Computer Aided Design for Construction in the Building Industry
Computer Aided Design for Construction in the Building Industry

Proefschrift

ter verkrijging van de graad van doctor aan de Technische Universiteit Delft, op gezag van de Rector Magnificus, prof.ir. K.F. Wakker, in het openbaar te verdedigen ten overstaan van een commissie aangewezen door het College van Dekanen op dinsdag 25 januari 1994 te 14.00 uur

door

Gijsbertus Theodorus Luiten
civiel ingenieur,
geboren te Harmelen.
Dit proefschrift is goedgekeurd door de promotor:
prof.ir. F.P. Tolman.

De promotiecommissie is samengesteld uit de leden:
prof. B.C. Björk,
prof.dr. H.G. Sol,
prof.ir. H. van Tongeren,
prof.ir. Ch.J. Vos en
prof.ir. J. Witteveen.
To Brechtje
The research reported in this thesis is sponsored by Ballast Nedam Engineering BV and the Applied Mechanics section of the Civil Engineering department of Delft University of Technology.

Published and distributed by
Gijsbertus Theodorus Luiten
2e Schuytstraat 108
2517 XJ The Hague
The Netherlands

Layout and cover designed by Helma Luiten.

CIP-DATA KONINKLIJKE BIBLIOTHEEK, DEN HAAG
Luiten, Gijsbertus Theodorus

Computer aided design for construction in the building industry / Gijsbertus Theodorus
Luiten. - [S.l. : s.n.]. - III.
Thesis Technische Universiteit Delft. - With index, ref. - With summary in Dutch
ISBN 90-9006779-5
Subject headings: computer integrated construction / project information modelling

Trademarks used in this thesis: AutoCAD and DXF (Autodesk, Inc.), Design++ (Design
Power, Inc.), Eiffel (Interactive Software Engineering, Inc.), Intergraph (Intergraph
Corporation, Inc.), Macintosh (Apple Computer, Inc.), Objective-C (Productivity
Products International), OSF/Motif (Open Software Foundation, Inc.), SUN and Sparc
(SUN Microsystems, Inc.), TeleUSE (TeleSoft AB Sweden), and UNIX (AT&T Bell
Laboratories).

© 1994 by Gijsbertus Theodorus Luiten

All rights reserved. No part of this publication may be reproduced, stored in a retrieval
system, or transmitted, in any form or by any means, electronic, mechanical,
photocopying, recording or otherwise, without the prior permission of the author.
Preface

Society increases its demands on the building industry. It demands higher quality, shorter lead times, custom made buildings, more environmental consideration, and better working conditions.

Improving the integration of design and construction processes can help the building industry in responding to these growing demands. Integration of design and construction implies the sharing of goals, knowledge, and information. Improving this sharing has always been important in the building industry. Information technologies might offer new possibilities for further improvement.

This thesis studies the possibilities of Computer Aided Design for Construction (CA DfC) in the building industry and proposes a strategy for its development and implementation. The strategy describes a way to integrate computer applications used during design, construction management, and construction. It also describes a way to support CA DfC with computers. The thesis is structured according to three related points of view on CA DfC, i.e., the integration concept, the computer implementation, and the organisational implementation.

The research reported in the thesis is part of a larger project at Delft University of Technology, called "Computer Integrated Construction" (CIC). In this CIC project six PhD students work on different aspects of a general concept for integration of building processes. The CIC project is partly funded by the Dutch Technology Foundation. The research is conducted in close relationship with the building companies Ballast Nedam, Waco-Liesbosch Beton, and Spanbeton, and the research institute TNO Building and Construction Research.

The thesis can be used in different ways. Of course, people who intend to implement the proposed strategy, or a similar one, can best read the whole book. A good first impression of CA DfC and the proposed strategy can be acquired by reading the summary, the introduction in chapter 1, the current state of DfC in chapter 2, and the conclusions in chapter 9.

Several readers may only be interested in a particular part of the research. Readers interested in the theoretical background of integration by means of project modelling should read chapter 4. Readers who are interested in developing tools to support information modelling should read chapter 5. People interested in the organisational aspects of CA DfC are suggested to read chapter 6. Chapters 3, 7, and 8 could provide readers with additional information about the state-of-the-art of current research
in this field, with results from an application of the strategy in a test case, and with an evaluation of the research.

For people interested in automation in the precast concrete industry the test case (chapter 7) might be of interest. The chapter describes an application of the strategy to integration of preliminary design and assembly on site of precast concrete structures.

The research and this thesis should not be credited to me alone. I could not have achieved it without the help of many others.

First I want to thank my supervisor, Frits Tolman, who always drummed the overall picture into my mind. I thank Ballast Nedam, Waco-Liesbosch Beton, and Spanbeton for the opportunities they have provided me to learn from practice and their patience during the case study. In this context Jacques Duivenvoorden deserves a separate mentioning for never closing his critical eye. I thank my TNO colleagues for the inspiring cooperation during the implementation phases and the many fruitful discussions. In particular I want to thank Bart Luijten, Peter Kuiper, Wim Bakkeren, and Michel Böhms.

Financially, this research has been made possible by Ballast Nedam Engineering and the Applied Mechanics section of the Civil Engineering department of the university. Their support is gratefully acknowledged.

It should be noted that the thesis would not look half as pretty without the help of my sister Helma Luijten. She is acknowledged for her design of the layout and the cover. I am also indebted to Martin Fischer, who worked hard on the first chapters of the thesis to fix my broken English.

I was not only supported in the research itself, also the personal attention I received has been invaluable. I want to express my appreciation to Tineke Kostwinder and Nico Stuurstraat. Their efforts have made me feel at home at TNO and the Civil Engineering Department.

Last but not least, I wish to express my gratitude to my parents for their unconditional support and to Brechtje for her love and the evenings and weekends she accompanied me at TNO.

Den Haag, December 1993

Bart Luijten
Contents

Summary xi

List of Abbreviations xvii

1 Introduction 1
  1.1 Introduction to the Subject of Research 1
  1.2 Introduction to the Research 5

2 Current State of DfC 9
  2.1 Analysis of the Building Process 9
  2.2 The Interactions of DfC 12
  2.3 Conclusions on DfC in Current Practice 17

3 Related Research 21
  3.1 Related Research on Integration Concepts 21
  3.2 Related Research on Computer Implementations 39
  3.3 Related Research on Organisational Implementations 43

4 Integration Concept: Building Project Model 47
  4.1 The Building Project Model (BPM) 47
  4.2 Abstraction mechanism 1: Product-Activity 48
  4.3 Abstraction mechanism 2: Concretisation 64
  4.4 BPM as Combination of the two Abstraction Mechanisms 73

5 Computer Implementation: PMshell 77
  5.1 Implementation Requirements 77
  5.2 Implementation of PMshell 88
  5.3 Implementation of the BPM with PMshell 97

6 Organisational Implementation 109
  6.1 Organisational developments 109
  6.2 Building Participants 112
  6.3 External Organisations 119
7 A Case: Precast Concrete Structures
   7.1 Organisational Implementation: Where to start? 123
   7.2 Integration Concept: a Project type Model 127
   7.3 Computer Implementation of the PtM 142
   7.4 A Test in Practice 146

8 Evaluation
   8.1 Integration Concept 151
   8.2 Computer Implementation 153
   8.3 Organisational Implementation 156

9 Conclusions and Recommendations
   9.1 Conclusions 157
   9.2 Recommendations 160

References 163

Author Index 173

Subject Index 175

Samenvatting 177

Curriculum Vitae 183

Bibliography 185

Appendix A The BPM in PMshell 187

Appendix B Project type Model for EKON structural systems 193
Summary

In the last decades, the building industry has faced an increase of demands from society. There is a demand for higher quality, shorter lead times, custom-made buildings, more environmental consideration, and better working conditions. This increase of demands is likely to continue.

One of the answers of the building industry on these demands might be to improve the integration of the design and construction processes. Integration of sub-processes in a building project can be defined as "the continuous and interdisciplinary sharing of goals, knowledge, and information among all project participants" [Fischer 1989]. Design for Construction (DfC) deals with integration of design and construction. DfC is the building industry's counterpart of well-known technologies such as Design for Assembly and Design for Manufacturing in the mechanical industry.

Since design and construction are increasingly, and successfully, supported by computers, it seems logical to support their integration with computers as well. For Design for Construction this leads to Computer Aided Design for Construction (CA DfC), which is part of Computer Integrated Construction (CIC). CIC and CA DfC are still technologies in their infancy. This PhD research aims a strategy to help CA DfC grow to a more mature state.

This research looks at the strategy for CA DfC from three related points of view:

- integration concept,
- computer implementation, and
- organisational implementation.

The integration concept starts from the presumption that for integration of building processes, their computer applications have to be integrated. To integrate computer applications, the information and knowledge used have to be sharable. To realise this, formalisation and exchange of information and knowledge have to be standardised.

Technical implementation of the integration concept requires computer support. Among others, support is needed for developing the standards and for using those standards in practice for exchange between computer applications.

Before the participants in the building industry will start applying CA DfC, they want to know what it has to offer and how they can reach it.
First, the thesis describes the current state of DfC. Then, the three chosen strategic viewpoints are worked out and tested in a case study. Finally, the results of the research are evaluated.

**Current state of Design for Construction**

The state of integration of design and construction in the current building process is characterised by the state of integration in six "interactions" between designers and constructors:

1. forward exchange of the building design,
2. feedback on the building design from construction,
3. backward exchange of constructors' information,
4. backward exchange of general constructability knowledge,
5. upstream shift of construction management tasks, and
6. downstream shift of design tasks.

Interactions 1, 2, and 3 cover the sharing of information, interaction 4 the sharing of knowledge, and interactions 5 and 6 the shift of tasks between design and construction. For each interaction both technical and organisational barriers hinder better integration of design and construction. Generally, the introduction of computers led to optimisation of sub-processes. However, communication between these so-called islands of automation did not change basically: in most cases participants in a building project still communicate with paper documents that have to be interpreted by humans.

**Integration concept for CA DfC**

The first point of view of the strategy for CA DfC is the integration concept. This concept standardises the formalisation and exchange of information and knowledge in a building project.

The integration concept for CA DfC is based on improving communication by using project models. Project modelling aims at formalisation and exchange of project information in building projects in such a way that computer applications can "understand" and use it directly. Based on related research, the integration concept is structured with a framework that uses the STEP product modelling approach [ISO/TC184 1993b] as much as possible.

STEP recently standardised modelling languages that can be used to formalise information structures (EXPRESS) and to exchange data between applications (the STEP physical file format). These languages ensure that computers can communicate using the same syntax.

However, for meaningful communication standardisation of the syntax alone is not enough. Also the words and their meaning (or semantics)
have to be agreed upon. This is dealt with in conceptual models. Conceptual models can be developed for any domain of knowledge. Tolman [1991] proposes to use three types of conceptual models to structure the modelling efforts:

- resource models,
- reference models, and
- project type models.

These conceptual models have different scopes and are related. Resource models are generally applicable for issues such as topology, geometry, material specification, and tolerances. A reference model is applicable for a type of industry. A reference model may use resource models. A project type model is applicable for a type of projects. A project type model is a specialisation of a reference model.

This research focuses on developing a reference model for the building industry. Resource models are already dealt with in the STEP community. The case study demonstrates the specialisation of the reference model to a project type model.

Besides standardisation of meaningful communication, EXPRESS also enables formalisation and exchange of simple constructability knowledge.

The reference model for the building industry is called the Building Project Model (BPM). It uses two abstraction mechanisms to structure information in a building project:

- a product-activity mechanism, and
- a concretisation mechanism.

The product-activity mechanism relates product information with activity, state, resource, and actor information. The model is based on: the logic of change of von Wright [1963], the IDEF0 process analysis method, and reference models found in related research.

The concretisation mechanism describes and relates the evolutionary states of building information in problem solving processes such as design and construction scheduling. It is based on the concretisation mechanism of the IMPPACT reference model [Gielingh and Suhm 1993]. It supports the separation of problem and solution, the divide-and-conquer principle, and the decomposition of objects.

Together the two mechanisms form the BPM: the concretisation mechanism is applied to product, activity and resource information, which are related with the product-activity mechanism.

**Computer implementation of the integration concept**

The second strategic viewpoint is the computer support of the technical implementation of the integration concept. Related research shows that development of conceptual models and their application in practice re-
quire computer support. This research investigates the possibilities to support development and application of conceptual models with an integrated project modelling system.

Based on an analysis of the needs of future users of project modelling, requirements for a project modelling system are specified.

The project modelling system should support the STEP languages EXPRESS and the STEP physical file format. For presentation and the early development phases, also a graphical conceptual modelling language such as NIAM should be supported. The project modelling system should also support a mechanism to relate conceptual models with different scopes. Finally, the project modelling system should support at least three groups of possible users: the developers of conceptual models, the programmers, and the end-users in practice. Each group of users has its own specific requirements. To ensure smooth co-operation between the three groups, the gaps between specification, implementation, and application of conceptual models have to be bridged.

In close co-operation with TNO, a prototype version of a project modelling system has been developed. This prototype system is called PMshell, from Project Modelling shell.

Because the object-oriented paradigm closely relates to conceptual modelling, PMshell is implemented in the object-oriented programming language Eiffel. It has a menu-driven user-interface for all users, and a graphical, NIAM-like user interface for the developers and programmers of conceptual models. It supports the STEP languages. PMshell supports the developer and programmer in specifying characteristics and behaviour of conceptual models per domain of interest. These conceptual models can be related. End-users can use the defined characteristics and behaviour to model information about a particular project. Developers and programmers are not only able to specify behaviour for end-users, but also for developers of other conceptual models. PMshell enables developers to define (constructability) rules and end-users to check whether their project model complies with these rules.

Representative parts of the reference model BPM are implemented with PMshell. To be able to implement the BPM, general issues, such as network functionality and shape definition, representation and presentation, are implemented. Concretisation of the product during design is implemented, including decomposition following the design process. For activities and resources, the relation with the product and its construction states has been implemented.
Organisational implementation of the integration concept

The third strategic viewpoint analysed in this thesis is the organisational implementation in practice.

In current practice, organisational developments are observed that might be supported with CA DfC. If participants want to adopt CA DfC, they have to use the semantic level of project models for communication between their applications from the beginning. Additional translators are needed that generate information at a lower semantic level from the project model for communication with participants that do not support the level of project models. For organisational implementation it is important that the building participants are actively involved and that the scale of communication grows step-by-step. External organisations such as umbrella organisations and the software industry might speed up this implementation of CA DfC. The umbrella organisations could work on the standardisation of conceptual models. The software industry could provide the building industry with modelling tools and computer applications that supports the semantic level of project models.

Application of the CA DfC strategy in a case study

In the case study, the CA DfC strategy is applied to a specific type of precast concrete structures for (office) buildings. Integration of design and construction for a precast element supplier is looked at from the three points of view considered in the thesis.

Following the recommendations for the organisational implementation a subject for demonstration has been selected, i.e., preliminary design and assembly on site of the highly standardised EKON system. This first demonstration is kept as simple as possible, but still shows many aspects of CA DfC. The BPM is specialised in a project type model for precast concrete structures. Product information used during preliminary design, and information on assembly states and activities, is included in the model. Also specific constructability rules are defined. The project type model is implemented using the BPM implementation in PMshell. With this implementation a project model is generated for design and assembly of a particular precast concrete structure in a test case.

Evaluation and conclusions

The case study demonstrated an implementation of CA DfC. It showed that the technical implementation is not easy, and that still a lot of work has to be done before the building industry can fully profit by CA DfC.

The case study showed how some relevant aspects of the interactions between design and construction can be supported with CA DfC. It seems that the integration of design and construction can be improved by sup-
porting the exchange (and formalisation) of knowledge and information between computer applications. The framework and Building Project Model are a step forward to CA DfC, but need extra attention before fully applicable in practice. The proposed way to formalise constructability was still of limited value. Evaluation of constructability can probably better be done in dedicated applications such as expert systems.

Application of PMshell in the research made it plausible that the technical implementation of the integration concept can be supported with computers. Although a good step in the right direction of a project modelling system, PMshell is not yet ready for application in practice. Among others, its robustness, user-friendliness, and flexibility have to be improved.

Because the case study focused on the integration concept and its technical implementation, not much can be concluded on the organisational implementation. The problems that showed up at the integration of design and construction in one company indicates already that integration over company borders will be extremely hard to realise. A more detailed migration route towards CA DfC in the building industry seems to be important and therefore needs further research.

As an overall conclusion can be stated that the strategy for CA DfC as proposed can contribute to better integration of design and construction. A number of the technical and organisational problems that currently hinder DfC in the six interactions between designers and constructors, can, at least partly, be overcome. The integration concept (with computer support) developed provides means to formalise and exchange information and knowledge between building participants. Whether these improved means of communication also will lead to sharing of goals, and ultimately, to better responses to society's increased demands, only future can tell.
List of Abbreviations

AEC    Architecture, Engineering and Construction
ANSI   American National Standards Institute
AP     Application Protocol
ARTB   Dutch technology policy advisory board for the building industry
B-rep  Boundary representation
BPM    Building Project Model
BSF    Bjelke-Soyle Forbindelse (Norwegian for beam-column connection)
CA Dfic Computer Aided Design for Construction
CAD    Computer Aided Design
CADr   Computer Aided Draughting
CAM    Computer Aided Manufacturing
CAM-I  Computer Aided Manufacturing-International
CASE   Computer Aided Systems Engineering
CIC    Computer Integrated Construction
CIFE   Center for Integrated Facility Engineering
CIM    Computer Integrated Manufacturing
CSG    Constructive Solid Geometry
CSTB   Centre Scientifique et Technique du Bâtiment (Centre for Building Science and Technology of France)
DFAT   Design for Assembly
DFC    Design for Construction
DFM    Design for Manufacturing
DXF    Drawing Interchange File Format of AutoCAD
EDI    Electronic Data Interchange
ESPRIT European Strategic Programme for Research and Development of Information Technology
FIP    Fédération Internationale de la Précontrainte (International Federation for Prestressed Concrete)
GARM   General AEC Reference Model
GenCOM General Construction Object Model
HVAC   Heating, Ventilation, and Air-Conditioning
ICAM   Integrated Computer Aided Manufacturing Programme of the U.S. Airforce
IDEF0  ICAM Function Definition Method
IDEF1X  ICAM Information Definition Method (Extended version)
IEC     International Electrotechnical Commission
IGES   Initial Graphics Exchange Standard
IMPPACT Integrated Modelling of Products and Processes using Advanced Computer Technologies, ESPRIT II project # 2165
IRDS   Information Resource Dictionary System
IRMA   Information Reference Model for AEC
ISO    International Organisation for Standardisation
KB     Knowledge-based
NC     Numerically Controlled
NIAM   Nijssen’s Information Analysis Methodology
PDES   Product Data Exchange Specification
PMSGQT PMSHELL Graphical Query Tool, a PMSHELL domain
PMSHELL Project Modelling Shell
<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIM</td>
<td>Project type Model</td>
</tr>
<tr>
<td>RATAS</td>
<td>Rakennusten TietokoneAvusteinen Suunnittelu (Finnish for Computer Aided Design of Buildings)</td>
</tr>
<tr>
<td>SBR</td>
<td>Dutch Foundation for Building Research</td>
</tr>
<tr>
<td>SDAI</td>
<td>Standard Data Access Interface</td>
</tr>
<tr>
<td>STEP</td>
<td>STandard for the Exchange of Product model data</td>
</tr>
<tr>
<td>TNO</td>
<td>Netherlands Organisation for applied Scientific Research</td>
</tr>
<tr>
<td>VTT</td>
<td>Technical Research Centre of Finland</td>
</tr>
<tr>
<td>WSN</td>
<td>Wirth Syntax Notation</td>
</tr>
<tr>
<td>XPDI</td>
<td>eXpert Product Data Interchange</td>
</tr>
</tbody>
</table>
1 Introduction

The research reported in this thesis tries to answer the question: “How can computers be used for the integration of design and construction processes in the building industry?” This chapter introduces this problem area and discusses the research goal and questions, and the research approach.

1.1 Introduction to the Subject of Research

This section discusses the relevance of the research subject. It argues that improved integration of design and construction processes may be one of the answers to the growing demands that society imposes on the building industry.

Challenges for the building industry

In the last decades, the building industry has faced increasing demands from society and the government. This increase in demands is likely to continue for the foreseeable future [ARTB 1993].

The demand for changes comes from different groups in society. Clients ask for higher quality and shorter lead times. Markets have opened up to international competition. Buildings situated in overcrowded cities become more complex. Also the scale of the buildings grows. On the other hand, individualisation, a trend both for people and companies, asks for customised buildings. This requires flexible building systems. People and companies near building sites do no longer allow disturbance of their normal way of living. Alternative, less disturbing, building methods have to be developed. Socially, the building industry has to improve working conditions of their personnel. And finally, the growing environmental awareness asks for better controlled construction methods and less harmful materials.

Not only society, also the government enforces changes. Traditionally, the government is a large client of the building industry. Increasingly, the government focuses on its core business: ruling the country. This means that it gradually withdraws from the engineering market. For example, the Dutch ministry of transport, public works and water management is
reducing engineering activities for civil structures. The ministry concentrates on the introduction of performance contracts and on controlling the building process. The actual building process, including engineering, is performed by third parties. The government also acts as the legal authority. It covers the demands of society on environmental impacts and working conditions with laws and regulations. It also changes laws on the legal organisation of the building industry. This is especially so in the European Community: because of anti-cartel laws and the open market requirement, the traditional way of tendering is changing.

Of course, people in the building industry try to adapt to these growing demands. They change their way of working. They acknowledge that companies have to evolve to face up to the imposed changes. There is a growing interest for quality of the product and the process. Building processes are increasingly supported with computer applications. Construction is becoming more industrialised by increased mechanisation on site and by extensive prefabrication of building elements in factories. The collaboration among project participants is changing with the introduction of building teams, co-makership, design-and-build projects, Total Quality Management, performance contracts and concurrent engineering. But these changes might not be enough to cope with the growing demands.

Integration as a possible solution

The building industry has always been fragmented and project oriented. As a result, most effort is put into the improvement of sub-processes, such as design and construction. Consequently, optimisation of the building process as a whole is lagging. A possible solution to bridge the growing gap between society's demands and building industry's supply, is the integration of sub-processes in building projects. Webster's unabridged dictionary defines "to integrate" as "to make whole or complete by adding or bringing together parts". Integration of sub-processes in a building project can be defined as "the continuous and interdisciplinary sharing of goals, knowledge, and information among all project participants" [Fischer 1989]. Currently, integration of the sub-processes is hindered because these processes are performed by different companies with different goals, and different knowledge and information representations.

The terms "knowledge and "information" deserve some extra explanation. Newell [1981] views knowledge from a computer perspective as "the competence to select actions to realise goals". Lucardie [1994] confines this goal-oriented view to knowledge of constraints, which is "a set of sufficient and necessary conditions which should be met by objects", and to knowledge of objects "needed to assess whether the objects comply

---

1 The original definition uses the terms data and partners for respectively information and participants.
with the constraints”. In this thesis knowledge of constraints is referred to as *knowledge*, and knowledge of objects as *information*. For example, constructability rules used by designers are viewed as knowledge, the data describing the design of a building as information.

**Design for Construction**

*Design for Construction (DfC)* deals with integration of two main building processes, i.e., design and construction. It also deals with related processes such as procurement, scheduling, and resource planning. According to Andreassen, Kähler, et al. [1988] *Design for Assembly (DfA)*, which is the manufacturing industry’s counterpart of DfC, will eventually lead to the improvement of:

- the profitability of manufacturing processes,
- the productivity of resources,
- the quality of products, and
- the working conditions for construction workers.

If similar improvements followed from DfC in the building industry, integration of design and construction would help bridge the gap between society’s demands and the building industry’s supply.

Although the term DfC is rather new, the notion of integration of design and construction is certainly not. After all, the main idea of design has always been to propose materials, shape, and a construction method in such a way that the proposed design can actually be built. This was true for Egyptians who built pyramids some 3000 years ago, as well as for builders of the most modern structures of today. The big difference is that the trial-and-error method is not accepted anymore.

Since the introduction of computers in the building industry, design and construction processes have been supported with numerous design, scheduling, and cost estimation applications. However, *integration* of design and construction has not really changed: goals, knowledge, and information are shared in the same way as before. Still, design and construction are performed by different companies with different, sometimes conflicting goals. Considering constructability aspects during design remains a matter of personal education and experience. And, because there is no way to formalise and exchange experience easily, designers do not always learn from earlier failures. It is questionable whether even experienced designers can oversee all consequences of design decisions on the construction process. In addition, the exchange of information has not really changed: in most cases the drawings resulting from the design process still have to be interpreted by humans before they can be used during construction management.
Clearly, better integration of design and construction is important for the building industry. However, it is not yet agreed how this can be achieved. A top-down approach would be to start with sharing goals, and then integrating knowledge and information. A bottom-up approach would be to start with sharing information, then knowledge, and finally goals. Although the goals of the building participants sometimes conflict, they always have to share information. Therefore, the bottom-up approach seems the most appropriate.

Since design and construction are increasingly, and successfully, supported by computers, it seems logical to support sharing of information and knowledge with computers as well. Integration of design and construction supported by computers is called Computer Aided Design for Construction (CA DfC).

Computer Aided Design for Construction

Integration of processes appears, at least, on three related levels (see figure 1.1) [AMICE Consortium 1989]. The first level of integration of design and construction is the organisational level. An obvious way to support integration of processes with computers is to integrate the computer applications used in these processes. This integration of applications is the second integration level. Computer applications can be integrated by making knowledge and information exchangeable and applicable. This is the third level of integration.

![Diagram](image)

*figure 1.1 three related levels of integration*
1.2 Introduction to the Research

In this section the main characteristics of the research are introduced: the goal and research questions, the research approach and its scope and focus. This section also introduces three strategic viewpoints for CA DfC: integration concept, computer implementation and organisational implementation.

Research goal and questions

As shown above, a possible answer to the growing demands from society might be improving the integration of the design and construction processes. Currently, designers and constructors do not always share the same goals, and knowledge and information can not be exchanged and applied easily without human interpretation. CA DfC might be a solution for these problems.

Because CA DfC is a technology in its infancy, the aim of this research is to help CA DfC grow to a more mature state. The notion of DfC is (and always has been) familiar to everybody in the building industry. Also, the idea of using computers for integration seems obvious. However, there is no generally accepted comprehensive strategy to realise computer aided integration. Thus, the research goal is formulated as follows:

Develop a strategy to support integration of design and construction in building projects with computers.

Because the goal of CA DfC is to integrate design and construction with the aid of computers, this research goal can also be formulated as follows:

Develop a strategy for Computer Aided Design for Construction.

The term “Computer Aided Design for Construction” has already been explained in the previous section. The word “strategy” is used to denote a long-term approach applicable for the building industry as a whole.

The three levels of integration of figure 1.1 in reverse order denote three possible points of view on CA DfC (see figure 1.2). Integration of information and knowledge requires a set of agreements about structuring and exchanging information and knowledge. These agreements can be combined in the first viewpoint: the integration concept. Implementation of the integration concept in practice requires computer support and an organisational plan. The second point of view focuses on supporting the computer implementation of the integration concept. The third point of view focuses on the organisational implementation of the concept.
The three viewpoints can be applied to the goal of this research: develop a strategy for CA DfC. This leads to three research questions to be answered:

1. What are the characteristics of an integration concept that supports CA DfC?
2. How can the technical implementation of the integration concept for CA DfC be supported with computers?
3. How can the integration concept for CA DfC be implemented organisationally?

The research focuses on the integration concept (research question 1) and its computer implementation (research question 2). Although an important aspect, the organisational implementation of computer-aided integration of processes (research question 3) only forms a small part of the research.

**Research approach and outline of the thesis**

The research approach to answer the research questions has four successive phases:

1. Study of the current state of (CA) DfC,
2. Proposal for a strategy to improve the current state of CA DfC,
3. Application of the proposed strategy in a case study, and
4. Evaluation of the proposed strategy.

In each phase the three points of view (integration concept, computer implementation and organisational implementation) are considered.
Figure 1.3 shows the four phases and their relations in an IDEF0 diagram. IDEF0 [SoffTech 1981] is a graphical process analysis technique often used in building research.

