant collection of linguistic or graphic representations that describe the structure of interest. An *instance* of a schema is a dynamic, time-variant collection of data that conforms to the structure of data defined by the schema. Each schema can have multiple instances. The evolution of the database can be seen as the transition from one schema instance to another schema instance caused by some data modification operation. Again, see Figure 5.4 for example.
The relationship between model, schema and instance is presented in Figure 5.3 (Batini, Ceri and Navathe, 1992).

Figure 5.3
The relationship between model, schema and instance
[from: Batini, Ceri and Navathe, 1992]


model
schema
instance
Perception of the structure of reality
Description of reality at a given point in time
The model provides rules for structuring data
The schema provides rules for verifying that an instance is valid

In any data model, it is important to distinguish between a *description* of the database and the *database itself*. The description of the database is the *database schema*, part of the data dictionary. A database schema is an extension of the database which 'understands' the data. The data modeling process is shown in Figure 5.4. Database schemas are displayed by diagrams called *schema diagrams*.

Data models are used to build schemas that are representations of the Universe of Discourse. The quality of resulting schemas depends on the quality of the selected data model. The schemas should represent the UoD's semantics as well as possible. The main elements of all data models as small collections of primitive abstraction mechanisms are:

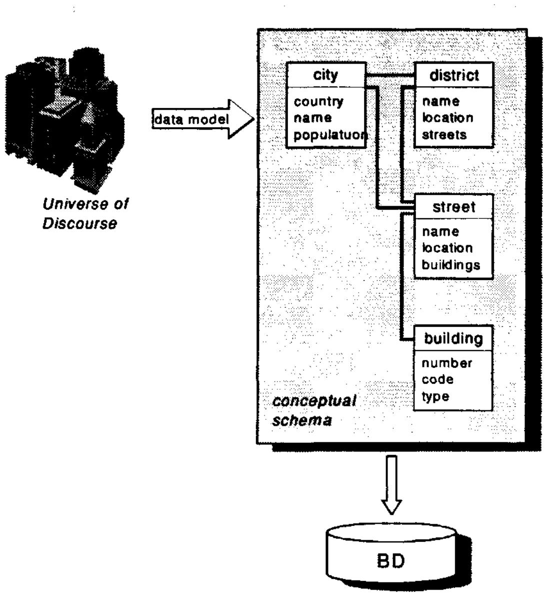
(1) *Typologisation*, used for defining one concept as a *type* of the UoD's entities characterized by common properties (such as city and streets);

(2) *aggregation* defines a new type from a set of other types that represent its component parts; the aggregation is represented by a one-level tree in which all nodes are types (such as a district in a city); and

(3) generalization defines a subset of relationships between the elements of two or more types; the generalization is also represented by a one-level tree in which all nodes are types; all the abstractions defined for the generic type are inherited by all the subset types (such as street in a city, or street in a district, or building in a street).

See also Figure 5.4 for example.

Figure 5.4
Data modeling process



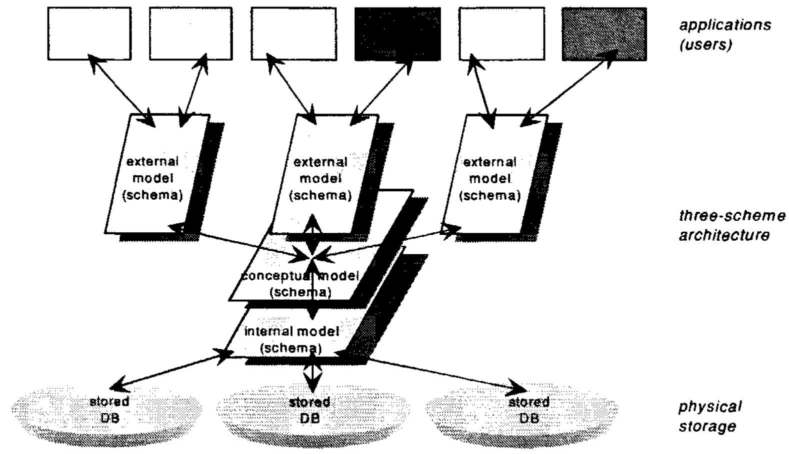
The basic structure/architecture of the system

The important characteristics of the database approach are: program-data and program-operation independence, support of multiple user views and use of the catalog to store the database description (schema). To help achieve and visualize these characteristics, *three-schema architecture* for database systems was proposed.

Three-schema architecture was developed for The Standard Plannings and Requirements Committee [SPARC] and The American National Standards Institute [ANSI] by the Study Group on Database Management Systems in 1972. The goal of three-schema architecture "is to separate the user applications and the physical database." (Elmasri & Navathe, 2000 p. 27).

Three-schema architecture is organized in three levels of data description, presented in Figure 5.5 (Eaglestone and Ridley, 1998, p. 29):

Figure 5.5
The Three-schema Architecture [ANSI/SPARC]
[from: Eaglestone and Ridley, 1998]



(1) *external,*
(2) *conceptual (logical),* and
(3) *internal.*

The *external level* describes the viewpoint of specific groups of users of the database and also application programs; it presents the information that is relevant to a particular user group. An implementation data model can be used at this level. The main task of three-schema architecture is separation of application programs.

The *conceptual (or logical) level* provides a machine-independent, high-level representation of the whole database. This level (schema) hides the details of physical storage structures and concentrates on describing entities, user operations and constraints. A high-level data model or implementation data model can be used at this level.

The *internal level* provides a machine-dependent description of the physical storage structure implementation of the database. The internal schema uses a physical data model and describes the details of data storage and access paths for the database.

The ranges of the separation are:

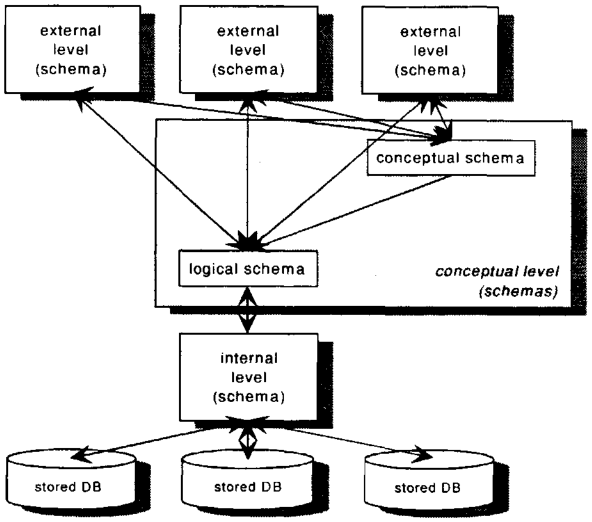
- logical and physical *data independence* - changing of conceptual schema
  without changing the application programs or databases,
- *multi-user* systems or *data sharing* - provides support of different application programs by one database, and
- using one *dictionary* for describing the database structure.

It is possible to add at least two levels in a conceptual schema in three-schema architecture. The reason for this can be found in new data models (*relational data model* and *entity-relationship model*) and their characteristics. There are two interpretations of conceptual database schemas:

- *conceptual schema*, and
- *abstraction* in the form of separation into one conceptual and at least one logical scheme.

A *logical schema* is a description of a database that can be used by one particular class of DBMS [not for one particular product]. The logical schema describes the database as the realization of the data model of one particular class of DBMS. The extended three-schema architecture is presented in Figure 5.6.

Figure 5.6
Extended three-schema architecture
[from: Eaglestone and Ridley, 1998]



In the context of three-schema architecture, conceptual models are languages for implementing the conceptual level. Conceptual models should be good tools for representing the UoD and they should possess the following qualities:

(1) *expressiveness;* they constitute a large variety in concept, therefore models that are rich in concepts are also very expressive;

(2) *simplicity;* to easily understand a conceptual model, for the designers and users of the database application;

(3) *minimality* is achieved if every concept present in the model has a distinct meaning with respect to every other concept; and

(4) *formality* requires that all concepts of the model have a unique, precise and well-defined interpretation.

Some models are not able to express all the properties of a given UoD. In general, the data model must be able to express different needs. The best example for understanding the conceptual model will be presented in Chapter 8 in our database model.

Three-schema architecture is only *descriptions* of data; the only data that *actually* exists is at the physical level. The process of transforming requests and results between levels in three-schema architecture is called *mapping*.

Data independence is explained by three-schema architecture as the capacity to change the schema at one level of a database system without having to change the schema at the next level. There are two types of data independence (Elmasri & Navathe, 2000 p. 29):

(1) *logical data independence* is the capacity to change the conceptual schema without having to change external schemas or application programs, and

(2) *physical data independence* is the capacity to change the internal schema without having to change conceptual (external) schemas.

Three-schema architecture makes it easier to achieve true data independence, both physical and conceptual.

As we have already said, data models are usually described through both kinds of representation: linguistically and graphically. The success of the model is often correlated with success in its graphic representation, which should have the following qualities:

(1) *graphic completeness:* a model is graphically complete if all its concepts have a graphic representation, and

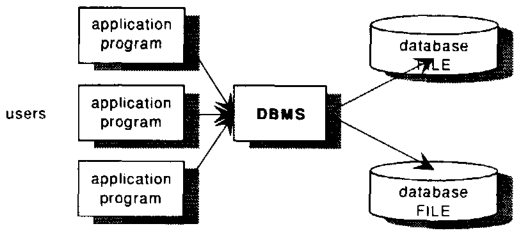
(2) *ease of reading* of the model, if all concepts are represented by graphic symbols that are clearly distinguishable from all other graphic symbols.

The best example for understanding the graphic representation of the data model will be presented in Chapter 8 in our database model.

The Database Management Systems

"A database management system [DBMS] is a collection of programs that enables users to create and maintain a database." (Elmasri & Navathe, 2000 p. 5). A DBMS is a complex application developed to create, manage and use a database. Data are stored as records in various database files that can be combined to produce meaningful information for users (see Figure 5.7). The DBMS controls all functions of capturing, processing, storing and

Figure 5.7
Relationships between users, DBMS and database files



retrieving data from databases and generates various forms of data output. The DBMS can contain hundreds of applications and files. Modeling users' data, as opposed to application programs, allows the definition of data objects that are important to the programs. These data objects are more stable and less likely to change than application programs. Because the representations of the data are separate from the physical implementation and access functions, the relationships between the data files are more apparent. Therefore, DBMS's have more flexibility than file processing systems and require less programming maintenance.

The collection of data managed by a DBMS is called a database and the database and DBMS together with the applications programs that use the database are collectively called the database system.

The database system covers many areas, including data types, data semantics, DBMS software, and physical file design. In studying each area individually, one might not grasp the relationships between them. However, it is important to understand how the areas are related in order to provide information and applications that fulfill user needs. The primary task of a DBMS is to provide the user with a workable, meaningful and relevant system.

The main characteristics of a DBMS, according to Eaglestone and Ridley (1998), are:

(1) it is a *computerized record-keeping system*,

(2) it *contains facilities* that allow the user to:

- *add* and *delete files* in the system, and
- *insert, retrieve, update* and *delete data* in existing files,

(3) it is *collection of databases* - a DBMS may contain many databases that can be used for separate tasks or combined to provide useful information.

The crucial functions of a DBMS defined also by Eaglestone and Ridley are:

(1) to *store data*,

(2) to *organize data*,

(3) to *control access to data*, and

(4) to *protect data*.

The main uses of a DBMS are:

(1) to *provide decision support*,

(2) managers and analysts *retrieve information* generated by the DBMS for inquiry, analysis, and decision-making,

(3) to *provide transaction processing*, and

(4) users *input*, *update*, and *process data* with transactions that generate information needed by managers and other users or by other departments.

A DBMS is a complex software system that consists of many different subsystems. The basic components of a DBMS can be divided into five subsystems:

(1) *data control*,

(2) *tool for database administration*,

(3) *end-user services*,

(4) *core database manager*, and

(5) *data-communication manager*.

The schematical relationship between the DBMS components, defined by Eaglestone and Ridley (1998), is presented in Figure 5.8.

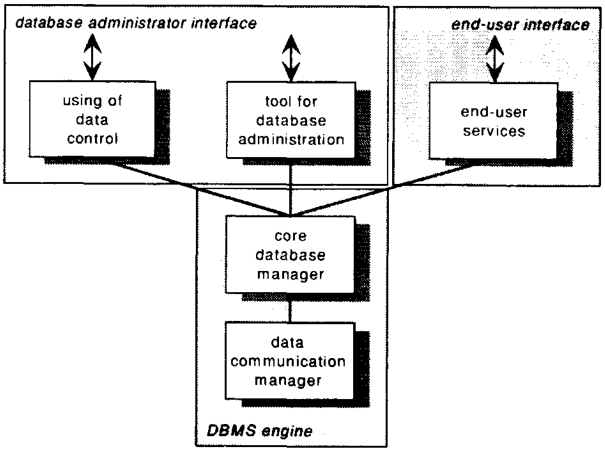
Most DBMS's have database utilities that help the database administrator, the specialist who manages the database system. Common utilities have the following types of functions: loading, backup, file reorganization and performance monitoring.

The criteria used for classification of the DBMS (as defined by Elmasri & Navathe, 2000 pp. 35-36) are:

(1) the *data model* on which the DBMS is based [some models: *relational, object, object-relational, hierarchical and network*],

(2) *number of users* supported by the system [*single-user systems* or *multi-user systems*],

Figure 5.8
The components of the DBMS
[From: Eaglestone and Ridley (1998)]


database administrator interface
end-user interface
using of data control
tool for database administration
end-user services
core database manager
data communication manager
DBMS engine

(3) *number of sites*[1] over which the database is distributed [*homogenous* or *heterogeneous*],
(4) *cost* of the DBMS [high or low],
(5) *types of access path*[2] options for storing files, and
(6) *general task* or *special-task*.

The basic functions that characterize a DBMS are:
(1) *data manipulation*
(2) *structure definition*
(3) *immunity of database* - requirements of database immunity:
- *recovery* (protection of physical destruction),
- *transaction management*,
- *integrity marking*,
- *concurrency*, and
- *data protection*

(4) *multi-user working*
(5) *performance control*, and
(6) *realization of the user interfaces*

[1] *Site* is differs from, for example, 'urban site', and has the meaning of *computer site*.
[2] *Path* differs from, for example, 'urban path', and has the meaning of *path options for storing files*.

A DBMS overcomes limitations of *file systems*, described by Eaglestone and Ridley (1998), in the following ways:

(1) *it eliminates separation and isolation of data* - in a DBMS, all data are stored in the database. The DBMS allows for complex relationships between data files, providing efficient data integration. Programmers specify how the data are to be combined; the DBMS performs the functions to provide the information;

(2) *it reduces data redundancy* - minimal duplication of data provides more storage space in the system and allows for efficient updates when changes in records occur, maintaining data integrity. For example, if a customer's address is stored in only one file and a change is made to that file, all affected applications are updated automatically;

(3) *it eliminates dependence between programs and data* - in a DBMS, record formats are defined in the database and accessed by the DBMS. Application programs only define which data items are needed, not data formats. Therefore, format changes are made to the database by the DBMS, and, since most fields are stored in only one file, the changes are made only to the affected file. Also, because structure formats and access methods are independent of the database, there is little impact on application programs if structure/access changes are made;

(4) *it allows for representation of data from the user's view* - relationships between data objects in a user's environment are stored in the DBMS. Data elements from any number of files can be combined to create useful forms, reports, and other applications; and

(5) *it increases data flexibility* - because of the data-independent structure of the DBMS, applications can be created without intensive programming. Thus, ad hoc requests for information and applications can be filled with less time and labor from programmers than with file processing systems.

As general remarks regarding DBMS's, we will present some possible advantages and disadvantages:

(1) *Advantages:*

- *centralized data reduces management problems,*
- *data redundancy and consistency are controllable,*
- *program - data interdependency is diminished,*

- *flexibility of data is increased.*

(2) *Disadvantages:*

- *reduction in speed of data access time,*
- *requires special knowledge,*
- *possible dependency of application programs to specific DBMS versions.*

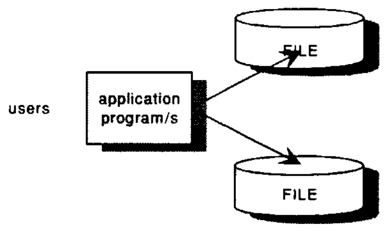
Generations of data models

The last three decades have been important for the tremendous growth of database development. The database has become an essential component in information technologies routinely used in all computers, ranging from large to personal. Databases first entered the software markets at the end of the 1960's. These systems were limited, rough tools: block diagrams and record structures were the common forms of specifications. Database design was often confused with database implementation. In the meantime, database technology has changed and developed; see Eaglestone and Ridley (1998). A new theoretical framework was given, database design methods and models have evolved, including relational theory of data, query processing and optimization, concurrency control, transaction and recovery management and so on. Here, we will explain the most relevant concept in database systems and their advantages and disadvantages. We will briefly explain the main generations of data models: *file systems, hierarchical systems, network systems* and *relational systems. An object-oriented system* is a crucial part of our Design Tool, and for this reason we will explain its principles and techniques in more detail in the following section of this chapter.

File systems

In early processing systems, an organization's information was stored as groups of records in separate files. These file-processing systems consisted of a few data files and many application programs (see Figure 5.9). Each file, called a flat file, contained and processed information for one specific function, such as accounting or inventory keeping. Programmers wrote

Figure 5.9
File processing system



applications that directly accessed flat files to perform data management services and provide information for users.

Originally, most organizations managed their operations with the help of a manual file system, usually consisting of a collection of file folders. The contents within each file folder were logically related and as long as the information needs and reporting responsibilities of the organization were relatively simple, this system worked (and still does work) quite well.

As information needs grow and become more complex, the manual system often becomes too cumbersome and timely information retrieval virtually impossible. Early computerized file systems simply mirrored manual systems, with individual files for different kinds of information. Because information could not be shared across files, there was a great deal of data redundancy, which often leads to a higher level of data inconsistency and data anomalies. Programmers would write a program to generate even the most rudimentary reports and as reporting requirements grew, so did the level and complexity of programming. Programmers used a third-generation language [such as COBOL, BASIC, FORTRAN] to tell the system both what to do and how to do it.

The main drawbacks to data management with the file system are *structural dependence* and *data dependence*. *Structural dependence* means that a change in the file structure, no matter how small, forces modifications in all the programs that use the data in that file. *Data dependence* means that any time you make changes in the characteristics of the data, all programs that access that data must also be changed. Because of these constraints, as well as the fact that programming is time-consuming, getting a quick result for a new information need is virtually impossible.

In creating the files and applications, developers focused on user processes, or how these processes ware transacted, and their interactions. However, user processes are dynamic, requiring continuous changes in files and applications. In addition, early programmers focused on physical implementation and access procedures when designing a database. These physical procedures were written into database applications; therefore, physical changes resulted in intensive rework on the part of the programmer. As systems became more complex, file-processing systems offered little flexibility, presented many limitations, and were difficult to maintain.

As we see in Figure 5.9, the file systems focus on application programs. According to Eaglestone and Ridley (1998), each file must have its own file management system, composed of programs that allow the user to:

(1) *create the file structure,*

(2) *add data to the file,*

(3) *delete data from the file,*

(4) *modify the data contained in the file*, and

(5) *list the file contents.*

Changing just one field in the original file requires the use of a program that:

(1) puts the new file structure into a special portion of the computer's memory [buffer],

(2) opens the original file, using different buffers,

(3) reads records from the original file,

(4) transforms the original data to conform to the new structure, using complex string manipulations, and

(5) writes the transformed data into the new file structure.

Limitations of File Systems

These are some evident limitations in file systems:

(1) *Separated and isolated data* - to make a decision, a user might need data from two separate files. First, the files were evaluated by analysts and programmers to determine the specific data required from each file and the relationships between the data. Then applications could be written in a third generation language to process and extract the required data. Imagine the work involved if data from several files was needed.

(2) *Data redundancy* - often, the same information was stored in more than one file. In addition to taking up more file space on the system, this replication of data caused loss of data integrity. For instance, if a customer's address was stored in four different files,

an address change would have to be updated in each file separately. If a user was not consistent in updating all files, no one would know which information was correct.

(3) *Program - data interdependence involving file formats and access techniques* - in file processing systems, files and records were described by specific physical formats that were coded into the application program by programmers. If the format of a certain record was changed, the code in each file containing that format had to be updated. For example, a field in the sales file might be coded as "decimal", while the same field in the customer file could be coded as "binary". In order to combine these fields into one application, a programmer would have to write code to convert every value of the "decimal" field in the sales file to a "binary" field (or the reverse) in addition to coding the application. Furthermore, instructions for data storage and access were written into the application's code. Therefore, changes in storage structure or access methods could greatly affect the processing or results of an application.

(4) *Difficulty in representing data from the user's view* - to create useful applications for the user, often data from various files must be combined. In file processing, it was difficult to determine relationships between isolated data in order to meet user requirements.

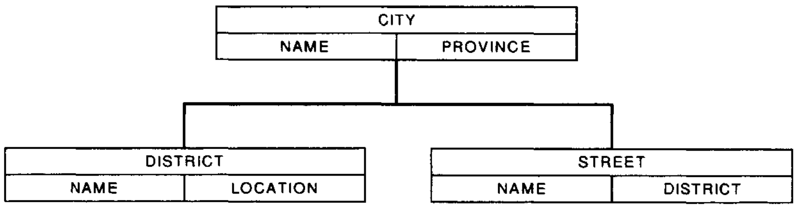
(5) *Data inflexibility* - program-data interdependency limited the flexibility of file processing systems in providing users with ad hoc information requests. Because designing applications was so programming-intensive, information requests were usually restricted by MIS department staff. Therefore, users often resorted to manual methods to obtain the information they required.

Hierarchical systems

Hierarchical database systems are well suited to those information systems that can be based naturally on the hierarchical model. Around 1965, IBM and North American Rockwell implemented a hierarchical data model in a join effort. A hierarchical model employs two main data structuring concepts: *records* and a *parent-child relationship* [PCR]. A record is a collection of *field values* that provide information on an entity or relationship instance. Records on the same type are grouped into *record types*. A record type is given a name and its structure is defined by a collection of named *fields* or *data items*. Each field has a certain data type, such as integer, real or string. A parent-child relationship type is a 1:N relationship between two record types. The record type on the 1-side is called the *parent record type*, and the one on the N-side is called the *child record type* of the PCR type.

A hierarchical data schema consists of a number of hierarchical schemas. Each hierarchical schema consists of a number of record types and PCR types. A hierarchical schema is displayed as a hierarchical diagram, in which record type names are displayed in rectangular boxes and PCR types are displayed as lines connecting the parent record type to the child record type. Figure 5.10 shows a simple hierarchical schema with three record types and two PCR types.

Figure 5.10
The hierarchical schema

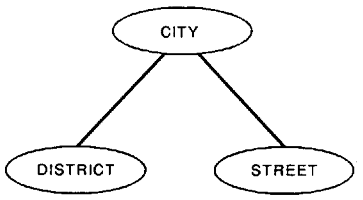


The definition of a hierarchical schema defines a *tree data structure*. In tree data structures, a record type corresponds to a *node* of the tree and PCR type to the *edge* of the tree. Usually in a hierarchical tree data schema, each tree edge is shown separately from other edges (Figure 5.11).

Generally, "there are many feasible methods of designing a database using the hierarchical model. In many cases, performance considerations are the most important factor in choosing one hierarchical database model over another." (Elmasri & Navathe, 2000 p. 948).

A hierarchical DBMS is obsolete for new applications. It may be encountered with legacy applications.

Figure 5.11
A tree representation of the hierarchical schema



There are some important *advantages* of the hierarchical database models:

(1) It is simple to construct and operate,

(2) it corresponds to a number of natural hierarchically organized domains - e.g. assemblies in manufacturing, personnel organization in companies, and so on.

(3) the language used is simple, with constructions like GET, GET UNIQUE, GET NEXT, GET NEXT WITHIN PARENT, etc.

There are also some *disadvantages* of the hierarchical database models:

(1) the navigational and procedural nature of processing,

(2) the database is visualized as a linear arrangement of records,

(3) inflexible for the majority of database applications,

(4) too much redundancy and inconsistencies, and

(5) little scope for 'query optimization'.

Network systems

Honeywell implemented a network database model for the first time in 1964-5 with a system called the IDS system, which was adopted due to the support by CODASYL [DBTG model]. The original CODASYL/DBTG report used COBOL as the host language. The network model and the object-oriented data model are both navigational in nature. The data structuring capability of the network model is much more elaborate and allows for explicit insertion/modification of semantic specification.

There are two basic data structures in the network model: *records* and *links*. A link is a set of physical pointers that establish an 'ownership' between one set of records and another. Data are stored in *records*. Each record consists of a group of related data values. Records are classified into *record types*, whereby each record type describes the structure of a group of records that store the same type of information. Each record type has a name, and for each *data item* (attribute) a name and format (data type) are also given.

A typical database application has numerous record types. To represent the relationship between records, the network model provides the modeling construct called the *set type*. A set type is a description of a 1:N relationship between two record types. Figure 5.12 shows how a set type is represented diagrammatically as an arrow. This type of diagrammatic representation is called a Bachman diagram (Bachman, 1969). Each set type definition consists of three basic elements:

(1) a name for the set type,

(2) an owner record type, and

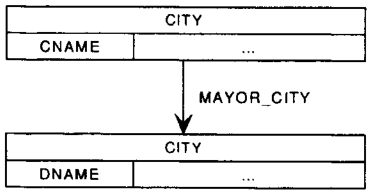
(3) a member record type.

The *set type* in Figure 5.12 is called MAJOR_CITY; CITY is the *owner* record type and DISTRICT is the *member* record type. This represents the 1:N relationship between city and districts majoring in this city. In the database itself there will be many set occurrences (or set instances) corresponding to a set type. Each instance relates one record from the owner record type - a CITY in our example - to the set of records from the member record type related to it. Each set occurrence is composed of (Elmasri & Navathe, 2000 p. 919):

(1) *one owner record from the owner record type*, and

(2) *a number of related member records (zero or more) from the member record type.*

Figure 5.12
The set type MAJOR_CITY represented in the network database model



A record from the member record type cannot exist in more than one set occurrence of a particular set type.

The network DBMS is obsolete for new applications. These are some of the *advantages* of the network database model:

(1) it is able to model complex relationships and represent the semantics in the relationships,
(2) it can handle most situations for modeling using record types and relationship types, and
(3) language is navigational; it uses constructs like: FIND, FIND member, FIND owner, FIND NEXT within set, GET etc. Programmers can execute optimal navigation through the database.

Some *disadvantages* of network database models:

(1) the navigational and procedural nature of processing,
(2) the database contains a complex array of pointers that thread through a set of records, and
(3) little scope for automated 'query optimization'.

Relational systems

As a response to the aforementioned limitations related to the complexity in the file systems, various *database systems* evolved. The main database concepts, as we explained earlier, are:

(1) file processing is replaced by an integrated database management system [DBMS],
(2) data are treated as a resource and are independent,
(3) redundancy avoided for consistency (logical) and efficiency (physical),
(4) flexible access through content, and
(5) interaction between separate tasks performed on the same data.

The basic difference between a *database system* and a *file system* is that a database consists of a single repository for related data, as opposed to separate and unrelated files as in the file system. In a database system, information is derived from data, which is usually stored in a database. To implement a database and to manage its contents, commercial software known as a Database Management System [DBMS] is needed. The database design defines the database structure; the DBMS stores the facts about the structure in the database itself. The database contains the data the user has collected and the "data about the data" is known as *metadata* (see

Figure 5.2). A good database design is important because even a good DBMS will perform poorly with a poorly designed database.

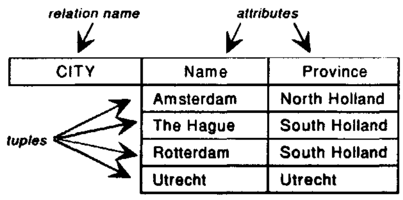
E. F. Codd, working for IBM, first developed the relational database model in 1970. It was considered revolutionary in principle but impractical at the time due to a general lack of sufficient computing power. The relational model represents the database as a collection of *relations*. Each relation resembles a table of values or a "flat" file of records. A relational database is a data structure that stores information about entities, the attributes of those entities, and the relationships among those entities. It is perceived by the user to be a collection of tables. Each row in a table represents a collection of the facts that correspond to a real-world entity or relationship. They are represented by a collection of related data values. All values in a column are of the same data type.

The relational database model offers the luxury of forgetting the actual physical data storage characteristics, thereby allowing one to concentrate on the logical view of the database. That is to say, the focus is on the human perception of data storage rather than on the often difficult-to-comprehend manner in which the computer sees the same data. Since the relational model achieves both *data independence* and *structural independence*, it becomes much easier to design the database and manage its contents.

One of the reasons for the relational database model's rise to dominance in the database market is its very powerful and flexible query capability. For most relational database software, the query language turns out to be Structured Query Language (SQL). SQL is a specific set of commands for retrieving data from relational databases. Currently the most popular of query languages, SQL, comes in a variety of formats, but the core statements are the same. While developers may add "extensions", the basic commands are consistent, a fact which theoretically enables SQL to function across different machines and different programs.

The basic data components in a relational database approach are *entities* and their *attributes*, and the basic logical structure is a two-dimensional *table* consisting of rows and columns. An *entity* is a "thing" in the real world [the Universe of Discourse] with an independent existence. It is something about which you want to store data; typically, a person, place, object, concept, or event. An *attribute* is a characteristic of an entity or object. It is a detailed piece of information describing an entity. Different kinds of attributes include single-valued attributes, multi-valued attributes, and derived attributes. Strictly speaking, the tables are called *relations*, the rows *tuples* and the columns *attributes*. The tuples are roughly equivalent to what is usually understood to be the records in a file, with the attributes indicating the meanings of the values in each tuple. No two tuples can have the same combination of values for *all* their attributes. The basic relational table is presented in Table 5.2. One of the fundamental principles of relational databases is that each table is a separate and independent unit (only logically speaking), although tables may be related to one another.

Table. 5.2
The attributes and tuples of a relation CITY, a basic example of a relational table



CITY	Name	Province
	Amsterdam	North Holland
	The Hague	South Holland
	Rotterdam	South Holland
	Utrecht	Utrecht

The operations of the relational model can be categorized into *retrievals* and *updates*.

There are three basic update operations:

(1) *insert* is used to insert new tuples in a relation; it provides a list of attribute values for a new tuple that is to be inserted into the relation,

(2) *delete* is used to delete tuples; it can violate only referential integrity, if the tuple being deleted is referenced by the foreign keys from others tuples in the database, and

(3) *modify* (or *update*) is used to change the values of attributes that are the same in existing tuples of the same relation.

A relational DBMS [RDBMS] manages tables of data and associated structures that increase the functionality and performance of tables. An RDBMS has three major aspects (Blaha & Premerlani, 1998 p. 182):

(1) data that is presented as tables,

(2) operators for manipulating tables, and

(3) constraints - supporting referential integrity and simple constraints.

These are some of the *advantages* of relational database models:

(1) they represent facts about both entities and relationships uniformly as relations,
(2) they usually contain many relations,
(3) the navigational and procedural nature of processing is based on relational algebra,
(4) theory and standards (for example SQL standard),
(5) wide availability,
(6) great extensibility,
(7) declarative access to data,
(8) data dictionaries,
(9) fast associative query, and
(10) robust security.

Some disadvantages of relational database models:

(1) *slow navigation* - perform slowly for applications with much navigation from object to object; joins inefficient for extensive navigation,
(2) *lack of some advanced features* - such as inheritance, schema evolution, versioning, configurations, long transactions and change notification,
(3) *few data types* - user cannot define new data types,
(4) *table is the only paradigm*, and
(5) *unfamiliarity to programmers* - RDBMS is unfamiliar to most programmers.

A relational data model must include a set of operations to manipulate data. It should be noted that the set of operations is based on the mathematical set theory. *Relational algebra* is a set of mathematical principles that form the basis for the manipulation of relational table contents; it comprises eight main functions: SELECT, PROJECT, JOIN, INTERSECT, UNION, DIFFERENCE, PRODUCT, and DIVIDE. It does not refer to the fact that the relational model utilizes relationships, and it would be erroneous to assume that other database models cannot permit the use of relationships.

Object-oriented database systems

Concepts

Object orientation is a recent "strategy for organizing systems as collections of interacting objects[3] that combine data and behavior[4]." (Blaha & Premerlani, 1998 p. 1). The object-oriented strategy creates a powerful synergy throughout the development life cycle by combining (ibid. p.1):

(1) *Abstraction*: allows a focus on essential aspects of an application while ignoring details, preserves design freedom until the later stages of development and this is probably the most important skill required from object-oriented development;

(2) *Encapsulation*: separates external specification from internal implementation; and

(3) *modularity/inheritance*: promotes coherence, understandability and symmetry by organizing a system into groups of closely related objects.

A system built with object-oriented methods is one whose components are encapsulated chunks of data and functions, which can inherit attributes and behavior from other such components and whose components communicate via messages with one another. Object-oriented concepts are applied in areas of computer systems such as software engineering, knowledge bases, artificial intelligence and databases. Object-oriented methods are especially helpful for database applications. Object-oriented databases were proposed to meet the needs of these more complex applications. The object-oriented approach offers the flexibility to handle some requirements without being limited by the data types and query languages available in traditional database systems. The key power of the object-oriented database is the fact that the designer of the DB system can specify both the structure of complex objects and the operations that can be applied to these objects. Database and program applications can be developed together for ease of conceptualization, implementation, maintenance and potential re-use.

An *object-oriented database* can be regarded as a persistent store of objects created by an object-oriented programming language [OOPL], such

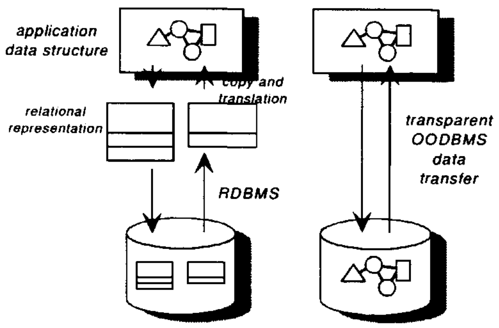
[3] *Object* is defined as a concept, abstraction, or thing that can be individually identified and has meaning for an application.

[4] *Behavior* is defined as a type of object based on the operations that can be externally applied to objects of that type.

as C++, SMALLTALK, or JAVA. OOPLs have their roots in the SIMULA language, which was proposed in the late 1960's. The concept of *objects, class, abstract data type* and *encapsulation* was proposed. An object typically has two components: state (value) and behavior (operations). With an ordinary programming language, objects cease to exist upon program termination; with an object-oriented database, objects persist beyond the confines of program execution. Object-oriented databases have adopted many of the concepts that were developed originally from object-oriented programming languages.

One goal of object-oriented databases is to maintain a direct correspondence between the Universe of Discourse and database objects, so that objects do not lose their identity and can easily be identified and operated upon. According to Cattell and Barry (1997), the object's structure in the application represents the basic functional structure in the object-oriented database (see Figure 5.13). The object-oriented database stores *persistent objects* permanently and allows the sharing of these objects among multiple programs and applications. An object-oriented database system interfaces with one or more OOPLs to provide persistent and shared object capabilities. An *object-oriented DBMS* manages the data, programming code and associated structures that constitute an object-oriented database. In contrast to a relational DBMS, an object-oriented DBMS is very broad in its syntax and capabilities.

Figure 5.13
Comparison of RDMS and ODBMS
[from: Cattell and Barry, 1997]


application data structure
copy and translation
relational representation
RDBMS
transparent OODBMS data transfer

One of the key concepts in object-oriented database systems is the type and class hierarchies and *inheritance*. This permits specification of new types or classes that inherit much of their structure and operations from previously defined types or classes.

Another concept of object-oriented database systems is representing *relationships* among objects. The relationship should not be explicitly represented but should instead be described by defining appropriate methods that locate related objects. This early approach of the object-oriented database system does not work very well for complex databases with many relationships, but recently developed technology solved this problem.

The concept of *operator polymorphism* refers to an operation's ability to be applied to different types of objects. This feature is also called *operator overloading*.

Data models

We already explained that the *object-oriented data model* is one type of data model. Data models are theoretical models, describing ways in which information can be represented and manipulated as a database. A theoretical model is a set of concepts, operations and rules, often expressed in mathematical form, which behave in a way that mimics the behavior of the thing that is being modelled. A data model has three aspects relating to the way information is modelled as data:

(1) the *structural* part of the data model is the set of structures with which a database can be constructed,

(2) the *manipulative* part is the set of operations by which databases can be manipulated, and

(3) the *integrity* part is the set of rules that constrain a database to be plausible.

The quality of the theoretical model can be judged by how accurate and complete a representation it provides of the things it models and the extent to which its predictions correspond to what happens in reality. Each model, as we have already analysed, differently maps and specifies entities of the universe of discourse. The quality of the data model must be also judged on the basis of how well it can represent and model information and its behavior. In this respect, *object-oriented data models* can capture more of the meaning of the information represented in a database than other earlier models, because they can represent both complex information structures and complex operations that occur.

The difference between an RD model and an OODB model is in the intermediate representation, which does not exist in an OODB model. In an OODB model, transfers of data are direct and transparent via the OODBMS (see Figure 5.13).

To represent a complex information structure and complex operations, an object-oriented data model must have concepts that comprise a methodology (UML), which occurs within the following steps (Blaha & Premerlani, 1998 pp. 5-6):

(1) *conceptualization* is the process of analyzing and formulating tentative requirements,

(2) *analysis* is aimed at specifying *what* needs to be done from the process of conceptualization without constraining *how* it is done,

(3) *system design* is the process of devising a high-level strategy for solving the application problem,

(4) *detailed design* (or *object design*) is the process of transforming the real-world analysis model into a form amenable to computer implementation,

(5) *implementation* is the translation of design into the actual programming language and database code, and

(6) *maintenance* is the process of developing documentation and traceability from the models through the code to facilitate subsequent maintenance.

The *UML methodology* uses three complementary models to present aspects of a problem and its solution. Each model presents a perspective of the system, which by itself is incomplete. The models combine to describe the system fully. These three models (Blaha & Premerlani, 1998 pp. 7-8) are:

(1) *object*, characterizes the static structure of things (objects); it looks at structure in terms of groups of analogous objects (classes), their similarities and differences (generalization) and their important relationships with one another (associations); an object model is expressed with one or more class diagrams; it is important for database applications because it concisely describes data structure and captures structural constraints;

(2) *dynamic*, describes temporal interactions between objects - various stimuli that occur and the response of objects to the stimuli; and

(3) *functional*, defines the computations that objects perform - how output values are computed from input values.

An object-oriented DBMS [OO-DBMS] manages the data, programming code and associated structures that constitute an object-oriented database. An OO-DBMS is considered for applications with a complex data structure or demanding distribution and concurrent user requirements. UML models largely describe properties of data structure and data manipulations. There are several other paradigms at the frontier of OO-DBMS technology:

(1) *real-time OO-DBMS* - in which the result must be correct and timely,

(2) *deductive OO-DBMS* - combines conventional data management abilities with the inference capability of an expert system, and

(3) *active OO-DBMS* - combines the management of data and management of state.

There are several *advantages* of the object-oriented database models:

(1) *one development program* - uniform abstraction for the design of both programming and database code; they map naturally to all major languages and the standard types of DBMS's,

(2) *they improve the quality of data* - they can weave many constraints into the database structure; this naturally leads to a normalized relational database schema,

(3) *better performance* - they simplify database tuning,

(4) *faster development* - is required to exploit DBMS capabilities fully; they can simplify database code,

(5) *less debugging* - fewer errors in the corresponding applications (they detect application errors),

(6) *easier migration* - software comes and goes as new systems are used and old systems become obsolete,

(7) *easier integration* - they provide a uniform representation that facilitates understanding and integration,

(8) *fast navigation,*

(9) *advanced features*, and

(10) *flexible locking protocols.*

On the other hand, the disadvantages are as follows:

(1) *incomplete theory and standards* - OO-DBMS's have been implemented without formal theory,

(2) *possible database corruption,*

(3) *lack of logical extensibility*, and

(4) *lack of support for meta-applications.*

The following kinds of applications can benefit from object-oriented database systems:

(1) *engineering design applications* - such as CAD, CAM, CIM and CASE,
(2) *multimedia applications* - appropriate for complex graphic, audio and video applications,
(3) *knowledge bases*,
(4) *applications with demanding distribution and concurrency requirements*,
(5) *applications that require advanced features*, and
(6) *electronic devices with embedded software*.

Structure

An *Object Data Management Group* [ODMG] describes object-oriented programming language, commonly known as *ODMG 3.0* (Cattell et al, 2000). It is becoming the standard for storing objects in databases. *Object Definition Language* [OML] is a specification language used to define the object types that conform to the ODMG Object Model. The ODMG enables many vendors to support and endorse a common object interface to which customers write their database applications.

In designing an object-oriented database, information is analyzed in terms of the *entities* it describes, the ways in which they are interrelated and their behavior. Entity is a general term, which, as we explained earlier, refers to anything that is uniquely identifiable and about which we wish to record facts.

Objects

Entities[5] are represented in an object-oriented database as a special structure, called an *object*. An object-oriented database is a collection of objects, each of them representing an entity. An object is a combination of data that represents the *state* or value of the entity that the object represents and procedures that represent its *behavior*. The data are called the object's state and the operations are its behavior. The procedure that implements a particular operation on an object is called a *method*. The procedures

[5] *Entities* are defined as objects with a physical existence, such as a particular person, object, house, or they may be objects with a conceptual existence such as a company, job, course etc.

associated with an object act as that object's *interface*, and are the only means by which the data and procedures are integrated within an object, which is called *encapsulation*. Encapsulation is also the bringing together of representations of both *state* and *behavior* within one object. The entities must be represented as objects in the object-oriented database.

Each object, in order to represent the characteristic of entity, has the following structure:

(1) *identity*, the fact that entities represented by objects are uniquely identifiable,
(2) the *state* of an object is the set of values that it contains, and
(3) the *behavior* of an object is the set of operations that can be applied to it.

In addition to these three characteristics, it is possible to assign to one object one or more meaningful *object names*. The behavior and state of an object are closely related.

Objects are used to represent entities that exist outside the object database and are relevant to the operation of the model that the database presents. Real-world entities may be physical (things you can touch, for example) or abstract (such as concepts, processes, agreements).

An object can also be used to represent entities, which exist only in the object database, called *object database entities*. Here are a few examples:

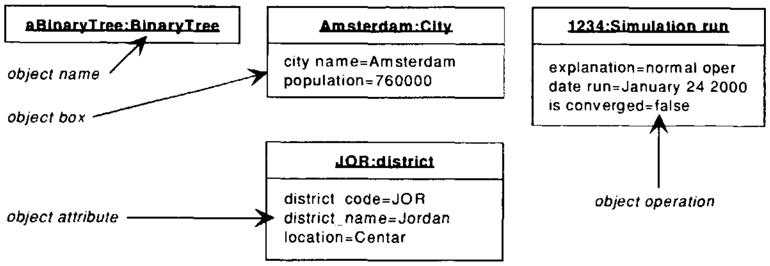
(1) objects representing *values*, such as numbers, text, or images that are used in the object database to model facts about entities represented by other objects,
(2) objects representing *entities* that model the different types of object that an object database may contain, or
(3) objects representing *entities* that model certain exceptional events that can occur within the object database system.

Generally, objects represent the Universe of Discourse as an abstraction of distinction between 'physical' and 'conceptual' objects. Two objects are equivalent if they have the same object status (identity), and they are equal if they have the same state values.

The state of an object represents the *properties* of the entity modelled by the object. There are two types of property that can be represented (see Table 5.3):

(1) *attributes*, the values of which represent facts about the entity modelled, and

Table 5.3
A graphical representation of *objects* and their *relationships* in an instance diagram [Based on: Blaha and Premerlani, 1998]



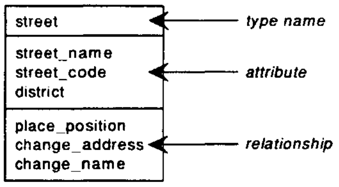
(2) *relationships*, the values of which represent the associations between the entity and other entities represented in the object-oriented database.

An object's state may carry a number of *attributes*, each of which represents a fact about the entity represented by that object. Each attribute has a descriptive name and value. A value is a piece of data. An attribute exists only as a part of an object and is not an object in its own right. An object attribute is a named property of a class that describes a value held by each object of the class. Object is to class as value is to attribute. Objects have identity, values do not.

As we see, an object's state may also include a *relationship*. Each relationship is given one or two descriptive names and has a value. The first name labels the relationship path from the object to the associated object and the optional second name labels the inverse relationship paths. A relationship, like an attribute, is not an object in its own right. Objects and their relationships are delineated on an instance diagram. An object is denoted as a box with an object name followed by a colon and the class name. The object and class names are both underlined and boldface as they are presented in Table 5.3. (Based on: Blaha and Premerlani, 1998).

Two notations are used for describing object types, one graphical and the other textual. The graphical representation is used primarily in the design of object-oriented databases. An object type can be graphically presented, as shown in Table 5.4.

Table 5.4
A graphical representation of the *object type properties*
[from: Eaglestone and Ridley, 1998]



An object's behavior is defined by a set of *operations*. These operations define all things that can be done to the object and therefore define the object's interface. An operation is a function or procedure that may be applied to or by objects in a class. Some operations are *polymorphic*, that is, they are applicable to many classes. In general there are two types of operation:

(1) *update operations,* which modify the state of an object, changes that happen to the entity represented by the object, and

(2) *retrieval operations,* which query the state of an object, necessary in order to determine the current state of the 'world'.

The small database system becomes the object, the database itself becomes the state of the object and the applications programs become the procedures which provide the interface to the object.

An object is organized into three parts:

(1) its *state*, which is the data it carries (each object will have a collection of state variables, the values of which are the object's state),

(2) its *interface*, which is the collection of operations by which the state of the object can be accessed and modified, and

(3) its *implementation*, which is the code that implements the operations.

The state and interface are defined by the object's type and the implementation by its class.

Many objects may share the same set of characteristics. Objects with the same characteristics are said to be of the same *object type*. An object type definition defines the possible states and behavior of objects of that type. Objects of a particular type are called *instances* of that type and the set of instances is called the type's *extent*. The types define the ways in which an object of that type can be used, but not how it is implemented. A type definition defines only the interface to the object. The implementation will define the representation and procedure that implement each operation. These procedures are called *methods*. A type can be implemented in many ways. An implemented type is called an *implementation class* or simply a *class*.

Class

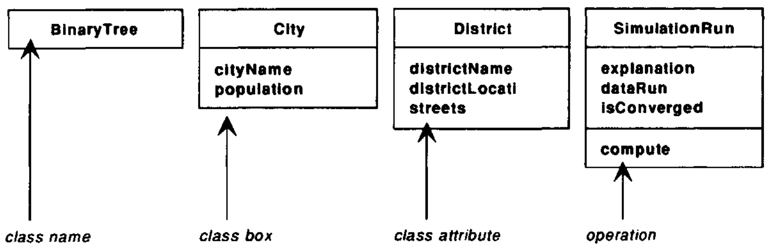
A set of objects with the same characteristics (i.e. the same descriptive attributes and the same behavior) is usually referred to as an object type or a *class*. The class belongs, as we explained earlier, to the UML. An object is an *instance* of a class. "A class is a description of a group of objects with similar properties (attributes of objects), common behavior, similar relationships to other objects and common semantics." (Blaha and Premerlani, 1998 p. 14). Classes provide a mechanism for sharing across similar objects. Some classes have real-world counterparts (street or district) while others are conceptual entities (simulation run and equation). Other classes still are purely artifacts of implementation. The choice of classes depends on the nature and scope of an application and is a matter of judgement.

Classes and their relationships are delineated in a *class diagram*, which shows the classes that correspond to the objects. Class is denoted by a box with the *class name* in the top portion of the box. An optional second portion of the box may be a *list of attributes*. A third optional portion of the box may be a *list of operations*. The convention is to list the class names in boldface, capitalize the first letter of the class name and use lowercase letters for the first letter of the attribute and operation name. See Table 5.5 for class diagrams (based on: Blaha and Premerlani, 1998).

The class definition was also given by Coad and Yourdon (1990, p.53): *"Class is a collection of one or more objects with a uniform set of attributes and services, including a description of how to create a new object in the class"*. There are three uses of the term classes (Cattell, 1991):

(1) a class defines the *intent* - the structure and behavior of objects of a particular type; the intent is a theme or essence of semantic meaning of a class (the intent consists of the properties, behavior and relationships to other classes of objects),

Table 5.5
A graphical representation of the *classes tree* [class diagram]
[Based on: Blaha and Premerlani, 1998]



(2) a class defines a *representative* - types represented by objects themselves (this meaning arises with the prototypical or delegation approach to object-oriented programming), and
(3) a class defines an *extent* - the set of objects with a particular type.

Relations

A class definition specifies the characteristics of the type instances, either explicitly or by specifying other objects from which characteristics are *inherited. Inheritance* is a special type of *relationship* that can exist between object types, whereby one type is a subtype of another and therefore inherits all of its characteristics. A class can inherit characteristics from other object types in two ways:

(1) *subtype/supertype relationships* - a class or interface definition can inherit behavior from one or more instances, and
(2) *extending a relationship* - a class can extend another class, in which case it inherits both behavior and state from the class.

A subtype/supertype relationship signifies the following:

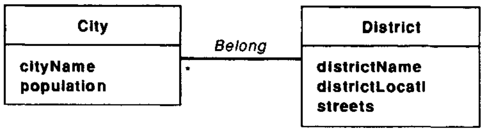
(1) an object subtype automatically *inherits* all of the attributes, relationships and operations of the object types which are its supertypes,
(2) an object subtype can have *additional characteristics,*
(3) an object subtype can refine the characteristics it inherits from its supertypes by redefining them,
(4) an object is defined as a class but is also an instance of each of its supertypes, and

(5) an object is a member of the extent of the type upon which it is defined and also of the extents of each of its supertypes.

Object types are themselves entities within the object-oriented database system and are represented as objects. This type of object is called a *meta object*, because it describes the characteristics of other objects. The data contained in meta objects is called *metadata* (we discussed 'data about data' earlier). The meta objects collectively define the *object database schema* and are used by the ODBMS to define and access objects contained in the object-oriented database. The meta objects which represent an object type have properties of their own. The values for these are specified in the *type properties* part of a class definition.

"A *link* is a physical or conceptual connection between objects." (Blaha and Premerlani, 1998 p. 17). Physical connections can be for example between two objects and a conceptual connection can be for example between city and district. Links relate two or more objects. An association is a description of a group of links with common structure and common semantics, for example city and district. A link is an instance of an association. The links of an association relate objects from the same classes and have similar properties. A link is denoted by a line connecting related objects; a line may consist of several line segments. One should not confuse links with objects. An object has inherent identity and exists in its own right, whereas a link derives its identity from the related objects and cannot exist in isolation. A link can exist only if all objects to which it refers also exist. An object attribute is a property of a class that describes a value held by each object of the class. A link attribute is a property of an association that describes a value held by each link of the association. Table 5.6 shows the graphical representation of an association, based on: Bedini, Ceri and Navathe (1992).

Table 5.6
The graphical representation of an *association*
[Based on: Bedini, Ceri and Navathe, 1992]



"An *association class* is an association whose links can participate in subsequent associations" (Blaha and Premerlani, 1998 p.23). Like a class, an association class can participate in association. A *qualified association* is an association in which the objects in a 'many' role are partially or fully disambiguated by an attribute called the *qualifier*. One-to-many and many-to-many associations may be qualified. A qualifier selects from among the target objects, reducing the effective multiplicity, often from 'many' to 'one'.

"*Generalization* is a relationship between a class (the *superclass*) and one or more variations of the class (the *subclasses*)" (ibid. p.26). Generalization organizes classes by their similarities and differences, structuring the description of objects. The superclass holds common attributes, operations, state diagrams and associations; the subclasses hold specific attributes, operations, state diagrams and associations.

"*Specialization* provides another perspective on a system's structure" (ibid. p.26). Specialization has the same meaning as generalization but takes a top-down perspective, starting with the superclass and splitting out variations. Generalization takes a bottom-up view, starting with the subclasses and abstracting commonality.

"The *discriminator* is an attribute that has one value for each subclass. The value indicates which classes further describe an object" (ibid. p.26). The discriminator is simply a name for the basis of generalization. Discriminators are superfluous for OO-DBMS's, but they can be useful for implementations that divide objects into multiple records.

"Generalization is the structural relationship that permits the *inheritance* mechanism to occur" (ibid. p.27). A subclass *inherits* the attributes, operations, state diagrams and associations of its superclass. Inherited properties can be reused from a superclass or overridden in the subclasses. Simple generalization is another name for *single inheritance*. The case in which a subclass may have multiple immediate superclasses is called *multiple inheritance*.

Unified Modeling Language

The Unified Modeling Language [UML], which was developed in 1998 by Booch, Rumbaugh and Jacobson, is mostly available for programming applications. The treatment of databases, especially the design and implementation process, is incomplete. Database applications require custom treatment because they are very different from programming applications. Database applications tend to have a much larger and more complex data structure and less rich behavior compared to programming applications.

The UML addresses programming as well as database applications. The UML obtains techniques and concepts for dynamic and functional modeling. The UML methodology is based on object-oriented modeling and it has a compatible graphical notation.

The basic UML elements are:

(1) *object* - a concept that helps understanding of specific problems of the *universe of discourse* and helps to implement them in the computer;

(2) *object-identity* - the identification of an object by its differences;

(3) *object class* - class is a group of objects with similar attributes and associations; objects in one class with the same attributes and associations are called an *object-type*;

(4) *attributes* - an attribute is a quality of the object in the class and it has a name in the class; there are basic attributes and derived attributes;

(5) *operation, methods* - operation is a function of transformation which the user can make with objects in one class (all objects in one class have the same operations); method is the implementation of the operations in the class;

(6) *links and associations* - a link is a physical and logical connection between instances, links make relations between objects; an association describes a group of links with a collective structure and semantics, an association is equivalent to a relationship-type,

(7) *multiplicity* - equivalent to cardinality;

(8) *aggregation* - a 'part-whole' or 'part-of' relationship type,

(9) *generalization* - denotes a large hollow arrowhead which points to the superclass; the superclass may be directly connected to each subclass,

(10) *candidate key* - minimal sets of attributes which identify one object or link; the object ID is always a candidate-key,

(11) *constraints* - the functional relationship between objects; constraints restrict the values of objects, classes, attributes, links and associations, and

(12) *group constructions* - two similar constructions: module and sheet.

Database design

The database design is the logical and physical structuring (through the DBMS) of one or more databases to accommodate the information needs of users for a defined set of applications. The goals of the database design are multiple:

(1) to satisfy the information content requirements of the specified users and applications,

(2) to provide a natural and easy-to-understand structuring of the information, and

(3) to support processing requirements and any performance objectives such as response time, processing time and storage space.

Generally, each database design consists of three separate phases called *conceptual, logical* and *physical* design. Here we will discus the importance of conceptual design within this methodological framework.

The database is just one of the components of information systems which include application programs, user interfaces and all other necessary software packages. The software packages for managing databases, mainly for storing, manipulating and retrieving data on a computer system, were explained earlier under *database management systems* (DBMS). According to Batini, Ceri and Navathe, (1992), the process of information systems, presented in Figure 5.14, includes the following parts:

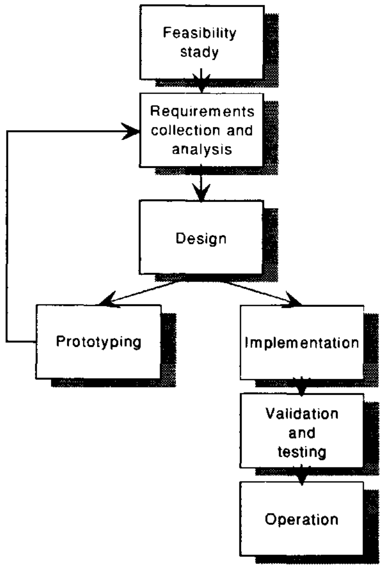
(1) the *feasibility study*, concerned with determining the effectiveness of various alternatives in the design of an information system and the priorities among the various system components;

(2) *requirement collection and analyses*, concerned within the enterprises and the problems that the system should solve (the process when users describe their needs and 'requirement specifications');

(3) *design*, concerned with the specification of the information system's structure (distinguishing between 'database design' and 'applications design');

(4) *prototyping* is a simplified, inefficient implementation that allows users to verify that the information system satisfies their needs (useful for correcting and adding requirements based on practical experimentation);

Figure 5.14
The process of information systems
[from: Batini, Ceri and Navathe, 1992]



(5) *implementation*, concerned with the programming of the final operational version of the information system;

(6) *validation and testing*, is the process of assuring that each phase of the development process is of acceptable quality and is an accurate transformation from the previous phase; and

(7) *operation*, starts with the initial loading of data and terminates when the system eventually becomes obsolate and has to be replaced (meintenance is required to adopt the system to new conditions, to enhance it with new functions or to correct errors that were not detected during validation).

The process of information systems shows us that database design should be preceded by requirement analysis, should be conducted parallel to application design and should be followed by the implementation of either a prototype or a final system.

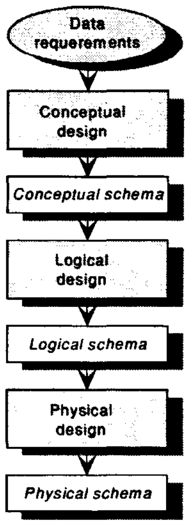
The database design also involves two parallel activities: *conceptual schema design* and *transaction and application design*. Conceptual schema design examines the data requirements and produces a *conceptual database schema*. Transaction and application design examines the database application analysed and produces high-level specifications for these applications.

Phases of database design

The database design process often begins with informal and poorly defined requirements. In contrast, the result of the database design process is a rigidly defined database schema that cannot easily be modified once the database is implemented.

According to Batini, Ceri and Navathe, (1992), the database design process can be broken down into the following stages, graphically represented in Figure 5.15:

Figure 5.15
The stages in the Database Design
[from: Batini, Ceri and Navathe, 1992]



(1) *The conceptual design* starts with the specification of requirements and results in the *conceptual schema* and *conceptual model*. The *conceptual schema* is a high-level description of the database structure and the *conceptual model* is the language that is used to describe the conceptual schema. The task of the conceptual design is to describe the conceptual schema. The other task of the conceptual design is to describe the *'information content'* of the database rather than 'storage structures' that will be required to manage this information;

(2) *The logical design* starts from the conceptual schema and results in the logical schema and logical model. A *logical schema* is a description of the structure of the database that can be processed by the DBMS software. A *logical model* is a language that is used to specify logical schemas, and belongs to the classes: rational, network and hierarchical; and

(3) *The physical design* starts from the logical schema and results in the *physical schema*. A *physical schema* is a description of the implementation of the database in memory and it describes the storage structures and methods used in order to access data.

For large and more complex databases, as we have already shown in Figure 5.14, Batini, Ceri and Navathe (1992) proposed six phases of the database design process:

(1) *requirements, collection and analysis,*
(2) *conceptual database design,*
(3) *choice of a DBMS,*
(4) *data model mapping* (also called *logical database design*),
(5) *physical database design*, and
(6) *database systems implementation and tuning.*

The design process consists of two parallel activities. The first activity involves the design of the data content and structure of the database. The second one relates to the design of database applications. There are a lot of interactions among these two parallel activities, and they are closely intertwined.

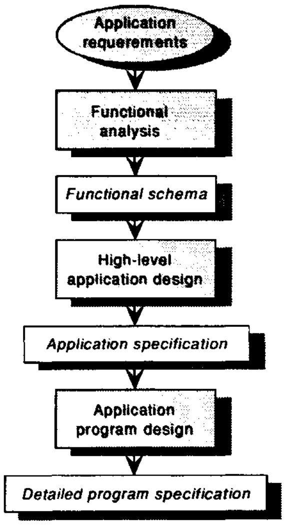
Traditionally, the database design process has primarily focused on the first of these activities, whereas software design has focused on the second. This may be called *data-driven* versus *process-driven design*. Database designers and software engineers are rapidly coming to recognize that the two activities should proceed hand in hand and design tools are increasingly combining them.

Functional analysis

The functional analysis starts from *application requirements*. Application requirements are high-level descriptions of the activities performed within an organization and of the information flows exchanged between activities. The result of the functional analysis is a collection of function schemas, which describe such activities and information flows through the use of specific function models.

The *high-level application design* is a part of the functional design which maps function schemas into application specifications (they describe how applications access databases). Functional analysis within Database Design according to Batini, Ceri and Navathe, (1992) is shown extended in Figure 5.16.

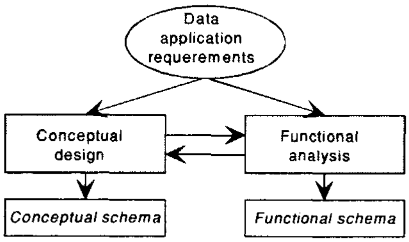
Figure 5.16
Functional analysis within Database Design
[from: Batini, Ceri and Navathe, 1992]


Application requerements
Functional analysis
Functional schema
High-level application design
Application specification
Application program design
Detailed program specification

The complementary relationship between the *conceptual design* and *functional analysis* in the process of the database design is presented in

Figure 5.17 (also related to Batini, Ceri and Navathe, 1992). They both contribute some good features and should be closely related.

Figure 5.17
***Conceptual design* and *functional analysis* within Database Design [from: Batini, Ceri and Navathe, 1992]**



Object-oriented database design

Object-oriented database design starts with the conceptualization of an idea for a new system. Conceptualization begins with thinking about how to provide new functionality or to simplify delivery of existing functionality. The output of conceptualization is a set of specific requirements that serve the following phase of analysis. The task of analysis is to construct models of the universe of discourse. During analysis, *what* must be done should be described, but not *how* it should be done. The problem must be understood before attempting the solution.

The next stage in the object-oriented database design process is *system design*. In system design, a high-level strategic decision is made for solving the problem by formulating system architecture. There are several principles for guiding this formulation, which we explained earlier. For difficult and complex problems, candidate architectures should first be generated, and then evaluated in separate steps. A specific OO-DBMS paradigm must be chosen. The last stage is to make a more detailed database model that can be programmed. Here we will briefly explain something of the importance and specificity of the object-oriented database design.

The object-oriented data model has sufficient expressiveness to implement semantic data models directly. Data analysis methods provide a systematic process by which conceptual models are derived. There are two main approaches for data analysis:

(1) *top down*: can be thought of as working from the general to the specific, and

(2) *bottom up*: from the specific to the more general.

In a top down approach, the analysis starts by attempting to identity the entities. This approach seems to fit better for object-oriented design, since a central feature of an object-oriented system is the representation of entities as objects, and important features such as inheritance have a natural top down flavor.

A top down method is based upon entity-relationship analysis. Batini, Ceri and Navathe (1992), proposed that design proceeds through the following stages:

(1) *identification of objects in the system,*

(2) *refining the objects* - the characteristics of the classes are determined in the following subtasks:

- *selection of classes to use*
- *identification of relationships*
- *determining the cardinality of relationships*
- *classification of relationships*
- *identifying attributes*
- *identifying operations*, and

(3) *revision of the attributes and classes* - the design is reviewed and refined.

The first stage is to identify candidate objects. Most object design methods suggest a list of the potential sources for classes and objects that include the following categories of entity that an object may represent:

(1) *things*, or *tangible things* are the physical objects in the system, and the easiest objects to identify,

(2) *people* – it may be easier and useful to regard them as a separate category,

(3) *roles* means the roles that people play within the system,

(4) *organizations* may be thought of as things that are not too tangible and form a useful subgroup,

(5) *concepts* may be useful for those things that are not tangible, i.e. do not have physical presence,

(6) *events* or *processes* is another grouping for non-tangible things (some of these may involve tangible objects), and

(7) *places* or *locations* may be important if we are actually concerned with physical locations.

The descriptions of the process that is analyzed are the output of a preliminary phase in which information about the process is gathered in the following ways:

(1) observing the process,
(2) interviewing people concerned with the process and the future object-oriented database system,
(3) inspecting existing procedures and documentation within the process,
(4) surveying experts and existing solutions to the problems the object-oriented database is to address, and
(5) utilizing other information already known by the designer/s.

Candidate objects are identified by analyzing the organization description to find all nouns or noun phrases. Each noun or noun phrase may name or describe some phenomenon which may be desired to be represented as an object within the object-oriented database.

Candidate objects must be carefully selected:

(1) candidate objects should be at an appropriate level of abstraction - for example, "thing" is too general to be useful and "MissSaigon" too specific,
(2) a candidate object will usually represent general categories of entities rather than individual ones - "person" is a better candidate than "MissSaigon", which is an instance of a person.

Candidate objects that are similar in some way are grouped in order to determine a set of classes that characterize the candidates. In the design process, candidate groups must be analyzed in an attempt to (Batini, Ceri and Navathe, 1992):

(1) *remove or refine objects* which are too general or too specific to be useful within the object database design,
(2) *remove redundant objects*, those which, for example, represent concepts already covered by other objects,
(3) *identify relationships* in which an object is a specialization of another and aggregation relationships where one object is part of another,
(4) *refine the groups into subgroups*, where appropriate, and
(5) *identify inter-group relationships*.

There are also classes that must tackle how to move from an initial set of candidate classes to the ones that will finally be used. There are a number of classes that should be eliminated from the list of candidates, such as:

(1) *redundant*,

(2) *irrelevant*, and

(3) *too vague*.

The determination of the relationships that can exist between the objects is an important task in the design process. In the same way that there are some general sources for objects, it is useful to have some guidelines for identifying relationships. There are a few common verbs, and verb phrases that can be good indicators of relationships between objects:

(1) *is a* - a phrase that is used in identifying objects since this relationship is a specialization which can be modeled with inheritance;

(2) *aggregation* - is a phrase where one object is composed of a number of other objects, described with phrases like *is made up of, is a part of, is composed of* etc.; and

(3) *has* - in many cases this will indicate attributes of an object.

It is also important to determine the characteristics of each relationship. This involves analyzing the relationship cardinalities i.e. the numbers and types of objects that may participate in occurrences of each relationship. The cardinality of a relationship is determined by analyzing the relationship paths in both directions and also the inverse of the relationship.

The attributes of a class are all the facts or things that are known about an object that are not represented by relationships to other objects. Identification of attributes may require more detailed descriptions of individual object types. In general, attributes will be the things that are totally dependent on the object.

Operations on objects serve two tasks within an object-oriented database. They can be used to represent derived attributes, i.e. values computed from an object's state, rather than stored as part of the state. Thy can also represent the behavior of the entity represented by an object - the behavioral semantics. Analysis of behavioral semantics is done by analyzing the changes that can occur to an object during its life and the manner in which those changes take place. There are two analysis techniques:

(1) *state diagrams* - representing all possible changes that can occur to the state of an object, and

(2) *message trace diagrams* and *object message diagrams* - representing the messages that pass between objects when specific transactions take place.

A *state diagram* can be produced for each class, but it is usual only for objects with an interesting, dynamic behavior. A state diagram is used to analyze which operations should be defined for an object. There must be an operation for each state transition, since the state of an object can be changed only by its operations. The testing of correctness of a state diagram is done by running through a number of *scenarios*. A scenario shows the use of the object-oriented database system within a particular situation. Describing scenarios is a way of testing the system design in a manner similar to the way in which programs are tested using test data.

A *message trace diagram* represents objects in a transaction as vertical lines. Arrows between those lines then represent messages passed between these objects. The time sequences of the messages is represented from top to bottom.

These diagrams represent ways in which the system is likely to be used in terms of the interactions that will take place. The scenario can then be followed through in greater detail, in terms of interactions within the object-oriented database system. This can be described using an object message diagram, in which the flow of messages and responses between objects is represented.

Finally, each operation must be specified in details so that it can be programmed. These specifications will include:

(1) *the operation name and signature*, the type of value that it returns and its parameters,

(2) *a statement of what it does*,

(3) *the input parameters* - the name and type of each one,

(4) *the output parameters* - the name and type of each one,

(5) *modified objects* - which objects have an altered state as a result of executing this operation,

(6) *preconditions* - these are the conditions that must hold for the operation to be executed correctly, and

(7) *postconditions* - the new conditions created by executing the operation.

As we see, object-oriented database design methods provide a systematic way of designing an object-oriented database. The unified modeling language (UML) provides a diagrammatic language for representing analyses and designs for object-oriented systems in general, and we briefly showed how it works in object-oriented database design.

Some general comments about Perspective-DB technology

Perspective-DB[6] is an object-oriented database management system (ODBMS) developed by Bitbybit Information Systems. A database management system is used to design, create and maintain a database. There are different kinds of database management systems, and object-oriented management technology is the core of the new generation of database systems. The starting point of this new database technology is the specification of the user's environment as object-types. Perspective-DB provides fast access to stored objects and efficient general object processing services. A property of Perspective-DB is the visual presentation of information.

The object-oriented graphical user interface provides an intuitive schematical overview of objects, their attributes, relations and operations. The system is compact, robust and easy to extend. Reasons for this are the layered and modular structure as the characteristics of the object model. The programming language of the system is C++, the accepted industry standard.

Characterization of the Perspective-DB object model

We took all material related to Perspective-DB technology, with permission, from the promotional Bitbybit web site[7] and from their developer documents.

At the highest abstraction level, the object model of Perspective-DB knows just the object. An object can be characterized by several other objects. Objects characterizing other objects are called members. For

[6] © Bitbybit Information Systems, Delft
[7] http://www.bitbybit-is.nl

developers and end-users, members play the role of attribute, relationship and operation. Operations are generic, analogous to generic database management, or specific, e.g. necessary in a particular application.

Perspective-DB has an object model with a corresponding graphical and textual representation. The graphical object model is extended with the dynamical behavior of the application (Essenius et al, 1998). As a pre-condition, the object model provides constructs to unify the notation of object definition and the presentation of the objects in the graphical-user-interfaces (GUI). The object schema not only defines the object base but is also used to manipulate the running application directly by the end-user. As a result, the end-user view of an object schema is the GUI for the application in an interactive employment mode.

The representation of an object within the GUI supports several views of the objects (schematic, forms, overview tables), which are synchronized and follow the same behavior. Synchronized means that when an update is made in one of the views, this update is immediately valid in the other views of the same object.