**Figure 1.3** *Research approach with four phases*

Phase 1 studies the current state of (CA) DfC in practice and research. What are relevant aspects for integration of design and construction? What are the technical and organisational barriers that stand in the way of DfC? How are these problems dealt with in state-of-the-art research? Chapters 2 and 3 answer these questions. This first phase has an exploratory character. Based on related research a direction for answering the three research questions is chosen.

Phase 2 proposes a strategy for CA DfC by answering the three research questions. This phase is the fundamental part of the research. Following the direction resulting from phase 1, a solution is proposed. Research question 1 on the integration concept is discussed in chapter 4. It describes a general model of information used in building projects. Chapter 5 answers research question 2 on the computer implementation. It describes the requirements and a prototype for a computer system that supports implementation of the general model. Chapter 6 answers research question 3 on the organisational implementation.

To validate the proposed strategy for CA DfC, it is applied in a case study. Phase 3, chapter 7, applies the strategy to design and construction of precast concrete structures of office buildings. The general integration concept is specialised to construction projects for precast concrete structures. This specialisation is implemented using the prototype computer system. The resulting integration system is demonstrated to the industrial research partners.
Phase 4, chapter 8, uses findings from the case study to evaluate the proposed strategy. This gives a first indication of the value of the proposed strategy for CA DfC.

Chapter 9 presents the conclusions on the future of CA DfC and recommendations for its further development and introduction into industry.

Scope and focus of research

Lack of integration in the building industry is too severe to expect one PhD research project to have any effect. Therefore, this research is part of a larger research project at Delft University of Technology, called "Computer Integrated Construction" (CIC). In this CIC project six PhD students work on different aspects of a general concept for integration of building processes [Tolman 1989; Waard 1992]. Partners in this CIC project are, among others, the general contractor Ballast Nedam BV and the research institute TNO Building and Construction Research. The precast concrete element supplier Spanbeton BV is closely related to the case study.

The scope of research is already expressed by the research goal: Develop a strategy for Computer Aided Design for Construction. Design for Construction focuses the research on the integration of the design and construction processes. Integrating the sub-processes of design, such as architectural, structural, and HVAC design, is out of focus, as is integrating the sub-processes of construction. Computer Aided focuses on those aspects that have something to do with computers. Changing design methods, construction methods, or organisational structures, is not the subject of this research. Strategy focuses the research on general, long-term issues applicable to the building industry as a whole.
Better integration of design and construction might answer the growing demands of society on the building industry. This chapter discusses the current state of integration of design and construction in today's building and construction practice. An analysis of the current situation produces a number of interactions relevant for DfC. These interactions are then discussed in more detail. The chapter ends with conclusions on the current state of DfC.

2.1 Analysis of the Building Process

Developing a strategy for CA DfC starts with analysing design and construction in the current situation. This section presents the results of this general analysis. The graphical process analysis technique IDEF0 is used, because it is easy to read and often applied in building research. Six relevant interactions between design and construction follow from the analysis.

The process analysis focuses on DfC: only design and construction during the realisation stage of the building life cycle are taken into account. Other life cycle stages, such as maintenance, operation, renovation, demolition, and retrofit, are not included. Neither included are policy making by the client and project management during the realisation stage.

The process model that results from the analysis combines common parts of several process models of building projects. These models are found in literature or developed in the course of this research. The Building Information Model (BIM) [IOP-Bouw 1989] contains an extensive process analysis of the realisation process of buildings. A process model for the realisation of office buildings at Ballast Nedam [Hoek 1991] describes the processes from the point of view of the general contractor. In the course of this research design, fabrication, and assembly of precast concrete elements have been analysed at Waco-Liesbosch Beton BV.

The process analysis focuses on the information and knowledge flows that relate design and construction processes. It has thus a restricted value

---

1 This process analysis has never been made public, because the department for custom-made precast elements was closed down during the research.
for other fields of interest. For example, the BIM offers a more complete process analysis. It should also be noted that the analysis is based on process models of building projects in the Netherlands only.

Figure 2.1 and 2.2 show the generalised activities and information flows relevant for DfC. Process analysis starts at the top-level. Figure 2.1 shows the realisation of a building with inputs (from the left), outputs (to the right), controls (from above), and mechanisms (from below).

**Figure 2.1** Realisation of a building

Realisation is controlled by project management information, realisation knowledge and building regulations. Input to realisation is both general and project-specific information. Information on products (or materials) and building methods is generally applicable. Requirements of the client, and information on the building site for instance are project-specific information. Realisation is executed by designers and constructors. They produce the building. They also generate intermediate results, such as building designs and construction schedules. Intermediate results and progress information are feedback for the project manager. During the realisation designers and constructors gain experience. This realisation knowledge can be used in future projects. Tangible output, such as the building itself and waste materials, is not included in the activity model.

Realisation of a building is detailed by dividing it into three sub-activities: design building, manage construction and construct building (see figure 2.2). This implies that realisation knowledge is also divided into three: design knowledge, planning knowledge and constructability knowledge. For readability purposes, the flows in parentheses in figure 2.1 are left out.
Realisation of a building project starts with design of the building by the designer. This activity includes feasibility study, conceptual design, preliminary design, final design and detailed engineering. Inputs are the requirements of the client and information on available products. The result is a building design, which functions as the control information for downstream activities. If the building, as specified in the design, is not constructable within time or financial constraints, the constructor gives feedback to the designer. In the figure this feedback is indicated with a dot.

The construction management activity includes procurement, scheduling, resource planning, and checking of schedules and plans during construction. Inputs are: requirements of the client, project-specific information on the building site, and general information on building methods. Construction management generates information such as the building method, schedules and resource plans. The activity is performed by the constructor.

The last activity, construct building, is also performed by the constructor. The results of design and construction management control this activity. Progress information of construction forms feedback for construction management.

The two IDEF0 mechanisms in the figure, designer and constructor, represent the people that fulfil the two main tasks in this part of the building process. Designer stands for all people that make proposals for the functions, shape, and materials of the building: architects, structural engineers, HVAC advisors, mechanical engineers, etc. Constructor stands
for all people that build the building, or manage the construction process. These two groups of people may work for architectural offices, engineering offices, main contractors, sub-contractors, suppliers, etc. For this general analysis of the activities in a building project, it is not relevant in what companies these two groups of people work.

In addition to the project-specific results, each activity also generates general knowledge, or experience. This knowledge can be used in future projects.

Six interactions between design and construction are derived from this analysis. These interactions are characterised by the exchange of information, knowledge and tasks between designers and constructors. The exchanges can be seen in the current situation, or are desirable improvements. The interactions relevant for DfC are (compare the numbers in figure 2.2):

1. forward exchange of the building design,
2. feedback on the building design from construction,
3. backward exchange of constructors’ information,
4. backward exchange of general constructability knowledge,
5. upstream shift of construction management tasks, and
6. downstream shift of design tasks.

When an interaction is not covered by an existing flow in the figure, a new flow is added with a dotted line. This is done for interactions 3, 5 and 6 that are hardly seen in current practice.

Interactions 1, 2 and 3 cover formalisation and exchange of information. Interaction 4 covers formalisation and exchange of knowledge. Interactions 5 and 6 cover the shift of tasks. In this thesis, to formalise means to structure and record information and knowledge in such a way that computers can access and use it. Communication is viewed as the exchange of information and knowledge. Formalisation of information and knowledge is needed to enable communication between the computer applications of the participants.

2.2 The Interactions of DfC

This section elaborates the six interactions of DfC that followed from the process analysis in the previous section. For each interaction three points of view are discussed: the current situation and the technical and organisational barriers that hinder better integration of design and construction. The focus is on the communication media and tools used.
Interaction 1: Forward exchange of the building design

The designer generates information on the building in the form of a building design. This design is used during construction. Information mainly concerns building products and materials. Product information is recorded in specifications, drawings and other documents. Increasingly, drawings are made with Computer Aided Draughting systems. This allows exchange of drawings in a digital exchange format. Examples of digital exchange formats are IGES (Initial Graphics Exchange Standard of ANSI and ISO) [Reed 1991] and the de facto standard DXF of AutoCAD [Autodesk 1990].

The building industry started to use computer applications mainly to support existing processes. Therefore, these applications focus on generation of the same communication media as before: detail drawings, tables and other documents. These documents have to be interpreted by humans before their information can be used in other applications. Another technical barrier that stands in the way of electronic information exchange is the low penetration of inter-corporate electronic networks in the building industry. This means that company borders can practically only be crossed with paper documents.

Forward communication on the building design is only partly formalised, and happens often on a low semantic level\(^1\). In the tender stage communication about the product is very formal with an exact description of the building in the specifications. Communication after this stage is rather informal: changes and instructions are communicated ad hoc. Often communication on the management level is more or less formalised. However, internal communication from management to the people who really do the work is often the bottleneck. Despite management agreements, construction planners often receive detailed design information only shortly before construction starts. This hinders a good preparation. The organisational structure of the building industry further encourages communication on a low semantic level. The low-level documents are often prescribed in building codes, and are therefore an important part of the legal structure. The ever-changing combination of participants makes it difficult to agree upon (and implement) a higher semantic level.

Interaction 2: Feedback on the building design from construction

During construction management and during construction, feedback information is generated on the building design. The proposed design may not be realisable with the available resources, or the actual construction

---

\(^1\) Communication on a low semantic level means communication with documents that have to be interpreted by humans before their information can be used in other processes. A typical example is communication with 2D drawings.
result may differ from the proposed design. In other cases, the constructor has better alternatives, which have to be approved by the designer. The constructor reacts on the finished design.

Currently there is no formalised way to exchange comments or suggestions from construction to design. This implies that the most frequently used instruments are fax, telephone and meetings. Fax and telephone instruments are ad hoc communication tools. Also change control and version management are typically not formalised. Especially in large building projects, meetings have a legal value. However, they mainly formalise communication on the management level, not on the execution level.

An organisational barrier is that the constructor, by making a bid, has to realise the product as described in the tender (or, a product with equal functionality and quality). Also the time gap between design and construction causes problems: the designer of the building is often already working on another project at the time the building is under construction. In the Dutch building industry, supervision of construction is often done by the architect and is thus closely related to design. As seen before, internal communication is often the bottleneck. Concurrent engineering, in which processes are performed rather in parallel than sequentially, might resolve some barriers mentioned above. With concurrent engineering, constructors are already working on the project in early design stages. There is more communication. Therefore, design choices are better thought through, and design changes can be met more easily. The building team is an example of concurrent engineering already applied in practice. A problem in concurrent engineering is legal liability: when constructors advise on design choices they become (partly) responsible for the design. Often participants in a building team do not share the financial risks. This implies that a building team mainly supports co-ordination of information flows between project participants, not integration with shared goals.

Interaction 3: Backward exchange of constructors’ information

When constructors’ information is taken into account during design, comments on the design can be (partly) avoided. Therefore, the constructors should provide information before design choices are made. This constructors’ information mainly concerns availability and cost information of resources, and conditions of the building site. The information is specific for a constructor and often also specific for one project. In the current situation this exchange of information usually only happens if design and construction are performed by one company.

The technical exchange mechanisms for this interaction are the same as for the second interaction. Therefore, the technical barriers are the same too.

A big organisational barrier is that the constructors are usually not yet known at the moment of design choices. Another barrier is that compa-
nies are reluctant to share internal information with other companies, at least, before having a contract. These other companies might be competitors in future projects. Finally, there is the barrier of different interests: designers try to minimise design cost, constructors construction cost. They do not try to minimise the total project cost. When this interaction is not supported, the constructor has to use the feedback interaction: he\(^1\) comments afterwards or proposes alternatives.

**Interaction 4: Backward exchange of general constructability knowledge**

In addition to project-specific information, general constructability knowledge must also be formalised and exchanged for use during design. This type of knowledge mainly consists of instructions for the design of specific types of products. General knowledge originates from experiences in earlier projects and therefore requires a long feedback loop.

The instruments currently used for exchange of constructability knowledge are not formalised. The main instruments are education and technical literature. Another completely informal instrument is the experience stored in the minds of designers and constructors. Sometimes this personal knowledge is implemented in (home-made) computer programs.

Exchange of general constructability knowledge is troubled by the lack of formalisation, and by the time gap between design and construction. Constructability rules are hardly ever prescribed in codes. For other building aspects official organisations exist, which prescribe codes for designers, and which check the designs. Examples are fire safety and structural behaviour. Something similar does not exist for constructability. In the contrary, constructability rules only exist in the heads of people. When those experienced people leave the company, their knowledge fades away.

**Interaction 5: Upstream shift of construction tasks**

Another way to integrate the two processes is to shift tasks between the participants. For example, construction tasks can be shifted upstream from a constructor to a designer. When the designer does part of construction management, he will get deeper understanding of construction consequences of his design choices. Scheduling will give him understanding of the time aspects of his choices, cost estimating of the financial aspects.

\(^1\) Note of the author: While I want to recognise the important contributions women make to the building industry, I use the masculine gender when I talk about individuals. This does not imply a gender bias or preference in any way. It only simplifies the reading.
To simulate the construction process, the designer needs construction applications that are integrated with design applications. Many designers use design applications, and many scheduling and cost estimating applications are available on the market. Today, designers are not able to simulate the construction process easily with computers. Designs are often only interpretable by humans. And, most scheduling and cost estimating applications require human input, or use non-standardised input formats. This means that the designer is not able to use construction applications on-line. Therefore, currently available tools do not help to bridge the time gap between when design decisions occur and when the consequences of the decisions manifest themselves.

The designer also needs constructors’ information to simulate the construction process. Because of the conservative nature of the building industry and the mistrust between partners, constructors are reluctant to pass responsibility and internal information to designers. Less responsibility means less influence, and probably less money. Internal information can be used against the constructor in future projects, when the designer might participate in a competing project team. The constructors are not aware that only part of the input for scheduling and cost estimating is directly derivable from the design. The skills of planners and procurers will always be needed for the final schedule and the final cost estimation.

Interaction 6: Downstream shift of design tasks

Likewise, it is possible to shift design tasks to constructors. In that case, design choices are made by constructors. The resulting choices are probably better suited for construction. This implies that the designer prescribes requirements and limitations for the solution. The constructor develops his own solution that fits best with his capabilities.

Technically, the problem is how to communicate requirements. Currently, communication is focused on solutions. Officially, constructors always have the right to offer a solution equivalent to the proposed design in the tender. But, it is difficult to find out what is equivalent if one does not know the requirements that form the basis of the proposed design.

Furthermore, designers are reluctant to share tasks and responsibility with constructors. They are afraid to lose control over the design process. They are responsible for co-ordination between design disciplines. How can designers control this co-ordination, when constructors do the design? In addition, when the constructor redesigns part of the building, the client pays twice for design. Changes are on the way with the introduction of performance contracts in the tender. In a performance contract the design is not specified in full detail, only the requirements for a design are specified. The design is then made by the constructor.
2.3 Conclusions on DfC in Current Practice

The technical possibilities and organisational structure in current practice, as described in the previous section, limit integration of design and construction. Chapter 1 argued that supporting the communication between participants with computers might be a solution. This section argues that the commonly used means of communication are not suited for computer support.

Since the introduction of computers in the building industry, design and construction processes have been supported with numerous design, scheduling, and cost estimating applications. The foregoing discussion of the six interactions of design and construction showed that integration of design and construction has not really changed: goals, knowledge, and information are shared in the same way as before. Communication between the computer aided processes has not changed: the applications still produce traditional documents, such as specification documents, detail drawings and schedules. These documents, on paper or electronic, have to be interpreted by humans before their information can be used in the next application (see also figure 2.3).

![Diagram showing interactions between design, drawing, and scheduling applications](image)

*figure 2.3 communication with documents as often seen in current practice; humans perform the communication by interpreting documents*

As discussed in section 1.1, CA DfC aims at improving integration of design and construction by improving the integration of the applications used during these processes. Most computer applications cannot interpret the information of the (electronic) documents like humans can. Applications need information with more explicitly defined meaning, or, in other words, information at a higher semantic level. High-level communication
in a format that is computer-interpretable minimises the need for human interpretation and enables integration of applications (see figure 2.4).

**Figure 2.4** standardised, high-level, computer-interpretable communication; humans control the communication

Computer-interpretable communication requires standardisation agreements. The building industry is too fragmented to make those agreements on the project level: the composition of building teams change too often to make standardisation agreements economically feasible for one group of participants. Therefore, an agreement on the industry level is required, supported by all participants, with all their applications, and preferably on an international scale. However, an industry-wide agreement for the building industry is very difficult to realise.

As the previous section showed, integration of design and construction is not only limited by the technical possibilities, but also by organisational barriers. As is often the case, the core issue is money. Participants ask about cost and benefits of changes. When design and construction are integrated, they have to share information, knowledge and goals. They have to take the requirements of others into account. These changes do not happen easily. Besides, what will they get in return?

To overcome the organisational barriers the six interactions of DfC have to be looked at together. Supporting one interaction separate from the others will not lead to DfC. It is impossible to formalise all constructability knowledge. It is also impossible to shift all design tasks to the constructor. Design is a specialist’s discipline. Co-ordination between design disciplines is too important to be left to constructors. It is also impossible to exchange all project-dependent constructors’ information to design before design decisions are made. It should therefore always be possible for constructors to comment upon design choices and to suggest alternative designs.
A common issue in the six interactions of DfC is formalisation and exchange of general and project-specific information and knowledge. Improvement of this communication issue is both a technical and an organisational challenge. When technically feasible, improvement of communication can have important organisational consequences, such as a shift of tasks, more co-operation, and sharing of information and knowledge. The building industry has to anticipate these consequences.

At the moment there is no comprehensive concept supporting the six interactions of DfC with computers. A concept that supports formalisation and exchange of information and knowledge on a high semantic level can mean a step in the direction of DfC.
3 Related Research

The technical and organisational barriers that limit integration of design and construction, as discussed in chapter 2, have been the subject of other research projects. This chapter discusses some of the related research. Especially STEP-based product modelling seems to provide a basis for formalising and exchanging building project information. The three sections of this chapter correspond to the three research questions. Per section conclusions are drawn on related research and a direction for answering the research question is given.

3.1 Related Research on Integration Concepts

An overall concept for CA DfC has not yet been extensively researched. Therefore, the scope of this section is enlarged to integration concepts for design and production in other industries, and to integration of computer applications in general.

Integration of design and production in other industries

Other industries, especially the mechanical industry, are already working on computer integration of design and production. Important concepts are Design for Manufacturing (DfM) and Design for Assembly (DfA), standardisation of geometric information, and feature modelling. The following briefly summarises these concepts and their possible value for CA DfC.

Design for Assembly (DfA)

One way to improve integration of design and production is backward exchange of production knowledge (interaction 4 in section 2.2). In the mechanical industry this is covered by DfA and DfA-like concepts.

As a formalisation of knowledge, Boothroyd and Dewhurst [1987] developed a quantitative assemblability evaluation method, based on an empirical study of cost associated with manual, robot and automated assembly processes. They suggest that the best way to reduce cost is to reduce the number of parts to be assembled. They say that it is only necessary to
introduce a new part: if that part has to move relatively to other parts, if it is necessary to use different materials, or for service and maintenance reasons. After consideration of these criteria for part-count reduction, a worksheet is used to determine the assemblability in terms of cost. This worksheet, computerised or on paper, forms the basis for comparison with other alternatives.

Another method for the evaluation of assemblability was developed by Hitachi [Miyiwaka and Toshirjo 1986]. This assemblability evaluation method helps to identify the weak points of a design. It prescribes a method to calculate the values of evaluation indices that can be compared with target values. If deemed necessary, design can be improved. The method gives indications where to find unfavourable parts or sub-assemblies in the design. As a consequence the assemblability evaluation method leads to new design alternatives.

In the field of DfA and Flexible Production Automation (FPA) Andreassen et al. [1987; 1988] worked on design rules for ease of assembly and flexibility in the mechanical industry. They describe rules for identification, orientation, transportation, storage, composing and quality checking of assemblies. They state that DfA and FPA ask for higher quality and better quality control, (flexible) mechanisation and automation, design for standard equipment, avoidance of variants, uniform variants and a good working environment.

The DfA concepts seem to be only applicable to one interaction of design and construction, i.e., formalisation (and use) of constructability knowledge. They do not deal with communication of information, and thus the other interactions are neglected. Application of the concepts is also limited to the industrial production of relatively small products. The concepts are only applicable for individual types of building products, e.g., for the assembly of precast concrete elements on site. Nevertheless, the idea of defining rules for ease of construction and the idea of developing constructability evaluation applications seems to be useful for the building industry.

**Standardisation of geometric information**

Formalisation and exchange of geometric (and topological) information was probably the first effort to improve communication between (mechanical engineering) computer applications. An important development by Computer Aided Manufacturing-International (CAM-I) was the Application Interface Specification (AIS) [Ranyak 1991], a standard for the exchange of solid models (CSG and B-rep).

Standardisation of geometric information as seen in the mechanical industry is one way to improve communication. The Application Interface Specification of CAM-I is an international standard, it is independent from
software vendors, and it is computer-interpretable. However, the semantic level of communication with detailed geometry descriptions is still not high enough for the building industry. Besides, solid models are hardly ever used in building practice.

**Feature modelling**

A concept, developed in the mechanical industry, with more emphasis on communication on a higher semantic level, is feature modelling. Wilson and Pratt [1986] define a feature as "a region of interest in a mechanical part". Figure 3.1 shows some examples of features. A feature can be, e.g., a through hole, a protrusion, a depression, an area, or a compound of simple features. It can also be a modification of a corner, edge or face.

Designers and producers may use different features on the same part. A designer can use features to store design information, such as strength requirements, measures and tolerances. A producer can use features to store production information, such as production sequence and selected tools. Geometry of the part can be derived from the feature definition. DfA can also be supported by features, assembly features in that case.

![Diagram of a simple mechanical part with features](attachment:image.png)

**Figure 3.1** Cross-section through a simple mechanical part with some typical features [Wilson and Pratt 1986]

Using feature modelling for formalisation and communication of information on mechanical parts is a step in the direction of high-level, computer-interpretable communication. It enables integration of design and production information and knowledge. It also reduces the amount of data to be exchanged. The notion of feature modelling is certainly relevant for the building industry too. However, it needs adaptation to the more complex nature of building products and construction processes.
Integration concepts for the building industry

The current state of DfC in practice, described in chapter 2, showed the need for high-level communication to integrate applications. This conclusion is strengthened by observations in other industries. DfA-like concepts require high-level communication for integration with other applications. Feature modelling showed its value for integration of design, process planning, programming of numerical controlled machines, and inspection of mechanical parts. Information and knowledge exchange (or communication as called in section 2.1) is seen as a basis for integration. It is therefore important to look at general concepts for communication reported or under research in the building industry.

There are at least six ways in which communication between computer applications is pursued:

- closed integrated systems,
- open integrated systems,
- communication with low semantic representations,
- classification and coding,
- product modelling, and
- knowledge-based technologies.

Closed integrated systems usually consist of vendor-specific suites of application programs. Communication between the applications is supported by a vendor standard. Communication outside the system is not really supported. An example is the CAD/CAM-system of the Intergraph Corporation [1988].

Open integrated systems provide their users with a “standard” user and database interface that supports communication between different application programs embedded in the integrated system. Examples are ICES and GENESYS. Information exchange outside the integrated system is not supported.

Another approach is to limit the semantic level of communication. An example is a limitation to geometric drawing information, as done in IGES [Reed 1991] and DXF [Autodesk 1990]. A disadvantage of communication on such a low semantic level is that still human interpretation is needed before information can be used in other applications.

Classification of project information aims at communication on a higher semantic level. The SfB\(^1\) classification methodology [Ray-Jones and McCann 1971] originated in Sweden in 1945, and is applied to national and international classification standards after that. An important classification organisation is the International Organisation for Standardisation (ISO) [ISO/TC59 1993]. ISO defines classification as “a set of concepts\(^2\)

---

\(^1\) SfB stands for Samarbetsskomitéen för Byggnadsfrågor, a Swedish committee for standardisation in the building industry.

\(^2\) In this ISO document concepts are resources, processes, and results.
arranged systematically according to chosen characteristics or criteria. Classification divides objects of interest in hierarchical categories, or classes. The relations between classes are often specialisations, or is-a relationships. For example, a precast concrete beam is a concrete element. Classification can be used as a basis for standardisation and coding of building products, activities and resources. Classification (and coding) enables communication of building project information in a standardised, computer-interpretable way.

Product modelling is again a way of communicating on a higher semantic level. Product models are used as communication medium, like detail drawings in today's practice. In a product model, information on the product is stored as objects with attributes, representing the product and its properties. These properties also include relations between objects that can not be formalised with classification. Information of the product is stored in a standardised format, independent of the applications that may use it. A product model can be stored in a central database. Product modelling supports formalisation of information during the whole building process. The major standardisation effort in the field of product modelling today is ISO standard 10303, also known as the international STandard for the Exchange of Product model data (STEP [ISO/TC184 1993b]).

Knowledge-based (KB) technologies (or the concept of expert systems) put the formalisation and usage of knowledge central. Knowledge on, e.g., a product is implemented as rules to which the product information should comply [Lucardie 1992]. In an expert system, knowledge of a specific domain is formalised to support decision making. The development of expert systems is often supported with shells. An expert system requires input (and generates output) in the terminology of the expert. To integrate expert systems with other applications this high-level input and output must be supported.

The six ways of communication between applications show a trend towards open (vendor independent) communication on a high semantic level.

Closed and integrated systems are not open enough: communication with application outside the system is not supported. In Europe and the USA no market leader has yet been able to prescribe their high-level "standard" to the building industry. Also, low-level communication reaches its limits when communication with another type of applications is pursued. Because only the presentation of the final result of a process is communicated, information fades away. For example, a drawing does not contain the reasons behind the design decisions. This low-level information still requires human interpretation, which is slow, costly, and open to misinterpretation.

Classification of information is a step in the direction of standardised communication on a higher level. However, it often lacks means to communicate other than is-a relations between objects. Product modelling
aims at supporting open, semantic-rich, computer-interpretable communication. Product models use the natural terminology of the user in practice, and thus offer the high-level information required to define knowledge rules in KB technologies and to integrate ES systems with other applications.

Product modelling builds on the strengths of the other integration concepts. Classification of information has proven its value in practice. Its division in categories and its terminology can be the basis of product models, as argued by Björk and Penttilä [1989]. Product modelling can also be seen as an evolution of feature modelling for mechanical products to the more complex nature and larger scale of building products. With its potentially high-level, open and standardised communication, product modelling may well form the basis for CA DfC.

**Product modelling**

For the realisation of CA DfC, which requires standardised high-level communication between applications, product modelling seems very promising. Therefore, the following looks at product modelling in more detail.

**History of product modelling**

In the seventies and eighties, CAD systems were considered to be the central points for integration. Models for the exchange of shape information grew more and more sophisticated, from wire frames and surface models, via solid models (e.g., CSG and B-rep), to relational reference models. However, it turned out that topology and/or geometry cannot be used as the central point of integration because (as often argued, e.g. by Dupagne [1991]):

- the shape of a product is not stable during the design process,
- there often is already information before a shape is chosen, and
- different participants in the project often use different shape representations.

Therefore, the geometric modelling concept evolved to the concept of product modelling, in which the building model is described with the semantics used in practice. A product model contains product information such as the properties of its parts. This includes geometry, topology, decomposition, material, behaviour, etc. The product model contains information for all life cycle stages and for all participants in the building process. Geometric information is not the core anymore, but one of the properties. Traditional documents, such as drawings, can ultimately be derived from the product model.
An important aspect of product models, is the separation of three seman-
tic levels of information: definition, representation, and presentation, as
described by Tolman and Gielingh [ISO/TC184 1989]. The definition of a
product contains product information on shape, material, behaviour, etc.
in a neutral format. Neutral here means independent of the computer
applications used. The representation of a product contains part of this
neutral information in a format suited for computer manipulation. A rep-
resentation is derivable from the definition. The presentation of a product
contains part of the neutral information in a format suited for human in-
terpretation. A presentation is derivable from the representation and def-
inition. For communication, ideally only the definition needs to be ex-
changed, because the other two can be derived and created on the fly.