The end-user activates an object and its members by clicking it in the owner instance. The user will be guided by color codes and a cursor to the following action or to choices between actions within a particular transaction. When arguments are needed, e.g. for an operation or a query, the system asks for them by coloring the place for insertion. When the place of argument values is predefined, it lists the values that can be chosen.

Perspective-DB knows two categories of relationship types:

(1) *ownership relationship type* represents an existential dependency constraint. It defines which instances of a particular object type cannot exist without the existence of another instance of the same or another object type. These types form ownership hierarchies. An instance has only one owner at any particular time. It may be the owner of several other instances at the same time. An ownership hierarchy is built with the root being the most abstract object type, and existence of the root is always guaranteed.

(2) *a relationship type* which does not build an ownership hierarchy is secondary. A secondary relationship type cannot violate existential dependency.

Providing an existential dependency constraint as a relationship type ensures that keeping this constraint is part of the generic object management functionality. Consistency in keeping information does not depend on the experience and 'good will' of the designer and end-user.

In addition to presentation in schematic form and tabular overviews, an instance can be presented, for example, as a graphic, photo or video. Each multi-media presentation of an object is an attribute of that object. Activating the attribute returns the media files as attribute values. The multi-media attributes can be activated within any of the aforementioned views.

The multi-media presentation provides the user with several perspectives of a particular piece of information without losing the semantic context of the object. We consider the fact that we can use multi-media presentations as attributes of objects direct within the running application as an advantage for the user. The multiple visual perspectives of the same information and the ability to offer them in the context of the object schema makes it possible to use all specified aspects and relationships between objects for information search.

Object representations in the Perspective-DB

An object representation (model) is a set of generic concepts to describe certain characteristics of real or imaginary objects. The object model is used to produce the definition of the object base. The result of the definition process is a set of object schemas. Here we shall give a short introduction to the main modeling concepts, which is necessary in order to understand the object schema described in the chapter.

An object type is a characterization of objects through the definition of its members. All objects that are characterized by the same types of members belong to the same object type. Members can have properties with their attributes, relations and operations. Relations when applied together with connectors form relationships. Relationships may have their attributes; they are represented attached to the connector. The ownership relationship forms a hierarchy. Any object that fits onto a particular object type is called an instance. An *attribute* is a characteristic of an object-type, for which the user can insert a particular value (for example: name, length, width, etc.). A *relation* is a possible connection between object-types that the user can make.

An *operation* is an action that the user can perform on an object-type. Operations can be specific for a certain object type (they are represented attached to the sides of the rectangle) or they can be generic, meaning that they can be applied to each object type (they are represented on the surface of the rectangle). If a generic member does not suit a certain object type, it will not be visible. There are two types of basic generic operations: (1) to view information and (2) to insert and remove information (see Table 5.7).

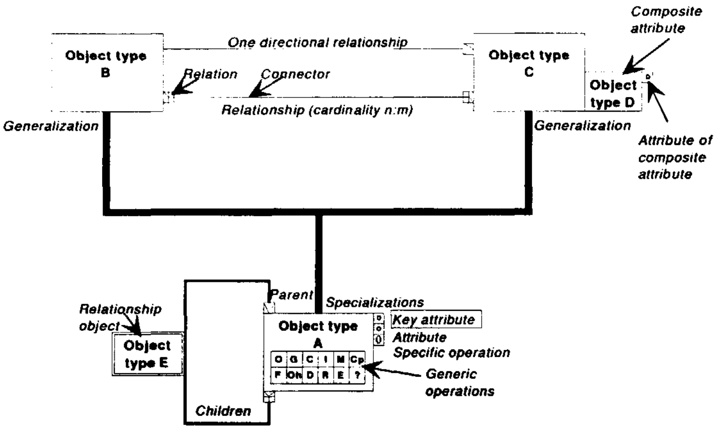
A *generalization* object type is an abstraction of several *specialization* object types; it describes the common members of its specialization. Generalization and specialization object types form a hierarchy. A specialization inherits members of its generalization. A specialization can choose not to use a member it inherits.

The object schema notations originate from (Essenius et al, 1997,1998; Gerhardt and Simon, 1999) and are presented here in Figure 5.18.

Table 5.7
The meaning of Perspective-DB notations [From: Bitbybit]

Symbol	Meaning
Large beige rectangle	Object-type
Small orange rectangle	Property
Small yellow rectangle	Relation
Small grey rectangle	Generic operation
Small pink rectangle	Special operation
O	Instance overview
F	Form related instances
G	Getting related instances
Oh	Ownership
?	Getting help
C	Creating new instances
D	Destroying instances
I	Inserting relations
R	Removing relations
M	Modifying values
E	Export
Cp	Copy

Figure 5.18
Object representation in Perspective-DB (developer's view)
[From: Gerhardt and Simon, 1999]



Graphic representation in the Perspective-DB

Graphically, an owned object type is always placed below its owner. An object type is represented graphically by a rectangle. Members are represented by rectangles attached to the object type they characterize. Connectors are lines between object types. The name of an object type is written on the rectangle's surface with the first letter capitalized. The name of a member is written beside the rectangle.

The position of member is fixed at the moment it is defined in the base object-type in the developer's view. As a convention, the key attribute of an object-type is placed (if possible) on the upper right side of the object-type. The attribute does not need to be redefined in the specializations, because they inherit the attribute from the base object-type. This means that all specializations manage the same position as well.

The generalization is positioned beneath the specialization. Sometimes it is not possible to put a specialization directly above the generalization. In that case, the generalization object-type can be put somewhere else, but both generalization and specialization must have a reference to each other.

The structure of an object type can be hidden in the schema in which it is used, e.g. in order not to overload the user with too many details. However, when required, it can be made to pop up and the user can zoom in to its structure. In this way, we can reach different levels of detail of a certain piece of information with the same generic operations and features Perspective-DB offers, e.g. querying and navigation.

The main advantages for the end-users are:

(1) to use the knowledge about how the objects are defined (within the object schema) directly during exploitation - this schematic view is the common basis for the other object views, and

(2) no need to deal with different presentation principles to be able to exploit the most suitable view for the task in hand.

Information retrieved in the Perspective-DB

Basic operations are those carried out to view, insert or remove information in the database. Each basic operation is represented by a gray attribute within an object-type. Specific operation is represented by pink attributes within an object type.

As we have already said, there are two types of basic generic operations. The operations for viewing information are: an instance overview of related instances, a form of related instances, getting related instances, ownership of an object-type and getting help on an object-type.

The operations which insert and remove information are the following: creating new instances, destroying instances, inserting relations, removing relations, modifying values, export and copy.

Graphically, generic operations are presented in capital bold type in a square located in the object type rectangle, beneath the object type's name (see also Figure 5.18).

System architecture in the Perspective-DB

Data models in Perspective-DB might be constructed in a modular structure, by dividing the system into subsystems. A system, as well as subsystems, consists of layers and modules. Subsystems may also incorporate other subsystems. A specialized subsystem can be constructed by incorporating a generic subsystem and adding necessary services. When a specialized system is designed in this manner it is done by stacking subsystems on top of each other in layers.

A layer describes all object types of an object schema on a similar abstraction level concerning the commonality of the objects (Gerhardt et al, 2000). Layers are distinguished by the roles they play in a system. The layer a user has to deal with is the application layer. The application layer describes all objects carrying application semantics. Layers are further decomposed into modules. The highest layer is built by the most specialized information system. The layer beneath provides more generic objects, which are typical for this application domain.

The graphical presentation functionality and support for dynamical behavior is reused from the Graphical Object Management System [GOMS] – see the system architecture overview in Figure 8.6. The frame layer deals with application administration. It contains a distribution and a messaging module to support communication between distributed objects. In one of the system layers, representing the Object Management System, a so-called OMS-Object is a means to function as object request broker. The brokerage functionality is used to realize interfaces with other systems.

Consequences of the modular structure in the Perspective-DB

An existential hierarchy is the main structure which forms a module. The root object of the ownership hierarchy is the module manager, and as such the owner of the module. Via the module manager, a module in its entirety can be handled as an object. Secondary relationship types between objects of the ownership hierarchy belong to the same module.

Between modules, only secondary relationship types may be applied. Any interoperability and interfacing between modules applies this category of relationships (they may or may not be visible to the user). Because secondary relationship types do not interfere with the existential dependency constraints, inserting or removing a relationship between modules does not impact on the consistency of the module. The effect of changes can be kept local.

Information that belongs semantically together will be kept unambiguous and separate from other information. An application can consist of an adaptable combination of orthogonal modules that are inter-operating in a well-defined manner.

If a module has to be updated, removed, or an additional one has to be added, all the interconnections (the secondary relationships between the modules) are located easily and can be modified as necessary. The modular structure related to our Design Tool will be developed in Chapter 8.

Conclusions

At the beginning of this chapter, we presented the basic ideas and overview of *database technology*. We explained *data, the universe of discourse* and generic database properties such as data models at their conceptual, logical and physical level. We also explained entities, attributes and relationships as well as instances, models and schemas with their relationships.

The second part of the chapter showed the main database structure, focusing on three-schema architecture. We also explained database languages in relation to three-schema architecture with their characteristics and specialization. We also explained the concept and types of data independence.

The next part of the chapter was focused on an explanation of database management systems [DBMS's] with all their characteristics, functions, uses, subsystems and criteria for classification. We also presented the limitations as well as advantages and disadvantages of the DBMS.

In the next section of the chapter we presented generations of database models with a historical overview. We explained the main characteristics of file systems, hierarchical systems, network systems and relational systems. We also discussed the differences between them. We pointed out the advantages and disadvantages of each of the systems.

Object-oriented database systems were introduced in the last part of the chapter. First we explained the main idea of the ODMG 3.0 standard with its relationship to object-oriented programming languages. Then we explained the basic principle of the Unified Modeling Language [UML] methodology and its characteristics. We divided the structure of the object-oriented database systems into the main elements: objects, classes and relations with all their subelements, characteristics and relations.

The next part of the chapter was focused on the prominence of *database design*, with a schematical representation of its logic and the relationships between its parts. We also presented the general phases of *database design* and their *functional analysis*. We then explained specific things in object-oriented database designing, such as top-down and bottom-up analysis, candidate objects and state diagrams. This part of the chapter presented a more general platform for understanding logic in conceptual database designing.

We selected the object-oriented database as the main computational support for our Design Tool, and for this reason we focused our investigation more on it. We presented important parts of the *object-oriented data model*, *object-oriented database structure* and the main phases in *object-oriented database design.*

In this chapter we proposed Perspective-DB technology, which we have adopted for developing the object-oriented database model of our Design Tool. Perspective-DB technology provides us with the possibilities to process a complex urban structure and its cognition. We explained the main characteristics of Perspective-DB technology such as object representation (properties, relations and operations). The functionality of the presented technology fits well into our urban and cognitive approach in all its details. At the end of this section of the chapter we explained the main principles of data models and system architecture.

The knowledge of the 'object-oriented database system', examined in this chapter, will be one of the main information theories and models for the development of our Design Tool. In Chapter 8, where we will develop an *object-oriented database model* for our Design Tool, the knowledge presented here will be applied.

The next chapter of the research will be the first chapter in which we will start developing the proposed Design Tool. In this chapter we will set out the framework for applying Lynch's theory. We will make a deep analysis of the four elements of the visual quality and the five elements of the urban environment, which will be essential for developing the database model further. We will also propose a set of requirements derived from Lynch's theory for our Design Tool.

Part three - Conceptual tool design

CHAPTER 6

APPLYING LYNCH'S THEORY TO THE DESIGN TOOL

***Task of the chapter**: Kevin Lynch's 'theory of urban form' contains universal categories, characteristics and elements – psychological as well as physical – of the urban environment, which can be applied in designing the database model and conceptual model of the proposed Design Tool.*

Introduction

In the previous three chapters we examined the theoretical background to the research. First we elaborated Kevin Lynch's *theory of urban form* and then we introduced the concept of *cognitive maps*. We then presented Maar's *computer vision* theory and Ullman's theory of *high-level vision*. The last chapter introduced general theories of database technologies, more specific object-oriented database systems and, at the end, Perspective-DB technology.

There are two tasks to this chapter. The first is to define the position of Lynch's theory in the field of practicing the urban environment, a framework that can guide the application of Lynch's theory in the development of the Design Tool. The second task is to carry out a complete systematization of the physical and psychological elements of the urban elements in their hierarchical order, which can later guide the designing of the computational data model.

The proposed Design Tool can be built through the utilization of the knowledge of the user, the designer and elements of the physical structure of the environment, all in connection with computer technologies. As we have seen in Chapter Three, Lynch's theory provides some important knowledge for the conceptual creation of the Design Tool: (1) the theory of urban form, and (2) the concept of cognitive mapping. For the discovering of the Design Tool we need to investigate this knowledge systematically and put all its elements into a structure which can be useful in later creating the computational data model. The idea of the Design Tool is that its database structure overlaps the structure of the urban environment in all of their details and spatial hierarchy. The urban elements, which Lynch provided in his theory, will be used in all their possible combinations to make the Design Tool as realistic as possible to designers' needs. At the end of this chapter we will set out the tool requirements derived from Lynch's theory.

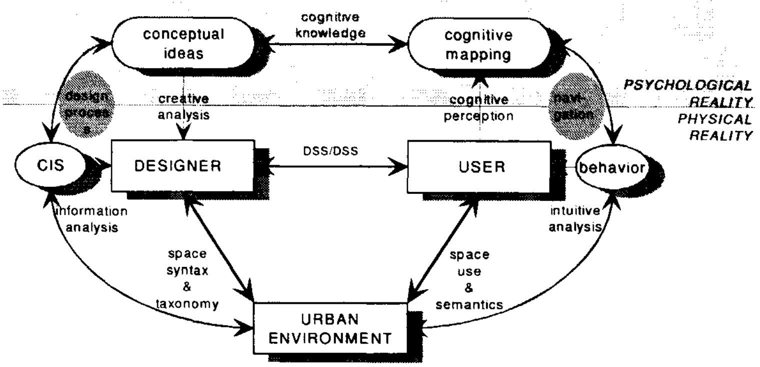
The following chapter will present the Computer Cognition Model of the Design Tool. We will also introduce the new concept of 'microunits' and 'global units', the elements that can build up the image of the urban environment in the Design Tool. The Design Tool requirements, which will derive from the Computer Cognition Model, will also be presented.

The framework for applying Lynch's theory to the Design Tool

In order to apply Lynch's theory to the Design Tool, we must first specify the framework to which the theory will correspond and then investigate their relations and elements. In the case of this research the urban environment consists of physical elements [objects], their users and their interactions.

Interactions between the environment and its users are a complex permanent process, which we called the *practicing urban environment*. The simplified outline of the practicing urban environment, which can be applied in this research, is presented in Figure 6.1. The schema illustrates the relationships between people and their urban environment. People are distinguished in two categories: the designer, who can shape and reshape the urban environment; and the user, who can navigate and conduct himself/herself in the urban environment.

Figure 6.1
Framework for applying Lynch's theory to the Design Tool - schematical outline of the research approach



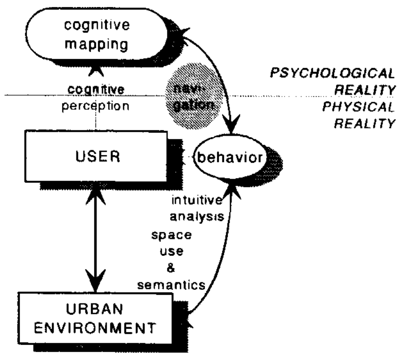
The schema therefore presents the three main elements that are interrelated: the urban environment, the user and the designer. The user and designer should be the same category [the designer *is* the user], but we make a distinction here according to the designer's responsibility to the urban environment as well as to the main task of the Design Tool. In the schema, reality is distinguished into *physical* reality and *psychological* reality. The whole process of permanent interaction between the user, designer and urban environment is located in *physical* reality. Psychological reality is exclusively occupied by cognitive knowledge of the all actors of physical reality [users and designers]. The *psychological* part of the schema is subdivided into the *cognitive mapping* of the user and the *conceptual ideas* of the designer. The processes that are between physical and psychological reality are the *navigation* of the user with the process of cognitive perception and the *design process* with the creative analyses of designer.

The schema also presents a model of the urban interaction process in its hierarchy. The interaction starts with real world information and the processing of it that results in creativity, as a physical/psychological phenomenon in the design domain. Real world information, here called the *conceptual information system* (CIS), is sets of relevant information that add to the process of creativity (the design process). These are the data transferred into the design process, such as survey, spatial, statistical, descriptive data, and so on. The result of the process of creativity is an improvement of an existing urban environment, or a new urban environment. This process is not only a process of considering urban tasks, but also the people [user] for whom the environment is created and built. For example, different design approaches are often required for different audiences. Different design theories developed over the years considered that the process of information transfer runs smoothly.

Lynch's *theory of urban form*, located in the right-hand side of the schema, is presented separately in Figure 6.2. Here we have to refer to Chapter 3, where Lynch's theory and the concept of cognitive mapping were examined. Lynch's theory was built on a few key elements, and they are also presented in Figure 6.2: the urban environment, users and cognitive mapping. According to Lynch, the urban environment consists of physical objects that are located in the environment. The user, who passes [conduct himself/herself and navigates] through the city, perceives these physical objects and memorizes them [cognitive perception] as the personalized image of the city. Sometimes, when a user needs to find a particular object in the city, he retrieves all the information on the object and all related objects around, and then makes a mental path between his position and the particular object. The process of making a mental path between the user and the object is called *cognitive mapping*. The mental path, which can be drawn as a sketch or image, is called a *cognitive map*. A cognitive map is the result of cognitive mapping and it can serve as a *guide* for physical navigation through the city.

As we have shown, Lynch's theory belongs to the given framework of the research, which could improve cognitive knowledge in the design process. Now we must engage all the urban elements from Lynch's *theory of* *urban form* and systematize their hierarchical structure, which will be useful for computational data modeling.

Figure 6.2
Schematic outline of Lynch's *theory of urban form*

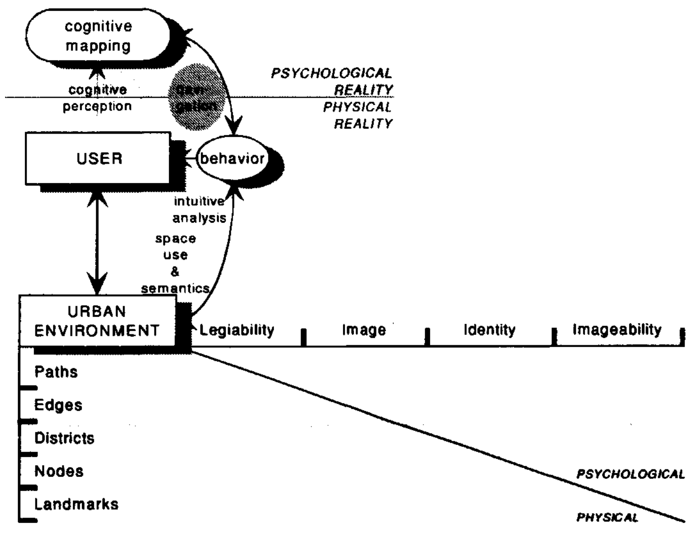


Systematization of urban elements derived from Lynch's theory for the Design Tool

First it should be noted that this exploitation and systematization can refer to Chapter 3, in which Lynch's theory was explained. Let us present Lynch's four categories of images of the urban environment and the five elements of the urban environment that comprise the city images.

Figure 6.3 presents an extension of the existing schema of Lynch's theory (see also Figure 6.2) with elements that build the patterns that make the image of the urban environment as a psychological process of users. These elements are (1) psychological - *legibility, image, identity* and *imageability*, and (2) physical - *paths, edges, district, nodes* and *landmarks*.

Figure 6.3
Physical and psychological elements of the urban environment



Analysis of the four elements of visual quality

Legibility, image, identity and *imageability* are the elements that have the main visual quality in Lynch's theory.

Legibility "By this we mean the ease with which its parts can be recognized and can be organized into a coherent pattern." Or, legibility "can be visually grasped as a related pattern of recognizable symbols, so a legible city would be one whose districts or landmarks or pathways are easily identifiable and are easily grouped into an overall pattern." (Lynch, 1960, pp. 2 - 3). For Lynch, legibility or clarity is crucial in the layout of the urban environment. There is not only the important property of the beauty of the urban environment, there is also the important issue of legibility or clarity for considering urban environments of different size, scale, time and complexity. This characteristic is also important for way-finding or orientation in an urban environment, as a strategic link between environmental images and their mental representations by a user. Good environmental images induce in users mental interpretation as a good feeling, emotional security, and harmony. Lynch also suggests that a kind of mystery or surprise must be granted as a quality. He also writes: "complete chaos without hint of connection is never pleasurable." (ibid. p. 6).

Image "Environmental images are the result of a two-way process between the observer and his environment. The environment suggests distinctions and relations, and the observer - with great adaptability and in the light on his own tasks - selects, organizes, and endows with meaning what he sees. The image so developed now limits and emphasizes what is seen, while the image itself is being tested against the filtered perceptual input in a constant interacting process. Thus the image of a given reality may vary significantly between different observers." (ibid. p. 6). In his text, Lynch distinguishes individual and public images of the urban environments. *Individual images* is later defined as *cognitive mapping* (cognitive mapping is discussed in more detail in Chapter 3). *Public image* is: "the common mental pictures carried by large numbers of a city's inhabitants: areas of agreement which might be expected to appear in the interaction of a single physical reality, a common culture, and a basic physiological nature." (ibid. p. 7). For a city's images, Lynch decided to use the conventions: path, edge, node, district and landmark.

Identity "An environmental image may be analyzed into three components: identity, structure and meaning. It is useful to abstract these for analysis, if it is remembered that in reality they always appear together. A workable image requires first the identification of an object, which implies its distinction from other things, its recognition as a separable entity. This is called identity, not in the sense of equality with something else, but with the meaning of individuality or oneness. Second, the image must include the spatial pattern relation of the object to the observer and to other objects. Finally these objects must have some meaning for the observer, whether practical or emotional. Meaning is also a relation, but quite a different one from spatial or pattern relation." (ibid. p. 8).

Imageability The physical quality of urban elements which relates to the attributes of identity and structure in the mental image "leads to the definition of what might be called *imageability*: that quality in a physical object which gives it a high probability of evoking a strong image in any given observer. It is that shape, color or arrangement which facilitates the making of vividly identified, powerfully structured, highly useful mental images of the environment. It might also be called *legibility*, or perhaps *visibility* in a heightened sense where objects are not only able to be seen, but are presented sharply and intensely to the senses." (ibid. p. 9). There are other basic properties that make beautiful environments, such as: meaning of expressiveness, sensuous delight, rhythm, stimulus, and choice; and the fact that city elements *"operate together, in context"*.

A detailed schematic presentation of Lynch's four main visual quality categories of the urban environment is given in Table 6.1.

Table 6.1
Lynch's four categories of images of the urban environment

URBAN ENVIRONMENT	Legibility	Image	Identity	Imageability
Paths	pattern	individual: - cognitive mapping	recognition	shape
Edges	symbols	public:	specification	color
Districts	urban scale	- paths	pattern relations	arrangement
Nodes	size	- edges	meaning	expressiveness
Landmarks	time	- districts		sensous delight
	complexity	- nodes		rhythm
	visual sensations	- landmarks		stimulus
				choice

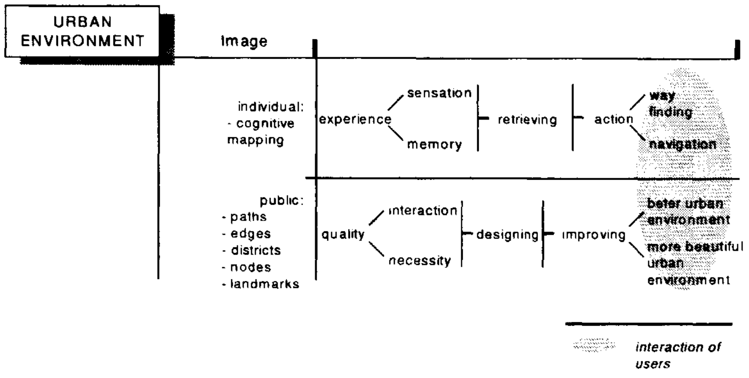
Lynch's *theory of urban form* is, more or less, focused on the *image* of urban form (see Table 6.2).

Table 6.2
***Image* as a focal point in Lynch's theory**

URBAN ENVIRONMENT	Legibility	Image	Identity	Imageability
Paths	pattern	individual: - cognitive mapping	recognition	shape
Edges	symbols	public:	specification	color
Districts	urban scale	- paths	pattern relations	arrangement
Nodes	size	- edges	meaning	expressiveness
Landmarks	time	- districts		sensous delight
	complexity	- nodes		rhythm
	visual sensations	- landmarks		stimulus
				choice

For Lynch, clarity or legibility in the city setting is crucial for identifying the environment. Identifying and memorizing the elements and patterns of the environment are important processes for way-finding by an individual. The image is the product of both immediate sensation and the memory of past experience, and it is used to interpret information and guide action. The image has wide practical and emotional importance for the individual. How the individual perceives and memorizes an urban environment, how he retrieves and uses the information in action is a process of *cognitive mapping*. Cognitive mapping is a process of mental structuring of spatial data by the individual located in a particular place and time. Cognitive mapping is not about maps, it is an accumulation or summary of experiences and schematic knowledge that a person has about a familiar environment. Cognitive mapping is a mental representation of a spatial environment formed in a person's mind (see Table 6.3).

Table 6.3
Parallel reflections between cognitive mapping and urban elements



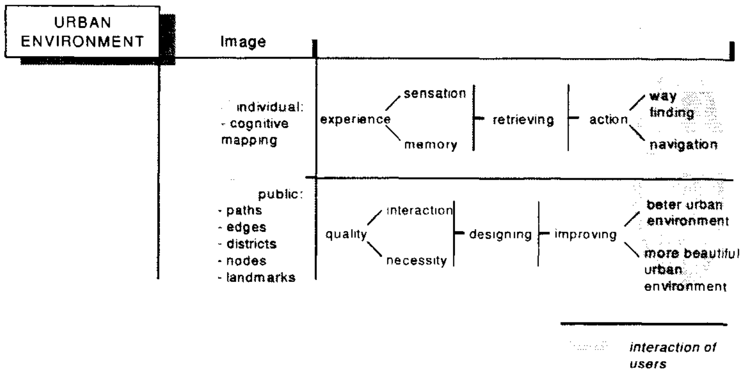
The *public image*, as we have already seen, is the common mental picture carried by large numbers of a city's inhabitants. It interests city planners who aspire to model an environment that will be used by many people.

Lynch summarizes the form quality, which directly concerns the design process of new urban elements (Table 6.4), with the following categories:

- singularity, or figure background clarity: sharpness of boundary, closure, contrast of surface, form, intensity, complexity, size, use, spatial location etc.;
- form simplicity: clarity and simplicity of visible form in the geometrical sense, limitation of parts, symmetry etc.;
- continuity: continuance of edge or surface, nearness of parts, reparation of rhythmical interval; similarity, analogy, or harmony of surface, form or use;
- dominance: of one part over others by means or size, intensity, or interest, resulting in the reading of the whole as a principal feature with an associated cluster;
- clarity of joint: high visibility of joints and seams; clear relation and interconnection;
- directional differentiation: asymmetries, gradients and radial references which differentiate one end from another; or one side from another; or one compass direction from another;
- visual scope: qualities which increase the range and penetration of vision, either actually or symbolically. This includes: transparencies, overlaps, vistas and panoramas that increase the depth of vision; articulating elements that visually explain the space; concavity, clues etc.;
- motion awareness: the qualities which make sensible to the observer, through both the visual and kinesthetic senses, his own actual or potential motion etc.;
- time series: series which are sensed over time, including both simple item-by-item linkages, where one element is simply knitted to the two elements before and behind it etc.; and
- names and meanings: non-physical characteristics, which may enhance the imageability of an element. Names are important for crystallizing identity etc.

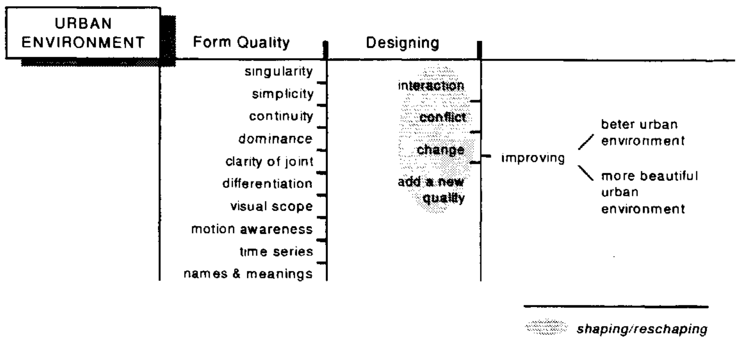
All these qualities do not work in isolation. They are in a state of interaction or conflict. They are in some kind of established relations, which can be changed by adding a new quality to environmental elements (Table 6.5). This operation must proceed at an entirely different scale of space and time.

Table 6.4
***Form quality* as a focal point in the design process**



The shaping or reshaping of the urban environment should be guided by what might be called a 'visual plan' for the city or metropolitan region: a set of recommendations and guidelines which would be concerned with the visual form of the urban environment (see also Table 6.5).

Table 6.5
Relationship between form quality, design process and improving



Analysis of the five elements of the urban environment

The five main elements of the urban environment are *paths*, *edges*, *districts*, *nodes* and *landmarks*, and derive from Lynch's analysis of the effects of physical, perceptible objects. Now we will go deeper into an investigation of the types and characteristics of the elements. This investigation is also important for focusing on the relationship between these elements, which will be helpful later in the computational modeling of the Design Tool.

The main types and characteristics of the five elements of the urban environment are presented in Table 6.6.

Paths are the main element that characterize an urban environment. They have their own identities and characteristics, mostly in terms of elements or functions which may or may not be present in them. Thus, paths have their own physical characteristics, activities and relationships with other urban elements and so on.

Table 6.6
Five elements of the urban environment and their types and characteristics

URBAN ENVIRONMENT	Legiability	Image	Identity	Imageability	
Paths	streets	walkways	transit lines	canals	railroads
Edges	boundaires between paths	shores	reilroad cuts	edges of development	wals
Districts	medium	large			
Nodes	junction	braik in transition	crossing/convergence of paths	shift of structure	concentrations
Landmarks	building	sign	store	mountain	

In this research we will use two main categories of paths, which relate to their importance in analyzing an urban environment and also in the *cognitive mapping* of the environment: *dominant* and *marginal* paths.

Dominant paths are those which, through their physical and other characteristics, are crucial to an environment and the users' physical and psychological correlation in/with them.

Marginal paths are, of course, the exact opposite to dominant paths: boring, narrow, empty and so on.

A *street* is the first element characterized as a path. In Table 6.7 we have put together almost all the characteristics and relationships of streets, ranging from physical to psychological characteristics. Other possible characteristics or relations that are not crucial to this research were not noted here.

Table 6.7
***Streets* and their characteristics and relationships**

URBAN ENVIRONMENT	*Physical*					*characteristics*
	Categories	Profile	Objects	Specificity	Dispositions	Trafic directions
dominant	wide: - boulevard - avenue - main street - ...	one or two side buildings rich with - trafic - pedestrian paths - greeneries	big buildings highrise buildings *exceptional:* small and low buildings	characteristic facades structure of pavements greenery	in: - city centre (downtown) - regional centre - suburb	two ways *exceptional:* one way
marginal	narow: - width street - side street - passages - ...	simple structure (two side buildings)	small buildings low buildings *exceptional:* big and highrise buildings	uniform facades structure of pavements *exceptional:* greenery	all over the city, mostly in 'historical centre'	one way or two ways

relationships						*Psychological*	
Continuity	Crossing	Activities	Visual exposure	Differencies	Direction changing	Colinearity	Distances
followed *rerly:* changing the wideness	simple: - crossing complex: - square - rotonde - passarela	concentration of activities heterogenuous *exeptional:* dominance of one activity	by itself: - partly - in totality other parts of the city	visible predictable	little visible predictable	foreseeable	foreseeable
no-continuity cutting or changing directions	simple: - vacancy *exeptional:* simple or rich passarela	without activities *exeptional:* few different activities	by itself: - partly - in totality	difficult for perceiving	non-visible non-predictable	non-foreseeable	difficult for foreseeable

To make a more complex analysis of all the characteristics or relationships is, of course, possible, but in our research we will concentrate only on those which will help us later in the computational process. Some of the previous characteristics and relationships from Table 6.7 are also useful in analyzing all other elements of paths. In the next few tables, we will categorize some more specific characteristics and relationships for each of the other elements of paths, such as: *walkways, transit lines, canals and railroads.*

Walkways are paths used by people for walking and social tasks. These walkways are mostly located in *downtown* or *historical* centers of cities. Sometimes these walkways are also passages, ways between two or more streets, some are small pedestrian streets and almost always they are attractive areas for users. Walkways are always out of bounds to all traffic, without parking lots, and they are small oases within cities. Such paths are also found in big shopping centers, and can also serve as entrances to parks. All of the different characteristics and relationships are presented in the Table 6.8 below.

Table 6.8
Walkways and their characteristics

	Physical		*Psychological*
WALK WAYS	Dispositions	Trafic (forbiden)	Attractiveness
dominant	city center (downtown) historical center regional center passages	pedestrian bicycles delivery (in a certain limited time)	very rich (different acctivities)
marginal	passages shopping centre park entrances	pedestrian bicycles	from rich to poor

Transit lines are the paths for vehicle traffic that pass through cities. Transit lines can be disposed in cities' peripheries as city rings or they can pass through cities close to their centers. Sometimes, when transit lines have to pass through central parts of cities, they can go to another vertical level, for example using tunnels or viaducts. Transit lines are highways or roads with very dense and frequent traffic. In some cities, public transport, as transit lines into the cities, offers high-speed bus and taxi roads. These are always kept separate from other traffic and they are on the ground level of the cities. Sometimes, they pass briefly through tunnels, and very occasionally onto viaducts. All these characteristics are presented in Table 6.9.

Table 6.9
***Transit* lines and their characteristics**

TRANSIT LINES	Physical		
	Categories	Dispositions	Objects
dominant	highways main roads high-speed bus & taxi roads	city's transversal city's ring through cities	ground lewel roads viaducts
marginal			tunnels

One of the paths from Lynch's five elements of the urban environment is *canals*. Canals are a very specific kind of path: users cannot walk or cross through them. Canals are mostly for water transport, tourist travel, recreation or for industrial uses. In some countries such as Holland, canals dominate city layouts. In some cities, there are no canals at all. Some cities have only a few, mostly artificial and for various tasks. In Table 6.10 we categorized canals as a path in urban areas.

Table 6.10
***Canals* in cities and their characteristics**

CANALAS (RIVERS)	Physical		
	Categories	Dispositions	Objects
dominant	big (main) canals canals	in a city's center through cities	ground lewel cannals canals on viaducts
marginal	small canals industrial canals	transversal (peripherial)	canals in tunnels

The last element of paths is *railroads*. Railroads constitute an important pattern in almost all cities. Railroads, as a type of traffic path that requires special treatment, are located in city center localities, sometimes in the very center of cities. The end or beginning points for each railroad are rail stations. Rail stations occupy large areas in the city, but a little further on from the station, railroads become narrow, fixed corridors in the cities. They may have one or two tracks, sometimes many more. In any case, railroads are mostly located on the ground level of cities; sometimes they are underground, and very occasionally on viaducts. Railroads are protected from all other city activities. Main railroads are very frequently used, as opposed to industrial railroads, which are always located in the suburbs of cities, and are used much less frequently. In Table 6.11 we present a categorization of railroads.

Table 6.11
***Railroads* and their characteristics**

RAILROADS	Physical		
	Categories	Dispositions	Objects
dominant	high speed railroads main railroads	in central city areas city's ring through cities	ground level railroads railroads on viaducts
marginal	industrial railroads	transversal (peripheral)	railroads in tunnels

Metros or 'undergrounds' are also elements that are part of the urban environment, part of the urban *path.* They have many similarities to railroads, but metros are local, fast transport through big cities. They are generally in tunnels under the ground, in some places they are on viaducts, occasionally they are on the ground level of cities and sometimes they use a combination of all. Metros always have two (never more) narrow tracks. Like railroads, they are isolated from all other activities in cities. Metros are symbols of modern, big cities. No table is given for metros, because such a table would be almost the same as Table 6.11.

Edges are the second element which characterize an urban environment in Lynch's theory of *urban forms.* Edges are a more abstract and mental category related to certain physical elements such as paths, shores or railroads. Edges are the boundaries between what may sometimes be different urban elements, environments, gradations, characteristics and

functions. Edges are mostly invisible, but in some cases they are very strong, dominant and powerful. In Table 6.12 the main characteristics of edges are presented (see also Table 6.6). This analysis is based on the two dominant characteristics of edges: *strong* and *weak* edges.
A *strong edge* is not only visually prominent, it is also continuous in form and impenetrable through cross movement.
A *weak edge* is, of course, the exact opposite, being far less visually prominent, with no continuity in form and is fully penetrable.

In this analysis we improve upon Lynch's types of edges with a new element: *canals*. Canals are in many cases a very important element, and should be analyzed along with all other important types of edges.

Table 6.12
***Edges* and their characteristics and relationships**

	Physical					
EDGES	Boundaries between paths	Shores	Railroad cuts	Edges of development	Walls	Canals (rivers)
strong	prominent continuous low impenetrable fragmentary	prominent continuous fixed impenetrable	prominent continuous fixed impenetrable	temporal fragmentary uniting seams	continuous impenetrable isolation	prominent continuous fixed low penetrable possible uniting seams
weak	less prominent less continuous penetrable uniting seams			temporal fragmentary uniting seams	less continuous possibly penetrable uniting seams	impenetrable

District is the next element that Lynch uses to characterize an urban environment. Districts are the parts of cities in which people (users) could *walk in* or *walk out* of and would know where they are. They know where they are due to the specific characteristics of that particular part of the city. Every district is different, but here we will use two main characteristics with which we will classify all districts: *large* and *medium*.
Large districts are those parts of big cities in which it is possible to recognize some *subregions* by their characteristics. Subregions are small parts of a district, like a block or a unit, which, with their small differences between each other, make a pattern characteristic to this particular district. The structure of a large district varies from complex to very simple, depending, of course, on the city.

Medium districts are parts of cities, large or small, with typical and homogenous characteristics, with generally simple structures. In some cities this structure could also be complex, but on a smaller scale.

In Table 6.13 below, we categorize the characteristics of districts and their relationships. Almost all characteristics and relationships apply to both *large* and *medium* districts, and only in structure and orientation is it possible to find opposing tendencies.

Table 6.13
***Districts* and their characteristics and relationships**

DISTRICTS	*Physical* →						*Psychological*
	Types	Structure	Boundaries	Particularity	Identity	Orientation	Function *(main)*
large	homogenous introvert extrovert chaotic isolated class overtones ethnic	complex *(subregions)* ↓ simple	hard definitive precise soft uncertain without boundaries	strong character historical context topographical features successful intervention in space	physical characteristics names of districts people other things *(sound, smell)*	difficult ↓ easy	business shopping culture administratio living recreation storages industry
medium		simple ↓ complex				easy ↓ difficult	

Nodes are sometimes large, sometimes small parts of cities in which people can enter or change transportation. Nodes can be corners or city squares with intensive activities. A node can also be a railroad station or big metro station above an interchange of several metro lines. A node may also be a small residential area with a local, largely introvert character. Nodes are a strategic part of all cities, and we will categorize nodes as *frequent* or important, and *less frequent* or less important.

Frequent nodes or important nodes, are nodes with many different and always intensive activities in what can be either a small or large focused city space.

Less frequent or less important nodes, are nodes mostly located in quiet city zones or suburbs, with introvert characteristics and, in almost all cases, focused in small city areas.

Nodes are important for an analysis of the urban environment as strategic points from which, and within which, some city activities can be changed or can be in transition. In Table 6.14, we categorize the characteristics and relationships of nodes.

Table 6.14
***Nodes* and their characteristics and relationships**

NODES	*Physical* Categories	Types	Forms	Dispositions	Characteristics	*Psychological* Thematic concentration
frequent	small points (corners) large square extended linear shapes central district whole cities	junction break in transportation transition beetwen structural units railroad stations rivers and canals	corner square clear form shapeless slippery	city center central city areas	unique intensive extrovert	railroad station commericial concentration different concentratio ns
less frequent	small points corner square	junction break in transition	corner square shapelless slippery	suburb	introvert	metro *(subway)* residental

Landmarks are external reference points for users that are physical elements dominating a space in some way. They vary in their locations, shapes, sizes, heights, colors and so on. Landmarks are always elements which are in some sense dominant in their surroundings. They help users to memorize a particular space and help them to navigate.
We define landmarks with two main categories: *distant* and *local.*
Distant landmarks are physical objects which are visible from a long distance, and which necessarily dominate their surroundings, such as on the city skyline.
Local landmarks are landmarks which are visible from a short distance, and which are important for local navigation and way-finding.
In Table 6.15, we give more of the characteristics and types of landmarks.

An analysis of urban elements and their relationships is important for making a visual interpretation of the city. We already explained the relationship within each of the urban elements and we analyzed elements individually. But how do these elements and their relationships behave between each other? How does a group of different elements work together? How do their interrelations build the image of the city? We could answer these questions by analyzing the interrelations between five main urban elements, but it would take much effort. For the tasks of the Design Tool we do not need such analyses; instead, in Table 6.16 we show these

Table 6.15
***Landmarks* and their characteristics and relationships**

	Physical					*Psychological*
LANDMARKS	Categories	Forms	Contex	Dispositions	Visibility	Identified
distant	uniqueness specialization singularity continuity	clear contrast with background singularity	associated activity historical meaning sign meaning	junction square street park other	from any locations	pleasant iritating easy
local					from nearby	pleasant iritating easy sound and smell

interrelations in a schematic way. This basic *typology,* a graphic interpretation of the correlation between urban elements, is the starting point for the concept of 'global units'. We will explain this concept in the next chapter.

In the scheme, some elements, such as paths and edges, are almost the same in their interrelations. In our case, we will present only the relations between main elements. The entire range of possible permutations and implementations from combinations of the main elements could be presented for specific tasks in the computational process. The intersection of the same elements, such as path and path, is presented in the forms or types of the paths.

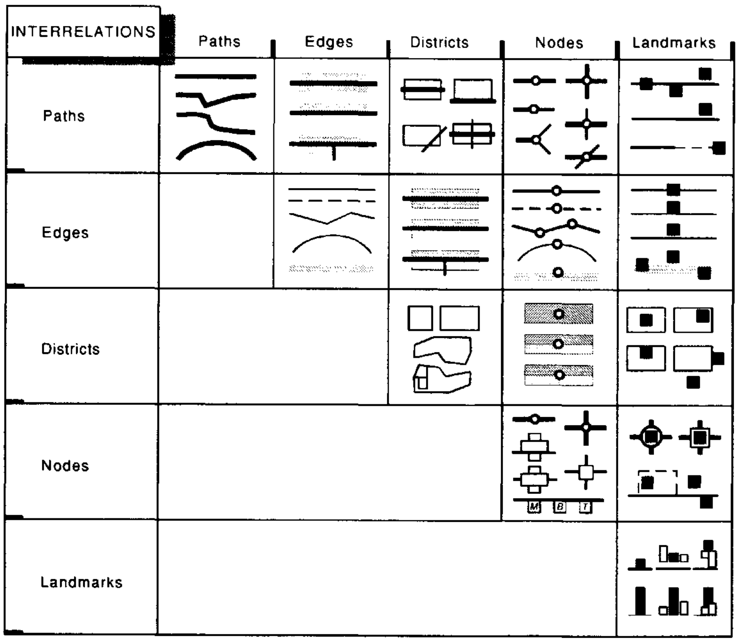
Graphical (schematic) presentations of interrelations are based on simple visual elements as a generic *typology* of the main urban elements. This typology symbolically represents the data of the *real world.* We will discuss this in more detail in the next two chapters.

The recognition of an object from the generic *typology* is as much dependent on context as on the form of the object itself. In his study, Lynch also defined all urban elements as arranged and interrelated in four structural stages:

(1) the various elements are free; there is no structure or interrelations between parts,
(2) the structure becomes positional; the parts are roughly related to each other,
(3) most often, the structure is flexible, and
(4) as connections multiply, the structure tends to become rigid.

This characteristic of structure might be applied in different ways at different levels of *imageability*, which can be automatically processed in the Design Tool.

Table 6.16
Interrelations between main elements of urban environments - generic *typology of urban elements*



How to apply Lynch's theory to the Design Tool

Lynch's theory will be applied dynamically to the Design Tool at the levels of:

(1) designing the database structure, and

(2) designing the tool.

Designing the database structure is the process of applying the *four elements of visual quality* and the *five elements of urban environment* in the hierarchical structure of the 'object-oriented database model' of the Design Tool (which will be examined in Chapter 8).

Designing the tool is the process of applying the knowledge of Lynch's *theory of urban form* and the concept of *cognitive mapping* in the 'conceptual model' of the Design Tool (this will be examined in Chapter 9).

Knowledge derived from Lynch's *theory of urban form* and the idea of *cognitive mapping* will form the basic platform for the conceptual modeling of the tool, and later the tool designing. This will be done with consideration of what makes an environment good visually, according to:

(1) remembrance,

(2) where we are,

(3) where is X in relation to Y,

(4) how X will find Y,

(5) alternative paths between X and Y, etc.

From Lynch's theory we have derived the four elements of visual quality and the five elements of urban environment, but how to interrelate these aspects and how to quantify the visual quality from them is not yet clear. This thesis will only provide the computational ingredients to recognize these elements from images of an urban scene. Further research will have to provide the conversion of these quantitative notions into measures of visual urban quality.

By testing the 'conceptual model' of the Design Tool we will indeed test the applicability of Lynch's theory for the Design Tool. Lynch's theory will be applied in a case study analysis of the campus of TU Delft, which will be presented in Chapter10.

Tool requirements derived from Lynch's theory

Through the deep investigation of Lynch's *theory of urban form* and the concept of *cognitive mapping*, we discovered some important characteristics that will be implemented in the process of the conceptual modeling of the tool, as well as in the designing of the tool.

The tool requirements derived from Lynch's theory should be as follows:

- applicable to different characteristics of urban environments such as geo-political, cultural, natural, social, physical, etc.,
- applicable to different types of urban environments such as small or medium-sized, dense, spread out, etc.,
- applicable to different types of input and output data (multimedia data),
- able to make both a global and detailed analysis,
- able to include the surroundings of even the smallest unit for analysis,
- able to analyze in different spatial scales and measurement systems, and
- able to make analyses of 'static' objects (maps, photographs, drawings).

These requirements will be implemented in different phases of the process of tool designing.

Here, the proposed requirements are the first set of requirements for the Design Tool, as derived from Lynch's theory. In Chapters 7 and 8 we will present the next sets of requirements for the Design Tool, related to cognitive science and the object-oriented database. These three sets of tool requirements will be the platform for the conceptual modeling of the Design Tool.

Conclusions

This chapter explored a part of Kevin Lynch's 'theory of urban form' which contains universal categories, characteristics and elements, psychological as well as physical, of the urban environment which can be applied to designing the *database model* and *conceptual model* of the proposed Design Tool.

To apply Lynch's theory for creating the Design Tool we choose two aspects of the theory: (1) the elements that constitute the *visual quality* of the urban environment, and (2) the idea of *cognitive mapping*. In this chapter we explored the first aspect; the second aspect, concerning cognitive mapping, was explored in Chapter 3.

At the beginning of this chapter we set out a framework for applying Lynch's theory to the Design Tool, defining the position of the theory within the context of this research as well as the Design Tool itself.

Second, we systematized the urban elements derived from Lynch's theory for the Design Tool. We then analyzed the four psychological elements related to the visual quality of an urban environment. After that, we analyzed the five physical elements of the urban environment and their hierarchical order. We also set out the generic schematical typology of the relationships between the five main elements.

In this chapter we also showed how to apply Lynch's theory to the Design Tool and at the end we gave the sets of requirements which derived from Lynch's theory.

We did not apply in this research and the Design Tool the aspect of Lynch's graphical notation. We also did not apply the practical aspect of cognitive mapping, the interview of users, their cognitive mapping drawings or analysis of them. These aspects of Lynch's theory are not relevant for our research and the Design Tool.

This chapter presents the results of an investigation into universal categories, characteristics and elements of the urban environment, and makes a precise systematization in a form that can be comprehensible and applicable for designing the *database model* and *conceptual model* of the Design Tool.

The next chapter will concentrate on cognitive science with a greater focus on computer cognition. The Computer Cognition Model, which will be utilized in this chapter, together with knowledge derived from cognitive science, will be an important part of the *conceptual model* of the proposed Design Tool.

CHAPTER 7

THE COMPUTER COGNITION MODEL OF THE DESIGN TOOL

***Task of the chapter**: the Computer Cognition Model is an important part of the proposed Design Tool, providing the 'recognition', 'filtering' and 'synthesis' processes, and is to be based on the various knowledge derived from the theory of urban form and cognitive science.*

Introduction

In the previous chapter we started developing the components of the Design Tool related to the various aspects of the theoretical background. The previous chapter developed the systematical hierarchical structure of the elements of the urban environment, which are important for the further development of the database model of the Design Tool. This hierarchical structure is also important for the visual quality and cognitive mapping of the urban environment, which will be integrated in the development of the *conceptual model* of the proposed Design Tool.

In this chapter we will develop the Computer Cognition Model of the Design Tool. The Computer Cognition Model will be a part of the *conceptual model*, which will be responsible for the 'recognition', 'filtering' and 'synthesis' processes. The development of the Computer Cognition

Model will be based on the cognitive science knowledge and the computer recognition aspect. Computer recognition [or *shape recognition*] will be incorporated in our Design Tool as a plug-in application adopted from Ullman (see Chapter 4). The Computer Cognition Model will be the first functional part of the *conceptual model* of the Design Tool and it will be logically connected to other modules of the model.

Both theories of urban form and cognitive science tend to concentrate strongly on cognitive mapping analysis for particular urban sites by using complex computational systems. The main reasons for this are the availability of data, both from the environment and from cognition, and the fact that input analysis systems are more ordered. The analysis of an urban environment by cognitive perception seems to be much more interesting and important, yet also difficult to investigate. Urban environment and cognitive perception models that have been suggested by architects and urban psychologists are mostly theoretical and unrealistic as tools for designers.

The Computer Cognition Model of the Design Tool, as we will propose here, will be assumed to be as realistic as possible, both from a cognition and implementational point of view. The result will be a qualitative model, with quite defined features. It will reflect the main characteristics of the 'real system' and will offer a plausible explanation to some of its main requirements that will be given at the end of this chapter.

In the next chapter we will develop the object-oriented database model of the Design Tool. The object-oriented database model will constitute crucial support for the modules of the *conceptual model* of the Design Tool.

Computer modeling in cognitive science

Computation is the central concept for the foundations of modern cognitive science. It is now commonplace to use computers to model human cognition. Computation provides a general framework for exploring cognitive processes and behavior. A bridge between mathematical theory of computation and cognitive science and AI, which ultimately deals with physical systems, is required. We need a system to *implement* the relation that exists between an abstract computational object and a physical system. Implementation may have cognitive and therefore syntactical properties. A physical system implements a given computation when the structure of the physical system mirrors the formal structure of the computation.

The central claim of the cognition model, according to Chalmers (1997), can be justified by dividing mental properties into: *psychological* – such as belief, learning and perception, and *phenomenal* – i.e. those that are characterized by the way in which they are consciously experienced. Psychological properties are concerned with what the mind *does* and phenomenal properties are concerned with the way it *feels*. But a computational model is *just* a simulation. If a computational simulation is appropriately designed it will share the casual topology of the system that is being modeled, so that the system is not simulated but *replicated*. Some computational frameworks can explain and replicate the human cognition process. Many researchers regard a computer implementation of a model as an important test for its suitability for describing the internal mechanism of human cognition. The reason computer models are regarded as a useful test for cognitive models is that for a model to actually run on a computer, it must be fully specified. Since the models of human cognition are necessarily complex, the danger of under-specification is large.

Testing the models on computer seems to constitute an appropriate test. The model generates the same results as the system being modeled, and this strongly supports the hypothesis that the modeled system operation is based on the same principles as the model. When constructing models for the operation of human cognition, the model space is huge, and it is essential to restrict it as much as possible. Without additional restrictions, the correlation between the behavior of human cognition and some specific model cannot be used as evidence for the correctness of the model. In a large model space, some models will replicate the behavior of the system in some situations by chance.

Considering the size of the information of human behavior, and the size of model space for intelligent systems, checking all of these models would take an infinite amount of time. It is therefore essential to put more constraints on the model space. The constraints of our research come from looking at cognitive mapping. We can claim that cognitive mapping is like a computer in that it is an 'information processing system', and to that degree of generality this is true. However, a closer look immediately reveals fundamental differences.

A model of cognitive mapping is necessarily limited to a relatively small number of items, which can form a 'simple system'. In general, models that run on computers must be fully specified for computers.

Another characteristic of cognitive mapping is that it is dynamic, i.e. any change in its functionality, in particular learning by perceiving, is done solely by the brain with guidance from an external source. Thus, when the model is designed for an activity of human cognition, we are trying to mimic *complex self-adaptive* or *self-learning* systems.

The belief that the simple model is likely to be correct for the complex system is based on two assumptions:

(1) the complex system performs the operation in a simply way, and

(2) there is a very small number of simple ways of performing the operation.

The second assumption is reasonable, but the first one is not, especially for self-learned operations.

A possible advantage of simplicity is a reduction in the number of items, which will make a simple sequence more economical. Simplicity is important because it allows the external designer to evaluate changes in the system more easily.

Two main conclusions should be applied to a computer model of an operation before it is accepted as a possible model for human cognition:

(1) to put very severe constraints on possible models, thus significantly increasing their probability of being correct, and

(2) for self-learned operations, there cannot be a simple model because their implementation is not simple.

The last possible general conclusion is that it is difficult to find a useful generic model of the operations of human cognition. But computation is sufficiently flexible that it can capture almost any kind of organization, whether the casual relations hold between low-level neural processes or among high-level representations.

Basic ideas of the Computer Cognition Model

The Computer Cognition Model will be based on the idea of performing analyses of some visual input data in the Design Tool. To develop the Computer Cognition Model, the fundamental differences between environment and cognition are important. The concept of analysis of input data is also important for this model and will be elaborated here. Input data analysis has evolved over a long time to achieve the best performance in its task; environment and cognition could not do that because their tasks are too variable.

Input data analyses are specialized systems to deal with specific tasks. It is not possible to predict the level of specialization, but it is not surprising to find specialization even in very similar tasks. A specialization means a pattern of connections between urban elements, which is spatially defined, or at least is very constrained. For example, the recognition of some similar tasks may be nevertheless implemented by different systems.

In an environment the situation is different, particularly in an urban environment. If we are looking for different urban elements they cannot be generalized by specialized systems. But an urban environment can be divided into different categories and characteristics, as defined in Chapter 6. It is almost the same with cognition. Even if we look at very different cognitive tasks they cannot be executed by specialized systems. This is true for almost all cognitive tasks. The conclusion might be that the system behind cognition could not be specialized for the task it is executing. Instead, the system must be a generic system specialized in the task of recognizing new 'objects' and their spatial behavior, which can be urban elements as explained in the previous chapter. This generic system can also 'learn' new 'things' such as properties of the urban elements, their associations, values, and so forth, all from an object-oriented database system [OODBS].

We do not believe that there is a sharp border between urban environments, cognition and input data analyses, and there are probably regions where specialized systems are mixed parts of the generic system.

The features of the Computer Cognition Model

Here are the features of the model that we will try to explain:

(1) the model of the tool deals effectively with a large number of *different* (urban and cognitive) elements,

(2) the model can form associations between urban and cognitive elements,

(3) the model is robust and degrades gracefully; in general, urban elements are associated with a specific location in the system and cognitive elements too,

(4) the operation of the model is precise and it is possible to repeat an operation exactly,

(5) the model can learn the new procedure from the input (output) data,

(6) the model can learn to deal with complex situations, when the outcome is not obviously associated with actions that lead to it,

(7) the model can recall and manipulate information from its history; there is no limitation on the type of information that can be recalled, it can recall its own activity,

(8) the model tends to recall *segments* from the near past; that means groups of elements which had been in effect at the same time; these include internal spatial elements or concepts, intentions etc.,

(9) the model can be limited in the number of elements that it can deal with at the same time; the elements which are dealt with at any time are specified by one of:

- the urban elements as units which the model is dealing with,
- the cognitive elements the model has been dealing with until this time,
- some input data, and less frequently
- the model deals with other elements inside of themselves (such as self-learning),

(10) the *user* is aware of a *part* of the model's operations , but not all; the part which the user is aware of can be directly recalled later from the model (free recall),

(11) the model has a strong tendency to take actions which benefit the user,

(12) the model has internal states; some of them tend to arise as a response to specific tasks and can lead to appropriate responses;

other internal states are not associated with any specific tasks or responses,

(13) the model can be good at visual recognition, analysis of elements, cognitive mapping and way-finding, and not quite so good at computational output representation.

We did not include in this list the following, more cognitive features: free will, intention, the ability to learn automatic processes and short-term memory. They are in a way included in the presented features.

Basic assumptions of the Computer Cognition Model

The basic assumptions of the model are presented as follows:

(1) the model is implemented by urban elements mainly in urban typology,

(2) the actual operations are done by comparison of generic urban elements stored in the model by activating/inhibiting other input urban elements (the psychological/mental elements may also take some part),

(3) the number of inputs and outputs of each analysis is large, but much smaller than the total number of stored elements,

(4) the amount of information in the activation state of a single unit is limited, corresponding to the number of possible available data from the OODBS,

(5) learning is done by changing the strength of the connections between main elements,

(6) all elements are accessed by their address (code), which is the way data is accessed in a computer – although for some specific, mostly psychological elements, there is no way for this,

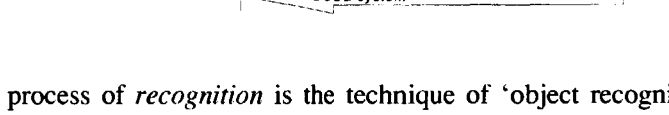
(7) at any given time, the number of active elements compares to the total number of specific active elements.

Description of the Computer Cognition Model

The Computer Cognition Model is the part of the conceptual model of the Design Tool that will be explained in Chapter 9. The structure of the Design Tool will be modular because of the complexity of the tool. The Design Tool will consist of several parts, whose modules are not yet defined, each related to one of the tool's main functions. All modules of the tool can interrelate to each other. The advantage of the modular structure of the Design Tool is the possibility that each module can be examined separately in all necessary investigations. The disadvantage, on the other hand, is that a global view of the tool can be constructed from the separate modules. The Computer Cognition Model consists of three main modules (presented in Figure 7.1):

(1) recognition,
(2) meta level, and
(3) synthesis

Figure 7.1
Scheme of the *Computer Cognition Model* and its modules



The process of *recognition* is the technique of 'object recognition', as developed by Ullman, which was explained with important details in Chapter 4. The recognition process is completely adopted in its autonomy as a plug-in in our system. The process of recognition is the first module in our Design Tool. The adopted technique of object recognition comes directly from cognitive theory, from 'computer recognition' and 'high-level vision'.

The second part of the model is called the *'meta level'* which is located between the process of object recognition and synthesis. The meta level links and unifies the process of recognition with the process of synthesis via the smallest elements of the urban environment called *microunits*, performing the function of a 'filtering process'. In the following sections of this chapter we will explain this part in more detail.

The third part of the cognition model is the process of *synthesis*. Synthesis is the process of composing an adequate image through *matching* and *mapping* urban elements by forming *global units*. Global units are elements higher than micro units and consist of the information from the stored database system belonging to particular urban elements (environment of the analysis). This part will be also explained in the following sections of this chapter.

'Meta level' module

The position of the *'meta level'* module in the conceptual model of the tool is between the *object recognition* module and the *synthesis* module. Figure 7.2 presents the *micro scale interaction* process through the *Computer Cognition Model.* The *meta level distribution* is one type of filtering process. Each input image consists of different urban elements called *objects*. Each object (urban element) has its own properties stored in the object-oriented database system. The properties are different and from different sources, from spatial GIS data to statistical alphanumeric data, represented in different forms.

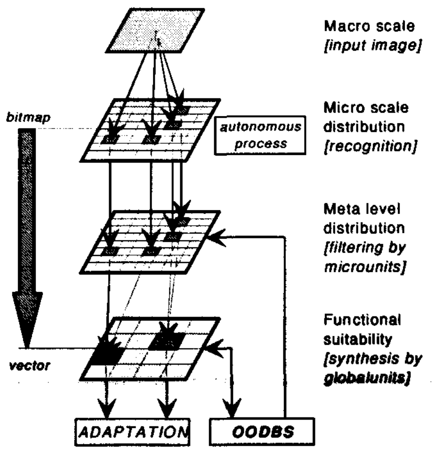
The input image (*macro scale*), after finishing the process of recognition (*micro scale distribution*), is in a weak shape, lines are often blurred and lacking sharpness. The 'recognized objects' from the input image cannot be analyzed by activating all their properties stored in the object-oriented database system [OODBS]. It is also not possible to manipulate shapes of the object inside the image. This image, after the process of 'object recognition', is a bitmap image. For these reasons, each image must be filtered after the process of object recognition. The main tasks of the filtering process are:

(1) to make the image sharp,

(2) to convert the bitmap image into a vector image, and

(3) to prepare the image for the process of synthesis.

Figure 7.2
Micro scale interaction through the *Computer Cognition Model*



Having a sharp image is important for the process of object manipulation inside the image, as well as for comparison with the original input image in 'layer0' (see Chapter 9 for more about '*layer0*'). Sharpness of the lines is also important for the precise definition of each object inside the image, especially in a densely patterned urban environment where it is usually difficult to define separate objects.

Input images are mostly, but not necessarily, bitmap images. It is very important for the process of object manipulation that all objects (in the image) can be manipulated (and also deleted or imported). For these reasons, a vector image is very helpful. Converting from a bitmap to a vector image is also important for some other reasons (for example, precise definition of physical properties of the objects, or easier computer calculations of the changes in relationships between objects in the image).

The image must be prepared for the next step in the cognitive process – synthesis (*functional suitability*), where all elements defined by microunits make one compact image with as many details as possible (which are not all necessarily visible in the original input image). Synthesis is a complex process of integration of data properties belonging to each object from the image and their position (geographical, from GIS and geometrical, between each other inside the image). A well-prepared image is highly defined and a precise image in all its details.

The main elements in the filtering process are microunits. The idea of microunits concerns an idea from cognitive science – *micronodes* – that belongs to brain simulation. The main process of filtering is the comparison, composition and integration between the image defined by the process of object recognition and many different microunits.

The concept of 'microunits'

We do not yet have exact and precise recognition of urban elements from the input image to be used directly in the model, and we adopted Ullman's 'object recognition' model instead. This model does not rely on the 'full recognition' that we need for the function of our Design Tool. In addition, we will use the concept of the 'extended idealized *visual* element', which we will call the *microunit.* A microunit is the smallest visual element (metaphorically – *urban* element), that can be used in terms of an individual microunit. It is always in terms of groups of microunits, and a group of microunits is really the 'unit' of the model.

The main features we will assume about microunits are listed below with all qualitative and actual values and the exact forms of the dependence are assumed to differ between microunits:

(1) microunits are located (stored) in the *typology of microunits* as a part of the DB,

(2) a microunit can be active or non-active,

(3) a group of microunits gets initial input directly from the 'data model' (see Figure 7.3) and sends *connection* output to many other microunits (*connection* is a term that means one of them gets input from the other),

(4) the activity of the microunit is determined by its inputs; an active microunit becomes inactive by finishing its duty or when it is without enough activation (input information),

(5) the strength of a connection between two microunits can be increased; this is dependent on both microunits being active,

(6) the process of activity of microunits above the input image is the filtering process,

(7) all active microunits create objects of the filtering image called the '*meta* image',

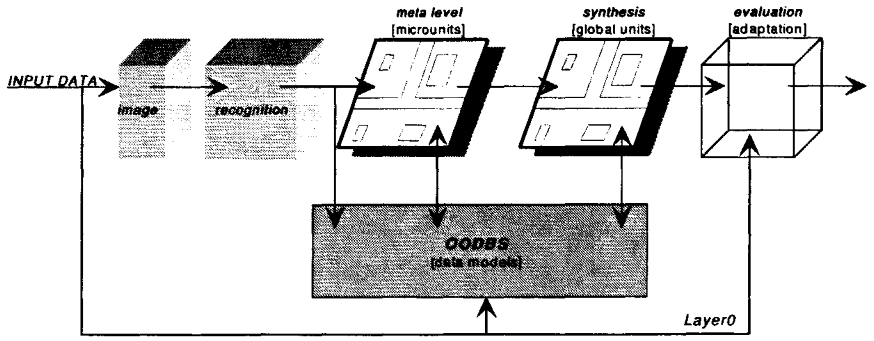
(8) the microunits send cohesive output of the filtering image - this has several consequences:

- the microunits which are activated, in general, cohere the lines of the 'recognized image' in new 'overlapping' lines of the filtering image,
- the *cohesive connections* are filtering, because of actually generated cohesion
- the cohesive connections take more microunits to make a filtering effect.

(9)the connections between microunits defuse when the cohesion of all lines in one image is finished.

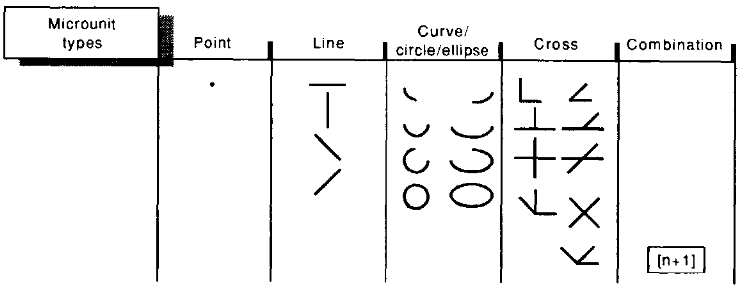
The process mechanism of microunits is probably more complex than we have explained here.

Figure 7.3
Relationship between *meta level*, *synthesis* and *OODBS*



As we see, microunits are the smallest visual elements, which consist of points, lines, curves, circles, ellipses, different crosses or combinations of all these elements. A combination of elements for each of the basic shapes is enormously large [the number of combinations might be: n+1], it is practically infinite and the computer will calculate them automatically depending on the shapes of the analyzed image. More generally, an illustration of the stored data typology of microunits in the OODBS is presented graphically in Table 7.1.

Table 7.1
Generic typology of microunits (illustration of some basic shapes)



The term 'active' refers to the active microunits in the model, and the 'level of activity' corresponds to the number of active microunits. It should be noted that the number of active microunits at any point is assumed to be large, and that an active process does not necessarily have more active microunits than non-active microunits.

It will be useful for the system to avoid activating too many objects (urban elements) in the image at the same time, so that it can focus on specific points. This restricts the activity of the system to this level. This level of activation corresponds to the smallest number of urban elements. Only microunits, which receive a large number of activating signals, will become active. The limited activity in the system means that when many microunits get activating input, only a small part of them will become active, and this will inhibit the rest.

This hypothesis explains that only a small number of urban elements can be active at any time, and the system has a strong tendency to stay focused on the currently active microunits.

The 'synthesis' module

Synthesis is the last module of the Computer Cognition Model (see Figure 7.1 and Figure 7.2). The main task of synthesis is functional suitability, whereby all objects of the image scene become stable for further manipulation and analysis. Synthesis is a complex process of image and data integration. The process of synthesis is subdivided into the processes of *matching* and *mapping*.

Matching is the identification of completed objects of the image scene ('global units') by a match method. We use three main methods for matching: the description method, feature identification and the analogy method. There are no functional borders between these methods in our model. The model will automatically select the proper method depending on the situation in the image scene and input data. Matching is important for defining the object or group of objects in the image scene, by comparison with a corresponding – or the same – object or objects from other images as well as from *Layer0* (compared with the 'input image' or by itself).

Mapping is the process of data definition of objects ('global units') from the image scene. We use the process of mapping for each single object from the image scene with corresponding data from the database system through a particular data tree. The data model is created to support physical elements of the urban environment (the 'hierarchy of urban elements') and psychological elements ('psychological preconditions'). The whole structure of the data model will be explained in detail in the next chapter.

All microunits belonging to one object of the image scene comprise this object. An object composed completely of microunits is called a 'global unit'. A global unit is composed of many microunits. One global unit might represent one object from the image scene. But one object might also consist of several global units. These global units should be independent from each other but mostly they have the same characteristics as the object. A global unit is a higher level of microunits. A global unit consists of physical elements (microunits) and data (values of the object or different data which correspond to the object). A couple or more global units might be recognizable in the generic 'typology of urban elements' (see Figure 6.17). It is also possible that global units do not belong to this typology (the objects of the image scene are not the objects of the urban environment), but might belong to some other stored data from the database system. The typology of urban elements is stored as a partition of the database system. The typology of urban elements is generic and consists of an infinite number of possible combinations of basic urban elements. By a process of matching all global units from one image scene, a '*working* image' is composed. A working image consists of the global units and corresponding stored data behind. All objects in the image scene become potentially active (activated by the user's choice), and all available data about each object from the image scene become potentially active too (by the process of mapping).

There are some technical requirements related to *synthesis* that have to be fulfilled:

- to form the generic 'typology of urban elements' (it might be also called 'typology of global units') as a partition in the database;
- to compose the 'working image' with all global units from the image scene (from the '*meta* image');
- to distinguish one object from other, surrounding objects (different types of objects derived from the 'typology of urban elements');
- to link the fully composed object of the image scene to the 'typology of urban elements';
- to link several objects of the image scene to the correlated urban scene from the 'typology of urban elements';
- to compare some objects of the image scene with one or more urban types from the 'typology of the urban elements';
- to link the object to the 'data model';
- to link the object with the object's values from the database system; and
- to link objects from the scene with the corresponding data from the database system.

There are also some further requirements:

- to link each object from a high density urban pattern to the data model (to take related data in this case);
- to link an invisible (partly visible) object to the data behind (different types of available data);
- to activate the proper *tree* from the data model related to the particular urban scene (from the 'typology of urban elements').

More about synthesis in relation to the data model will be explained in general in Chapter 9.

The concept of 'global units'

As we explained above, the model contains infinite numbers of global units, which always depends on the input image. The global units are not unified but the operations that they carry are assumed to be generically programmed for the specific task. The global units carry some specific operations. In particular, they carry processing at the level of urban elements (recognition in the 'typology of urban elements'), but they also carry an assumption of data by activating the proper data tree from the data model (which will be explained in the next two chapters).

The main features of the global units are:

(1) they are located in the image scene (the 'working image'),
(2) they can be all or selectively active, or non-active if there is no initial impulse,
(3) the activity of the global units is determined by their initial impulse; one active microunit becomes inactive by finishing its duty,
(4) they get initial impulse directly from the user's decision (clicking the mouse in the 'evaluation module') and send recognized output via the 'typology of urban elements' to the data model,
(5) they get directly corresponding properties from the database in different graphic forms,
(6) they may monitor the activity in the model or part of the model,
(7) they may affect the overall activity of the model or part thereof,
(8) they may receive input and send output to/from other global units in the model (some other input images related to corresponding data), and
(9) the input or output of global units may be connected to some of the other possible images in the model (from the stored database).

Some assumptions of the existence of global units:

(1) it may control the synthesis of activities between matching the objects and mapping the corresponding data (self-control activity),
(2) it may modulate the control of the level of activity in the input analysis model (controlled by the model itself),
(3) it may control output to the evaluation process of the model,
(4) it may monitor activity on the border between urban and cognition input,
(5) it may monitor the level of activity, and
(6) it may detect significant changes in the level of activity.

Global units, as we explained above, are the main elements of synthesis between objects of the image scene and assumptions of corresponding data from the database system. The main goal of the global units is to make correspondence between a static image ('working image') and all data available to the objects from this image, which are contained in the database system of the model. Another goal of the global units is to help in processing specific tasks of the user, such as cognitive mapping, way-finding or analysis of a path or district. Global units have an important role in these tasks because they can be analyzed by object geometry (physical characteristics of the urban site, such as distance or route) as well as psychological preconditions (such as the visual quality of a particular footpath related to cognitive mapping) by using all corresponding data. They will both give output results in different representational, mostly graphic ways. In Chapter 8 we will explain the relationship between global units and the corresponding 'typology of urban elements' and the database. There we will present the main characteristics of the structure of the data model, the organization of the database and its partitions ('typology of microunits', 'typology of urban elements', GIS and some other data types).

Tool requirements derived from the Computer Cognition Model

In this chapter we presented some new important organizational and technical characteristics related to the *Computer Cognition Model* of our Design Tool. These new things should be very important in the conceptual modeling of the tool. Here we will present the main set of requirements for the tool derived from the *Computer Cognition Model.*

The tool requirements might be as follows:

- applicable to two main types of analysis of the urban environment: according to urban elements of the environment (from the 'typology of urban elements') and according to psychological preconditions (psychological characteristics belonging to urban cognition, such as cognitive mapping, or visual quality of the urban site);

- applicable to fast recognition of the urban elements ('objects recognition') from the input image, fast filtering of the urban elements (by 'microunits') and fast synthesis of the objects from the image scene and corresponding data from the database system (by 'global units');
- applicable to all types of input and output image (different file formats) and their corresponding data (properties of the objects, physical geometry of the objects, psychological characteristics, etc);
- applicable to the precise analysis of different sorts of input images (high density urban pattern, cognitive map sketches, different scales, etc.);
- applicable to the precise dividing of each of the objects into the image scene (with precise assumptions of corresponding data from the database);
- applicable to making an analysis between urban elements in the image scene (such as the route distance between two objects, or a cognitive mapping analysis of the 'best walking lap', etc.);
- applicable to include in urban analysis and data assumptions the objects which are 'partly visible' in the image scene;
- applicable to the analysis and data assumptions for an object presented not only orthogonally, but also 'three-dimensionally' (2½ D) or in 'vertical projection'; and
- applicable to different (mostly visual) data representations as output of the analysis of the image scene (both for physical and psychological tasks).

These requirements are the second set of tool requirements, derived from computer cognition. We will implement these tool requirements in different phases of the process of tool designing (most of them in the integration part of the conceptual design in Chapter 9.). In the next chapter we will explain the basic data structure of the Design Tool with new sets of tool requirements. All together, three sets of tool requirements will be the proposed platform for developing the conceptual model of our Design Tool.

Conclusion

In this chapter we explained the main concept of the Computer Cognition Model related to our Design Tool.

First we explained the ideas concerning general modeling in cognitive science, where computer cognition plays a significant role in the simulation of some basic cognitive and behavioral processes. We then presented the basic idea of the model, its features and some basic assumptions.

This chapter also presented the *Computer Cognition Model* by explaining its main functions, characteristics and elements. The Computer Cognition Model consists of three modules: *recognition* (adopted techniques of 'object recognition'), the *meta level* or *filtering* (the module in which 'microunits' create the '*meta* image') and *synthesis* (the module where each of the objects from the scene, called 'global units', assumes corresponding data from the database). The adopted *recognition* module was explained in detail in Chapter 3. The second two modules are indicated by each of their main functional characteristics. Here we only showed the role of the 'microunits' and 'global units' and their main relationships; more details will be explored in Chapter 9.

At the end of this chapter, we set out the set of tool requirements derived from the *Computer Cognition Model.* This set of requirements, together with the set already given in the previous chapter, will be the main platform for the conceptual design of our Design Tool. This chapter shows that the Computer Cognition Model plays an important role in the process of developing the *conceptual model* of the Design Tool.

The next chapter will explain more of the object-oriented database model for our tool. We will focus on presenting the integration data model for both urban elements and psychological elements. This data model will be a crucial part of the whole *conceptual* and *functional model* of our Design Tool.

CHAPTER 8

THE OBJECT-ORIENTED DATABASE MODEL OF THE DESIGN TOOL

***Task of the chapter**: developing a database model with Perspective-DB technology, which provides the means to present and process the complex structure of any urban environment and its cognition within a multi-media presentation of application objects in their semantical context and exploit them within this context, for the proposed Design Tool.*

Introduction

In the previous two chapters we developed important parts of our Design Tool. First we developed, on the basis of a deep investigation of Lynch's theory, the hierarchical structure of elements that create the visual quality of the urban environment related to the cognitive mapping approach. We then developed the Computer Cognition Model that is based on an investigation of cognitive science. In this chapter we will develop an object-oriented database model, which will be based on the previously introduced Perspective-DB technology. These three developing steps are important for building the conceptual model and developing the proposed Design Tool.

The core of the proposed Design Tool is an object-oriented database system [ODBS]. As we mentioned in Chapter 5, the object-oriented database management system [ODBMS] provides generic features to the Design Tools supporting the designer, e.g. for computation, reconstruction, representation, simulation and pattern recognition. The Design Tools share the definition of the database by means of the data schema. It is a kind of common language, which the Design Tools use to communicate with their own modules in order to serve the designer. Each tool deals with a specific part of the information processing during the design process.

This chapter will focus on developing the basic data models related to the object-oriented database approach', the urban environment approach' and the cognitive approach, as already explained in previous chapters (Chapter 5, 6 and 7).

In this research, data models are used to build schemas that are representations of urban environments and their cognition. Data models are collections of concepts that are used to describe the structures, operations and conditions of databases. The structure of the database describes the data types, relationships and constraints that should apply to the data. The operations on the database and the conditions to realize these operations are called behavior.

This chapter will focus on developing a *conceptual* data model, to provide the structure and behavior of urban environments. A conceptual data model will present the high-level description of the hierarchy and relationships of the urban environment and cognitive mapping approaches in general. Here we will develop the structures of the object types, attributes and relations. Then we will develop the basic operations of schemas presented and some dependencies to guarantee the integrity of the object base.

Advantages of applying Perspective-DB technology to the Design Tool

Perspective-DB technology[1] was developed by Bitbybit Information Systems as an object-oriented database management system [OODBS]. Perspective-DB is used to design, create and maintain a database by the specification of the user's environment as object-types and with the visual

[1] Theoretical aspects of Perspective-DB technology are explained in Chapter 5.

presentation of information. Given the characteristics of the proposed Design Tool, we have chosen Perspective-DB technology mainly for the following reasons:

(1) There are no restrictions or limitations with respect to the types of structure or behavior that can be represented: the structure reflects the natural structure of objects specifying information used by several tools and understood by the human designer as well. The behavior reflects the operations that can be executed on the objects.

(2) The structure and behavior of objects can be updated or extended easily.

(3) Different layers of abstraction can be supported; that means each object can be represented in several grades of detail (like zooming in on or zooming out from a certain object) and with different views of relationships between objects at various abstraction levels.

(4) Several representations (multi-media) of the same object can be supported naturally.

(5) An object can have several well-defined interfaces offering the requisite information to a tool, hiding other information, and hiding implementation details.

Building the modular software is simplified through offering support for the reuse of objects and the system's functionality as well as the possibility of tailoring the generic system to the real needs of a particular organization.

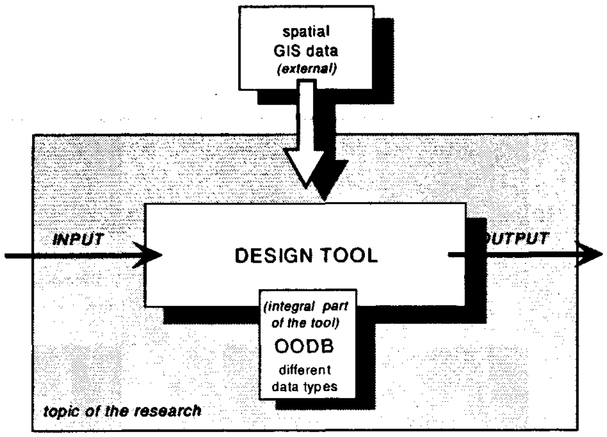
As we have already explained, the object-oriented database model [OODB] is an integral part of the Design Tool. The importance of the OODB is huge because the whole Design Tool functionality is directly (or indirectly, e.g. in the *recognition process*) related to it. The position of the OODB is 'central' in the system and it is not visible for the user. The 'central' position in the Design Tool hierarchy means that the OODB is aided from the 'background' in all modules of the Design Tool by its services. Services are not only manipulations of the information from stored databases but also interactions between all modules through their hierarchy. The global positional scheme of the OODB is shown in Figure 8.1. The position of the OODB in the functional model of the Design Tool will be presented in Chapter 9. The OODB model could be easily connected (e.g. via the Internet) to any other available spatial GIS data or statistical data (for example the Central Office of Statistics). Through this linkage, the system would be able to process an almost infinite amount of spatial data for much greater numbers of users and their tasks. Preconditions for this are the interfaces that can map the data structure and behavior.

The general structure of the basic OODB model for the Design Tool consists of four parts, presented in Figure 8.1:

(1) *general manager*, manages all exchanges between functional modules of the Design Tool and their related data modules, the stored database and any external data module,

(2) *data models*, are the 'mirrored' hierarchical data models which functionally 'copy' the spatial hierarchy of urban elements and their cognition (data models will be developed in this chapter),

Figure 8.1
Positional schema of the OODB system



(3) *data models*, are the 'mirrored' hierarchical data models which functionally 'copy' the spatial hierarchy of urban elements and their cognition (data models will be developed in this chapter),

(4) *stored data*, consisting of several modules:

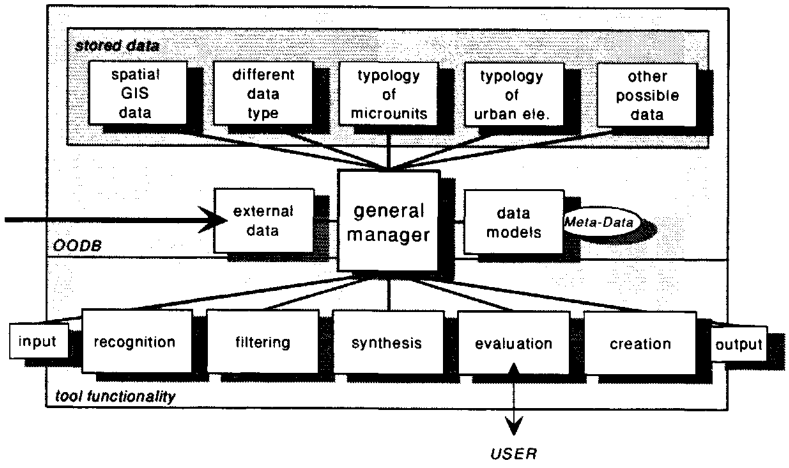
- *spatial GIS data*, is spatial alpha numerical data which is the crucial data for supporting the main Design Tool functionality, such as map (objects) composing and analysis;

- *different data types*, such as statistical data, data about traffic etc., which will be additional support data for more specific tasks;
- *typology of microunits*, is the generic typology of the smallest visual elements which form an image scene and support the 'filtering process';
- *typology of urban elements*, is the typology of generic precedents of urban elements which supports the 'composition process'; and
- *other possible related data*, is a reserved module for adding new data for some special tasks, and

(5) *external data* is a module which could provide on-line linkage via the Internet to several available remote databases.

The general manager, presented in Figure 8.2, is physically and functionally located between the Design Tool and the OODB and belongs to both of them.

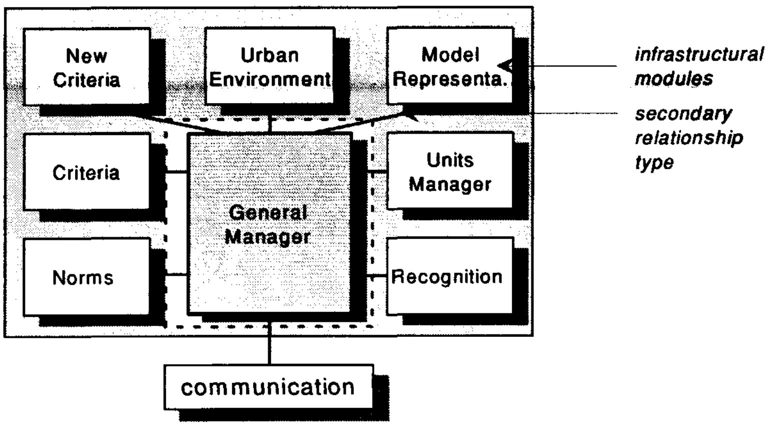
Figure 8.2
The general structure of the basic OODB model



The modular structure in the database model of the Design Tool

An existential hierarchy is the main structure that forms a module. If a module has to be updated or removed, or if an additional one has to be added, all the interconnections (the secondary relationships between the modules) are located easily and can be modified as required. The general principle of the OODB system modules related to each other is presented for our Design Tool in Figure 8.3.

Figure 8.3
The modular structure of the OODB system for the Design Tool



The modular structure of the OODB system of our Design Tool consists of three functional groups:

(1) the *general manager*, which exchanges functions between modules;

(2) *infrastructural modules*, which represent the modules of the database models such as the hierarchically structured urban environment, model representation, recognition, criteria, new criteria, norms and units (local) manager, and

(3) *communication*, which helps the general manager in exchanging different protocols and formats.

All three of these functional groups interrelate to each other in a logical, functional and hierarchical order, depending on the user's task.

Guaranteed by the aforementioned concepts, and in addition to the consistency of information, the maintenance of the system is supported by keeping relations between the modules separate and guaranteeing module autonomy. The OODB system can be easily adapted to new situations and needs. Reconfiguration mainly consists of removing, inserting and re-interpreting secondary relationships between the general manager and other modules.

Structuring the database model of the Design Tool

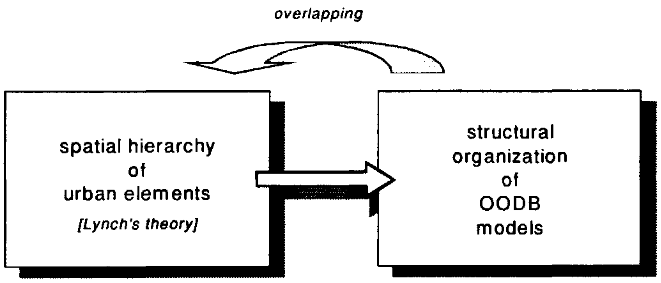
In Chapter 6 we developed the structure of the spatial hierarchy of the urban environment elements and their cognition, based on Lynch's theory of '*urban form*' and the cognitive mapping approach. From the key points of the results of these developments it can be concluded that the urban environment consists of two types of elements, one belonging to the physical structure or *urban elements*, and the second type being the *psychological elements*, from which the cognitive mapping of the urban environment can be built.

The *urban elements* consist of five elements that can compose the visual quality of the urban environment. These elements are *district, path, edge, node* and *landmark*. Each of these elements is subdivided into its sub-elements, and these sub-elements into their sub-elements, down to the smallest physical units of the urban environment.

From the other side, *psychological elements* are subdivided into four elements, *legibility, image, identity* and *imageability*, which, together with the urban elements, can build the cognitive mapping of the urban environment. These elements in the OODB model will be called 'psychological precategorization'.

The main idea of structuring the OODB models (schemas) is to follow the urban spatial hierarchy. The way of structuring is that each urban element is an object in the database. Each object in the database is defined as a physical object in the urban environment. Psychological elements are also objects in the database. The database structures completely overlap the structure of the urban environment and its hierarchy and make the functionality of the Design Tool more efficient and precise. A simple schematic overview of the relationship between the spatial hierarchy and OODB models is given in Figure 8.4.

Figure 8.4
Relationship between spatial hierarchy and OODB models


overlapping
spatial hierarchy of urban elements
[Lynch's theory]
structural organization of OODB models

Therefore, the main logic of structuring is that each object in an urban scene must correspond to similar objects in the database model, and each object from the database will have its own properties, relations and operations. There are several, mostly functional reasons for this:

(1) *query* - the Design Tool can make a query through all the related objects in the database model concerning the query task, which can be defined at the beginning of the query process as an 'input data' [*example*: if the designer needs to find one particular object [bridge] in the image scene, then the Design Tool will help him by query processing through the stored data of this image scene and will then indicate this bridge object to the designer];

(2) *mapping* - the Design Tool can create a new map (mostly 'cognitive maps') from a different selected object, also taken from stored data, following the spatial structure and hierarchy which are related to precedents stored in the 'typology of urban elements' [*example*: the designer fills in input parameters for creating some particular map according to a public transport task and then the Design Tool creates a new map related to the task above the selected map of the indicated area];

(3) *way-finding* - is similar to a query, the system can make different routes and route analyses, graphically presented on the selected

map, according to all related objects taken from the stored data indicated at the beginning of the way-finding process as 'input data' [*example*: the designer selects two or more objects from the image scene and fills in the input info according to the task, and then the Design Tool creates different routes between objects and presents all requested information, such as distances or a public transport timetable (if requested)]

(4) *cognitive mapping* - the system can create a cognitive map and manipulate it in various ways, created realistically above the selected map (topological, city or other map type stored in the database system), supported by all related objects from the stored data according to the cognitive mapping task; the system can analyze this map depending on the designer's needs, and the whole process is fully supported with GIS data [*example*: the designer fills in input parameters for a cognitive mapping task and then the Design Tool creates a cognitive map above the selected map of the indicated area];

(5) *shaping* and *reshaping* - the system can provide the designer with the option to change the objects of an image scene, by shaping or reshaping them, or adding or deleting objects - some basic 'point and click capabilities' are available [*example*: the designer wants to change the form of an object from the image scene and he/she indicate this, then an additional tool of the Design Tool becomes available, and the designer freely changes all shapes of the object or even deletes it];

(6) *comparison* - the system can compare different objects' forms, or some other characteristics, in an image scene by some simple functionality using related objects from stored data [*example*: if the designer wants to compare the height of two objects in the image scene, then he/she selects both objects, indicate the tasks and comparison results will appear showing the height measurements];

(7) *adaptation* - the system can change an image scene by implementing some objects from another image scene or from stored data objects [*example*: if the designer wants to check how his/her new design of a building will 'fit' in an image scene, he/she indicates this task and simply inserts the new object into the image, and then the Design Tool will perform analyses according to the indicated task and will present all results]; and

(8) *creation* - the system can create a new image scene using stored objects, according to the target urban type [*example*: the designer would like to make a new urban design using an existing urban scene from stored data, so he/she indicates the scene and the task and the Design Tool creates a new image scene using the selected existing image scene and according to the given task].

Structuring the Design Tool in a manner close to the real urban environment (Universe of Discourse) will help the designer to use the proposed Design Tool in many intuitive ways, with many more functional possibilities. More illustrative examples of the Design Tool functionality will be presented during the simulation of the case objects in Chapter 10, which relates to the testing and evaluation of the Design Tool.

The object-oriented database model of the Design Tool - the user's view

First we will develop the structural part of the object model, which can follow the real urban environment's (Universe of Discourse) hierarchical structure. Following that, the behavior with generic operations will be developed.

Occupying the main and hierarchically highest position is the 'General Manager' (see Figure 8.6). The 'General Manager' object type is a representation of the object characteristics through a definition of its members. In the case of our Design Tool, the members are:

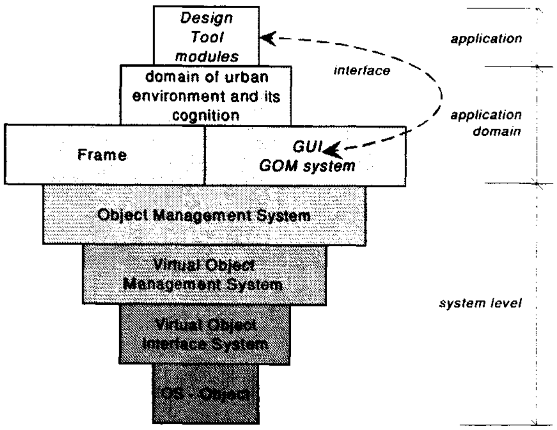
- Urban Environment,
- Model Representation,
- Recognition,
- Criteria - Norms, and
- Unit's Manager

All of the object types with their hierarchy are presented in the object types' schemas. The behavior schemas with the objects' operations and dependence will be presented and explained separately from the object types. The following part of the chapter presents the main schemas of the object-oriented database model of the proposed Design Tool related to the urban environment.

Overview

Before we start to develop our database structure in full detail, which will follow the hierarchy of the urban environment, we need to give a general overview of the system architecture. Our Design Tool is built up in a modular form. The system architecture consists of several subsystems. The specialized system is constructed by incorporating generic subsystems and adding the necessary services. The system is designed by stacking subsystems on top of each other, from top to bottom in 'pyramidal order' (see Figure 8.5). The layers are distinguished by the role the subsystems play in the system.

Figure 8.5
Operational system architecture of the Design Tool



The operational system architecture consists of three layers:
(1) *application*,
(2) *application domain* and

(3) *system level.*

The highest layer is '*Design Tool* modules', which is built as the most specialized information system and belongs to the *application* layer. The *Design Tool* modules will be explained in Chapter 9.

The layer beneath provides generic objects to it, which are typical for this *application domain* of the urban environment and its cognition. The graphic presentation functionality and support for dynamic behavior is a combination of standard GUI and the Graphic Object Management System. This part of the 'pyramid' will interact together with the higher layer of the 'pyramid'; we called both of them the 'application' and 'application domain'.

All other levels of the 'pyramid' are *system levels*, and they are 'invisible' to the end user. The 'frame' layer deals with application administration. It contains a distribution and messaging module to support communication between distributed objects over a network. In one of the system layers, representing the Object Management System (OMS-Object), is a means to function as object request broker. The brokerage functionality is used to realize interfaces with other systems. Other subsystem layers, 'virtual objects management system', 'virtual object interface system' and 'OS - object' are the standard operational part of the system, which we do not need to explain here. They also belong to *our* invisible 'system level'.

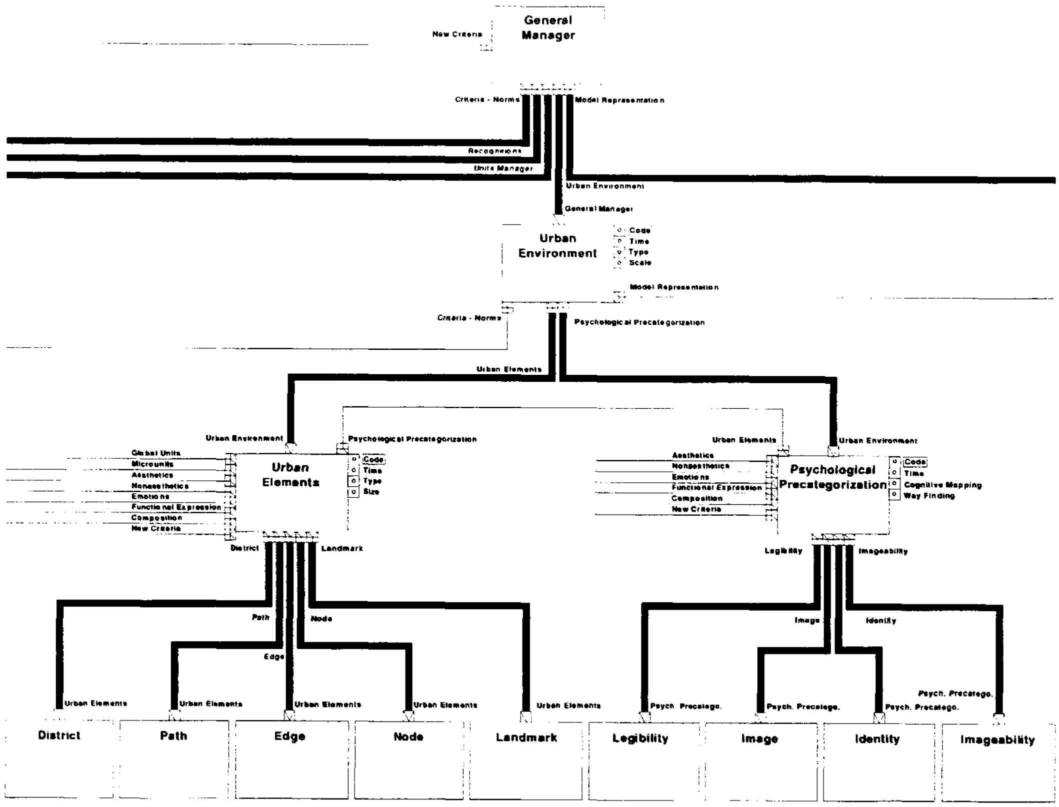
The system overview is only presented here for the task of indicating the position of '*Design Tool* modules' and their interfaces in the operational system architecture.

Structure [object types + attributes + relations]

The developer view object schema describes the generalization/specialization hierarchy of the object types needed to derive the information used for the design, taking cognitive mapping into consideration. The object type 'Urban Environment' inherits its members from the 'virtual object' offered by the Perspective-DB (not represented here). 'Urban Environment' consists of other generic members: 'Urban Elements' and 'Psychological Precategorization' of urban elements (see Figure 8.6). 'Urban Environment' has secondary relationships with another two generic members: 'Model Representation' and 'Criteria - Norms'. The generic attributes of 'Urban Environment' are:

- 'Code' - the uniform system of coding for each object, or 'object of the object', will be explained in the next chapter; coding is very

Figure 8.6
Basic object types: General manager, Urban Environment, Urban Elements and Psychological Precategorization



important for the efficiency of defining an object and queries through the system;

- 'Time' - is an important attribute which precisely indicates the date related to the object;
- 'Scale' - is the attribute that defines the scale of the urban environment, important for spatial analysis; and
- 'Type' - is the attribute that indicates the object type related to the urban element's typology.

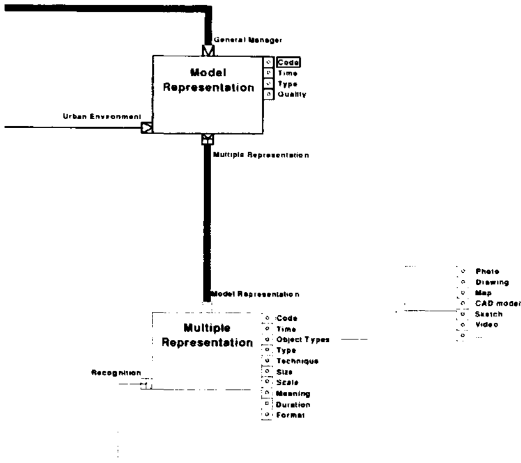
Figure 8.6 also presents the inheritance schema of the object types 'Urban Elements' and 'Psychological Precategorization'. The object type 'Urban Elements' consists of the following generic members, adopted from Lynch's theory: 'District', 'Path', 'Edge', 'Node' and 'Landmark'. 'Urban Elements' has cardinality relationships with other generic members: 'Global Units' and 'Microunits', and with all objects of the object type 'Criteria - Norms': 'Aesthetics', 'Nonaesthetics', 'Emotions', 'Functional Expression', 'Composition' and 'New Criteria'.

There is also a relationship between "Urban Elements' and 'Psychological Precategorization', where 'Urban Elements' consists of 'Psychological Precategorization', but not the other way around.
The set of generic attributes of 'Urban Elements' are the same as the attributes of "Urban Environment': 'Code', 'Time', 'Type', and the only new attribute is 'Size', which defines the physical size of a particular urban element (by x, y, or z dimensions).

The object type 'Psychological Precategorization' consists of generic members also defined by Lynch: 'Legibility', 'Image', 'Identity' and 'Imageability'. Here there are also relationships with other generic members from the object type 'Criteria - Norms': 'Aesthetics', 'Nonaesthetics', 'Emotions', 'Functional Expression', 'Composition' and 'New Criteria'.
The generic attributes of 'Psychological Precategorization' are partly the same as the attributes of 'Urban Environment' and 'Urban Elements': 'Code' and 'Time'. The new attributes are 'Cognitive Mapping' and 'Way-Finding', which indicate the main attributes of the Design Tool tasks.

In Figure 8.7, the user's view of the object schema is shown. Here we see the ownership relations between all specialized object types explicitly determining the media types that represent the urban elements. An element can have several suitable media representations. 'General Manager' owns 'Model Representation', that in itself owns the 'Multiple Representation' object type. 'Model Representation' has a cardinality relationship with 'Urban Environment'. The new generic attribute of 'Model Representation' is 'Quality', which describes the quality of the representation, such as very good, good, bad and worst.

Figure 8.7
Basic object types: Model Representation and Multiple Representation

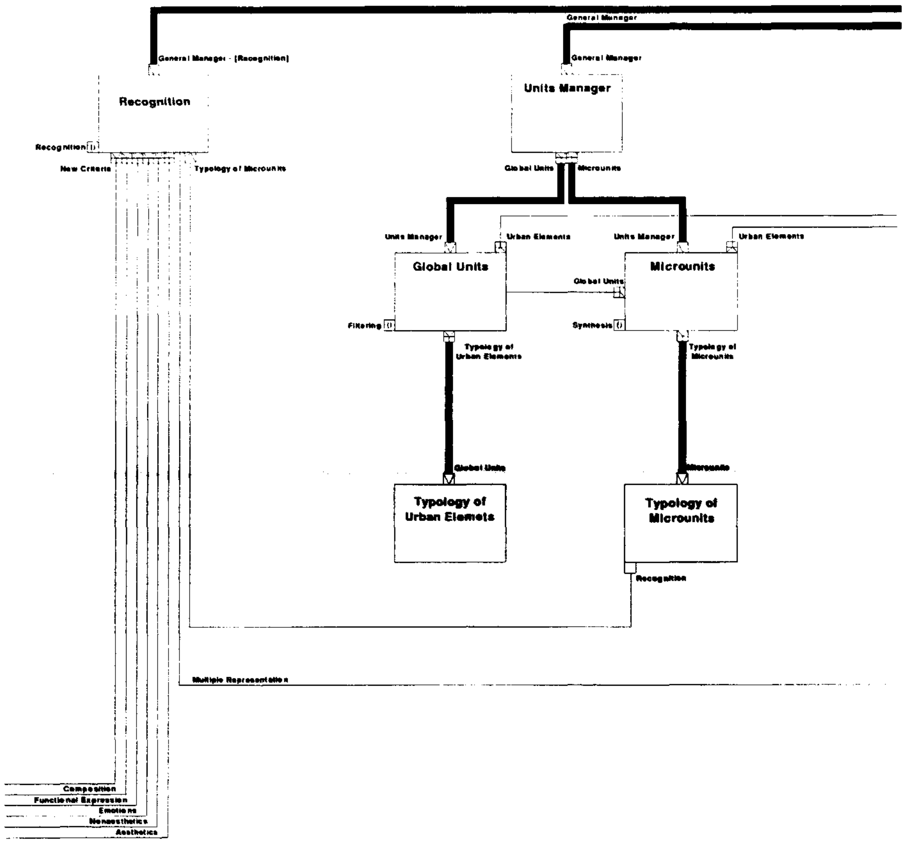


'Model Representation' owns the 'Multiple Representation' object type. 'Multiple Representation' belongs by relationship to 'Recognition', but 'Recognition' does not relate to 'Multiple Representation'.

The new generic attributes of 'Multiple Representation' are: 'Object Types', 'Techniques', 'Size', 'Scale', 'Meaning', 'Duration' and 'Format'. All attributes present characteristics of the different media of visual representation. 'Object type' is the 'object of the object' and represents several different media objects, such as: 'Photo', 'Drawing', 'Map', 'CAD model', Sketch', 'Video', etc. 'Object of the object' is presented in Figure 8.7 as a separate object which appears from the object type 'Multiple Representation'.

As we see, 'General Manager' also owns 'Recognition' and 'Units Manager', which then owns the 'Global Units' and 'Microunits' object types presented in Figure 8.8.

Figure 8.8
Basic object types: Recognition, Units manager, Global Units, Microunits, Typology of Urban Elements and Typology of Microunits



The object type 'Recognition' has one relationship with 'Typology of Microunits', 'Multiple Representation', and with all objects of 'Criteria - Norms' ('Aesthetics', 'Nonaesthetics', 'Emotions', 'Functional Expression', 'Composition' and 'New Criteria'). Instead of attributes, 'Recognition' has one specific operation: 'Recognition', which is the adopted plug-in application of 'high-level vision' developed by Ullman (as explained in Chapter 4).

The function of 'Units Manager' is to relate all functions of the object types 'Global Units' and 'Microunits'. 'Global Units' consists of 'Typology of Urban Elements'; and 'Microunits' consists of 'Topology of Microunits'. 'Global Units' has its own relationship with Microunits, and a cardinality relationship with 'Urban Elements'. 'Global Units' has one specific operation: 'Filtering', which is a process belonging to the Design Tool application and will be explored in Chapter 9.

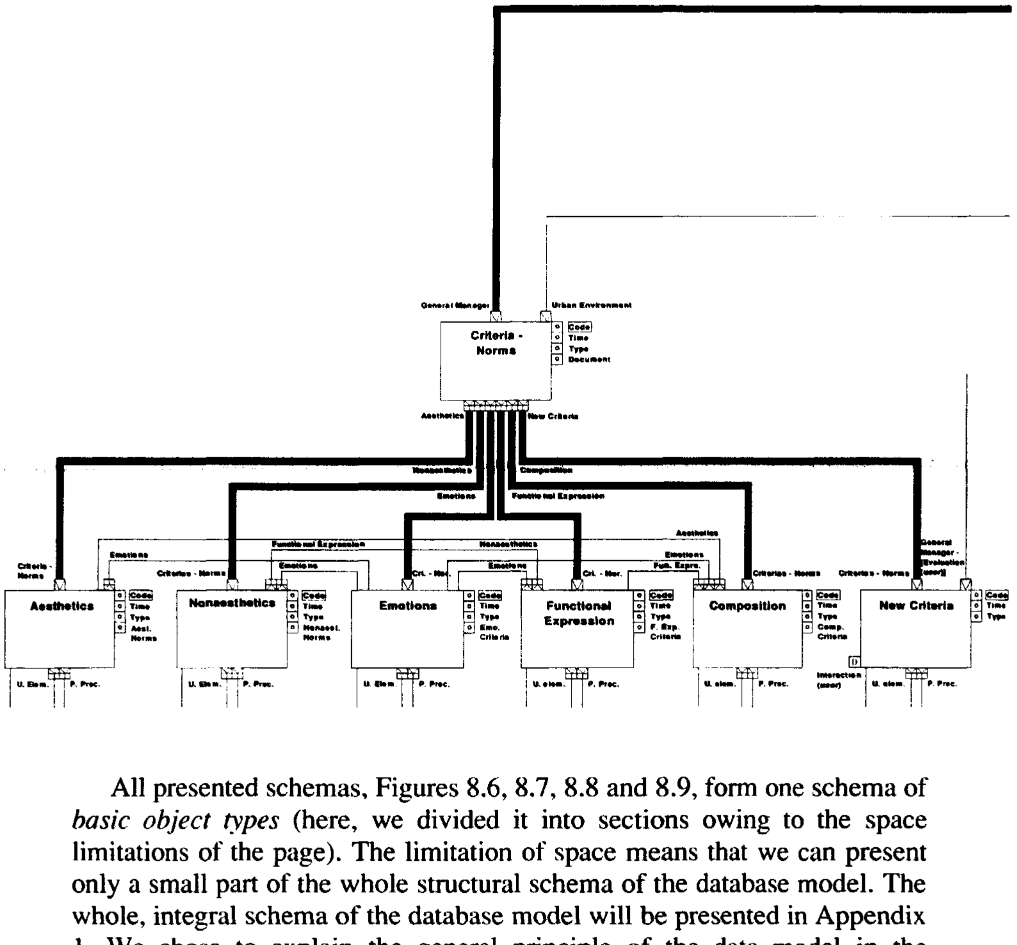
'Microunits' also has a relationship with 'Urban Elements' and one specific operation: 'Synthesis', which is one of the functions of the Design Tool application and will also be explored in Chapter 9.

Figure 8.9 presents the basic object types 'Criteria - Norms', which own different object types such as: 'Aesthetics', 'Nonaesthetics', 'Emotions', 'Functional Expression', 'Composition' and 'New Criteria'.

'Criteria - Norms' have a relationship that belongs to 'Urban Environment' and one generic attribute called 'Document', which refers to different normative laws.

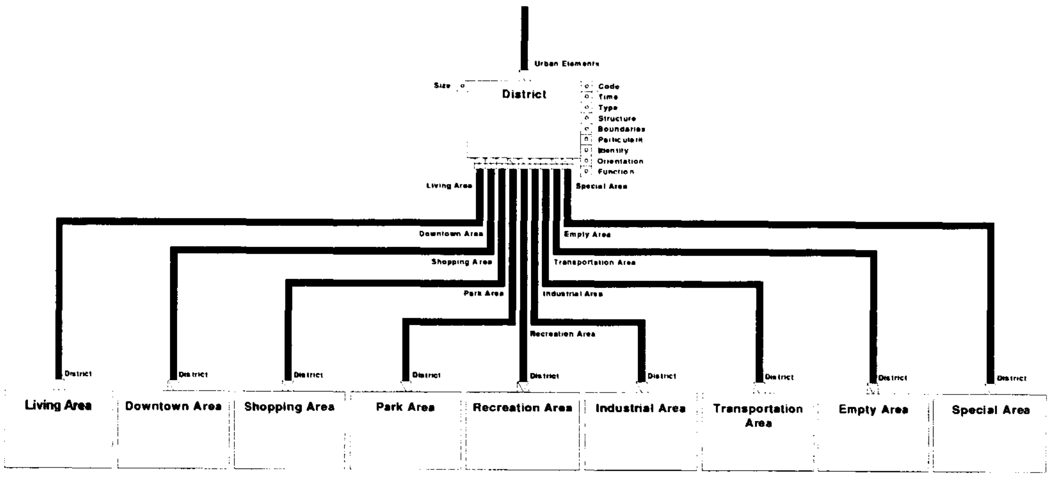
Object types that belong to 'Criteria - Norms', such as 'Aesthetics', 'Nonaesthetics', 'Emotions', 'Functional Expression' and 'Composition', have relationships that belong to almost all of them. They also have relationships that belong to 'Recognition' and they have relationships to 'Urban Elements' and 'Psychological Precategorization'. Only 'New Criteria' has a relationship directly with 'General Manager', because it is linked to the *evaluation module* of the Design Tool and directly interacts with the user (the user could create a set of 'New Criteria'). 'New Criteria' also has relationships with 'Urban Elements' and 'Psychological Precategorization', and a relationship that belongs to 'recognition'. All of the object types have the standard set of generic attributes ('Code', 'Time' and 'Type') and one more, such as 'Aesthetics Norms', which belongs to the 'Aesthetics' object type. The attributes of other object types are defined using the same principle. 'New Criteria' has a specific operation attribute called 'Interaction', which directly interacts with the user's wishes via 'General Manager'.

Figure 8.9
Basic object types: Criteria - Norms, Aesthetics, Nonaesthetics, Emotions, Functional Expression, Composition and New Criteria



All presented schemas, Figures 8.6, 8.7, 8.8 and 8.9, form one schema of *basic object types* (here, we divided it into sections owing to the space limitations of the page). The limitation of space means that we can present only a small part of the whole structural schema of the database model. The whole, integral schema of the database model will be presented in Appendix 1. We chose to explain the general principle of the data model in the direction of 'specialization' of the 'District' object type, presented in Figure 8.10 (see also Figure 8.6 for reference).

Figure 8.10
Object type 'District'



The object type 'District' belongs to the object type 'Urban Elements' and consists of several object types: 'Living Area', 'Downtown Area', 'Shopping Area', 'Park Area', 'Recreation Area', 'Industrial Area', 'Transportstion Area', 'Empty Area' and 'Special Area'. We will explain all of these object types which belong to 'District'. The extended atributes of 'District' are the folowing: 'Structure', 'Boundaries', 'Particularity', 'Identity', 'Orientation' and 'Function'.

The first part of the specialization of 'District' is presented in Figure 8.11 below. The first level of the 'District' specialization are the object types 'Living Area', 'Downtown Area' and 'Shopping Area'. The specialization of 'Living Area' is 'Neighborhood', and the specialization of 'Neighborhood' is 'Building'. The specialization of 'Downtown area' is 'Suburban Center' and its specialization is 'Neighborhood Center'. The specialization of 'Shopping Area' is 'Shopping Mall' and the specialization of it is 'Shopping Street'.

Figure 8.11
Object types 'Living Area', 'Downtown Area' and 'Shopping Area' and their specializations, attributes and relations

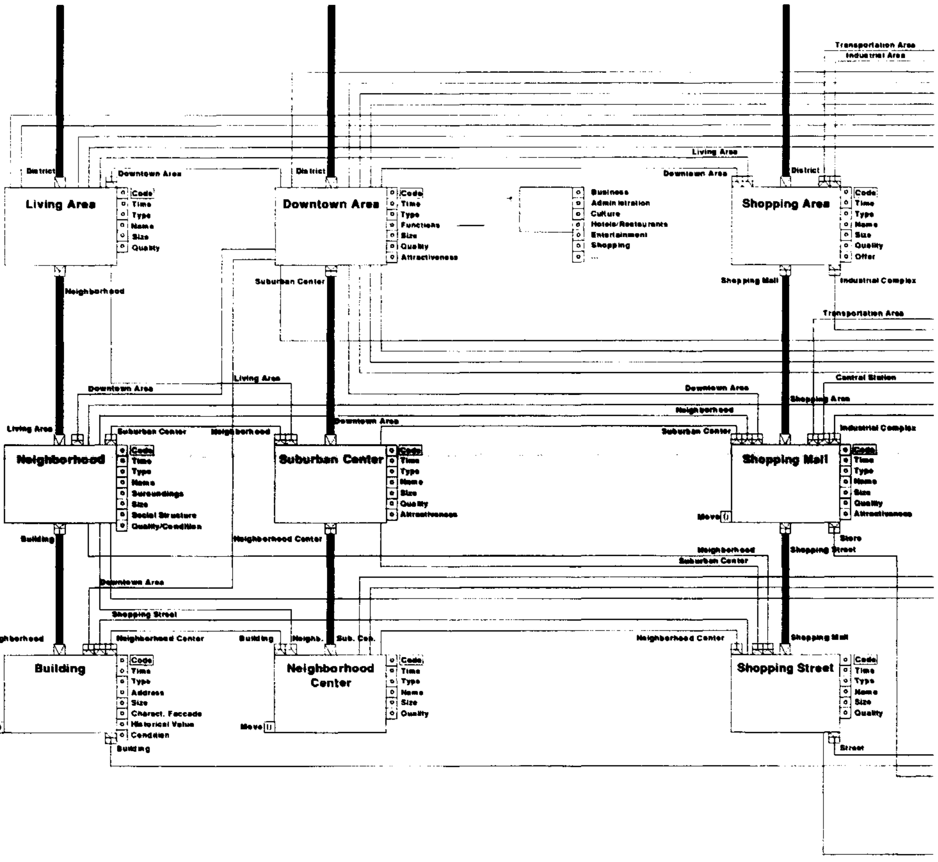


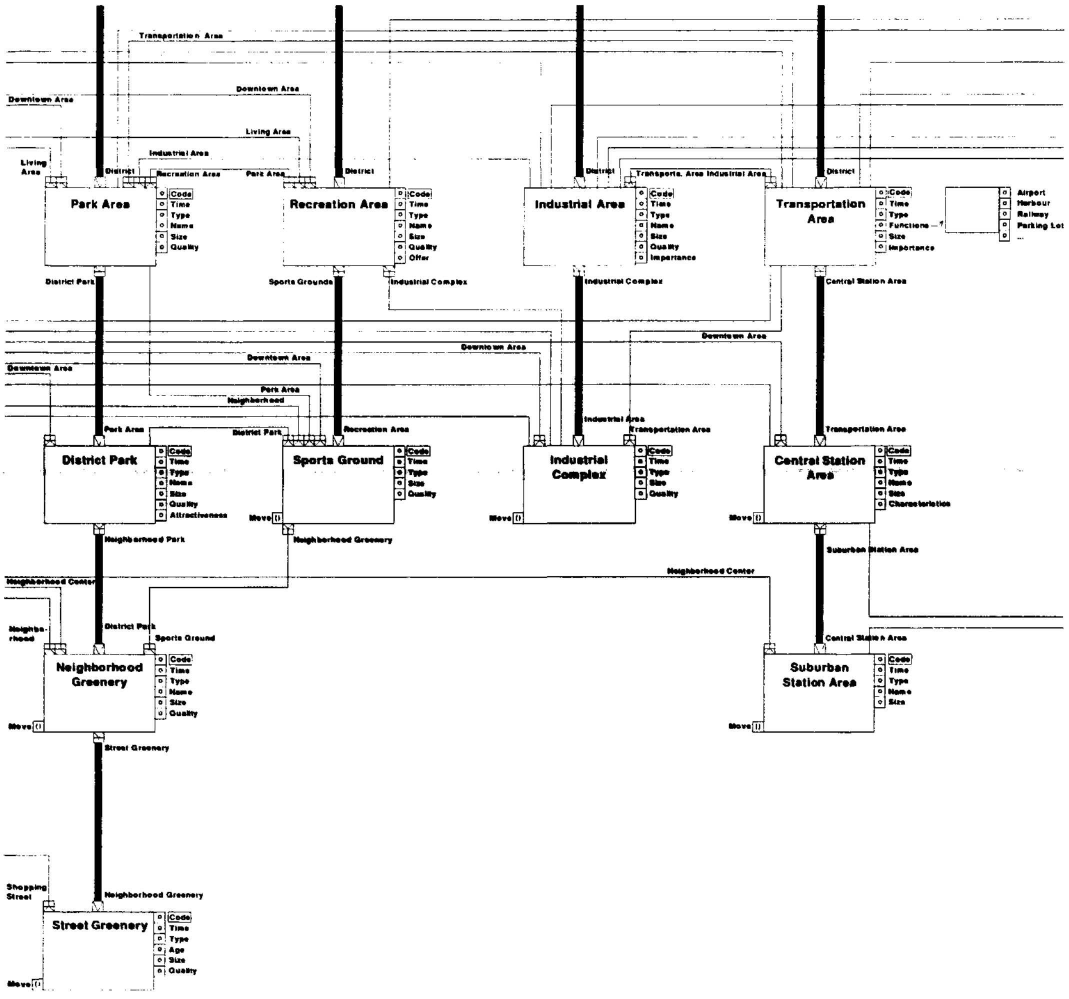
Figure 8.11 shows all the possible relations between the object types in the presented schemas and all others that will be presented later or in Appendix 1. We do not need to explain here each of the relationships because this would require more time and space; instead, all of them are clearly visible in the schema. We also do not need to explain all of the generic attributes that belong to the presented object types. The object type

'Downtown Area' has the generic attribute 'Functions', which has its own generic 'object of the object' types: 'Business', 'Administration', 'Culture', 'Hotels/Restaurants', 'Entertainment', 'Shopping' etc. with their generic attributes. Here we will also mention some specific operations. The object types 'Building', 'Neighborhood Center' and 'Shopping Mall' have the specific operation 'Move', which means that all objects from the images belonging to these object types can be moved to another position in the analysed image.

The second part of the specialization of the object type 'District' is presented in Figure 8.12. The first level of specialization are the object types 'Park Area', 'Recreation Area', 'Industrial Area' and 'Transportation Area'. The specialization of the object type 'Park Area' is 'District Park', and the specialization of it is 'Neighborhood Greenery', and the specialization of it is 'Street Greenery'. The specialization of the object type 'Recreation Area' is the object type 'Sports Ground'. The specialization of the object type 'Industrial Area' is the object type 'Industrial Complex'. The specialization of the object type 'Transportation Area' is 'Central Station', and the specialization of it is 'Suburban Station'. Also presented in this schema are all types of relations between all coresponding object types from Figure 12 and all other related object types. The generic attributes of all object types are also presented in the schema; only the object type 'Transporatation Area' has the composite attribute 'Functions' with its own 'objects of the object': 'Airport', 'Harbor', Railway', Parking Lot' etc. with their own generic attributes. The object types 'Neighborhood Greenery', 'Street Greenery', 'Sport Ground', 'Industrial complex', 'Central Station' and 'Suburban Station' have the specific operation 'Move', with the same functionality as explained earlier.

The third and final part of the specialization of the object type 'District' is presented in Figure 8.13. The first and last level of the specialization are 'Empty Area' and 'Specific Area'. 'Empty Area' has the relations correlating to 'Downtown Area', 'Living Area' and 'Industrial Area'. The 'Special Area' is mostly under military occupation, it is beyond our interest and we shall not analyze it! Both of the specialization objects have the same generic attributes; 'Empty Area' has a new attribute: 'Further Function'. 'Empty Area' also has the specific operation 'Move'.

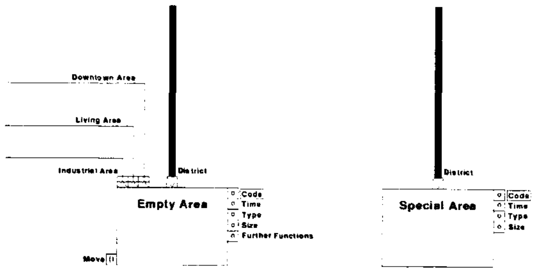
Figure 8.12
Object types 'Park Area', 'Recreation Area', 'Industrial area' and 'Transportation Area' and their specializations, attributes and relations



The figures 8.10, 8.11, 8.12 and 8.13 presented and explained here comprise one larger schema of the specialization branch of the object type

'District'. The same type of schemas present the specializations of the other object types and other branches, which belong to 'Urban Elements' and 'Psychological Precategorization' (see Figure 8.6 for reference). The other maps of the whole database structure will be presented in Appendix 1.

Figure 8.13
Object types 'Empty Area' and 'Special Area' and their attributes and relations



Behavior

In the previous section of this chapter we developed the object-database structure of the proposed Design Tool. We also presented the main object types, attributes and relations. In this section of the chapter we will develop the behavior of the objects.

The *behavior* of an object type is based on the *generic operations* that will be externally applied to each object type. The notation meaning of generic operations was explained in Chapter 5 (see also Figure 5.7). The internal structure of the object is hidden and the object is accessible only through a number of predefined operations. Some operations may be used to create (insert) new objects, and we call them 'object constructors'. Some operations may be used to destroy (delete) objects, and we call them 'object destructor'. Other operations may update the object states. Some operations may be used to retrieve parts of the object state or to apply some calculations, and they are called 'object modifiers'. Some other operations may perform a combination of retrieval, calculation and update. The implementation of an operation provides a power in defining the operations.

Each object type will have a number of generic operations. Operation names are normally unique for each object type, but they can be overloaded by having the same operation name for distinct object types. The operation signature can also specify the names of exceptions that can occur during operation execution. The operation signature sends a 'message' to the object to execute the corresponding operational implementation.

To present basic *operations* in our database model, we can use the same branch of the database structure that we used to explain the *structure* of the database [*object types, attributes* and *relations*]. We will use the same figures and we will implement generic operation signatures in the positions of object types. The generic operations and their graphic representations were explained earlier in Chapter 5.

Operations of the object types 'General Manager', 'Urban Environment', 'Urban Elements' and 'Psychological Precategorization' are presented in Figure 8.14. The object types 'General Manager' and 'Urban Environment' might operate *all* possible operational functions, because they are responsible for the functionality of the whole data structure. The object types 'Urban Elements' and 'Psychological Precategorization' have all operations except 'export' and 'copy'. They cannot copy or export object types for themselves.

Operations of the object types 'Model Representation' and 'Multiple Representation' are presented in Figure 8.15. Both model types have the whole sets of operational signatures because they are dealing with graphic objects. The object type 'Multiple Representation' has a sub-object type called 'Object Types'. 'Object Types' has its own operational signature: 'create a new instance', 'destroy an instance', insert a relation', 'remove a relation', 'modify value' and 'obtain help'.

The next presentation of operations is in Figure 8.16. The object type 'Recognition' has the whole set of possible operational signatures, the same as 'Units Manager', which is responsible for the operations of sub-object types 'Global Units' and 'Microunits'. 'Global Units', 'Microunits', 'Typology of Urban Elements' and 'Typology of Microunits' have the same sets of operational signatures: 'produce a form of properties and relations of an instance', 'produce an overview of related instance', obtain related instance', 'ownership of an object type' and 'obtain help'.

Figure 8.14
Operations of the object types: 'Park Area', 'Recreation Area', 'Industrial area' and 'Transportation

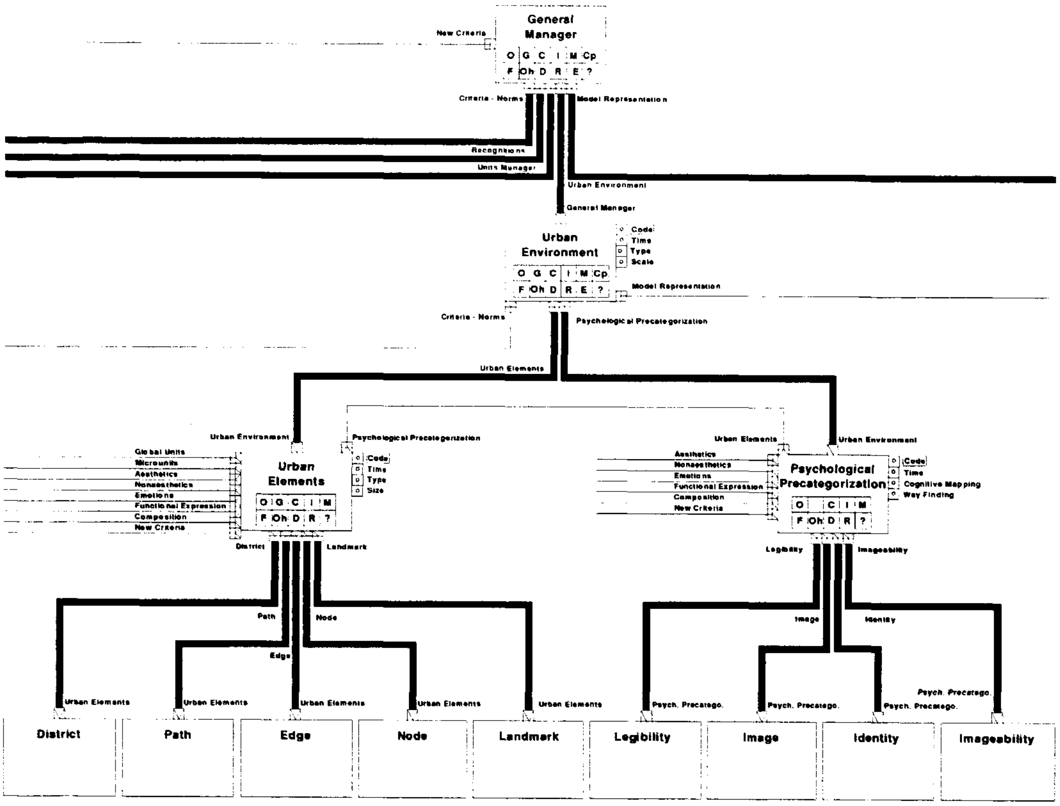


Figure 8.15
Operations of object types: Model Representation and Multiple Representation

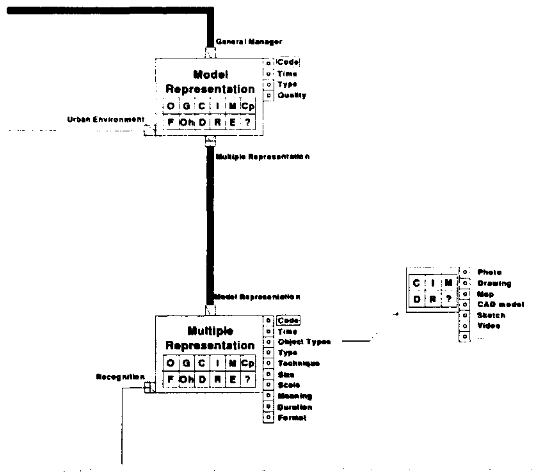


Figure 8.17 presents the operational signatures for the object type 'Criteria - Norms' and its specializations 'Aesthetics', 'Nonaesthetics', 'Emotions', 'Functional Expression', 'Composition' and 'New Criteria'. The object type 'Criteria - Norms' has the whole set of generic operations. All other object types presented in the schema have the following operations: 'produce a form of properties and relations of an instance', 'produce an overview of related instance', 'obtain related instance', 'ownership of an object type', 'create a new instance', 'destroy an instance', 'insert a relation', 'remove a relation' and 'obtain help'.

Figures 8.14, 8.15, 8.16 and 8.17 all together form one schema of basic object types operations. All other schemas will be presented as a map of operations in Appendix 2.

Figure 8.16
Operations of the object types: 'Recognition', 'Units Manager', 'Global Units', 'Microunits', 'Typology of Urban Elements' and 'Typology of Microunits'

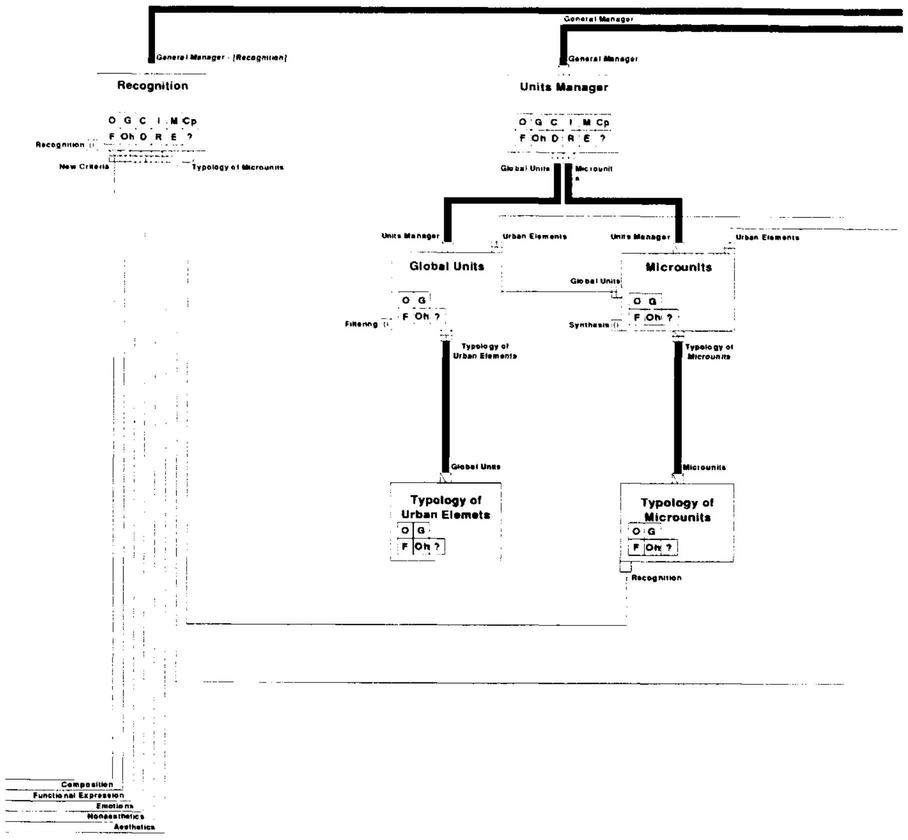
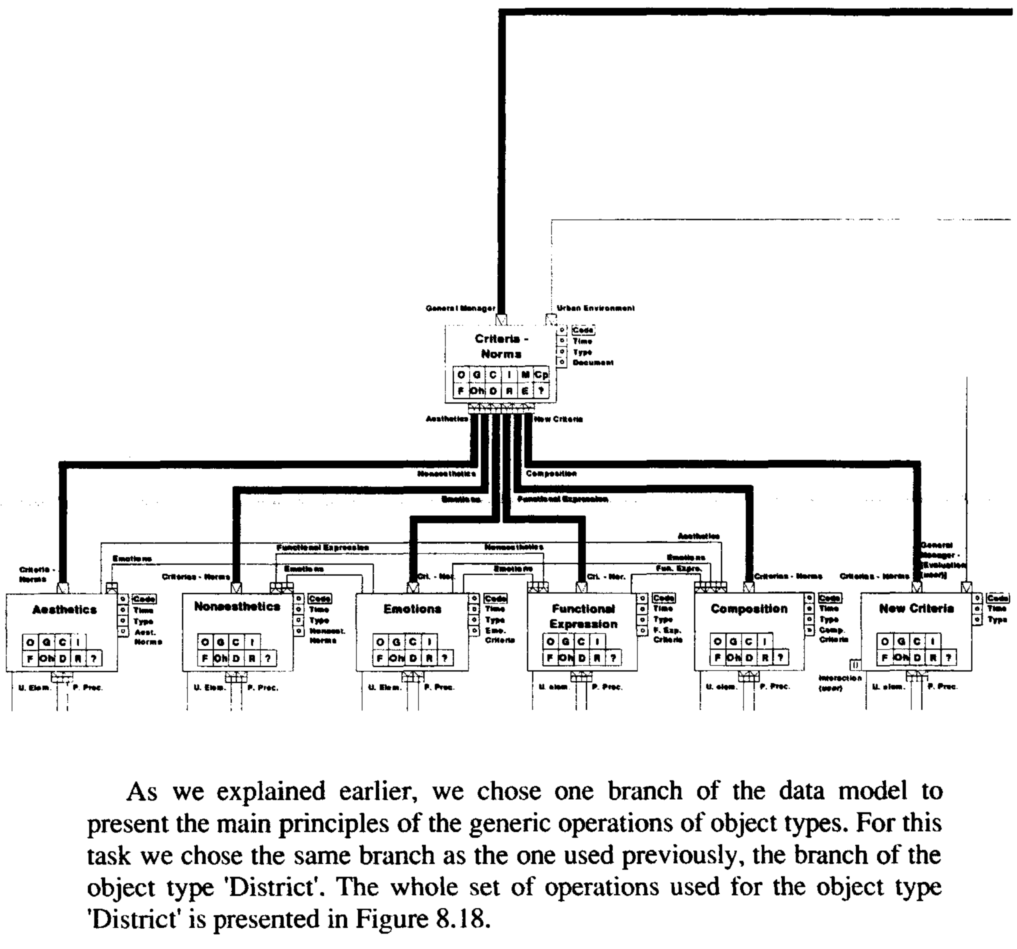
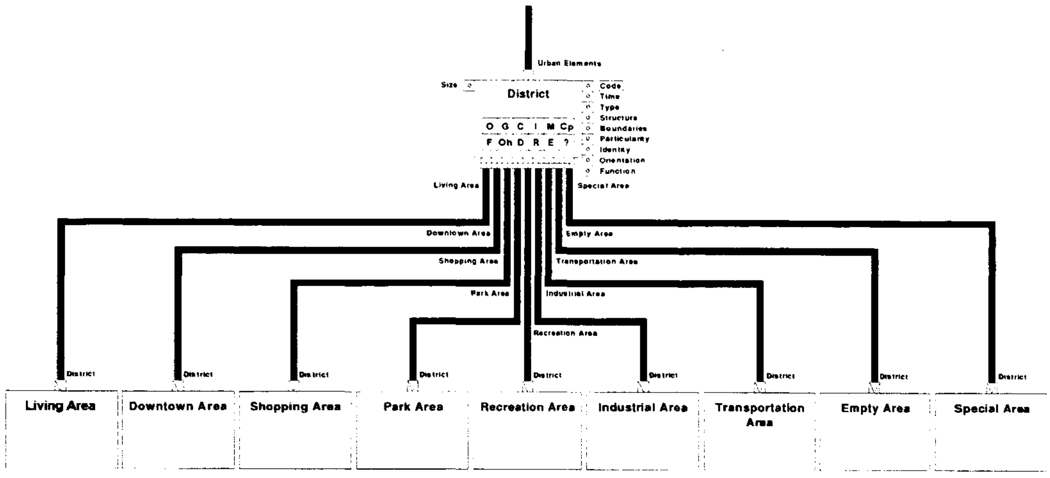


Figure 8.17
Operations of the object type: 'Criteria - Norms' and its sub-objects



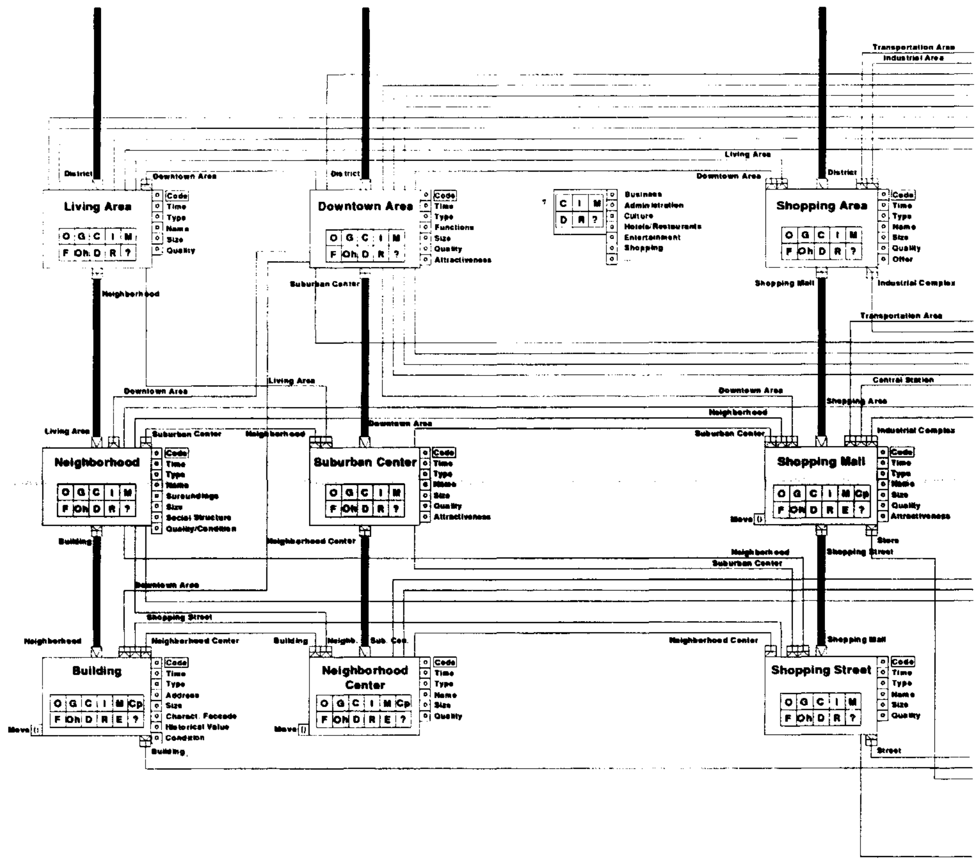
As we explained earlier, we chose one branch of the data model to present the main principles of the generic operations of object types. For this task we chose the same branch as the one used previously, the branch of the object type 'District'. The whole set of operations used for the object type 'District' is presented in Figure 8.18.