For example, a steel beam can be defined as IPE400 with a length of
3.00 meters. The representation of the shape of this product in a CAD
system can be a B-rep with vertices, edges, faces, a volume, and topologi-
cal relations. A CAD system can manipulate this shape. For instance, the
volume of the beam can be calculated. The presentation of this shape for
human use can be a 3D picture on a CAD screen, or a set of (electronic) 2D
drawings of top and front views. The 2D drawings can for instance be
used during construction on site. Exchanging the definition is usually
enough to provide the information needed for computer manipulation and
human interpretation.

ISO-STEP

The major standardisation effort in product modelling today is ISO-STEP,
the international STandard for the Exchange of Product model data. The
purpose of this international standard is to specify a format for the un-
ambiguous definition and exchange of computer-interpretable product in-
formation throughout the life of a product [ISO/TC184 1993b]. The de-
development of the standard started in 1984 and builds on the Product Data
Exchange Specification (PDES) [Smith 1986]. PDES is a development of
the American National Standards Institute (ANSI) and is closely related to
the Initial Graphics Exchange Standard (IGES).

The STEP approach originated from an architecture of PDES which con-
sists of three layers [Smith 1986]:

- application layer,
- logical layer, and
- physical layer.

This architecture corresponds to the three schema architecture of
ANSI/SPARC [1985]. In this architecture schemata in the three layers are
called respectively external schemata, conceptual schemata, and internal
schemata. A conceptual schema defines the information independently of
users' views and implementation. An external schema defines (part of)
that information in the way users look at it. And, an internal schema defines the information in the way it is stored and manipulated by computers. In this way the conceptual schemata in the logical layer are independent of changes in the way users view the information and in the way data is stored.

For a standardisation body such as ISO-STEP the logical layer is most important, because it is view and application independent. For this logical layer, STEP languages have been developed for specifying data structures and for exchanging data. The value of the STEP languages can be measured by their wide-spread use, even outside the STEP community.

For specification of information structures the lexical modelling language EXPRESS has been developed. Recently, the first version of EXPRESS has been officially standardised in an international standard [ISO/TC184 1992a]. A sub-set of EXPRESS has a graphical presentation: EXPRESS-G. Other graphical presentations that may be used in STEP are NIAM [Nijssen and Halpin 1989] and IDEF1X [APPLETON COMPANY 1985]. Because EXPRESS schemata are computer system independent and have a standardised format, they can be easily exchanged between participants.

An EXPRESS schema defines entities and their attributes, which denote sets of objects with common properties. A database can be structured according to an EXPRESS schema. In this database, information on the objects of one particular product (of that type of product) can be stored. Information in the database can be exchanged with a file or via standardised access to the database. For file exchange STEP developed the STEP physical file format [ISO/TC184 1993d], a draft international standard. For standardised access to databases the Standard Data Access Interface (SDAI) is in development [ISO/TC184 1993e].

An EXPRESS schema contains, e.g., the specification of an information structure for a type of product. In this thesis the specification of an information structure is called a conceptual model and the sets of objects defined by EXPRESS entities are called object types. Object types correspond to classes used in classification. Information on a particular product (like the White House) that is structured according to a conceptual model is called a product model.

Next to defining the modelling languages, the STEP community focuses on standardising conceptual models with different scopes. The basic idea is that there are two types of conceptual models [ISO/TC184 1993b]:

- integrated resources, and
- application protocols.

Integrated resources define related parts of conceptual models that can be used in other conceptual models. There are two types of integrated resources: generic resources and application resources. Generic resources are application (type) independent. Examples are: geometry and topology
representations, product structure and configuration management, dimensions and tolerances, and form features. Application resources are applicable to specified ranges of applications, like draughting, finite element analysis, and presentation. An Application Protocol (AP) specifies the representation of product information for one or more applications. The definition of "application" in this context is rather broad: "a group of one or more processes creating or using product data.". This means that there are application protocols for general processes, such as 2D-draughting, but also for specific processes, such as ship or a road design.

Although the STEP methodology has undergone already several refinements, relating resources and APs still causes problems. Generic resources may reference each other. Application resources may not. An AP uses (parts of) the integrated resources. Different proposals circulate how to manage these relationships. Also inter-operability between APs and the notion of formal subsets of APs is still a largely unsolved issue.

Despite of the unresolved issue of relating the conceptual models, many resources and APs are already under development. The development of APs is rather bottom-up, driven by the needs of the applications in question. Until now only an application protocol for exchanging mechanical and civil engineering drawings has been agreed upon [ISO/TC184 1993a; 1992b]. In the near future however dozens of new resources and APs will be submitted for evaluation and integration.

Though still in development, STEP based product modelling has much to offer. The STEP modelling languages are of great value because they are standardised and widely used. Resource models and APs, standardised in STEP, seem to be promising for the future, because only an international (ISO) standard will be widely implemented by the vendors.

The following describes related research that might offer useful extensions to the STEP approach on the issues of:

- relating modelling languages,
- relating conceptual models,
- structuring product information using decomposition,
- extending product modelling to project modelling,
- extending product modelling with KB-technology, and
- relating the above mentioned issues in a framework.

Relating modelling languages

The language issue is currently researched thoroughly by a joint ISO-IEC technical committee for information technology. IEC stands for the International Electrotechnical Commission. One of the results is the Information Resource Dictionary System (IRDS) framework [ISO/IEC 1990]. This framework consists of four so-called data levels (see also figure 3.2):
- IRD definition schema level,
- IRD definition level,
- Information Resource Dictionary (IRD) level, and
- application level.

![Diagram showing data levels in the Information Resource Dictionary System (IRDS) framework](image)

**Figure 3.2**  

The bottom level is closest to daily life practice: the application level contains actual product data. The types of this data are defined in application schemata (or conceptual models as they are called in this research) on the IRD level. These application schemata are defined by IRD schemata on the IRD definition level. These IRD schemata are defined by the IRD defi-

---

1 Note the difference between the use of the word *application* in the original STEP (and PDES) architecture and in the IRDS framework. In STEP the application layer contains conceptual models that represent application views on data structures. In IRDS the application layer contains actual data.
nition schema on the top level. This IRDS framework corresponds closely to the four levels of data description of ANSI/SPARC [1985].

The STEP modelling languages fit in this framework (see figure 3.2). The STEP physical file format and SDAI are used at the application level. An EXPRESS schema at the IRD level defines (object) types that can be used in a STEP physical file. Because SDAI combines the definition of object types and the instantiated objects, retrieval of application information is independent of the way data storage is implemented. At the IRD definition level the EXPRESS syntax is defined in the international standard. The syntax is defined using the Wirth Syntax Notation (WSN) [Wirth 1977] and the meaning of the tokens that can be used is defined using natural language. At the IRD definition level also mapping with other conceptual modelling languages can be defined, such as mapping with NIAM and IDEF1X. The Wirth Syntax Notation is defined at the IRD definition schema level.

Relating conceptual models

As described above, one of the issues of the STEP product modelling approach is to relate conceptual models. A solution proposed is to distinguish between reference models and product type models.

As one of the first efforts to develop a comprehensive product model for the building industry, Gielingh proposed the General AEC Reference Model (GARM) [1988]. The GARM can be used to model a product and its characteristics in different life cycle stages (see figure 3.3). The GARM contains information structures not included in the modelling languages that support decomposition, evaluation and selection of alternative solutions, connectivity, shape representation, and classification of building aspects.

Originally, models like the GARM were intended to be instantiated directly as product models, representing one particular product. However, soon the need for more product type specific conceptual models emerged.

Tolman [1991] proposed the introduction of product type models structured according to a common reference model called the Integration Core Model (ICM). Product type models contain information structures valid for one family of products only. The ICM was a GARM-like reference model, but now intended for the development of uniformly structured product type models. The terminology used to define a product type model closely follows the terminology commonly used for product descriptions in current practice. Tolman suggested to use three types of conceptual models, each with its own scope:

- general resources,
- reference models, and
- product type models.
General resources correspond to the STEP integrated resources and are applicable for several types of industry. A reference model contains information structures applicable to a whole type of industry. For example, the GARM can serve as a reference model for the building industry. A product type model is a conceptual model for a specific type of products, e.g., for office buildings.

Two types of relations are possible between conceptual models: reference relations and specialisation relations. A reference relation denotes that one conceptual model uses part of another conceptual model. For example, a reference model references the geometry resource model for the definition of shape. A specialisation relation denotes that one conceptual model adds specific information structures to another conceptual model. For example, a conceptual model for office buildings adds specific building information structures to the GARM.

After this interpretation, the GARM has often been used as a reference model for product type models. Examples are: (1,2) the Road Model

---

1 Originally, the GARM was presented using the modelling language IDEF\textsubscript{X} [Appleton Company 1985]. In this thesis all conceptual models are presented in NIAM diagrams [Nijssen and Halpin 1989].

This architecture with conceptual models with different scopes has also been used in the IMPPACT reference model [Gielingh and Suhm 1993]. The IMPPACT reference model has been successfully specialised in product type models for sheet metal parts and complex shaped parts, like ship propellers.

The introduction of product type models seems to be useful to relate conceptual models. It adds extra structure to the two types of conceptual models of STEP. It ensures inter-operability between product type models, because two product type models based on the same reference model can now be used in one product model. Another big advantage is reusability of software: the scope of the conceptual models is at the same time the scope of the software supporting these conceptual models. In addition, it enables the development of product type specific modelling applications that are based on information structures valid for a whole industry.

**Structuring product information using decomposition**

A way to structure information in a product models is to follow the aggregation/decomposition of the product itself. The objects are related with so-called part-of relationships. Two typical examples of using decomposition for structuring product information are the RATAS building model and the GARM.

The RATAS building model [Björk 1993] describes the building as a set of related objects with attributes. One of the types of relationship is the part-of relationship. The model has five levels of decomposition (see figure 3.4). A building is decomposed in its systems, such as the spatial, structural, and HVAC system. These systems are decomposed in their functional subsystems, which may be subdivided in parts. Parts can be decomposed in their constituent details. The building model describes the building just after construction.
Decomposition as modelled in the GARM [Gielingh 1988] supports various design strategies, such as the separation of problem and solution, divide-and-conquer (or top-down design) and bottom-up design. The first two stages in the life cycle of a product (see figure 3.3) are used to represent a product as a network of interrelated components. A component of a product can be described on its turn as a (sub-)network. Networks and components can be modelled independently. But they can also be combined to evaluate the whole product (or a part of it) in more detail. This principle is realised by separating the network description from its usage as a component in another network. The network description is called a Technical Solution, and a component is called a Functional Unit. The requirements of a functional unit are fulfilled by the properties of a technical solution. For each functional unit, different technical solutions may be proposed. A technical solution decomposes in functional units of a lower order. A technical solution may exist before it is used in a more complex network. A good example of a successful application of this way of decomposing products is the Road Model Kernel of Willems [1990].

Decomposition is very important for structuring (product) information. For DfC, the way GARM decomposes products seems to be a good basis. It is flexible: it does not limit the number of decomposition levels nor the content of the levels. It not only represents the end result of a design process, but also intermediate stages of design including design arguments and alternative solutions. This implies that it can be used to support communication in all phases of design, also in the global phases at the beginning of the process.
Extending product modelling to project modelling

The STEP approach currently focuses mainly on *product* information. However, integrating design and construction is not restricted to product information. As shown in chapter 2, the interactions of design and construction also require communication on construction activities and resources. These other types of *project* information are also relevant. Two examples of current efforts to extend product modelling to project modelling are the IMPPACT reference model and the Unified Approach Model.

The IMPPACT reference model [Gielingh and Suhm 1993] relates product information to activity information for the mechanical industry (see figure 3.5). The product part of the project reference model corresponds closely to GARM. For the other parts of the model also GARM-like modelling constructs are used. The IMPPACT project extensively tested the validity of the reference model.

![Diagram](image)

*Figure 3.5 part of the IMPPACT reference model [Gielingh and Suhm 1993] showing the relation of the shape of a product and a production activity via a production stage.*

The Unified Approach Model is a project reference model developed by Björk et al. at VTT in Finland [1991]. This very generic model suggests how the use of a single conceptual model for all kinds of construction information would facilitate integration of different applications. The conceptual model relates information on activities, results, resources, agents (or participants) and legal contracts in a building project. Application of the Unified Approach Model in practice has not been reported.

Project modelling, as an extension of the STEP product modelling approach, might contribute to integrating design (which puts product information central) and construction (which puts activity and resource information central). Often the relation between product information in designs and activity information in schedules is not made explicit. With project modelling these implicit relations can be made explicit.
Extending product modelling with KB technology

The STEP approach focuses on information exchange. As shown in section 2.1, also formalising and exchanging knowledge is important for CA DfC. Therefore, combining product modelling with KB-technology can be useful to CA DfC.

In his thesis on Computer Aided Conformance Checking de Waard [1992] showed how product modelling can be used as basis for checking a design against building regulations. He suggest to model designs of buildings in a product type model and to formalise building regulations in knowledge rules. De Waard concentrates on passive conformance checking, i.e., the design (as contained in a product model) is sent to the building authorities, who check it against the regulations. The building authorities have an EXPRESS schema of their view on buildings, which includes the rules described in regulations. The product model of the designer is used to instantiate the EXPRESS schema of the building authorities as a building authorities’ product model. This model is checked against the regulations.

De Waard also recommends to research the value of his approach for active conformance checking. In that case, the EXPRESS model of the building authorities (including the rules contained in the building regulations) is directly connected to the CAD system of the designer. This allows on-line checking of design decisions.

Extending the STEP product modelling approach with knowledge formalisation seems useful for DfC. It may contribute to DfC interaction 4 of section 2.2: formalising and exchanging constructability knowledge. When the approach of de Waard is followed, constructability knowledge is formalised in knowledge rules. These rules can be used passively (i.e., for a constructability check by the constructor after design) and maybe even actively (i.e., for an interactive constructability check by the designer during design). Although EXPRESS may not provide the most suitable way to formalise knowledge, it illustrates how KB-technology can be combined with product modelling. At least, EXPRESS provides a neutral way to exchange knowledge and enables the use of the high-level semantics in defining knowledge.

Relating the above mentioned issues in a framework

In STEP the above mentioned issues are not related to each other. The general overview of the STEP standard [ISO/TC184 1993b] just announces that there are modelling languages and several types of conceptual models with different scopes. In his thesis Böhms [1991] defines a framework for modelling Computer Integrated Manufacturing (CIM) that can help to relate some of the issues discussed above.
Böhms develops a framework that spans a multi-dimensional space (see figure 3.6). First he distinguishes two basic dimensions: the epistemology and the language dimension. The epistemology dimension defines that a CIM framework describes, prescribes, interrelates and classifies models of reality. The language dimension relates modelling languages, which can be used for defining the models of reality.

![Diagram](image)

**figure 3.6 empty framework for modelling CIM [Böhms 1991]**

Then, a five dimensional framework is defined. The dimensions are: evolution, product life cycle stage, aspect, scope, and composition. The evolution dimension indicates the time scale of the model. The product life cycle stage indicates the stages of the complete product life cycle that are considered. The aspect dimension indicates what aspects (or views) are considered relevant for modelling. The scope dimension indicates the intended range of application of the model. The composition dimension indicates the different levels of aggregation. A descriptive CIM framework can be formed by identifying sets of points for every dimension.

Böhms uses this framework to describe, position and compare some existing research projects.

The framework of Böhms can also be used to relate the above mentioned issues and to describe this research (see figure 3.7).

For the epistemology dimension, this research focuses on a model of reality of building projects. This model is described in the framework of figure 3.7.

The four data levels of IRDS can be used for defining the points of the language dimension. In this research the modelling languages EXPRESS en NIAM are used to define the model of reality of building projects.

In the evolution dimension three points are used: definition of a prescriptive conceptual model for project information, implementation of this conceptual model, and usage of the implementation in a case study. Research question 1 aims at a prescriptive conceptual model. Research question 2 aims at computer support for the implementation of such a
conceptual model. In the case study the conceptual project model and the supporting system are prototyped.

Because this research aims at the integration of design and construction processes, it can be located on the life cycle dimension somewhere halfway between design and construction.

With respect to the aspect dimension, this research aims at project modelling. In STEP mainly product modelling is considered. Because the modelling languages do not support interrelating product, activity, and resource information, this has to be provided by the model.

For defining the points of the scope dimension the approach of Tolman with three types of conceptual models can be used. The scope of this research is the building industry in general. Therefore, this research concentrates on a reference model for modelling information in building projects. General resource models, e.g., for topology and geometry, are not dealt with in this research, because they are (or will be) dealt with in the generic resources of STEP. In the case study the reference model is specialised to a project type model for a specific type of building projects.

With respect to the composition dimension, this research looks at the realisation of complete buildings. The case study looks at one of the systems of a building: the structural system. Because neither the modelling languages nor the resource models deal with decomposition, it has to be dealt with in the reference model. For this research, the way GARM models decomposition will be used.

Conclusions on integration concepts

Research question 1 is, as mentioned in section 1.2:

What are the characteristics of an integration concept that supports CA DfC?
Inventory of the current state of DfC, chapter 2, showed the importance of standardised, high-level, computer-interpretable communication, for the integration of design and construction processes. Related research showed that the STEP product modelling approach could provide such communication. Related research also showed how to extend the STEP approach to serve as a basis for CA DfC. Relevant issues are modelling languages, scope, decomposition, project modelling and knowledge formalisation. These issues can be related in a so-called CIM framework.

A partial answer to the first research question might be an extension of the STEP product modelling approach to project modelling and the development of a reference model that integrates design and construction information. In this reference model attention should be paid to relate product, activity and resource information, and to the definition of decomposition.

Formalisation and exchange of knowledge is not considered for the building industry as a whole, but only on the project level. Constructability knowledge is therefore not included in the reference model for building projects. Related research showed how constructability knowledge may be formalised and exchanged using EXPRESS.

3.2 Related Research on Computer Implementations

This section describes related research on the computer implementation of integration concepts. Most attention will be paid to the implementation of product modelling.

Product modelling implementation history

The (short) history of the implementation of product models started soon after the concept of product modelling was introduced. The first implementations used relational databases or physical files as central data sharing points. Most of these implementations were research prototypes to show the value of the product modelling concept. A problem often encountered was the lack of applications used in practice: only few sub-processes of a building project were supported with computers. Often, only CAD applications were available, which meant that the systems developed not only had to integrate existing applications, but also to support information generation on the semantic level of a product model.

Examples of product modelling implementations are RATAS, prototype implementations in the USA, and ProMod.
The RATAS building model (see figure 3.4) is implemented in a hypermedia environment [Penttilä, Talonpoika, et al. 1990]. This prototype shows how building data can be browsed through and manipulated in an object-oriented programming language. It shows the relevance of data hiding and the use of a graphical representation for ease of survey.

In the USA much research has been done in prototype product modelling systems. Examples are CIFECAD of Stanford University [Kolountzakis and Fischer 1991], the Engineering Data Model (EDM) of the University of California, Los Angeles [Eastman, Bond, et al. 1991], the Integrated Building Design Environment (IBDE) of Carnegie-Mellon [Fenves, Hendrickson, et al. 1990], the Distributed and Integrated environment for Computer-aided Engineering (DICE) of the Massachusetts Institute of Technology (MIT) [Sriram, Logcher, et al. 1989], and ICADS of the California Polytechnic State University [Pohl 1991]. These prototypes often not focus on standardisation of information exchange, as ISO-STEP, but on design systems based on product models.

Another implementation example is ProMod of TNO [Kuiper 1989]. ProMod is a prototype implementation of a part of GARM in its original interpretation: the object types Functional Unit and Technical Solution are instantiated directly as objects in a product model. ProMod provides tools to generate bills of quantity, graphical output and STEP physical files. The prototype showed the need for an additional product type layer between GARM and the product models (see section 3.1).

An important observation in the history of implementing product modelling is the increased use of object-oriented programming. Object-oriented programming, as described by Meyer [1988] and Booch [1991], better corresponds to conceptual modelling than traditional procedural programming [Sol 1982]. Hannus [1987] and Turk [1992] argue the value of object-oriented programming for product modelling. They say that the conceptual modelling notions of object types with characteristics corresponds to the object-oriented notions of classes, properties and behaviour.

The history of product modelling in the building industry shows little or no experience with practical, applicable implementations. A problem is, and probably will be for some time, the lack of applications that can communicate on the required semantic level. The goal of CA DfC is to integrate islands of automation, but some islands are not even automated!

Of course, this picture can also be turned around. Without an international standard for project data interchange, there is no market for software that communicates on a high semantic level. So, who knows what the future will bring?
Vendor or company specific implementations

A more complete solution is offered by vendor, or company specific implementations. Typical examples are Intergraph and LORAN-T.

The Intergraph system [Intergraph Corporation 1988] is still one of the world leaders in the CAD/CAM area. The integrated system comprises hardware, applications software, data management and communication components. There are applications for design, structural analysis, and manufacturing in different areas, as mechanical engineering and AEC.

In Japan the large general contractors, the so-called Big-6, are able to develop systems that integrate internally used applications. An example of such a system is the LOng Range Architectural Networking system in Tai-sei (LORAN-T) [Kita, Kato, et al. 1992]. LORAN-T is an integrated CAD system for building design with an architectural semantic model in a central database. The model contains information on architectural, structural and utility equipment design. It integrates internal applications for planning and schematic design, architectural design, structural design, mechanical and electrical design, cost estimation, CAM, and control of construction robots.

Vendor- or company-specific implementations of product modelling concepts have shown their value in practice. However, no single vendor or company can enforce a high-level communication standard on the international building industry. Therefore, this thesis concentrates on the implementation of open (STEP) product modelling concepts.

CASE tools

The foregoing showed the need for Computer Aided Systems Engineering (CASE) tools that support a standardised approach for product (project) modelling. Tools are required to support the development of conceptual models. Also tools are required to enable application of the conceptual models in practice.

The first step in the development of conceptual models is analysing the processes involved. Commonly used process analysis techniques, such as IDEF0, are supported by computer tools. The next step is the specifying the conceptual models. Currently most modelling languages are supported by CASE tools. For example, an extensive list of EXPRESS tools is described by Wilson [1992]. The third step is generating a database structure according to the specified conceptual model. Several EXPRESS based database generation tools are available, e.g., the EXIS environment of TNO [Brujin 1990].

Less straightforward is the implementation of a tool that can be used to manipulate the data. The next step is therefore generating an object-oriented implementation from the conceptual model. For this no commercial tools are currently available. Other important tools that are not
yet available are tools to support the development of pre- and post-processors between databases and applications. Also tools are required for the exchange of the actual data.

Existing CASE tools support parts of the development and use of conceptual models. These tools are important to support implementation of CA DIC. However, existing tools do not cover the whole conceptual modelling process and are not always linkable. CASE tools are implemented increasingly in object-oriented programming languages.

Implementation systems

Currently, prototype systems are under development that incorporate several linked CASE tools. Two typical examples are XPDI station and SIFRAME.

CSTB in France is developing the eXpert Product Data Interchange (XPDI) station [Poyet, Debras, et al. 1993]. XPDI is a meta environment for the development of CASE tools. It concentrates on supporting the development and implementation of, among others, conceptual models. XPDI offers an artificial intelligence (or knowledge-based) kernel, supplemented with product modelling functionality. In addition, it provides tools for conceptual modelling, function modelling, CAD facilities, STEP-based exchanges, and hypermedia facilities. It is implemented in an object-oriented programming language.

SIFRAME [Siemens Nixdorf 1992] has a different focus: it is a platform for application integration and usage. It concentrates on the end-user in practice. SIFRAME is developed by Siemens Nixdorf Information systems. It can be used to encapsulate, execute and manage data and applications in multi-user design projects. It provides tools to model processes and applications. Functionality and information requirements of applications can also be modelled. SIFRAME contains an object-oriented database system. It supports standards such as EXPRESS and the STEP physical file format.

Also interesting in this field are the design systems that manipulate product information on the semantic level of product models. A typical example is Design++.

Design++ [Design Power 1991] is a design system that enables the exchange of product information between a knowledge-based (KB) system, a database, and a CAD system. It supports the development of the knowledge rules and information structures. The designer can manipulate information in the KB system or in the CAD system. The KB system ensures consistency of the information with the knowledge rules. The CAD system provides a representation and a 3D presentation of the shape of the product. The information can be stored in and retrieved from a database. Design++ does not yet support the STEP languages.
Systems for the implementation of standardised integration concepts concentrate either on the end-user in practice, such as SIFRAME, or the developer of applications, such as XPD1. Design systems such as Design++ are useful to develop modelling systems for end-users that manipulate information on the semantic level of product models. These two kinds of systems are very important for a successful introduction of product modelling in practice.

Conclusions on implementing product modelling

Research question 2 is, as mentioned in section 1.2:

How can the technical implementation of the integration concept for CA DfC be supported with computers?

Section 3.1 argued that project modelling can contribute to CA DfC. Project modelling is seen as an extension of (STEP) product modelling. Related research on implementing product modelling showed the relevance of CASE tools for developing and applying conceptual models. These CASE tools can be used separately, or integrated in one suite of applications, or system. Because the object-oriented paradigm closely corresponds to conceptual modelling, CASE tools and systems are currently often implemented in object-oriented programming languages. What is often missing at this moment, is a direct link between the development of conceptual models and their application in practice.

An answer to the second research question might be to specify a modelling system that supports both development and application of the conceptual models developed in answering research question 1. This specification can be verified by the development and usage of a prototype version of a modelling system. Implementing the prototype with an object-oriented programming language seems appropriate, as demonstrated in related research.

3.3 Related Research on Organisational Implementations

Because of the immature state of CA DfC and project modelling only little research has been done into their implementation in practice. The previ-

1 This conclusion is rather oversimplified. In fact, the border between end-user and developer is not always completely clear, and the two systems approach each other more and more.
ous section discussed that until now product modelling is mainly imple-
mented in (research) prototypes. One of the reasons is the lack of mod-
ellling tools that support the technical implementation. Research question 2
aims at providing such tools. When technically within reach, the building
participants might get interested in implementing CA DfC, when they see
what CA DfC has to offer. This section discusses some organisational de-
velopments that can be observed in the current building industry or in
other industries. It indicates how these developments might be supported
with improved integration of design and construction.

Organisational developments in the building industry

Research into organisational changes of the Dutch building industry has
been done by the Dutch Foundation for Building Research (SBR) [Dekker
and Vreeze 1985; Haselhoff and Rijlaarsdam 1988; Joosten and Werven
1991], the Dutch Technology policy Advisory Board for the Building
industry (ARTB) [ARTB 1993] and the Dutch Scientific counsel for
governmental policy [Mischgofsky 1991]. For DfC relevant developments
mentioned are: increased influence of the suppliers, design-and-build (or
turnkey) projects, long-term (or co-makership) relations of building
participants, automation on site or in factories, total life cycle
optimisation, and performance contracts.

The organisational developments mentioned above might be supported
with the computerised availability of the six interactions of DfC (see
chapter 2).

Increasingly suppliers are designing, fabricating, and assembling their
part of the building in an industrialised environment. In a design-and-
build project one company designs and builds the product. In these two
developments design and construction are in the hand of one company. It
is evident that this company will enjoy the benefits of better integration of
the two processes. Because information, knowledge, and tasks do not have
to cross company borders, integration will be realised easier then in usual
building projects with many participants. A similar effect can be reached
when companies work together for a longer period.

Automation on site or in an industrialised environment requires elec-
tronic control commands for the machines, as argued by Krom [1992]. If
DfC interaction 1, forward exchange of building information, is supported
with computers, the design of the building is available in a computer-in-
terpretable format. The design can then be used as input for applications
that generate the control commands for the machines. If interaction 4, the
exchange of constructability knowledge, is supported, special require-
ments of the machines can be taken into account.

Mischgofsky [1991] and the ARTB [1993] see a trend towards total life
time optimisation. This means that not only the construction cost is taken
into account but also operation, maintenance, and demolition costs. If the
interactions of DfC are supported with computers, information on the building is available in a computer-interpretable format. This information might be useful for controlling operation (as in facility management) and maintenance. When it is possible to exchange and use constructability knowledge with computers (interaction 4), a similar approach might be applicable to exchanging operation and maintenance knowledge to design.

Another trend observed is the growing use of so-called performance contracts. This implies that the client does not specify a final solution, but only the requirements for the solution. In his tender the producer proposes a design with a price ticket. Such a client-producer relation can be formed between the owner of the building and a general contractor, but also between the main contractor and his suppliers. Also this development can be helped with CA DfC. Interaction 1 aims at forward exchange of design information. When not only the exchange of the final design is supported, but also the design intent, it can be used for exchanging the design requirements of the client to the producer.