Figure 8.18
Operations of the object type 'District'



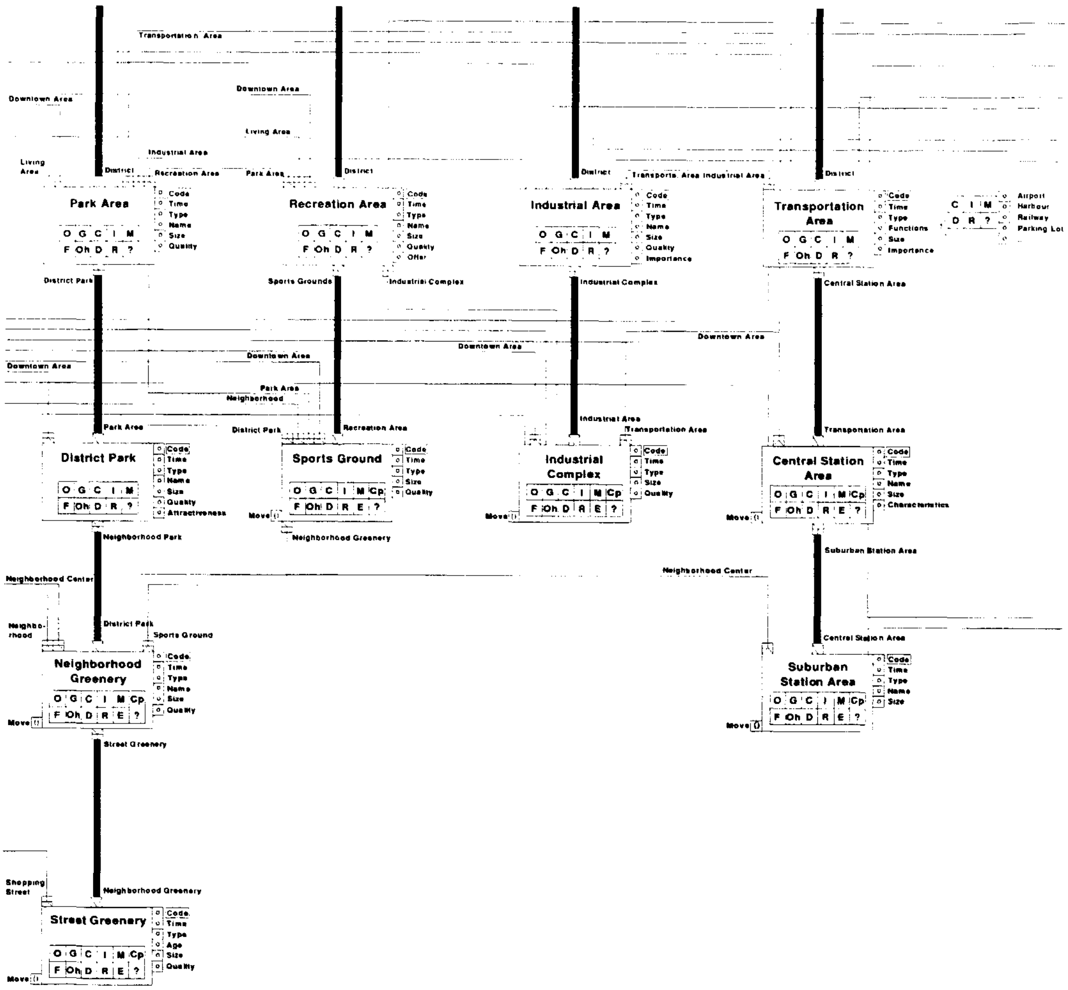
The object types 'Living Area', 'Downtown Area', and 'Shopping Area' and their specialization object types are presented in Figure 8.19. The object types 'Living Area', 'Downtown Area', 'Shopping Area', 'Neighborhood', 'Suburban Center' and 'Shopping Street' have the following operations: 'produce a form of properties and relations of an instance', 'produce an overview of related instance', 'obtain related instance', 'ownership of an object type', 'create a new instance', 'destroy an instance', 'insert a relation', 'remove a relation', 'modify value' and 'obtain help'. The object type 'Downtown Area' has its own sub-object type 'Functions' with its operational signatures: 'create a new instance', 'destroy an instance', 'insert a relation', 'remove a relation', 'modify value' and 'obtain help'. The object types 'Shopping Mall', 'Building' and 'Neighborhood Center' have the special operation 'Move', and for this reason they have the full sets of operational signatures.

Figure 8.19
Operations of the object types: 'Living area', 'Downtown Area' and 'Shopping Area'



The object types 'Park Area', 'Recreation Area', 'Industrial Area' and 'Transportation Area' and all their specialization object types are presented in Figure 8.20. 'Park Area', 'Recreation Area', 'Industrial Area', 'Transportation Area' and 'District Park' have the following operations: 'produce a form of properties and relations of an instance', 'produce an overview of related instance', 'obtain related instance', 'ownership of an object type', 'create a new instance', 'destroy an instance', 'insert a relation',

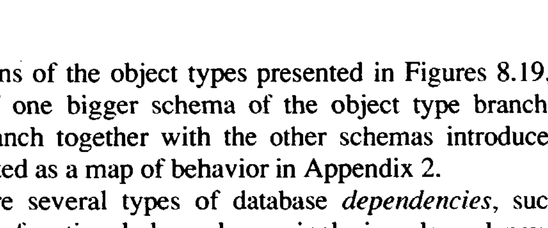
Figure 8.20
Operations of the object types: 'Park Area', 'Recreation Area', 'Industrial Area' and 'Transportation Area'



'remove a relation', 'modify value' and 'obtain help'. The object type 'Transportation Area' has its own sub-object type 'Functions' with its own operations: 'create a new instance', 'destroy an instance', 'insert a relation', 'remove a relation', 'modify value' and 'obtain help'. Object types that have the special operation 'Move' have the full sets of generic operations. These object types are 'Industrial Complex', 'Central Station', 'Neighborhood Greenery', 'Suburban Station' and 'Street Greenery'. All these object types represent movable objects from the urban environment.

The object types 'Empty Area' and 'Special Area' are presented in Figure 8.21. The object type 'Empty Area' has the following operations: 'produce a form of properties and relations of an instance', 'produce an overview of related instance', obtain related instance', 'ownership of an object type', 'create a new instance', 'destroy an instance', 'insert a relation', 'remove a relation', 'modify value' and 'obtain help'.
The object type 'Special Area' has a couple of basic operations: 'produce a form of properties and relations of an instance', 'produce an overview of related instance', 'obtain related instance', 'ownership of an object type' and 'obtain help'.

Figure 8.21
Operations of the object types: 'Empty Area' and 'Special Area'

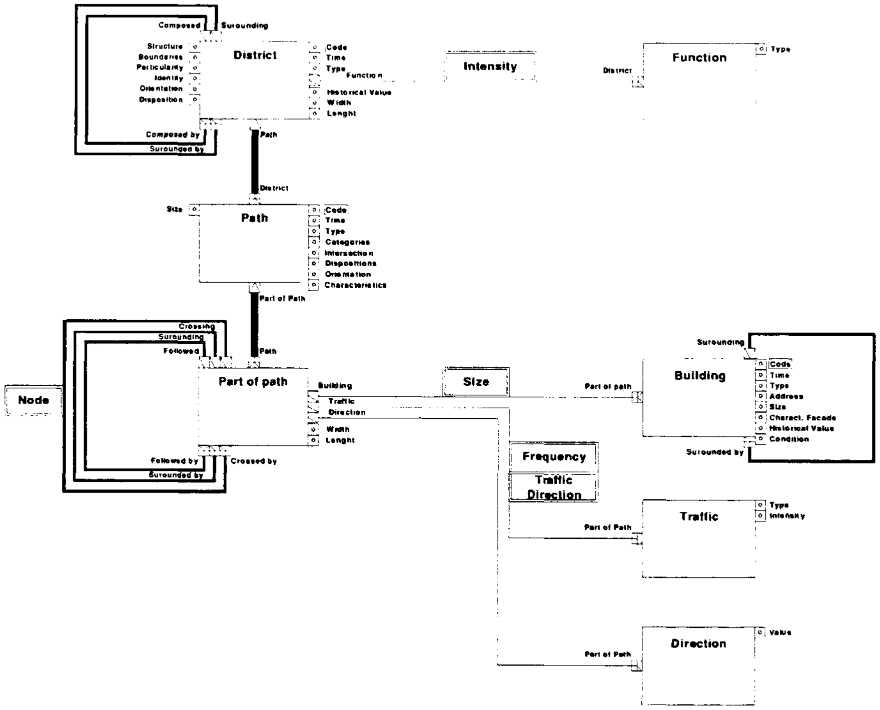


Operations of the object types presented in Figures 8.19, 8.20 and 8.21 are parts of one bigger schema of the object type branch 'District'. The 'District' branch together with the other schemas introduced here will be fully presented as a map of behavior in Appendix 2.

There are several types of database *dependencies*, such as existence dependency, functional dependency, inclusion dependency, etc. Here we will give a couple of examples of *existence dependency*. Existence dependency is defined whereby "the participation constraint specifies whether the existence of an entity depends on its being related to another entity via the relationship type" (Elmasri & Navathe, 2000, p. 57).

Additional relationships between the urban elements 'District' and 'Path' and their semantic categorization are presented in Figure 8.22. The schema uses the information coded in the generalization/specialization hierarchy and the two schemes are coupled. The schema presents only the relevant parts of 'District' and 'Path'.

Figure 8.22
A part of the user view of object types 'District' and 'Path'



In the scheme, a 'District' is represented by:

- A member "type" (inherited from "Urban Elements") expresses the characterization of a 'District' in terms such as introvert, extrovert, isolated etc.

- Further attributes are, for example, geographic orientation, disposition, kinds of boundary, width and length (inherited) and a historical value to indicate memorials etc. inside the 'District'.
- A 'District' can be composed of 'Sub-Districts'. A 'District' can itself be a 'Sub-District'.
- A 'District' can own 'Paths'. These are the 'Paths' situated inside the 'District'. That means each 'Path' will be designed 'out of' a 'District' and each 'Path' will be identified inside its corresponding 'District'.
- The 'Sub-Districts' and the owned 'Paths' build the structure of a 'District'.
- Other 'Districts' can surround a 'District'.
- A 'District' has 'Functions' like living, culture, recreation and shopping functions. Each 'Function', which belongs to a 'District', can be characterized by its 'Intensity'.

A 'Path' is the owner of its parts. A 'Path' can be divided into as many parts as necessary. It can also be just one. The parts are partitions of a 'Path' with different characteristics of that 'Path'. The partitioning of a 'Path' is flexible, which means that a path can be described as accurately as necessary. For these reasons we have modeled the characteristics of a 'Path' related to its parts.

In Figure 8.22 we describe a 'Path' (through its parts) as follows:

- If the 'Path' is a street, the member 'type' (inherited from 'Urban Elements') expresses its characteristics in terms of boulevard, avenue, passage, etc. If the 'Path' is a canal, 'type' expresses its characteristics in terms of 'Big', 'Small', 'Industrial', etc. This domain of the value of a member depends on the specialization of the object type 'Path'. Perspective-DB chooses the right domain automatically.
- Further attributes are 'Width' and 'Length' (inherited).
- A 'Path Part' can follow a certain 'Path Part' or can be followed by another 'Part'.
- A 'Path Part' can cross or be crossed by another part. The relationship with the object type 'Node' specifies which 'Node' is placed at the crossing. This relationship 'Node' is another perspective on the object type 'Node', which is visible as a specialization of 'Urban Elements' in Figure 8.6.

- A 'Path Part' can be surrounded by other parts or can surround other 'Paths'. Surrounded 'Paths' do not cross a 'Path' but are situated near a considered 'Path'.
- Other elements such as 'Buildings' can be situated beside a 'Path'. Here it is specified on which side of the 'Path' a 'Building' is situated.
- 'Traffic' is ongoing on a 'Path' (part). Characteristics of the 'Traffic' can be specified, such as the 'Direction' and the 'Frequency' of the 'Traffic'.
- The geographical direction gives the compass-orientation.

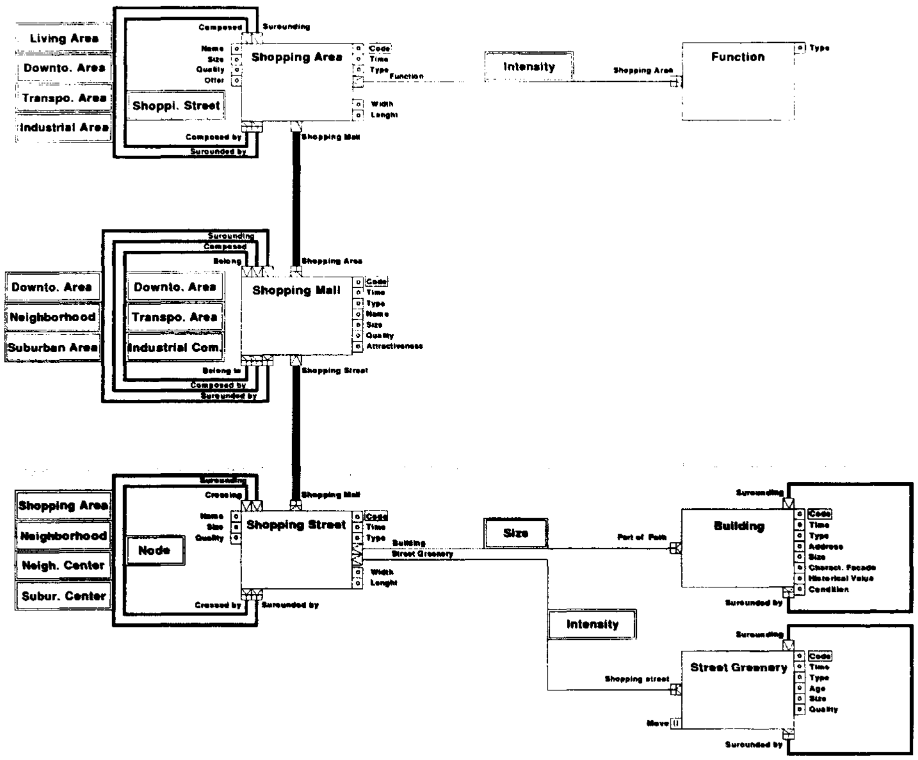
'Building', 'Traffic' and 'Direction' are also object types, which are specialized from generalization objects; they are not described here.
The thin connectors going between, for example, 'Part of Path' and 'Building', 'Traffic' and also 'Direction' specify relationships that are not 'ownership'. That means, for example, that a 'Building' can be created without needing to situate it on a 'Path'.

The next example will present the relationships between the urban elements 'Shopping Area', 'Shopping Mall' and 'Shopping Street' (see Figure 8.23).

In the scheme, the 'Shopping Area' is presented by:

- A member 'type' inherited from 'District' expresses the characterization of a 'Shopping Area' in different terms, such as expression, connectivity, extrovert, etc.
- The following attributes are, for example, 'Name', 'Size', 'Quality' and 'Offer', and indicate the area.
- A 'Shopping Area' can be composed of 'Sub-Shopping Areas', and a 'Shopping Area' can in itself be a sub-area. A 'Shopping Area' can be composed of 'Shopping Streets'.
- Other 'Shopping Areas' can surround a 'Shopping Area'. It can be surrounded also by 'Living Area', 'Downtown Area', 'Transportation Area' and 'Industrial Area'.
- 'Shopping Area' can own 'Shopping Mall'. This is 'Shopping Mall' situated inside the 'Shopping Area'. That means each 'Shopping Mall' will be designed 'out of' a 'Shopping Area' and each 'Shopping Mall' will be identified inside its corresponding 'Shopping Area'.
- A 'Sub-Shopping Area' and the owned 'Shopping Mall' build the structure of a 'Shopping Area'.

Figure 8.23
A part of the user view of the object types 'Shopping Area', 'Shopping Mall' and 'Shopping Street'



- A 'Shopping Area' has a 'Function' like living, entertainment, culture, etc. Each 'Function' which belongs to 'Shopping Area' can be characterized by its 'Intensity'.

We describe in the scheme 'Shopping Mall' the following:

- A 'Shopping Mall' belongs to 'Shopping Area' and to itself. 'Shopping Mall' might belong to 'Downtown Area', 'Transportation Area' and 'Industrial Complex'.
- The following attributes, such as 'Name', 'Size', 'Quality' and 'Attractiveness', indicate 'Shopping Mall'.

- A 'Shopping Mall' can be composed of 'Sub-Shopping Malls'. 'Shopping Mall' can be by itself a 'Sub-Shopping Mall'.
- A 'Shopping Mall' can be surrounded by other 'Shopping Malls'. 'Downtown Area', 'Neighborhood' and 'Suburban Area' can also surround 'Shopping Mall'.
- 'Shopping Mall' is the owner of 'Shopping Street'. A 'Shopping Mall' can be divided into as many 'Shopping Streets' as necessary, or just one. The 'Shopping Street' is a partition of 'Shopping Mall' with characteristics different to the 'Shopping Mall'. The partitioning of a 'Shopping Mall' is flexible, which means the 'Shopping Mall' can be described as accurately as necessary. For these reasons we modeled the characteristics of a 'Shopping Mall' related to its 'Shopping Street'.

In Figure 8.23 we also describe 'Shopping Street' as follows:

- The 'Shopping Street' belongs to 'Shopping Mall' and to 'Shopping Area'.
- The main attributes of the 'Shopping Street' are 'Name', 'Size', 'Quality', 'Width' and 'Length' (inherited).
- A 'Shopping Street' can cross or be crossed by another 'Shopping Street'. The relationship with the object type 'Node' specifies which 'Node' is placed at the crossing. This relationship 'Node' is another perspective on the object type 'Node', which is visible as a specialization of 'Urban Elements' in Figure 8.6.
- Other 'Shopping Streets' can surround a 'Shopping Street'. Surrounded 'Shopping Streets' do not cross a 'Shopping Street' but are surrounded near a considered 'Shopping Street'. 'Shopping area', 'Neighborhood', 'Neighborhood Center' and 'Suburban Center' can also surround a 'Shopping Street'.
- Other elements such as 'Buildings' can be situated beside a 'Shopping Street'. Here it is specified on which side of the 'Path' a 'Building' is situated. 'Building' has its own attribute. 'Building' can be characterized by its 'Size'.
- 'Street Greenery' belongs to the 'Shopping Street', and its main attributes are 'Age', 'Size' and 'Quality'. Characteristics of the 'Street Greenery' can be specified as 'Intensity'.

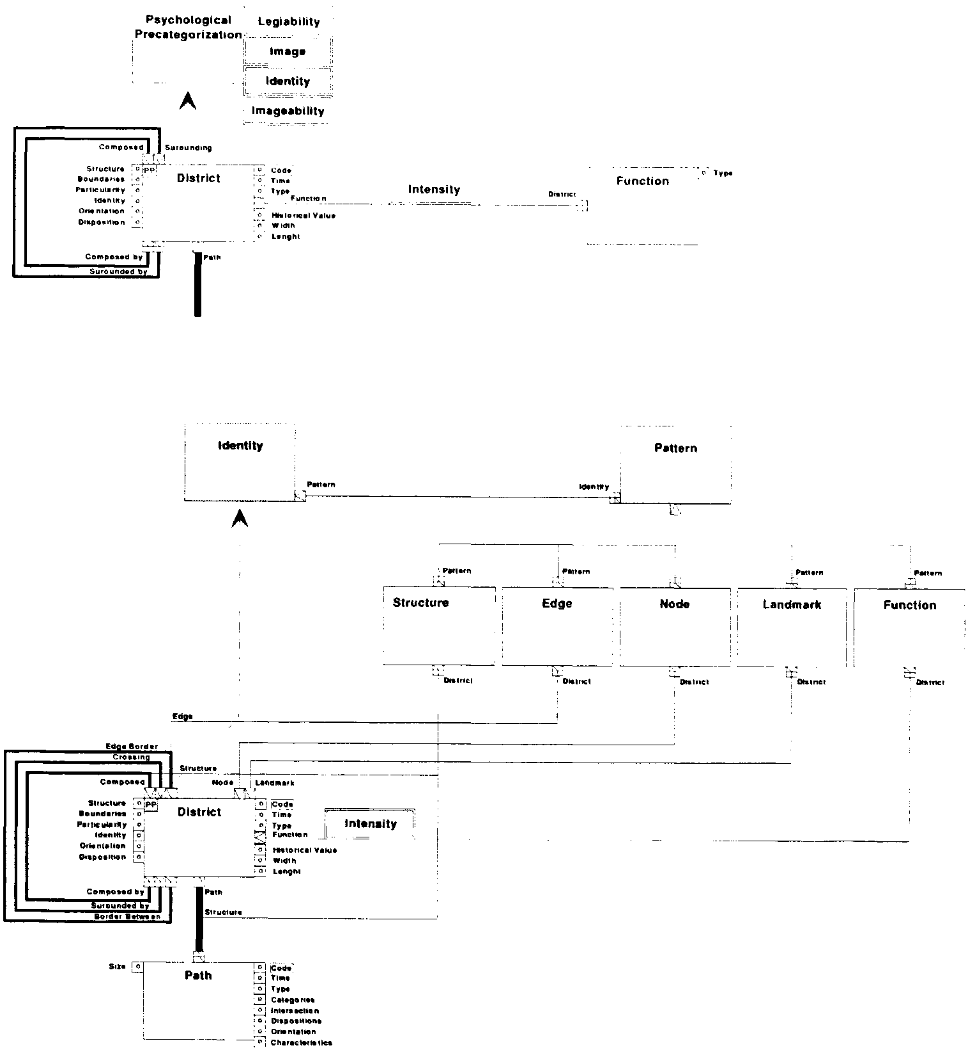
Here we discuss 'Shopping Street' as 'Pedestrian Street'. There are also some object types, 'Traffic' and 'Direction', which are specialized from generalization objects, and they are not described here.

The last example of dependence is the relationships between 'Urban Elements' and 'Psychological Precategorization' described with the object type 'District' as an example for an 'Urban Element', and the object type 'Identity' as an example for 'Psychological Precategorization'. The object type 'Psychological Precategorization' is a member of 'District', inherited from the generalization 'Urban Element' (see Figure 8.24). We will briefly explain only object types and their relations that belong to the dependence between 'District' and 'Identity', because the example is complex and we only want to give an overview.

When this member [button 'PP' - 'Psychological Precategorization'] is activated, it pops up its embedded objects, in this case 'Legibility', 'Image', 'Identity' and 'Imageability' (see the schema at the top of Figure 8.24). The Perspective-DB supports an active object schema. All schema elements can be activated by clicking them. When 'Identity' is activated the object type pops up, showing its members. Here only the relationship 'Pattern' as a member of 'Identity' is presented. This relationship represents that identity is associated, among other things, with a certain pattern in the human mind. The relationship between 'Identity' and 'Pattern' enables us to identify a district by a certain pattern, or to find the pattern which identifies a certain district. The 'Pattern' itself is composed of the object types 'Function', 'Edge', 'Node', 'Landmark' and 'Structure'. The pattern, which contributes to identifying a district, is composed of certain functions, edges, nodes, landmarks and the structure of the district.

The general role of the schemas for our Design Tool is to graphically represent the structure of the data system. The important process of the Design Tool coding will be based on the graphic structure of the data system and will be the next phase of the work. The data system is a part of the Design Tool which provides the main support to its functionality. Each part of the data structure, presented in the schemas, is hierarchically subdivided into the functional and logical order of related elements - objects. This order of functional hierarchy is the first step in the logic of the 'query'. The query of the Design Tool starts from the hierarchical top of the data structure, e.g. 'general manager' and follows the branch of the 'query task' to the smallest units of one object, the lowest hierarchical level, e.g. an object of the 'urban environment' such as 'building'. The 'query question' activates the specific branch of the data structure, via the 'general manager' or just 'managers' of

Figure 8.24
A part of the user view of the object types 'District' and 'Identity'



the objects, and all objects and relationships related to the query task. The query process can deal with very specific 'query questions' via multi-object relationships, e.g. a query of an urban pattern. The multi-object relationship is the process of 'parallel processing' several objects and their relationships at the same time. The results of queries – actually their *routes* – will be memorized in the 'meta-data' which is a part of the 'data models' (see Figure 8.3). Generally, this process of memorizing *routes* is called a 'self-learning process'. The structure of 'meta-data' is not very important for our data models and we do not need to explain it here; it is only important to mention that it exists.

Tool requirements derived from the object-oriented database model

In this chapter we have presented the object-oriented database model of the Design Tool. This model will be very important for the conceptual modeling of the tool. We will now propose the set of requirements derived from the object-oriented database model.

The tool requirements derived from object-oriented database model might be as follows:

- applicable for analyzing different urban environments, in terms of type, scale, density, morphology, etc.
- applicable for analysis of physical and psychological elements of the urban environment,
- applicable for different [multimedia] input data representing the urban environment,
- applicable for analyzing very complex urban structures, such as urban patterns,
- applicable for performing analyses from global to 'local' scale, such as an analysis of a city map or, on the other hand, one street of the city,
- applicable for combining different tasks related to an urban environment, such as performing an analysis of way-finding according to a given cognitive map,

- applicable for analyzing specific tasks related to the urban environment, such as greenery or traffic,
- applicable for integrating analyses of different specific tasks, such as traffic with mobility,
- applicable for the use of cognitive mapping, way-finding and navigational approaches in the analysis of any urban environment,
- applicable for recognizing and selecting a similar procedure [objects] from the existing stored database,
- applicable for self-learning tasks, such as remembering procedural routes,
- applicable for support from an external database, such as linkage to different remote databases via the Internet,
- applicable for creating output of analyses of the urban environment in multimedia form, such as text, images or even animation, and
- applicable for interacting with different computer applications, such as GIS, CAD or even text processors.

This is the set of main requirements, but the list might be extended to include some other, less important requirements.

The requirements proposed here are the last set of requirements for the Design Tool derived from the object-oriented database model. These requirements, together with the requirements derived from Lynch's theory and from the Computer Cognition Model, will be the platform for the conceptual modeling of the Design Tool. In the next chapter we will explore the *conceptual* and *functional model* of the Design Tool.

Conclusion

In this chapter we developed the object-oriented database model for our Design Tool. The object-oriented database model is based on the hierarchical structure of the elements of the urban environment and its cognition. The database model explored in this chapter overlaps the urban structure in all details. Together with previous developments in the domain of urban and cognitive theories, we are now prepared for the final development of the *conceptual model* of the proposed Design Tool.

Before we developed the detailed database model, we provided an overview of the top-to-bottom pyramidal system architecture, which consists of seven different and functional layers. The complex object-oriented database model, explored using Perspective-DB technology, was presented in two parts, and broken down into several schemas. The first part presents the object types and their attributes and relations. The second part presents the behavior by their operations. In Appendix 1 we will present all other schemas belonging to the object-oriented database model. At the end of this chapter we set out a couple of very detailed examples of dependencies.

Therefore, the database model for the proposed Design Tool developed through Perspective-DB technology provides the means to present and process the complex structure of any urban environment and its cognition within a multi-media presentation of application objects in their semantical context, and to exploit them within this context.
In this chapter we also outlined the set of main requirements derived from the object-oriented database model.

The next chapter will focus on developing the *conceptual model* of the proposed Design Tool. The *conceptual model* will be an integration of different modules, some of which have already been developed in the last three chapters. The *conceptual model* of the Design Tool is the central point of this research.

CHAPTER 9

TOOL DESIGN - THE CONCEPTUAL MODEL OF THE DESIGN TOOL

***Task of the chapter**: the conceptual model and 'functional model 'play crucial roles in developing a realistic and useful Design Tool, both based on the 'theoretical framework' and 'system framework' derived from the theory of 'urban forms', computer cognition and object-oriented database systems, for design practice.*

Introduction

In the previous three chapters we developed important parts of the *conceptual model* of the Design Tool. First we systematized urban elements in their hierarchy for both the visual quality of urban the environment and its cognition. Then we developed the Computer Cognition Model with the functions of 'recognition', 'filtering' and 'synthesis' all based on the integration of cognitive and urban approaches. The last development was in the domain of the object-oriented database system [OODB] and resulted in the object data model. The object data model will support the *conceptual model* of the Design Tool in all functional instances.

The *conceptual model* of the *Design Tool* will therefore be the result of a synthesis of material drawn from the areas of design, cognition and computation. First in this chapter we will explain the framework for the Design Tool. The framework consists of the 'theoretical framework' and the 'system framework'. Both frameworks influence the *conceptual model* of the Design Tool and its functionality by intending to be unified and coherent, as well as realistic and useful based on different tool requirements.

In this chapter we will develop the *conceptual model* of the Design Tool. The *conceptual model* consists of several modules. There are two main groups of modules: the first one is the 'input-output' of data, and the second one consists of the main functional modules of the Design Tool. Each of the modules will be explained in terms of their functionality and relationships with each other. The "recognition" module (see also Chapter 7) will only be indicated in terms of its function, because this module is working as a plug-in, adopted from Ullman, as examined in Chapter 4. All other modules will be presented in more detail.

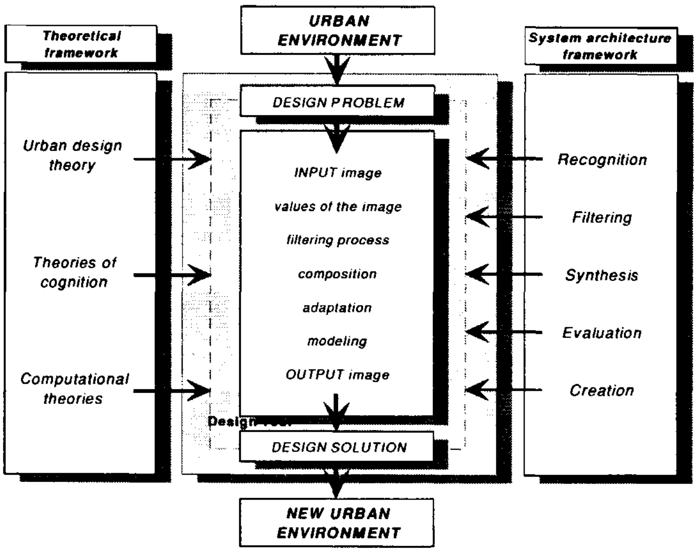
The *functional model* of the Design Tool will be based on and will follow the structure of the *conceptual model.* The main task of the functional model is to show explicitly the position of the object-oriented database system [OODB] within the Design Tool structure, and its support of the modules. The OODB system is a 'background' module, which supports almost all Design Tool modules. The structure of the OODB system and data models was presented in the previous chapter . The real functionality of the Design Tool, with more possibilities and details, will be presented in the next chapter, when we will perform tests by simulating the tool.

The proposed *conceptual* and *functional models* of the Design Tool are intended to be applicable for design practice according to our main 'cognitive mapping' approach. The main task of this research and the Design Tool concept is to use knowledge of 'cognitive maps' in different phases of the design process. At the end of the chapter we will set out a short conclusion on the aim of this chapter.

Framework of the Design Tool

To structure our *conceptual model* of the Design Tool we located a highly appropriate framework. Here we will use different frames of the same tool to describe the same situation in different ways and because that one frame is not enough to describe such a complex system. The framework of our Design Tool consists of the 'theoretical framework' and 'system framework' (see Figure 9.1). We selected an initial frame (theoretical) to meet some given conditions and we assigned and selected subframes (system) to represent additional details. The frameworks are simplifications of 'matching reality' and "their real power depends upon additional knowledge about interactions between them" (Minsky, 1975, p. 114). In the example of our research, the real power will be in a field between the 'theoretical' and 'system' frameworks, in a field of the Design Tool where the elaborated knowledge of this research will be implemented (see also Figure 9.1).

Figure 9.1
Frameworks of the *Design Tool*



The frameworks, the theoretical and system architecture, have a crucial influence on the conceptual model of our Design Tool. Figure 9.1 shows the position of the Design Tool in its place between the frameworks. The influence of the theoretical and system frameworks on the Design Tool is visually presented in the scheme. The Design Tool is symbolically located in a simplified design process, which can influence the urban environment through a new design solution. The Design Tool is presented in the scheme in terms of the conceptual model's modules. They will be explained in detail later in this chapter. The theoretical and system frameworks have an important role in the *conceptual model* of the Design Tool.

The theoretical framework of the Design Tool

The theoretical framework as an initial frame will be directly reflected in the *conceptual model* of Design Tool. There are three subframes in the theoretical framework, representing the main theories on which our research is based: urban design theory, theories of cognition and computational theories.

Urban design theory presented in the theoretical framework is based on Lynch's *theory of urban form* described in the book 'Image of the City' (1960). Lynch introduced cognitive perception of the urban environment and the concept of *cognitive mapping* in the domain between space representation and cognitive psychology. We provided a critical analysis and indicated the main points of the incompleteness of Lynch's theory and suggested new possibilities. The benefit of such new possibilities is a set of new elements or criteria applicable to a more universal urban and cultural context. The result of the analysis of Lynch's theory is a definition of needs and requirements providing a platform on which we can build the *conceptual model.* More details on urban design theory and the analysis of Lynch's book are presented in Chapters 3 and 6.

Theories of cognition is a theoretical framework that consists of two theories: *computer vision* conceived by David Marr and *high-level vision*, the theory of computational face recognition conceived by Shimon Ullman. The theory of 'computer vision' focuses on several aspects of visual information processing, different ways in which motion may be used to recover structure from a sequence of images and how the results of such processes can be represented efficiently for recognition. 'High-level vision' or *object recognition and visual cognition* are related to some aspects of computer visual perception and the cognition of objects. Both theories and their influence on the conceptual model of the Design Tool are presented with their main definitions in Chapter 4.

Computational theories as a framework are focused on the *information system technology* and *object-oriented database technology*. The information system is a collection of activities that regulate the sharing and distribution of information and the storage of data. The activities and organization of an urban environment do not operate at random or in isolation. They must be coordinated by a flow of information within the environment and between the environment and the outside world. *Object-orientation* is a recent strategy for organizing systems as collections of interacting objects that combine data and behavior. *Object-oriented databases* were proposed to meet the needs of more complex applications. The key power of the object-oriented database is the fact that the designer can specify both the structures of complex objects and operations that can be applied to these objects. The database and program applications can be developed together for ease of conceptualization, implementation, maintenance and potential re-use. In our research, the database system is a crucial part of the functionality of the Design Tool. More details on object-oriented database systems and our object-oriented database model are presented in Chapters 5 and 8.

The system framework of the Design Tool

The system framework defined additional operational details necessary for developing the *conceptual model* of the Design Tool according to the given requirements and goals. All subframes of the system framework are directly implemented in the *conceptual model* as well as in the Design Tool as its integral functional modules. The system architecture consists of several subframes, such as: recognition, filtering, synthesis, evaluation and creation (see also Figure 9.1).

Recognition is a subframe that directly relates to the theoretical definition of shape recognition, adopted from Ullman's theory of 'high-level vision' and 'face recognition'. Recognition is a process adopted as a 'plug-in' application, which will be used independently as a part (module) of the Design Tool.

Filtering is a subframe of the system framework, which defines the process of shape definition by filtering and using the 'typology of microunits'. The process of filtering is based on the knowledge of transforming a 'bitmap' image into a 'vector' image, which is necessary for 'directly using and manipulating data related to an image scene'.

Synthesis is a subframe that defines the process of 'image definition' by using the 'typology of urban elements'. Synthesis is also a finishing process of 'filtering' where all elements from an image scene are defined in terms of their relationships with their correlated database.

Evaluation is a subframe of the system framework, which defines the process of operational execution of users' tasks and needs. Together with the user, the Design Tool in the process of 'evaluation' will make all necessary analyses, changes and evaluations of an image scene, according to the given task. Evaluation is a direct process of interaction between the Design Tool and user.

Creation is a subframe that defines the process of final execution of the 'evaluation' process. 'Creation' and 'evaluation' consist of several small sub-tools, all necessary for finishing given tasks, which will be presented in a 'new image scene' or in text.

All subframes of the system framework will be presented in detail in the part of this chapter that explains the structure of the *conceptual model* of the Design Tool. The best summary of the subframes and their functionality will be presented with examples during the testing simulations of the case objects in Chapter 10.

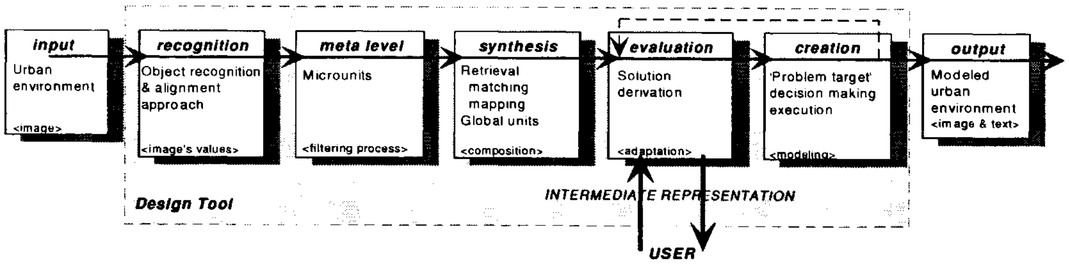
The conceptual model of the Design Tool

In the creation of the Design Tool, the *conceptual model* is first used to produce a high-level description of the structure of the Design Tool, and then to translate this structure into functional relations. The *conceptual model* is presented as a *schema*[1] that represents specific modules of the Design Tool. The schema is a collection of linguistic and graphic representations that describe the structure of interactions and functions of different logical segments of the Design Tool. Each segment (module) has its own logical and functional structure, which together build up the Design Tool.

[1] *Schema* is defined here as an outline of the Design Tool's functional elements [modules].

It is possible to make a distinction between a *description* of the *conceptual model* and the *conceptual model itself* [*meta* model]. The description of the *conceptual model* is the *conceptual model schema*, which will be explained later. The 'conceptual model itself' [*meta* model] summarizes the *conceptual model schema* and does not need to be explained here. The description of the *conceptual model* is an extension of the conceptual model, which *understands* the model. The conceptual model is shown in Figure 9.2. The conceptual model schema displays the structure of Design Tool.

Figure 9.2
The *conceptual model* of the *Design Tool*



Structure of the Design Tool

The *theoretical* and *system frameworks* (see Figure 9.1) influence the structure of the *conceptual model* of the Design Tool. From the one side the theoretical background has its influence through the different adopted approaches, which were elaborated in Chapters 3, 4 and 5. From the other side the system framework elaborates different features and requirements of the Design Tool, as developed in chapters 6, 7 and 8. The focus of explanation will be the main structural and functional characteristics of the *conceptual model*, ignoring theoretical and other influences, because they have already been explained earlier.

The *conceptual model* presents the structure of the Design Tool. The *conceptual model* consists of several modules. There are two main groups of modules: *input-output* modules and *tool* modules. Both groups of modules are important for the usefulness of the Design Tool. It is also possible to distinguish some other sub-groups inside the *tool* modules, such as the Computer Cognition Model, which was explained in Chapter 7. The position and function of the OODB system, which is a specific module supporting all other modules of the tool, will be explained later in this chapter.

The structure of the *conceptual model* of the Design Tool consists of seven modules. Each module is functional entirety through its own logical structure, inner functions and specific tool design. Here we will explain the crucial characteristics of each module and their relationships. We do not need to explain in detail all segments of each module because we are not going to make the tool design necessary for the coding phase. The task of this explanation is to highlight the main logic of the modules. The modules that will be explained are: *input, recognition, meta level, synthesis, creation* and *output*. Each of the modules will be explained in the order of their functional activities during the action of the Design Tool.

Input is the first module of the *conceptual model*. Figure 9.3 presents the structure of the 'input module', which consists of:

(1) *image*,

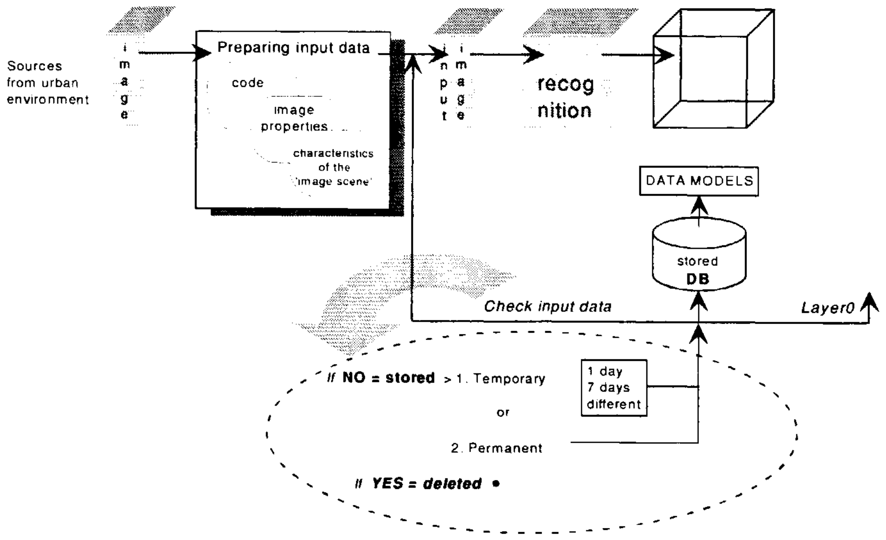
(2) *preparing input data* and

(3) *input image*.

The *image* is the source, which represents some aspects of the urban environment (called the *image scene* - for example, some photo of a particular urban environment). The *image scene* is always related to some problems or questions of the urban environment, which the 'Design Tool', in interaction with the user, can analyze and solve. The image might be a map, drawing, sketch, photo, CAD model or something else. It should be a 2D or 2½D image. Most of the images are static, but sometimes might also be dynamic (some CAD models). The image could also be black and white or color and in a traditional (paper) or digital format.

There are two steps in preparing the image for processing in the Design Tool. The *first* step is the selection of the representative and proper image that will fit well according to the user's task. Only a proper image will give a successful result from the Design Tool process. The selected image must be of good quality and the user might have additional information about the image source, about the 'image scene' that is represented by the image and about the image itself. The *second* step is related only to an image that is in traditional paper form. The Design Tool works only with digital images and all traditional paper images must be scanned. The image can be scanned

Figure 9.3
The structure of the 'input module'



directly from the Design Tool, as one of several additional tools, but it is not a priority that can be explained here.

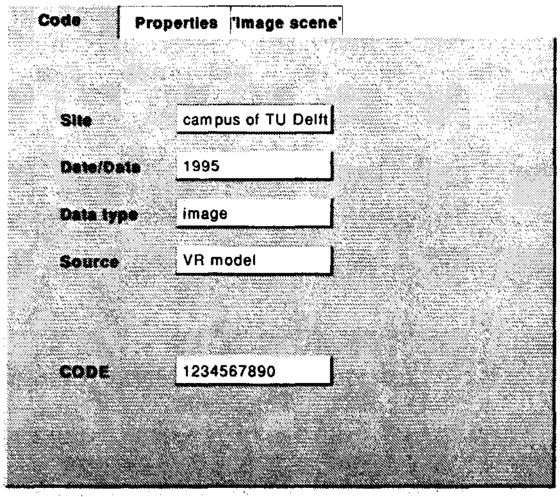
Preparing input data is the second phase in the 'input' module, which consists of: *code, image properties* and *characteristics of 'image scene'.*

(1) *code* - in the first window (see Figure 9.4), which appears by starting the Design Tool, the user can fill this in by typing in several important pieces of information related to the input image:

- *site* - location presented on the 'image scene' e.g. campus of TU Delft,
- *date/data* - the date of data source, e.g. 1995,
- *data type* - type of data represented in the form of e.g. image, or sketch, or 2D map, and
- *source* - information on which source the data was taken from e.g. VR model or Faculty of Geodesic Engineering.

In this first window there is also a position for *code*, which will be automatically presented as soon as all the input information is filled in and confirmed.

Figure 9.4
Preparing input data: 'code'



Site	campus of TU Delft
Date/Data	1995
Data type	image
Source	VR model
CODE	1234567890

(2) *image properties* - in the second window (see Figure 9.5) that appears on top of the first window after it has been filled in. This window can be filled with information related to image properties, such as *size* (width/height) presented in pixels, inches or centimeters, selected from a pop-up menu. The image size will be automatically filled in after the process of scanning or the process of taking the image from other digital sources/applications like *AutoCAD*. Their is also a possibility for each image to be *resized* to a format other than the original one, also presented with size (width/height) in pixels, inches or centimeters. Also presented are: *mode* for color such as RGB color or gray scale, then *image type* such as 2D, *file format* e.g. .jpg and *quality* e.g. good. The user must select all the right information (if this is not automatically done) from the pop-up menu, and then the third window will appear on top of the first two.

Figure 9.5
Preparing input data: 'image properties'

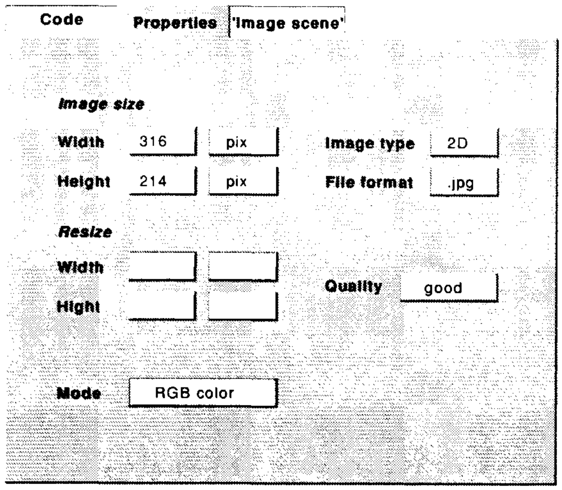


Image size

Width 316 pix

Height 214 pix

Image type 2D

File format .jpg

Resize

Width

Hight

Quality good

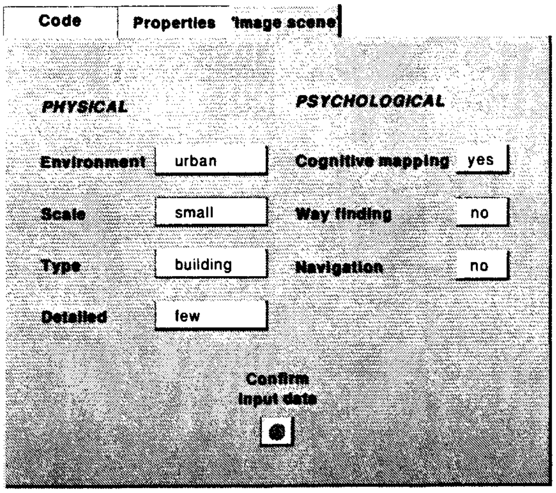
Mode RGB color

(3) *characteristics of the image scene* - in the third and last window (see Figure 9.6) for preparing input data. There are two types of information which the user has to fill in:

- *physical* - related to the urban environment presented in the 'image scene' such as: *environment* e.g. urban, *scale* e.g. small or 1:10000, *type* e.g. building or street and *detailed* e.g. few or rich (the user can select an option from the pop-up menu or fill it in if it is different to what is proposed), and
- *psychological* - related mostly to the cognitive task of the 'image scene' such as: *cognitive mapping, way-finding* and *navigation.* The user must select from the pop-up menu only yes or no from a window related to one of the psychological characteristics which has to be done during the Design Tool processing.

When all information is filled in, the user must confirm the input data by clicking on the correct button.

Figure 9.6
Preparing input data: 'image scene'



PHYSICAL		PSYCHOLOGICAL	
Environment	urban	Cognitive mapping	yes
Scale	small	Way finding	no
Type	building	Navigation	no
Detailed	few		

When the input data is confirmed by the user the tool will activate the process of data-checking through the database system. All codes are stored in a 'code library' as a part of the 'meta data' (see also Figure 9.3 and Figure 8.3). Automatically, after the very fast process of checking the input data, the first window will appear (again) with a new CODE or with a code that already exists in the 'code library'. If the tool does not find exactly the same data in the 'code library', then the tool will ask the user, with a small new window, to define for how long he or she would like to store that new input data. There are two possibilities:

(1) *temporary* - the user can select from the pop-up menu: one day, one week or different, for which the user can fill in the period which is convenient for his task, e.g. 13 days, or till 13 December 2005, and

(2) *permanent* - storage of data without any time limitations.

In both cases the tool will store the code and input image and all relevant data that is filled in the first three windows. It is also possible for the user to type in just a code in the first window if he or she needs to use a known, stored case. In this case of retrieval, the data-checking system will take all

the data and images related to that particular code and the process of 'evaluation' will be automatically activated.
If the checking system finds the same data in the stored database then the input data cannot be saved and all entered input data will be deleted automatically and the 'input image', taken this time from the stored data, will appear in the new window related to the process of 'evaluation'. In this case related 'data models' will also be automatically activated.

When the checking of new data entered is successfully completed [with *no* stored code in the 'code library'] and before the 'recognition' process is started, the tool automatically activates possible related parts of the database system: *data models* and *typology of microunits*. If data is retrieved from the stored database, the tool will automatically activate all relevant and related data: *data models, typology of urban elements, GIS data*, and some others [but not *typology of microunits*, because in this case the tool will jump directly to the 'synthesis' module]. The tool is then 'ready' for further processing.

The system of *coding* is adopted from Perspective-DB technology. This system of coding is based on numerical repetition, which can be the same in the hierarchical order that followed the data model structure, whereby each data object has a different code number. In this code system there is no possibility of doubled codes for one data object. Each data object, down to the smallest units, has its own code. For example, one 'image scene' consists of numerous objects and their codes. The hierarchical order related to *data models* also builds a code hierarchy, which results in *one* unique code for this particular 'image scene'. The hierarchical code structure will make the checking and taking of data through and from the 'code library' and database system very fast and efficient.

The *input image* is the third and last phase in the 'input' module. When the new *code* of input data appears in the first window, then the 'input image' is ready for the 'recognition' process. The 'recognition' process automatically starts the processing of the 'input image'. At the same time, one copy of the original 'input image' will be sent to the *Layer0* in the 'adaptation' module. More details on *Layer0* and its function will be given in the explanation of the 'adaptation' module.

Recognition, meta level and *synthesis* are the three modules related to the 'computer cognition model', which we already explained in Chapter 7. Here we will give a brief reminder of the main characteristics and functions of each of them.

Recognition is the second module of the Design Tool, and the first module in the Computer Cognition Model. Recognition, as explained previously, is a module completely adopted from Ullman. Chapter 4 presented Ullman's theory and system of shape recognition, called 'high-level vision', and there is no need to repeat it again here. The recognition module works as an independent part of the Design Tool, as a 'plug-in' application.

When the process of 'inputting' new data is finished and a new 'code' is established, the process of 'recognition' is automatically started. The Design Tool needs the process of 'recognition' for several reasons, and the three main ones are as follows:

(1) each selected digital (scanned) image is in a bitmap format and we need to transfer it to a vector format; the reason for this requirement is to allow calculation of the geometry of the objects from the image;

(2) each 'image scene' consists of several represented urban objects or patterns which cannot be manipulated, analyzed or changed into an image and we need to do this; the reason for this requirement is the different possibilities which the geometry of objects might offer, like the measurement of distances between objects or their heights; and

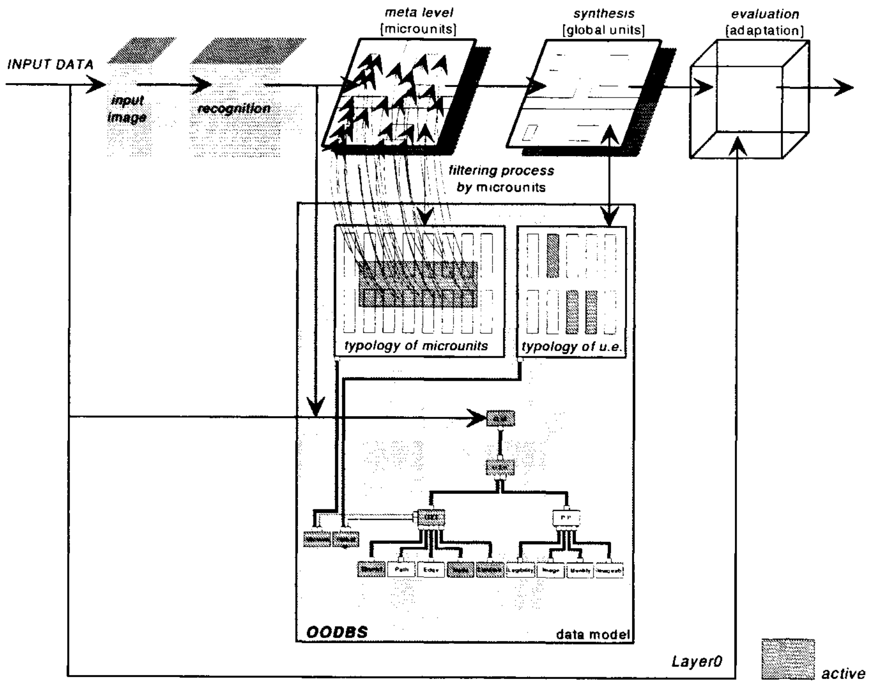
(3) each object from an 'image scene' should represent or consist of some data, or even relate to some data, but it is not a connection between objects from the 'image scene' and their corresponding data and we need to use data directly from the objects from the 'image scene'; the reason for this requirement is the possibility of integrating the object-oriented database system with the geometry of objects from the 'image scene'.

The process of 'recognition' cannot solve the proposed requirements and problems by itself, but it will help in the recognition of each object from an 'image scene'. It will 'separate' objects and 'recognize' patterns and will make a new image out of this process. The new image, which completes the process of 'recognition', will be prepared for further processing (*filtering* and *synthesis*), after which the 'input image' will be transformed and ready for further uses by the user.

The ***meta level*** is a module between the 'recognition' and 'synthesis' modules. This module is called the 'meta level' according to the process which should also belong to the 'synthesis' module. They are separated into two modules to make clear both very similar functional processes, which work almost simultaneously. The 'meta level' performs the process of filtering by microunits, as explained earlier in Chapter 7.

However, here we will explain the 'missing' part that relates to the functions of the data model with microunits in the filtering process. This part was not explained earlier [in Chapter 7] because the data model had not yet been developed. The continuity of the process presented in Figure 7.3 will be followed by Figure 9.7. Figure 7.3 indicates the position of the OODBS and data model, and their relations with the 'meta level' and 'synthesis'. Figure 9.7 shows how the filtering process works in interaction with the 'input image' and OODBS.

Figure 9.7
The 'meta level' module and the process of filtering by microunits
- *Example 1*



When the 'input' data is confirmed by the user, the three main processes then simultaneously start:

(1) checking the code through the 'code library' and creating a new code [if necessary],

(2) the 'input image' will activate the 'recognition' process, and

(3) 'input data' will be sent to the OODBS.

The first two processes have already been explained. Now we will explain the last process.

When the 'general manager' of the data model receives 'input data' [represented by its code], then it will automatically activate:

(1) the 'data model' with its related branches, and always the object types 'urban environment' and 'urban elements' (from there the connections go to 'microunits' and 'global units'),

(2) 'microunits' and the 'typology of microunits', which are crucial for the filtering process; they start to be active right after completion of the 'recognition' process, and

'global units' and the 'typology of urban elements', which are important for the synthesis process; they will stay in 'stand-by' mode until the filtering process is complete.

The active branches in the *data model* directly connect to all related objects from the selected 'image scene', which is partly indicated by the 'input' data. All active object types from the data model are 'preliminarily active' because some changes must be possible after the completion of the processes of 'filtering' and 'synthesis'. The difference cannot be big, but it could be important. The precise definition of active object types from the data model is very important for the Design Tool's efficiency. The data model, presented in Chapter 8, has an important role in 'taking data' from objects of an 'image scene'. The best means of explaining the general functioning of the presented process is through examples.

Example 1 is presented in Figure 9.7:

The user selects and prepares an 'input image' and 'input data'. The user wants to perform an analysis of the urban elements related to the 'node' and 'landmark' of a particular district, presented in the 'input image' in map form. The user confirms the 'input data' related to the 'input image'. The tool then starts processing. The 'input image' is put through the process of 'recognition'. The copy of the 'input image' is sent to the *Layer0*. The 'input data', represented by its code, activates the 'general manager', which then activates the following branch [object types] of the 'data model' according to the 'input data' and 'input image':

- 'urban environment' object type,
- 'urban elements' object type,
- 'district', 'node' and 'landmarks' object types, and
- all other object types under these three which belong to them.

The 'urban elements' object type has a connection to 'microunits' and 'global units', which are also automatically activated. 'Microunits' is connected with 'typology of microunits' and from it all related generic microunits will be active as soon as the 'recognition' process is finished. The rest is the filtering process, which is done by microunits. The filtering process was explained in Chapter 7. 'Global units' and the 'typology of urban elements' are also active, but only initially, until the filtering process is finished and then the process of synthesis starts.

As we see from the example, the user did not select any psychological activity of the image to analyze. For this reason the 'psychological preconditions' will not be active as one of the two main branches of the data model.

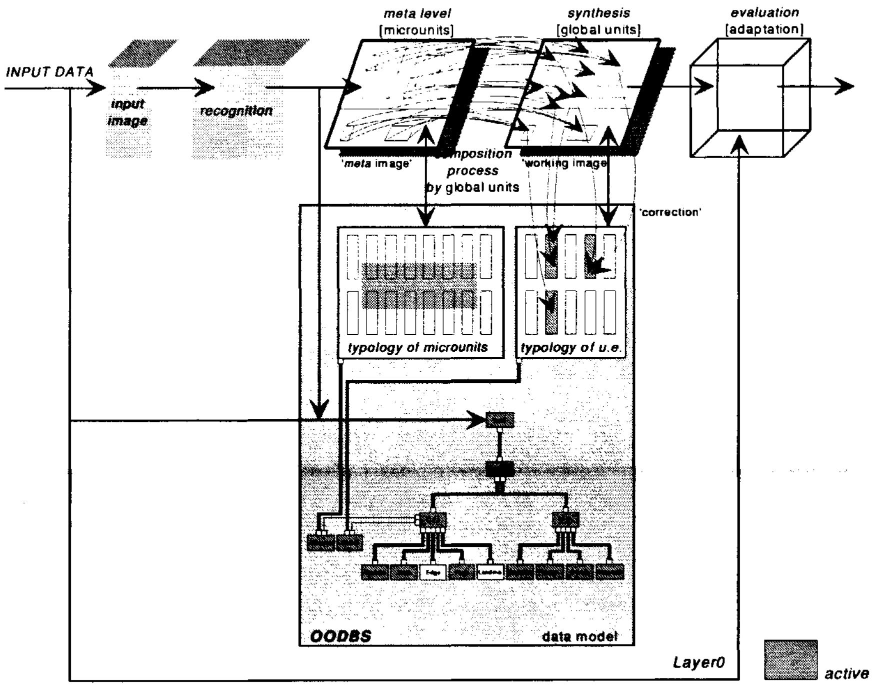
Synthesis is the fourth module of the *conceptual model* of the Design Tool. 'Synthesis' is also the third and last module in the 'computer cognition model'. The 'synthesis' module is located between the 'meta level' and 'evaluation' modules. The 'meta level' and 'synthesis' modules interact together and produce the final results of the 'computer cognition model' - the *working image*. The process of 'synthesis' was explained in Chapter 7. Some details, which will also be explained here, are presented in Figure 9.8.

When the 'filtering' process is complete and a temporary 'meta image' is formed, then the process of 'synthesis' starts. 'Synthesis' consists of four main processes:

(1) 'composition' is the process of forming global units by grouping several microunits that belong to one object from the 'meta image',
(2) 'synthesis' is the process of forming a 'working image' by unifying all objects from the 'meta image' defined by global units,
(3) 'correcting' is the process of checking the initially activated 'typology of urban elements' and all proper partitions of it, and
(4) 'feedback' is the process of correcting active branches [object types] of the data model.

After the filtering process, when microunits overlap the 'input image' and create a new image called the 'meta image', all microunits transform into large numbers of global units. Microunits that belong to an object form several global units that give the final shape of the object. All objects from

Figure 9.8
The 'synthesis' module and the process of composition by global units - *Example 2*



the 'meta image' form a new image called the 'working image'. Then each object from the 'working image' is checked by its global units through the 'typology of urban elements'. Checking is also a process of correcting active partitions of the 'topology of urban elements' as well as active branches of the data model. The circle of initial checking of 'input' data through activating branches of the 'data model' and 'microunits' and 'global units' is closed. The process of 'synthesis' is finished as well as the processing of the 'computer cognition model'. The new image called the 'working image' is ready for the user and further manipulation or analysis. The second example will show how the process of 'synthesis' specifically works.

Example 2 is shown in Figure 9.8.

The user selects an 'input image' in the form of a 2D map and fills in the 'input data'. The user wants to perform analyses of streets in a location in a district, by 'way-finding' and 'cognitive mapping'. The user confirms the 'input data' and the Design Tool starts processing. The process of 'recognition' of the image map is started and the original input map is sent to *Layer0*. The 'input data' represented by its code activates the 'general manager', which then activates the following branches of the 'data model':

- 'urban environment' object type,
- 'urban elements' and 'psychological preconditions' object types,
- 'district', 'path' and 'node', all object types under the 'urban elements'
- 'legibility', 'image', 'identity' and 'imageability', all object types under the 'psychological preconditions' object type,
- all other object types under these object types, down to the smallest elements of the data model.

The 'urban elements' object type is connected to 'microunits' and 'global units', which are connected to 'typology of microunits' and 'typology of urban elements' and they are all automatically activated. Then the processes related to 'synthesis' is started, and they are explained above. 'Feedback' checks the correctness of pre-selected branches of the data model. The data model will be checked according to the initial data related to the user's needs [in our case an analysis of streets by 'way-finding' and 'cognitive mapping']. Some of the object types are re-activated, others are activated and the final correction results in the final stage. This is the last step in this part of the Design Tool processing – the next one is 'evaluation'.

Evaluation is the next module in which the user, for the first time, directly interacts with the Design Tool. The process of 'evaluation' consists of two sub-processes:

(1) *derivation* - is the process of contriving adequate possibilities for design tasks which are taken through all disposable information from the database system, and

(2) *solution* - is the process of defining appropriate solutions related to the designer's tasks according to the process of derivation.

The processes of 'derivation' and 'solution' interact together simultaneously without any borders between them. One 'working image' can be *adapted* to the designer's tasks related to all available data from the OODBS. This adaptation of one image according to new tasks is a complex process in which almost all parts of the OODB system plays some role. In the 'evaluation' process some active role is played by:

(1) the *functional model*, which is defined by the 'input' data and by the action of the user,
(2) the *general OODB model*, some of those functions depend on selected functional model/s/, and
(3) the *data model*, with active branches depending on activities defined by the 'general OODB model', even on 'input' data.

Before explaining the 'evaluation' process in more detail, it is necessary to define the position of the user and the 'user's view'. After filling in 'input' data and giving a 'task' to the Design Tool, the user again becomes active in the 'evaluation' process. There the user can interact directly with the tool. The user's interaction and 'freedom of creativity' depend on the 'given task'. If the 'given task' is more open then the user can have more freedom in this interaction. If the 'given task' is strictly precise then the Design Tool will perform most of the process. The 'given task' is the fully filled-in 'input' data, which also presents requirements and questions that the user wants to be answered by the Design Tool. If user entered a precise task in the third window, e.g. 'cognitive mapping', then his or her freedom will be limited to the 'given task'. In any case, the user can, for example, improve the existing sets of criteria or information by adding additional ones, for example traffic density. The Design Tool is structured to be open for users' needs and in the 'evaluation' module gives possibilities for the user to improve the tool by adding new relevant information. The process of adding new information to the tool is through the 'add' menu, which is located under 'edit' on the *main menu*. This new additional information will be directly located and saved to the 'general OODB model', and to the 'external data' module via the 'general manager'.

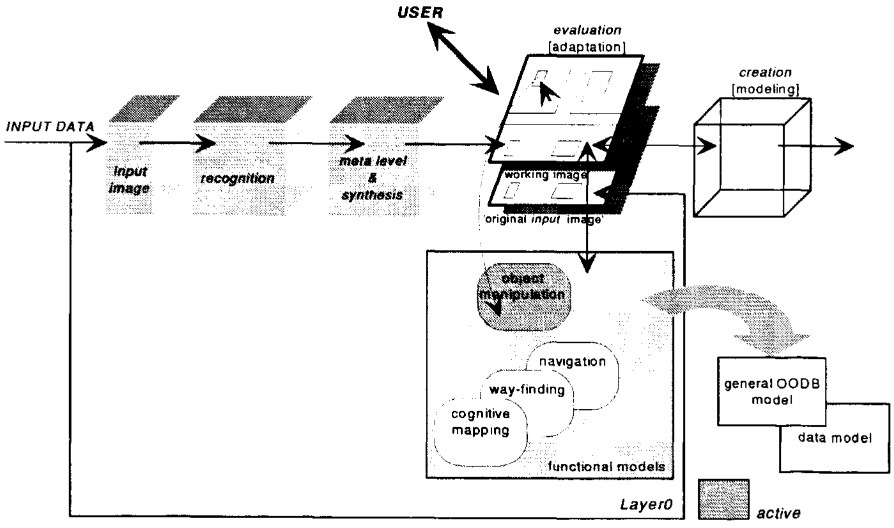
The 'user's view' is the functional organization of the screen on which the user works. The screen is subdivided into three main working parts:

(1) *main menu* with sets of pop-up icons which help the user to work effectively (most icons of the menu are universal such as file, edit, view, format, tools, etc.),
(2) *'working image'* is 'ready to use' 'input image' and
(3) *Layer0* is the original 'input image', which has a function to help the user to effectively compare all adaptation and changes.

The 'working image' and *Layer0* might be different in their formats, or the same. They can stay beside each other, or the 'working image' can partly or completely overlap with *Layer0*. It is also possible to put them together in 'layer mode' positions, when they can be saved with possibilities of different percentages of 'transparency mode'. It is also possible for a small number of images to be active at the same time, for example three 'working maps' and three original 'input maps'. This function is not dependent on the screen size. More about the 'user's view' will be explained through examples in this chapter.

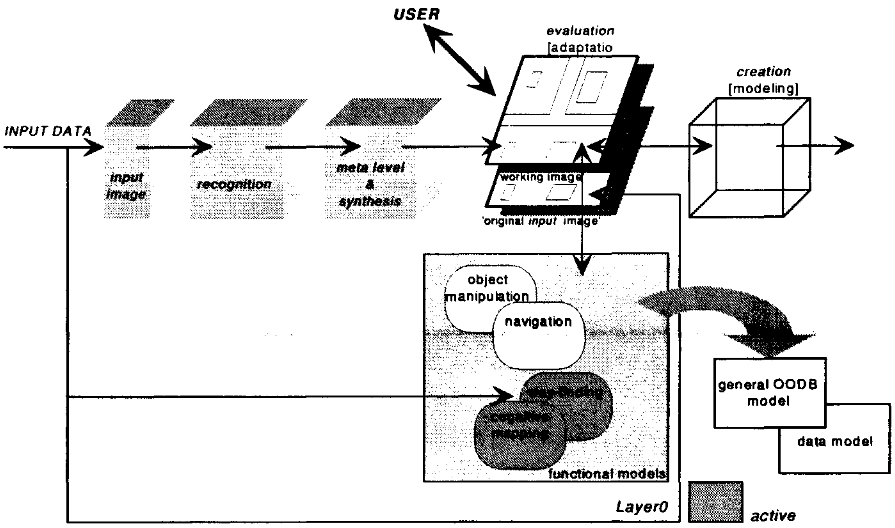
The *functional model* may or may not be defined by 'input' data. If it is not defined exactly then the 'functional model' will be selected when the user starts manipulation of the 'objects' from the 'working image'. Figure 9.9 illustrates this case. The user selects one object from the 'working image' and starts to redesign the shape of the object by simply 'playing' with it. After this, the user starts manipulating other objects from the 'working image'. In both cases the tool will automatically select adequate 'functional models' - 'object manipulation'. After that, the general manager will activate related parts of the OODB model and correct, if necessary, the already active data model. The selection and activation of proper 'functional model/s' is important for activating the proper 'menus' and some further functions.

Figure 9.9
The 'evaluation' module and selection of 'functional model'



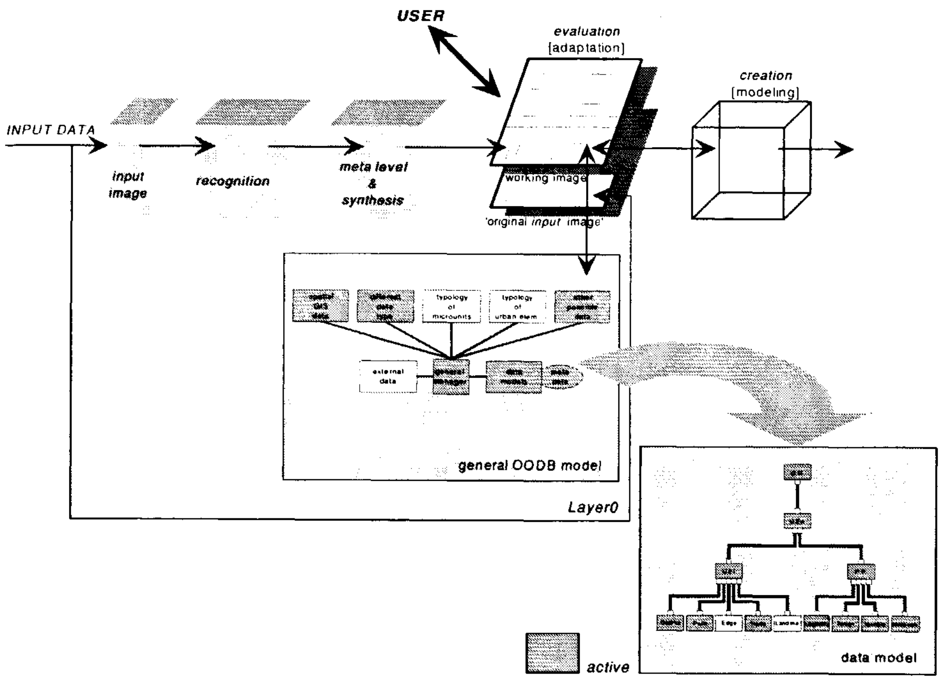
If the 'input' data defines a precise functional model, e.g. 'way-finding' and 'cognitive mapping' [see *example 2*], then the functional model will be selected directly from the 'input' data (see Figure 9.10).

Figure 9.10
The 'evaluation' module and selection of 'functional model'



In our case, the 'input' data directly activates 'way-finding' and 'cognitive mapping'. During the process of 'meta level', the 'general OODB model' and 'data model' are already activated. Figure 9.11 illustrates these activities. The 'general OODB model' presents all active parts related to the actual 'functional model'. In the case when 'object manipulation' is an active 'functional model', then 'external data' must be an active part in the 'general OODB model', if the user adds some new criteria e.g. traffic density. Some adequate branches of the 'data model' are also active, on the schema related to 'way-finding and 'cognitive mapping'.

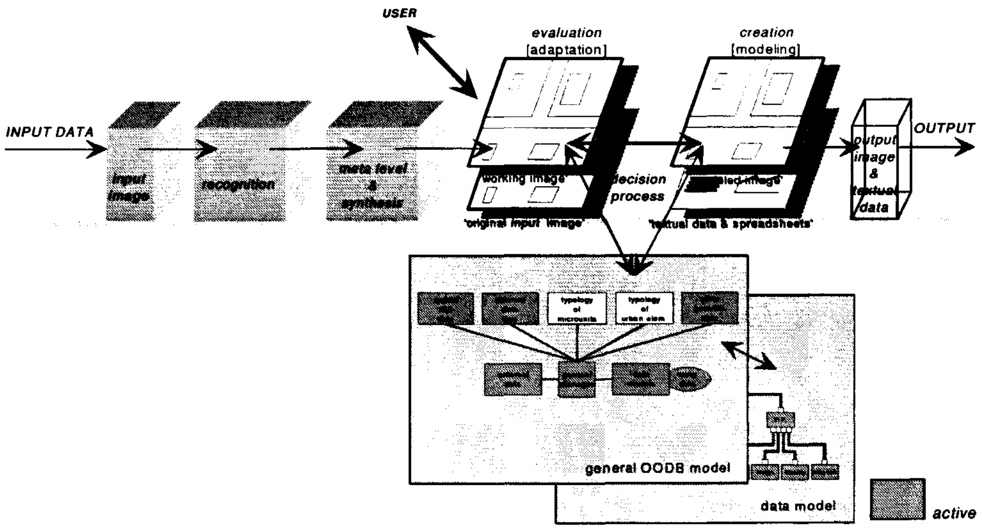
Figure 9.11
The 'evaluation' module and activities of the 'general OODB model' and 'data model'



Whenever the user changes some objects related to his or her tasks or adds some new information, both the 'general OODB model' and 'data model' will automatically execute all of them. All changes will be registered in the database system. All changes will be visually presented immediately in the 'working image'. The results will be visible and comparable with *Layer0*. If the Design Tool does some independent processing, mostly analyses, this will also be visually presented in the 'working image' and comparable with *Layer0*. In all cases, all changes will be visually presented in a 'working image' and executed through the database system.

Creation is the module located between the 'evaluation' and 'output' modules. The process of 'creation' directly interacts, constantly processing back and forth, with the 'evaluation module', until the final solution is decided upon. In the process of 'creation', all available data from the 'general OODB model' and 'data model' play an active role. An important role is also played by the user, whose interaction with the Design Tool should help in completing the process of decision making. The general relationship between the user, 'evaluation' module and 'creation' module is presented in Figure 9.12.

Figure 9.12
The 'creation' module and its interactions with the user, 'evaluation' module, 'general OODB model' and 'data model'



The process of 'creation' can be subdivided into two main processes related to 'problem targeting' and finishing the given task: the *decision* process and *execution* of decision. The *decision* process is an essential function of the 'creation' module as well as of the Design Tool. *Execution* of decision can be subdivided into:

(1) finishing the task, and
(2) data saving.

One of the requirements of the Design Tool is meant to help the user to reach decisions, which can be considered a 'decision support system' [DSS] as a part of the Design Tool. The goal of the research is not to develop the whole concept of the DSS system for the Design Tool, because developing a DSS is a very complex process and it can give rise to a new research, rather than indicate its functions. Here we will explain some of the main premises of the DSS related to the specificity of our Design Tool.

The more general specificity is that the Design Tool works with the urban environment, and accordingly the decision process is based on a spatial decision support system [SDSS]. The SDSS can be developed for use with a domain database that has a spatial dimension, or for situations where the solution space of a problem has a spatial dimension. The essential characteristics of the SDSS implemented in the Design Tool are the integration of a Geographical Information System [GIS] with spatial image and map analyses and display modules ['evaluation', 'creation' and 'output' modules]. The main components of the SDSS are:

(1) the general OODB model,
(2) the data model, and
(3) the user interface.

The first two components have already been explained. The *user interface* is a dialog manager that assists with all aspects of communication between the user and the hardware and software that comprise the *tool* (Power, 1996). The user interface is the part of the tool that the user can see. It represents all the mechanisms whereby information is inputted into the tool, the user's interactions with the tool and output from the tool. It includes all input screens through which the user requests data and models. It also includes all dialog screens through which the user 'communicates' with the tool during the process, and also includes all the screens through which the user obtains the results as output.

Each of these components gathers information in its specific way and these ways change in time. All components are used to couple them together and to make the best decision, which is termed *rational* [or *optimal*]. The properties of a rational decision are:

(1) a rational decision is one (or more) of a specified set of possible decisions (there may be more than one 'best' decision if they are equally good; if they are then a decision must be generated),
(2) a rational decision depends on the decision principles (criteria) integrated in the 'data model' or employed by the user ('new criteria'),

(3) the rational decision for a decision situation may differ among users and their 'interactions' with the tool,
(4) a rational decision is dependent on relevant data available to a tool and to a user, and
(5) a rational decision must be consistent with the user's preferences and beliefs.

The rational or optimal decision will be the first step in the completion of the Design Tool process.

Designing the decision-aiding system is based on several theoretical principles of decision theory and their implementation through decision analysis. Decision analysis is therefore defined as a technology that develops methods of assisting the process of assessing a choice problem. The underlying procedures of our Design Tool SDSS involve:

(1) extracting relevant information from the 'problem target' related to the goal in the 'input' module [user],
(2) selecting the clusters from the 'general OODB model' [tool],
(3) structuring the data from the 'data model' [tool],
(4) defining the alternatives [tool/user],
(5) using 'criteria' and if needed 'new criteria' implemented by the user [tool],
(6) evaluating all of them [tool],
(7) choosing or identifying various possible solution/s [tool/user], and
(8) resulting consequences (effects) [tool].

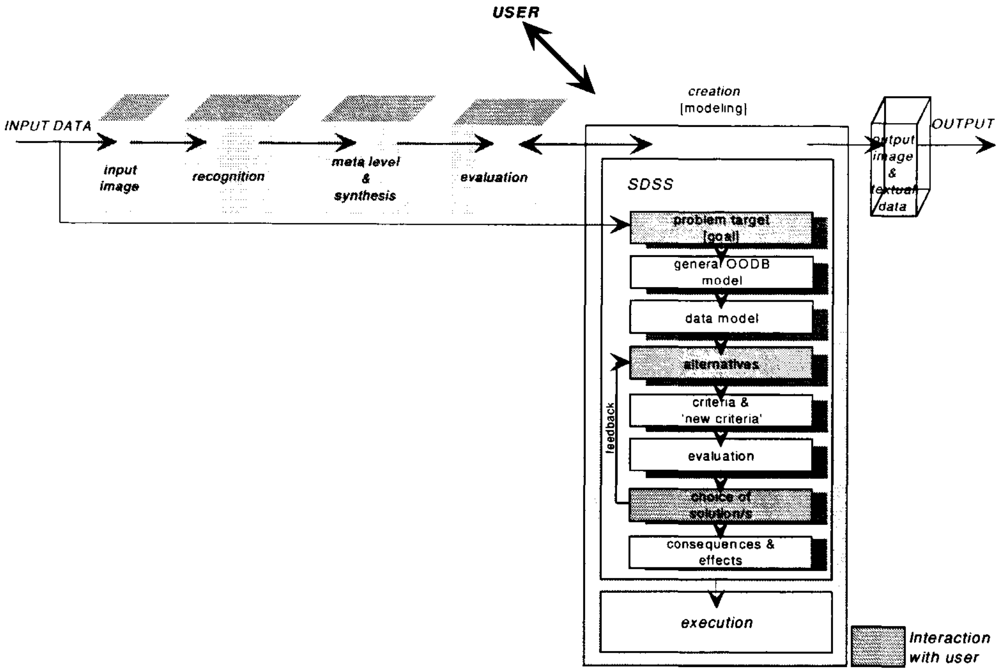
Figure 9.13 presents the concept of the SDSS procedure in the 'creation' module.

The interaction between the user and the tool is given in three procedural phases of the SDSS:

(1) *problem target* is the phase in which the user defines the 'input' data related to the goal which the Design Tool has to solve,
(2) *alternatives* is the phase where the user selects from presented alternatives one of them for further processing according to the 'problem target', and
(3) *choice of solutions* is the phase in which the user makes a definitive choice from proposed solutions that the tool has to process according to the task and all available criteria and data.

If the tool successfully finishes processing according to the user's choice and all other components, then the next phase will be started. If the finished process is not acceptable to the tool then the tool suggests that the user

Figure 9.13
The 'creation' module and the concept of the SDSS procedure

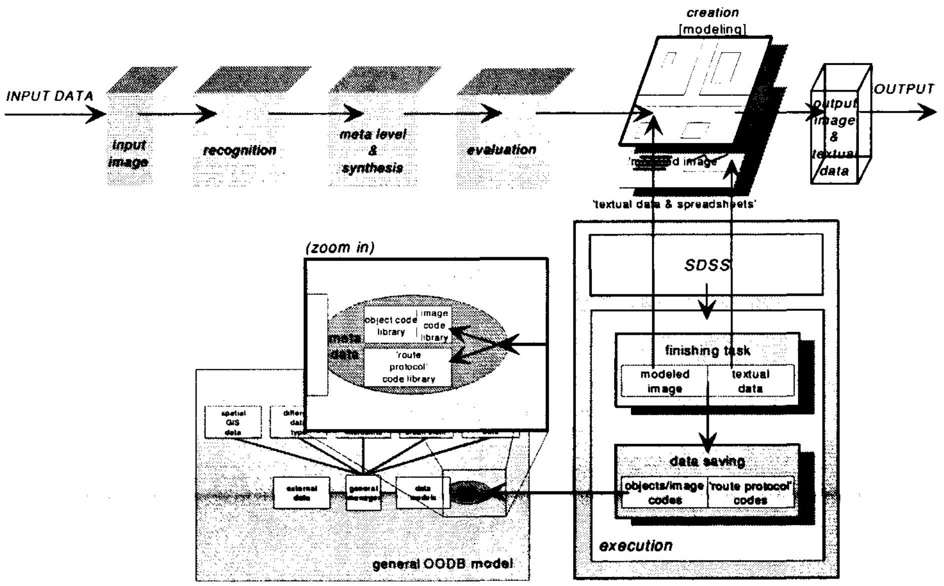


makes a new choice or selects some other alternative. This process will be active until:

(1) the tool creates an acceptable solution [acceptable for both the user and the tool],
(2) the tool does not have any acceptable alternative solution - then no new results will be presented as the output [in this case 'input image' will be the 'output image'], and then
(3) the tool suggests that the user changes the 'input' data and some other parameters [e.g. 'new criteria'] until acceptable (compromise) results appear as the completion of the process.

The next step in the 'creation' module is *execution*. Execution consists, as we mentioned, of 'finishing task' and 'data saving', as presented in Figure 9.14.

Figure 9.14
The 'creation' module and the concept of the 'execution' procedure



'Finishing task' is the finalization of modeling the new image called the 'modeled image' and the creation of 'textual data'. The user redesigns the 'working image' during the creative interaction process in the 'evaluation' module. For redesigning, the user uses a simple graphic tool, selected from the main menu. The user manipulates and changes the existing objects from the 'working image'. The 'modeled image', by contrast, is composed directly by the tool using the geometry of the objects. The composition of the 'modeled image' is a final creation based on an integration of knowledge of: the 'working image', the given task, sets of criteria and decision making. The 'modeled image' is an image always accepted by the user and tool. The 'modeled image' should be different in file format, media representation or image type to the 'input image'. A 'modeled image' can also be selected to be different in file format, media representation or image type to the 'input image'. The 'modeled image' always represents the results of the Design Tool processing and it is an 'output image'. Most of the time the 'output image' does not present enough information related to the given task, and for this reason the tool can present some additional, mostly 'textual data'. The 'textual data' presents all kinds of necessary information relating to different phases of analysis but always related only to the task [it cannot present any information related to the computational process]. The 'textual data' can present simple explanations or some qualitative or quantitative data, which are mostly the results of analyses related to the given task. These results could be presented textually, as a spreadsheet or in diagram form, depending on the type of information to be shown. If the user wants to compare heights of several buildings from an image scene, than as a result a table will appear with all heights for each of the buildings. Some other practical examples of 'modeled images' and 'textual data' will be presented in the next chapter.

'Data saving' is the last step in the 'creation' module (see also Figure 9.14). When the 'modeled image' is definitely defined, then all objects from the 'image scene' that are different (through changing) to the objects from the 'input image' will automatically receive new codes. For example, if the task of analysis is the comparison of the heights of buildings from an image scene, the result of analyses will be automatically saved and the same result can be used at any time later. Each changed or new object will receive a new code. All new codes of the objects are automatically saved in the 'code library'. The 'modeled image' also receives its own code, which is stored in the 'code library' under the 'meta data' cluster. The tool will also automatically save a 'route protocol' in the 'route protocol library' under the 'meta data' cluster. The 'route protocol' consists of numerous different protocol codes. The 'route protocol' represents the way some particular urban environment was analyzed [the process from 'input image' to 'modeled image']. Saving and retrieving a 'route protocol' means that the Design Tool is a 'self-learning' system. If the user wants to use the same 'modeled image' some other time, he or she has to type the code of this image in the 'input' data windows and then the 'modeled image' will automatically appear as a 'working image' in the 'evaluation' module [with the copy of the same image as *Layer0*]. All stored 'input images' represented by their codes are available to the user as thumbnail images retrieved from the 'code library' whenever the user needs them. This 'retrieval image' will also retrieve all related objects [via their codes] from which the image and all related data consist. Also, if the user wants to analyze a new image that has almost the same tasks and characteristics as some existing stored 'input' data, then the tool will automatically repeat the same routine as the existing one stored in the tool by retrieving the associated 'route protocol' from the 'route protocol library'. The tool becomes a 'self-learning system', which is a great advantage of Perspective-DB technology.

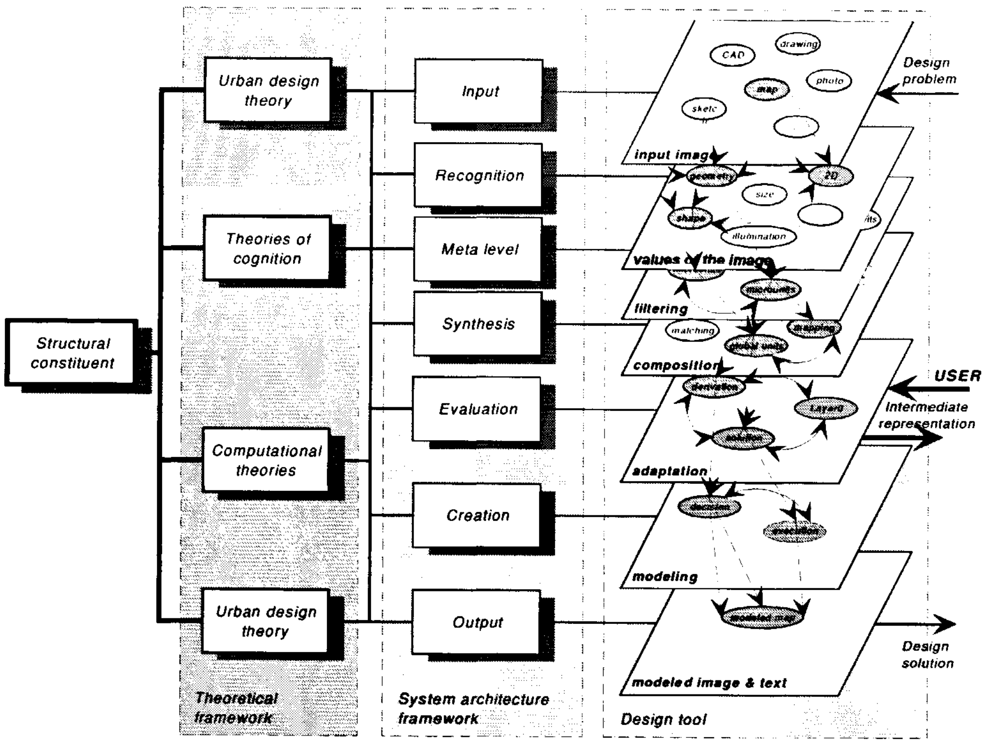
Output is the last module of the *conceptual model* of the Design Tool. The 'output module' presents the final results of the processing of the Design Tool e.g. an analysis of some particular urban environment. 'Output' is always an image, called the 'modeled image', and if necessary some additional text or spreadsheets are available (see also Figure 9.14). For example, the result of height analyses, presented in the form of a table, can be printed as the final result of these analyses. 'Output' could be answer/s to the 'input' question/s, but is always the result of analyses of the given task. 'Output' can be a print or prints, or a digital representation, for example a CAD model [static or dynamic]. 'Output' can also be a VR model, but it depends on the corresponding 'input' data. The results of Design Tool analyses can help urban planers and designers in their creative process. The results of the processing of the Design Tool can be implemented in such a way in urban design, e.g. an additional part of an actual urban design project.

Structural analysis of the Design Tool

The task of structural analysis is to make clear some relationships between the 'theoretical framework', the 'system framework' and the 'Design Tool', as shown in Figure 9.15. Each of these three important segments of the research arises from a previous one.

The structural constituent of the research consists of a more general 'theoretical framework'. As we explained earlier, the 'theoretical framework' is based on three main fields of related sciences: 'urban design theory', 'theories of cognition' and 'computational theories'. To each of these theories some very specific theories belong, which are important for defining the next segment of the research: the 'system framework'. 'Urban design theory' is focused on Lynch's theory of 'urban form' and his concept of 'cognitive mapping', which are explained in Chapters 3 and 6. 'Theories of cognition' consist of two selected and relevant theories related to computer cognition and vision. The first is a theory devised by Marr called 'computer vision' and the second is 'high-level vision' developed by Ullman. Both theories are explained in Chapter 4with their important parts that relate to our research. 'Computational theories' are based on wide fields of database theories, from which we selected the 'object-oriented database'. In Chapters 5 and 8 we

Figure 9.15
Structural decomposition of the research



explained the main characteristics of this theory and the chosen Perspective-DB technology. All the specific characteristics of the selected theories from the theoretical framework result in the main structure of the 'system framework'.

The 'system framework' is the logical product of the main functional characteristics of the specificity of the chosen theories. As we have already explained, the 'system framework' consists of several parts: *input, recognition, meta level, synthesis, evaluation, creation* and *output*. Each of the parts is one of the modules of the *conceptual model* of the Design Tool. All modules together form the *conceptual model* of the Design Tool, as previously explained (see Figure 9.2). The relationship between the 'system framework', i.e. the *conceptual model* and the Design Tool, is presented in the right-hand part of Figure 9.15. Each module of the *conceptual model* has an equivalent functional step in the structure of the Design Tool. For example, the 'input module' directly relates to the 'design problem' represented as 'input image'. In our example, the 'input image' is a *map*. The 'recognition' module directly relates to the 'process of recognition', which is adopted from Ullman, represented by 'shape recognition' and 'values of the image', such as *geometry*, *shape* or *2D* (in our brief example). The 'meta level' module is the process of 'filtering' by 'microunits'. The 'synthesis' module is also the equivalent to the next step in the Design Tool processing called 'composition'. 'Composition' consists of several integrated processes; in our example they are active 'global units' and 'mapping'. The next module of the *conceptual model* is 'evaluation', which in the equivalent process of the Design Tool structure is 'adaptation'. In 'adaptation' the user can directly interact with the tool. The Design Tool will present an 'intermediate representation' to the user with which he or she will directly interact. In our example the processes of interactivity will be *derivation, solution* and activation of *Layer0*. The next module of the *conceptual model* is 'creation', which has its equivalent structure in the Design Tool called 'modeling'. 'Modeling' consists of 'decision' and 'execution' processes, both active in our example. The last module is 'output', which is also the last structural step of the Design Tool processing. Output is the final 'design solution' for the input 'design problem' represented with a 'modeled image' and possible additional texts [and spreadsheets]. In our example, 'output' is a 'modeled map', a map which the tool together in interaction with the user created and modeled depending on the given tasks, criteria and all relevant and available data from the tool.

The functional steps in the Design Tool processing define its 'vertical structure'. The 'vertical structure' is conduced by hierarchical logic. In the example presented here, it was shown how a simple 'input map' becomes a more complex 'modeled map' through the different structural and functional steps in the Design Tool processing. The next chapter will make clearer each structural step through different examples.

The functional model of the Design Tool

The role that the object-oriented database [OODB] system plays inside the *conceptual model* of the Design Tool was not previously explained. The main task of the 'functional model' is to show the indisputable importance that the OODB system has for the comprehensive structure of the *conceptual model* of the Design Tool. Almost all modules of the *conceptual model* have some specific relationships to the OODB system, which will be explained here with their main characteristics.

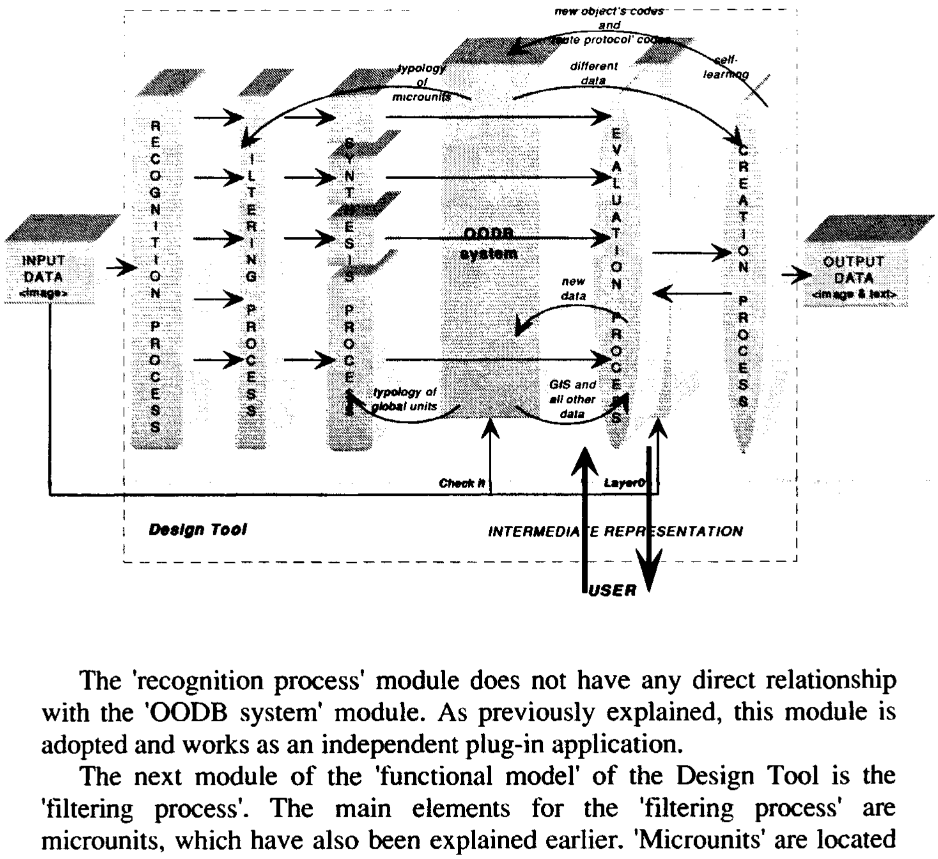
Structure of the Design Tool

Generally the structure of the 'functional model' is based on the *conceptual model* structure, which was developed and explained before. All modules of the *conceptual model* also have the same structural position in the 'functional model'. The only new module in the structure of the 'functional model' is the 'OODB system' (see Figure 9.16). The position of the 'OODB system' module in the presented schema is only 'symbolically' *behind* the other modules. The reason for this is that the 'OODB system' module assists other modules with different aids and it is not in the linear functional order like the other modules from the *conceptual model.* The OODB system assists all modules from the *background.*

The structure of the 'functional model' consists of the following modules: *input data, recognition process, filtering process, synthesis process, evaluation process* (with *Layer0*), *creation process, output data* and *OODB system.*

The relationship between the 'input data' module and the 'OODB system' module only exists during the checking of the codes of the 'input image'. The checking was explained earlier when the structural part of the *conceptual model* related to the 'input data' module was presented. Each item of input data has to be checked through the all storage codes from the 'code library' which is located in the 'meta data' cluster of the 'general OODB model'. To avoid possible mistakes it is necessary to say that the 'general OODB model' from the structure of the 'database model' (see Figure 8.4) is the same as the 'OODB system' module in the 'functional model'.

Figure 9.16
The 'functional model' of the *Design Tool*



The 'recognition process' module does not have any direct relationship with the 'OODB system' module. As previously explained, this module is adopted and works as an independent plug-in application.

The next module of the 'functional model' of the Design Tool is the 'filtering process'. The main elements for the 'filtering process' are microunits, which have also been explained earlier. 'Microunits' are located in the generic 'typology of microunits', which are stationed in the 'OODB system' module. When the 'microunits' are activated the whole 'data model' from the 'OODB system' module is also activated, or at least potentially activated.

The 'synthesis process' is the next module in the structure of the 'functional model'. The 'synthesis process' is automatically activated after the completion of the 'filtering process', also explained earlier. The 'OODB system' module helps the 'synthesis process' through 'global units' which are located in the generic 'typology of urban elements'. In the 'synthesis process', a 'working image' is composed of different segments of corresponding objects from the image scene. In the schema in Figure 9.16, the 'synthesis process' is symbolically presented in a few sub-clusters.

The module in which the user interacts directly with the Design Tool is called the 'evaluation process' module. The 'evaluation process' module consists of a 'working image' on which the user directly works, and *Layer0*, the original 'input image'. The 'OODB system' module helps the 'evaluation process' with existing stored data e.g. GIS data or different sets of criteria. The usage of data is interactive, and depends on each simple change to the image performed by the user. For each object from an image scene composing a 'working image', all related data is available through a simple point-and-click capability. The input of 'new criteria' or some other 'new data' can also be performed here.

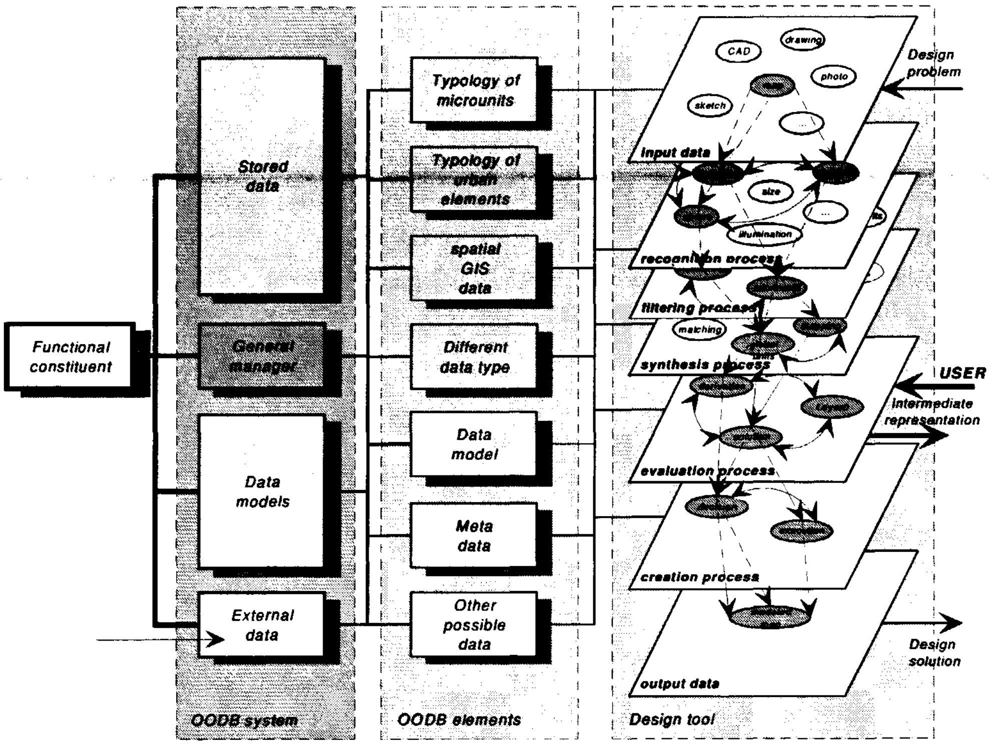
The 'evaluation process' interacts directly with the 'creation process' by exchanging all visual and other data back and forth, until the final solution is decided upon. This process was explained earlier. The 'creation process' module is also aided by the 'OODB system' module. All data needed for making decisions and finishing tasks are available. After finishing the task, some new possible data belonging to the 'modeled image', e.g. new codes, will be automatically saved in the 'code library' before any output will be made. This 'new' set of data also contains the 'route protocols' for all analyses performed and they can also be saved in the 'meta data' cluster. The process of saving 'route protocols' is called a self-learning process, which was briefly explained earlier. The saving of 'new' data and 'route protocols' is done automatically after the tasks are finished. The 'new' data will be saved in the 'code library' which is a part of the 'meta data' cluster. The 'route protocols' can also be saved in the 'meta data' cluster directly under the 'route protocol library' (see also Figure 9.14).

The last module in the 'functional model' structure is the 'output data' module. The 'output data' module presents the results of the Design Tool processing, as explained earlier. The 'output data' module does not have any connection with the 'OODB system' module, it is only a representation of the completion of tasks through the 'modeled image', as we explained in example of printing the results of the height analyses of buildings from an image scene.

Functional analysis of the Design Tool

Functional analyses have the task of making clear the main relationships between the 'OODB system' module and the Design Tool, by giving their *functional decomposition* as presented in Figure. 9.17. The structure of the *functional decomposition* consists of the 'OODB system' and 'OODB elements'.

Figure 9.17
Functional decomposition of the OODB system in relation to the *Design*



The 'OODB system' is subdivided into the following functional parts, which are decomposed into 'OODB elements':

(1) the *general manager* is a software manager that regulates all relations between the different parts of 'OODB system', 'OODB elements' and the Design Tool,

(1) *stored data* contains different 'OODB elements' such as the 'typology of microunits', the 'typology of urban elements', 'spatial GIS data' and 'different data types'.

(2) *data models* consists of 'data model' and 'meta data', and

(3) *external data* has its equivalent in the 'OODB elements' as the part called 'other possible data'.

More details about elements presented by the 'general OODB model' are given in the previous chapter (see also Figure 8.3).

The relations of each element of the decomposed 'OODB system' to different Design Tool processes will be explained here in a brief example. Each new item of 'input data', as we explained earlier, must be checked to see whether it already exists in the tool by checking the codes through the 'meta data' cluster and the 'code library' partition. The 'recognition process' is an autonomous, independent process that works as a plug-in application and it has no direct connection to any of the OODB elements. The 'typology of microunits' is a generic typology that helps in the 'filtering process'. The functional role of the 'typology of microunits' was also explained in Chapter 8.

The role of 'global units' and their generic 'typology of urban elements' was also explained earlier. They provide a direct aid to the 'synthesis process' where a 'meta image' becomes a 'working image'; in our example a 'working map'. In the 'evaluation process', different data elements play important roles, such as 'spatial GIS data', 'different data type', 'possible external or other data' and 'data model'. In this phase of the Design Tool processing the user can directly interact in changing or implementing new things in the tool. Each new implemented data, or changed existing data, will have an immediate reaction in the OODB system, such as in 'data model', 'different data type', 'other possible data', 'spatial GIS data' and in 'meta data'. The 'evaluation process' is a complex interactive process which allows the open structure of the Design Tool. The 'evaluation process' and the next phase, the 'creation process', interact together in back-and-forth functional steps until the final decision of the new solution is reached. All the time during these interactive processes all available and related data elements are in operation. When the 'new design solution' is decided on, then the new 'output data' can be automatically saved by its codes into partitions inside the 'meta data' cluster. We explained this process earlier in this chapter. The 'output data' is only the executive presentation of the final results of the Design Tool processing and has no contact with any OODB element. Some other aspects of the 'output data' were also explained previously.

Conclusion

Three things were explained in this chapter: frameworks of the research, and the conceptual and functional models of the Design Tool. Structural and functional analyses of both the conceptual and functional models were also given.

The theoretical background of the research had a logical influence on the 'theoretical framework', which is composed of 'urban design theory', 'cognitive theories' and 'computational theories'. The specificity, mostly functional, of these theories influenced the development of the 'system framework'. The 'system framework' consists of several subframes: *recognition, filtering, synthesis, evaluation* and *creation*, which defined the main operations of the tool.

The *conceptual model* described the logic and structure of the Design Tool. The *conceptual model* also describes the structure of interactions and functions of different elements of the model. The main elements are: *input, recognition, meta level, synthesis, evaluation, creation* and *output.* All elements and their internal organization were explained. We subsequently gave 'structural analyses' of the *conceptual model* through its decomposition. Some brief examples were also presented.

The 'functional model' structurally followed the *conceptual model* elements by implementing a new element: the 'OODB system'. The 'functional model' explained the role and functional position and relations of the object-oriented database system [OODB] and the Design Tool. Each step of their interaction and functional specificity was briefly explained by example in 'functional analyses'.

This chapter, together with elements developed from the previous chapters 6, 7 and 8, presents the *conceptual model* of the Design Tool. The *conceptual model* and *functional model* form an important platform from which a realistic and useful Design Tool will be developed for design practice. The next phase in developing the Design Tool will be programming, which is not a task of this research.

In the next chapter we will perform tests and elaborate our Design Tool as well as the research. First we will introduce the testing model and criteria on which we will perform the simulation of the test case objects. Then we will simulate the four characteristic test case objects, all related to cognitive mapping, way-finding and navigation, according to the main approach of this research. The results of the testing will be presented as an analysis related to the proposed test criteria.

Part four - Evaluation

CHAPTER 10

TESTING AND EVALUATION OF THE DESIGN TOOL

***Task of the chapter**: in order to present completeness and consistency of the complex structure of the 'conceptual model' of the Design Tool, a test must be performed by simulating different objects of the cases, based on recent testing theories and a method newly developed especially for this case.*

Introduction

Over the previous chapters we developed the *conceptual model* of the Design Tool. The *conceptual model* consists of several modules: *input, recognition, meta level, synthesis, evaluation, creation* and *output*. The *conceptual model* has become complex in structure and the main question that should be answered is: is the proposed *conceptual model* complete and consistent as a Design Tool? Completeness and consistency of the tool are widely used indicators that imply user satisfaction (Hamel, 1988). To answer this question, it is necessary to test the *conceptual model* of the

Design Tool using appropriate performance criteria. The performance criteria will be implemented in a new testing model proposed for this research.

The testing of the *conceptual model* of the Design Tool will be a simulation of case studies. Simulation of the *conceptual model* should be able to reconstruct the facts of functionality in practice by applying a set of review procedures, such as image recognition, filtering or analysis of the urban environment. The case study simulation will examine the *conceptual model* of the Design Tool. The test cases will be used heuristically to present the possibilities of use of the Design Tool that will be drawn from different professional tasks.

To proceed with the test cases, it is necessary to present the method by which the *conceptual model* will be tested. First, the recent testing theory chosen and the case study methods will be explained, and then we will look at the testing model and how this model will be applied to the cases and which aspects will be tested. Different objects of the cases will be tested by applying the general approach of the research, such as cognitive mapping, way-finding and navigation. The testing of the *conceptual model* of the Design Tool will be finalized with an elaboration of the results.

The evaluation of the *conceptual model* of the Design Tool will be presented through a study of the *conceptual model* and its applicability. At the end of the chapter, indications for further research developments and a conclusion will be given.

Tool testing theory and methodology

The scope of this research is to develop a *conceptual model* of a Design Tool that can be a software tool. Evaluation of the Design Tool [software application] is essential for making improvements and generating standard practices for future projects, but it is probably the most neglected activity of the information system cycle (Ahituv, Even-Tsur & Sadan, 1986). One reason that it is neglected is because of the high cost of additional equipment and materials. We too do not have a prototype of the Design Tool, and thus it is not possible to perform any experimental testing on the tool. Instead, a simulation of the test cases will be made, which will execute the test results based on *acceptance criteria*. The *acceptance criteria* are a synthesis of the *usability method* and *adequacy criteria*.

The usability method

According to Nielsen (1994), the *usability method* is carrying out 'experiments' to find out specific information about a design. Usability is defined from the designer's perspective (i.e. what designers need to do to ensure that a usable system and services are developed). Design guidance is offered to ensure that the appropriate enabling state exists for each of the user's goal tasks. A goal task is what the user wants to do, and an 'enabling task' is what the user must do to create a state that enables the goal task to be performed.

In order to specify or measure usability it is necessary to identify the goals and decompose *effectiveness*, *efficiency* and *satisfaction* and the components of the context of use into sub-components with measurable and variable attributes (Rubin & Jeffrey, 1994).

Effectiveness is the accuracy and completeness, by which specified users can achieve specified goals in particular environments.

Efficiency is the resources expended in relation to the accuracy and completeness of goals achieved.

Satisfaction is the comfort and acceptability of the work system to its users affected by its use.

The standard states that when specifying or measuring usability, the following information is needed:

(1) a description of intended goals,

(2) a description of the components of the context of use including users, tasks, equipment and environment, and

(3) target or actual values of effectiveness, efficiency and satisfaction for the intended context.

Measures of usability are assumed to be of two kinds:

(1) *performance measures*, which are objective measures or observations of user behavior and are focused on task performance (i.e. how well the user can achieve a specific task), and

(2) *attitude measures*, which are subjective measures or observations of the users' opinion of working with the system (i.e. how much they like to use the system).

These two measures are highly dependent on the context, task and type of users concerned. Performance and attitude measures are also complementary in the sense that both contribute to the complete evaluation of the usability of a human/machine system.

This multifaceted approach to testing can also be measured empirically and subjectively. As Adelman (1991) suggested, empirical methods are used for testing performance criteria and subjective methods are used for obtaining the users' opinion about an interface's strengths and weaknesses.

The usability method refers to a specific kind of task, user and environment and in this sense cannot be generalized over different kinds of tasks, users and environmental conditions.

The *adequacy criteria*, standardized engineering criteria, were fulfilled in the test case methodology. These criteria are:

(1) *validity* is a measure of the degree to which the tool achieves its operations and functions based on the relationship between the tool and the *conceptual model* on which it has been developed;

(2) *effectiveness* is the degree to which the tool should be able to support its user in achieving their professional goals, and concerns the relationship between the tool and practice,

(3) *efficiency* indicates that the tool should be efficient if fewer resources are used than would be required without its use,

(4) *reliability* is a measure of the degree to which the users of the tool are able to use and operate the tool in a wide variety of conditions, and

(5) *robustness* is the range of variations of elements and relations over which the tool should be flexible and acceptable to a certain degree.

The taxonomy of the five basic criteria of adequacy was developed for the task of general engineering research.

In the case of our research these *adequacy criteria* will be used as part of the testing model. According to the usability method, we will call these criteria in our test model the criteria of performance usability. Our test model will be based on the interaction of four main parts: *goals*, *test simulation* of the *conceptual model* of the Design Tool with different objects of the cases [all from urban practice], *criteria of acceptance* and *context of use*, where some testing conditions will be defined, such as user, equipment or environment. Later in this chapter we will propose the test model with more details.

Case studies methodology

The case study method is based on the assumption that an in-depth study of an example of a particular phenomenon can, through interpretation, provide a valuable insight into the nature of the chosen phenomenon. It is generally seen as an alternative to broad statistical or survey studies. Case studies, as they rely on interpretation rather than quantification, fall under the rubric of qualitative methods.

In performing a case study, the choice of case is always crucial. The case need not always be 'representative', although the generalizability of the findings of a case study are important for an exploratory investigation that could validate or eliminate a theory or a general conceptual model.

The application of the case study method to architectural research is relatively recent, but there is now a fair number of case studies appearing in the literature.

This research started with a theory based on a review of the various literature related to the subject under investigation, i.e. design theory, cognition theory and computational theories. These theories have to be validated through tool testing by simulating the specific case objects, procedures and phenomena. Simulating is substituting for, or imitating, reality. Because the Design Tool has not yet been built, simulation can describe the major features of the tool, its operational characteristics and the like. Computer "simulating is substituting for, or imitating, reality. If a system has been built, simulation provides it with an artificial environment for the purposes of testing and evaluation." (Rechtin, 1991 p. 63). According to Godelier (1982), in relation to our test method, a simulation can reconstruct the fact of functionality, give the meaning within the scope of the theoretical framework, and can use a set of review procedures to provide an analysis of the method itself.

This deductive approach is traditionally used to corroborate any scientific theory. A theory must be constructed before it can be validated. All theories are initially based on a particular case or object. The in-depth study of this case will elicit one or more theories (in our circumstances) that could be validated by other cases. This process will assess their general applicability. The general applicability of their explanatory value must be assessed by validating these empirical elements. Direct experience, as understood through various empirical elements, will be broken down into objects that could reveal the properties of the spatial relationships that make up these experiences. By *object of study* we mean "that which may be used to construct theoretical models [in the case of our research it is the *conceptual model* of the Design Tool], which can be manipulated according to explicit rules and which can be evaluated using specifically defined and codified tests" (Granger, 1989 p. 32). This process involves consolidating collected empirical materials into an object of study that will reveal the properties involved in the spatial relationship that constitutes the direct experiences of spatial knowledge.

The case study appears to serve this task. As it has been traditionally defined, it proves to be a descriptive study, par excellence and in-depth. The case study serves as "the most complete and detailed sort of presentation of the subject under investigation, by giving special attention to totalizing in the observation, reconstruction and analysis of the objects under study" (Zonabend, 1992 p. 7). A case study is, by definition, an in-depth study of a *particular* case. Studying other cases makes it possible to moderate not only the limits, but also the failings, as a comparison between cases puts the first study into perspective. The methodological virtues or qualities of a selected case make it 'a well-constructed single case'. Singularities of case are the skeletons of phenomena. In our test simulation we will examine four different objects of cases, all based on urban practice.

The scope of the case study is based on such choices and tactics, the methodological qualities of which may be understood in action, through the description provided of the object under study. The movement from local to global is determined by identifying singularities and understood in the sense used in mathematics and scientific epistemology. The case under investigation, through the methodological qualities assigned to it pursuant to the explanation of the recommended tactics and selections, becomes a sort of experimental prototype, to make an analogy with laboratory tests in the exact or natural sciences. This 'experimental prototype' is a computer based tool [in the case of our research it is the *conceptual model* of the Design Tool], which permits an understanding of the functions of it and an explanation of its properties, which, on such a scale, become evident. In accordance with this definition, an experimental prototype makes it possible to perceive the singularity of the object of study and ensures the transformation from local to global necessary for its explanation. Singularity is thus characterized as a concentration of the global in the local. Singularity is not perceived as a particular feature of a fact, an element, or a thing. It is seen, rather, as characterizing a fact, an element or a thing. In our test cases the singularity will also be presented with the different tasks of the analysis of an urban environment, such as cognitive mapping, way-finding or navigation.

The method involved in discovering singularities and in making the transformation from the local to the global may be summarized in three words: *describing*, *understanding* and *explaining*. *Describing* something well requires an *understanding* of it. Such a description should attempt to *explain*. Description should be understood to be the illustration of a whole and its sectioning into parts. Explaining means inserting this system into a broader one on which its genesis, stability and decline will ultimately depend.
This method [theory] essentially presumes the movement involved in the dominance of qualitative knowledge, and which uses singularities to detect the global within them and thereby reconstruct them (Granger, 1988).

As we see, the case study has proven to be in complete harmony with the three key words that characterize any qualitative method: describing, understanding and explaining. Such a study is best able to describe and understand the case under investigation. This study is considered to be a superior method of description and the choices and tactics that define it also precisely define the process of transformation from local to global. This ensures the general applicability of the explanation.

The generality of a case study is relative to the methodological value of the test process involved in the experiment itself. The methodological value of the experimental devices in the case study is essentially based on:

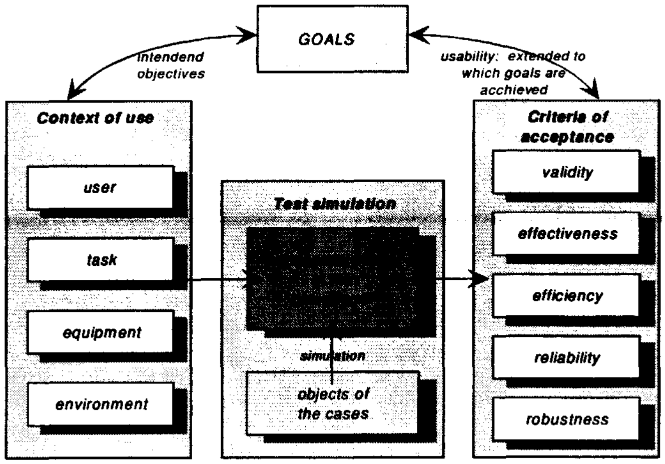
(1) the quality of strategies selected in defining the object of study and in the selection of the spatial unit (case objects) that makes up the ideal vantage point from which to understand it, and
(2) the methodological rigor displayed in the description of this subject in the form of a spatial analysis that can be understood in action.

This analysis must be properly reproduced to test its generality through other cases selected on the basis of the same object of study and which incorporate the same strategic qualities, so that it may be understood. More examples that can demonstrate the theoretical background of the case study methodology presented here will be examined later in the chapter, when we will explain the steps of test case simulation. The entire global theoretical meaning, explained through different specific cases, will be presented in this section.

The testing model for the *conceptual model* of the Design Tool

The new testing model, developed for testing the *conceptual model* of the Design Tool is presented in Figure 10.1.

Figure 10.1
The testing model for the *conceptual model* of the *Design Tool*
[based on the ISO9241 standard]



The testing model is based on the ISO 9241 standard, with some suitable improvements made. Generally, the ISO 9241 standard describes the ergonomic requirements for office work with visual display terminals. This standard defines how to specify and measure the usability of products and defines the factors that have an effect on usability. The improvements to the ISO 9241 standard are within the domain of 'adequacy criteria' and the simulation process of the objects of the cases. Here we will explain all elements pertaining to the testing model.

The closed chain of the testing model consists of several components and relationships between them: *goals, context of use, conceptual model with the objects of the cases* and *criteria of acceptance*.

The usability *goals* are a description of the intended goals. Goals are things desired and states that should be achieved for the system or service to be judged as usable. The goals should be particularized in measurable criteria, either absolute (e.g. the minimum level the system or service should meet) or relative (in comparison with previous systems or prototypes).

The *context of use* includes the following factors:

(1) *user* is the description of the target group of the testing. It is essential to perform sampling of the selection of users who will participate in the testing and evaluation process. Characteristics of the users need to be described. This can include knowledge, skill, experience, education, training, physical attributes and motor and sensory capabilities.

(2) *tasks* are usually many and very varied. Appropriate sampling or identification of the critical tasks for testing and evaluation are the basis for further steps in the testing and evaluation process. Tasks are activities undertaken to achieve a goal. The characteristics of a task, which may influence usability, should be described. Any description of the activities and steps involved in performing the task should be related to the goals that are to be achieved.

(3) *equipment* is the description of hardware, software and materials, and

(4) *environment* is the relevant characteristics of the physical and social environment in which the testing and evaluation have to be placed, and these need to be described. There are several attributes of the wider technical environment such as ambient environment or cultural environment.

In our case, four factors of context of use – *user, tasks, equipment* and *environment* – will also be simulated, or only partly. The first two factors will be directly involved in the test case simulation. The last two will be less integrated in testing, because *equipment*, for example, cannot make sense in our testing. The *user* can be an urban designer or any other professional who is dealing with the urban environment. The simulated user can be one of them. *Tasks* in the case of our test cases will be related to professional needs, such as shaping or re-shaping the urban environment, cognitive mapping, way-finding, navigation, and some other tasks such as distance measuring or the analysis of paths. All test cases, depending on the context of use factors, will be simulated and explored later in the chapter.

The *conceptual model* of the Design Tool will simulate the *objects of the cases*. The *conceptual model* of the Design Tool consists of seven modules, which were described earlier. More about objects of the cases and simulation of the Design Tool will be presented in the next section of the chapter.

The *criteria of acceptance* include *validity, effectiveness, efficiency, reliability* and *robustness. Effectiveness* and *efficiency* are the two criteria that are used in both concepts (*usability method* and *adequacy criteria*). We adopted the five elements of the *adequacy criteria,* instead of the three elements of the *usability measures* i.e. *effectiveness, efficiency* and *satisfaction.* The reason for this choice is that we do not need *satisfaction* criteria, because if the test and evaluation are successful then the satisfaction will be *per se* integrated.

A test case simulation

In order to perform the test of the *conceptual model* of the Design Tool, a new method is applied, shown in Figure 10.1. With this test case simulation we want to find out two things:

(1) whether the *testing model* is adequate for the testing and evaluation of the *conceptual model* of the Design Tool, and

(2) whether the *conceptual model* of the Design Tool is complete and consistent.

At the same time we will test the *testing model* and the completeness and consistency of the *conceptual model* of the Design Tool.

Goals of the test case

The usability *goals,* as explained earlier, are a description of the intended goals, which can be called the *usability heuristics.* The goals of our test case are based on several *usability heuristics* developed (Molich & Nielsen, 1990; Nielsen & Molich, 1990) in order to check:

(1) *visibility of system status* - the system should always keep the user informed about what is going on through appropriate feedback;
(2) *match between system and the real world* - the system should follow real-world conventions with words, phrases, symbols and concepts, making information appear in a natural and logical order familiar to the user, rather than system-oriented terms;
(3) *user control and freedom* - the user needs to have full control and freedom in using a system supported with certain functions (e.g. *undo* and *redo*);
(4) *consistency and standard* - the user should not have to wonder whether different words, situations or actions mean the same thing; the system must follow platform conventions;
(5) *error prevention* - even better than good error messages is a careful design which prevents a problem from occurring in the first place;
(6) *recognition rather than recall* - all object, actions and options must be visible and the user should not have to remember information from one part of the dialog to another; instructions for using the system should be visible or easily retrievable whenever appropriate;
(7) *flexibility and efficiency of use* - the system allows the user to speed up the interaction for expert users so that the system can cater for both inexperienced and experienced users and to tailor frequent actions;
(8) *aesthetics and minimalist design* - the system should not contain information which is irrelevant or rarely need; every superfluous unit of information in the dialog competes with the relevant units of information and diminishes their relative visibility;
(9) *helping the user to recognize, diagnose and recover from errors* - the system should express error messages in plain language, not in codes, precisely indicating the problem and constructively suggesting a solution; and
(10) *help and documentation* - the system should be easy to search, focused on the user's task, and should list concrete steps to be carried out and not be overly large; even though it is better to be used without documentation, it may be necessary to provide help and documentation.

According to their relevancy, these ten *goals* will be fully integrated in the development of the Design Tool. In our test simulation, all of these goals will be performed in a way that will help the user to interact with the Design Tool via intuitive interfaces. These goals should also be a set of requirements for developing [computer application programming] the Design Tool. Most of these goals will be directly linked to our test case simulation as examples of them.

Context of use of testing

The user is a group of professionals who are dealing with urban planning and designing. They might be architects, urbanists, landscape architects, transport engineers, geodesic engineers, geographers or any other professionals working with a spatial built up environment.

The task is executive testing by using the *testing model*, which was previously developed, by simulating the *conceptual model* of the Design Tool. Testing is executed through different types of objects of the cases, all related to object manipulations, cognitive mapping, way-finding and analysis of a particular urban environment.

The equipment necessary for testing is: a powerful PC computer with a large memory, a scanner and a printer. There should also be a digital camera as an additional piece of equipment. The computer must be linked via an Intranet or the Internet to some available GIS data or to some other specific database directly related to spatial data, e.g. cadastre, traffic, water, etc. The computer could be also filled with internal data prepared for testing a particular case.

The environment is the normal architectural or urban planning office, which has the all necessary facilities and conditions for testing. The testing environment must be relaxed without any physical or psychological pressure, such as noise, interruption or time pressure.

Test simulations

Case study selection and performance

A researcher chooses a field because he specifically intends to study a special subject, and this field would appear to be such an ideal place to observe it that the definition of the object is often confused with that of the field, or of the case itself. It is appropriate to distinguish between the object of study and the selected case study for the tasks of observation. The researcher should always be guided to generate the problem or phenomenon under consideration by the study and in defining the object of the study. Certainly, the researcher's subjectivity will intervene, and must intervene, to produce a definition of the object. As Zonabend (1992, p. 32) points out, objectivity and subjectivity cannot be contrasted: "We must be aware that the most rigorous objectivity is only possible through the most intrepid subjectivity".

The methodological strategy results from a theory, an *initial theory* to be more precise, the formulation of which is revealed by this strategy. Initial theory means the initial idea that a researcher had of the perceived spatial issue or phenomenon. This initial idea is mainly presented in the definition of the object and this definition must have been determined within the theoretical frameworks, in our case of urban theory, cognition and computation. More precisely, in our research the initial theory is derived from the concept of cognitive mapping of the urban environment, which is the main phenomenon of the research.

The selected case is not representative because of the observed frequency at which a spatial issue or phenomenon occurs. It is representative in terms of an initial urban theory, which presents it as the selected observation point from an object of study. An analysis of this object will establish how general it is. This generality is *analytic*, as Yin (1989) so aptly put it. It is analytic in the sense that it is derived from the analysis of the case that is presented as the preferred vantage point. Thus, the representativeness of the case is relative to the methodological qualities attributed to it. The definition of the case permits an assessment of its generality in view of results of the analysis it has made possible.

The case study is an in-depth investigation. It accordingly uses different methods to collect various kinds of data and to make observations. This is mostly empirical material through which the object of study will be understood. The case study is based on a great wealth of empirical materials, notably because of their variety. But this wide variety of empirical materials presents analytical problems. The case study considers materials of different origins, which are produced by different types of knowledge. Knowledge of science can be unequivocally understood as "an attempt to rationalize facts with a knowledge that can be demonstrated" (Granger, 1986, p. 29), because it facilitates a clear understanding of what the object of study refers to in the materials.

The object under analysis is the object on which the analysis is based within the selected materials. It is presented in the appropriate form for each set of materials. The in-depth description provides a demonstrable means of understanding how the elements of materials under analysis, which appear in specific form, specifically refer to the object of study, an object that has been defined from a urban theory perspective.

It is appropriate to rigorously determine the object of study in the transformation from the theoretical definition of the object of study to its empirical construction within the selected materials. The variety of this material will safeguard the case study. The rigor of the definition of the object under analysis depends here on the description characteristics of the case study approach.

The case study must produce an explanation that cannot refer to or suggest any intuitive, tacit or imitative fields of knowledge that the study cannot provide and contain within a written statement. The theoretical and methodological foundations of the explanation resulting from the case study may be clearly understood through the description of the object of study.

Test case

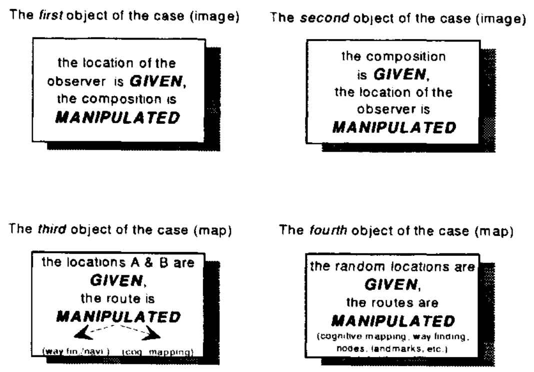
As we have already mentioned and evaluated in previous chapters, the main *initial theory* or *initial idea* of this research is derived from the concept of cognitive mapping. Cognitive mapping will be our case study approach, but we can also evaluate some other aspects of the proposed Design Tool. To apply the cognitive mapping approach in the testing model we selected objects of the case that can illustrate this initial idea. The selected objects of the case are perhaps not fully representative for illustrating the initial idea but they can represent the generality of it. The initial idea will be illustrated under the specific analysis of objects of the case, which are based within the selected material.
The test case can be defined in our research as the integration of *case study area, types of case, object of case* and *test protocol.*

The *case study area* is the selected urban area that will be subject to test analyses [test simulation]. The Campus of Delft University of Technology is the selected area for testing the *conceptual model* of our Design Tool. This area is selected for several reasons. The main reason is that the case study area is familiar to the users of the test case simulation. Familiarity with an urban environment is an important factor in the cognitive mapping approach, especially for conducting an analysis of it. In the case of our research familiarity is also important for the aspect of controlling the output results of the test simulation. Because of our familiarity with the selected case study area we will easily recognize possible difficulties in the test results. The second reason is that the area selected for testing consists of specific and characteristic urban elements typical for a Dutch environment, such as flatness, canals, greenery incorporated in a built up area, and so forth. The last, more practical reason is that a large amount of data and images related to the selected area are available.

Two *types of case*, related to the input image, are selected: the *image type* and the *map type*. The *image type* is selected mostly because it relates to the possible manipulation of the objects inside the image scene. The images of the *image type* are still snapshots taken from a video, which was produced as a result of the previous research[1]. The *map type* is selected in relation to way-finding and cognitive mapping analyses. The maps of the *map type* are taken from existing intersections of different maps, mostly topographical 2D maps, from the greater Delft area.

The selected *objects of the cases* are focused on four different objects, presented in Table 10.1 below. Each object of the case will be briefly introduced for each test case simulation.

Table 10.1
The objects of the cases


The *first* object of the case (image)
the location of the observer is **GIVEN**, the composition is **MANIPULATED**
The *second* object of the case (image)
the composition is **GIVEN**, the location of the observer is **MANIPULATED**
The *third* object of the case (map)
the locations A & B are **GIVEN**, the route is **MANIPULATED**
(way fin./navi.)
(cog. mapping)
The *fourth* object of the case (map)
the random locations are **GIVEN**, the routes are **MANIPULATED**
(cognitive mapping, way finding, nodes, landmarks, etc.)

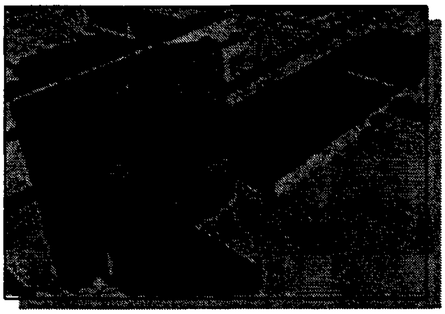
[1] See: Sidjanin 1995.

The process of the test case simulation is based on the *test protocol*. The *test protocol* will strictly follow the structure and function of the *conceptual model* of the Design Tool. This process will be presented in the test simulation by each simple functional step. It is not necessary to present all functions, which are repeated each time, such as the *recognition* and *filtering* process, for all four of the test cases. They will be presented fully in the first test case, and only indicated later. The focus of the test case will be on the specificity of each of the objects of the case, not on their commonality.

Test case simulation 1

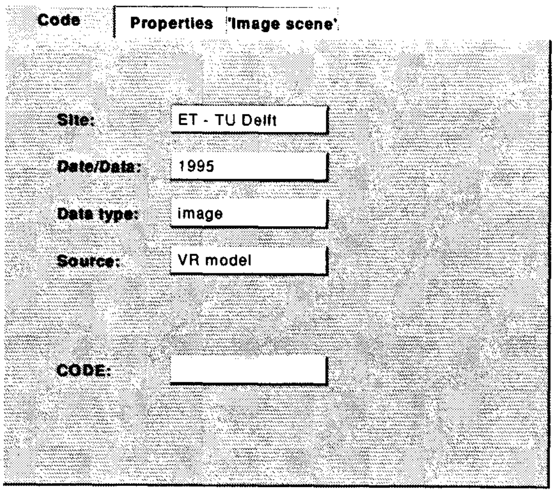
Main information
The first object of the case: the location of the observer is GIVEN, the composition is MANIPULATED.
Given object: the Faculty of Electrical Engineering with surrounding area.
Observers' position: flying mode.
Source of the image: snapshot from the VR model of the TU Delft campus.
Selected image: digital image.
Step one:
'Image selection' is the first step in *test case 1*. The user selects one image from several that exist in the computer image database. The image is in digital form and it does not need to be scanned. The selected image is the snapshot taken from the VR model of the TU Delft campus, Figure 10.2 (below).

Figure 10.2
The *input* image



Step two:
'Preparing input data' is the second step in *test case 1*. After selecting the image with the Design Tool the first window for input data appears. The user must fill it with the necessary information. The user fills the data window with the following information about the site - campus TU Delft, from which period of time is image - 1995, data type - image and from which source is the image taken - VR model. The user cannot fill the space for code; it will be automatically filled in after all data entered in the input data windows has been checked. The data entered in the first input data window is shown in Figure 10.3.

Figure 10.3
First *input data* window 'code'



Site:	ET - TU Delft
Date/Data:	1995
Data type:	image
Source:	VR model
CODE:	

After filling in the first input data window, the user can select the second one by clicking on the 'properties' button, and then the next window appears. This window concerns information related to image properties. The size of the image is entered automatically by the Design Tool and the user must enter all other information, e.g. image type - 2D, file format - .jpg, quality of the image - good and selected color mode - RGB color. The second input data window is presented in Figure 10.4. The image can be resized. This option is mostly used when the user is dealing with a number of images of different sizes and all of them need to be the same size.

Figure 10.4
Second *input data* window 'properties'

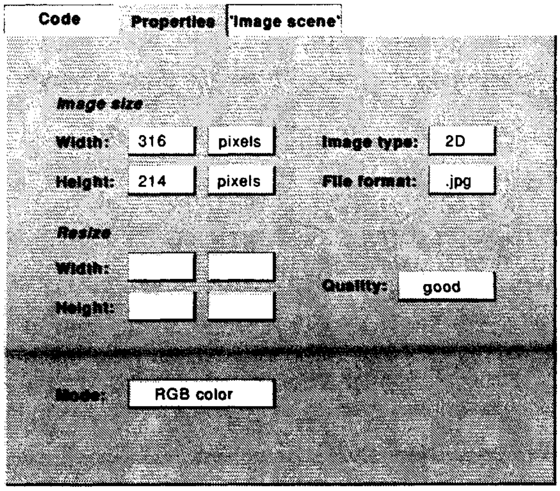
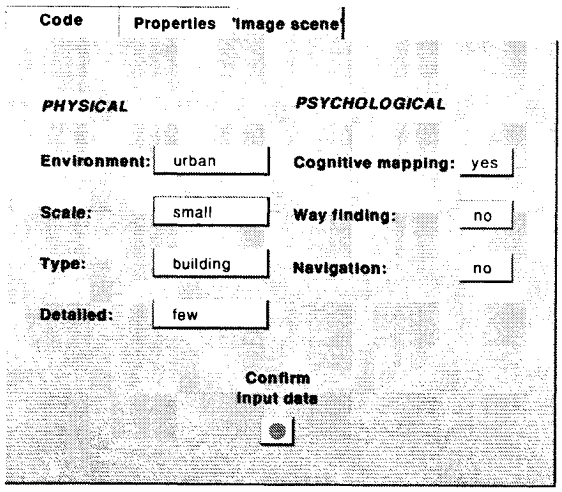


Image size
Width: 316 pixels
Height: 214 pixels
Image type: 2D
File format: .jpg
Resize
Width:
Height:
Quality: good
Mode: RGB color

After input data has been entered in the second window, the user can select the image scene button. Than the third and last *input data* window appears. Data belonging to this window relates to characteristics of the image scene and to general tasks of the Design Tool. There are two groups of information, the first belonging to *physical* and the second to *psychological* characteristics. Physical characteristics are defined by environment - urban, scale - small, type - building and detailed - few. Psychological characteristics are confirmed only by the cognitive mapping button - yes, and not confirmed by way-finding and navigation. When all *input* data from all three windows is satisfactorily entered, then the user can confirm the *input* data by clicking on the button. The third input data window is shown in Figure 10.5.

Figure 10.5
Third *input data* window 'image scene'



PHYSICAL		*PSYCHOLOGICAL*	
Environment:	urban	Cognitive mapping:	yes
Scale:	small	Way finding:	no
Type:	building	Navigation:	no
Detailed:	few		

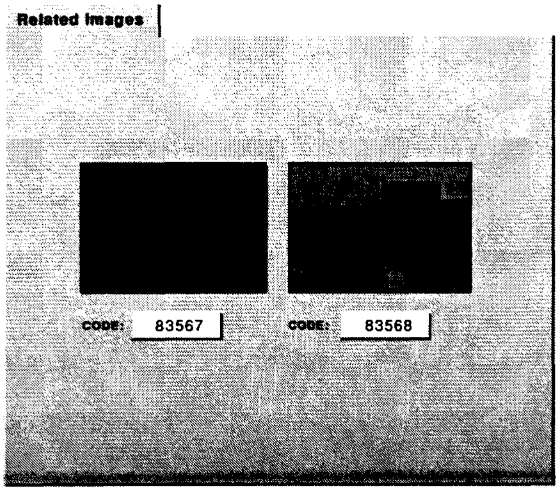
After confirmation of the *input* data, the system automatically checks it through the 'codes library'. In *test case 1*, after finishing checking the *input* data, the system automatically gives a new *code*.

If the user needs to for any reason, he can make the Design Tool show all 'related images' to the new open image in a new window as a thumbnail. These images can, with all stored data, belong to each other and to the new open image. They have 'codes' which are physically close to each other, which also means that the object from the image scene is the same. The images can be chosen for further processing directly from the window (see Figure 10.6).

Process in 'between':

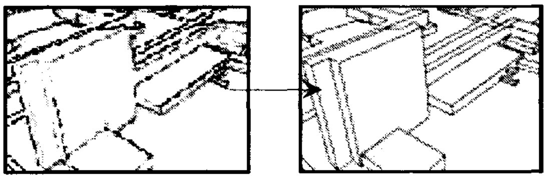
The process in 'between' is the process that simulates the processes based on the 'computer cognition model'. In the 'recognition', 'filtering' and 'synthesis' processes, the user cannot directly interact with the Design Tool. The result of the 'recognition' process is a 'contour map' image (see Figure 10.7 - A). The same *recognized* image after the 'filtering' and 'synthesis' processes is also a 'contour map' image, called a 'working image' (see Figure 10.7 - B). Both images are presented as the results of the indicated processes of the *input* image, which is shown in Figure 10.2. The 'working image' is the final

Figure 10.6
Related images' to the *input* image



result (and final image) of the process based on the 'computer cognition model'. The 'working image' is a visible image for the user and with it starts the 'evaluation' process.

Figure 10.7
The images after finishing the 'computer cognition processes'



A. The 'contour map' of the original *input* image, after the 'recognition' process

B. The 'working image' is the 'contour map' of the *recognized* image after finishing the 'filtering' and 'synthesis' processes

Step three:
'Evaluation' is the next step in the testing process. The 'working image' appears and overlaps the *Layer0* image. There is also an option ['layers'], selected from the main menu whereby these two images stay beside each other. The 'working image' is a 'contour map' image and *Layer0* is a copy of the original *input* image. By using the 'layers' option, these two images can be integrated into one, or the changed 'working image' can be rendered. The user can select each of the objects from the image scene [from 'working image'] by simply clicking on one of them with the mouse. The user can change the shapes of the selected object. The user can use the 'copy' and 'paste' options for moving the object to a new position inside the image scene. It is also possible to 'delete' a selected object or to 'insert' a new object into the image scene. To 'insert' a new object, the user must pre-prepare this new object for the insertion process.

In *test case 1*, the user selects the object of the Faculty of Electrical Engineering. The selected object is activated by its frame contour (see Figure 10.8). The user wants to change the shape of the selected object, according to the object case task [location is *given*, composition is *manipulated*]. With the mouse the user 'grabs' the indicator point on the contour frame of the selected object and changes the shape and the height of the object (see Figure 10.9). By this process the user can change the shapes and forms of simple and very complex objects from the same image scene. Objects of the image scene that can be manipulated by the user are greenery, water and paths.

Figure 10.8
Selected object of the Faculty of Electrical Engineering

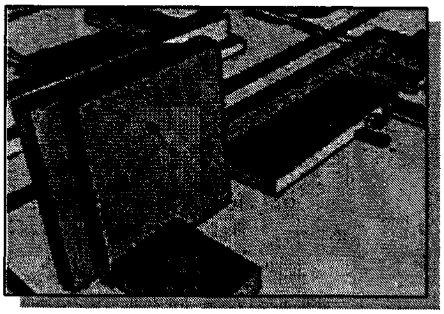
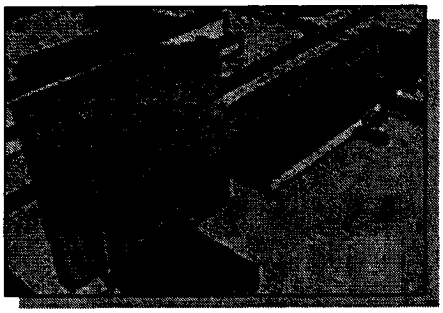


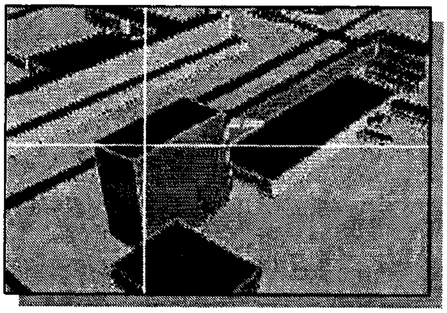
Figure 10.9
Changing the shape and height of the selected object



Step four:
'Creation' is the next step in the testing process in which a final decision can be made. If the user is satisfied with the reshaping of the existing object/s, or the insertion of new object/s or even the deletion of some object/s, then he or she can select the 'finish the process' option from the main menu. The Design Tool will immediately process all relevant information related to the image scene [to the *input* image *code*] from the database system [data model, criteria, GIS and other data, etc.] until the new final solution is automatically made. There are a couple of possibilities for a final solution, which will be explained in our *test case* example.

If a solution proposed by the user is acceptable to the Design Tool after the decision process is finished, then the final solution will be the same as what the designer proposed. This will be the 'ideal solution' where the designer has full support from the Design Tool. In our *test case* the final solution is presented as an *output* image in Figure 10.10.

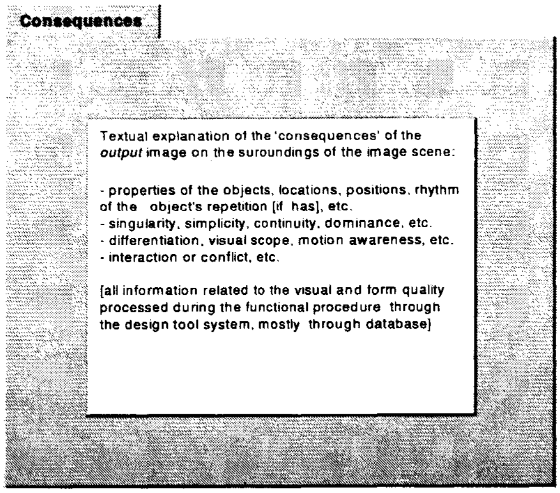
Figure 10.10
The - *output* image



If the user wants to know the consequences and effects (or even comments) which this acceptable manipulated new-modeled image has on the surrounding environment from the image scene, then he or she has to select the 'consequences' option from the main menu. A new set of information related to the visual and form quality will immediately appear in textual or spreadsheet form, shown in Figure 10.11.