Organisational developments in other industries

Two developments in the manufacturing industry that might be interesting for the building industry are lean production and concurrent engineering.

Especially in the car industry, interesting organisational developments take place. For example, Womack, Jones, et al. [1990] describe lean production, and call it the reason for success of the Japanese car industry. The lean production philosophy states that there are only two aspects important in production systems: conversion activities and flows. Information and material flows are the basic units of analysis. Flows are characterised by time, cost, and value. While all activities imply cost and consume time, only conversion activities add value to the flows. Therefore, lean production concentrates on the two main conversion activities processes: design (the main information conversion process) and construction (the main material conversion process). Lean production tries (1) to improve those activities, and (2) to eliminate all other activities, including organisation, profit sharing, research, etc. Koskela [1992] argues that at least parts of the philosophy of lean production might also be applicable to the building industry, despite of the one-of-a-kind nature of its projects, its site production, and its many temporary relations between participants.

Another development that might be interesting for the building industry is concurrent engineering. With concurrent engineering processes are performed more in parallel than sequentially [Prasad, Morenc, et al. 1993]. A similar idea is brought into practice in the building industry by the introduction of so-called building teams.

Introduction of the two manufacturing developments in the building industry might be easier when CA DfC is realised.
CA DfC might help bringing the lean production philosophy into practice. The active use of constructability knowledge during design is an example of information flow optimisation, which might help optimising the construction process. Exchange of information between computers might help to reduce activities that do not add value to design and construction management. For example, generation of detail drawings and human interpretation of designs by construction schedulers might be minimised. Of course, CA DfC will certainly not bring lean production to the building industry, but it might offer some useful tools.

Concurrent engineering may also be supported with CA DfC. Concurrent engineering requires intensive and fast communication. It also asks for rapid feedback on decisions made by others. Interaction 1 and 2, forward information exchange and feedback from constructors, might offer tools to realise this objective.

Conclusions on organisational implementation

Research question 3 is, as mentioned in section 1.2:

How can the integration concept for CA DfC be implemented organisationally?

Section 3.1 and 3.2 respectively discussed how project modelling might help realise CA DfC and described how an object-oriented project modelling system might support the technical implementation of CA DfC. This section mentioned some organisational developments in the building industry and other industries that might be supported with CA DfC.

Before the applicability of CA DfC and project modelling are better known, it is not possible to answer the third research question in full detail. Because of the immature state of CA DfC, only possible benefits of CA DfC to existing organisational developments can be indicated at the moment.
4

Integration Concept: Building Project Model

This chapter concentrates on research question 1: "What are the characteristics of an integration concept that supports CA DfC?". As discussed in section 3.1, the answer of this question may be a reference model for building projects that integrates product, activity and resource information. This chapter presents a proposal for such a reference model: the Building Project Model (BPM).

4.1 The Building Project Model (BPM)

Related research on integration concepts (section 3.1) showed a possible answer on research question 1: a reference model for integrating design and construction information in building projects. In this chapter, a proposal for a reference model is defined and presented in NIAM [Nijssen and Halpin 1989]. The reference model, called Building Project Model (BPM), relates project, activity and resource information and supports decomposition. General aspects, like topology and geometry, are not dealt with in the reference model, because they are (or will be) dealt with in the generic resources of STEP.

Abstraction mechanisms

To organise a comprehensive conceptual model such as a reference model, it should emphasise the essentials of the problem, and suppress the irrelevant details. In conceptual modelling research this is called abstraction.


The classification mechanism groups objects that share common characteristics. This mechanism is already supported by the modelling languages chosen. The groups of objects corresponds to the conceptual
modelling object types, the EXPRESS entities, and the object-oriented classes.

The generalisation-specialisation mechanism relates object types in a is-a hierarchy. Also the specialisation mechanism does not need much further attention and development in this research. EXPRESS and NIAM both contain a mechanism to specialise object types, i.e., subtyping. An entity of EXPRESS, or an object type of NIAM denotes a set of objects with common properties. A subtype denotes a subset of these objects that adds specific properties to the common properties. An object type can be subtype of object types in the same conceptual model and/or in other conceptual models.

The decomposition-aggregation mechanism relates object types in a part-of hierarchy. Because neither the modelling languages nor the resource models of STEP provide such an explicit decomposition mechanism, it has to be incorporated in the BPM. Section 3.1 mentioned a few requirements for this mechanism: the number of object decomposition levels and the contents of these levels should not be restricted, and the decomposition mechanism should follow the design process.

Based on related research, two abstraction mechanisms are chosen to be elaborated in the BPM: (1) product-activity and (2) concretisation. The product-activity mechanism emphasises the relations between product, activity, and resource information in a building project. The concretisation mechanism emphasises the information needs during problem solving processes such as design and construction scheduling. The concretisation mechanism is also used to model decomposition.

Because both abstraction mechanisms are not supported in the modelling languages nor in the (STEP) resource models, they have to be modelled in the BPM. Sections 4.2 and 4.3 describe the two abstraction mechanisms and the resulting conceptual models. Section 4.4 presents the BPM as combination of these two conceptual models.

4.2 Abstraction mechanism 1: Product-Activity

This section discusses the first abstraction mechanism of the BPM: product-activity. It relates the main types of information in a building project: product, activity and resource information.

Types of information

The main types of information in a building project are product, activity, and resource information. In practice, these types of information can be found in the product design, the activity schedules, and the resource plans.
Design mainly focuses on product information, construction on activity and resource information. To integrate design and construction, the types of information used have to be integrated in a way that is natural for the building and construction industries.

The three types of information are often modelled separately, e.g., by using the STEP product modelling methodology and by using an activity analysis method. The first goal of the BPM is to relate the three types of information in one conceptual model.

In figure 4.1 the three main types of information are modelled as subtypes of Project Object. A project object is part of a project.

![Diagram](image)

*figure 4.1 a project consists of project objects of several kinds*

Of course, this product-activity model is not complete. Not all types of information in a project are modelled. Also the relations between the types are missing. The missing types and relations have to be examined in more detail. The following describes how these relations are dealt with in a logic of change, an activity analysis methods, and in other reference models. These findings are then combined to a product-activity model for CA DfC.

**Logic of change**

The logic of change describes the relations between activities, their results, and the people that perform them. In *Norm and Action* von Wright [1963] discusses the concept of human action as a basis for his deontic logic, the theory of duty or moral obligation. The ideas of von Wright are modelled conceptually using NIAM. For each of its aspects a conceptual model is made. These sub-models are combined to one overall model.

---

1 Please, note that the NIAM diagrams in this section are only an *interpretation* by the author of a *part* of the work of von Wright.
Von Wright distinguishes two types of Facts: State of Affairs and Event\(^1\) (see figure 4.2). A state of affairs describes a situation in the world on a certain occasion, or in other words something which *is*. An event describes something which *happens*, in the sense of *taking place*. An event is a change in the world. State of affairs has to do with *static behaviour*, where event is related to *dynamic behaviour*.

\[
\text{state of affairs} \quad \xrightarrow{\text{has initial}} \quad \xrightarrow{\text{has end-}} \quad \text{event} \quad \xrightarrow{\text{fact}} \quad \text{event}
\]

*figure 4.2  Event and State of Affairs as subtypes of Fact [Wright 1963]*

These two subtypes of Fact are related. An event is the change or transition from a state of affairs obtained on an earlier occasion, to a new state. In other words, an event is the transformation from an initial state to an end-state.

Von Wright relates the notion of (human) Action to the notion of Event. An action is not an event, but it is the bringing about or effecting of an event. The logical difference between Action and Event is a difference between *activity* and *passivity*. An action effects an event (see figure 4.3).

\[
\text{event} \quad \xrightarrow{\text{action}} \quad \text{effects}
\]

*figure 4.3  Action [Wright 1963]*

An action is performed by an agent. An agent performs an action with the intention to reach a result (see figure 4.4). By the result of an action can be understood (1) the *intended* change of an action, or (2) the end-state of this change. Von Wright does not make a choice between the two

\(^1\) Actually, von Wright also distinguishes a third type of Fact: Process. A process describes something which *goes on* and *happens* over a certain period of time. For CA DfC the subtle difference between Process and Event is not considered relevant.
interpretations. In the following the first interpretation is used. This makes Result a subtype of Event.

![Diagram of event, result, agent, and action relationships]

**Figure 4.4** *Result and Agent [Wright 1963]*

An agent only performs actions, which he is able to perform (see figure 4.5). An agent who can do something (or knows how to do it), has the ability to perform an action.

![Diagram of result, agent, ability, and action relationships]

**Figure 4.5** *Ability [Wright 1963]*

Figure 4.6 combines the ideas of von Wright on the logic of change, presented in the preceding figures, in one conceptual model.
An analogy with activity analysis methods

The resulting model of von Wright, as shown in figure 4.6, gives a good overview of the relations between Action, Agent and Result. However, with what the actions are performed (the resources) and how these actions are controlled (the controls) is not specified. The IDEF0 paradigm [SofTech 1981] deals with these issues.

An IDEF0 model represents (or describes) activities\(^1\) of a system, with a certain goal, and from a certain point of view. A model consists of a series of diagrams and supporting documentation (see figure 4.7).

An IDEF0 diagram has activity boxes and flow arrows. Figures 2.1 and 2.2 show examples of IDEF0 diagrams. A diagram represents an activity. An activity is "anything that can be named with an active verb phrase", and occurs over a period of time. The activities in a diagram are called child activities. The activity represented in a diagram is called the parent activity.

An activity box is part of an IDEF0 diagram. Or, in other words, it is a child in an IDEF0 diagram. It contains an active verb phrase and an identification number. It represents a child activity, but can also decompose into (sub-) child activities. For this decomposition a new diagram is used. An activity, which is child (represented with an activity box) on one decomposition level, is parent (represented with a diagram) on a lower level.

---

\(^1\) In fact, IDEF0 models functions [SofTech 1981]. A function shows what must be accomplished without identifying any other necessary aspects, as needs or means. Function is an overall term, covering activities, actions, processes and operations. For the analogy the term activity is used.
level. The name of a parent diagram coincides with the active verb phrase in the child activity box.

An IDEF0 diagram also contains flow arrows, which relate the activity boxes. A flow arrow represents a flow. A flow is "information, objects, or anything that can be described with nouns". A flow can play four roles. It can be input or output of an activity. It can control an activity. Or, it can be a mechanism (or resource) used in an activity.

![Diagram of an IDEF0 model](image)

**Figure 4.7** A conceptual model of an IDEF0 diagram [SoftTech 1981]

**Product-Activity in other reference models**

The relation between products and activities has been dealt with in several other reference models. Examples are: the IMPACT reference model, the Unified Approach Model, GenCOM and IRMA.

**IMPPACT reference model**

The IMPPACT reference model [Gielingh and Suhm 1993] models the relation between Product and Production Activity for the mechanical industry. Figure 4.8 shows the so-called activity model. A Production Activity is a generalisation of a Pass (a single move of a tool), an Operation (an activity done with one tool), a Process (an activity done at one worksta-
tion), and a Process Group. During an activity the properties of a product may change. A production product is the result at a certain production stage. This stage precedes one production activity and succeeds another. The word “production” in Production Product denotes the production point of view on a product.

![Diagram of the activity model in the IMPPACT reference model](image)

*Figure 4.8* the activity model in the IMPPACT reference model [Gielingh and Suhrm 1993]

For the four subtypes of Production Activity the used resources are indicated. In the general part of the reference model (i.e., the upper part of the model) there is no generalisation of these resources.

**Unified Approach Model**

The Unified Approach Model was developed by VTT, Finland, as a generic model for the building industry [Björk 1991]. The central object type in the model is Activity (see figure 4.9). It denotes any activity in the building process. The result produced by an activity can be something physical, a piece of information or a service. An activity is performed by an agent. Client, producer and result are related via a contract. An activity necessitates the use of resources. Resource as well as Resource Use are modelled explicitly. Resources can be durable (like equipment), consumable (like materials), or a department of a company (like an office or factory). A result can function as a resource. A durable resource can function as a micro-level agent.
GenCOM

The General Construction Object Model (GenCOM) was developed at Stanford University to improve integration of project management software using standardised conceptual models of construction projects [Froese 1992]. Figure 4.10 shows some of its object types. At the heart of the model are component of a facility, and activities that operate on the components. Activities are part of an overall Construction plan. Each activity requires actions to be performed, using a particular method and a set of resources. Again Resource and Resource Use are modelled explicitly. Activities are performed by a project participant.
This model focuses on construction. This means that only construction activities are taken into account.

IRMA

The Information Reference Model for AEC (IRMA) is the result of the information modelling working group of a workshop on frameworks for computer-integrated construction [Björk and Hannus 1992]. At this workshop researchers from around the world presented their individual modelling efforts. These models’ common goals and abundant similarities led the participants to combine their individual efforts. IRMA defines generally applicable relationships between products, activities, resources, and participants in a building project.

The IRMA resulted, among others, from an earlier version of the BPM. The two reference models mutually influenced each other. Therefore, the models have much in common.

The current version of the IRMA [Luiten, Froese, et al. 1992] has four subtypes of Project Object: Product, Activity, Resource and Contract (see figure 4.11). Agent is a subtype of Resource. An agent can be an organisation or an individual, referred to as micro-level agent.

Again, Activity is the central object type. An activity requires initial states of project objects and results in new states. An activity is performed by an agent. Resources are used for the activities. Resource and Resource Use are modelled explicitly. The resulting state is defined in a contract between a client and a producer agent.
Product-Activity for the building industry

The models discussed in the previous subsections seem to contain useful aspects for integrating design and construction information. The following aims at combining these aspects in one product-activity model for the building industry. This combining implies among others that the terminology is reconciled, that choices between conflicting alternatives are made, that irrelevancies are omitted, and that some generalisations are made.

The logic of change of von Wright, as shown in figure 4.6, forms the basis of the product-activity model. Some simplifications can be made, as shown in figure 4.12.

In this interpretation an event is always a change of the state of affairs. Result is the only subtype of Event. Every action effects exactly one event. This means that the notions of Event, Result and Action can be combined: an Action directly results in the change of one state into another. A state of affairs can be divided into two parts: (1) state and (2) affairs. For this reference model affairs corresponds with Project object of figure 4.1, and state is modelled as a new object type.
Figure 4.12 shows the reconciliation of terminology of the presented models. Agent, Actor and Project Participant are represented by the term "Actor". Action and Activity are represented by the term "Activity".

\[ \text{figure 4.12} \quad \text{von Wright's ideas on the logic of change translated to the building industry} \]

From the IDEF0 analogy and the other reference models also some ideas can be used. This results in figure 4.13.

Because IDEF0 is only a representation of activities, only the notions behind it are modelled. Relevant is the notion that flows can play different roles. For example, a precast concrete element is output of a fabrication activity, and is input for an assembly activity. Project Object models such a flow. Also relevant is the notion that activities are controlled by flows. When an activity is controlled, also its incoming and outgoing flows should correspond to requirements. The should-corrrespond-to relation between project objects models the control flow of IDEF0. For example, a designed product should correspond to product requirements defined in regulations. Or, a construction process should correspond to the designed product.

In the presented reference models Resource and Actor are modelled as separate object types, with characteristic properties. Resources are used for activities. Actors perform activities. In IRMA, Actor is seen as a special type of Resource. An actor on one level can use actors on another level. For example, on the project management level, the designer is one
of the resources. On a lower level, the designer performs the design activities.

The IMPPACT reference model showed that states\(^1\) succeed each other. This notion can be added to the product-activity model by defining a succeeds/precedes relation between states. In principle, this is also accomplished by saying that every activity has an initial and an end-state, and that the end-state of one process can be the initial state for the next. But, this is not always enough. For example, at the beginning of the scheduling process, the required activities are unknown. The first thing to determine is the successive states of the product during construction. Only after that, the activities can be specified. Therefore, it is convenient to have both mechanisms: states that succeed each other, and activities that concatenate states. In earlier phases of the planning process successive states are normative, in later phases the activities are normative.

The three main kinds of Project object have characteristic properties. Shape and Material are the characteristic properties of Product, as shown in figures 4.14. Shape is researched in detail by Willems [1993].

---

\(^1\) In the IMPPACT reference model State is called Stage.
For modelling explicit shape and material, STEP resource models can be used.

![Diagram of product, shape, and material]

**Figure 4.14** Product and its shape and material properties

For Activity the characteristic property is Time, as shown in figure 4.15. An activity has a starting time, an ending time and a duration. Of course, these notions are related. In the beginning of the planning process, only an estimated duration of an activity is known. Later, when activities are related to each other, the start and end time will be known. The actual duration can be derived.

![Diagram of activity, duration, start, and end]

**Figure 4.15** Activity and its time properties

For Resource the characteristic property is Cost (see figure 4.16). The Unified Approach Model (as does GenCOM) shows the value of modelling Resource Use explicitly. With Resource Use it is possible to use one resource at several activities and deal with every use separately. For example, it is possible to calculate the cost of one use of a resource separately from other uses of that same resource. In figures 4.16 and 4.13 this is modelled by making the use relation between activity and resource an objectified relation. Extra functionality can be added to Resource Use. For example, when a resource has initial cost and cost per unit, and when a number of units is used, the total cost for that resource use can be derived. Or, when a resource is only used during a part of the duration of the activity, this can be indicated at the corresponding resource use.
In the Unified Approach Model resources are material, equipment or factories. This is partly in contrast to the IDEF0 analogy, where material is seen as an input. In the interpretation of input of an activity so far, it is seen as a state of a project object. This does not cover properties of material, like its cost. Material has more properties in common with resource. Therefore, material is seen as a resource and not as an initial state of a project object.

**Product-Activity applied to DfC**

The product-activity model, as presented in the previous, is very general, and too comprehensive for information exchange required for DfC. For example, Activity stands for any activity in the building process. This means that an activity can be a construction activity, a planning activity, a design activity, or even a project management activity. For DfC only construction activities are important. Therefore, the product-activity model of figure 4.13 is specialised to construction activities, construction resources, and construction states. This leads to figure 4.17.

During construction the state of the project normally refers to the state of the product under construction. It can also refer to the state of construction resources. Because only construction activities are relevant for DfC, the only relevant actors are constructors like general contractors, sub-contractors and suppliers. Subtypes of Resource are Labour Force, Material and Equipment, or compositions of these three basic resources. Examples of resource compositions are working groups in a factory, or on a building site, or even companies like (sub-)contractors and suppliers. For CA DfC a product should conform to constructability requirements. These requirements can be formalised in the object type Product.
As described in the Shop Floor Production Model of ISO-STEP [ISO/TC184 1988] for discrete parts manufacturing in the mechanical industry, there are four types of construction activities: (1) transform, (2) transport, (3) store, and (4) verify activities (see figure 4.17). Transformation is anything in which materials or parts are brought in a new shape. Examples in the building industry are welding, pouring, assembling, and fixing. Materials, building parts and resources have to be transported to the right place. If necessary for logistic reasons, the materials, building parts and resources are stored first. Verification is needed to be sure that the other activities are properly performed. The verification results form the feedback for the planners.

**Use of product-activity**

The product-activity model is developed to relate product information, activity information, and resource information. The relation between the three types of information is important because design is limited by the construction possibilities and construction is largely determined by the design of the building. The design has to be constructable at acceptable cost, within acceptable time, and with the available resources. The design prescribes what the constructor has to build.
The relation between the three types of information often only exists in the minds of the people involved: the relations only exist implicitly. Design and construction management are normally performed by different persons, who possibly interpret the design, the schedules, and their relations differently. As long as the relations stay implicit, such differences in interpretation may exist and may cause problems.

When the relations are made explicit in a project model, computer applications can support storing and manipulating these probably large amounts of information. Computer applications might also support using this information, e.g., for evaluating the constructability of a design and for deriving construction and resource schedules from the design.

It should be noted that the product-activity model is part of the BPM. This means that it is intended to be used as reference model for specialised conceptual models (the so-called project type models (PtMs)) for one type of building project. In these PtMs the product-activity information structures are specialised to products and activities of that type of project and new information structures are added.

The product-activity model (or a specialisation of it in a PtM) is able to contribute to several interactions of DfC (see chapter 2).

The product-activity model might support the forward exchange of information (interaction 1) using one shared project model that is extended in the course of a building project. First, the designer proposes a product. This designed product is the starting point for the planner. He defines the successive states of the product (and if necessary of the resources) during construction. Next, he defines the construction activities to reach the defined states and he defines the required resources. The constructor acquires the resources and performs the activities, which results in the product. The information generated in these processes can be stored in the project model. The project model can also be used to exchange feedback information (interaction 2) from the constructor to the designer. For example, the constructor can send the designer a proposal for an improved design of a part of the building.

The product-activity model might also support (parts of) other interactions. For example, sometimes a design choice implies a certain construction method. With the product-activity model the designer can add this activity (or construction state) information to his design and exchange it to the constructor (part of interaction 1). The model can also be used to exchange resource information from the constructor to the designer (interaction 3). During design, the construction states of the building have to be taken into account. For example, a structural engineer often takes the loading combinations that occur during transportation and assembly into account. With the product-activity model the constructor can pass this feedback information to the designer (interaction 2). When the designer has done part of the construction planning to see the consequences of his design choices (interaction 5), he can use the project model to exchange this scheduling information to the construction manager. The construction
manager can see this (global) schedule as a starting point for the scheduling process.

An important property of the presented product-activity model is that not only the static definition of the final product is described, but also the dynamic change of a product from material and components to physical reality. A separate product model normally only describes the designed product and a separate activity model only the scheduled construction activities. This product-activity model also describes how the product changes during construction activities.

This property might be important, e.g., for the construction phase. Normally, a construction manager is only interested in the activities and resources. However, the constructor is also interested in the states that result from the activities. Today, these intermediate results are checked against the designed product. When a project model should also describe the intermediate results of an construction activity, construction progress control might be easier. In that case, the person that controls the progress of construction knows the state of the building at any time during construction.

The product-activity model relates design and construction information, which might support (1) incorporating construction possibilities in the design process, (2) fitting construction management to design, and (3) ensuring that construction leads to the designed product.

4.3 Abstraction mechanism 2: Concretisation

This section describes the second abstraction mechanism of the BPM: concretisation. It starts with its origin: the process of problem solving. Then, concretisation in other reference models is discussed. The section ends with a definition of concretisation and a presentation of the resulting concretisation model.

Problem solving

The second abstraction mechanism of the BPM emphasises the information needs during the problem solving processes in a building project. The building process is a problem solving process: the functional requirements of the client have to be fulfilled by a building. Also the sub-processes are problem solving processes: during design a solution has to be found for the functional requirements, and during construction management a solution has to be found to realise the design. The concretisation abstraction
mechanism supports the information needs that the problem solving processes in a building project have in common.

Two important principles of problem solving are considered: (1) the explicit separation of problem and solution and (2) the divide-and-conquer problem solving strategy.

Separation of problem and solution

A basic principle of most strategies for problem solving is to make a clear distinction between the problem and its solution. First the problem has to be specified, with criteria to be fulfilled by the solution. Then a number of alternative solutions is proposed and worked out into such detail that the alternatives can be compared to the criteria and to each other. Finally, one of the alternatives is selected and realised. After the chosen solution is realised, it can be compared with the criteria stated in the beginning.

The whole process of creating a building is a problem solving process. The problem to be solved is to fulfil the needs of the client for a new building. In the design brief the client specifies his criteria for the new building. In general, criteria on this level are functional requirements with time and cost limitations. The designer, normally the architect, develops a number of alternatives, compares them with the imposed criteria and selects one. The selected alternative is worked out in full detail and realised by constructors. Finally, while using the building, the client can tell whether the imposed criteria are really fulfilled.

Not only the whole building process, also sub-processes are problem solving processes. The designer solves the problems of a required building. The scheduler solves the problem of defining construction activities and resources.

It should be noted that the preferred strategy of explicitly separating problem and solution is not always followed in current practice. Often only solutions are made explicit in communication, especially on detailed levels. For example, specification of HVAC installations normally does not contain requirements, like “this tube should be able to transport that volume of air per second under those circumstances”. But, it prescribes specific solutions right away, like “this tube should be made of that material with those measurements”. Specifying solutions makes it easier to compare tenders, and to select the cheapest realisation of the pre-defined solution. However, it is questionable, whether this way the best solution is selected.

To support the separation of problem and solution the concretisation abstraction mechanism is used. During the problem solving process the definition of a project object (i.e., a product, an activity, or a resource) becomes increasingly concrete. In general, project objects occur in three evolutionary states:
• as required objects,
• as proposed objects, and
• as realised objects.

These three states denote the three object types of the concretisation mechanism. These types are related. As required objects define criteria. As proposed objects define possible solutions for these criteria. And, as realised objects define realised solutions.

Divide-and-conquer

When problems are large, a second principle of problem solving is often applied: divide-and-conquer. The main idea is that a large problem can be solved by dividing (or decomposing) it into sub-problems. Normally, the sub-problems are related to each other. The divide-and-conquer strategy therefore starts with a clear definition of the problem interfaces.

The two principles can be combined. Each alternative solution is decomposed into sub-problems. Criteria for the solution of those sub-problems can be derived from criteria of the main problem. For these sub-problems an alternative solution is proposed. This proposed solution can again be divided in sub-sub-problems. The principle can be applied until the problem is small enough to be solved. On the lowest decomposition level solutions are pre-defined, and can be specified by parameter values.

In a building process it is also inevitable to use the divide-and-conquer principle: the problem of realising a complete building requires too many skills. Therefore, the problem is decomposed into sub-problems. For example, when a client wants office facilities to do his business, one of the alternatives is to build a new office building. The realisation of an office building is then decomposed in design, construction management, and construction. Design still requires too many skills and is therefore decomposed in architectural design, structural design, HVAC design, etc. A similar division is made for construction management and for construction. Of course, the parts are related. For example, the architectural choices influence structural design. In this way the divide-and-conquer principle enables participants in a complex process to work together. It structures multi-disciplinary work teams.

Besides the separation of problem and solution, the concretisation abstraction mechanism can also be used to support the divide-and-conquer principle. This means that the second abstraction mechanism supports the decomposition of an alternative solution in related sub-problems.

Supporting problem solving in reference models

Supporting communication in problem solving processes has been dealt with more or less in other reference models too. The following discusses
some interesting models, i.e., the GARM, the IMPPACT reference model, and the conceptual model used for classification by ISO. It is also discussed what happens when reference models do not support the information needs of problem solving.

**GARM**

The subject of supporting communication in problem solving was first addressed by Gielingh in the GARM [1988]. The GARM models the product and its characteristics in different life cycle stages (see figure 3.3). A product in its as required stage is called a Functional Unit, and a product in its as proposed stage a Technical Solution. A functional unit has required characteristics, a technical solution has proposed characteristics. The upper part of figure 4.18 shows the relations between Functional Unit and Technical Solution.

![Diagram](image)

**figure 4.18 the main relations between Functional Unit and Technical Solution in the GARM [Gielingh 1988]**

A functional unit is fulfilled by a technical solution. In some cases, the proposed characteristics correspond directly to the required characteristics. For example, governmental rules require that a certain building has a certain maximum height. This means that an alternative for a high rise office building with a certain height that exceeds the maximum height is rejected. In other cases, when the proposed characteristics do not directly correspond to the required characteristics, the behaviour of a proposed solution must be derived from the proposed characteristics. For example, the strength of a proposed concrete beam, derived from its shape and
material characteristics, can be compared with the required bearing load. The way the behaviour is derived, and the way alternatives are selected and rejected, is modelled extensively in the GARM.

A technical solution contains functional units. The GARM uses the divide-and-conquer principle to decompose a product (see figure 4.18). The main idea is that a technical solution decomposes into functional units of a lower (decomposition) level. Functional units are related. A functional unit can only be related directly to functional units that are part of the same technical solution.

In the GARM also the relations between functional units are worked out in detail. The relations are called (functional) interfaces, and connect two mated ends (see figure 4.18). Free ends are those ends that are not connected by an interface, but that are part of a higher level port. Every end of a functional unit locates a port of the corresponding technical solution. The network of functional units and interfaces is called a functional network.

The GARM intends to be a neutral reference model, independent of the point of view of any of the participants. Therefore, the functional network models the most neutral relations possible: the topological relations between the topological representations of functional units.