A second possibility arises if after decision processing the Design Tool shows information that the solution proposed by the designer is not acceptable. If the user wants to know the reason for the rejection by the Design Tool, he or she has to select the 'more information' option. A new window with relevant information about the reasons for rejection and showing instructions for necessary corrections will appear. Then the user can follow the instructions and he or she can change, in our *test case*, the shape of the reshaped object of the Faculty of Electrical Engineering. After this, the Design Tool will execute another decision process. If the new proposed shape is acceptable then the output image can be formed. If the proposed shape is again rejected, then the whole process must be repeated. It is also possible for the user to implement a new set of criteria which, after the entire reshaping of the object, should be acceptable. If not, then the Design Tool will suggest that the input image should be the same as the output image. The Design Tool does not change an image that is compatible [for several reasons; in our *test case* because the selected object of the Faculty of Electrical Engineering is the main landmark and by changing the height of the object the object will lose this important characteristic] with the new one which has been rejected several times.

Figure 10.11
The *consequence* window - textual form



Textual explanation of the 'consequences' of the *output* image on the suroundings of the image scene:

- properties of the objects, locations, positions, rhythm of the object's repetition [if has], etc.
- singularity, simplicity, continuity, dominance, etc.
- differentiation, visual scope, motion awareness, etc.
- interaction or conflict, etc.

[all information related to the visual and form quality processed during the functional procedure through the design tool system, mostly through database]

The last and exceptional possibility is when all [input and reshaping images] solutions proposed by the designer are strictly rejected by the Design Tool. In this case a new window appears with the following information: 'NO accepted design solutions'. In this case the user can finish his task without any new results except to start processing again with completely new tasks.

Test case simulation 2

Main information

The second object of the case: the composition is GIVEN, the location of the observer is MANIPULATED.
Given object: the Faculty of Electrical Engineering with surrounding area.
Observers' position: flying mode.
Source of the image: the snapshots from the VR model of the TU Delft campus.
Selected image: digital image.

Step one:
'Image selection' is also the first step in *test case 2*. The user selects three images from the stored image library. The selected images are the same as those which relate to the previously analyzed image in *test case 1* (see Figures 10.2 and 10.6).
Step two:
'Preparing input data' is the second step in *test case 2*. After selecting the images with the Design Tool the first window for input data appears. The user fills in the data window with the following information about the site - campus TU Delft, from which period of time is the image - 1995, data type - image and from which source is the image taken - VR model. The first data window filled in has the same information as the first data window from *test case 1*. The user cannot fill the space for code; it will be automatically filled in after all entered data in the input data windows and all codes of the selected images are checked. The data entered in the first input data window is the same as the window shown in Figure 10.3.

The second data window contains information related to the image properties. In this case the image size cannot be automatically shown. The user must resize all three images to the same size: width: 243 pixels and height: 165 pixels. The remaining information is the same as in *test case 1*, i.e. image type - 2D, file format - .jpg, quality of the image - good and selected color mode - RGB color. The second input data window is presented in Figure 10.12.

The data of the last *input data* window concerns characteristics of the image scene and the general tasks of the Design Tool. There are two groups of information, the first belonging to *physical* and the second to *psychological* characteristics. Physical characteristics are defined by environment - urban, scale - small, type - building and detailed - surrounding. Psychological characteristics are confirmed only by navigation button - yes, which means that the Design Tool will work in 'navigation mode'. The other two buttons for cognitive mapping and way-finding are cancelled with no. When all input data is entered then the user can click with the mouse on the confirm input data button. The third input data window is shown in Figure 10.13.

Figure 10.12
Second *input data* window

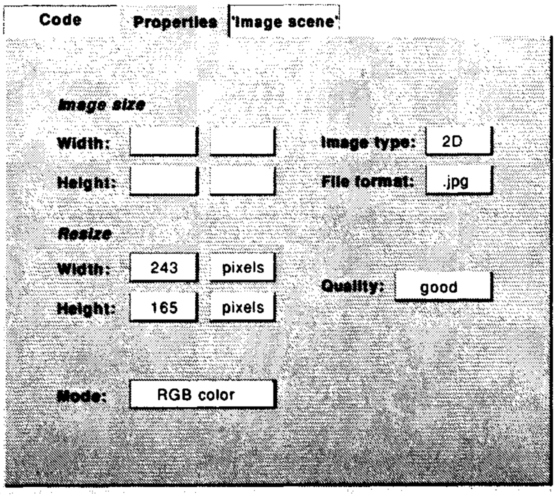


Image size

Width:

Height:

Image type: 2D

File format: .jpg

Resize

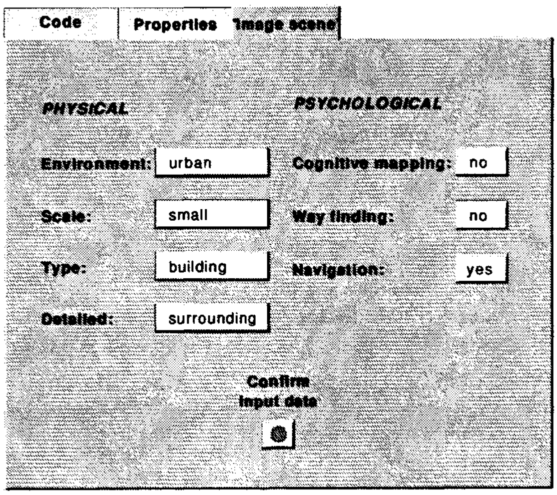
Width: 243 pixels

Height: 165 pixels

Quality: good

Mode: RGB color

Figure 10.13
Third *input data* window



PHYSICAL

Environment: urban

Scale: small

Type: building

Detailed: surrounding

PSYCHOLOGICAL

Cognitive mapping: no

Way finding: no

Navigation: yes

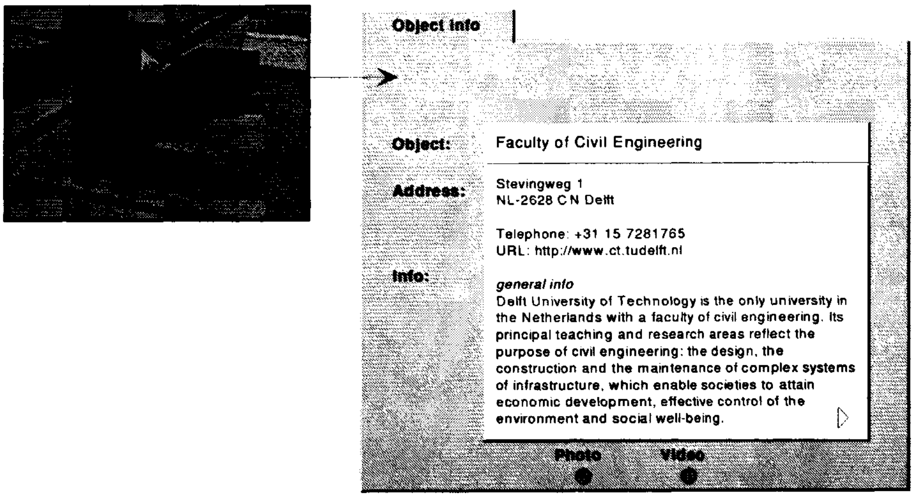
After confirmation of the input data, the system automatically executes a check of the input data through the 'codes library'. After checking the *input* data, the system automatically gives a new *code*.
Step three:
'Evaluation' is the next step in the testing process. The 'working images' appear and overlap the *Layer0* images. The 'working images' are 'contour map' images and *Layer0* is a copy of the original *input* images, all presented in Figure 10.14.

Figure 10.14
The 'working images' overlapping the *Layer0* images

One of the important possibilities of the Design Tool is the ability to take all available data about the objects directly from the image scene. In our test case the user selects the object of the Faculty of Civil Engineering with two short clicks on the *left button* of the mouse, from one of the three available images. A new object info window immediately appears, with all relevant information about the Faculty of Civil Engineering e.g. address, info phone, web address [with active link, if the user's computer is connected to the web], main functions of the Faculty, departments and all other relevant information. All information is available on one or several window pages. Searching through the pages is very simple, and is done by clicking on the left or right navigation arrow. The procedure for the object info window is the same for each object from the image scene in any other case. The object info window is shown in Figure 10.15.

Another possibility that the Design Tool can offer the user is to measure the distance between objects inside an image scene. The user first has to select objects with a longer click on the chosen objects. Then the object distance window appears with relevant information. Another possibility is that the user first chooses from the menu the function of object distance and then selects objects for measuring. There are three types of distances: air [shortest distance between centers of the objects], objective [real distance

Figure 10.15
The *object info* window



Object: Faculty of Civil Engineering

Address: Stevingweg 1
NL-2628 CN Delft

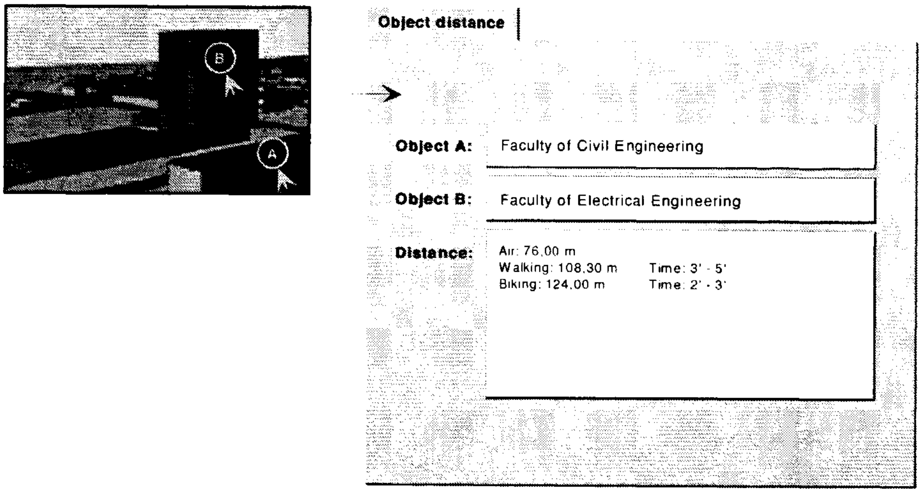
Telephone: +31 15 7281765
URL: http://www.ct.tudelft.nl

Info:
general info
Delft University of Technology is the only university in the Netherlands with a faculty of civil engineering. Its principal teaching and research areas reflect the purpose of civil engineering: the design, the construction and the maintenance of complex systems of infrastructure, which enable societies to attain economic development, effective control of the environment and social well-being.

between objects] and subjective [distance between objects according to the chosen mode: walking, cycling, car and public transport]. The user can select one or more of the types. Distance measurement of objects is based on image geometry, which is always related to the geographical coordinative system. One example is shown in Figure 10.16, where the user chooses to measure air, walking and cycling distance.

If the user wants to choose the object type *street*, he or she has to click with the left mouse button over it. Then the selected object type *street* is activated. A new street info window will immediately appear. The following information related to the selected street is presented on the street info window: name, characteristics and buildings. The street name is indicated separately. Street characteristics are based on the street profile and other related data from the database. The street buildings are a listing of all the buildings visible on the selected image and belonging to the street by their address and physical coexistence. All buildings on the listing are indicated with their street addresses. If the user wants to see more information about a particular building from a chosen street, he or she simply has to click on a desired name and a new object info window appears. This object info window presents all relevant information about the chosen building. If some

Figure 10.16
The *object distance* window



Object A: Faculty of Civil Engineering

Object B: Faculty of Electrical Engineering

Distance:

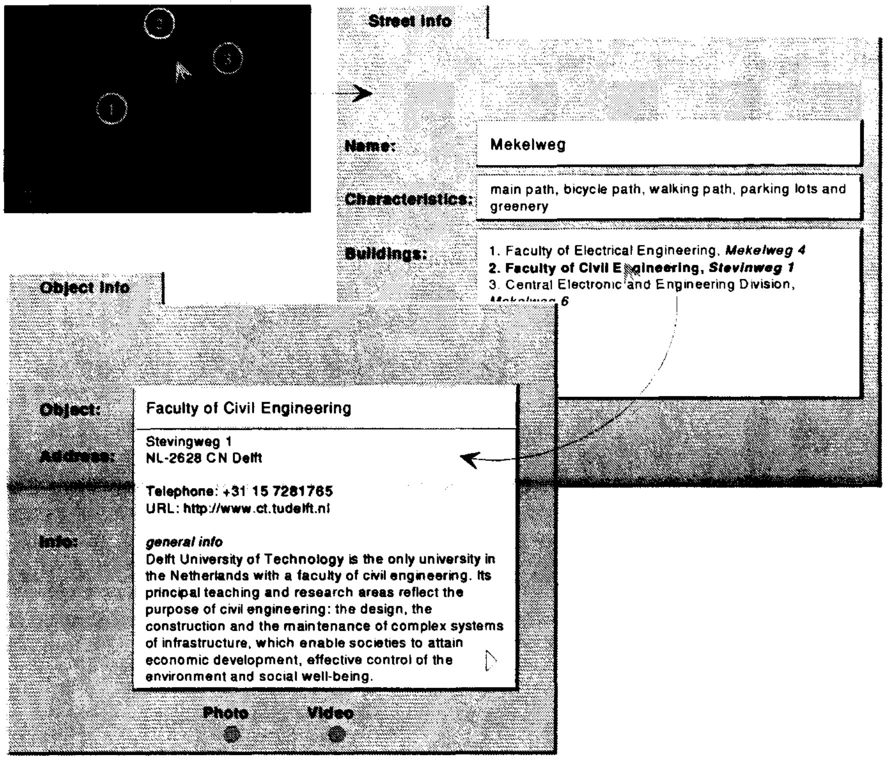
```
Air: 76.00 m
Walking: 108.30 m      Time: 3' - 5'
Biking: 124.00 m       Time: 2' - 3'
```

photo or video data exists in the database, then this information will be indicated in the object info window. By simply clicking the user will open one of them in a new photo or video window. The street info window is shown in Figure 10. 17.

Inside each selected image it is possible to locate the position of north, simply by clicking on the right mouse button. The position of north is displayed in the top-right corner of the image. The position of north inside the image is related to the geometry of the objects and their geographical coordinative system. It is always an absolute indication of north. The procedure for the direction of north is always the same for each active 'working image'.

As we see, the Design Tool presents information taken from the database related to the objects from the active 'working image'. There are several types of information. The first is GIS base data, which directly relates to the geometry of the objects from the image scene and the geographical coordinative system. The algorithmic processing helps the user to take some information which is not precisely indicated by the image, such as distances or even the object's elevation. For checking the height of the object from the image scene the user can use the object elevation option from the main

Figure 10.17
The *street info* and *object info* windows



Name: Mekelweg

Characteristics: main path, bicycle path, walking path, parking lots and greenery

Buildings:
1. Faculty of Electrical Engineering, *Mekelweg 4*
2. **Faculty of Civil Engineering, *Stevinweg 1***
3. Central Electronic and Engineering Division, *Mekelweg 6*

Object: Faculty of Civil Engineering

Address: Stevingweg 1
NL-2628 CN Delft

Telephone: +31 15 7281765
URL: http://www.ct.tudelft.nl

Info: *general info*
Delft University of Technology is the only university in the Netherlands with a faculty of civil engineering. Its principal teaching and research areas reflect the purpose of civil engineering: the design, the construction and the maintenance of complex systems of infrastructure, which enable societies to attain economic development, effective control of the environment and social well-being.

menu. Then he or she has to select the object whose height they want to see, and the resulting number presented in meters will immediately appear flashing on the top-right of the selected object (see also Figure 10.10). The second important type of information is taken from different databases, such as data of object assignments, a University database, street profiles or greenery. And the last data type is related to data about the image itself, such as the indication of the direction of north.

The Design Tool integrates all different data types in efficient and usable information for the user, displaying them in a simple, clear and highly visible way. All functional procedures presented here are the same for all other cases, images and tasks. They are generic in the Design Tool.

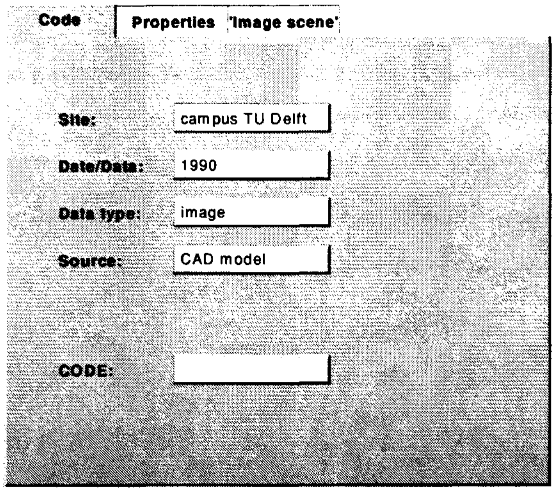
Test case simulation 3 - A & B

Main information

The third object of the case: the locations A and B are GIVEN, the route between them is MANIPULATED.
Given object: A - the Aula of TU Delft and B - Interfaculty Reactor Institute of TU Delft.
Observers' position: flying mode.
Source of the image: the CAD model of the TU Delft campus.
Selected image: digital image.
Tasks: way-finding and navigation.
Step one:
The first step in *test case 3* is also the same as in the previous test cases: 'image selection'. The user selects a 2½D-image map for testing from a particular CAD project, which is in digital form and does not need to be scanned.
Step two:
The *input* data must be prepared. After selecting the image with the Design Tool, the first window for *input* data appears. The user fills in the data window with the following information about the site - campus TU Delft, from which period of time is the image - 1990, data type - image and from which source is the image taken - CAD model. The code will be automatically filled in after all entered data is checked. The first *input* data window is shown in Figure 10.18.

After filling in the first code input data window, the user can select a second one: properties. This window concerns information related to image properties. The image size is automatically entered by the Design Tool and the user must fill in all other information, e.g. image type - 2½D, file format - .dxf, quality of the image - excellent and selected color mode - gray scale. The second input data window is presented in Figure 10.19.

Figure 10.18
First *input data* window 'code'



Site:	campus TU Delft
Date/Data:	1990
Data type:	image
Source:	CAD model
CODE:	

Figure 10.19
Second *Input data* window 'properties'

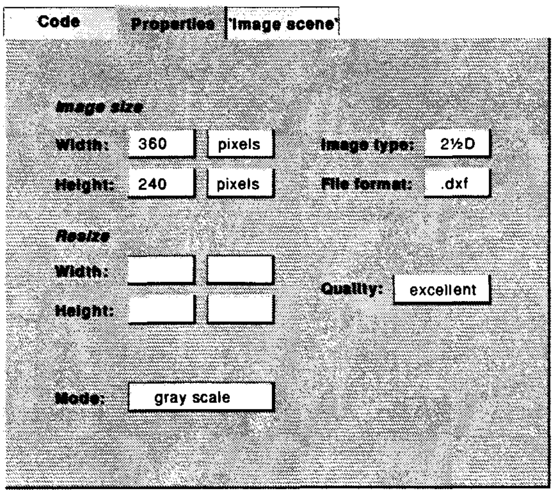


Image size

Width:	360	pixels	Image type:	2½D
Height:	240	pixels	File format:	.dxf

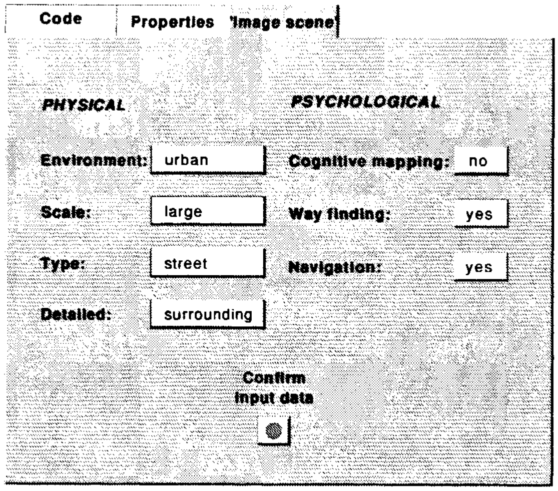
Resize

Width:			Quality:	excellent
Height:				

Mode: gray scale

After finishing filling in the properties window, the user can select the image scene button. Then the third and last *input data* window appears. Data that belongs to this window relates to *physical* and *psychological* characteristics. Physical characteristics are defined by environment - urban, scale - large, type - street and detailed - surrounding. Psychological characteristics are confirmed by way-finding and navigation buttons - yes, and not to cancel cognitive mapping. When the *input* data is finished then the user can confirm it by clicking on the button. The third input data window is shown in Figure 10.20.

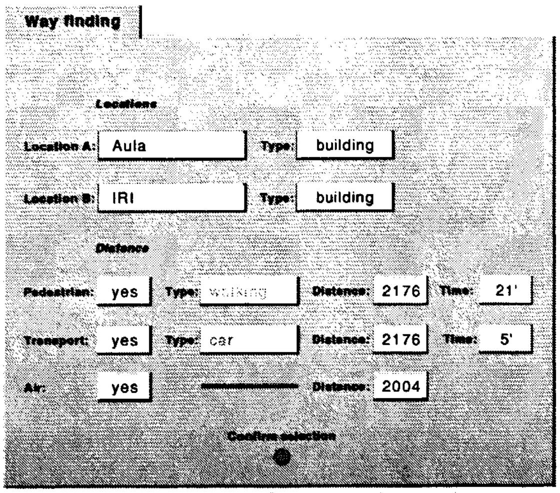
Figure 10.20
Third *input data* window 'image scene'



PHYSICAL		PSYCHOLOGICAL	
Environment:	urban	Cognitive mapping:	no
Scale:	large	Way finding:	yes
Type:	street	Navigation:	yes
Detailed:	surrounding		

After confirmation of the *input* data, the Design Tool automatically executes a check of the *input* data and gives a new *code.*
Step three:
'Evaluation' is the next step in the testing process with which ***test case 3 - A*** started. After finishing the process of filling in *input* data, the user can choose the way-finding button from the main menu and a new window appears. The user can then fill this window with a new set of information related to the task. The filled way-finding window is shown in Figure 10.21.

Figure 10.21
The *way-finding* window



Way finding

Locations

Location A: Aula Type: building

Location B: IRI Type: building

Distance

Pedestrian: yes Type: walking Distance: 2176 Time: 21'

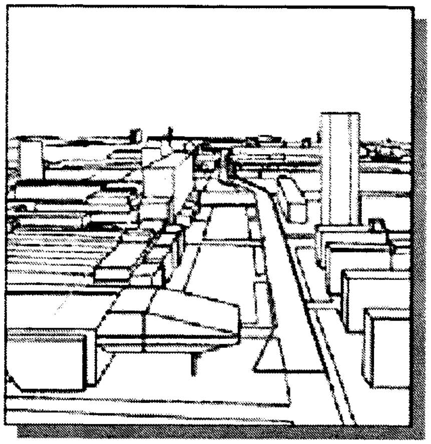
Transport: yes Type: car Distance: 2176 Time: 5'

Air: yes Distance: 2004

The user chooses to test way-finding and navigation through the CAD model map related to walking, car and air distance. When he or she enters information about locations A and B and defines which distance is wanted, e.g. walking and car, the distance measurement in meters and time in minutes automatically appear in the window. The walking distance is indicated in green, the same one that will be presented on the image map. The car distance is indicated in red. The air distance is indicated in the window only by a blue line that will be the same on the image map.

The 'working images' appear and overlap the *Layer0* images. As we already explained, the 'working images' are 'contour map' images and *Layer0* is a copy of the original *input* images. The 'working images' appear automatically, after the way-finding window has been filled in, next to it. The three types of chosen distances are graphically presented on the image. The walking distance in red, the cars distance in green and the air distance in blue lines. These three lines graphically indicate the route [way-finding] between the selected objects A and B through the map image shown in Figure 10.22.

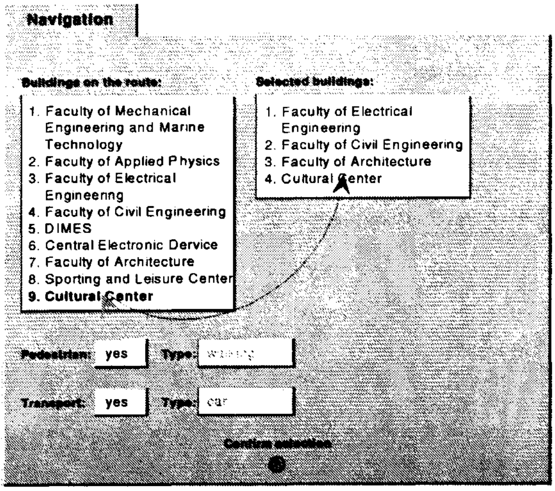
Figure 10.22
The *routes* between objects A & B



After finishing the way-finding testing, the user can choose the navigation button from the main menu and ***test case 3 - B*** starts. The new navigation window appears with a listing of all buildings that belong to the routes. The user makes a selection of some of the buildings from the list for which he or she would like to know positions on the map image. By clicking with the mouse over the selected building's name from the list, the user makes a selection of buildings. The list of new selected buildings appears in a new window beside the listing window. The user also has the opportunity to opt for an additional route that can connect the selected buildings to the main route. The navigation window is shown in Figure 10.23.

When the user selects the buildings and route types, he or she has to confirm selection by pressing the corresponding button. The Design Tool quickly presents the results of the testing tasks in a new window, shown in Figure 10.24. Indicating numbers related to the listed buildings are on top of the corresponding objects on the map image. A sub-route to the selected buildings is added to the main routes.

Figure 10.23
The *navigation* window



Navigation

Buildings on the route:

1. Faculty of Mechanical Engineering and Marine Technology
2. Faculty of Applied Physics
3. Faculty of Electrical Engineering
4. Faculty of Civil Engineering
5. DIMES
6. Central Electronic Dervice
7. Faculty of Architecture
8. Sporting and Leisure Center
9. **Cultural Center**

Selected buildings:

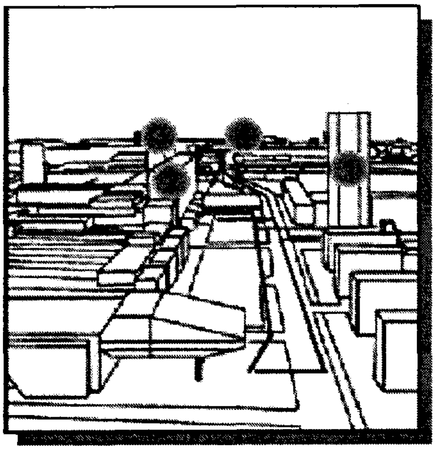
1. Faculty of Electrical Engineering
2. Faculty of Civil Engineering
3. Faculty of Architecture
4. Cultural Center

Pedestrian: yes Type: walking

Transport: yes Type: car

Confirm selection

Figure 10.24
The positions of the objects and new routes



On this *output* image the user can see the location and position of the selected buildings in the image scene and the route showing how to get from location A [Aula] to the selected buildings according to his or her choice [walking or by car]. If the user needs more information about the selected objects, e.g. about the Faculty of Civil Engineering, he or she can click on this particular object and then a new object info window appears [the same one that was presented in Figures 10.15 and 10.17]. We explained this functional operation in more detail earlier.

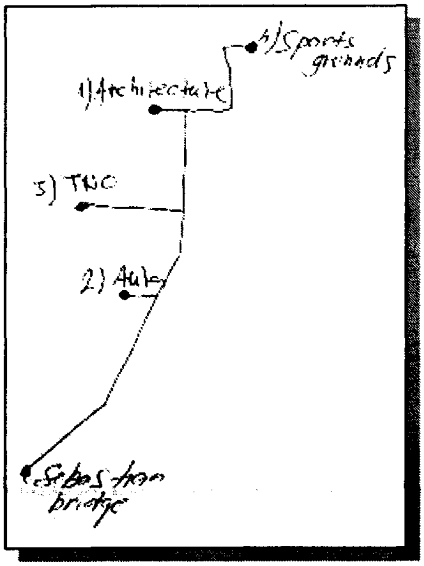
Test case simulation 4

Main information

The fourth object of the case: the random locations are GIVEN, the routes are MANIPULATED.
Given object: couples of random locations of TU Delft.
Source of the image: the sketch drawing - cognitive map.
Selected image: sketch on paper.
Tasks: analysis of different routes according to cognitive map.
Step one:
In *test case 4* the 'image selection' is different to previous test cases. The user has made a sketch drawing as a cognitive map according to his or her day schedule. The cognitive map drawing has to be scanned before the computer cognition processing will start. The scanned *input* image is presented in figure 10.25.

The cognitive map sketch drawing presents the user's daily time schedule. It is a cognitive map of the campus of TU Delft. The user wants to 'enter' the campus from 'Sebastian bridge' and then go to '1) Architecture' [Faculty of Architecture]. After some time the user wants to go to have lunch in '2) Aula', probably with some friends. After lunch the user has a meeting at '3) TNO' [an institute which does not belong directly to the buildings of the TU Delft campus]. After finishing the meeting the user wants to go and play tennis on '4) Sport grounds'. To implement this time schedule presented by the cognitive map the user has to know the routes between the indicated buildings. The task of *test case 4* is to perform an analysis of the routes between buildings randomly selected by the user. But first, the user has to fill in all input data, according to cognitive map drawings and tasks, in the *input* data windows.

Figure 10.25
The scanned cognitive map sketch drawing

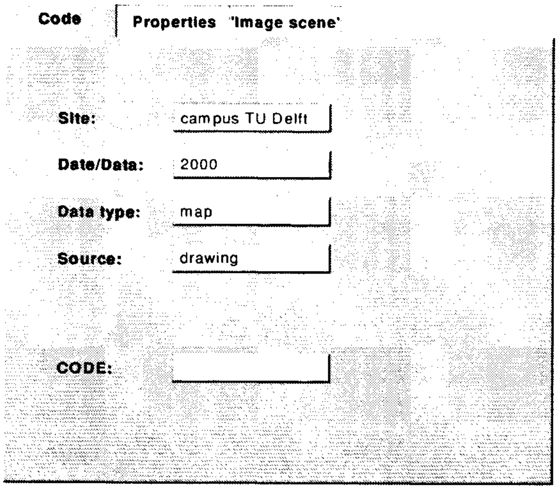


Step two:
After scanning the sketch drawing, the user can start the Design Tool and the first window for *input* data appears. The user fills the first data window with the following information shown in Figure 10.26.

The second *input* data window is also filled in (see Figure 10.27). The third and last *input* data window is filled with relevant information related to the cognitive map and main tasks. The third input data window is shown in Figure 10.28.

After filling in the *input* data, the user has to choose the map on which the Design Tool will perform analyses. After confirmation of the *input* data the new related maps window will appear. In this window, thumbnail maps directly related to the indicated area - campus of TU Delft, are presented. The maps shown are also presented with their *codes*. A 'working map' can be directly chosen by the user for further processing, by simply clicking with the mouse on it.

Figure 10.26
First *input data* window 'code'



Site: campus TU Delft
Date/Data: 2000
Data type: map
Source: drawing
CODE:

Figure 10.27
The second *input data* window 'properties'

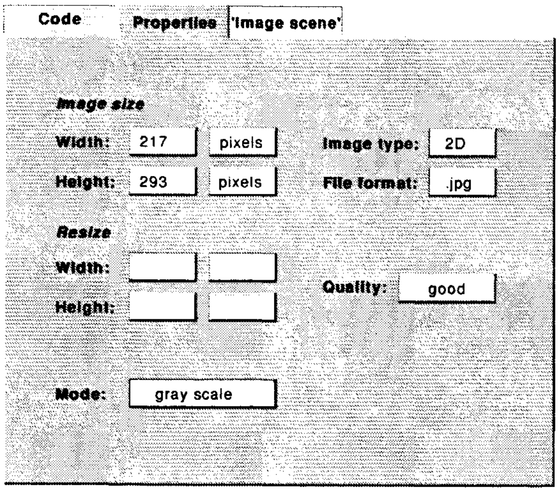
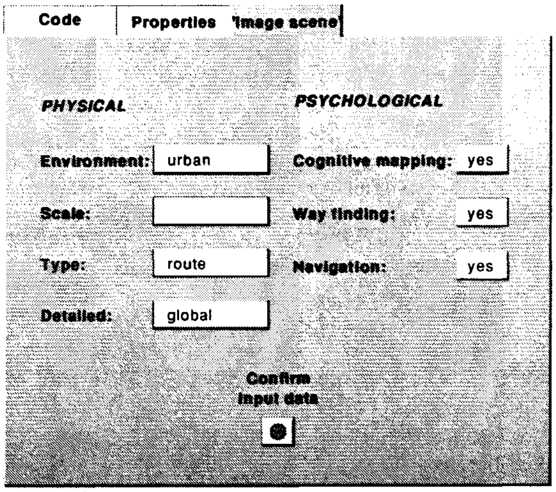


Image size
Width: 217 pixels
Height: 293 pixels
Image type: 2D
File format: .jpg
Resize
Width:
Height:
Quality: good
Mode: gray scale

Figure 10.28
The third *input data* window 'image scene'



PHYSICAL

Environment: urban

Scale:

Type: route

Detailed: global

PSYCHOLOGICAL

Cognitive mapping: yes

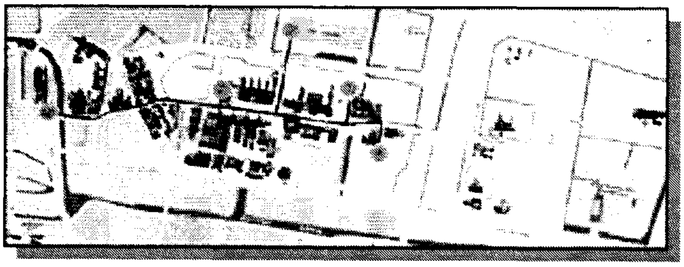
Way finding: yes

Navigation: yes

Step three:
'Evaluation' is the next step in the testing process. After the process of entering *input* data, selecting the 'working map' and the completion of the 'invisible' process of computer cognition, the new cognitive map window appears. In this window the input cognitive map overlaps and stretches the selected 'working map' (see Figure 10. 29). The cognitive map becomes 'real' because all the indicated objects on the sketch drawing occupy places in real positions of buildings on the map. All buildings on the map are indicated with numbers and connected with lines. The cognitive map, which was indicated by a sketch drawing, becomes a real spatial cognitive map.

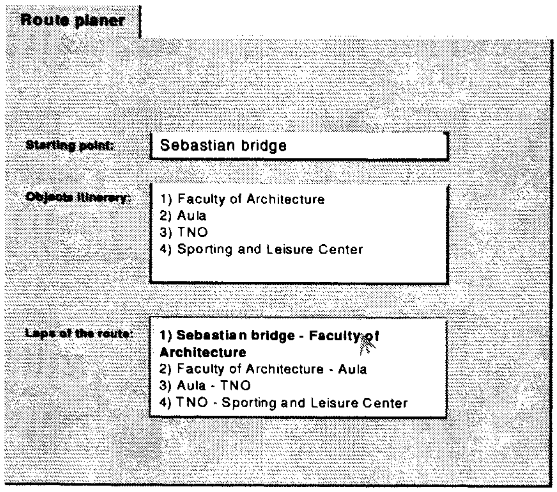
The starting point [sp] for the new cognitive map is Sebastian bridge. The other points [buildings] are indicated with numbers [from 1 to 4], which are located on top of the representative buildings on the selected map. As the map is in digital form and deals with spatial geometry and GIS data, all kinds of possible analyses are available. Here, the user will perform tests with some of the tool possibilities e.g. way-finding, navigation, route planning and route analyses.

Figure 10.29
The real cognitive map stretching and overlapping the 'working map'



If the user wants to test a route of the cognitive map he or she has to select the route planer option from the main menu. The new route planer window will appear with all indicated routes (see Figure 10. 30).

Figure 10.30
The *route planner* window



Route planer

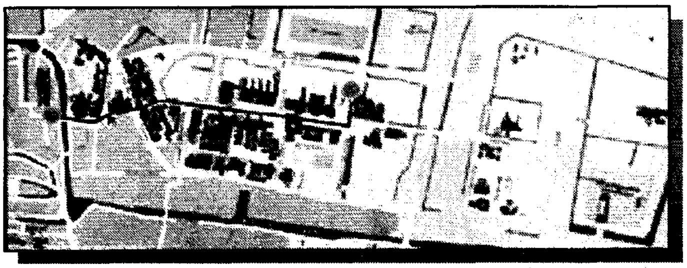
Starting point: Sebastian bridge

Objects itinerary:
1) Faculty of Architecture
2) Aula
3) TNO
4) Sporting and Leisure Center

Laps of the route:
1) Sebastian bridge - Faculty of Architecture
2) Faculty of Architecture - Aula
3) Aula - TNO
4) TNO - Sporting and Leisure Center

The route planner presents the starting point, object itinerary [buildings] and their laps of the route, graphically presented on the cognitive map by numbers and lines. The user has to choose one lap from the laps of the route, such as Sebastian bridge - Faculty of Architecture for further analyses. The selected lap of the route is shown in Figure 10.31.

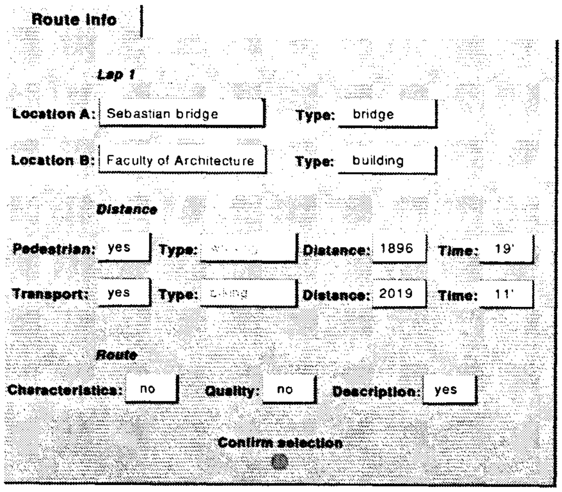
Figure 10.31
The selected lap: Sebastian bridge - Faculty of Architecture



If the user wants to take information from the selected lap he or she has to click with the mouse on the lap line and the new route info window shortly appears beside the other two open windows [route planner and 'working map' with the selected lap]. The route info is shown in Figure 10.32.

The route info window presents information related to the main objects of lap 1, distances between objects and some information related to the route such as characteristics, quality and description. If the user wants to test the description of the route he or she first has to confirm the selection by clicking on the button. The new route description window appears beside the other open windows. The route description window presents in textual form a general description of the selected route [lap 1], such as direction, street names, main buildings etc. If the user wants, for example, to test the quality of the selected route, he or she has to confirm the quality button on the route info. The new route quality window will appear (see Figure 10.33).

Figure 10.32
The *route info* window



Lap 1

Location A: Sebastian bridge Type: bridge

Location B: Faculty of Architecture Type: building

Distance

Pedestrian: yes Type: [illegible] Distance: 1896 Time: 19'

Transport: yes Type: biking Distance: 2019 Time: 11'

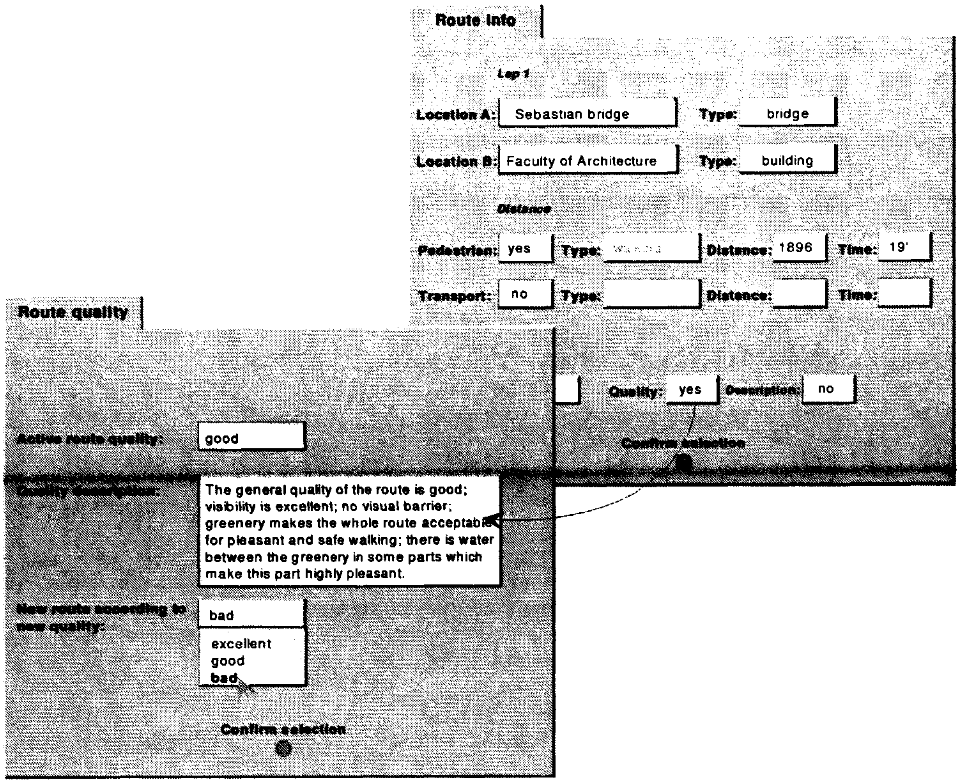
Route

Characteristics: no Quality: no Description: yes

The route quality window presents the Design Tool's statement of the quality of the active route - good and a short quality description. One possibility is that the user can choose a new route according to a selection of route quality. He or she has to select one from the three categories of quality from the pop-up menu: excellent, good and bad. For the test case the user chooses the bad quality route. Then the bad route for *lap 1* will be shown in the new 'working map' window. This new route can be tested again with the same procedure.

If the user wants to test the route characteristics he or she has to confirm the characteristic button from the route info window. Then the new route characteristic window will appear (see Figure 10.34).

Figure 10.33
The *route info* and *route quality* windows


Route info

Lap 1

Location A: Sebastian bridge Type: bridge

Location B: Faculty of Architecture Type: building

Distance

Pedestrian: yes Type: [illegible] Distance: 1896 Time: 19'

Transport: no Type: Distance: Time:

Quality: yes Description: no

Confirm selection

Route quality

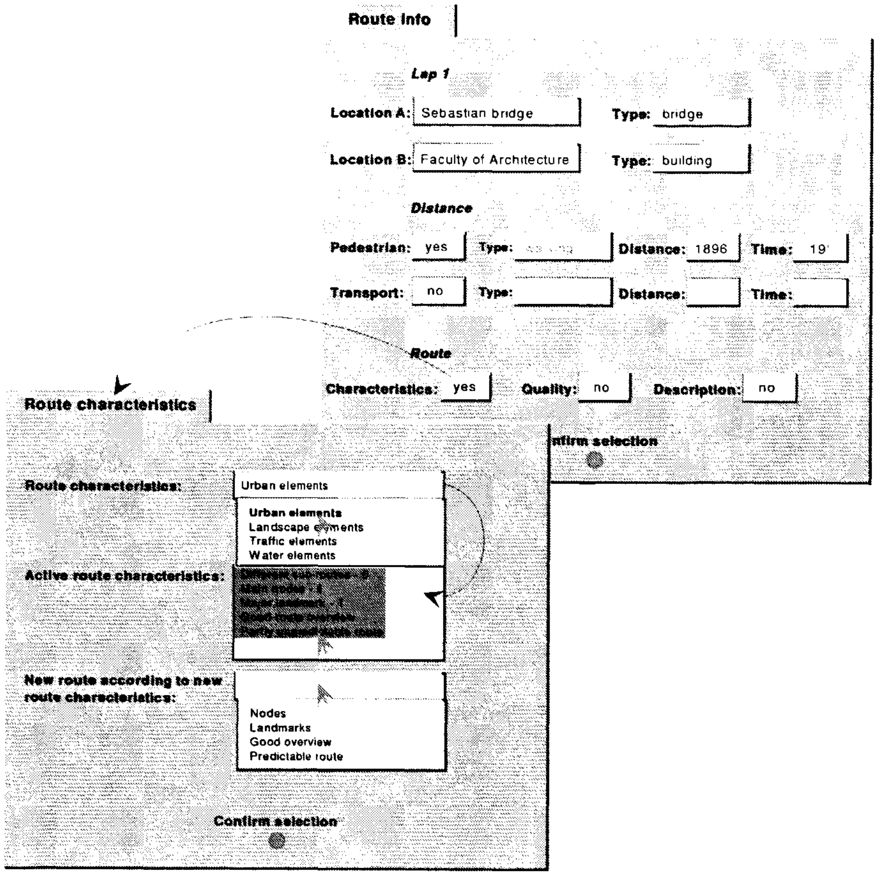
Active route quality: good

Quality description: The general quality of the route is good; visibility is excellent; no visual barrier; greenery makes the whole route acceptable for pleasant and safe walking; there is water between the greenery in some parts which make this part highly pleasant.

New route according to new quality: bad

In the route characteristics window, the user first has to decide which route characteristic he or she would like the Design Tool to analyze. He or she has to choose from the pop-up menu one from urban, landscape, traffic and water characteristics. In *test case 4* the user decides to analyze the route according to urban characteristics. The Design Tool then shows the results of analyses in a sub-window, the actual route characteristics, according to selected urban characteristics. If the user selects one of these resulting characteristics, it will be graphically presented immediately in the window already open with *lap 1*. If the user selects all characteristics, then all of

Figure 10.34
The *route info* and *route characteristics* windows


Route Info

Lap 1

Location A: Sebastian bridge **Type:** bridge

Location B: Faculty of Architecture **Type:** building

Distance

Pedestrian: yes **Type:** [illegible] **Distance:** 1896 **Time:** 19'

Transport: no **Type:** **Distance:** **Time:**

Route

Characteristics: yes **Quality:** no **Description:** no

nfirm selection

Route characteristics

Route characteristics: Urban elements

Urban elements
Landscape elements
Traffic elements
Water elements

Active route characteristics: [illegible]

New route according to new route characteristics:

Nodes
Landmarks
Good overview
Predictable route

Confirm selection

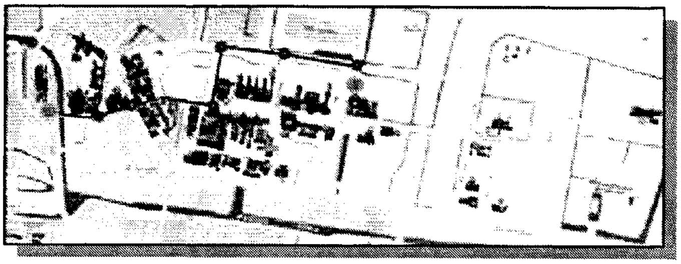
them will be graphically presented on the *lap 1* route map (see Figure 10. 35). If the user wants to analyze, for example, a sub-route from the presented graphic results, he or she has to click on it and the new sub-route info window will appear. If the user wants to obtain information, for example on nodes or landmarks, he or she has to click on it and the new info window related to it will appear.

Figure 10.35
The selected lap 1 with the route analysis - *route characteristics*

An additional possibility allows the user to define completely new route characteristics with which he or she would like to test the new route between two objects on *lap 1*. The user has to select one of the urban characteristics from the presented nodes, landmarks, good overview or predictable route, all shown as a pop-up menu in the new route according to net route characteristics sub-window. The user can also enter some other urban characteristics that are not on the list. If the user selects, for example, nodes, the Design Tool will perform analyses on an extended area belonging to the two objects of *lap 1,* according to new selected route characteristics. Then the Design Tool will decide, compose and propose the new routes between these two objects, which will be presented in the proposed new route window, shown in Figure 10.36. The user can then manipulate the proposed routes and 'interact' with the Design Tool, until a new acceptable solution is found.

With *test case 4* the testing of the *conceptual model* of the Design Tool is complete. The following examination of the criteria of performance usability will show the results of testing in such a way. Later on the applicability of the conceptual model will be explained.

Figure 10.36
The selected lap 1 with the new proposed route related to the *nodes*



Criteria of acceptance and usability heuristics

As mentioned earlier, the five criteria of acceptance of the Design Tool are *validity*, *effectiveness*, *efficiency*, *reliability* and *robustness*. In the light of the test simulation of the *conceptual model* of the Design Tool, we have to draw a conclusion about the development of the Design Tool with respect to these five criteria. The testing scale was limited but nevertheless there were certain achievements in these criteria.

Validity

The test case simulations examined above demonstrate that the structure of the *conceptual model* of the Design Tool has shown that the technique has been successfully applied to analyze particular urban environments with different user tasks. The test results according to *validity* are satisfactory, the analyses of the environmental tasks, which can help the user in a further design process or decision-making, are also satisfactory. A general conclusion can be deduced from the test case simulation: the *conceptual model* of the proposed Design Tool, according to the criteria of *validity*, fully and satisfactorily meets the criteria.

Effectiveness

The *conceptual model* of the Design Tool has been decomposed during the test cases simulation into its different functional parts, each one of which demonstrating full effectiveness. The results also demonstrate good graphic and textual [factual] descriptions, which might be reviewed, rearranged or modified effectively. The process of testing also demonstrates that the different processes fully relate and interact together and that the object-oriented database plays a crucial role in supporting the process of analysis. The general conclusion deduced from the test case simulation related to *effectiveness* is that each of the steps of functionality of the proposed Design Tool's *conceptual model* is fully satisfactory.

Efficiency

In the test case simulation of the new *conceptual model* of the Design Tool we demonstrated the performance of different analyses of the urban environment. In the test case we applied a limited number of the four objects of the case and presented only a limited number of the Design Tool possibilities. We did not increase the cost or duration of the testing process because we applied both simple analyses and a limited amount of requisite data. We believe that the Design Tool will perform equally efficiently in the full design process. The general conclusion based on *efficiency* is that the tested *conceptual model* of the proposed Design Tool demonstrates adequate results.

Reliability

The test case simulations were selected to apply the *conceptual model* of Design Tool to different simulated situations close to real designers' needs and to present the new test method developed in this research. The test demonstrates that the *reliability* of the *conceptual model* is high and that a designer can use it for different tasks in varying circumstances. The current testing method, which is under the constraints described above, is also highly applicable to such circumstances. The general conclusion of the test case simulation of the *conceptual model* of the proposed Design Tool, according to *reliability* criteria, is again that it is fully satisfactory.

Robustness

The testing of the *conceptual model* of Design Tool demonstrates that the proposed concept of the tool could be easily adapted for analysis using different approaches other than the cognitive mapping applied in this research. The Design Tool can also be easily applied in the field of landscape architecture, traffic or some other field. The structure of the proposed Design Tool is an open platform with a modular structure, offering possibilities of easy adaptation to other needs. The test simulation also demonstrates that the proposed Design Tool can be an additional support system for analysis or decision making of different specific tasks in relation to the urban environment. The general conclusion derived from the test case simulation of the *conceptual model* of the proposed Design Tool pertaining to *robustness* criteria is once again that it is fully satisfactory.

The usability *goals*, which have also been called *usability heuristics*, have been indirectly tested. During the testing process, the interaction between the user and the simulated Design Tool, via natural and intuitive interfaces, showed a high fulfillment of the *usability heuristics*. Some of them, e.g. *error presentation*, have not been tested because they cannot be simulated by prediction. The full results of the *usability heuristics* will possibly be obtained after some future testing, when the Design Tool application begins to live in urban design offices.

The tabular overview of the results of the acceptance criteria are presented in the matrix below (see table 10.2). The sign + means a fully satisfactory result and the sign +/- has the meaning of an adequate result.

Table 10.2
The results of the acceptance criteria

results OVERVIEW	Test case 1	Test case 2	Test case 3		Test case 4
			A	B	
Validity	+	+	+	+	+
Effectiveness	+	+	+	+	+
Efficiency	+	+/−	+/−	+/−	+/−
Rliability	+	+	+	+	+
Robustnes	+	+	+	+	+

Evaluation of the tested *conceptual model* of the Design Tool

The presented testing model and test simulation of the object cases also constituted an overview of the proposed *conceptual model* of the Design Tool. Here we can conclude some important aspects of the *conceptual model* of the Design Tool.

The general conclusion, derived after the test case simulation of the proposed *conceptual model*, demonstrates the following:

(1) that *universality* can make the Design Tool useful as an additional computer program for professionals who are dealing with urban space from different perspectives, such as architects, urban planners or urban designers;

(2) that *completeness* and *consistency* can make the Design Tool applicable for analysis of the visual quality of different urban environments using different approaches, such as cognitive mapping, way-finding, navigation and others;

(3) that *adaptability* can make the Design Tool available to professionals who are dealing with specific aspects of urban space, such as landscape architects, traffic engineers, and others;

(4) that *openness* can make the Design Tool accessible for remote collaboration via the Internet by sharing different remote databases, such as GIS data, statistical data, and others; and

(5) that *flexibility* can make the Design Tool fully collaborative with other computer applications used by professionals in their offices, such as GIS applications, CAD applications, and others.

These five main concluded aspects derived after the testing of the *conceptual model* indicate the *generalizability* of the proposed Design Tool. The Design Tool will encounter different professional problems related to the visual quality of the urban environment, not just according to the cognitive mapping approach. Here we also demonstrate that the Design Tool has the potential to offer much more to professionals.

We will now examine the general applicability of the *conceptual model* of the Design Tool, its limitations and the improvements that have to be made.

Applicability of the *conceptual model*

The *conceptual model* was developed in this research which can lead to the development of the Design Tool application. The examination of the general applicability of the *conceptual model* started with the testing of cases, presented earlier in this chapter. The *conceptual model* was located, as already mentioned, in a contemporary design office. The potential users of the Design Tool are professionals who fully understand the urban design process and the position of the application of the Design Tool in their practice. The testing of the *conceptual model* was focused on human-cognitive factors in direct testing of the intuitiveness of the modeled interfaces. The natural interaction between the user and the simulated Design Tool shows that the cognitive activities are based on a visual and linguistic approach. Graphic representation follows the linguistic [textual] indications and represents them. Control of the process is based on 1) the user's knowledge and 2) machine reasoning, both based on human-cognitive factors.

The selected test cases are simplifications of projects that the user may encounter in practice. The reason for this is not only in relation to efficiency criteria but also because simple cases indicate possibilities and enrich the testing process to make it more comprehensive. The main testing process focused on whether the *conceptual model* can successfully analyze different cognitive mapping and way-finding tasks according to the main requirements of the Design Tool, presented in Chapter 1. The results are textual descriptions and graphic representations as results of the computational processing [data processing] and the user's interaction [controlling]. The results of testing can serve as an important and useful reference for developing the Design Tool application.

The test case did not simulate an analysis on a large urban scale and analyses according to it e.g. testing different districts or sub-districts of the large-scale urban environment, urban density or urban patterns. There are a few reasons for these limitations. The main reason is the limitation of the scope of this research. The second is that these tests will need more effort and time in data gathering and analysis. This limitation of the test cases should be corrected in an extension of this research and also in some further testing.

Another limitation is testing the *conceptual model* with the use of remote databases via the Internet. A good justification for this limitation is the minor position of this additional part in the *conceptual model* and because the simulation cannot be appropriate without a real connection to the web. This also has to be corrected with some further testing.

Possibly a general test limitation is also the way in which we simulated the process, without repeating and showing all steps in the process and without simply presenting the functions of each module of the *conceptual model.* We chose the shortest variant for reasons of space limitation of the research, since otherwise this chapter would be out of proportion to all others and the focus of the research would be on testing. We hope that these reasons are understandable.

Further tests

The expected extension of the *conceptual model* of the Design Tool, which has been developed in this research, can be the Design Tool application. The Design Tool application can be a contribution to the urban design office in the coming years. Then the new *ideal* test of the Design Tool application will be the adoption by the professionals and their close observation. Further tests, in the circumstances of the Design Tool application, can extend the test model that we proposed in this research. The full set of usability testing, interface testing and system software testing will be the normal status of testing. Furthermore, when the Design Tool starts to live its working cycle in a design office, the social aspect of the Design Tool will be tested as well. Such a test would confirm the validity and efficiency of the Design Tool, which has been proposed by this research.

Conclusions

In this chapter we proposed a new testing model of the *conceptual model* of the Design Tool, which extends the usability model [ISO9241 standard] and implies acceptance criteria. The testing model consists of four parts: *goals, context of use, conceptual model with the objects of the cases* and *criteria of* *acceptance*. Each of them is explained and demonstrated later within the test case simulation.

The four objects of the test cases were then simulated. In these test cases only the main characteristics of the developed *conceptual model* of the Design Tool were applied. The test case is also limited because we tested only a couple of different urban situations. Through the test we showed how the *conceptual model* of the Design Tool is applicable to simulated-settings situations.

The test cases examine the fulfillment of the proposed *conceptual model* of the Design Tool. The results of the test are mainly satisfactory, in the light of the given limitations which have to be extended in further testing. A future test should be more adequate because the test will be on the full Design Tool application in a real-settings situation.

At the end of this chapter we evaluated the tested *conceptual model* of the Design Tool and we gave some important concluded aspects, such as *universality, completeness* and *consistency, adaptability, openness* and *flexibility*. We also examined the general applicability of the *conceptual model* of the Design Tool, its limitations and the improvements that have to be made.

In the next chapter, the last of this research, we will make conclusions by generating the tool. We will also reflect on the research assumptions. Then we will conclude the results and outline some contributions. At the end of the next chapter we will indicate the way forward for further research in this field.

CHAPTER 11

CONCLUSIONS

Concluding remarks

In this chapter we will give concluding remarks and overview on the research. This research has set out the development of the *conceptual model* of a Design Tool that could help professionals in urban design practice. This will be facilitated by helping them in analyzing and checking the visual quality of the urban environment through the image scene. For this the cognitive mapping approach, based on Lynch's theory is used. This research has generated and integrated some methods and techniques for structuring and composing a new Design Tool.

Reflection on assumptions

The research assumptions presented in Chapter 1.3 were examined according to the given frameworks, conceptual modeling and controlling by this research. We can draw some conclusions in terms of a reflection on the research assumptions.

The increasing quality and efficiency of the urban design process by the proposed *conceptual model* of the Design Tool became evident through testing. The *conceptual model* of the Design Tool would be aimed at performing different analyses and controlling the visual quality of the urban environment through the image scene according to the cognitive mapping approach. The testing of the *conceptual model* showed that the simulated tool works instrumentally well and satisfactorily. The *conceptual model* of the tool and database behind it evidently work properly and they will prove beneficial as they are used with time.

After testing and measuring the acceptance criteria, the applicability of the Design Tool is also evident. For certain tasks in cognitive mapping testing, the Design Tool's performance cannot be compared to some other existing tools because they do not have these capabilities. The proposed Design Tool improves design practice with the cognitive mapping approach and by taking data directly from the image scene.

Results and contributions of the research

Here we will refer to the results and contribution of the research according to each chapter.

In Chapter 2 we reviewed some of the theoretical approaches in the field of the visual quality of the urban environment and its aesthetics and related computer applications. The main theory that was adopted in this research was the theory of 'urban form' and the concept of 'cognitive mapping' developed by Kevin Lynch. Some other researchers developed Lynch's ideas further in different fields of scientific interest, such as Nasar's 'evaluative image of the city', which is also elaborated in the chapter. Lynch's *five elements* of the urban environment, which build the imageability of the city, played an important role in creating the "WayMaker" computer application. The second application reviewed was the "web-based mapping and survey tool". Both applications gave important information about computational design tools in the field of our research interest. In this chapter we also reviewed studies in the domain of aesthetics and controlling urban design.
The important result of this chapter was that the review of the literature and 'research applications' showed us that there are no computer applications able to analyze the visual quality of the urban environment by cognitive mapping precedents. Nothing was found relating to this particular subject. The wide literature investigation showed that the main topic of this research is in many respects new in its approach.

In Chapter 3 we investigated theoretical knowledge relating to urban design. There we examined urban design theory from two perspectives: the hierarchical structure of urban elements and the concept of cognitive maps, which constituted the main theoretical platform for the further development of the proposed Design Tool. These two aspects of urban design theory were developed by Kevin Lynch in his 'theory of urban form'. We explained four elements that belong to the visual quality of the city: *legibility, building the image* [*image*], *structure and identity* [*identity*], and *imageability*. We also briefly explained the five elements of the urban environment: *paths, edges, districts, nodes* and *landmarks*. Both sets of elements would be crucial for the subsequent development of the object-oriented database model. The graphic notational system, which Lynch proposed in his theory, was also explained in the chapter. A section of the chapter was dedicated to post-Lynchian thinking and prior research on the topic. There we elaborated different theoretical approaches related to cognition, spatial representation, cognitive mapping, modeling and some other theories, all appearing after the publication of Lynch's theory. This chapter represented the results of an examination of theories that could be useful for the Design Tool development.
In this chapter we put forward some criticisms of Lynch's theory and also made some suggestions for modification during the development of the Design Tool.

Chapter 4 was focused on cognitive science. We elaborated two theories, one of which was Marr's theory of *computer vision*, while the other was Ullman's theory of *high-level vision*. Both theories gave us theoretical as well as practical approaches to human and computer vision. Vision entails recovering the structure and properties of the visual world. In this chapter we focused on the science of computational vision, the primary goal being the elucidation of fundamental concepts. Issues of algorithms and implementation and system architecture received little or none of our attention.
The contribution of this chapter was that we adopted Ullman's theory and method of object recognition as a plug-in module of the *conceptual model* of our Design Tool.

In Chapter 5 we presented the basic ideas and overview of *database technology*. In the chapter we elaborated the main concepts such as *data, the universe of discourse* and generic database properties, then entities, attributes and relationships as well as instances, models and schemas with their relationships. We also introduced the main database structure, focusing on three-schema architecture as well as database management systems [DBMS's] with all their characteristics, functions, uses, subsystems and criteria for classification. The generations of database models with a historical overview was also presented as well as the main characteristics of file systems, hierarchical systems, network systems and relational systems. We also discussed the differences between them. Object-oriented database systems were introduced with the basic principle of the Unified Modeling Language [UML] methodology and its characteristics. The structure of object-oriented database systems was divided into the main elements: objects, classes and relations with all their subelements, characteristics and relations. We also elaborated the object-oriented database design . The object-oriented database is the main computational support for our Design Tool.

The contribution of this chapter concerns Perspective-DB technology, which was adopted for developing the object-oriented database model of our Design Tool. Perspective-DB technology provides the possibilities to process a complex urban structure and its cognition.

In Chapter 6 we explored Lynch's 'theory of urban form' which contains universal categories, characteristics and elements, psychological as well as physical, of the urban environment which are applied to the development of the *database model* and *conceptual model* of the Design Tool. The framework for applying Lynch's theory to the Design Tool was introduced. The elements of urban form were systematized and the four psychological elements related to the visual quality of an urban environment were analyzed. We also analyzed the five physical elements of the urban environment and their hierarchical order. There we also set out the generic schematical typology of the relationships between the five main elements.
The results of this chapter were universal categories, characteristics and elements of the urban environment, and a precise systematization of them in a form that could be comprehensible and applicable for designing the *database model* and *conceptual model* of the Design Tool. This chapter also showed how to apply Lynch's theory to the Design Tool and gave the sets of requirements that derived from the theory.

In Chapter 7 we explained the main concept of the Computer Cognition Model related to our Design Tool. There we explained the ideas concerning general modeling in cognitive science, where computer cognition plays a significant role in the simulation of some basic cognitive and behavioral processes.

The main result of the chapter was the development and explanation of the *Computer Cognition Model* through its main functions, characteristics and elements. The Computer Cognition Model consists of three modules: *recognition* (adopted techniques from 'object recognition'), the *meta level* or *filtering* (the module in which 'microunits' create the '*meta* image') and *synthesis* (the module where each of the objects from the scene, called 'global units', assumes corresponding data from the database). In this chapter we also gave the set of tool requirements derived from the *Computer Cognition Model.*

In Chapter 8 we developed the object-oriented database model for our Design Tool. The object-oriented database model is based on the hierarchical structure of the elements of the urban environment and its cognition. The database model explored in this chapter overlaps the urban structure in all details. The complex object-oriented database model, explored using Perspective-DB technology, was presented in two parts, and broken down into several schemas. The first part presented the object types and their attributes and relations. The second part presented the behavior by their operations. The database model for the proposed Design Tool, developed through Perspective-DB technology, provides the means to present and process the complex structure of any urban environment and its cognition within a multi-media presentation of application objects in their semantical context, and to exploit them within this context.
In this chapter we also outlined the set of main requirements derived from the object-oriented database model.

In Chapter 9 we explained three things: frameworks of the research, and the conceptual and functional models of the Design Tool. Structural and functional analysis of both the conceptual and functional models we also given. The theoretical background of the research had a logical influence on the 'theoretical framework', which was composed of 'urban design theory', 'cognitive theories' and 'computational theories'. The specificity, mostly functional, of these theories influenced the development of the 'system framework'. The 'system framework' consists of several subframes: *recognition, filtering, synthesis, evaluation* and *creation*, which defined the main operations of the tool. The *conceptual model* describes the logic and structure of the Design Tool. The *conceptual model* also describes the structure of interactions and functions of different elements of the model. The main elements are: *input, recognition, meta level, synthesis, evaluation, creation* and *output*. All elements and their internal organization were explained. The 'functional model' structurally follows the *conceptual model*

elements by implementing a new element: the 'OODB system'. The 'functional model' explain the role and functional position and relations of the object-oriented database system [OODB] and the Design Tool.
The *conceptual model* and *functional model* form an important platform from which a realistic and useful Design Tool will be developed [by programming] for design practice.

In Chapter 10 we proposed a new testing model for the *conceptual model* of the Design Tool, which extended the usability model [ISO9241 standard] and implied acceptance criteria. The testing model consisted of four parts: *goals, context of use, conceptual model with the objects of the cases* and *criteria of acceptance.* Each of them was explained and demonstrated within the test case simulation. The four objects of the test cases were simulated. In these test cases only the main characteristics of the developed *conceptual model* of the Design Tool were applied. Through the test we showed how the *conceptual model* of the Design Tool is applicable to simulated-settings situations. The test cases examined the fulfillment of the proposed *conceptual model* of the Design Tool. The results of the test were mainly satisfactory, in the light of the given limitations which have to be overcome in further testing. A future test should be more comprehensive because the test will be on the full Design Tool application in a real-settings situation.
At the end of this chapter we evaluated the tested *conceptual model* of the Design Tool and made some important concluding remarks on aspects such as *universality, completeness* and *consistency, adaptability, openness* and *flexibility*. We also examined the general applicability of the *conceptual model* of the Design Tool, its limitations and the improvements that have to be made.

The novelty of the research

The novelty of this research will be seen in comparison with the research analyzed and the computer application reviewed in Chapter 1. These applications, which are developed for urban design practice, have not been developed using the cognitive mapping approach or for the purposes of analyzing and controlling visual quality through an image scene. From these aspects our research became unique and highly original.

There are three main novelties and results derived from this research:

(1) the cognitive mapping approach was implemented on the *conceptual model* of the Design Tool;
(2) the OODB structure (data model) overlaps the urban elements and the spatial hierarchy of the urban environment; and
(3) all available data from the stored database belonging to objects from an image scene can be directly taken and manipulated.

In our research, the taking of data directly from an image scene is only conceptually presented and indicated, because it is not the priority of this research.

Advantages and disadvantages of the research

The advantages of this research can be divided into two groups, whereby one represents the *theoretical* advantages, while the other represents the *applicable* advantages.

Theoretical advantages can be the following:
(1) this research provided a *theoretical model* for developing a tool for urban designers, which is based on the cognitive mapping approach;
(2) this research *integrated different scientific theories and methods* in its theoretical method, derived from theories of design, cognitive science and computational techniques;
(3) this research provided a theoretical *systematization* and *hierarchy* of the psychological and physical elements of the urban environment, which build the visual quality of the urban environment, from Lynch's theory of urban form;
(4) this research introduced the *Computer Cognition Model* that provided the concepts of 'filtering' by microunits and 'synthesis' by global units, based on an integration of design theory and cognitive science;
(5) this research introduced the *object-oriented database model*, which is based on the hierarchical structure of the urban environment, to support the conceptual model of the Design Tool;
(6) this research proposed a *conceptual* and *functional model* of the Design Tool, integrating related theories and methods and some computational techniques; and

(7) this research introduced a *testing model*, integrating some existing testing methods and criteria, integrated specially for this test case simulation of the conceptual model of the Design Tool.

Applicable advantages can be the following:

(1) the Design Tool could be built as a *computer application*, according to aspects of some of the proposed theoretical models such as the conceptual model of the Design Tool and the object-oriented database model;

(2) after the testing of the conceptual model of the Design Tool, we have proved that this tool will *improve* the process of controlling and analyzing the visual quality of the urban environment, according to cognitive mapping, way-finding and navigation in design practice; and

(3) in terms of implementing the proposed Design Tool as a computer application in design practice, some *social* and *financial effect* could be predicted.

Some of the main *applicable* disadvantages of this research can be the following:

(1) to build the Design Tool as an computer application a following *deep investigation* and *additional research* have to be performed in the field of the Computer Cognition Model, the computation, visualization and presentation, which were only conceptually introduced in this research;

(2) *testing* of the Design Tool, with all the aspects of different criteria, has to be done after completing the Design Tool computer application, and before applying it in design offices; and

(3) the social and financial *acceptability* of the Design Tool can be only effectively implemented after the usage of the Design Tool computer application in urban design practice.

Overview of the research

The developed conceptual model of the Design Tool provides both a scientific and practical contribution. The scientific contribution is provided by means of integrational thinking and the execution of different theoretical approaches in the conceptual tool developing. During the examination of the research, the gap between purely theoretical approaches and applied research was bridged. The systematic and schematical overview of the research process is presented in the identification diagram (Figure 11.1).

The identification diagram consists of three main branches, which are the parts of the theoretical framework.

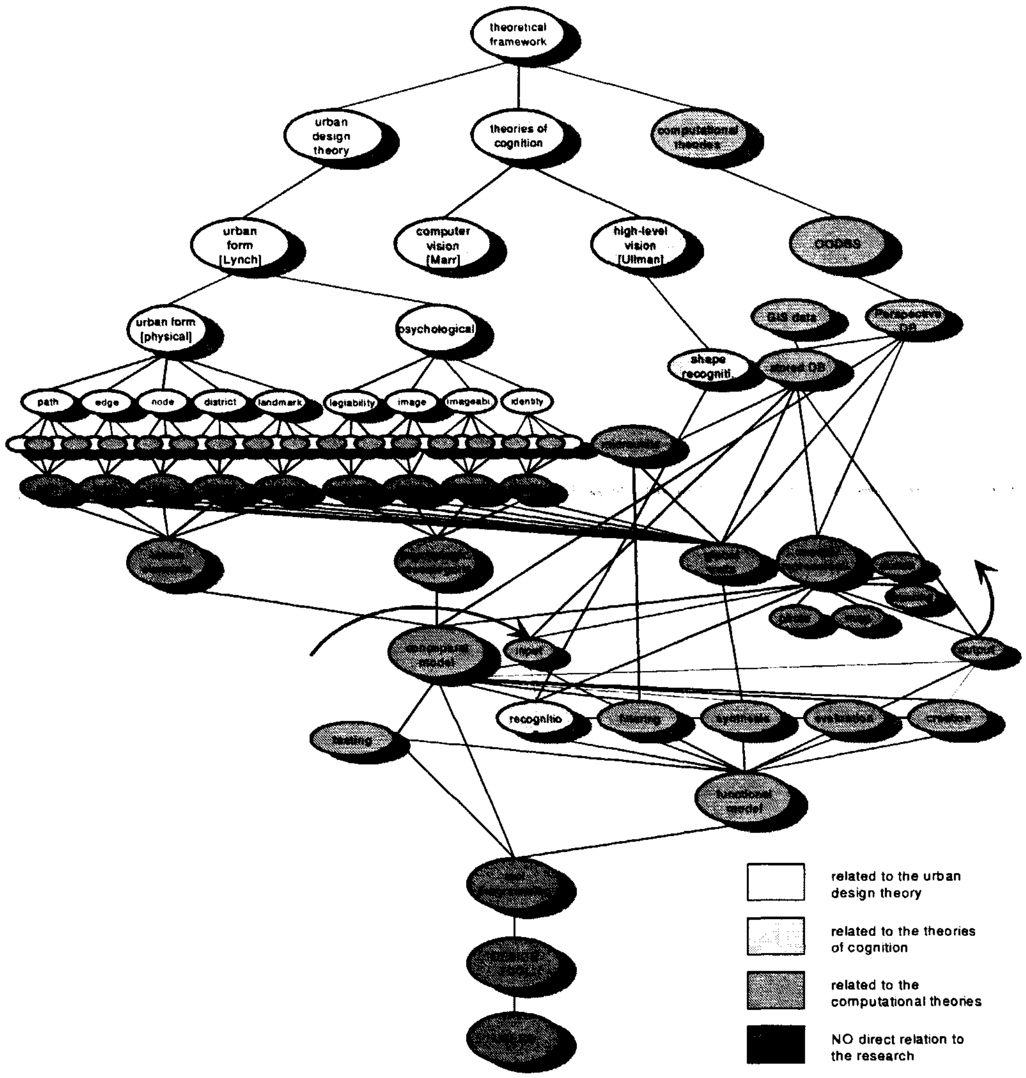
The first branch belongs to *urban design theory* and shows Lynch's theory of *urban form* with its physical ('urban form' and hierarchy of urban environment) and psychological ('cognitive mapping') characteristics. Each of them is subdivided into main elements. Each of the elements is also subdivided into its elements and so on. They are only symbolically indicated on the identification diagram without naming all of them; they are all presented in the OODB model.

The second branch belongs to the *theories of cognition* and shows two theoretical approaches: one is Marr's theory of 'computer vision' and the second is Ullman's 'high-level vision' theory. Ullman's theory also provides the method of shape recognition, which we adopted as a 'plug-in' application for the conceptual model of the Design Tool.

The third branch belongs to *computational theories*, from which we decided to use the object-oriented database (OODB). This 'computational' branch is more dominant than the other two and it brings the research from the theoretical to the applied. In this branch the positions and relations of different elements and parts of the research are indicated such as stored database, model representation, psychological precategorization or urban elements. 'Urban elements' and 'psychological precategorization' are a part of the database system which are equivalent to 'urban form' and 'cognitive mapping' from the theoretical part of the *urban form* theory. The subdivided parts of their structures are the same as in theory and they appear mirrored, which is what they are. They completely overlap with the elements and hierarchical structure of the urban environment. In the computational branch, the *conceptual model* of the Design Tool is also presented with its modules: input, recognition, filtering, synthesis, evaluation, creation and output. The functional model and testing are also presented in this branch.

A fourth part, not a branch, can be the logical continuity of the research, which should be realized in the future, and is not directly related to this research. This is the tool programming (coding) and constructing the Design Tool as a final product (computer application). At the end of this chain the *user* logically stays as the consumer of the product of this research and the Design Tool application produced in the future.

Figure 11.1
The identification diagram of the research



All related parts, from theoretical to applied, are connected between each other and indicate more global rather than real relations. The real relations are more complex in the research than in the identification diagram, which is merely a basic schematical overview of the complexity of the research. The input and output of the Design Tool are also indicated symbolically and without any spatial logic of the user's position.

Further research

Further research can be subdivided into two levels 1) new scientific research and 2) improving the presented research in itself.

At the scientific level of further research in the field of urban design, there can be an investigation of the possible influence of neurological and brain science as well as computer vision. Neuro science and computer vision are increasingly developing new theories and techniques of perceiving, memorizing and retrieving objects from the surrounding environment, related to computer cognition, and some of these approaches could be useful for developing a new Design Tool for urban designers and planners. Theories of design, as well as theories of urban design and planning, are also becoming increasingly innovative with new approaches and uses of new technologies. The integration of different scientific disciplines, all based on computer technologies, offers tool designers many promising possibilities, especially in the domain of conceptual research. From some aspects, the research presented here, which also explored the domain of cognitive science, might provide some inspiration for some further improvements.

The research presented here was restricted to the development of the *conceptual model* of the Design Tool, based on Lynch's theory. This was done in the hope of examining theoretical possibilities to infer a cognitive mapping approach in urban design practice through a computer-based Design Tool. A second hope was to develop a method for directly taking all related data from a selected object from the image scene. Another hope was to implement OODB technology, with all the great possibilities that it offers, in the *conceptual model* of the Design Tool, which can also support the second hope. Therefore, this research provides techniques and methods to analyze images of the urban environment and to extract shape elements and features (microunits and global units) and to represent and store them in an object-oriented database.

The next step in the process of the tool realization will be coding. The programming of the tool will need certain investment, time and a group of experts in different programming languages with the necessary capabilities, skills and conditions. Before programming, some parts of the research have to be investigated deeply because we only presented them conceptually. As we mentioned before, the focus of the research was on developing the *conceptual model* of the Design Tool and developing the generic model of the OODB system for it. All other parts (modules) of the *conceptual model* are more or less only conceptually developed. One possible area for deeper research can be in the domain of filtering and synthesis, where we outlined the general concept of the functionality of 'microunits' and 'global units'. Another area for further research can be in the domain of the decision-support system – only the existence of which we indicated – as a subsystem of the evaluation module of the *conceptual model* of the Design Tool. The computation, visualization and presentation also have to be developed. The important problem which has to be solved in the further research are algorithms which can be able to calculate a quantitative assessment of the visual and functional quality of the urban environment, based on Lynch's theory. These new, further research will use from this research derived elementary shape characteristic. Some smaller developments can also be made such as data coding, a self-learning protocol, an Internet protocol and some others. This is all normal creative work in software development.

Another possibility for improving the presented research can be in the domain of the objects of the cases. For the testing of our research we used the four specific objects of one case. But more case objects can also serve more purposes in demonstrating the research results. We also chose one case study area, the campus of TU Delft, but with other different case study areas the testing of the tool can show the wider applicability of the tool in different urban areas in urban design practice.

SUMMARY

Thesis abstract

In current architectural, urban design and planning practice, the role of new information technology has become important. The use of computer models has been going on in architectural design and urban planning since the 1950s. Early architectural drafting was transformed through computer-aided design (CAD) packages, such as the now widely used AutoCAD. In urban planning, computation began with municipal information systems and landuse transportation modelling. In the last decade there have been major developments in tools for visualization and representation, particularly through the development of geographical information systems (GIS). With the development of GIS information technology, it has become possible to link up with available 2D or 3D spatial data through new desktop packages such as ArcInfo, ArcView or MapInfo, which have become standard. On the other hand, urban design was for many years untouched by computational developments. Now it is set to slowly change.

This multidisciplinary research is focused on improving design methods and theory by developing new knowledge about designing as a cognitive process embedded in urban design practice. The main scope and the starting point of the research is Lynch's theory of 'urban form' and his concept of *cognitive mapping*. This research proposes a cognitive approach that can be integrated into computer processing and involves the integration of knowledge from design theory, cognitive science and the use of computational techniques. With the support of a theoretical platform, this integration leads to a *conceptual model* of the Design Tool.

The goal of this research is to develop a *conceptual model* of a Design Tool that will help in analysing, evaluating and controlling the visual quality of the urban environment in urban design practice. The *conceptual model* of the Design Tool is based on the idea of cognitive mapping. The computational aspect of it is provided by an object-oriented database system (OODB). The *conceptual model* of the Design Tool provides the knowledge that will support the application level of the Design Tool.

The research method 1) includes an examination of the theoretical background in the field of research interest, 2) provides different computational techniques which can support the development of the *conceptual model* of the Design Tool, and 3) proposes a new testing method conducted through the simulation of case study objects.

There are two main results of this research: 1) the *conceptual model* of the Design Tool, which provides a clear description of how the Design Tool should be processed in each of the different stages, and 2) the object-oriented database model, which provides the main computational support for the Design Tool. In the research some other functional solutions are indicated, such as *synthesis* or *evaluation*, but full attention can be focused on them in some further research.

This research provides new theoretical insights into the structure and the process of analysing, evaluating and controlling the visual quality of the urban environment, and a useful prescriptive model that can be applied in design practice. The implementation of the results of this research, as a practical software application available for commercial urban design exploitation, must be examined.

SAMENVATTING
In Dutch

In de huidige archtitectonische, stedebouwkundige en planologische praktijk is de rol van nieuwe informatietechnologie belangrijk geworden. Het gebruik van computermodellen heeft zich sinds de jaren 50 in architectonische ontwerpen en stedelijke planologie doorgezet. Het vroege architectonische ontwerpen transformeerde door computer-aided design (CAD) pakketten, zoals het nu op grote schaal gebruikte AutoCAD. In de planologie/stedebouwkunde begon het computergebruik met gemeentelijke informatiesystemen en vervoersmodellen voor landgebruik. In het laatste decennium hebben er ingrijpende veranderingen plaatsgevonden in de instrumenten voor visualisatie en representatie, met name door de ontwikkeling van geografische informatie systemen (GIS). Met de ontwikkeling van GIS informatietechnologie is het mogelijk geworden de beschikbare 2D en 3D ruimtelijke data aan elkaar te koppelen middels nieuwe desktop pakketten als ArcInfo, ArcView en MapInfo, die de standaard zijn geworden. Daar staat tegenover dat de stedebouwkunde jarenlang van ontwikkelingen op het gebied van computergebruik verstoken bleef. Nu begint zij langzaamaan te veranderen.

Dit multidisciplinair onderzoek is gericht op verrijkende ontwerpmethoden en theorieën door nieuwe kennis te ontwikkelen over ontwerpen zijnde een cognitief proces ingebed in de stedebouwkundige praktijk. Het belangrijkste terrein tevens uitgangspunt van het onderzoek is Lynch's theorie van 'urban form' en zijn concept van *cognitive mapping*. Dit onderzoek stelt een cognitieve benadering voor die in computerprocessen geïntegreerd kan worden en de integratie van kennis uit design theorien, cognitieve wetenschappen en het gebruik van computertechnieken met zich meebrengt. Met behulp van een theoretisch platform leidt deze integratie tot een *conceptual model* van het Design Tool.

Het doel van dit onderzoek is het ontwikkelen van een *conceptual model* van een Design Tool dat kan helpen bij het analyseren, evalueren en controleren van de visuele kwaliteit van de stedelijk omgeving in de stedebouwkundige praktijk. Het *conceptual model* van het Design Tool is

gebaseerd op het idee van cognitive mapping. Het computeraspect ervan wordt aangedragen door een objectgeoriënteerd databasesysteem (OODB). Het *conceptual model* van het Design Tool levert de kennis dat het applicatieniveau van het Design Tool ondersteunt.

De onderzoeksmethode 1) omvat een examinatie van de theoretische achtergrond op het terrein van het onderzoeksgebied 2) levert verschillende computertechnieken die de ontwikkeling van het *conceptual model* van het Design Tool kunnen ondersteunen en 3) stelt een nieuwe testmethode voor, geleid door de simulatie van case-study onderwerpen.

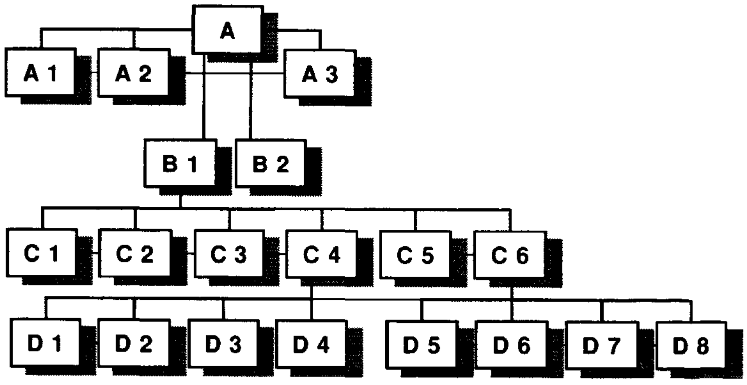
De twee belangrijkste resultaten van het onderzoek zijn: 1) het *conceptual model* van het Design Tool, dat een duidelijke beschrijving geeft van de wijze waarop het Design Tool in elk van de verschillende stadia verwerkt zou moeten worden en 2) het objectgeoriënteerde databasemodel, welke de belangrijkste computerondersteuning voor het Design Tool levert. In het onderzoek worden nog enkele andere functionele oplossingen aangegeven, zoals *synthesis* en *evaluation*, maar de volle aandacht kan daarop gericht worden in een nader onderzoek.

Dit onderzoek levert nieuwe theoretische inzichten in de structuur en het proces van analyseren, evalueren en controleren van de visuele kwaliteit van de stedelijke omgeving, en een bruikbaar prescriptief model dat in de ontwerppraktijk toegepast kan worden. De toepassing van de resultaten van dit onderzoek, als een voor commerciële stedebouwkundige exploitatie beschikbare, praktische software-applicatie, behoeft nader onderzoek.

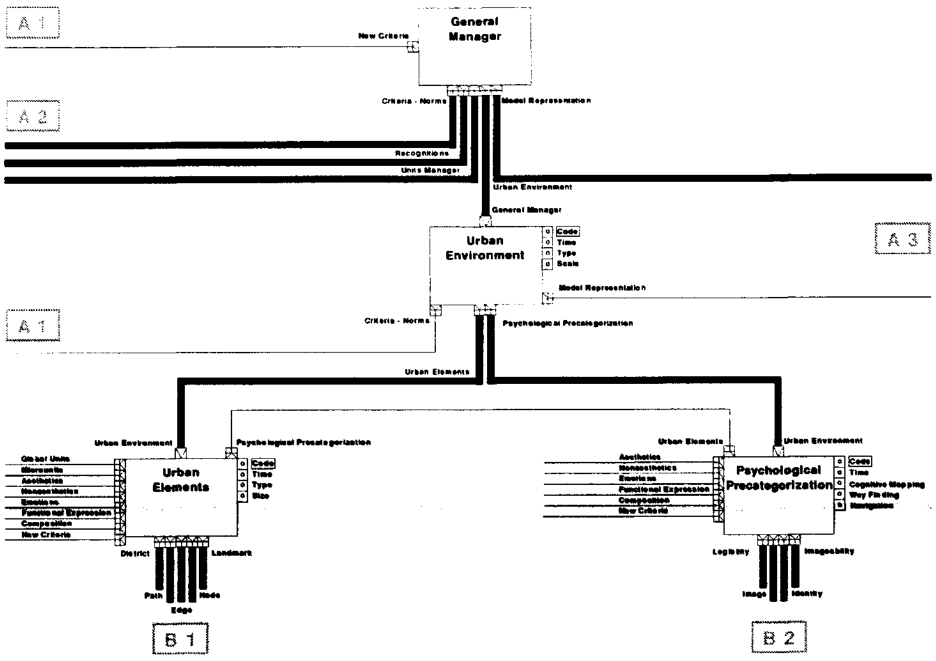
APPENDIX A

OBJECT-ORIENTED DATABASE MODEL

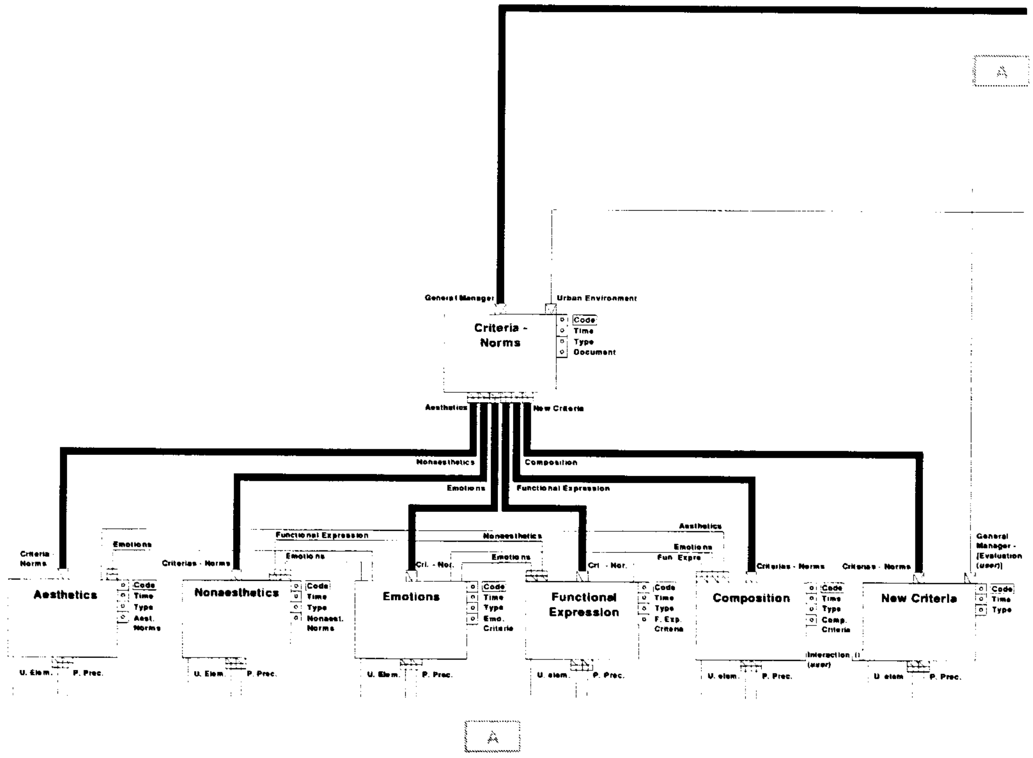
Map of the elements of the *object-oriented database model* and connections between them



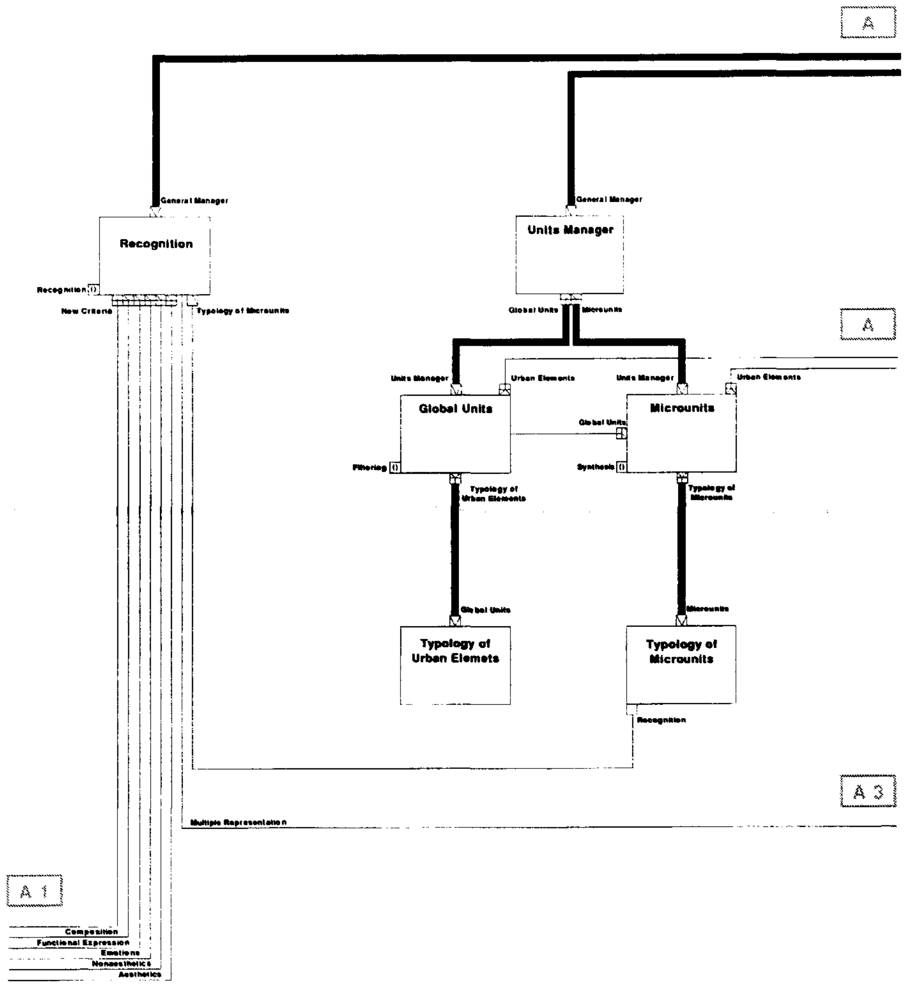
Element: A


A 1
General Manager
New Criteria
Criteria - Norms
Model Representation
A 2
Recognitions
Units Manager
Urban Environment
General Manager
Urban Environment
Code
Time
Type
Scale
A 3
Model Representation
A 1
Criteria - Norms
Psychological Precategorization
Urban Elements
Urban Environment
Psychological Precategorization
Global Units
Microunits
Aesthetics
Nonaesthetics
Emotions
Functional Expression
Composition
New Criteria
Urban Elements
Code
Time
Type
Size
District
Landmark
Path
Node
Edge
B 1
Urban Elements
Urban Environment
Aesthetics
Nonaesthetics
Emotions
Functional Expression
Composition
New Criteria
Psychological Precategorization
Code
Time
Cognitive Mapping
Way Finding
Navigation
Legibility
Imageability
Image
Identity
B 2

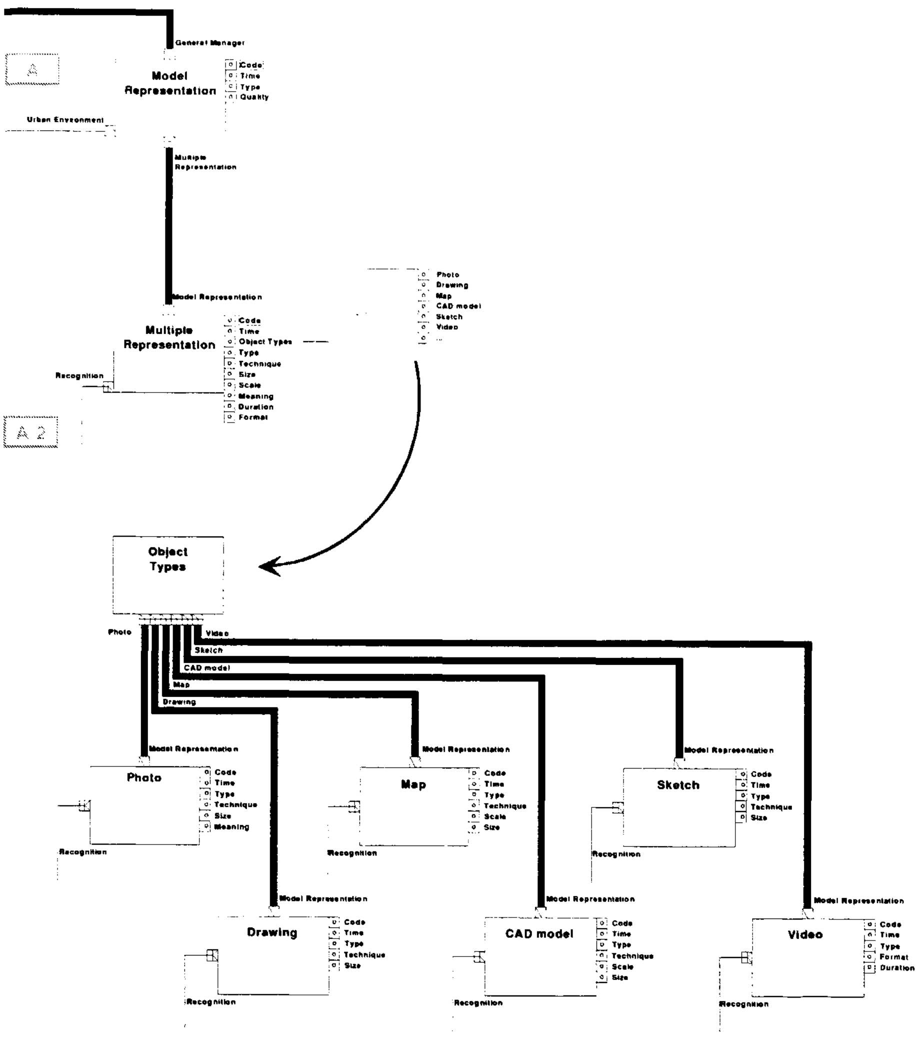
Element: A 1


A
General Manager
Urban Environment
Criteria - Norms
Code
Time
Type
Document
Aesthetics
New Criteria
Nonaesthetics
Composition
Emotions
Functional Expression
Criteria Norms
Emotions
Aesthetics
Code
Time
Type
Aest. Norms
U. Elem.
P. Prec.
Criterias - Norms
Functional Expression
Emotions
Nonaesthetics
Code
Time
Type
Nonaest. Norms
U. Elem.
P. Prec.
Nonaesthetics
Emotions
Cri. - Nor.
Emotions
Code
Time
Type
Emo. Criteria
U. Elem.
P. Prec.
Aesthetics
Emotions
Fun Expre
Cri - Nor.
Functional Expression
Code
Time
Type
F. Exp. Criteria
U. elem.
P. Prec.
Criterias - Norms
Composition
Code
Time
Type
Comp. Criteria
Interaction () (user)
U. elem.
P. Prec.
Criterias - Norms
General Manager - [Evaluation (user)]
New Criteria
Code
Time
Type
U. elem
P. Prec.
A

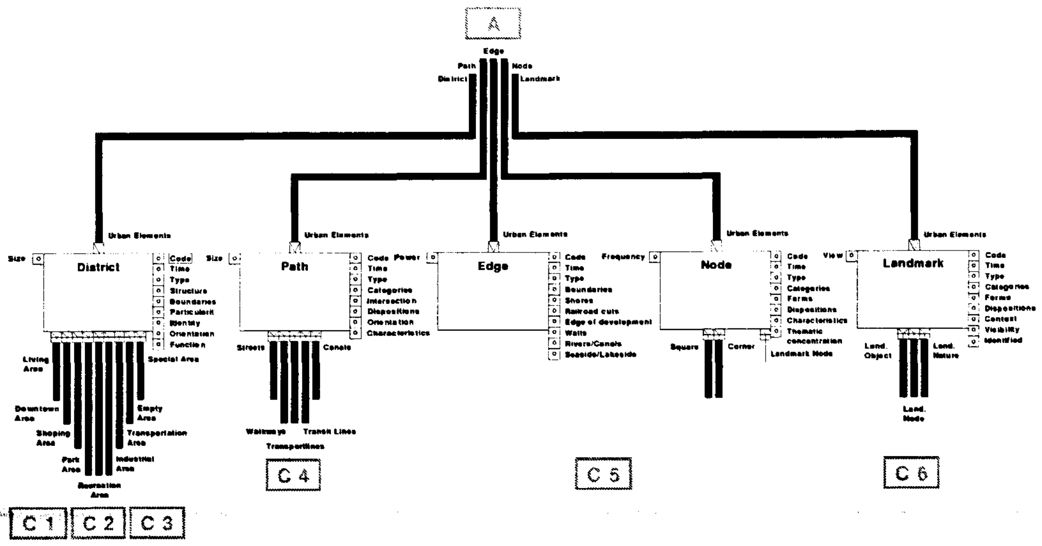
Element: A 2


A
General Manager
Recognition
Recognition ()
New Criteria
Typology of Microunits
General Manager
Units Manager
Global Units
Microunits
A
Units Manager
Urban Elements
Units Manager
Urban Elements
Global Units
Global Units
Microunits
Filtering ()
Synthesis ()
Typology of Urban Elements
Typology of Microunits
Global Units
Microunits
Typology of Urban Elemets
Typology of Microunits
Recognition
A 3
Multiple Representation
A 1
Composition
Functional Expression
Emotions
Nonaesthetics
Aesthetics

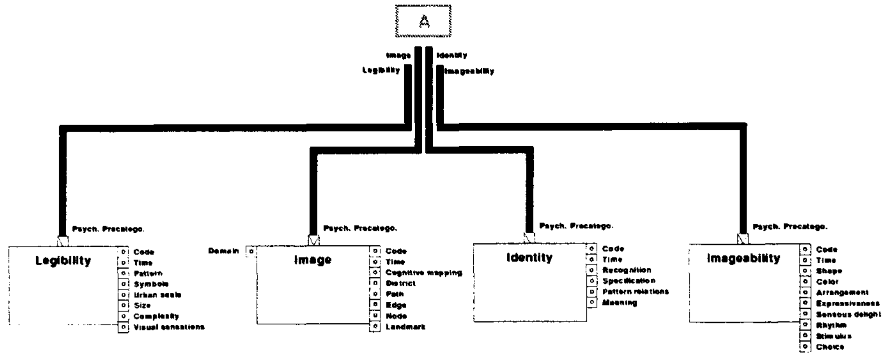
Element: A 3


General Manager
A
Model Representation
Code
Time
Type
Quality
Urban Environment
Multiple Representation
Photo
Drawing
Map
CAD model
Sketch
Video
Model Representation
Multiple Representation
Code
Time
Object Types
Type
Technique
Size
Scale
Meaning
Duration
Format
Recognition
A 2
Object Types
Photo
Video
Sketch
CAD model
Map
Drawing
Model Representation
Photo
Code
Time
Type
Technique
Size
Meaning
Recognition
Model Representation
Map
Code
Time
Type
Technique
Scale
Size
Recognition
Model Representation
Sketch
Code
Time
Type
Technique
Size
Recognition
Model Representation
Drawing
Code
Time
Type
Technique
Size
Recognition
Model Representation
CAD model
Code
Time
Type
Technique
Scale
Size
Recognition
Model Representation
Video
Code
Time
Type
Format
Duration
Recognition

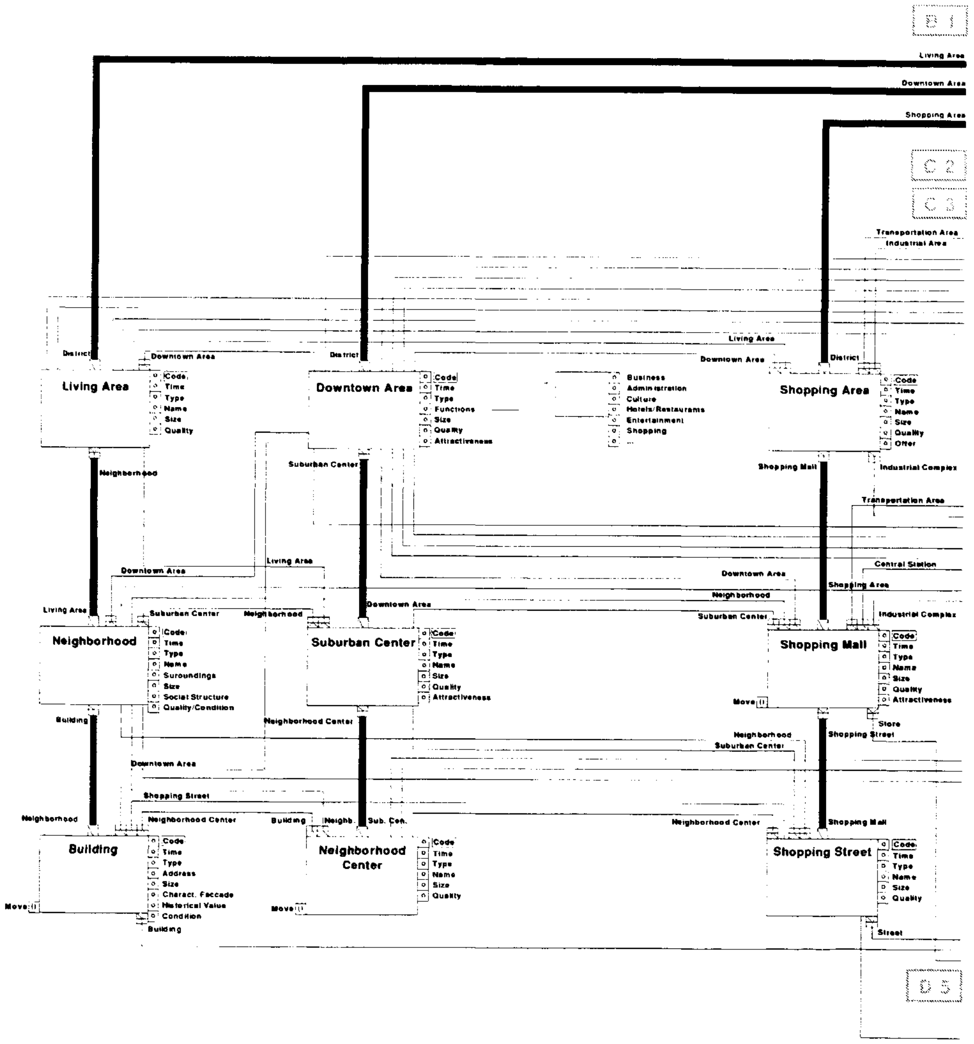
Element: B 1



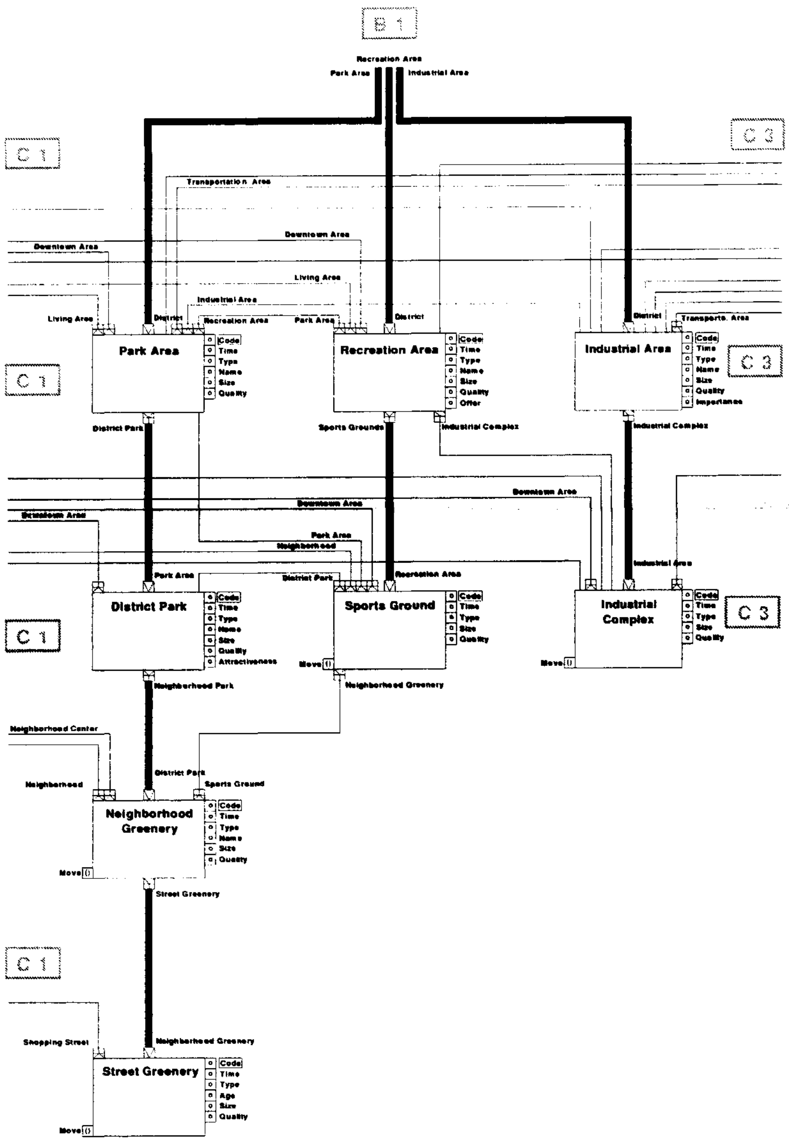
Element: B 2



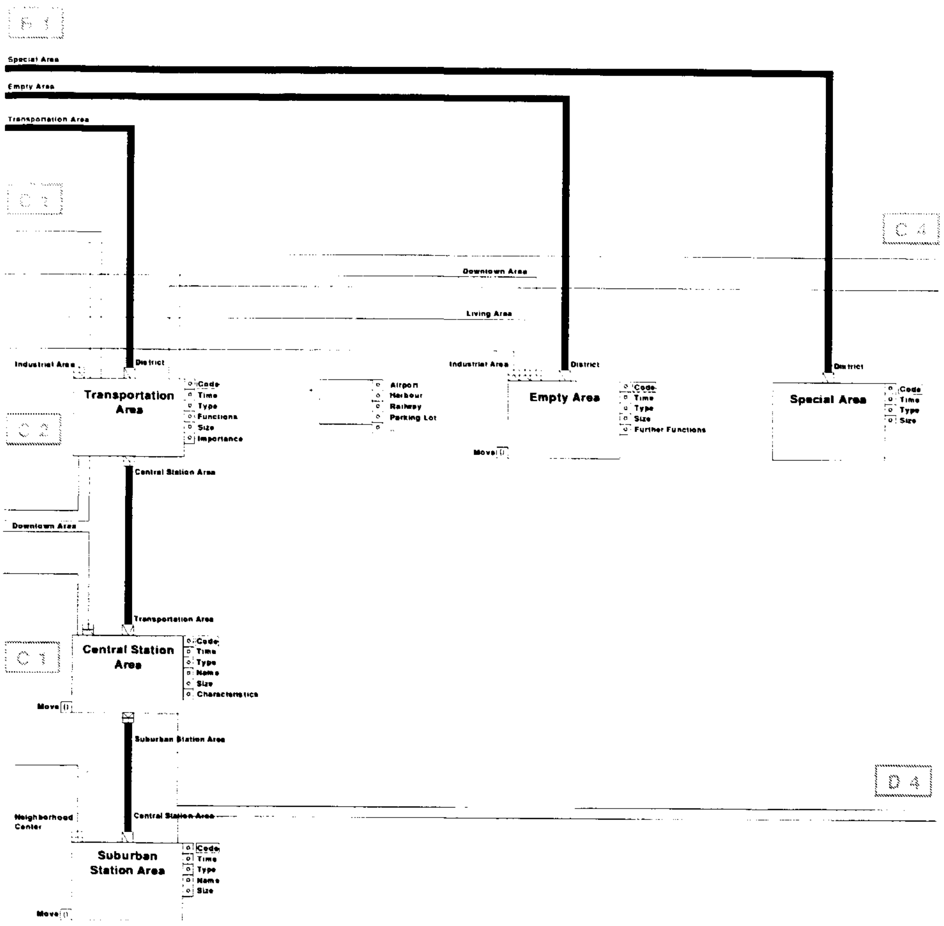
Element: C 1


D 1
Living Area
Downtown Area
Shopping Area
C 2
C 3
Transportation Area
Industrial Area
Living Area
Downtown Area
District
Downtown Area
Living Area
Code
Time
Type
Name
Size
Quality
District
Downtown Area
Code
Time
Type
Functions
Size
Quality
Attractiveness
Business
Administration
Culture
Hotels/Restaurants
Entertainment
Shopping
...
District
Shopping Area
Code
Time
Type
Name
Size
Quality
Offer
Neighborhood
Suburban Center
Shopping Mall
Industrial Complex
Transportation Area
Central Station
Living Area
Downtown Area
Downtown Area
Shopping Area
Neighborhood
Living Area
Suburban Center
Neighborhood
Downtown Area
Suburban Center
Industrial Complex
Neighborhood
Code
Time
Type
Name
Suroundings
Size
Social Structure
Quality/Condition
Suburban Center
Code
Time
Type
Name
Size
Quality
Attractiveness
Shopping Mall
Code
Time
Type
Name
Size
Quality
Attractiveness
Move
Building
Neighborhood Center
Store
Neighborhood
Shopping Street
Suburban Center
Downtown Area
Shopping Street
Neighborhood
Neighborhood Center
Building
Neighb.
Sub. Cen.
Neighborhood Center
Shopping Mall
Building
Code
Time
Type
Address
Size
Charact. Faccade
Historical Value
Condition
Move
Building
Neighborhood Center
Code
Time
Type
Name
Size
Quality
Move
Shopping Street
Code
Time
Type
Name
Size
Quality
Street
D 3

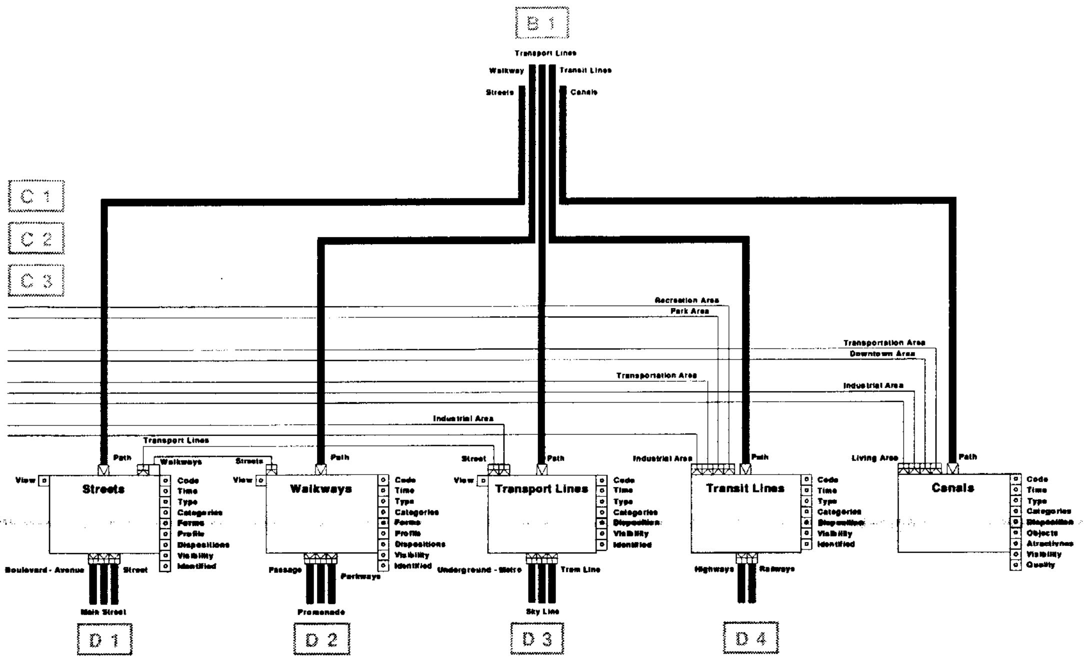
Element: C 2


B 1
Recreation Area
Park Area
Industrial Area
C 1
C 3
Transportation Area
Downtown Area
Downtown Area
Living Area
Industrial Area
Living Area
District
Recreation Area
Park Area
District
District
Transporta. Area
Park Area
Code
Time
Type
Name
Size
Quality
Recreation Area
Code
Time
Type
Name
Size
Quality
Offer
Industrial Area
Code
Time
Type
Name
Size
Quality
Importance
C 1
C 3
District Park
Sports Grounds
Industrial Complex
Industrial Complex
Downtown Area
Downtown Area
Downtown Area
Park Area
Neighborhood
Industrial Area
Park Area
District Park
Recreation Area
District Park
Code
Time
Type
Name
Size
Quality
Attractiveness
Sports Ground
Code
Time
Type
Size
Quality
Industrial Complex
Code
Time
Type
Size
Quality
C 1
C 3
Move
Move
Neighborhood Park
Neighborhood Greenery
Neighborhood Center
District Park
Neighborhood
Sports Ground
Neighborhood Greenery
Code
Time
Type
Name
Size
Quality
Move
Street Greenery
C 1
Shopping Street
Neighborhood Greenery
Street Greenery
Code
Time
Type
Age
Size
Quality
Move

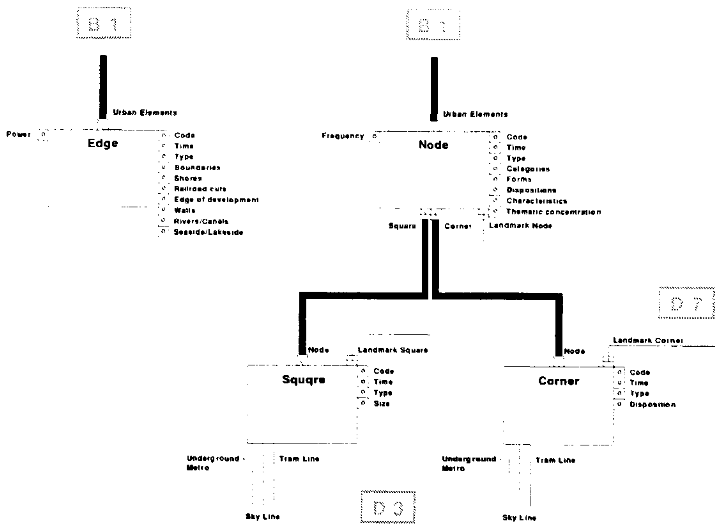
Element: C 3


B 1
Special Area
Empty Area
Transportation Area
C 3
C 4
Downtown Area
Living Area
Industrial Area
District
Transportation Area
Code
Time
Type
Functions
Size
Importance
Airport
Harbour
Railway
Parking Lot
Industrial Area
District
Empty Area
Code
Time
Type
Size
Further Functions
Move
District
Special Area
Code
Time
Type
Size
C 2
Central Station Area
Downtown Area
Transportation Area
C 1
Central Station Area
Code
Time
Type
Name
Size
Characteristics
Move
Suburban Station Area
D 4
Neighborhood Center
Central Station Area
Suburban Station Area
Code
Time
Type
Name
Size
Move

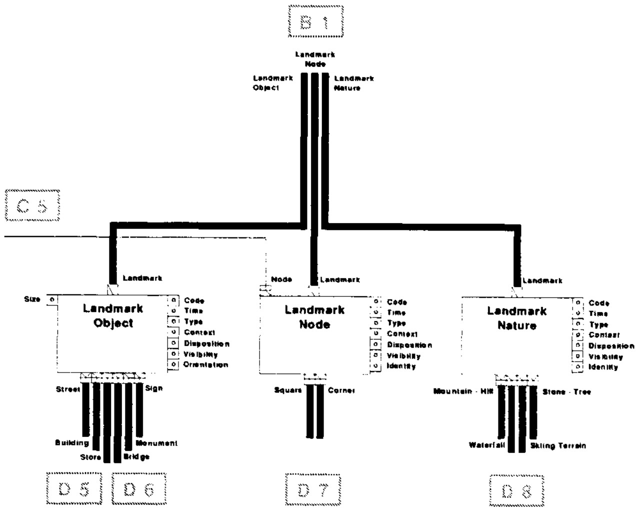
Element: C 4


B 1
Transport Lines
Walkway
Transit Lines
Streets
Canals
C 1
C 2
C 3
Recreation Area
Park Area
Transportation Area
Downtown Area
Transportation Area
Industrial Area
Industrial Area
Transport Lines
Path
Walkways
Streets
Path
Street
Path
Industrial Area
Path
Living Area
Path
View
Streets
Code
Time
Type
Categories
Forms
Profile
Dispositions
Visibility
Identified
Boulevard - Avenue
Street
Main Street
D 1
View
Walkways
Code
Time
Type
Categories
Forms
Profile
Dispositions
Visibility
Identified
Passage
Parkways
Promenade
D 2
View
Transport Lines
Code
Time
Type
Categories
Disposition
Visibility
Identified
Underground - Metro
Tram Line
Sky Line
D 3
Transit Lines
Code
Time
Type
Categories
Disposition
Visibility
Identified
Highways
Railways
D 4
Canals
Code
Time
Type
Categories
Disposition
Objects
Atractivnes
Visibility
Quality

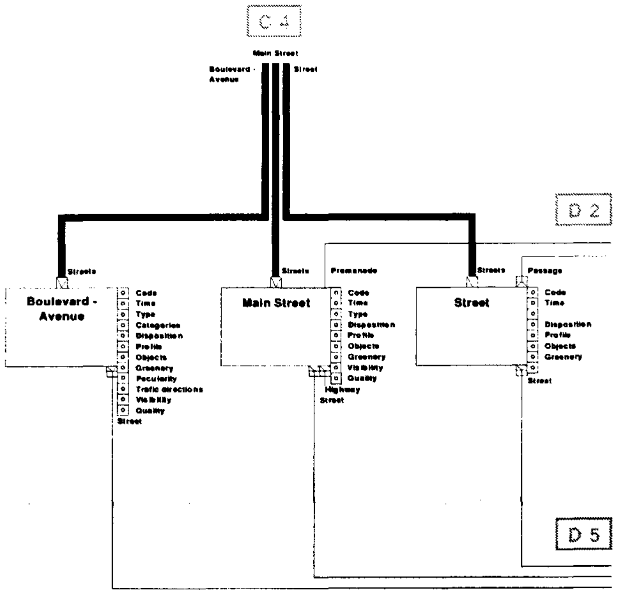
Element: C 5



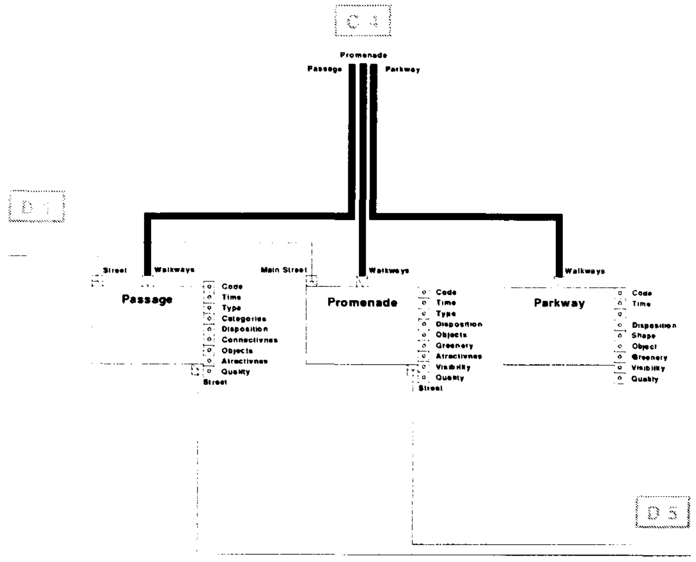
Element: C 6



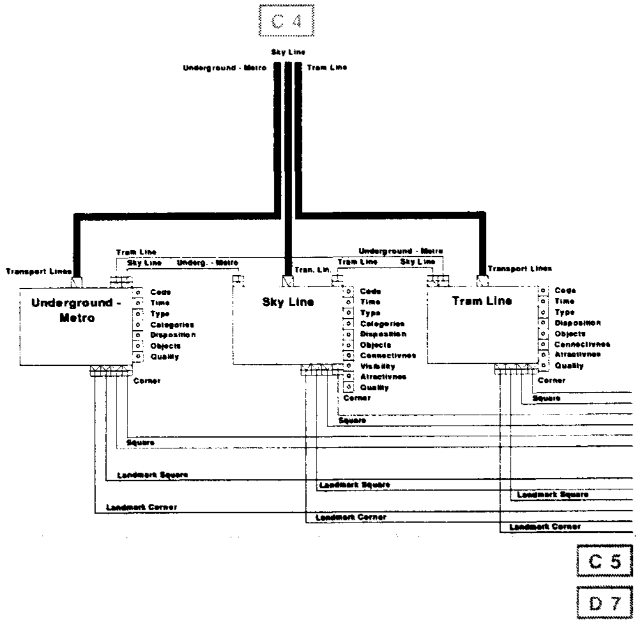
Element: D 1


C 4
Main Street
Boulevard - Avenue
Street
D 2
Streets
Streets
Promenade
Streets
Passage
Boulevard - Avenue
Code
Time
Type
Categories
Disposition
Profile
Objects
Greenery
Peculiarity
Trafic directions
Visibility
Quality
Street
Main Street
Code
Time
Type
Disposition
Profile
Objects
Greenery
Visibility
Quality
Highway
Street
Street
Code
Time
Disposition
Profile
Objects
Greenery
Street
D 5

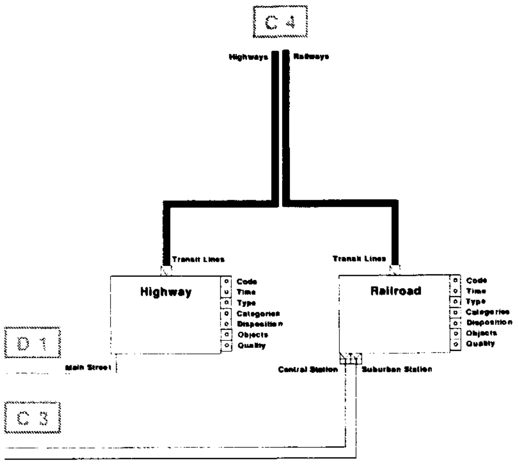
Element: D 2


C 4
Promenade
Passage
Parkway
D 1
Street
Walkways
Main Street
Walkways
Walkways
Passage
Code
Time
Type
Categories
Disposition
Connectivnes
Objects
Atractivnes
Quality
Street
Promenade
Code
Time
Type
Disposition
Objects
Greenery
Atractivnes
Visibility
Quality
Street
Parkway
Code
Time
Disposition
Shape
Object
Greenery
Visibility
Quality
D 5

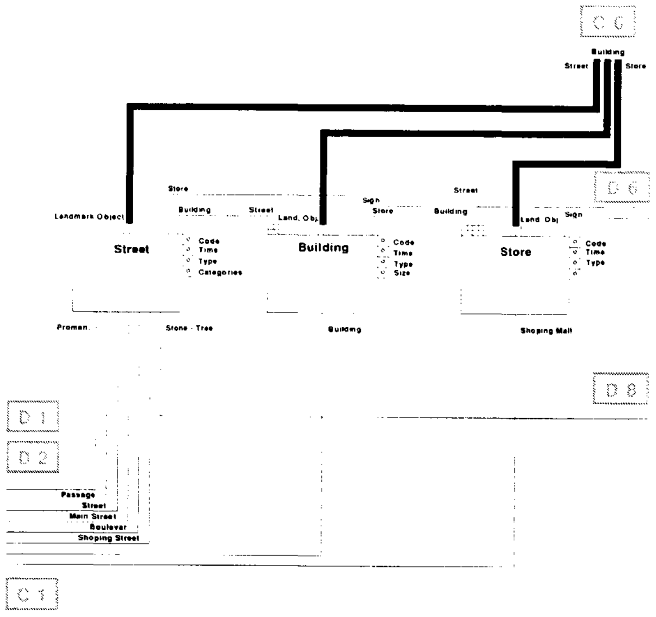
Element: D 3



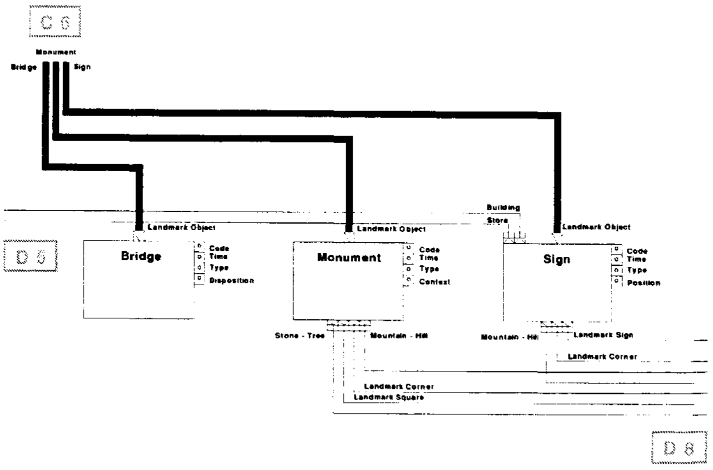
Element: D 4



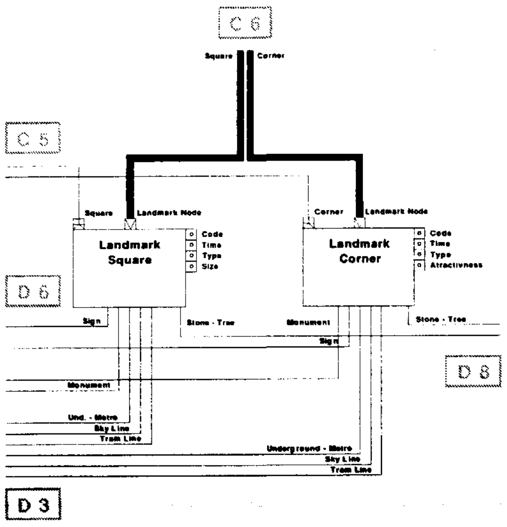
Element: D 5



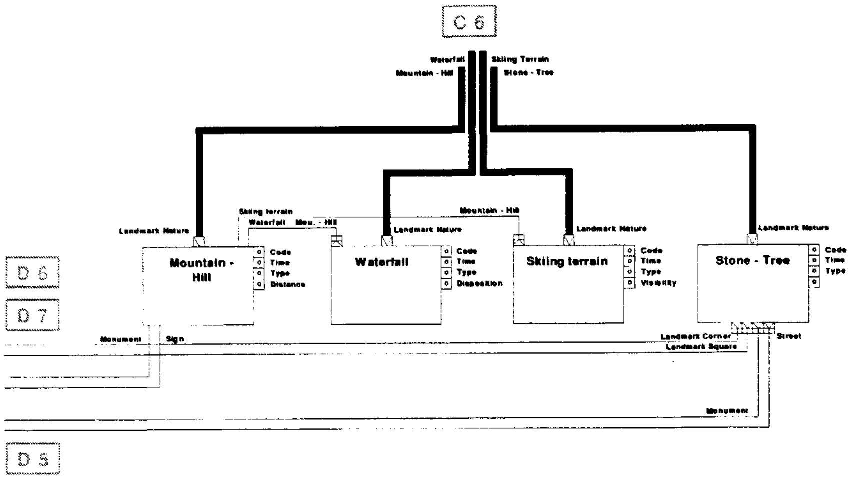
Element: D 6



Element: D 7



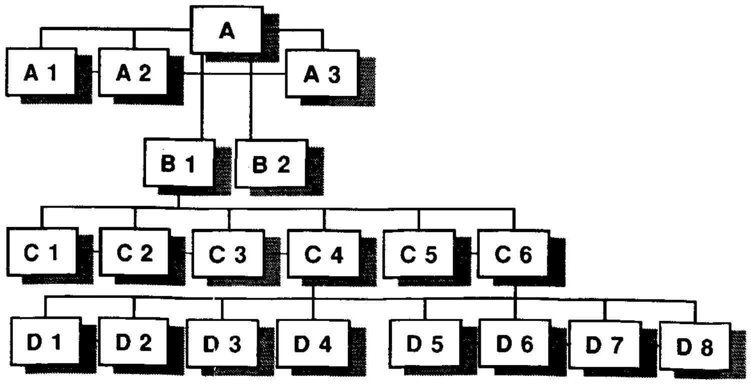
Element: D 8



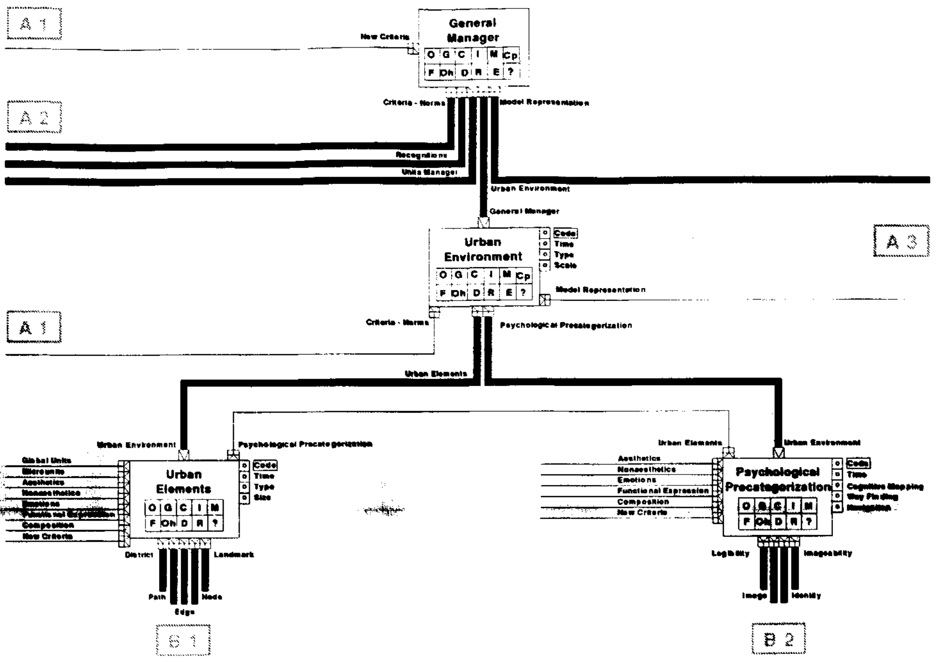
APPENDIX B

OBJECT-ORIENTED DATABASE MODEL - BEHAVIOR

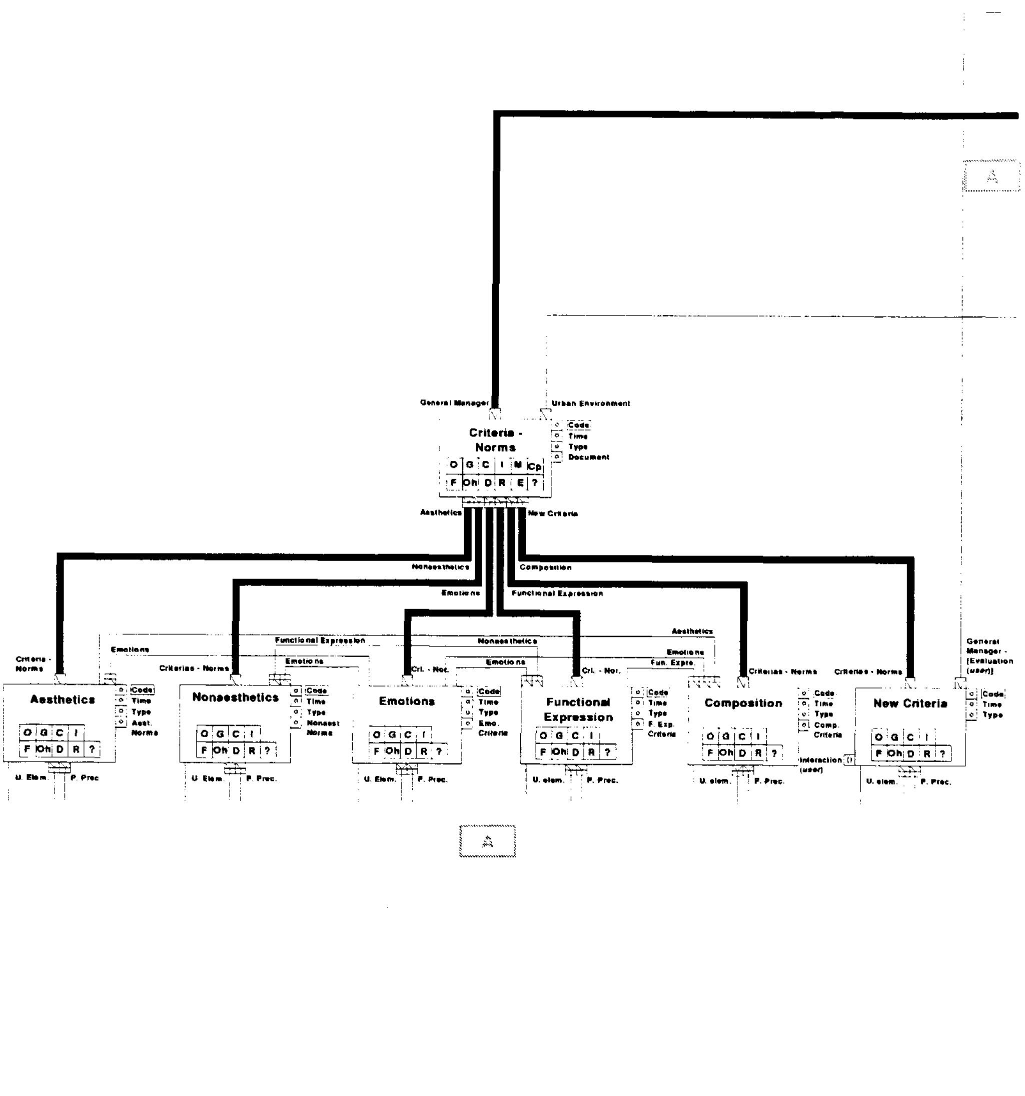
Map of the elements of the *object-oriented database model - behavior* and connections between them



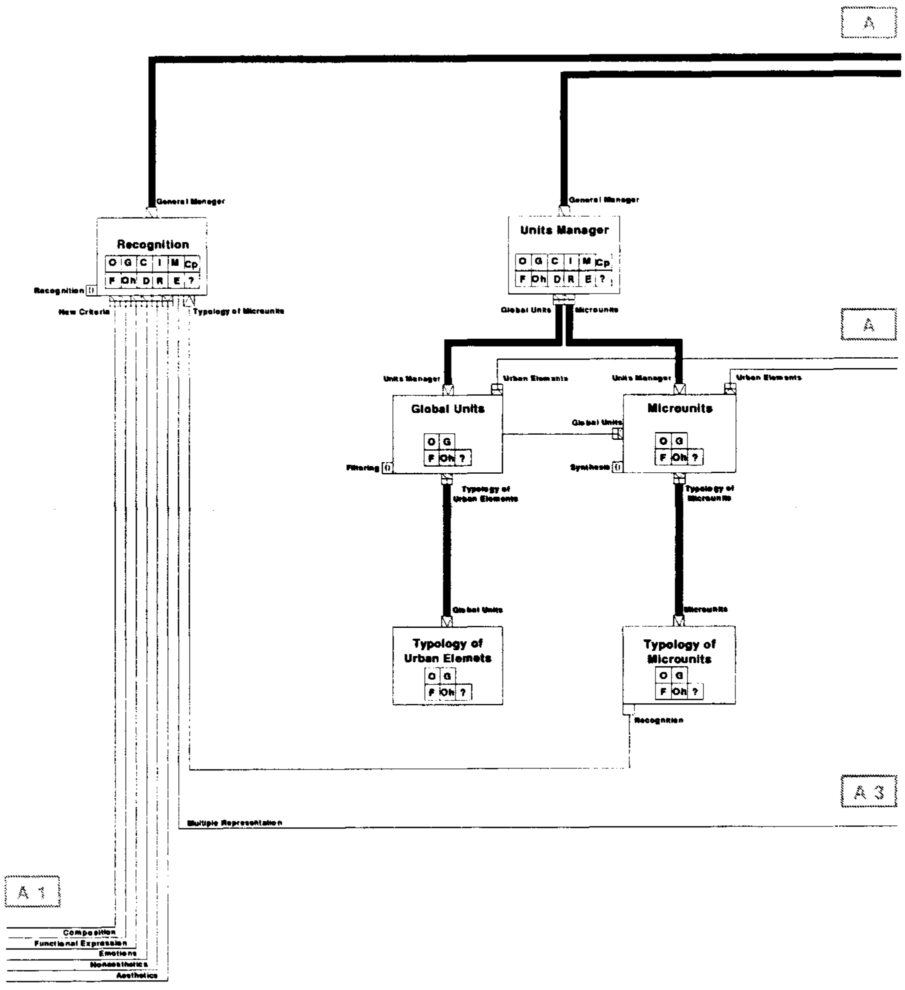
Element: A


A 1
General Manager
New Criteria
O G C I M Cp
F Oh D R E ?
A 2
Criteria - Norms
Model Representation
Recognitions
Units Manager
Urban Environment
General Manager
A 3
Urban Environment
Code
Time
Type
Scale
O G C I M Cp
F Oh D R E ?
Model Representation
A 1
Criteria - Norms
Psychological Precategorization
Urban Elements
Urban Environment
Psychological Precategorization
Global Units
Microunits
Aesthetics
Nonaesthetics
Emotions
Functional Expression
Composition
New Criteria
Urban Elements
Code
Time
Type
Size
O G C I M
F Oh D R ?
District
Landmark
Path
Node
Edge
B 1
Urban Elements
Urban Environment
Aesthetics
Nonaesthetics
Emotions
Functional Expression
Composition
New Criteria
Psychological Precategorization
Code
Time
Cognitive Mapping
Way Finding
Navigation
O G C I M
F Oh D R ?
Legibility
Imageability
Image
Identity
B 2

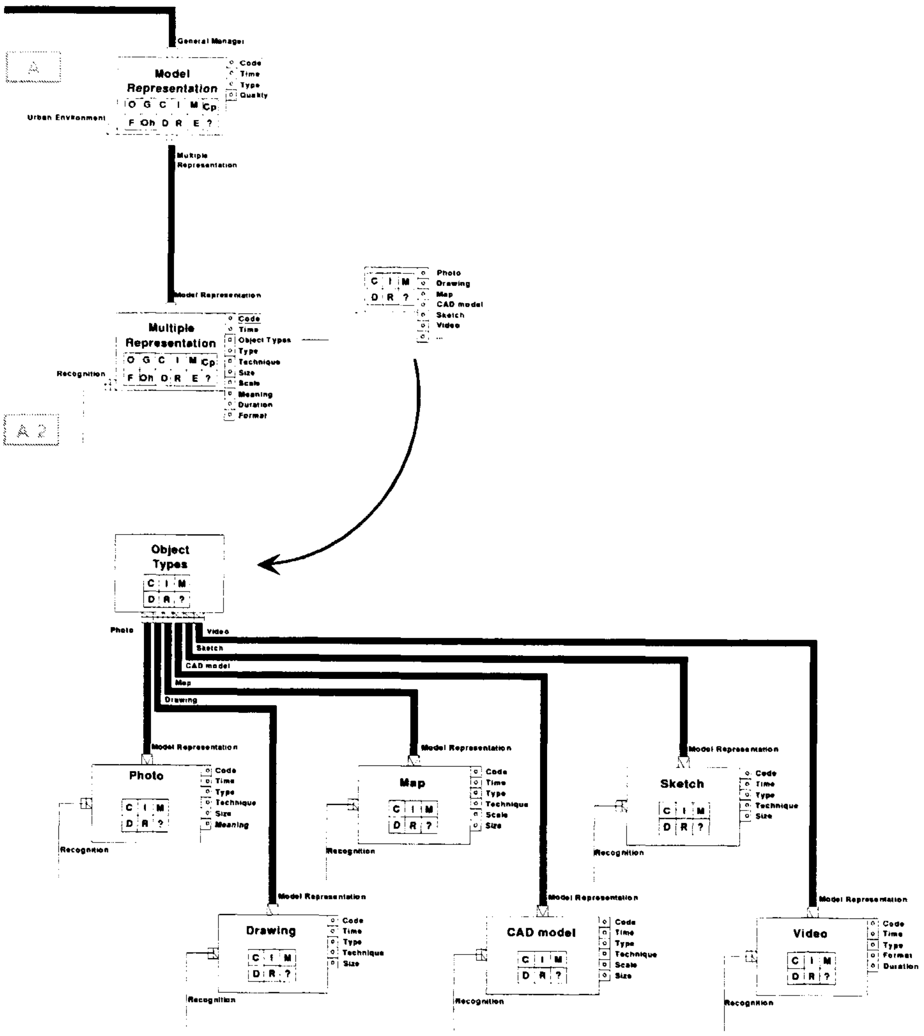
Element: A 1


General Manager
Urban Environment
Criteria - Norms
O G C I M Cp
F Oh D R E ?
Code
Time
Type
Document
Aesthetics
New Criteria
Nonaesthetics
Composition
Emotions
Functional Expression
Criteria - Norms
Aesthetics
Nonaesthetics
Emotions
Functional Expression
Composition
New Criteria
O G C I
F Oh D R ?
U. Elem.
P. Prec.
Code
Time
Type
Aest. Norms
Nonaest Norms
Emo. Criteria
F. Exp. Criteria
Comp. Criteria
Interaction (user)
General Manager - (Evaluation (user))
Cri. - Nor.
Fun. Expre.