To model topology in a general way the GARM uses meta-topology, as defined by Willems [1988]. Meta-topology is a generalisation of topological representations like the boundary representation (B-rep). It is a semantic topology scheme that allows complete non-manifold topology.

**IMPPACT reference model**

The IMPPACT reference model [Gielingh and Suhm 1993] extends the life cycle mechanism of the GARM. It uses two mechanisms: (1) a concretisation mechanism and (2) a mechanism for the life cycle stages of a product.

The IMPPACT reference model introduces the term concretisation by identifying three types of related products or activities: Required, Proposed and Realised (see figure 4.19). The relations between required and proposed objects correspond to the relations between functional units and technical solutions in the GARM. In IMPPACT this concretisation model is seen as a generalisation of the product life cycle model of the GARM: it is not only valid for products, but also for activities.
Indepedently of (or orthogonal on) this concretisation model, the IMPPACT reference model defines a model for the life cycle stages of a product. The model distinguishes three successive life cycle stages (see figure 4.20):

- Production,
- Operation, and
- Demolition.

The idea is that in each life cycle stage the three concretisation stages appear. Operation of the product imposes functionality requirements on design. These requirements are to be fulfilled by the proposed shape and material of the designed product. Production starts with this designed product as the required stage. A schedule proposes the product's successive production states during construction. After production the realised product is measured. Finally, the product is really used in operation, and the realised functionality of the product is measured.
The IMPACT reference model aims at supporting communication needed for the problem solving processes in a project by combining the concretisation and the life cycle stage models.

ISO classification

The draft version for an ISO classification standard [ISO/TC59 1993] uses a conceptual model of products (the physical objects) and spaces (the non-physical objects) as a basis to classify building information. The conceptual model consists of one general model and several life cycle stage models. The general model defines subtypes and characteristic attributes of Product and Space. The general model includes decomposition relations between the subtypes. The attributes of the objects may appear in three states: as required, as expected, and as measured.

For each project life cycle stage (briefing, functional design, technical design, as-built) a separate conceptual model is developed. These stage models contain parts of the general model. For example, in the briefing model only the building as a whole is relevant. Parts of the building (elements, work sections, or building products) are not included in the briefing model. Object types in the stage models may have attributes in different states. For example, all attributes in the briefing model are as required. Attributes in the functional design model are partly as required, and partly as expected.

An important observation is that concretisation can be used in several design stages. During every stage information is partly as required and partly as expected. Every designer adds new as expected information, but also new as required information on a more detailed level. For this required information another designer has to find expected information.

Reference models not supporting the separation of problem and solution

Most other product or project reference models do not support the separation of problem and solution. Often models only contain selected solutions. Decomposition is modelled straight forward by decomposing solutions in part-solutions. A typical example is the RATAS building model (see figure 3.4 in section 3.1).

The reason why the separation of problem and solution is not supported can be found in the currently used means for communication. Drawings, now the basis of communication, normally only contain the eventually selected solution. The problems that lie behind the selected solutions (the design intent) are normally not made explicit in communication. The reference models cover the current way of communication and contain the geometrical information of those drawings.

The advantage of not supporting the separation of problem and solution is that the resulting models are less complex, and very recognisable.
for people from practice. They are accustomed to solution oriented communication!

The disadvantage of communicating only solutions is that the earlier vague stages of the design process are not supported, neither is the creative problem solving process in a working team. In the beginning of the building process only requirements are known, and some vague alternatives. However, these first design stages require communication too. When criteria, alternatives, and arguments for choices are not communicated, much information will probably fade away. Communication of solutions does not stimulate development of new alternatives, and is therefore not flexible. Also the idea of working teams, in which each participant has his own skills and tasks, is not supported. It is impossible to communicate about what you want from someone, because, when it is only possible to communicate about solutions, you can only tell someone how he has to do it.

Concretisation for the building industry

The second abstraction mechanism of the BPM aims at supporting communication in problem solving processes. The discussion above showed the need for supporting the three evolutionary types of project objects and the decomposition of proposed solutions into related sub-problems. The interpretation of the GARM life cycle stages model used in the IMPPACT reference model aims at fulfilling these needs. The IMPPACT reference model introduced the term concretisation for this mechanism. For this research, concretisation is defined as describing and relating the evolutionary states of project objects in a building project. Applying the concretisation mechanism to building information results in a concretisation model.

The basis of the concretisation model in the BPM is the interpretation of the GARM life cycle stages model as done in the IMPPACT reference model. This concretisation model supports the separation of problem and solution and the divide-and-conquer principles. This implies concretisation with three stages: Required, Proposed, and Realised. The three concretisation stages are valid for all types of project objects.

Figure 4.21 shows the concretisation model in the BPM. A required object can be fulfilled by several proposed objects. One of these proposed objects is selected to fulfil the required object. A proposed object decomposes in required objects that are related to each other. This relation is objectified to Interface. The means that the relation between required objects can also seen as an object type, to which characteristics can be added. The object that is finally realised should conform to the proposed object.
An Analogue to the IMPPACT reference model this concretisation model does not only apply to products, but also to activities and resources.

For example, when activities are specified during the scheduling process, the requirements for an activity have to be specified first. The requirements of an activity deal with its characteristic property Time, with costs of resources to be used, and with the initial and end-state. These requirements can be fulfilled by a number of alternative activities, of which one is selected. This proposed activity will be executed. The results of this activity, with respect to time, cost of used resources, and the initial and end-state, are measured and compared to the proposed activity.

**Use of concretisation**

The concretisation model is developed to support communication in a creative problem solving process in which participants work together in a team. Concretisation is intended to be used any time when communication is required on choices and alternatives. For DfC this means during design of the product and during scheduling of production activities and production resources.

Using concretisation to model project information in a project model might have several advantages:

- With concretisation, a project model is also usable in the rather vague earlier phases of the project, in which only global requirements are known.
- The separation of problem and solution could make it possible to define tasks of participants in a work team, and to relate successive phases in a project.
• Because concretisation supports the communication of the design intent, the constructor knows the reasons for design decisions and can thus propose an alternative that better fits to his construction possibilities. With the concretisation model the designer can exchange this alternative to designer (part of interaction 2 of section 2.2).

• Because concretisation ensures communication on requirements and solutions, it can support performance contracts. Performance contracts specify required achievements, in stead of traditional contracts that specify solutions. For DfC this could enable a shift of some tasks from the designer to the constructor (interaction 6 of section 2.2).

• Concretisation could make a project type model more flexible, because new alternatives can be added and compared easily.

• It might also improve the flexibility of project type models towards differences in activities in design stages between different projects. For example, in preliminary design of one project only vague requirements of a beam are defined, in another project the structural engineer completely designs a beam. Both projects can probably use the same project type model.

4.4 BPM as Combination of the two Abstraction Mechanisms

Together the two abstraction mechanisms described in the previous sections form the BPM. This section describes how the models that result from applying the mechanisms are combined. It also describes the resulting kernel of the BPM.

Combining abstraction mechanisms

Both abstraction mechanisms of the BPM focus on one aspect and suppress all other aspects. For communication in a building project such a limited view is not enough. Both aspects have to be considered at the same time. This can be seen on the examples in section 4.3. The object types of the concretisation model are always used in combination with one of the object types from the product-activity model. The examples use terms like “required product” or “proposed activity”. On the other hand, products, activities and resources are always required, or proposed, or realised.

In NIAM this is modelled with the exclusive/all constraint between the subtypes of project object (as done in figures 4.13 and 4.21). This means that a project object as such does not exist. The exclusive/all constraint in the product-activity model says that it is always a product, or an activity,
or a resource. The exclusive/all constraint in the concretisation model says that a project object always is required, or proposed, or realised.

The combination of these two exclusive/all constraints can be visualised with Venn diagrams [Gielingh and Suhm 1993]. Figure 4.22 shows that the set of all project objects is divided vertically in the three subsets: products, activities, and resources. The set of all project objects is divided horizontally in the three subsets: required, proposed and realised objects. When both mechanisms are used together nine (three times three) subsets appear.

![Venn Diagram](image)

*figure 4.22 combining the exclusive/all constraints of the NIAM diagrams of figures 4.13 and 4.21 explained with Venn diagrams*

**Complete model for the BPM**

The Venn diagrams only explains how new subtypes emerge when several conceptual models with exclusive or exclusive/all subtypes are used. It explains nothing about the redefinition of relations between subtypes. For example, for the BPM it makes no sense to let a required product be fulfilled by a proposed activity. Therefore, relations between subtypes always have to be redefined.

Figure 4.23 shows the new object types that emerge after defining the new subtypes. The figure combines the product-activity model (restricted to DfC, figure 4.17) and the concretisation model (figure 4.21). It shows also shows the redefined relations between the object types.

Relations in the combined BPM are redefinitions of relations in the product-activity model and the concretisation model. First, concretisation is applied to product, construction activity, and construction resource. Then, the product-activity model is applied to the concretisation stages of these project objects.
figure 4.23  *kernel of the Building Project Model as a combination of the product-activity model and the concretisation model*

The complete BPM is formed by the separate models for product-activity (figure 4.17) and concretisation (figure 4.21), the detailed models for Product (figure 4.14), Activity (figure 4.15), and Resource (figure 4.16), and the combined model (figure 4.23).
5 Computer Implementation: PMshell

This chapter answers the second research question: "How can the technical implementation of CA DFC be supported with computers?". It specifies requirements for a computer system that supports implementation of the integration concept, discussed in chapter 4. It presents a prototype project modelling environment: PMshell. PMshell bridges the gap between development of conceptual models, their computer implementation, and their application in practice. PMshell is applied for the implementation of the BPM.

5.1 Implementation Requirements

As pointed out in section 3.2, implementation of product or project modelling can be supported with modelling CASE tools. These CASE tools can be integrated in one project modelling system. This chapter discusses a project modelling system aims at supporting the computer implementation of the concept for CA DFC discussed in chapter 4. This section analyses the users of project modelling and their required way of working. Section 5.2 describes the implementation of PMshell, a prototype project modelling environment. Section 5.3 discusses the application of PMshell for implementing the BPM.

Three groups of users involved in project modelling

As shown in figure 5.1, there are three groups of users involved in project modelling:

- developers of conceptual models,
- programmers of project modelling systems, and
- end-users in practice.

In the following the three groups are referred to as respectively developers, programmers and end-users.
Developers make conceptual models, e.g., project type models (PtMs). Programmers implement the conceptual models in project modelling systems. They structure databases and implement interfaces with existing computer applications. The end-users (the designers and constructors in a building project) use the project modelling systems to exchange information between computer applications used in the sub-processes. The information is exchanged with project models structured according to a PtM.

Research question 2 aims at supporting these user groups. The following discusses their specific requirements and how they want to co-operate.

End-users in practice

The participants in a building project will finally apply project modelling in practice. These designers, engineers, construction managers, constructors, etc. are referred to as the end-users of project modelling.

The end-users want to exchange information between computer applications. They want to store information in such a way that other participants can use it. They also want to retrieve information from others. Chapter 4 argued that meaningful information exchange is possible when structured in a project model according to a PtM. Meaningful information exchange means here that also applications that use information of different types of information can communicate without human interpretation of the information.

Besides information exchange, the end-users also want to use constructability knowledge. The designer wants to check his choices against the general constructability rules during design. The constructor wants to check the finished design against his own constructability rules.
Project model based communication

The end-users want to communicate the project models between their applications. The STEP product modelling approach [ISO/TC184 1993b] identifies three ways to exchange product models between applications:

- file exchange,
- database implementation, and
- application programming interface.

For file exchange, STEP developed the STEP physical file format [ISO/TC184 1993d]. For database implementation, STEP developed the Standard Data Access Interface (SDAI) [ISO/TC184 1993e].

To determine the requirements of end-users on a project modelling system, the three ways of communicating project models are worked out below.

Project model exchange via file exchange

The first way to exchange project models is via physical files. Computer applications write output to and read input from an physical file in a special format (see figure 5.2), e.g., the STEP physical file format [ISO/TC184 1993d]. The structure of information in the file is defined in a PtM. Every application has an output and an input interface. The output interface translates information from the internal database to a physical file. The input interface translates information from the physical file to the internal database of the application.

![Diagram showing communication between applications via a specially formatted physical file]

*Figure 5.2* communication between applications via a specially formatted physical file
Project model exchange via a database implementation

The second way to exchange project models is via a central database. Applications read information from and write information to this central database (see figure 5.3). The database should be freely accessible, e.g. via the Standard Data Access Interface (SDAI) of STEP [ISO/TC184 1993e]. The structure of information of this central database is defined in a PtM, and thus known to all applications involved. An application (1) reads the whole database into its internal data structure via interfaces. Or, it (2) selectively reads that part of the database it needs.

![Diagram showing communication between applications via a central database](image)

Figure 5.3 Communication between applications via a central database

Project model exchange via an application programming interface

A third way to exchange project models is via a so-called application programming interface. This means that applications can call procedures of other applications. This can be realised with a procedural interface (see figure 5.4). Information is stored in an internal database. Upon request of the user, the procedural interface calls procedures that activate applications to manipulate information of the project model.

In the ideal situation the three ways can be used together (see figure 5.5). A modelling system forms the basis. The database of the modelling system is freely accessible, and thus acts as a central database. The system exports and imports physical files, and calls procedures.
figure 5.4  communication via procedure calls

figure 5.5  in the ideal situation applications can communicate via physical files and via a central database and via procedure calls
It depends on the technical possibilities available which option is used. Communication is limited by the computer applications and the computer infrastructure of the participants. When company borders have to be passed and when there are no network connections, physical files are probably the most appropriate. For example, the design of a building can be passed from an architect to a general contractor via a physical file. Within one company a central database seems to be a good alternative. When a designer and a cost estimator use the same database, cost estimations can be updated easily when the design changes. When several applications are often used by one user, procedure calls are probably more appropriate. Take, for example, an engineer with two applications: one for graphical design and one for structural analysis. When he uses a procedural interface, he can alternately design and check against structural requirements.

**End-user requirements**

The requirements of end-users (or project participants) on the modelling system follow from the three ways to exchange information in project models and from the way they want to use the information. Their requirements concern:

- information exchange,
- information input,
- information management,
- application of knowledge rules, and
- flexibility to changing conceptual models.

**Information exchange**

The end-user uses the modelling system to control information exchange. He uses computer applications to generate project model information. He uses the system to store this information in a central database. The system should provide the applications with information in the right format. Some applications require physical files. Therefore, the system should be able to generate and import STEP physical files. Other applications require an open database. Therefore, the database of the system should be freely accessible via SDAI calls.

**Information input**

End-users also want to use the system to input new project information and to inspect information in the database. The system should provide general input and inspection functionality for end-users. The system should also support developers and programmers with options to define and implement this specific input and inspection functionality.
Not all parts of the building process are supported with computers at the moment. Therefore, it should be possible to add project information generated without computers manually to the project model. In the building industry especially the earlier design stages are currently only scarcely supported with tools. When this information is not included in the project model, important information will be lost.

Applications only enable access to parts of the project model. The end-user wants to inspect the project model as a whole. Therefore, the system should provide means to browse through the project model and to query it. An example of the need of query tools is the precast concrete element supplier who wants to know the number of elements in a building. He only has applications for the design and analysis of separate beams. The applications can retrieve loading requirements for a separate beam from the project model. The supplier wants to query the project model about the number of beams with equal requirements in the building.

Information management

Because different project participants exchange much information with the system, the system should also support management of this information. Typical issues are data protection, data conversion and data consistency.

A major concern of many (possible) users of central databases is protection of their information. A supplier does not want to give the general contractor insight in his cost estimations. A structural engineer does not want a HVAC engineer to design a hole in "his" load bearing beam. The system should thus provide possibilities to protect information. For every piece of information it must be possible to define who may change it, who may read it, and to whom it is hidden.

Because different users have different views on the same information, information has to be converted. For example, an application may require axial co-ordinates, but a central database may only use Cartesian co-ordinates. Then, the modelling system should be able to convert geometrical information between the two co-ordinate systems. It should thus be possible to define, implement, and activate such information conversions.

The end-user wants to be sure his project model is consistent with the defined data structure. When the properties of an object change, also related objects have to change. For example, when the length of a beam changes, also its supports have to move. Or at least, the system should signalise that the beam is not supported correctly anymore. Before it is incorporated in the project model, the consistency of information provided by other designers has to be checked. The system should therefore provide means to define, implement, and check consistency rules.
Application of knowledge rules

The end-user wants to check the information in the project model against knowledge rules. For example, the designer wants to check his design against constructability rules. He may want to check a single rule of an object, all rules valid for an object, or all rules valid for one project. With the results from the check the designer knows what rules are violated and where he has to improve his design. The system should provide means to define, implement, and apply these knowledge rules.

Flexibility to changing conceptual models

It should be possible to incorporate updates of conceptual models easily in systems already in use in practice. Likely, use in practice of a PtM will require improvements or extensions. Imperfections may show up. New applications may require information not yet incorporated. The end-user wants to use the same system and the same database when a PtM is updated. No project information should be lost. Consequently, use in practice should not hinder further development of conceptual models.

Developers of conceptual models

The end-users use PtMs to structure their project models. These PtMs are developed by the developers. Developers define conceptual models with object types, properties, behaviour, and knowledge rules. They developers have requirements on:

- modelling languages,
- conceptual models with different scopes,
- modularity, and
- meta-functionality.

The first two requirements correspond to two dimensions of the CIM framework described in section 3.1: the language dimension and the scope dimension. To enable developers to define conceptual models both dimensions have to be supported.

Modelling languages

The developer defines a conceptual model using conceptual modelling languages. He wants to use graphical modelling languages in the first development phases, for presentations, or for overviews. He wants to use lexical languages for detailed modelling. The developer wants to alternate between graphical languages and lexical languages.

As discussed in section 3.1, this research uses NIAM as graphical language in the early modelling phases, and EXPRESS as lexical language in the detailed modelling phases. Developers want to switch between these
two languages without loss of information. The system should support NIAM and EXPRESS and this switching. Because the EXPRESS format is standardised, EXPRESS schemata are often used to exchange conceptual models between developers. The developer wants to be able to import and export these schemata. Also new languages may come up, like EXPRESS-G that should be easy to incorporate in the system.

These requirements concerning the modelling language corresponds to the language dimension of the CIM framework. In section 3.1 the choice was made to use STEP languages and place them in the data levels of IRDS (see figure 3.2).

Conceptual models with different scopes

For the scope dimension of the CIM framework the approach of Tolman [1991] with three types of conceptual models is chosen: resource models, reference models, and project type models. Developers want to be able to define conceptual models with a certain scope based on conceptual models with a larger scope. Because there are three type of conceptual models, there are also three types of developers:

- developers of resource models,
- developers of reference models, and
- developers of project type models (PtMs).

The three types of models can be related via specialisation and/or via reference relations. A developer wants to specialise object types defined in conceptual models with a larger scope. For example, the developers of a PtM specialises the general object types of a reference model. This means that he defines subtypes of object types of the reference models. He also redefines the relations between the subtypes. By doing this he can add project type specific information to a generally applicable conceptual model. Likewise, object types of other conceptual models are referenced. This means that parts of other conceptual models are used in a conceptual models. For example, the developer of a reference model references to a resource model for topology to use its shape functionality.

The system should support the layered architecture of conceptual models with different scopes and both types of relations between these conceptual models.

Modularity

The developer of a conceptual model for a certain field of interest wants to restrict his effort to that field. This is called modularity of conceptual models. For example, the developer of a conceptual model for topology does not want to be bothered by tedious issues as an interface with a database. The engineer that develops a PtM for precast concrete elements in office buildings has no interest in implementing topology. But, of course, he wants to incorporate topology implemented by others.
The system should provide developers with possibilities to bound the field of interest. Conceptual models and object types outside this boundary can only be used, not changed. The other way around, the developer can define what parts of his conceptual model can be used by others and what parts are hidden.

Meta-functionality

Developers want to have possibilities to guide other developers how to specialise their models easily. They want to offer them meta-functionality: specific functionality to support the development of conceptual models based on the conceptual model involved.

For example, the developer of a reference model defines a general mechanism to decompose products. He knows that developers of PtMs want to redefine this mechanism. A concrete beam does not decompose in any product, but (among others) in a concrete body and reinforcement bars. He provides the PtM developer with meta-functionality to redefine the decomposition mechanism. With this meta-functionality, the PtM developer can easily restrict the possible object types.

Programmers of modelling environments

In between developers and end-users the programmers, the third user group, do their job: implementing application interfaces, incorporating procedure calls, and tailoring end-user interfaces. The requirements of the programmers for the project modelling environment concern general software development requirements as openness and modularity. An open system enables programming users to incorporate other applications (or parts of applications) easily. An open system is also easily extensible, and independent of the computer platform. Modularity ensures manageability, reuse, and extendibility of programs.

Bridging the gap between the three user groups

For a project modelling environment it is not only important to support the three user groups separately: it is essential to bring developers, programmers and end-users together. Transition from developer, via programmer to end-user should go smoothly. This means: without loss of information, and without unnecessary extra work. Also the feedback loops should be supported.

Developers want to have the possibility to check their conceptual models in a prototype information system. This means that they want to have the possibility to act as an end-user in a prototype version of the end-user system. This prototype should be easily derivable from the definition of the
conceptual model. This is called *rapid prototyping*. This can be done by automatically generating computer programming code from the conceptual models, and by automatically structuring a database.

Rapid prototyping also requires the simulation of functionality of applications. Incorporation of (parts of) applications, or interfacing applications with project models, is a difficult job, which requires programming skills and time. When checking a data structure, the developer is not interested in the exact functionality of the applications. He only wants to simulate functionality with regard to supplying the data required and storing the data generated. Therefore, the system should provide options to define (simple) behaviour for object types.

End-users only instantiate PTMs into project models. But, also developers of resource models and reference models want to check their conceptual models without having to specialise them first to PTMs. For example, the developer of a resource model for topology wants to instantiate his object types, and to test the behaviour of topological objects. Also the developer of reference models wants to instantiate his general object types. This means that also the generally applicable conceptual models should be instantiable.

Programmers should be able to built on the work of developers. Developers end with conceptual models and prototype implementations. Step-by-step programmers replace simulated functionality by incorporating applications, or by developing interfaces. When they gradually extend the prototype version of the developers, the time gap between development and final implementation decreases. Likely, they encounter some imperfections in the conceptual models. Therefore, the way back from programmer to developer should be open.

Like developers, programmers want to use a standard end-user interface to test their implementation rapidly. This standard user interface should be automatically derivable from the PtM. When they have checked the implementation, they want to be able to tailor the user interface easily to the specific needs of the end-users.

For the development of conceptual models bridging the gap between developers, programmers, and end-users is of major importance. An integrated project modelling environment might function as a bridge between the three groups of users.

Because the work of programmers follows naturally from the work of developers, it is not possible to tell where one ends and when the other starts. Both groups require practically the same tools. Therefore, the following treats the two groups as one. When emphasis lies on conceptual modelling, they are referred to as developers. When emphasis lies on implementing functionality, they are referred to as programmers.
5.2 Implementation of PMshell

To test the integration concept of chapter 4 and the requirements specified in the previous section a prototype implementation has been developed, called PMshell\(^1\). This section discusses the computer environment, architecture, data structure and functionality of the kernel, and the user interfaces. A detailed description of PMshell can be found in [Luijtjen and Luiten 1993].

Computer environment

Conditions for implementation are formed by the computer environments available at the research partners. PMshell is developed in close co-operation with TNO Building and Construction Research. TNO has used PMshell in several demonstration projects [Kuiper, Tolman, et al. 1992; Kuiper, Boender, et al. 1992; Mooij 1993]. PMshell is also used for practical work at the university [Tolman 1990]. The co-operation with TNO has lead to choices concerning:

- hardware platform,
- computer languages, and
- incorporated tools.

PMshell is implemented on SUN Sparc stations 1 and 2 under UNIX.

As discussed in section 3.2 the object-oriented paradigm closely corresponds to conceptual modelling. Therefore, PMshell is implemented in an object-oriented programming language. Eiffel (version 2.3) [Meyer 1988] is chosen\(^2\), because it supports most of the required object-oriented characteristics. Eiffel is a pure object-oriented language without a procedural history, like other object-oriented languages. This choice was justified by the results of a research of Woodruff [1993] comparing three object-oriented languages. He states that Eiffel is truly revolutionary in that it provides tools for specifying and building reusable systems. Although there are still some performance related issues, Eiffel was found superior for large application development over Objective-C and C++.

Computer implementation experience at TNO serves as basis for PMshell. TNO tools incorporated are:

- an EXPRESS schema parser [Eijs 1993],
- EXIS, a database with tools [Bruijn 1990],
- STEP physical files tools [Eijs 1992], and
- a Configurable Graphical Editor [Vogel 1991].

\(^1\) To give an indication about the effort put in the implementation of PMshell and the surrounding tools: the kernel, the system interface, and the supporting libraries contain 88 Eiffel classes with 20,000 lines of code in total.

\(^2\) TNO's commercial version of PMshell will be implemented in C++. 
Architecture

The architecture of PMshell results directly from the main user requirements. The most important requirement is an integrated system, in which developers can easily switch between conceptual modelling, (prototype) programming and acting as end-user. Also important are modularity of the system and of conceptual models, and support of STEP languages.

The resulting architecture consists of layers with domains. A layers in PMshell can be interpreted as a group of conceptual models with the same scope. A domain represents a field of interest, a so-called universe of discourse. A domain implements a conceptual model. Like conceptual models, domains are related.

Figure 5.6 shows an example of a domain structure. Domains are symbolised with boxes, the relation between domains with arrows. An arrow points from a domain to a domain it uses. In this context "uses" means that object types in the using domain specialise object types of the used domain, or reference to object types of the used domain. For example, Concrete Beam in the PtM for Concrete Structures, specialises Product in the BPM domain. And, Product in the BPM domain references to object types in domains for Topology and Geometry for shape definitions. Also in a layer domains can be related. For example, the object type Office Building in the PtM for Office Buildings references to the domain for Concrete Structure for its bearing structure.

The notions used in PMshell have corresponding notions in EXPRESS. A domain in PMshell corresponds to a schema in EXPRESS (which corresponds to a conceptual model). The specialisation and reference relations between domains corresponds with the "reference from" or "use" statements in EXPRESS. An object type of a domain corresponds to an entity in EXPRESS. The subtype relations between object types in PMshell corresponds to subtyping in EXPRESS. And, the relations between object types in PMshell correspond to complex attributes in EXPRESS.

---

1 This implies that some users (e.g., structural engineers, concrete element suppliers) look at a conceptual model like Concrete Structures as a separate PtM, others (e.g., architects of office buildings, or architects of row houses) look at it as a resource model.

2 A complex attribute in EXPRESS denotes an attribute with an entity as type. This means that the attribute value references to an object. When an attribute is not complex but basic, the attribute value references to a string, integer, real, character, or boolean.
The first domain in PMshell is the *kernel*. All other domains use the kernel. The kernel contains general functionality, like database access, support of STEP languages, and connection with user interfaces. It contains functionality for developers to define new domains with new object types, and to generate computer code in Eiffel. The kernel also contains functionality for end-users to instantiate project models and to apply the defined functionality. By using the kernel all domains inherit this functionality.

In fact, without the effort of developers PMshell is only the kernel with its tools. The other domains are added by the developers. The kernel and the tools provide an environment to develop, implement, and apply conceptual models in domains. Figure 5.7 shows the system architecture of PMshell. The dashed line indicates the border of PMshell. PMshell consists of the kernel, a system interface and interfaces.

The user (developer, programmer, or end-user) communicates with PMshell via a *user interface*. This user interface communicates via a protocol with the *system interface*. The system interface calls procedures in the *kernel*. In this way, the conceptual model (for developers and programmers) or the project model (for end-users) can be manipulated. For example, new object types can be added, or functionality can be activated. Upon request, the kernel gives information about the state of the (conceptual) model. Via the kernel, information can be stored in or retrieved from a database. Via the kernel, EXPRESS files and STEP physical files can be imported and exported. Via the kernel, also Eiffel programming code and documentation can be generated for new domains. After
compiling the generated code, the new domain becomes part of the system.

---

**Data structure of the kernel**

The kernel is the central part of PMshell. The data structure and functionality of the kernel result from the requirements stated in section 5.1. The data structure closely follows the object-oriented definition of object types, as described by Meyer [1988]. Parts of the kernel are used by end-users, other parts by developers. Some parts are used by both.

Figure 5.8 shows the object types of the kernel and their subtype relations. The main object types are Domain, Object, and Feature. Features are Properties or Routines. Properties denote characteristics of objects, Routines denote behaviour. Properties are arguments, attributes or functions. Routines are functions or procedures. Function is thus subtype of both Property and Routine: it is a characteristic that has to be derived. Assertion is subtype of Function, because it denotes a special function with a boolean result. Just for implementation reasons a supertype of all object types is introduced: Pm-thing. Pm-thing has general mechanisms for generation of program code, exchange of data, database management.

---

1 The kernel is implemented in close co-operation with Bart Luijten. Others helped with suggestions and comments.
and user interface control. The subtypes redefine this general functionality.

![Object types and subtype relations in the kernel](image)

Figure 5.8 object types and subtype relations in the kernel

Figure 5.9 shows the relations between object types in the kernel.

A domain contains objects. For specification of these objects, a domain uses other domains via a domain-interface. Domain-interfaces are related to parent domain-interfaces. A domain-interface describes the object types that can be used in child domains. This implements the layered architecture of PMshell. A domain has assertions as global rules the whole model should conform to.

An object has defined features: attributes, functions, procedures and assertions. Assertions are the local rules the object should conform to. An object also has redefined and renamed features. An object can have partners. An object without partners defines a single subtype relation. An object with partners defines a multiple subtype relation.

The five kinds of feature have specific characteristics. The type of a property (attribute, function or argument) is a pm-thing. This type is restricted to a basic type, a property type, or an object type. A basic type is a boolean, an integer, a real, a double, a character, or a string. An object
type corresponds to a complex attribute of EXPRESS. A property type de-
finesthe type equal to the referenced property. A property has a lower
bound and an upper bound, which are the minimum and maximum num-
ber of values of the property. This is called the cardinality of a property.
In addition, a routine has a body and arguments. The body of a routine is
the Eiffel implementation, arguments are the input parameters.
Every object type has a unique identifying string and a comment
string for documentation. This is modelled in Pm-thing, the global super-
type.

![Diagram of object types in the kernel](image)

**Figure 5.9** relations between object types in the kernel

The data structure of the kernel is used both by developers and end-users.
The two groups of users have different interpretations of the data struc-
ture and use different parts.
Developers interpret instances of Object as new object types. The type
of an instance denotes the supertype. An instance specialises its supertype.
This subtyping mechanism is called *specialisation-by-instantiation* [Willems, Kuiper, et al. 1991]. An instance of Domain is a new *domain*. Developers define new features for object types. They also redefine and rename inherited features for object types.

End-users interpret instances of Object as single *objects*. An instance of Domain is a *project model* for a particular project. End-users activate and inspect the features defined by developers.

**Functionality of the kernel**

The kernel contains the general functionality of PMshell. The kernel has features for:

- administration of the specification of a new domain,
- description of the interface of an object type,
- generation of programming code and documentation, and
- data management.

*Administration of the specification of a new domain*

Specification by the developer of a new domain requires an administration of new objects, new features and renamed and redefined features. Object types in the kernel have routines to handle the administration.

Part of the administration concerns definition of new object types, their characteristics and their behaviour. These definitions are stored in the data structure of the kernel. Instances of Object can be added to the domain. For an object new features can be defined, inherited features can be renamed, and partner objects can be selected. Developers can define functionality for the end-user. Examples are, functionality for data input, data inspection, data conversion and application simulation. Developers can also define functionality for other developers: the meta-functionality.

Another part of the administration concerns redefinition of the type of properties. This is realised with *data fields* in an object. Every property has a corresponding field in an object. Attributes, functions and arguments are subtype of property, and thus have a field in an object. The redefined type of an inherited property is stored as a value in that field. Because only objects conforming to the type of a field can be stored in that field, a redefinition of a property type always makes sense.

The final part of the administration concerns the definition and redefinition of the body (or Eiffel implementation) of routines. The body of a routine is stored in a physical file with the name of the routine and placed in a directory with the name of the object type. If there is such a file for an inherited routine, this is interpreted as a redefinition of the body of the routine. The body file contains Eiffel programming code. In Eiffel it is
possible to incorporate applications in routines by including procedure calls. These applications may be written in other computer languages.

**Description of the interface of an object type**

Uniform treatment of objects of different object types requires an uniform description of their interface. Therefore, objects contain lists of defined attributes, functions, procedures and assertions\(^1\).

With this description the features of every object are known and the system interface (see figure 5.7) can handle every object in a similar way. End-users use the description to activate and inspect features. Developers use this description to activate and inspect meta-functionality, and to re-define and rename inherited features.

Distinction must be made between the administration and the interface description. The administration is only used by the developer to define new objects and features. It is based on the data structure of Object. It is different for every instance of an object type. The interface description is used by both users. It tells what features are available. The description is identical for all instances of an object type.

**Generation of programming code and documentation**

Developers generate programming code and documentation for the newly defined domain.

For generation of Eiffel programming code the kernel uses the administration. After re-compilation of the new code, the new object types become part of the system. Then, the object types are available for specialisation by developers, or for instantiation by end-users.

For the generation of documentation the short option of Eiffel is used in combination with a small program. The short option filters implementation details from Eiffel code. The program generates an easy to read format. For extra explanation of object types and features the developer can use the comment string, available in every pm-thing.

**Data management**

The kernel enables users to store their data permanently in a data base or in a physical file. Developers can store their conceptual models. End-users can store their project models.

---

\(^1\) These lists are identical for all objects of one object type. Therefore, the lists are implemented in Eiffel as so-called *once* functions. This means that they are only instantiated once for all objects of one object type. Because the lists are implemented as (once) functions, they do not appear in the data structure of figure 5.9.
To store data in a database entities corresponding to the object types are generated in an EXIS database. Entities are generated for the object types of the kernel and for the object types defined by the developers. The database is structured according to the information structure used for administration.

For communication with non PMshell users data can also be stored in physical files with a standardised format. For developers the definition of a domain is stored in an EXPRESS schema. For end-users the project model is stored in a STEP physical file. These physical files can also be imported in the system, for which existing TNO tools are used.

User interfaces

The user interfaces of PMshell\(^1\) conforms to the requirements of section 5.1. The architecture is open and modular. One user interface supports the modelling language NIAM.

The openness and modularity of the user interface is guaranteed by the protocol connection between the user interface and the system interface (see figure 5.10). The protocol is described with Wirth Syntax Notation [Wirth 1977]. The user interface and PMshell are separate programs. New user interfaces that support the protocol as described can be added without changing PMshell.

Three user interfaces are implemented:

- a menu driven OSF-Motif user interface,
- a graphical NIAM-like user interface, and
- a Macintosh user interface.

The menu driven user interface uses the OSF-Motif user interface description. It supports all functionality of PMshell, both for developers and end-users. The menus are built with the interface builder TeleUSE [TeleSoft AB 1990].

The NIAM-like user interface uses the Configurable Graphical Editor for defining the data structure of conceptual models. For other parts, like behaviour, it uses the menus of the OSF-Motif user interface. The graphical interface is only usable for developers. The user interface is NIAM-like, because it lacks some NIAM functionality and it adds some object-oriented functionality. It lacks, among others, possibilities to define constraints between subtypes and between relations. It adds graphical representations for functions and redefined attributes and functions. Because the object-oriented paradigm does not support bi-directional relations, the user interface only supports uni-directional relations. It has some special graphical symbols (see figure A.1 in appendix A).

\(^1\) The user interfaces of PMshell are implemented by Bart Luijten, Anita Eijs, Ruud van Mieghem, Marcel Boender and Peter Willems.
The Macintosh user interface is menu driven. It has the same functionality as the OSF-Motif user interface, but with the Macintosh look-and-feel. The user interface runs on a Macintosh. PMshell on a UNIX system. The programs communicate the protocol via a network or a modem.

5.3 Implementation of the BPM with PMshell

This section describes the PMshell implementation of the BPM, as described in chapter 4. Only some representative properties of the BPM are implemented.

Layer and domain structure

The PMshell implementation\(^1\) consists of five layers, as figure 5.11 shows. The lower three layers correspond to the three scopes of the CIM framework, described in section 3.1. The resource layer implements topology

\(^1\) To give an indication of the effort put in the implementation: the 7 domains contain 75 object types with 30.000 lines of code in total, of which 75 % is generated automatically.
and geometry. The reference layer implements the BPM. The upper two layers, the kernel layer and the graph layer, implement respectively general modelling functionality and network functionality. Each layer consists of domains. The figure shows the layers and the domains with their relations.

![Diagram showing the layers and domains implemented in PMSshell](image)

*Figure 5.11 The layers and domains implemented in PMSshell*

The kernel layer is described in section 5.2. The graph layer, the resource layer, and the reference layer are described globally in this section. Appendix A shows the corresponding NIAM-like schemata of the graphical user interface. Chapter 7 describes an example of a project type model: a PtM for concrete structures.

**Graph layer**

Network functionality is implemented in two domains: Basic-Graph and Sub-Super-Graph. Basic-Graph implements basic network functionality. Sub-Super-Graph implements a network of parent and child graphs and a filtering mechanisms for deriving child graphs.
Basic-Graph

Basic-Graph implements basic network functionality\(^1\). Sowa [1984] gives a mathematical background of graphs.

A graph consists of nodes and links, as figure 5.12 shows. A link connects two nodes. A link has a direction: it has a source node and a target node. A node has arriving and departing links. In figure 5.13 this is modelled in a conceptual model.

---

![Graph Diagram]

**figure 5.12** a graph consists of nodes, connected by uni-directional links

---

![Conceptual Model Diagram]

**figure 5.13** a graph with nodes and links

Network functionality can be used anytime relational networks are relevant, e.g., for activity networks. Basic-Graph implements general network functionality that is specialised in other domains. In activity networks states of the product denote the end of preceding activities and the start of succeeding activities. When State and Activity are defined as subtypes of respectively Node and Link, new functionality can be added. For example, when activities are given a duration, the duration of a series of activities can be calculated. This idea can be extended to critical path analysis, optimisation, etc. Activity network functionality can be incorporated in a scheduling application.

\(^1\) Implementation of Basic-Graph builds on prior work of Willems [1992].
Basic-Graph contains functionality for:

- building graphs,
- querying graphs, and
- constraining types and cardinality.

Building graphs

Graph, Node and Link contain end-user functionality to build graphs by connecting links and adding nodes and links to a graph. Links can only exist if they connect two nodes. Basic-Graph also contains the negative counterparts of these procedures: deleting nodes and links, and disconnecting (and deleting) links and nodes.

In addition, Graph contains functionality to manipulate graphs, such as merging, intersecting, subtracting, duplicating, and clearing graphs.

Querying graphs

The end-user uses functionality to query a graph and its nodes and links. For example, the function "connected nodes" determines the nodes that are directly and indirectly connected to a node via a selected type of links. A filtering mechanism filters the links of a selected type out of the arriving (or departing) links. Another filtering mechanism filters the target (or source) nodes of a selected node type out of all departing (or arriving) links of a selected link type.

Graph also contains functionality to identify the relation with other graphs. Does this graph contain exactly the same nodes and links as the other? Or, in other words, is this graph identical to the other? Does this graph contain links and nodes that are all part of the other? Or, in other words, is this graph a sub-graph of the other? Does the other graph contain links and nodes that are all part of this graph? Or, in other words, is this graph a super-graph of the other?

Constraining types and cardinality

Programmers want to constrain the types of links and nodes that can be connected to a node. For example, the programmer of a concretisation model (as described in section 4.3) of products wants to define that required products can only be fulfilled by one proposed product. Therefore, he defines the object type Required Product and Proposed Product as subtypes of Node. He defines Is-fulfilled-by as subtype of Link. Then, he wants to constrain the departing links of Required Product: he only allows one instance of Is-fulfilled-by in the list of departing links. Standard object-oriented redefinition would not satisfy: it would imply that only is-fulfilled-by links are allowed, and in any number. Then, a required product can only be part of a concretisation network. It can not be
part of, e.g., an activity network anymore. This is not what the programmer wants.

To enable programming-users to constrain link types and node types, Node has special meta-functionality. The building graph procedures, as used by the end-user, check the defined constraints before execution of the graph building functionality. Figure 5.14 shows the data structure of this part of Basic-Graph.

![Figure 5.14: meta-functionality in Basic-Graph: constraining link types, node types and cardinality](image)

The programming-user defines link types that are constrained by adding instances of Link-constraint to the pu-arriving\(^1\) or pu-departing link constraints. For each link-constraint he defines which node-types are allowed. For each allowed-node-type he defines the cardinality by defining the upper and the lower bound. When he is ready, he evaluates the defined link-constraints. This results in an initialise procedure that is executed anytime a new instance of the node type is created by an end-user. Execution of this initialise procedure results in lists for departing and arriving link constraints of the instance. These two lists are consulted before graph building procedures are executed. When the intended graph building procedure would result in a state not conforming the defined constraints, the procedure is not executed and a warning message is shown. In addition, Node defines two local rules that check the state of the instance.

Figures A.2 and A.3 in appendix A show the conceptual models of Basic-Graph as modelled with the graphical (NIAM-like) user interface of PMshell.

---

\(^1\) The prefix “pu-” before a feature name stands for “programming user”. It denotes that the feature is meta-functionality to be used by programming users.
**Sub-Super-Graph**

Important graph functionality is a graph filtering mechanism. Sub-Super-Graph implements a mechanism to define a hierarchy of sub- and super-graphs, and to filter the links and nodes of a super-graph. Sub-graphs contain per definition part of the links and nodes of the super-graph. Which links and nodes is determined with a filtering mechanism.

Figure 5.15 shows that an S-graph, which is subtype of Graph, has at most one super-graph and several sub-graphs. An s-graph has four parameters that are used for the filtering mechanism.

![Diagram](image)

*Figure 5.15: Graphs in a hierarchy with filtering mechanisms in Sub-Super-Graph*

S-graph has two filtering mechanisms: (1) a keyhole mechanism and (2) a sunglasses mechanism. The keyhole mechanism enables the end-user to define a sub-graph that contains part of the nodes and all links connecting those nodes. The end-user selects a top-node and a link type. The filter procedure filters all nodes connected to the top-node via links of the selected link type. The sunglasses mechanism enables the end-user to define a sub-graph that contains nodes and links of selected types. The two mechanisms can also be used in combination.

The contents of a sub-graph is updated when deemed necessary. When new nodes and links are added to the super-graph, it is possible that they also belong to the sub-graph. To filter the nodes and links of its super-graph again, a sub-graph stores the selected top-node, node type and link types. When a node or link is added to a sub-graph, it is also added to its super-graph.

Figure A.4 in appendix A shows the conceptual model of Sup-Super-Graph as modelled with the graphical (NIAM-like) user interface of PMshell.
Resource layer

The resource layer contains domains for modelling and manipulating shape. The description of shape is separated in three parts: shape definition, shape representation and shape presentation (see also section 3.1). For each a domain is implemented, respectively Shape, Generalised-Topology and PMSGQT.

Shape presentation: PMSGQT

For presentation of shapes the Graphical Query Tool (GQT) [Balkenende 1990] of TNO is incorporated in a domain\(^1\). The GQT enables visualisation of simple wire frames.

The GQT is a set of procedures controlling a GKS\(^2\) window. It has been part of ProMod, the former prototype product modelling environment of TNO. The procedures are written in C.

Luijten incorporated (part of) the GQT procedures in PMSGQT (the PMshell GQT domain). PMSGQT contains the basic object types GQT-Model and GQT-Object, and subtypes of GQT-Object: Vertex, Edge, Loop, and Face. The topological relations between the GQT-objects are very simple. A face consists of one outer-loop and zero or more inner-loops. A loop has three or more vertices. An edge has exactly two vertices. And, a vertex has three co-ordinates. The geometry is also very simple: all lines are straight, and all faces are flat.

The end-user controls visualisation from within an instance of GQT-model. A GQT-model contains GQT-objects. Vertices, edges and loops can be presented in a GKS window with three 2D side views and one perspective 3D view. The end-user can manipulate the presentation from a GQT-model. He can set the view-point and eye-point. He can zoom in, zoom out, and choose which view he wants in the main window. Finally, he can examine the vertices and edges shown in the GKS window.

Figure A.5 in appendix A shows the conceptual model of PMSGQT as modelled with the graphical (NIAM-like) user interface of PMshell. Figures 7.14 and 7.15 in chapter 7 show examples of shape presentations generated with PMSGQT.

\(^1\) PMSGQT, the domain for shape presentation, is implemented by Bart Luijten.
\(^2\) GKS stands for the Graphical Kernel System, an ISO standard for computer graphics.
**Shape representation: Generalised-Topology**

Generalised-Topology implements a general view on topological relations\(^1\). From this general view more commonly used topological representations can be derived, such as CSG and B-rep. It is an implementation of work of Willems [1993]. The domain consists of a topology part, a geometry part and a general mathematical part.

The main object type in the topology part is Cell. A cell denotes a topological instance. The dimension of a cell denotes what kind of topological instance: 0 denotes a vertex, 1 an edge, 2 a face, and 3 a volume. A cell is defined by the cells it encloses. Therefore, it is connected to other cells via enclosure relations. To be able to use network functionality, Cell is subtype of Node and Enclosure of Link. A cell can also be defined by the cells it is bounded by. Such a cell is called a contour-cell. Contour-cell is subtype of Cell. A contour-cell may function as a boundary cell for another contour-cell. The dimension of the bounding cell is always exactly one less than the dimension of the bounded cell. A contour-cell is related to the contour-cells it bounds or by which it is bounded via boundary links. Boundary is subtype of Enclosure.

To define the geometry of a cell it resides in a space. Spaces are embedded in other spaces via space-links. Space is subtype of Node, Space-link of Link. A space-link has a place and a deformation, which denote respectively how the target space is placed and deformed relatively to the source space. A place has a location vector, a primary orientation vector, and a secondary orientation vector. These three vectors are enough to define the position and orientation of the target space [Boender and Pronk 1991]. A deformation defines procedures that deform vectors. In the current version three deformations are implemented: (1) a linear deformation that results in straight lines, (2) a shift deformation that results in a translation, and (3) a circular deformation that results in circular lines. The linear deformation is default.

The domain contains two general mathematical object types: Matrix and Vector. The dimension of a matrix can be 1 to 4. A matrix can be transposed, multiplied with another matrix, multiplied with a scalar, duplicated and rotated around a vector over an angle. The dimension of a vector can be 0 to 3. The scalar product, the vector product, and the angle with another vector can be calculated. A vector can be normalised, duplicated, multiplied with a scalar, multiplied with a vector, added to another vector, and subtracted from another vector. Also the length of a vector can be calculated. Functionality of Matrix and Vector is used to determine the placed and deformed spaces of a cell.

The placed and deformed space of a cell is visualised with PMSGQT. A cell has a GQT-model and functionality to derive and visualise it.

---

\(^1\) Generalised-Topology, the domain for shape representation, is implemented by Peter Willems.
Figure A.6 in appendix A shows the conceptual model of Generalised-Topology as modelled with the graphical (NIAM-like) user interface of PMshell.

Shape definition: Shape

Generalised-Topology is to general to use it easily. For example, when representing a simple block 103 instances (27 cells, 11 spaces, 11 space-links, and 54 boundaries) have to be created and linked by hand. Therefore, Shape\(^1\) implements standard shapes that automatically derive those instances from relatively few parameters.

The main object type is Shape-definition. All other object types in Shape are its subtype. A shape-definition has a cell as representation. It uses an anchor space to specify the origin of the co-ordinate system, in which the space of a representation cell is placed. It has an evaluation procedure that calculates the representation cell from specific parameters. Each subtype of Shape-definition has specific parameters and a specific evaluation procedure.

Simple subtypes of Shape-definition implemented are: Point, Straight-line, Circle-segment, Polygon, Rectangular and Cylinder. These subtypes have their own specific parameters, with which the redefined evaluate-procedures derive the corresponding representation cells.

Complex subtypes of Shape-definition implemented are: Translation-sweep, Translation-block and Compound. A translation-sweep has a surface polygon and a sweep-vector as specific parameters. The 3D shape is derived by sweeping the surface over the vector. A special type of translation-sweep is the translation-block: it defines a rectangular surface of size block-length and block-width, and an orthogonal vector of length block-height. This block has a very simple function that calculates its volume. The complex shape-definition Compound has a list of shapes and a list of places as specific parameters. A compound adds the shapes in the list to one shape. The part shapes are placed relatively to the anchor of the compound with the place vectors.

Figure A.7 in appendix A shows the conceptual model of Shape as modelled with the graphical (NIAM-like) user interface of PMshell.

Reference layer

The reference layer contains an implementation of some representative properties of the BPM. It consists of two domains: Product and Product-Activity. Product implements concretisation of the product during design.

\(^1\) Implementation of Shape, the domain for shape definition, builds on prior work of Peter Willems.
Product-Activity implements relations between products, activities and resources.

**Product**

The Product domain implements the concretisation mechanism of the BPM for products (see section 4.3). It implements the relations between required and proposed products: required products are fulfilled by proposed products, which decompose in required products of a lower order that have network relations. It specialises Basic-Graph functionality for the network relations. It uses Sub-Super-Graph for deriving project overviews. And, it uses Shape for the definition of the shapes.

Figure 4.21, the concretisation model of the BPM, is applied to product information. Product, the supertype of Required and Proposed Product, is subtype of Node (see figure 5.16). Is-fulfilled-by, Decomposes-in and Interface are subtype of Link.

---

**figure 5.16** concretisation of the product during design using Basic-Graph functionality

Tedious graph properties are hidden for the end-user. He does not see all the instances of Link. When he wants to traverse the decomposition tree, he can go directly from one node (a required or a proposed product) to the other, without passing by the connecting links. In addition, the relations can be defined directly from within the nodes. When defining the relations from within the nodes consistency is assured: the end-user can only choose from a list of correct nodes. For example, the end-user can
only connect a required product to another required product that is part of the same proposed product. Also the defined constraints (see Basic-Graph) are checked.

To simplify product models, Interface can also be seen as a Required Product. Therefore, every instance of Interface has a RP-part of type Interface-RP\(^1\). Interface-RP is subtype of Required Product and can thus be fulfilled by a proposed product. When an interface is initialised, its RP-part is also created. An interface-RP is automatically part of the decomposition tree. Interface-RP can be fulfilled by Structural Connection, a subtype of Proposed Product.

To help the programmer of project type models, Required Product and Proposed Product have meta-functionality, which enables easy redefinition of the decomposition and network relations. It uses Basic-Graph meta-functionality to constrain link and node types. A programmer can easily define the first version of his PtM by defining new subtypes of Required Product and Proposed Product, by redefining the shapes of these new types, and by constraining the relations. For a later version, he extends the first version with functionality belonging to his PtM, without bothering decomposition, functional networks and shapes.

Shape of products is implemented using the Shape domain. Every product has a shape-definition. From a product the representation cell (using Generalised-Topology) and the presentation model (using PMSGQ) can be derived and visualised on-line. For a required product the shape denotes an abstract shape showing the approximate shape of its proposed product. For a proposed product the shape denotes its exact shape or, when it is decomposed, the bounding box of the shapes of its parts. In addition, a proposed product has a compound shape and an analysed shape. The compound shape is the combination of the shapes of the part required products. The part products are placed relatively to the anchor of shape of the proposed product. The analysed shape is the combination of the shapes of the part products several decomposition levels deep. The shape of a proposed product is placed relatively to the anchor of the shape of the required product it fulfils. The compound and analysed shapes can be derived from the decomposition tree, and can be visualised in a GQT window upon request of the end-user.

A project contains all required and proposed products (the nodes) and their connecting links. To get an overview Project is defined as subtype of S-graph. This means that the nodes and links of a project can be filtered in sub-graphs. Now, the programmer of PtMs can define sub-graphs with

\(^1\) A better OO solution would have been to use multiple sub-typing (or, multiple inheritance) in this case: Interface as sub-type of Link and Required Product. Alas, Eiffel version 2.3 has some insurmountable difficulties with repeated sub-typing. And because all object types are (indirectly) sub-types of Object, multiple sub-typing in PMshell always implies repeated sub-typing.
part of the total project information. For example, in Project a sub-graph can be derived containing the required products and interfaces of the functional network of a proposed product.

Figure A.8 in appendix A shows the conceptual model of Product as modelled with the graphical (NIAM-like) user interface of PMshell.

Product-Activity

Product-Activity implements the information structural part of the product-activity mechanism of the BPM (see section 4.2). It can be used to store planning information, as it results from a scheduling and a resource planning application. Concretisation of activities during planning is not supported, neither is decomposition of activities.

Figure 5.17 shows the information structure implemented. Proposed products have constructions states that succeeds each other. Activities change states: they start with a state and end with a new state. Activities use resources.

![Diagram](image)

*figure 5.17 relations between products, construction states, activities and resources*

Only simple functionality is implemented to copy information from the scheduling and resource planning applications.

Figure A.9 in appendix A shows the conceptual model of Product-Activity as modelled with the graphical (NIAM-like) user interface of PMshell.
6 Organisational Implementation

This chapter discusses research question 3: "How can the integration concept for CA DfC be implemented organisationally?" The chapter describes some current developments that could be supported by the improved integration as proposed in the previous chapters. It then outlines a possible migration route with consecutive stages that the participants in a building project could follow to implement CA DfC. Finally, it shows how external organisations might speed up or slow down this implementation.

This chapter only has an indicative character: results are mainly expectations and global guidelines.

6.1 Organisational developments

Chapter 4 and 5 discussed a concept for the integration of information and knowledge used in design and construction applications, and outlined how this integration can be supported with computers. The concept for CA DfC is based on project modelling. Section 3.3 described some organisational developments that might be supported with improved integration of design and construction. This section highlights some of these organisational developments and indicates how the strategy for CA DfC developed in the previous chapters may support these developments.

Increasing the influence of suppliers

According to the Dutch technology policy advisory board for the building industry [ARTB 1993], a more important role of the suppliers is of great influence for rationalisation of the construction process. Increasingly, suppliers control the design, production and assembly on site of their products in an industrialised environment. Independent of individual projects, they are able to standardise their products and optimise their production systems.

Industrialisation of the production of building elements implies the introduction of production automation with numerically controlled (NC)
machines. When product information is available electronically, control commands for these machines can be derived automatically. This idea is used in an exchange format for reinforcement information developed in the Netherlands [Vos and Elkhuizen 1992]. When using this format, the reinforcement supplier receives the reinforcement bill of quantities, electronically from the structural engineer. The supplier can derive control commands for the NC cut-and-bend machines automatically from this bill of quantities.

CA DfC might support this rationalisation by integrating the sub-processes of the suppliers. A project modelling system could be useful for integrating the project information used by the design, fabrication, and assembly departments of a supplier. Information about resources and constructability knowledge can be made available to the designers. A project model might provide the information about the product (and the construction activities) needed for deriving the NC commands.

**Automating construction on site**

CA DfC might also support flexible automation on site. Automation on site can be realised with controlled building environments, and construction robots or NC machines.

An example of a NC building environment is SMART of Shimizu in Japan [Miyatake, Yamazaki, et al. 1992]. SMART stands for the Shimizu Manufacturing system by Advanced Robotics Technology. It consists of a self-elevating, automated assembly platform, which provides an integrated building construction environment for high-rise buildings. The system assembles prefabricated components of the steel structure, the walls and the floors. Joints are specially designed to enable automated assembly. The assembling process is orchestrated by real-time computer control. Shimizu aims at halving construction time and cost with SMART. In addition, they hope to improve the poor image of the building industry. This should help to respond to the current labour shortages in Japanese building industry.

Krom [1992] discusses how project models can be used to control construction robots on site. He argues that, in addition to hardware shortcomings of the heavy, big and inflexible robots, also control of the robots is not fast and easy enough for the custom-made products in the building industry. To be able to derive robot instructions, information is needed about the building, the construction schedule and the states of the building during construction. Also knowledge on capabilities of the robot is needed. A project model including the design and the construction schedule can be useful in this matter.
Design-and-build projects

The building industry’s market asks for so-called design-and-construct projects, especially for international projects. In these projects one company designs and builds the product. The Dutch technology policy advisory board for the building industry [ARTB 1993] sees a growing demand for design-and-build projects. Ballast Nedam Review [1993] names the following major benefits of design-and-build: “certainty of price; certainty of programme; certainty of quality; total commitment by the contractor; practical buildability; very direct and clear lines of communication; and the harnessing of all the designer’s, main contractor’s and sub-contractors’ innovation and energy to solving the specific needs of our projects.”