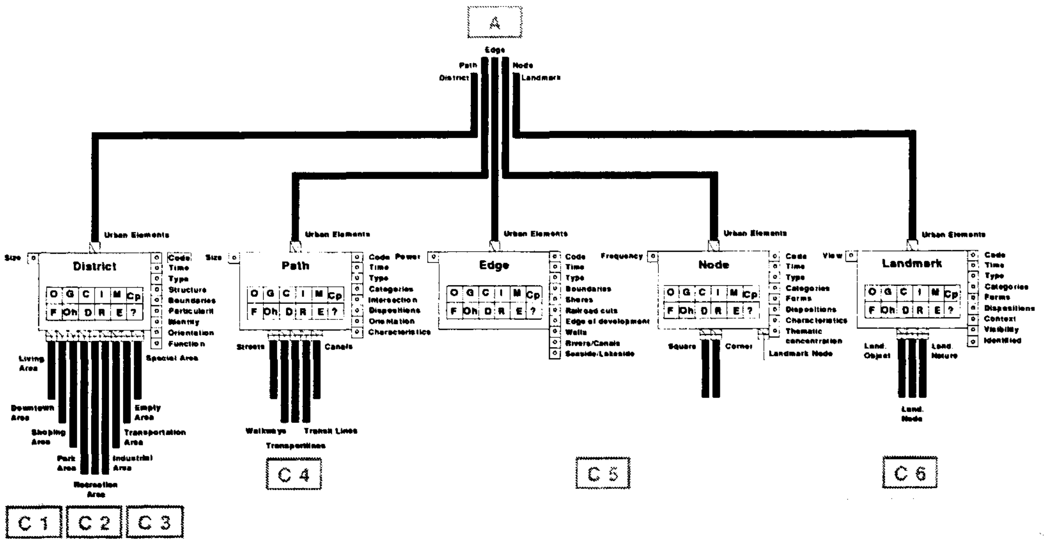
Element: A 2


A
General Manager
Recognition
O G C I M Cp
F Oh D R E ?
Recognition
New Criteria
Typology of Microunits
General Manager
Units Manager
O G C I M Cp
F Oh D R E ?
Global Units
Microunits
A
Units Manager
Urban Elements
Units Manager
Urban Elements
Global Units
O G
F Oh ?
Filtering
Global Units
Microunits
O G
F Oh ?
Synthesis
Typology of Urban Elements
Typology of Microunits
Global Units
Microunits
Typology of Urban Elemets
O G
F Oh ?
Typology of Microunits
O G
F Oh ?
Recognition
A 3
Multiple Representation
A 1
Composition
Functional Expression
Emotions
Nonaesthetics
Aesthetics

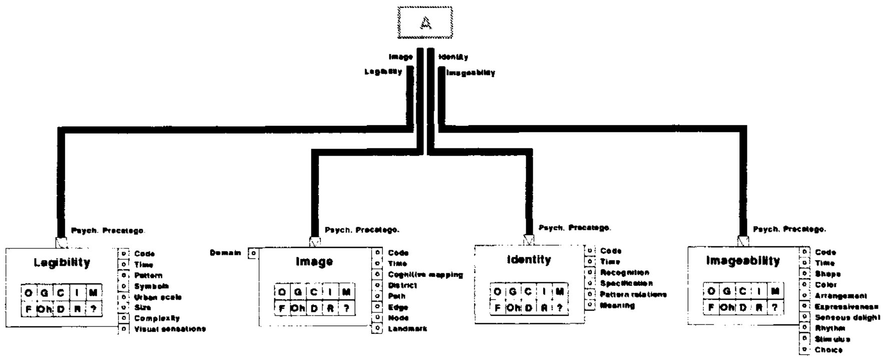
Element: A 3


General Manager
A
Model Representation
O G C I M Cp
F Oh D R E ?
Code
Time
Type
Quality
Urban Environment
Multiple Representation
Model Representation
Multiple Representation
Code
Time
Object Types
Type
Technique
Size
Scale
Meaning
Duration
Format
Recognition
A 2
C I M
D R ?
Photo
Drawing
Map
CAD model
Sketch
Video
...
Object Types
Photo
Video
Sketch
CAD model
Map
Drawing
Model Representation
Photo
Meaning
Recognition
Map
Sketch
Drawing
CAD model
Video
Technique
Format
Duration

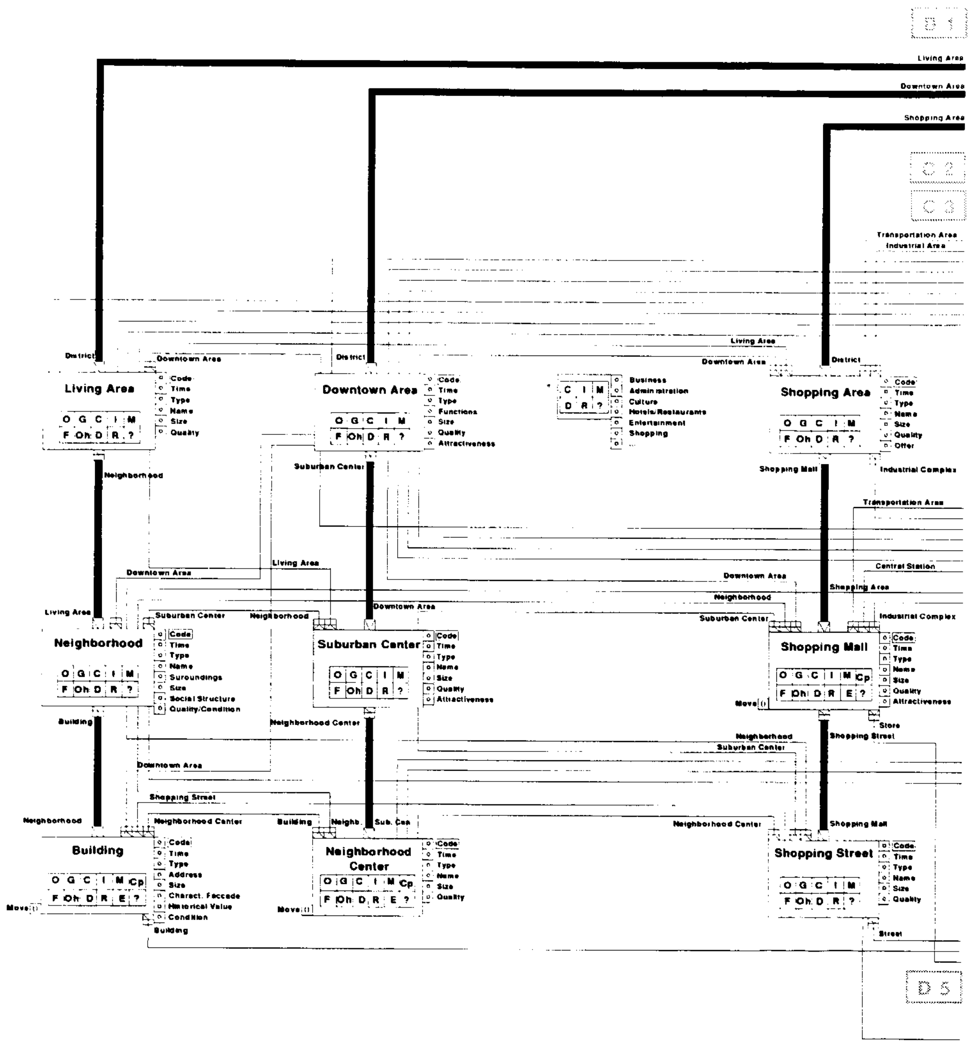
Element: B 1



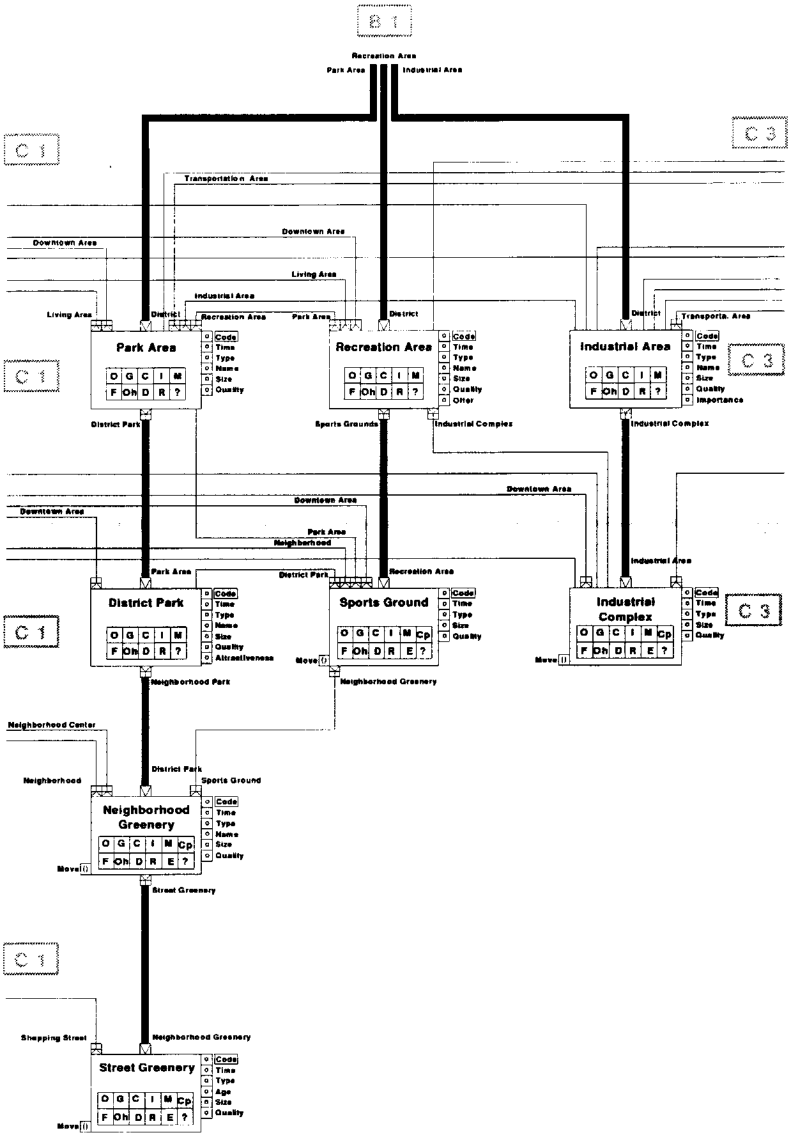
Element: B 2



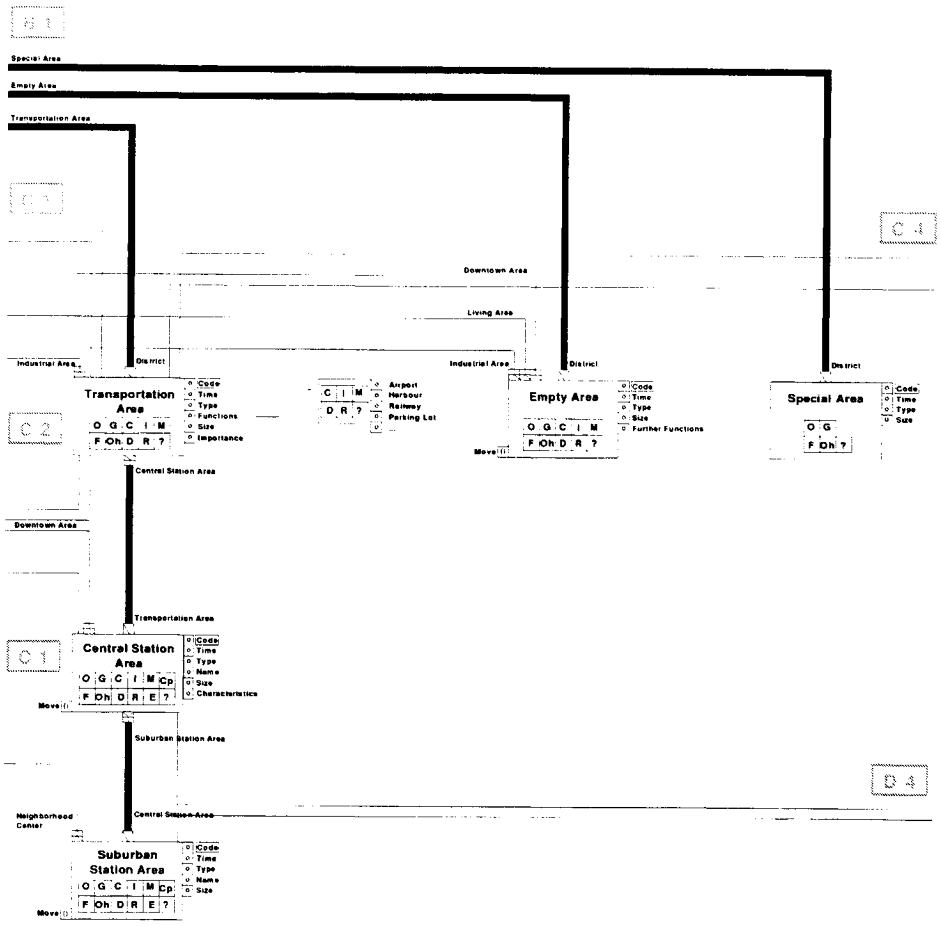
Element: C 1


B 1
Living Area
Downtown Area
Shopping Area
C 2
C 3
Transportation Area
Industrial Area
District
Downtown Area
Living Area
Code
Time
Type
Name
Size
Quality
O G C I M
F Oh D R ?
District
Downtown Area
Code
Time
Type
Functions
Size
Quality
Attractiveness
C I M
D R ?
Business
Administration
Culture
Hotels/Restaurants
Entertainment
Shopping
...
Living Area
Downtown Area
District
Shopping Area
Code
Time
Type
Name
Size
Quality
Offer
Neighborhood
Suburban Center
Shopping Mall
Industrial Complex
Transportation Area
Living Area
Downtown Area
Central Station
Downtown Area
Shopping Area
Neighborhood
Suburban Center
Industrial Complex
Living Area
Suburban Center
Neighborhood
Downtown Area
Neighborhood
Code
Time
Type
Name
Suroundings
Size
Social Structure
Quality/Condition
Suburban Center
Code
Time
Type
Name
Size
Quality
Attractiveness
Shopping Mall
O G C I M Cp
F Oh D R E ?
Code
Time
Type
Name
Size
Quality
Attractiveness
Move()
Store
Shopping Street
Building
Neighborhood Center
Neighborhood
Suburban Center
Downtown Area
Shopping Street
Neighborhood
Neighborhood Center
Building
Neigh.
Sub. Cen
Neighborhood Center
Shopping Mall
Building
Code
Time
Type
Address
Size
Charact. Faccade
Historical Value
Condition
Move()
Building
Neighborhood Center
Code
Time
Type
Name
Size
Quality
Move()
Shopping Street
Code
Time
Type
Name
Size
Quality
Street
D 5

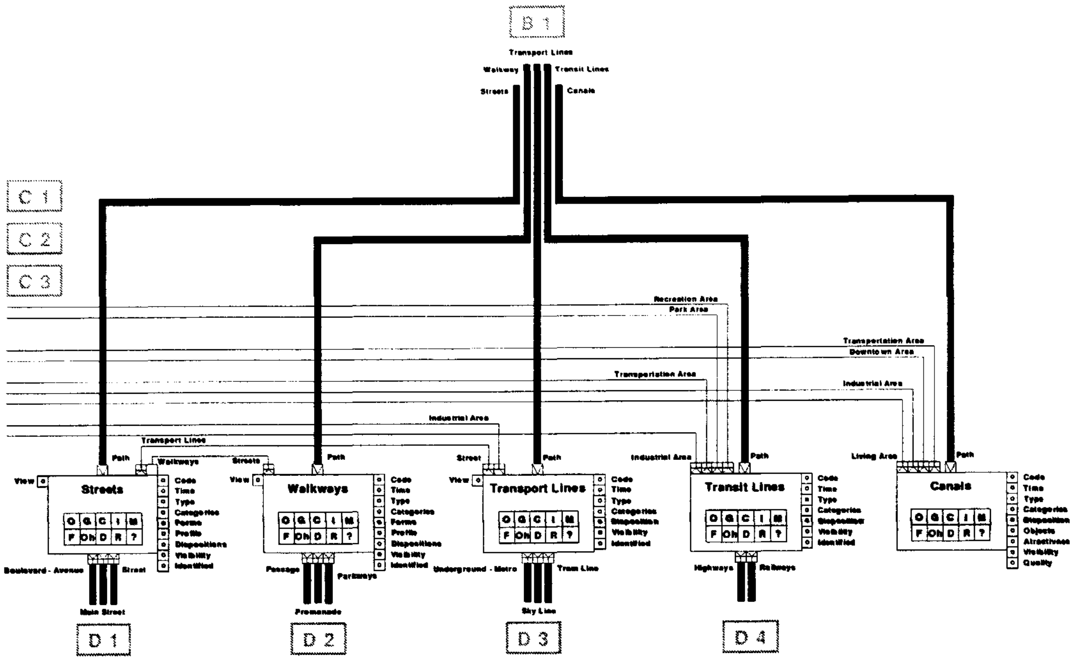
Element: C 2


B 1
Recreation Area
Park Area
Industrial Area
C 1
C 3
Transportation Area
Downtown Area
Living Area
Industrial Area
District
Recreation Area
Park Area
Transporta. Area
Park Area
Recreation Area
Industrial Area
Code
Time
Type
Name
Size
Quality
Other
Importance
O G C I M
F Oh D R ?
District Park
Sports Grounds
Industrial Complex
Neighborhood
District Park
Sports Ground
Industrial Complex
Attractiveness
Move
O G C I M Cp
F Oh D R E ?
Neighborhood Park
Neighborhood Greenery
Neighborhood Center
Neighborhood Greenery
Street Greenery
Shopping Street
Age

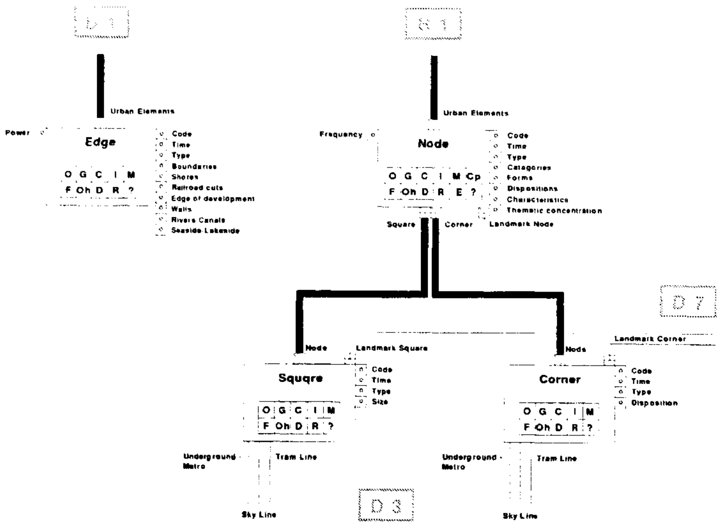
Element: C 3


B 1
Special Area
Empty Area
Transportation Area
C 3
C 4
Downtown Area
Living Area
Industrial Area
District
Industrial Area
District
District
Transportation Area
O G C I M
F Oh D R ?
Code
Time
Type
Functions
Size
Importance
C I M
D R ?
Airport
Harbour
Railway
Parking Lot
...
Empty Area
O G C I M
F Oh D R ?
Code
Time
Type
Size
Further Functions
Move()
Special Area
O G
F Oh ?
Code
Time
Type
Size
C 2
Central Station Area
Downtown Area
Transportation Area
C 1
Central Station Area
O G C I M Cp
F Oh D R E ?
Code
Time
Type
Name
Size
Characteristics
Move()
Suburban Station Area
D 4
Neighborhood Center
Central Station Area
Suburban Station Area
O G C I M Cp
F Oh D R E ?
Code
Time
Type
Name
Size
Move()

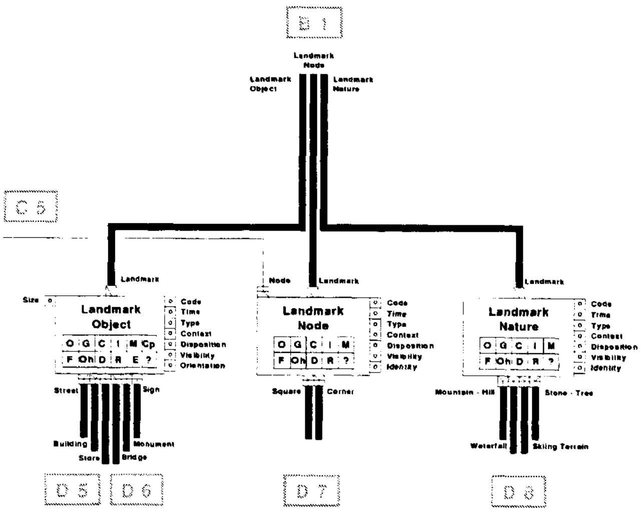
Element: C 4


B 1
Transport Lines
Walkway
Transit Lines
Streets
Canals
C 1
C 2
C 3
Recreation Area
Park Area
Transportation Area
Downtown Area
Transportation Area
Industrial Area
Industrial Area
Transport Lines
Path
Walkways
Streets
Street
Path
Industrial Area
Path
Living Area
Path
View
Streets
O G C I M
F Oh D R ?
Code
Time
Type
Categories
Forms
Profile
Dispositions
Visibility
Identified
Boulevard - Avenue
Street
Main Street
D 1
View
Walkways
O G C I M
F Oh D R ?
Code
Time
Type
Categories
Forms
Profile
Dispositions
Visibility
Identified
Passage
Parkways
Promenade
D 2
View
Transport Lines
O G C I M
F Oh D R ?
Code
Time
Type
Categories
Disposition
Visibility
Identified
Underground - Metro
Tram Line
Sky Line
D 3
Transit Lines
O G C I M
F Oh D R ?
Code
Time
Type
Categories
Disposition
Visibility
Identified
Highways
Railways
D 4
Canals
O G C I M
F Oh D R ?
Code
Time
Type
Categories
Disposition
Objects
Atractivness
Visibility
Quality

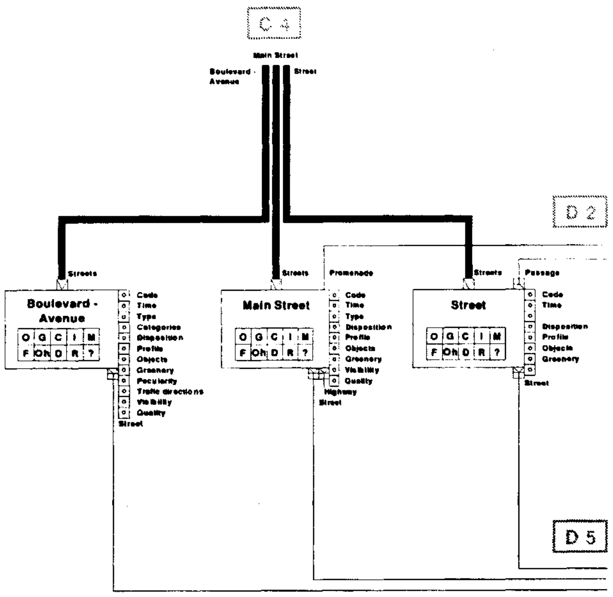
Element: C 5



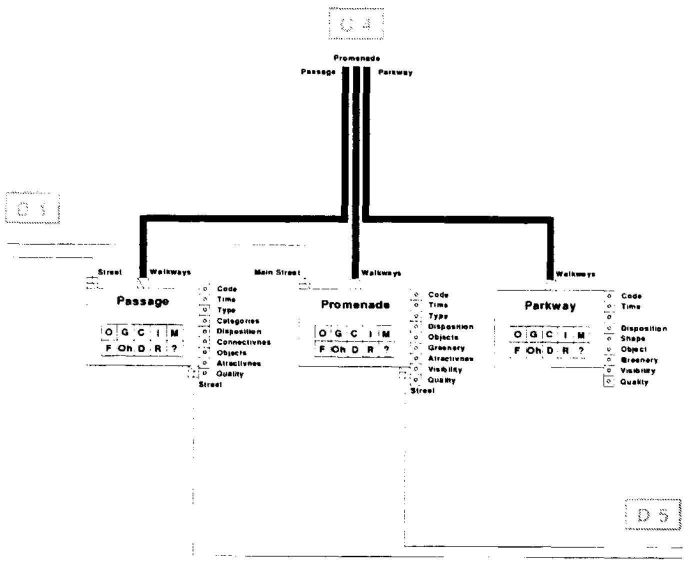
Element: C 6



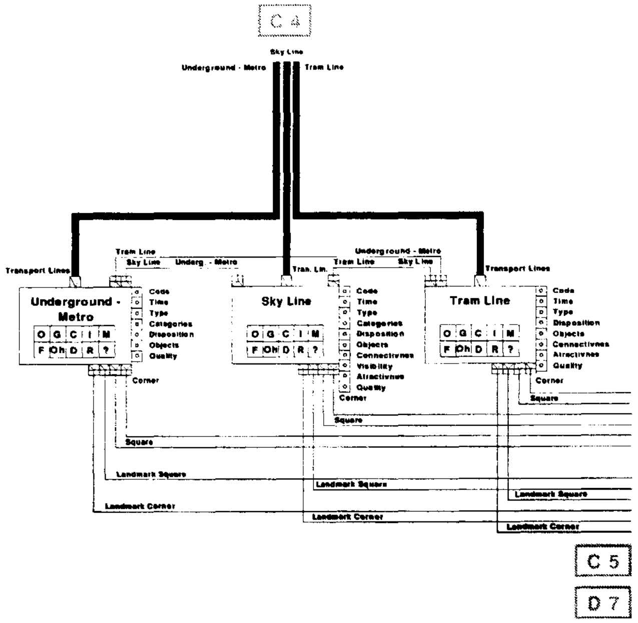
Element: D 1


C 4
Main Street
Boulevard - Avenue
Street
D 2
Streets
Boulevard - Avenue
O G C I M
F Oh D R ?
Code
Time
Type
Categories
Disposition
Profile
Objects
Greenery
Peculiarity
Traffic directions
Visibility
Quality
Street
Streets
Promenade
Main Street
O G C I M
F Oh D R ?
Code
Time
Type
Disposition
Profile
Objects
Greenery
Visibility
Quality
Highway
Street
Streets
Passage
Street
O G C I M
F Oh D R ?
Code
Time
Disposition
Profile
Objects
Greenery
Street
D 5

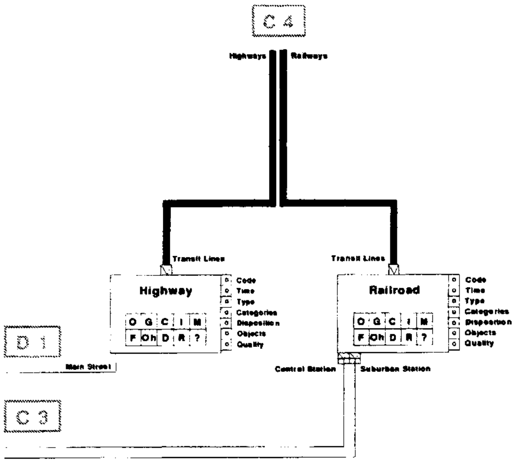
Element: D 2


C 4
Promenade
Passage
Parkway
D 1
Street
Walkways
Main Street
Walkways
Walkways
Passage
O G C I M
F Oh D R ?
Code
Time
Type
Categories
Disposition
Connectivnes
Objects
Attractivnes
Quality
Street
Promenade
O G C I M
F Oh D R ?
Code
Time
Type
Disposition
Objects
Greenery
Attractivnes
Visibility
Quality
Street
Parkway
O G C I M
F Oh D R ?
Code
Time
Disposition
Shape
Object
Greenery
Visibility
Quality
D 5

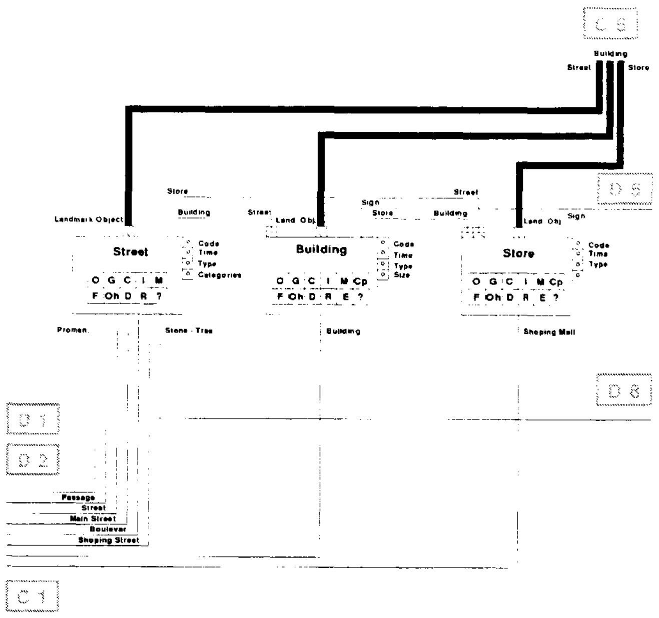
Element: D 3



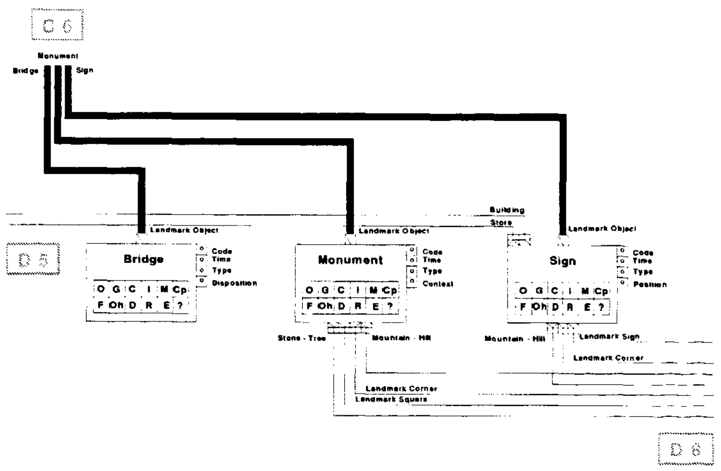
Element: D 4



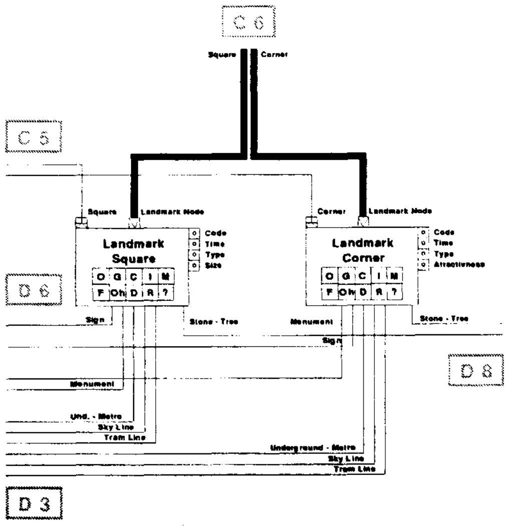
Element: D 5



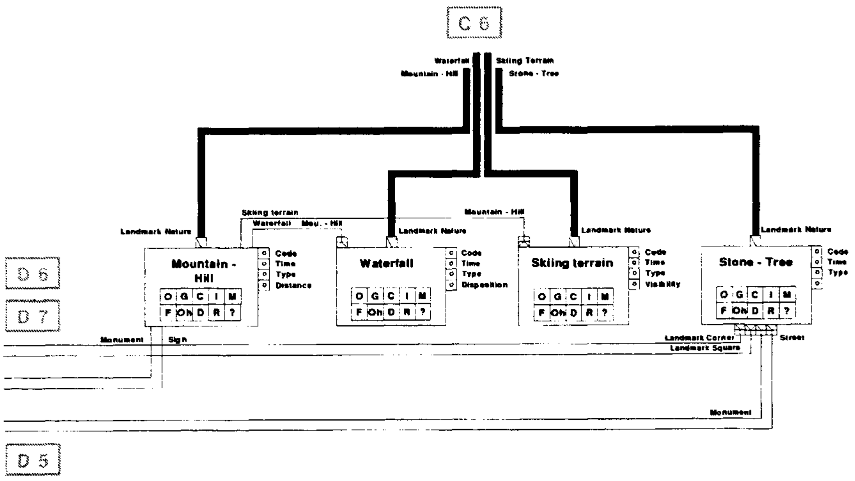
Element: D 6



Element: D 7



Element: D 8



REFERENCES

Adelman, L. 1991. Experiments, quasi-experiments and case studies: A review of empirical methods for evaluating decision support systems. In *IEEE Transactions on Systems, Man and Cybernetics*, 21(2), pp. 293-301.

Adler, R. M., and B. H. Cottman. 1989. A Development Framework for Distributed Artificial Intelligence. In *IEEE Proceedings of the Fifth Conference on Artificial Intelligence Applications.*

Ahituv, N., Even-Tsur, D., and B. Sadan. 1986. *Procedures and practices for conducting postevaluation of information systems.* Journal of Information Systems Management, 3(2).

Al-Kodmany. 2000. Using Web-Based Technologies and Geographical Information Systems in Community Planning. *Journal of Urban Technology* Vol. 7 No. 1. pp. 1-30.

Al-Kodmany. 2000. Extending Geographical Information Systems to Meet Neighborhood Planning Needs: The Case of Three Chicago Communities. *URISA Journal* Vol. 12 No. 3. pp. 19-37.

Alexander, C. 1969. *Major changes in environmental form required by social and psychological form.* Ekistics.

Alexander, C. et al. 1977. *A Pattern Language.* New York: Oxford University Press.

Anderson, J. R. 1983. *The architecture of cognition.* Cambridge, MA: Harvard University Press.

Anooshian, L. J. 1996. Diversity within Spatial Cognition - Strategies Underlying Spatial Knowledge. *Environment and Behavior* Vol. 28 No. 4. pp. 471-493.

Appleyard, D., Lynch, K., and J. R. Meyer. 1964. *The view from the road.* Cambridge, Massachusetts: MIT Press.

Appleyard, D. 1970. Styles and Methods of Structuring a City. *Environment and Behavior* 2(1). pp. 100-116.

Appelyard, D. 1976. *Planning a pluralist city*. Cambridge, Massachusetts: MIT Press.

Appleyard, D. 1981. *Livable Streets*. Berkeley, CA: University of California Press.

Barnett, J. 1982. *An introduction to urban design*. New York: Harper and Row Publisher Inc.

Bartlett, F. C. 1932. *Remembering: A Study in Experimental and Social Psychology*. Cambridge, UK: Cambridge University Press.

Batini, C., Ceri, S., and B. S. Navathe. 1992. *Conceptual Database Design - An Entity-Relationship Approach*. Redwood City: The Benjamin/Cummings Publishing Company, Inc.

Batty, M., Dodge, M., Jiang B., and A. Smith. 1998. GIS and urban design. In *CASA paper* 3. http://www.casa.ucl.acuk/urbandesifinal.pdf

Bereiter, C., and M. Scardamalia. 1985. Cognitive Coping Strategies and the Problem of 'Inert Knowledge'. In *Thinking and Learning Skills: Research and Open Questions (Vol. 2)* edited by S. Chipman, J. Segal and R. Glaser. Hillsdale, NJ: Lawrence Erlbaum.

Bitbybit Information Systems, Delft - http://www.bitbybit-is.nl

Blaha, M., and W. Premerlani. 1998. *Object-Oriented Modeling and Design for Database Applications*. New Jersey: Prentice Hall.

Briggs, R. 1971. *Urban cognitive distances*. Ohio State Univ.

Bruce, V., and P. R. Green. 1985. *Visual perception: Physiology, psychology & ecology*. Hillsdale N. J.: L. Erlbaum.
Bruce, V., Green, P. R., and M. A.. Georgeson. 1996. *Visual Perception*. Hove East Sussex, UK: Psychology Press.

Brysch, K. A., and J. Dickinson. 1996. Studies in Cognitive Maps - The Equiavailability Principle and Symmetry. *Environment and Behavior* Vol. 28, No. 2, pp. 183-203.

Burrough, P. A.. 1986. *Principles of Geographical Information systems for Land Resources Assessment*. Clarendon Press.

Carr, S. 1966. The city of the mind. Paper commissioned for the *Conference of the American Institute of Planners*.

Caar, S., and D. Schissler. 1969. The city as trip. *Environment and Behavior.* Vol. 1.

Carmona M. 1996. Controlling Urban Design - Part 1: A Possible Renaissance? *Journal of Urban design* Vol. 1, No. 1, pp. 47-73.

Cattell R. 1991. *Object Data Management: Object-Oriented and Extended Relational Database Systems.* Reading, Massachusetts: Addison-Wesley.

Coad, P., and E. Yourdon. 1990. *Object-Oriented Analysis.* 2nd ed., Englewood Cliffs, NJ: Yourdon Press/Prentice Hill.

Cooper, C. 1972. Resident Dissatisfaction in Multifamily Housing. In *Behavior, Design and Policy aspects of Human Habitats* edited by W. M. Smith. (Green Bay: University of Wisconsin, pp. 119-146).

Coton, B., and R. Oliver. 1994. *The Cyberspace Lexicon.* London: Phaidon Press Ltd.

Danahy, J. 1988. Engaging intuitive visual thinking in urban design modeling. In *ARCADIA: Workshop Proceedings* edited by P. Bancroft. (Ann Arbor, MI: University of Michigan, pp. 87-97).

Dave, B., and G. Schmitt. 1994. Information systems for urban analysis and design development. *Environment and Planning B: Planning and Design,* 21. pp. 83-96.

Dekker, J. 1992. Computers as tools for analysis of urban spaces. *Cities,* 9 (3). pp. 170-176.

Densham, P. 1991. Spatial decision support systems. In *Geographical Information Systems* edited by D. J. Maguire, M. F. Goodchild, and D. W. Rhind. (Harlow, UK: Longmans, pp. 403-412)

Darken, R. P., and J. L. Sibert. 1996. Wayfinding strategies and behaviors in large virtual worlds. Proceedings of *ACM special interest group on computer-human interaction.*

Eaglestone, B., and M. Ridley. 1998. *Object Databases: An Introduction.* Berkshire: McGraw-Hill

Edited by Bradshaw, J. M. 1997. *Software Agents.* Menlo Perk, CA: AAAI & Cambridge, Massachusetts: MIT Press.

Edited by Cattell R. G. G., and K. D. Barry. 2000. *The Object Database Standard: ODMG 3.0.* San Francisco, Ca.: Morgan Kaufmann Publishers, Inc.

Edited by Downs, R. M., and D. Stea. 1971. *Cognitive mapping: Images of spatial environment*. Chicago: Aldine.

Edited by Gero, J. S., and M. L. Mahev. 1993. *Modeling Creativity and Knowledge-Based Creative Design*. Hillsdale, New Jersey: Lawrence Erlbaum Associates, Publishers.

Edited by Hoc, J. M., Cacciabue, P. C. and E. Hollnagel. 1995. *Expertise and Technology: Cognition & Human-Computer Cooperation*. Hillsdale, New Jersey: Lawrence Erlbaum Associates, Publishers.

Edited by Kosslyn, S. M., and R. A. Andersen. 1995. *Frontiers in Cognitive Neuroscience*. Cambridge, Massachusetts: MIT Press.

Edited by Moore, G. T., and R. G. Golledge. (with a foreword by K. Lynch) 1976. *Environmental Knowing*. Stroudsburg, PE.: Dowden, Hutchingon & Ross, Inc.

Edited by Peuquet, D., and D. F. Marble. 1990. *Introductory Readings in Geographic Information Systems*. Taylor & Francis.

Edited by Thalmann, D. 1990. *Scientific Visualization and Geographic Simulation*. John Wiley & Sons.

Elmasri R., and Sh. B. Navathe. 2000. *Fundamentals of Database Systems*. Reading, Massachusetts:
Addison-Wesley.

Essenius E., M. Sim, N. Simon, P. Kist, and W. Gerhardt. 1998. A Working Generic Canonical Schematic UIMS for an ODBMS. In *Visual Database Systems (VDB4)*, edited by Y. Ioannidis and W. Klas. (Chapman & Hill. pp. 168-174).

Essenius E., N. Simon., P. Kist., W. Gerhardt., and M. Dankoor. 1997. *Reuse applied to be development of a GUI from ODBM*. Technical report. Delft: Delft University of Technology.

Evans, G. W. (1980). Environmental Cognition. *Psychological Bulletin* Vol. 88(2), pp. 259-287.

Evans, G. W., and K. Pezdek. 1980. Cognitive Mapping: Knowledge of real-world distance and location information. *Journal of Experimental Psychology*: (Human Learning and Memory Vol. 6. pp. 13-24).

Fang, N. 1993. *A Knowledge-Based Computational Approach to Architectural Precedent Analysis*. Doctoral thesis. Delft: TU Delft, Faculty of Architecture, DKS group.

Ferreira, J. and L. Wiggins. 1993. Computing technology for land use and regional planning. In *Third International Conference on Computers in Urban Planning and Urban Management* edited by R. E. Klosterman and S. P. French. Atlanta, GA: Georgia Institute of Technology.

Ferguson, E. L. and M. Hegarty. 1994. Properties of Cognitive Maps Constructed from Texts. *Memory & Cognition*. Vol. 22, pp. 455-473.

Foley, et. al. 1990. *Computer Graphics - Principles and Practice*. Addison-Wesley Publishing.

Forrester, J. W. 1971. Counterintuitive behavior of social systems. *Technology Review*, 73:52.

Forman, N. and R. Gillet. 1997, 1998. *A Handbook of Spatial Research Paradigms and Methodologies*. London: Psychology Press. (On http://www.psypress.com/BKFILES/0863777996.htm)

Gale, N., Golladge, R. G., Pellegrino, J. W. and S. Doherty. 1990. The Acquisition and Integration of Route Knowledge in an Unfamiliar Neighborhood. *Journal of Environmental Psychology*. Vol. 10, pp. 3-25.

Gamba, R. J. and S. Oskamp. 1994. Factors Influencing Community Residents' Participation in Commingled Curbside Recycling Programs. *Environment and Behavior*. Vol. 26 (5), pp. 587-612.

Gerhardt, W. and N. Simon. 1999. Characteristics of an Object Model and Object Management to Support Decision Processes. In *Proceedings of the 11th International Conference on Systems Research, Informatics and Cybernetics* edited by H. Lasker. Baden-Baden, Germany.

Gerhardt, W., Sidjanin P. and W. Bloom. 2000. Support the Information Processes of Real-Estate Corporations. In *Proceedings of the 'Euromedia 2000'* edited by P. Grail. Antwerp, Belgium.

Gibson, J. J. 1950. *The Perception of the Visual World*. Boston: Houghton Mifflin.

Gibson, J. J. 1979. *The ecological approach to visual perception*. Boston: Houghton Mifflin.

Godelier, M. (ed.). 1982. *Social science in France*. Paris: La Documentation francaise.

Godlier, M. 1984. *Imagined and material reality*. Paris: Fayard.

Golledge, R. G. and G. Zannaras. 1970. The perception of urban structure: An experimental approach. In *EDRA two - Proceedings of the second annual*

Environmental Design Research Association conference edited by J. Archea and C. Eastman. Pittsburgh, Pennsylvania: Carnegie Press.

Golledge, R. G., Briggs, R. and D. Demko. 1969. The configuration of distances in intra-urban space. *Proceedings of the Association of American Geographers.*

Granger, G. G. 1986. An epistemology of scientific work. In *La philosophie des science aujourd'hui* edited by J. Hamburger. pp. 111-122. Paris: Gauthier-Villars.

Granger, G. G. 1988. *For philosophical knowledge.* Paris: Odile Jacob.

Granger, G. G. 1989. *Can we surmise the limits of scientific knowledge?* In *Karl Poper et la science aujourd'hui* edited by R. Bouveresse. pp. 9-19. Paris: Aubier.

Grimson, W. E. L. and D. Marr. 1979. A computer implementation of the theory of human stereo vision, In *Proceedings of ARPA Image Understanding Workshop* edited by L.S. Baumann. SRI. pp. 41-45.

Groat, L. and C. Despres. 1990. The Significance of Architectural Theory for Environmental Design Research. *Advances in Environment, Behavior and Design* edited by E. H. Zube and G. T. Moore. Vol. 4, pp. 3-53. New York: Plenum.

Harris, B. 1989. Beyond geographical information systems: computers and the planning professional. *Journal of the American Planning Association.* Vol. 55, pp. 85-90.

Hart, R. and M. Berzok. 1982. Children's strategies for mapping the geographic-scale environment. In *Spatial abilities: Development and physiological foundations* edited by M. Potegal. New York: Academic Press.

Holahan, C. J. 1986. Environmental Psychology. *Annual Review of Psychology.* Vol. 37, pp. 381-407.

Hollnagel, E., Cacciabue, P. C. and J. M. Hoc. 1995. Work with Technology: Some Fundamental Issues. In *Expertise and Technology (Cognition & Human-Computer Cooperation)* edited by Hoc, Cacciabue and Hollnagel. Hillsdale, New Jersey: Lawrence Erlbaum Associates, Publishers.

Ingram, R. and S. Bendorf. 1995. Legibility enhancement for information visualization. *'Visualization '95 proceedings'.* Los Alamitos, CA: IEEE Computer Society Press.

Ittelson, W. H. 1970. The perception of the large-scale environment. Paper presented to *the New York Academy of Science*, New York.

Ittelson, W. H. 1973. *Environment and cognition.* New York: Seminar Press.

Jeng, H.E. 1995. *A dialogical Model for Participatory Design*. Doctoral dissertation. Delft: TU Delft, Faculty of Architecture, DKS group.

Kaplan, S. 1982. Attention and Fascination: The Search for Cognitive Clarity. In *Human scape: Environments for People* edited by S. Kaplan and R. Kaplan. Ann Arbor, MI: Ulrich's.

Kaplan, S. and R. Kaplan. 1982. *Cognition and environment*. New York: Prager Publishers.

Kaplan, S. and R. Kaplan. 1989. *Cognition and Environment: Functioning in an Uncertain World*. Ann Arbor, MI: Ulrich's.

Kerney, A.R. and S. Kaplan. 1997. Toward a Methodology for the Measurement of Knowledge Structures of Ordinary People - The Conceptual Content Cognitive Map (3CM). *Environment and Behavior*. Vol. 29 No.5, pp. 579-617.

Kjaer-Hansen, J. 1995. Unitary Theories of Cognitive Architecture. In *Expertise and Technology (Cognition & Human-Computer Cooperation)* edited by Hoc, Cacciabue & Hollnagel. Hillsdale, New Jersey: Lawrence Erlbaum Associates, Publishers.

Klosterman, R. E. 1997. Planning support systems: a new perspective on computer-aided planning. *Journal of Planning Education and Research*.

Koenderink, J. and A. van Doorn. 1982. The shape of smooth objects and the way contours end. *Perception*.

Koenderink, J. 1989. *Solid shape*. Cambridge, Massachusetts: MIT Press.

Koutamanis, A. 1990. *Development of a Computerized Handbook of Architectural Plans*. Doctoral thesis. Delft: TU Delft, Faculty of Architecture, DKS group.

Kosslyn, S. M. 1994. *Image and Brain*. Cambridge, Massachusetts: A Bradford Book The MIT Press.

Kosslyn, S. M. and O. Koening. 1992. *Wet mind: The new cognitive neuroscience*. New York: The Free Press.

Kraak, M. J., G. Smets. and P. Sidjanin. 1995. Virtual Reality, the new 3D interface for Geographical Information Systems. *Proceedings of 1st Conference on Spatial Multimedia and Virtual Reality*. Lisbon, Portugal.

Kraak, M. J., G. Smets and P. Sidjanin. 1999. Virtual Reality, the new 3D interface for Geographical Information Systems. In the book *Spatial Multimedia and Virtual Reality* edited by A.S. Camara and J. Raper. London: Taylor & Francis.

Kuipers, B. 1982. The "Map in the Head" metaphor. *Environment and Behavior*. Vol. 14, pp. 202-220.

Kutcher, A. 1973. *The New Jerusalem*. London: Thames and Hudson.

Lang, J. 1988. *Symbolic Aesthetics in Architecture: Toward a Research Agenda*. In *Environmental Aesthetics: Theory, Research and Applications* edited by J.L. Nasar . pp. 11-26. New York: Cambridge University Press.

Langran, G. 1992. *Time in Geographic Information Systems*. London: Taylor & Francis.

Lazarus, R. S. 1984. On the Primacy of Cognition. *American Psychologist*.Vol. 39, pp. 124-129.

Lee, T. 1970. *Psychology of spatial orientation*. Architectural Association Quarterly.

Levine, M., I. N. Jankovic, and Palij M. 1982. Principles of spatial problem solving. *Journal of Experimental Psychology: General*. Vol. 111(2), pp. 157-175.

Lindzey, G. and J. H. Thorpe. 1968. Projective techniques. In *International encyclopedia of the social sciences* edited by D. L. Sills. Vol. 12. New York: Free Press.

Lynch, K. 1960. *The image of the city*. Cambridge, Massachusetts: MIT Press.

Lynch, K. 1971. *Site planning*. Cambridge, Massachusetts: MIT Press.

Lynch, K. 1972. *What time is this place?* Cambridge, Massachusetts: MIT Press.

Lynch, K. 1976. *Managing the Sense of a Region*. Cambridge, Massachusetts: MIT Press.

Lynch, K. 1984. Reconsidering the image of the city. In *Cities of the Mind: Images and Themes of the City in the Social Sciences* edited by L. Rodwin and R. M. Hollister. pp. 151-161. New York: Plenum Press.

Marr, D. 1976. *Early processing of visual information*. Phil. Trans. R. Soc. London, England, Vol. 275, pp. 483-524.

Marr, D. 1977. *Analysis of occluding contour*. Proc. R. Soc. London, England, B 197, pp. 441-175.

Marr, D. and H. K. Nishihara. 1978. Representation and recognition of the spatial organization of three-dimensional shapes. Proceedings R.S. London, B 207.

Marr, D. 1982. *Vision*. New York: Freeman.

McLuhan, M. 1965. *Understanding Media*. New York: McGraw-Hill.

Mediyckyj, S. and H. M. Hearnshow. 1993. *Human factors In Geographical Information Systems*. Belhaven Press.

Milgram, S. 1970. The experience of living in cities. *Science*. Vol. 167.

Ministry VROM. 1995. *Notitie over het beleid voor ruimtelijke kwaliteit*. Den Haag, RPD. 17 oktober.

Minsky, M. 1975. A framework for representative knowledge. In *Mind Design: philosophy, psychology, artificial intelligence* edited by J. H. Haugeland. Cambridge, Massachusetts: MIT Press.

Molich, R. and J. Nielsen. 1990. Improving a human-computer dialogue. *Communications of the ACM 33*. Vol. 3 (March), pp. 338-348.

Moughtin, C. 1992. *Urban Design: Street and Square*. Oxford: Butterworths Architecture.

Nasar, J. L. 1983. Adult Viewer's Preferences in Residential Scenes: A Study of the Relationship of Environmental Attributes to Preference. *Environment and Behavior*. Vol. 15, pp. 589-614.

Nasar, J. L. 1987. Effects of Signscape Complexity and Coherence on the Perceived Visual Quality of Retail Scenes. *Journal of the American Planning Association*. Vol. 53, pp. 499-509.

Nasar, J. L. 1988. Editor's introduction. In Environmental Aesthetics: Theory, Research and Applications edited by J. L. Nasar. pp. 257-259. New York: Cambridge University Press.

Nasar, J. L. 1989. Perception, Cognition and Evaluation of Urban Places. In Public Places and Spaces: Human Behavior and Environment edited by I. Altman and E. H. Zube. Vol. 10, pp. 31-56. New York: Plenum.

Nasar, J. L. 1990. The Evaluative Image of the City. The Journal of American Planning Association. Vol. 56, pp. 41-53.

Nasar, J. L. 1994. Urban Design Aesthetics - The Evaluative Qualities of Building Exteriors. *Environment and Behavior*. Vol. 26, No 26, pp. 377-401.

Nasar, J. L. 1998. *The Evaluative Image of the City*. Thousand Oaks, CA.: Sage Publications, Inc.

Neisser, U. 1967. *Cognitive Psychology*. New York: Appleton-Century-Crofts.

Newell, A. 1990. *Unified theories of cognition*. Cambridge, MA: Harvard University Press.

Nielsen, J. and R. Molich. 1990. Heuristic evaluation of user interface. *Conference proceedings ACM CHI'90*. pp. 249-256. Seattle, WA, 1-5 April.

Norberg-Schulz, C. 1965. *Intentions in Architecture*. Cambridge, Massachusetts: MIT Press.

Norman A. D. 1999. *The Design of Everyday Things*. Cambridge, Massachusetts: MIT Press.

Overbeeke, C. J. and M. H. Stratmann. (1988). *Space through movement*. Doctoral thesis. Delft: TU Delft, Faculty of Industrial Design.

Passini, R. 1984. *Wayfinding in Architecture*. New York: Van Nostrand Reinhold.

Passini, R. at al. 1990. The spatio-cognitive abilities of the visual impaired population. *Environment and behavior*. Vol. 22(1), pp. 91-118.

Posner, M. I. and S. W. Keele. 1968. *On the Genesis of Abstract Ideas. Journal of Experimental Psychology*. Vol. 77, pp. 353-363.

Posner, M. I. 1973. *Cognition: An Introduction*. Glenview, IL: Scott, Foresman.

Power, D. J. 1996-1998. *DSS glossary in terms*. On: http://www.dss.cba.edu/djp/dssglosary.html

Preece, P. F. W. 1976. Mapping Cognitive Structure: A Comparison of Methods. *Journal of Experimental Psychology*. Vol. 68(1), pp. 1-8.

Presson, C. C. 1987. The Development of Landmarks in Spatial Memory: The role of Different Experience. *Journal of Experimental Child Psychology*. Vol. 44, pp. 317-334.

Presson, C. C., DeLange, N. S. and M. D. Hazelrigg. 1989. Orientation-Specificity in Spatial Memory: What Makes a Path Different from a Map of the Path? *Journal of Experimental Psychology: Learning, Memory and Cognition*. Vol. 15, pp. 887-897.

Rechtin, E. 1991. *System Architecting - Creating and Building Complex Systems*. Englewood Cliffs, New Jersey: Prentice Hall.

Roberts, L. G. 1965. Machine perception of three-dimensional solids. In *Optical and electro optical information processing* edited by J. T. Tippett et al. pp. 159-197. Cambridge, Massachusetts: MIT Press.

Rosch, E. 1977. Principles of Categorization. In *Cognition and Categorization* edited by E. Rosch and B. Lloyd. Hillsdale, NJ: Lawrence Erlbaum.

Rowley, A. 1994. Definitions of Urban Design. *Planing Practice and Research*. Vol. 9 (3), pp. 179-197.

Reason, J. 1990. *Human error*. Cambridge, UK: Cambridge University Press.

Russell, J. A. and L. M. Ward. 1982. Environmental Psychology. *Annual Review of Psychology*. Vol. 33, pp. 651-688.

Russell, J. 1988. *Affective Appraisals of Scenes*. In *Environmental Aesthetics: Theory, Research and Applications* edited by J. L. Nasar. pp. 120-129. New York: Cambridge University Press.

Safdie, M. 1986. *The Harvard Jerusalem Studio*. Cambridge, Massachusetts: MIT Press.

Saarinen, T. F. 1964. *Image of the Chicago Loop*. University of Chicago.

Schank, R. 1991. *Case-Based Teaching: Four Experiences in Educational Software Design*. Technical Report #7. Evanston, IL: Institute for Learning Science, Northwestern University.

Schön, D. A. 1983. *The reflective practitioner; how professionals think in action*. Avebury: Alderschot. (reprinted 1991).

Schön, D. A. et. al. 1996. High Technology and Low-Income Communities: Prospects for the Positive Use of Advanced Information Technology. Colloquium on *Advanced Technology, Low-Income Communities and the City*, MIT. http://sap.mit.edu/projects/colloquium/book.html

Shiffer, M. J. 1992. Towards a collaborative planning system. *Environment and Planning B: Planning and Design*. Vol. 19, pp. 709-722.

Shneiderman, B. 1987. *Designing the User Interface - Strategies for Effective Human-Computer Interaction*. Addison-Wesley Publishing Company.

Siegel, A.. W. and S. H. White. 1975. The development of spatial representation of large-scale environments. In *Advances in child development and behavior* edited by H.W. Reese. Vol. 10. New York: Academic Press.

Siegel, A.. W., Kirasic, K. C. and R. V. Kail. 1978. Stalking the Elusive Cognitive Map. In *Human Behavior and Environment* edited by I. Altman and J. F. Wohlwill. Vol. 3, pp. 223-258. New York: Plenum.

Siegel, A. W. 1981. The Externalization of Cognitive Maps by Children and Adults: In Search of Ways to Ask Better Questions. In *Spatial Representation and Behavior Across the Life Span* edited by L. Liben, A. Patterson and N. Newcombe. pp. 167-194.New York: Academic Press.

Sidjanin, P. 1995. *A computer simulation model of the TU district of Delft with use of the GIS and VR*. Master thesis. Delft: Delft University of Technology, Faculty of Architecture.

Sidjanin, P., Kraak M. J. and G. J. F. Smets. 1995. The Delft University of Technology's Campus Information System accessed by GIS and Virtual Reality technology. In *Proceedings of JEC*. The Hague, The Netherlands.

Sidjanin, P. 1996. A computer simulation model of TU district of Delft with use of the GIS and VR. In *Proceedings of 3re International Conference on Design and Decision Support Systems in Architecture and Urban Planning*. Spa, Belgium.

Sidjanin, P. 1996. GIS and VR - an integration. *EUROMEDIA 96*. London, England.

Sidjanin, P. and W. Gerhardt. 1998. A design tool for analysis and visual quality control of urban environments supported by object databases. In *Proceedings of 4th International Conference on Design and Decision Support Systems in Architecture and Urban Planning*. Maastricht, The Netherlands.

Sidjanin, P. 1998. A computer based design tool for cognitive mapping, analysis and visual environmental quality control. In *Proceedings of International Conference on Information Visu*alization. London, England.

Sidjanin, P. 1999. A design tool for analysis and visual quality control of urban environment. *URISA'99 - The Urban and Regional Information Systems Association Annual Conference*. Chicago, USA. (published in proceedings CD-ROM of the URISA '2000).

Sidjanin, P. and W. Gerhardt 2000. Outline of a design tool for analysis and visual quality control of urban environments. In *Proceedings of the IRMA 2000 - Information Resources Management Association International Conference*. Anchorage, Alaska, USA.

Sidjanin, P. 2000. Architectural research - between heuristic and empirical: a case study of the design tool. In *Proceedings of AEEA conference*. Paris, France.

Sidjanin, P. 2000. *A conceptual model of the new design tool for analysis of urban environment.* In Proceedings CD-ROM of the *URISA'2000 - The Urban and Regional Information Systems Association Annual Conference.* Orlando, USA.

Sidjanin, P. 2000. A conceptual model of the design tool for analysis and visual quality control of urban environment. In *Proceedings of 5th International Conference on Design and Decision Support Systems in Architecture and Urban Planning.* Nijkerk, The Netherlands.

Sidjanin, P. 2000. The role of the new computer visualization in architecture - a change of paradigm in architectural practice. La Carre Blau. Vol. 3/4. Paris, France.

Sprout, H. and M. Sprout. 1965. *The ecological perspective on human affairs.* Princeton, New Jersey: Princeton University Press.

Stea, D. 1969. *Environmental perception and cognition: Toward a model for 'mental maps'*. Raleigh, North Carolina: North Carolina State University.

Stea, D. and M. R. Downs. 1973. *Image and Environment.* London: Edward Arnold.

Strohecker, C. and B. Barros. 1998. Make Way for WayMaker. In *Presence: Teleoperators and Virtual Environments.* Cambridge, Massachusetts: MIT Press.

Strohecker, C. 1999. Toward a Developmental Image of the City. In *Proceedings of Visual and spatial Reasoning in Design.* Cambridge, Massachusetts: MIT Press and University of Sydney, Sydney, Australia.

Sutherland, N. S. 1979. The representation of three-dimensional objects. *Nature.* Vol. 27, pp. 395-398.

Syme, G. J., Beven, C. E. and N. R. Sumner. 1993. Motivation for Reported Involvement in Local Wetland Preservation: The Roles of Knowledge, Disposition, Problem-Assessment and Arousal. *Environment and Behavior.* Vol. 25(5), pp. 586-606.

Thimbleby, H. 1990. *User Interface Design.* Addison-Wesley Publisher.

Thorndyke, P. W. 1981. *Spatial Cognition and Reasoning.* In *Cognition, Spatial Behavior and the Environment* edited by J. H. Harvey. pp. 137-149. Hillsdale, NJ: Lawrence Erlbaum.

Tolman, E.C. 1948. Cognitive Maps in Rats and Men. *Psychological Review.* Vol. 55, pp. 189-203.

Tzonis, A. and L. Lefaivre. 1994. 'Thinking in forms as well as words': Kevin Lynch and Cognitive Theory of the City. *Design Book Review*. Vol. 26, pp. 23-29.

Tzonis, A. and L. Lefaivre. 1986. *Classical Architecture: The Poetics of order*. Cambridge, Massachusetts: MIT Press.

Tversky, B. 1991. Spatial Mental Models. *The Psychology of Learning and Motivation*. Vol. 27, pp. 109-145.

Ullman, S. 1976. On visual detection of light sources. *Biol. Cybernetics*. Vol. 21, pp. 205-212.

Ullman, S. 1979. *The Interpretation of Visual Motion*. Cambridge, Massachusetts: MIT Press.

Ullman, S. 1996. *High-level vision*. Cambridge, Massachusetts: MIT Press.

Ulrich, R. S. 1983. Aesthetic and Affective Response to Natural Environment. *Human Behavior and Environment: Advances in Theory and Research* edited by Alteman, I. and J. F. Wohlwill. Vol. 6, pp. 85-125. New York: Plenum.

Vining, J. and A. Ebreo. 1990. What Makes a Recycler? A Comparison of Recyclers and Nonrecyclers. *Environment and Behavior*. Vol. 22(1), pp. 55-73.

Vygotsky, L. S. 1962. *Thought and Language*. Cambridge, Massachusetts: MIT Press

Waltz, D. 1975. Understanding Line Drawings of Scenes with Shadows. *Psychology of Computer Vision* edited by H. P. Winston. Cambridge, Massachusetts: MIT Press.

Ward, L.M. and J. A. Russel. 1981. The Psychological Representation of Molar Physical Environments. *Journal of Experimental Psychology: General*. Vol. 110, pp. 121-152.

Winograd, T. and F. Flores. 1986. *Understanding computers and cognition*. Addison-Wesley Publishing Company.

Winston, P. H. 1992. *Artificial Intelligence*. Reading, MA: Addison-Wesley Publishing Company.

Wohlwill, J. F. 1976. Environmental Aesthetics: The Environment as a Source of Affect. In *Human Behavior and the Environment: Advances in Theory and Research* edited by I. Altman and J. F. Wohlwill. Vol. 1, pp. 37-86. New York: Plenum.

Xiaoding, L. 1993. *Meaning of the site*. Doctoral thesis. Delft: TU Delft, Faculty of Architecture, DKS group.

Yin, R. K. 1989. *Case study research: Design and Method* (rev. ed.). Newbury Park, CA: Sage.

Zajonc, R. B. 1984. On the Primacy of Affect. *American Psychologist*. Vol. 39, pp. 117-123.

Zandi-Nai, A. 1992. *Topogene: An Artificial Intelligence Approach to a Design Process*. Doctoral thesis. Delft: TU Delft, Faculty of Architecture, DKS group.

Zannaras, G. 1969. *An empirical analysis of urban neighborhood perception*. Ohio State University.

Zonabend, F. 1992. The monograph in European ethnology. Current sociology. Vol. 40(1), pp. 49-54.

INDEX

ABOUT THE AUTHOR

Predrag Sidjanin was born in Novi Sad, Yugoslavia in 1953. He obtained his bachelor's degree in 1981 at the Faculty of Architecture, Belgrade University, Yugoslavia. From 1981 to 1987 he worked as a project leader in the architectural department in the 'Naftagas-inzenjering' corporation in Novi Sad. From 1987 to 1991 he worked as an urban designer and was the head of the Informatics Department at the Institute for Urban and Regional Planning of Vojvodina, Novi Sad. In the period between 1988 and 1991 he also worked in own design office, 'Multi Image Studio', a studio for computer-based design, visualization and publishing, in Novi Sad.

When civil war broke out in Yugoslavia in 1991, for political reasons he left the country with his family and went to the Netherlands, where he still resides as a Dutch citizen. He found a job as an AIO at the OPB at the Faculty of Architecture, TU Delft, in 1993. He completed his Master's thesis in 1995 with the title: "A computer simulation model of TU district of Delft with use of GIS and VR". In the same year he joined the Design Knowledge Research Center at the Faculty of Architecture, Delft University of Technology, and started his Ph. D. research there. In 1999 he won an academic award for a conference paper, the Hoorwood Critique Prize - honorable mention URISA [Urban and Regional Information Systems Association] conference in Chicago, USA.

He has been a professional artist since 1972, when his first solo exhibition was held, using different media in his expression, from classical to analog and digital. He has had 27 solo exhibitions and more than 200 group exhibitions all over the world. His works can be seen in several museums, galleries and private collections. He is a member of various international and national professional art and design associations such as ICOGRADA/UPIDIV, ULUV and STROOM. He is also member of The Electronic Literature Organization[1] and TRACE on-line writing community.

[1] See his contribution at: http://www.directory.eliterature.org

In current architectural, urban design and planning practice, the role of new information technology has become important. The use of computer models has been going on in architectural design and urban planning since the 1950s. Early architectural drafting was transformed through computer-aided design (CAD) packages, such as the now widely used *AutoCAD*. In urban planning, computation began with municipal information systems and landuse transportation modelling. In the last decade there have been major developments in tools for visualization and representation, particularly through the development of geographical information systems (GIS). With the development of GIS technology, it has become possible to link up with available 2D or 3D spatial data through new desktop packages such as *ArcInfo*, *ArcView* or *MapInfo*, which have become standard. On the other hand, urban design was for many years untouched by computational developments. Now it is set to slowly change.

This multidisciplinary research is focused on improving design methods and theory by developing new knowledge about designing as a cognitive process embedded in urban design practice. The main scope and the starting point of the research is Lynch's theory of 'urban form' and his concept of cognitive mapping. This research proposes a cognitive approach that can be integrated into computer processing and involves the integration of knowledge from design theory, cognitive science and the use of computational techniques. With the support of a theoretical platform, this integration leads to a conceptual model of the Design Tool.

This research provides new theoretical insights into the structure and the process of analysing, evaluating and controlling the visual quality of the urban environment, and a useful prescriptive model that can be applied in design practice. The implementation of the results of this research, as a practical software application available for commercial urban design exploitation, must be examined.

ISBN 90-9014862-0