For a company that offers design-and-build projects, integrating the design and construction processes will improve the building process and probably the product-cost-ratio. This integration might be supported with improved communication of project information between the design and construction departments of that company. When a company has several of these projects, it can become economically feasible to formalise constructability knowledge and to enable on-line use of this knowledge during design. CA DfC can offer means to improve communication and to formalise and apply constructability knowledge.

Integration with other life cycle stages

The idea of integrating design and construction responsibility can be extended to other product life cycle stages, such as maintenance and facility management.

For example, the Finnish contractor Haka plans to offer their clients long-time service contracts [Laitinen 1992]. Normally, contractors guarantee maintenance for a period of only one year. Haka expects to be able to rationalise maintenance because they can take maintenance requirements into account during design (Design for Maintenance). Haka plans to support the complete life cycle of the building with product models.

Project models used for CA DfC might be useful for facility management and maintenance. The description of the building, which is part of the project model, can also be used in these stages. In addition, requirements of these stages might be taken into account during design in the same way construction requirements are accounted for.
6.2  Building Participants

The previous section showed some developments that might be stimulated with project modelling and CA DfC. This section discusses how the various participants of a building project could implement CA DfC. It argues that participants should be involved actively in the implementation, and that communication on the level of project models is needed to realise integration of applications that use different types of information.

Learning-by-doing

Integration of design and construction might result in a new way of working, communicating, and co-operating. When design and construction are integrated, project participants will be partners who share the same goals, knowledge and information. The building process will be a continuous information exchange process, in which all partners are involved at all times.

Such changes can only be realised when proven to be profitable and when realised from within practice. The Dutch Foundation for Building Research calls “learning-by-doing” the most realistic and feasible way to establish changes in the building industry [Haselhoff and Rijlaarsdam 1988]. This means that building professionals should be actively involved in implementing CA DfC and project modelling. A successful demonstration project might be a good start. When CA DfC is demonstrated successfully, management can see what can be earned on the short term, and what can be reached on the long term. The people that do the work will see that project modelling does not threaten their jobs: it takes away the tedious, repetitious parts of work, and puts more emphasis on the creative part. It will probably make their jobs more interesting.

Building participants should start using project modelling and CA DfC where there is most chance of success, i.e.:

- processes that are supported by computer applications,
- processes that require much, but relatively simple, communication, and
- parts of the organisation in which much time is spent at input and output of information.

To increase chances for acceptance of the changes, project modelling should support general business goals. Examples of such goals are: improvement of working conditions, Total Quality Management, enlargement of the market, shorter lead times, and meeting labour shortage. For example, the Finnish contractor Haka hopes to enlarge its market by using product modelling, and the Japanese contractor Shimizu hopes to meet labour shortages with far-reaching construction automation.
A migration route to CA DfC

Before building participants can decide to pursue CA DfC using the integration concept developed in the previous chapters, they have to know what steps to make. As discussed above, the participants should be involved actively in the implementation and a successful demonstration of CA DfC is important to increase acceptance. This subsection describes the choice of the semantic level for communication needed for CA DfC and outlines a possible migration route.

Communication on different semantic levels

As discussed in section 1.1 CA DfC implies the integration of information and knowledge used by design, planning and cost estimation applications. The discussion of the history of product modelling in section 3.1 showed that project information can be exchanged electronically on three semantic levels:

- the presentation level,
- the representation level, and
- the definition level.

Examples of product information on these levels are respectively: 2D drawings, geometrical models, and product or project models. 2D drawings are generated with Computer Aided Draughting (CADr) systems and can be exchanged to other CADr systems electronically, e.g., by using DXF of AutoCAD [Autodesk 1990]. The geometric information in the electronic drawings can be extended with attribute information, also called annotations. In the draft STEP standard for associative draughting [ISO/TC184 1993c] annotation is defined as "text and/or symbol used for the purpose of communicating product data and/or drawing interpretation information." 3D geometrical models can be generated with a geometry modeller. These models can be exchanged to other geometry modellers, e.g., by using the STEP resource model for geometry [ISO/TC184 1993f]. Project models integrate project information generated by various applications, and can be exchanged, e.g., by using the project modelling concept suggested in chapter 4. When only the product information of project models has to be exchanged, the STEP product modelling approach, described in section 3.1, can be used.

Information in 2D drawings has a low semantic level: humans form an image of the product by interpreting its shape presentations, such as 2D side views, top views, and sections. With this image in mind humans are able to derive information needed in other applications, e.g., for scheduling and cost-estimation applications. The semantic level of 3D geometric models is higher: the model already contains a 3D representation of the product. Geometry related information, such as the volume of a component, can be derived automatically from this 3D image. Information
not directly related to geometry, such as cost or time aspects, has to be derived by humans. The semantic level of project models is again higher: a project model contains the definition of products, activities and resources. As argued in section 3.1, representation of the product in a 3D geometric model and presentation in 2D drawings can be derived automatically from the definition.

Building participants have to determine on what semantic level they want their computer applications to communicate inside their company and with other companies.

As discussed in section 3.1, information of a lower semantic level can be derived from information of a higher semantic level. However, information of a high semantic level can only be derived from information of a low semantic level with human interpretation.

When participants choose to communicate only between similar applications on the same semantic level they can use the first two semantic levels. For example, when they only want to communicate between CADr applications, they can use electronic 2D drawings, if necessary with annotations. However, when they want to communicate between applications that use different types of information, they should use the third semantic level: the level of product or project models. When they also want to exchange knowledge rules, they also need the semantic level of project models to define the rules.

The integration concept for CA DfC as developed in chapter 4 implies communication between applications that use different types of information. Therefore, it requires information exchange at the level of project models.

A migration route to project modelling and CA DfC

Information exchange at the level of project models needed for CA DfC cannot be implemented at once: the gap between practice and research is large. Moreover, the integration concept of CA DfC and the supporting tools for project modelling are not yet elaborated to such level of detail that they already can be applied in practice (see also section 3.2). In addition, the building participants first want to see what CA DfC and project modelling have to offer. They also need to be actively involved in the development of the integration concept. Therefore, a step-by-step migration route is recommended starting with the current way of exchanging project information and resulting in using project models.

An evolutionary migration route can be realised by enlarging the scale of information exchange step-by-step. In that case, the migration route can be divided in three stages:
• internal integration,
• external exchange in a building team, and
• external integration.

The migration route starts with internal integrating existing applications using project models at the scale of one company. The second and third stage extend the scale of integration first to exchanging information between a limited group of project participants, and finally to integrating with any possible participants in the industry. The consecutive stages naturally evolve from each other, because systems used in early stages can be used in later stages. In the early stages the participants can get familiar with project modelling and the supporting tools. They can also elaborate the integration concept using their modelling experiences. The resulting optimised concept can be used in later stages.

The following describes a possible blue-print for the three stages of this migration route. Participants that want to apply the integration concept for CA DfC have to pass through these three stages. Because of the importance of successful demonstrations mentioned above, project participants are indicated that are likely to have a large chance of success.

**Stage 1: Internal integration**

Implementing project modelling starts at the scale of internal integration of existing applications. This stage aims at communication with project models in a company. These project models are structured according to a project type model (PtM) that is developed by the company itself.

As discussed before, the first integration project in the first migration stage is mainly a demonstration project and has a missionary function. To enlarge the chance of success, the first problem to tackle should be the internal exchange of information of a simple product. Preferably, the product identification names are already classified and coded. Later on, this forward exchange of product models can be extended to other types of information (e.g., activity and resource information), to backward exchange of information, and to exchange of knowledge.

Actions to take during this stage are:

• develop a PtM based on the information requirements of existing applications, using available modelling tools,
• structure a database according to the PtM,
• implement interfaces between applications and the database (or, adapt applications that work directly on the database),
• advance step-by-step from forward information exchange to backward information exchange and knowledge exchange.

At the end of the first stage, a project participant will use project models for internal communication of information and knowledge inside the
company. Communication with other participants might still require traditional media such as drawings on paper or in electronic format. In that case, geometrical models and 2D drawings should be derivable from the project models.

With currently available product or project modelling tools, companies can start the first stage right now with small demonstration projects. An example of such a modelling tool for product information is Design++ [Design Power 1991] discussed in section 3.2. It enables implementation of a product type model in a knowledge based modelling environment with a direct link to a CADr system and a database. When its database is structured according to a PtM and when it has an open (standardised) interface, such a modelling environment can fit in the project modelling approach. A project modelling system such as PMshell, described in chapter 5, might evolve to an environment to define and apply project type models.

Project participants that are product oriented, such as suppliers, are most likely to benefit from internal integration, and should thus start with using project models internally. Because this stage deals with internal integration, application of project modelling only leads to CA DfC when design and construction are in one hand. This is the fact for, among others, suppliers of precast concrete elements.

Stage 2: External exchange in a building team

After internal integration, project participants can work on external exchange in a building team. A building team is formed by the companies that participate in a project and are selected before the tender phase. The second migration stage aims at communication with project models between these participants. It will show organisational consequences of using project modelling for communication beyond company borders.

During the second stage, the focus is on DfC interaction 1 of section 2.2: forward exchange of information at discrete points in a building project. The participants in a building team develop a joint PtM based on the information needs of their internal systems. If available, this PtM should also be based on a reference model developed and standardised, e.g., by umbrella organisations of the participants. If necessary, they adapt their individual systems to the joint PtM. In the building team, they communicate with project models.

The currently used drawings (and other paperwork) are replaced by project models. These are mainly communicated with physical files, using floppy disks, modems, or electronic mail. EDI messages [UN/ECE 1990] can be used as an envelope for the physical files.

Not too much attention should yet be paid to information management. Participants should only send data that others are allowed to use.
And, every participant should check the consistency of the information offered before he incorporates it in his system.

Actions to take by the participants in a building team during this stage are:

- develop a joint PtM based on the information requirements of the participants and, if available, on a standardised reference model and standardised resource models,
- adapt each internal system to the new PtM,
- use project models for forward exchange of project information, and
- advance step-by-step to backward information exchange and knowledge exchange.

At the end of the second stage, participants in the building team will enjoy the benefits of improved communication. During the project, the participants can share information and knowledge via project models. Already in the early design phases, constructors can influence design choices. External exchange in a building team is thus a first step towards integration of design and construction beyond company borders. Outside the building team the participants communicate with the traditional media, which should be derivable from the project model.

For the second stage, areas with standardised products described with much and simple information are most likely to succeed. Examples are reinforcement bars and steel structures. Process oriented project participants such as general contractors and project managers, should stimulate this stage. They are responsible for managing the building process, and thus will benefit from better integration.

This stage can start when at least two successive participants have passed stage 1 and use project models internally. However, the first small demonstration projects can start already today. Inevitably, these demonstrations will be restricted, because today the participants only have parts of the information available electronically. An example of such a restricted demonstration is the electronic exchange of bills of quantities for reinforcement mentioned earlier [Vos and Elkuizen 1992].

**Stage 3: External integration**

When participants are able to share information beyond company borders, they are ready to realise external integration. At the end of the third migration stage, project modelling and CA DfC will be accepted industry-wide.

During the final stage, communication will grow to the ideal situation as sketched in figure 5.5 in section 5.1: computer applications can continu-
ously communicate project models via physical files, databases, or direct procedure calls. The information exchanged should be structured according to a PtM standardised, e.g., by the umbrella organisations of the participants involved. Project participants should establish direct network connections, so they can use information and knowledge any time they want. An information management system will assure protection and consistency of information.

Actions to take by building participants in this stage are:

- adapt the internal systems to standardised PtMs,
- put up an open electronic network, and
- use project modelling to communicate with partners.

At the end of the third stage, project modelling will be the rule rather than the exception. This implies that the communication infrastructure of the building industry will be adapted to electronic communication. New software has to have interfaces based on standardised PtMs. Project participants that do not support the new way of working together will fall behind.

The third stage is most likely to succeed first in areas in which participants have worked together for a long period. Preferably, the participants involved are integrated internally and have experience with project model communication beyond company borders. In that case, external integration only implies a relatively small extension of the internal system.

A possible way to realise long-term relationships is co-makership. Co-makership relations can be established on the level of sub-systems, e.g., between a general contractor and a supplier. Co-makership relations can also be established between constructors and designers. For example, a general contractor can use the same structural engineering office for several projects. Because of these long-term relationships, the companies are able to set up an infrastructure to share information and knowledge, and investments in project modelling can be spread over several projects.

In areas where relatively few clients control the demand side of the market, long-term relationships can be established for the realisation of complete structures. For the public works sector Mischgofsky [1991] sees long-term relationships between the government and general contractors as a stimulant for investments in new technologies. Interesting areas are maintenance of a civil structure over a long period of time and realisation of a series of civil structures by the same contractor.
6.3 External Organisations

External organisations might speed up or slow down the implementation of project modelling and CA DfC described in the previous section. Relevant in this matter are: the clients, the government, the umbrella organisations and the software industry.

Clients

A clear separation of client and producer tasks (and responsibilities) might facilitate integration of design and construction. This can be realised with performance contracts that separate the design process from the client and add it to the responsibility of the producer. The producer designs and constructs the building according to the client’s requirements.

In contrast to traditional tenders, which describe the building in great detail, performance contracts only prescribe requirements. In traditional tenders, the relation between client and producer is troubled, because the client performs (part of) the design where the producer is legally liable for the final product. This means that producers always have to verify (and, if deemed necessary, to redesign) the design of the client. In his study Mischgofsky [1991] argues that innovation in the sector of public works is hindered by the meddlesome attitude of the government, one of the largest clients of the building industry. The government wants to have too much influence on the realisation of civil structures.

Performance contracts clarify this troubled relation and relieve the client of investments in building design and construction management, which are not part of its core business. Clients can concentrate on what they want, not on how it is realised. Because several producers will propose a design, the clients can compare and choose the best solution.

Performance contracts could also stimulate a quality orientation in the tendering phase. In traditional tenders construction cost is the decisive argument for choosing a constructor. With performance contracts, several producers propose a building with a price-ticket. In that case, next to the price-ticket, also the quality of the proposed building is decisive. For example, with performance contracts it can become clear that a more costly, but durable building competes with a cheaper, but maintenance-sensitive, building.

The government

The government can influence the implementation of project modelling and CA DfC by changing the legal system and by funding CA DfC research. Of course, the government is also a large client of the building industry, and can thus influence the implementation of project modelling as described above.
Changing the legal system

For a smooth introduction of CA DfC it is important that the legal system evolves with the state-of-the-art technology. For example, it should not be necessary anymore to generate drawings and other paperwork as part of contract documents. Instead, databases structured according to standards should be accepted as legal "documents". An important issue to resolve is the ownership of information. Another positive stimulant for the implementation of CA DfC would be that the building authorities make building regulations electronically available, e.g., by using the concepts of de Waard [1992]. A third stimulant could be the legalisation of the tendering procedures with performance contracts as described above. An issue to resolve is how to measure the performance of building and how to formalise this in legislation.

Funding CA DfC research

Governments can stimulate the implementation of project modelling by funding CA DfC projects. Representatives from three groups of organisations should participate in these projects: building participants, software vendors, and research institutes. Research institutes provide the necessary project modelling expertise. The software companies provide their computer applications and their information technology expertise. The building participants provide the building information and knowledge. As described above, the building companies must really play an active role in applying CA DfC, because else it will not be accepted. In ESPRIT projects funded by the European Community, it is already required to have representatives of the three groups as participant.

Umbrella organisations

As section 6.2 showed, standardised modelling languages, resource model, reference models, and PTMs might help passing the three migrations stages. Umbrella organisations of the building companies can play an important role in these standardisation efforts.

ISO-STEP has recently standardised modelling languages and some general resource models. Umbrella organisations are appropriate organisations to develop conceptual models for their sector of the building industry because they are closely related to the building participants that use conceptual models. These conceptual models could be offered to ISO-STEP for international standardisation.
The software industry

The software industry can influence the implementation of CA DfC in different ways. Relevant are the development of project modelling tools and project modelling systems, and the conformation to standardised (STEP) languages and conceptual models.

As discussed in section 3.2, an important support for the building participants would be a commercially available project modelling system that integrates project modelling CASE tools. Requirements for such a system are described in section 5.1. This system should incorporate the modelling tools for the developer and programmer of conceptual models, and the information management tools for the end-users. Recently, several commercial systems have become available that fulfil (parts of) these requirements, such as XPDI, SIFRAME, and Design++. It is also important for the building companies that software companies continue to support building processes with computers. When one process in a chain of processes is not supported with computers, integration of the chain becomes more difficult. Processes that were not interesting for automation before (e.g., because they require too much input and provide too little added value), are interesting for automation when input can be computer supported. In this way, automation and integration will influence each other: more automated processes make integration possible, and an integration concept makes further automation economically interesting.

Computer applications, developed by software companies, should be open for integration. When an application only has a menu driven input interface, it is not open for integration. Such an application can not read physical files or databases, and can thus not be integrated. Preferably, applications will use the STEP modelling languages.

The computer applications should also support the semantic level of project models. Otherwise, human interpretation is required to communicate with project models. Preferably, the applications will use standardised reference models or standardised PIMs.

Building participants and institutional clients can stimulate software companies to adapt standards by letting them actively participate in the route to CA DfC.
A Case: Precast Concrete Structures

This chapter describes a practical application of the answers on the three research questions in the previous chapters. On the basis of the recommendations of chapter 6 a demonstration project is chosen: precast concrete elements. For a special type of structure a PIM is developed based on the BPM of chapter 4. This PIM is partly implemented with aid of PMshell, as described in chapter 5. The prototype implementation is used for a test case.

7.1 Organisational Implementation: Where to start?

The previous three chapters aimed at answering the three research questions of section 1.2. As part of the research approach of these answers are validated in practice in this chapter. This section discusses the chosen area of industry, companies, and type of product for a demonstration project.

Area of industry: precast concrete structures

The recommendations on the organisational implementation of CA DfC, as described in chapter 6, are used to determine the area of industry for validation.

It is important to start in an area that has a large chance of success. Acceptance of CA DfC largely depends on the results of the first demonstration projects. Section 6.2 argued that project models should be used first for internal integration by participants that are product oriented such as suppliers. This is the first stage of the migration route. Internal integration with project models only leads to CA DfC when design and construction are in one hand.

An interesting area is precast concrete structures: precast concrete suppliers are product oriented and they design and fabricate the precast elements. Often, they also assemble the elements on site. Another advantage is that this area of the building industry has a relatively large penetration of computers [Toepoel 1992]. Production is industrialised. The
products are relatively simple and standardised per company. And finally, concrete structures form a relevant part of many buildings.

The companies: Spanbeton and Ballast Nedam

The industrial partners for this demonstration project are Spanbeton and Ballast Nedam. Spanbeton is a Dutch precast concrete element supplier. Ballast Nedam is an international general contractor that realises, among others, buildings.

Spanbeton BV\(^1\) produces precast concrete elements for buildings and public works. Spanbeton mainly operates on the Dutch market. The company designs, fabricates and assembles standard and custom-made elements, among others, for structural systems of buildings. Spanbeton is part of the Partek group. Partek is a Finnish company with subsidiaries in several European countries. The companies, all involved in prefabrication of concrete elements, regularly share information and knowledge.

Ballast Nedam BV is one of the largest general contractors in the Netherlands. They are involved in practically all sectors of the building industry, among which the realisation of office buildings. Ballast Nedam Engineering BV designs and engineers office building, that are built by other subsidiary companies.

The mutual interest of the two companies is the construction of office buildings, with special emphasis on the structural system.

Automation of information flows at Spanbeton

Automation of information flows at Spanbeton is split in two areas: automation of administrative information and automation of technical information. The first area is clearly ahead of the second. The two areas are not integrated. The CAD information plan of Spanbeton [Morrel and Smies 1991] discusses the current state of automation in more detail.

Most administrative processes are supported with integrated computers applications. The administrative processes are cost estimation, scheduling, financial administration and stock administration. The computer applications used are integrated in a custom-made system, called CUBIC, based on a central database. The database is only accessible via the integrated applications or a menu-driven user interface.

The technical processes are only partly supported with computer applications. The technical processes are preliminary design, structural engineering and detailed design. Preliminary design is completely manual using tables and graphics. Structural engineering is partly automated: di-

\(^1\) The case study started at the structural systems department of Schokbeton BV. In the course of the case study, Schokbeton BV reorganised and split up. The structural systems department became independent in Spanbeton BV. There, the case study continued.
mensioning is still manual, but strength analysis is computer aided. Most applications used are home-made, with manually edited batch files as input, and output on paper. Draughting in detailed design is supported with CAD applications. Communication between the three technical processes is still paper-based. There are plans to integrate the technical processes with aid of a CAD system and a central database.

The administrative and technical information flows are not integrated. Quantity lists, resulting from the design phases, have to be imported manually in the CUBIC system. Quantity lists from preliminary design are used for tendering and global scheduling. Quantity lists from detailed design are used for detailed scheduling, and financial and stock administration.

The product: the EKON system

One of the load bearing structural systems produced by Spanbeton is EKON [Spanbeton B.V. 1991]. The EKON system is developed for (office) buildings with maximum floor spans of 16 meter and beam spans of 9.6 meter. The system consists of a skeleton of columns and beams (see figure 7.1). The beams and columns are connected with specially designed connection elements. The floors consist of hollow core slabs. The elements are structurally integrated: they highly co-operate in bearing the loads. Bennenk and van Boom [Bennenk and Boom 1992b; Bennenk and Boom 1992a] describe the EKON system in full detail.

Figure 7.1 the EKON system, with standardised columns, beams, hollow core slabs and connections [Bennenk and Boom 1992b; Bennenk and Boom 1992a]

The EKON system is developed with the following criteria in mind:
• no corbels,
• minimal construction height,
• maximum freedom of design,
• little on site poured concrete, and
• a short assembly time.

Because its high level of standardisation and the relatively simple elements, the EKON system is particularly suitable for a project modelling demonstration project.

The shapes and materials of the elements are highly standardised. This results in systematic design, structural analysis, fabrication and assembly processes. Systematic processes are easy to support with computers. The information flows between the processes are also standardised, and thus relatively easy to support with computers. Both prefabrication in the factory and assembly on site are highly standardised. Prefabrication of the slabs is highly automated.

The demonstration project

The demonstration project has a missionary character: it has to show how CA DfC based on project modelling works for internal and external integration. The research focuses on preliminary design and assembly of the elements on site.

The demonstration project aims at indications for Spanbeton how to integrate the technical information flows internally. Also integration of administrative and technical information is taken into account. The project does not aim at software ready-for-use in practice. It shows the relevant aspects to be taken into account when supporting processes with computer applications and integrating the applications. It gives ideas on the requirements to impose on the database and the CAD system. The project should show how the six interactions of DfC (see section 2.2) can be supported with project modelling.

In addition to indications on the implementation of internal integration, the demonstration project should also give some first ideas about external integration, i.e. integration with other companies. It concerns integration with Spanbeton’s clients, normally the producers of the building, and Spanbeton’s suppliers.

The demonstration project focuses on information integration in preliminary structural design and assembly on site. Preliminary design is very systematic, and thus easy to automate. Nevertheless, already many decisions are taken that influence construction. On site the elements are assembled like a building box. Preliminary design and assembly have the same scope: the elements. The interior of the elements is not relevant.

The following sections describe the demonstration project. Section 7.2 describes a project type model (PtM) for information in EKON projects. The
PtM is a specialisation of the BPM. Section 7.3 describes an implementation of the PtM in PMshell. The chapter ends with section 7.4 describing a case project, including some organisational aspects.

7.2 Integration Concept: a Project type Model

The integration concept is based on standardised communication between computer applications. To standardise communication, information of EKON systems is formalised in a project type model. This PtM is a specialisation of the BPM (see chapter 4). The PtM contains information of:

- the designed product,
- the assembly process, and
- constructability knowledge.

Modelling the designed product shows how the concretisation mechanism (section 4.3) can be used to support information exchange during design. Modelling the assembly process shows how the product-activity mechanism (section 4.2) can be used to relate product, activity and resource. Modelling constructability knowledge shows how knowledge can be formalised in a PtM (section 3.1). Appendix B shows an EXPRESS schema of the project type model for EKON structural systems.

The designed product

The EKON system is easy to model: it has few types of elements with standardised properties. It is also obvious what the main function of the product is: load bearing. The mechanism for concretisation is used to show how product information needed and generated during preliminary design is structured.

The design process for EKON structures is mainly controlled by structural engineering arguments (of course, within the spatial requirements of the architect). The horizontal and vertical loads are guided through the structure and the foundation to the ground. By analysing the route the loads travel by, the forces on the separate elements are derived. With the forces (and the spatial requirements), the main dimensions of the elements are determined. This is done mainly during preliminary design. The forces inside the elements are analysed during detailed engineering. With the internal forces, the shape and material characteristics of internal parts (concrete body, reinforcement, prestressed steel cables, etc.) are specified.

To be able to formalise all preliminary design information on EKON structures, the concretisation mechanism of section 4.3 is applied to product information. The object types of the product part of the PtM for EKON
structures are subtypes of Required Product, Proposed Product and Interface (see figure 4.21). Load bearing requirements are formalised in Required Product, chosen shape and material properties in Proposed Product. Interface is the objectification of is-related-to relations in functional networks. This corresponds to the design element representation approach of Meyer and Fenves [1993] in which network functionality is used to map functional requirements and physical structure. They see both the elements and the connections as objects. This means that both Required Product and Interface can be fulfilled by a Proposed Product. Because load bearing is the main function of the elements, is-related-to relations are interpreted as is-supported-by relations.

The EXPRESS schema of appendix B includes the product information. Structural relations, such as “supported-by”, “supports” and “connects”, are derived from functional networks. Geometrical information such as place and orientation is already modelled in the Proposed Product and thus inherited. Material is modelled as a string containing the concrete or steel quality code.

Preliminary design of an EKON system starts at the level of the complete building and ends with the elements. The following describes the EKON elements and connections, and decomposition of an EKON system.

The EKON elements and connections

The Proposed Products at the lowest decomposition level in preliminary design are the EKON elements and connections. The elements are columns, beams and floor slabs. Common properties of the elements, such as material specification, are generalised in Precast Element.

EKON column

EKON columns are very simple. They have a uniform cross-section over their entire length. Columns are supported by the foundation or by other columns. Columns support the beams. The columns are made of reinforced concrete. To minimise cracking during production and transport the reinforcement can be prestressed.

The sides of the rectangular surface can be 240, 290, 390, 490, 590, 690 or 790 mm. The cross-section dimensions are determined following a standard method. The columns can be 1 to 5 storeys high, with a maximum of 20 m. One-storey columns can be interconnected in length. Interconnecting columns on site is time consuming. Longer columns require less assembly time. Columns have a lifting eye and connection points for supports during assembly.
EKON beam

The prestressed EKON beams are also highly standardised. They are T-(or, actually L-) and L-shaped. T-beams are used inside a floor field, L-beams on the edges. A beam is simply supported by columns on both ends. A beam supports floor slabs. The slabs are stacked on the flanges.

Four parameters specify the cross-section (see figure 7.2). Only the flange height and width do not depend on the dimensions of other elements. The flange height can be 100 or 150 mm. The flange width can be 150 or 200 mm. The width and height of the beam directly depend on respectively the width of the supporting column and the height of the supported floor slabs. The beams are profiled to ensure good co-operation with the floor slabs. To ensure smooth removal of the formwork and less optical damage, the edges are bevelled.

![Figure 7.2 Cross-section of EKON T- and L-beams](image)

The flange dimensions are determined using graphics developed by Spanbeton. In these graphics the variables are the imposed load during assembly and during operation, the span of the beams, the height of the supported hollow core slabs, and the distance between the centres of the beams.

VARIAX hollow core slabs

The floor consists of VARIAX hollow core slabs. Reinforcement in these slabs is prestressed. Production of the slabs is highly automated with a extrusion process developed in Finland. VARIAX floor slabs minimise material use. The extrusion process ensures high quality concrete. The prestressed reinforcement and the hollow channels ensure full use of the concrete. The slabs are supported by beams.

There are four possible heights: 200, 265, 320 and 400 mm. Slabs have a width of 1200 mm and four, five or six channels depending on the height (see figure 7.3). The width of the slabs can be adapted by cutting the slabs in length, through the centre of a channel. The maximum length of the slabs is 16 m.
The height of a VARIAX slab is determined with graphics developed by Spanbeton. In these graphics the variables are the span and the imposed load. The span is the distance between the centres of the supporting beams. The imposed load is uniformly distributed and includes dead load and life load. Dead load does not include self weight. Life load differs per type of building and is formalised in regulations.

**EKON connections**

The EKON system also standardises the element connections. Relevant are connections between (1) column and foundation, (2) in line columns, (3) beam and column, (4) slab and beam, and (5) adjacent slabs. The connections are realised with on-site poured concrete and often with reinforcement bars. Common properties of the connections, such as the material specifications, are generalised in Structural Connection. Structural connections fulfil the requirements of the interfaces on the lowest decomposition level.

The column-foundation connection is formed by reinforcement bars from the foundation that fit in gains of the column (see figure 7.4). The gains are filled with concrete to form a fixed-ended connection. The number of gains possible depends on the dimensions of the column. The fixed-ended columns guide horizontal loads to the foundation, and thus contribute to the stability of the structure.
Figure 7.4: Section of a column-foundation connection with reinforcement bars in gains; on site the gains are filled with concrete.

The connection between in line columns is formed by reinforcement bars from the bottom column (see figure 7.5). Because this type of connection is time consuming, it is preferable not to use it.

Figure 7.5: Section of a column-column connection; beams stacked on top of the bottom column and connected via reinforcement bars; the top column is stacked on top of the beams and connected with the bottom column via reinforcement bars; on site the gains and the spaces between the columns and the beams are filled with concrete.

Less time consuming is the BSF\textsuperscript{1} beam-column connection. The BSF connection is developed in Norway, and has been extensively tested by

\textsuperscript{1} BSF stands for Bjelke-Soyle Forbindelse, which is Norwegian for beam-column connection.
Spanbeton. A steel BSF connection replaces a concrete console. It consists of a column house, a beam house and a movable plate (see figure 7.6). On site the plate is shoved from the beam house in the column house. For extra stability and protection of the steel parts, the houses are filled with concrete. To guide the forces from the houses to the rest of the elements, extra reinforcement can be placed. The connection can be schematised as a hinge. There are two versions: BSF150 and BSF200. The numbers denote the thickness of the steel plate. Which version is used, depends on the reaction force and the available space in the column and the beam.

![Diagram of BSF connection]

**figure 7.6** section of a steel BSF connection between beam and column; the connection consists of a column house, a beam house and a movable plate; on site the houses and the space between the beam and the column are filled with concrete

The connection between the hollow core slabs and the beam is realised with extra reinforcement bars (see figure 7.7). The slabs lay on the flanges of the beams. The seam between the beam and the slabs is filled with concrete. In the beam-slab seam reinforcement bars are placed. To prevent the concrete to fill up the hollow slabs, plugs are placed in the channels. The beams have a gain on the level of the seam between two slabs. Through this gain a reinforcement bar is put. The strongly profiled beam-slab seam and the reinforcement bars ensure full structural co-operation of the beam and the slabs.
The slab-slab connection is also formed by a strongly profiled seam (see figures 7.7 and 7.8). This seam is strong enough to ensure spreading of imposed loads over several slabs.
Decomposition of an EKON bearing structure

Decomposition of the structural system follows the concretisation of products during preliminary design. Preliminary design starts at the level of the complete building and ends with specification of the EKON elements. In several decomposition steps the properties of the elements are specified. In his master’s thesis van den Hoek [1991] describes decomposition of structural systems of office buildings in more detail.

Preliminary design starts with the spatial requirements of the architect. The architect has defined a grid and a maximum free space for the structural elements. The structural engineer decomposes the building structure in a foundation, stability facilities, load bearing facilities and dilatation facilities. A stability facility guides the horizontal loads to the foundation, a load bearing facility the vertical loads. Elements can be part of both facilities. Dilatation facilities make the load bearing facilities independent. This research focuses on load bearing facilities.

A precast concrete element supplier chooses to fulfil the load bearing facility by a precast concrete structure. This structure can be divided in structural areas. For each structural area a type of structural system can be chosen. Van den Hoek decomposed an office building in regular and irregular structural areas (see figure 7.9). Regular areas have much repetition, and are thus very suitable to be fulfilled by a highly standardised system such as the EKON system.

---

![Floor-plan of example load bearing structure of a building decomposing in regular and irregular areas [Hoek 1991]](image)

---

For decomposition of the EKON structural system a top-down geometric modelling approach is chosen. This approach is also described by Choi and Kim [1993]. They suggest to decompose a building in four successive steps:

1. define all surface primitives,
2. determine intersection lines,
3. define points and edges, and
4. define solid models for the edges.
The EKON system decomposes in surface primitives for floor fields and column rows (step 1 of Choi and Kim). Figure 7.10 shows the grid and the surface primitives, both schematic and in NIAM (see appendix B for a more detail EXPRESS version of the PM). The surface primitives are determined with aid of the grid, as defined by the architect. The column rows support the floor fields. The floor fields have imposed loads.

![Diagram of EKON system decomposition](image)

**Figure 7.10** Decomposition of an EKON system in column rows and floor fields

A column row is fulfilled by an EKON column system. A column system decomposes in vertical line elements (one storey high or longer), line element connections and foundation connections. Figure 7.11 shows the line elements and the connections, both schematic and in NIAM. The lines and points are determined with aid of the grid and the intersections of the column rows and the floor fields (step 2 and 3 of Choi and Kim). Both the line elements and the connections are seen as objects.

A floor field is fulfilled by an EKON floor system. An EKON floor system decomposes in horizontal surface elements, horizontal line elements, surface element connections, surface element-line element connections and horizontal line element-vertical line element connections (see figure 7.12). The surfaces, lines and points are determined with aid of the grid and the intersections of the column rows and the floor fields.

---

1 For readability purposes, the supertypes of the object types in figure 7.10, 7.11, and 7.12 are indicated with abbreviations: RP stands for Required Product, PP for Proposed Product and IF for Interface.
**Figure 7.11**  Decomposition of an EKON column system in vertical line elements and connections

**Figure 7.12**  Decomposition of an EKON floor system in horizontal line elements, surface elements and connections
The required products that are part of the column rows and the floor fields are fulfilled by the proposed EKON elements. The objectified interfaces that relate required products of the column rows and the floor fields are fulfilled by the proposed EKON connections. This corresponds to the fourth step of Choi and Kim: the abstract surfaces, lines and points get shape and material. With the loads on the required products the properties of the proposed products are determined. The main sequence in this process is first to specify the elements and then the connections. First, elements are specified that are supported by other elements.

The assembly process

For the assembly process the states of the product, the activities and the resources are modelled in the PtM. The assembly process is modelled to show how these types of information can related using the product-activity model of section 4.2. Only Proposed Products on the lowest decomposition level are regarded.

Assembly states of the product

Assembly of an EKON structure on the building site means putting the elements together like a building box and realising the connections. The elements are prefabricated at the factory. Each element goes through a number of successive states during assembly on site. The connections are realised completely on site. The states of the elements and realisation of the connections are related with "succeeds" relations.

Each element goes through five successive assembly states: (1) ready-for-assembly, (2) on site, (3) on-position, (4) fixed and (5) realised. Ready-for-assembly means fabricated in the factory. On site means transported to the building site. On-position means lifted to its final position. Fixed means fixed to its supporting elements or to special assembly supports. Realised means that the connections to supporting elements are realised and that the assembly supports are removed.

The states and realisation of the connections are related. For each element its states have to be passed in the given sequence. In addition, an element can only be lifted to its final position and be fixed if its supporting elements are fixed. Connections can only be realised after the connected elements are fixed. An element can only be finished after its connections with supporting elements are realised.
Assembly activities

The activities to be executed during the assembly process follow directly from the assembly states of the product. The sequence of the activities is determined by the required sequence of the product states and logistics.

There are five types of activities to be executed: (1) transport element to the building site, (2) lift element to the right position, (3) fix element, (4) realise connection and (5) finish element. Scheduling these assembly activities starts with determining the required sequence of the product states. For EKON structures, fixing the elements is decisive, or, in other words, is on the critical path. Only when elements are fixed, other elements can be supported and connections can be realised. The other activities can be adjusted to optimise logistics of the resources.

Assembly resources

The assembly activities use resources. For assembly of the EKON elements five types of resources are used: (1) trucks, (2) cranes, (3) assembly supports, and (4) workmen.

Trucks are used to transport the elements to the building site. Cranes are used to lift the elements to their final position. Cranes are also used to hold the columns and beams to that position during fixation. Assembly supports are used to stabilise the columns and the BSF connections while their connections are realised. Stabilising the BSF connections is only necessary when the span of the supported VARIAX elements is more than 9m. All activities use workmen.

Conceptual model for assembly

Figure 7.13 shows the conceptual model containing the above described information required for scheduling assembly. This model specialises the proposed part of the kernel of the BPM as shown in figure 4.23.
figure 7.13 conceptual model of assembly states of the product, assembly activities and assembly resources; assembly resources such as workmen and materials are left out

1 For readability purposes, the supertypes of the object types in this figure are indicated with abbreviations: PP stands for Proposed Product, PS for Proposed State, PA for Proposed Activity, and PR for Proposed Resource.
Constructability knowledge

Section 3.1 argued that constructability knowledge is only relevant for combined use with project modelling per type of product. To show how constructability knowledge can be incorporated in a PtM, some rules for the EKON structural system are modelled in EXPRESS.

The basis for improvement of the efficiency of assembly is minimisation and simplification of the activities [Boothroyd and Dewhurst 1987; Andreassen, Kähler, et al. 1988]. Rules that ensure a decrease and simplification of the activities required for assembly of EKON structures can be derived from four sources:

- DfA knowledge in general,
- DfC knowledge for concrete structures,
- DfC knowledge for precast concrete structures, and
- DfC knowledge typical for EKON structures.

Andreassen, Kähler, et al. [1988] define specific rules for Design for Assembly in the mechanical industry. A selection of their rules is also valid for assembly of precast concrete structures:

- minimise the number of elements (e.g., by integrating functionality),
- minimise the number of connections (e.g., by stacking the elements in stead of connecting),
- minimise storage and waiting times,
- standardise dimensions,
- design with standard solutions,
- avoid tolerance demands,
- ensure fast identification of elements,
- ensure fast orientation of the elements,
- reduce chance on mistakes (e.g., by using symmetrical elements),
- ensure high, constant quality,
- ensure good gripping of the elements,
- use standard elements in a building box, and
- let one team of workmen do the complete assembly (to enlarge the involvement in the building process).

In his PhD thesis, Fischer [1991] defines DfC rules for in-situ concrete structures. A selection of his rules is also valid for preliminary design of precast concrete structures:

- use one layout scheme per project,
- keep layout as constant as possible,
- respect standard sizes of prefabrication forms,
- minimise openings,
- standardise openings,
- keep dimensions as constant as possible, and
- minimise the number of different concrete mixes.
Some of the general rules mentioned above can be specialised to the assembly of precast concrete structures. Some of these specialised rules can be found in FIP\(^1\) recommendations for precast structures [FIP 1986]. The specialised rules are:

- simplify the connections to be realised on site,
- use elements as large as possible,
- consider capacity of available transportation and lifting equipment,
- minimise use of assembly supports,
- do not use storage on site (place elements directly from the truck to their final position),
- provide good lifting points on the elements,
- use columns that continue over several storeys,
- ensure stability after fixing, and
- use structural concrete topping of floors as little as possible.

In addition to the above mentioned rules, there are some typical EKON rules (or recommendations) that ensure optimisation of assembly:

- use VARIAX slabs of 1200 mm as much as possible,
- make centre distance of columns a multiple of 1200 mm,
- do not use columns wider than 500 mm,
- use BSF connections also for beam-wall connections,
- do not use VARIAX elements longer than 9 m (because then assembly supports are required for the BSF connection),
- use the same height of slabs left and right of the beam,
- let large holes in the floor (e.g., for stairs) coincide with VARIAX slabs, and
- do not use interconnections of EKON columns.

In the EKON structural system many of these rules are already fulfilled. However, the designer still has freedom in the layout and the dimensions of the elements. His choices can influence the efficiency of the assembly process significantly.

As described by de Waard [1992], simple constructability knowledge can be formalised in EXPRESS. In the EXPRESS schema of the PtM in appendix B this is demonstrated. The constructability rules are incorporated in the local rules of the object types (or entities).

The layout and dimension rules are spread over several object types. Correctness of the layout is checked in EKON Structural System. Standardisation of the dimensions of the separate elements is checked in the corresponding object types. In EKON Project, the object type that gathers all objects, it is checked whether the dimensions are constant over the

---

\(^1\) FIP stands for Fédération Internationale de la Précontrainte, the International Federation for Prestressed Concrete.
complete structure. In EKON Project also the capacity of the available equipment is checked.

When using EXPRESS for formalisation of constructability knowledge, it is only possible to identify whether a rule is violated or obeyed. In case of conflicting rules, it is not possible to weight one rule against another.

Several rules mentioned above use terms as “optimise this” or “minimise that”. These rules can not be dealt with by formalising knowledge in a PtM in an EXPRESS schema. Optimisation follows from comparing several alternatives. This is not a subject to be dealt with in a PtM. It is rather the task of the specific computer applications. The PtM can only help to speed up communication between the applications, so that more alternatives can be looked at.

7.3 Computer Implementation of the PtM

This section describes the implementation of the PtM for EKON structural systems in PMshell, using the BPM implementation. The goal of this computer implementation is to show how implementations of a modelling system (i.e., PMshell), a reference model (i.e., the BPM in PMshell), and a PtM can be used in practice.

To be able to concentrate on project modelling, interfaces with existing computer applications are simulated. This means that information needed and generated by the applications is available in the system, but human intervention is required to exchange the information. Design processes, not yet supported with computers, are simulated with simplified design functionality.

The designed product

Implementation of the product part of the PtM for EKON structural systems starts with the information structure of the conceptual model of an EKON project (see the EXPRESS schema of appendix B). To this information structure functionality is added to manipulate the information.

The information structure of EKON structural systems is implemented in four steps:

- define object types as subtypes of the BPM object types,
- use BPM meta-functionality to define decomposition and network relations,
- redefine shape, and
- define product-specific attributes.
The object types in the conceptual model are implemented as subtypes of the object types of the Product domain, as described in section 5.3, figure 5.16. The Product domain is an implementation of the concretisation mechanism of the BPM applied to products, as described in section 4.3. This means that the object types are subtype of Required Product, Proposed Product and Structural Connection.

Decomposition of the EKON column and floor system (figures 7.11 and 7.12) shows that the type of an interface is completely determined by the type of the connected required products. Therefore, these (functional) interfaces are not defined as new object types. The structural connections that fulfil the functional interfaces can only fulfil interfaces that connect the right required products. For example, a BSF connection can only fulfil an interface that connects a horizontal line element with a vertical line element.

Decomposition and network relations are implemented using the meta-functionality of the BPM.

Using this meta-functionality enables the programming user to redefine decomposition structures and functional networks easily. In addition, it ensures that special BPM functionality, such as deriving compound shapes of proposed products, is available for end-users. It also ensures that end-users only can add decomposition relations that conform to the redefined decomposition structure.

The decomposition relations are defined bottom-up. For example, an EKON bearing structure can only fulfil a structural area. The corresponding relation, a structural area is fulfilled by, is not restricted. This ensures extendibility of the PiM: in future other implemented bearing structures can also fulfil the structural area.

The relations in a functional network of required products are interpreted as "is-supported-by" relations. These relations are implemented as links of a graph. The links point from supported elements to supporting elements.

Shape of the product types is defined by redefining the "shape" attribute for each Required and Proposed Product. For this the Shape domain is used.

Each EKON object type gets its representative shape as shown in the schematic figures of figures 7.10, 7.11 and 7.12. The shape object types have characteristic properties to define their geometry. For example, Rectangular has a length and a width. These characteristic properties can be used to specify the geometry of the products.

The shape of a required product can be placed (and oriented) relatively to the shape of the proposed product it is part of. The shape of a proposed product can be placed relatively to the shape of the required product it fulfills.
By using BPM meta-functionality and the Shape domain the decomposition and shape related part of the information structure is implemented easily without bothering graph and geometric functionality. The product-specific attributes are implemented using standard PMshell functionality. This mainly concerns attributes like the load of a floor field, and the characteristic properties of the EKON elements and connections.

To manipulate the information structure EKON specific functionality is implemented. General functionality such as decomposing a particular product conforming the PtM, is already inherited from the BPM. In the EKON domain functionality is implemented:

- to simulate input interfaces,
- to simulate applications, and
- to simulate output interfaces.

Input functionality is implemented to simulate a simple design application and to specify the product-specific attributes.

With functionality inherited from the BPM, a particular EKON bearing structure with all its parts can be specified. However, this is very time-consuming: the required and proposed products have to be instantiated, added to the decomposition tree and placed one-by-one. The designer wants to define the type of all floor slabs on one floor at once. Therefore, functionality is implemented to group standard BPM functionality according to the design process as described in section 7.2.

For an EKON bearing structure the grid of the architect and the bearing direction of the floor slabs can be specified. With this grid and direction the column rows and floor fields are derived automatically.

A floor system and a column system can be decomposed in surface and line elements and their connections. All surface elements of a floor system can be fulfilled by VARIAX elements with the same height.

This kind of functionality simulates a simplified design application. Normally, a design applications would use graphics interactively. In this simulation, only the output can be viewed graphically.

To specify the values of the product-specific attributes also functionality is implemented. For example, the loads on the elements, calculated with structural analysis applications, can be put in the system.

Next to decomposing the EKON structure in its parts, also other processes are simulated. During preliminary design the dimensions of the EKON elements are determined with aid of graphics and tables from the product documentation. For VARIAX slabs and EKON beams these graphics are incorporated in procedures. The VARIAX graphics are approximated with third degree polynomials, the EKON beam graphics with linear polynomials. With these procedures the dimensions of a single element can be determined. During detailed design this analysis can be done again and more accurately.
Output interfaces are simulated by presenting the information on the particular EKON structure in such a way that the end-user can use it for input in other applications.

For graphical presentation, the structure, or parts of the structure, can be viewed with PMSGQT. To interface with detailed design, information on each beam can be presented so that input for an analysis application can be easily derived. This information can also be used as parameters for pre-defined macros in a CAD system. To interface with the administration system, a quantity list can be presented. For this the object type Project is implemented as a graph. It filters all EKON elements from the decomposition tree.

**The assembly process**

Implementation of the assembly process of EKON structures starts with the information structure of the conceptual model of an EKON project (see the EXPRESS schema of appendix B). To this information structure simple functionality is added to copy planning information from planning applications.

The object types are subtype of the Product-Activity domain, as shown in figure 5.17. The relations between the object types are implemented using standard PMshell functionality for defining attributes. The sequence of the states is checked in local rules.

**Constructability rules**

The relatively simple constructability rules of the PtM are implemented using standard PMshell functionality. This enables a qualitative first check of the constructability of a designed structure.

In the EXPRESS schema of appendix B the rules valid for EKON structures are formalised in local rules per object type. In PMshell these local rules are implemented in the assertions defined per object type (see section 5.2, figure 5.9). An assertion is a function that checks the current state of an object on one rule. It results in a message telling whether the object obeys or violates the rule. The name of the assertion corresponds to the name of the rule in the EXPRESS schema. The implementation of the assertion can be very comprehensive, using all possibilities of the Eiffel programming language. This means that the implementation body of the rules, often indicated in the EXPRESS schema with dots, are implemented in PMshell.

The user interface of PMshell gives the end-user the possibility to check one local rule of an object, all rules of an object, or all rules of all objects of a project model. A report can be printed of the violated rules. This report can be used for evaluation of the design.
7.4 A Test in Practice

The implementation of the EKON model in PMshell, described in the previous section, is used in a small test. This test should give the end-user in practice an idea how he could work with the system. In the test a simple rectangular structural system is designed. To this product model assembly process information is added. The constructability rules of the combined project are checked. As example for an interface with other applications, the interface with detailed design is shown for one of the beams.

Preliminary design of an EKON structure

Preliminary design follows the top-down geometric modelling approach described in section 7.2.

First the grid and bearing direction of the floor slabs are specified. With this information, the column rows and floor fields of the EKON system are derived. Figure 7.14 shows the graphical presentation of the decomposition of the EKON system.

The floor fields and the column rows are fulfilled by EKON floor fields and EKON column rows. These proposed products are decomposed in line elements, surface elements and connections. Figure 7.15 shows the graphical presentation of the decomposition of an EKON floor system.

Per floor field the superimposed loads are specified. With this information the dimensions of the VARIAX elements and the EKON beams are calculated using the implemented graphics. The dimensions of the columns are not calculated. Per column the cross-section is simply defined as square, with one side equal to the width of the EKON beams it supports.

Integration with assembly planning process

Integration of the modelling system with the assembly planning process is two-fold: (1) the quantity lists of EKON elements form the input for the planning system, and (2) the output of the planning system forms the input for the modelling system.

Quantity lists of the used EKON elements are derived using functionality of the object type Project. These lists can be put in the planning system CUBIC manually. This corresponds to the current way of working.

Normally, the CUBIC system does not determine the assembly activities to the detail of the elements. For this test case the output of the planning system is simulated. The assembly states and activities of two columns and one supported EKON beam are put in the system. The local rules of the three objects are checked to determine whether the right sequence is used.
figure 7.14  graphical presentation of the first level decomposition of an EKON system
Constructability rules

The implemented constructability rules are checked. This results in a violation report. Figure 7.16 shows the output window of PMshell containing this violation report for an EKON bearing structure.

As shown, per rule only the verdict “not violated” or “violated” is determined. It is not possible to weight the importance of the rule. However, the constructor wants that some rules are never violated. Other rules are preferably not violated. For example, the width of a VARIAX slab may never be more than 1200 mm, because the limitations of the ma-
chinery in the factory. The width of a VARIAX slab is preferably not less than 1200 mm, because cutting a slab is costly. Therefore, the report with violated rules always has to be interpreted by humans. The violation report only gives a first impression on the of the designed structure.

```
object "bil_ebs" of type "EKON_BEAR_STRUCT":

invariants:

   correct_grid: VIOLATED
   -- Is the grid well defined?
   grid_conforms_variax: VIOLATED
   -- Does the grid in combination with the bearing_direction conforms
   -- to the limitations of VARIAX elements?
   available_crane_ok: not violated
   -- Is the weight of the largest element in the structure smaller
   -- than the capacity of the available crane?
   available_truck_ok: VIOLATED
   -- Are the dimensions of the largest element in the structure smaller
   -- than the capacity of the available truck?
   slabs_height_ok: not violated
   -- Are the heights of the VARIAX slabs on the left and the right side
   -- of each EKON beam in the structure the same?
   no_columns_interconnected: not violated
   -- Are there no EKON columns that are interconnected?
   lower_bound_obeyed: not violated
   -- Is the lower_bound obeyed for all link_types and node_types?
   upper_bound_obeyed: not violated
   -- Is the upper_bound obeyed for all link_types and node_types?
```

Figure 7.16 output window of PMshell showing the violation report of an EKON bearing structure

**Integration with detailed design**

In detailed design the exact dimensions and the interior parts of the elements are determined. This engineering is supported with computer applications. Input for these applications comes from preliminary design. In the implementation this is simulated for EKON beams. Per beam a report can be generated containing the information from the project model needed by the analysis application. This basic information has to be extended with more detailed information such as material specifications before the application can run.
8 Evaluation

The preceding chapters described a strategy for CA DfC and its application in a case study. This chapter summarises and evaluates the findings.

8.1 Integration Concept

Chapter 4 discussed an integration concept to support the realisation of CA DfC in the building industry. The concept standardises the formalisation and exchange of information and knowledge in a building project. Section 7.2 described an application of this concept for a specific type of building projects. Both, a global summary of the results, and their evaluation, are discussed below.

The integration concept proposed, is based on improved communication achieved with standardised (STEP) project modelling.

For CA DfC, the most important part of the concept is a reference model for the building industry: the Building Project Model (BPM). The BPM aims at formalising the information that is exchanged between computer applications used in design and construction. Because computers are not able to interpret information the way humans can, relations that often only exist implicitly are made explicit. For this, the BPM uses two abstraction mechanisms. The concretisation mechanism describes and relates the evolutionary states of project objects in problem solving processes such as design and construction scheduling. The product-activity mechanism relates product information to activity and resource information.

In the case study, the BPM is specialised to preliminary design and assembly on site of a type of precast concrete structural system, the EKON system. Part of the constructability knowledge about precast concrete elements is formalised and applied during preliminary design using the STEP modelling language EXPRESS.

With the experiences in developing, implementing, and applying the BPM, the proposed integration concept for CA DfC is evaluated on the following issues:
• the choice for project modelling,
• the framework,
• the Building Project Model (BPM), and
• knowledge exchange.

Application of the developed integration concept in the test case showed that project models can be used to communicate different types of information between computer applications. The project model contained information needed for preliminary design of a precast structure and for assembly of the precast elements on site. It contained product, activity, and resource information. Because this information can be stored in an open database or a physical file, design, scheduling, and cost estimating applications can access and use the information in the project model. Project models can thus be used to improve communication in a building project.

The CIM framework and the chosen interpretation of its dimensions proved to be suitable for structuring the modelling efforts in this research.

The modelling languages NIAM and EXPRESS were found useful for formalising information structures of the BPM and the PtM. EXPRESS also allowed formalisation of (simple) knowledge rules in the PtM.

The three layered scope architecture of Tolman was successfully applied in this research: it enabled the development of a reference model for the support of DfC in the whole building industry. Development of this reference model was not hindered by general issues such as shape definition, neither by detailed issues of specific project types.

The case study also made clear that some aspects need extra attention. There were long lists of separately defined, but identical precast elements. This made the database redundant. A data reduction mechanism should be included. In the case study, product decomposition is completely controlled by structural arguments, and thus completely directed to structural engineers. However, other participants, such as architects or HVAC engineers, might use another decomposition of the same building. These different views cannot be related. Another problem was object decomposition. Because the modelling languages (and the object-oriented language) do not support a decomposition abstraction mechanism, it had to be modelled in the BPM.

The BPM aims at defining the relations between objects explicitly. The two abstraction mechanisms that ensure this are used separately in the case study.

With the concretisation mechanism it was possible to support preliminary design of precast concrete structures. The divide-and-conquer principle supported a top-down design approach. The separation of problem and solution enabled relating parametrically defined components (the precast elements and their connections) to this decomposition structure. At
the level of precast elements the product-activity mechanism is used to check the sequence of activities and states of the product during assembly.

The case study showed that the BPM with its explicitly defined relations cannot be used for all applications. Because most applications support processes in which humans relate the different types of information, not all information is electronically available. For example, the scheduling application used in the case study did not deal with information about the states of a product during construction. Therefore, state information could not be communicated and was included in the project model. Consequently, product and activity were not related anymore. A similar problem occurred with the STEP resource model for topology and geometry [ISO/TC184 1993f]. This resource model is so general, that it was not possible to refer to it for the simple shapes used in the case study.

Knowledge exchange from construction to design has proven to be more than just defining simple rules. With EXPRESS, relatively simple constructability rules could be formalised and exchanged. However, applying these rules during design did not provide a complete picture of the consequences of the design choices for construction. Applying a rule only provided a qualitative evaluation of one aspect separate from the other aspects. Moreover, conflicting rules could not be compared.

8.2 Computer Implementation

Chapter 5 proposed a project modelling system that supports the technical implementation of the integration concept for CA DfC. Section 7.3 described an application of this project modelling system in a case study. The following briefly summarises and evaluates the findings.

Related research showed that implementation of project modelling asks for support with CASE tools, preferably integrated in one system. The project modelling system should support the STEP languages EXPRESS and the STEP physical file format. For presentation and the early development phases, also a graphical conceptual modelling language such as NIAM should be supported. The project modelling system should also support a mechanism to relate conceptual models with different scopes. Finally, the project modelling system should support at least three groups of possible users: the developers of conceptual models, the programmers, and the end-users in practice. In close co-operation with TNO, PMshell has been developed to fulfil most of these requirements. PMshell is a prototype version of a project modelling shell implemented in an object-oriented language.

Representative parts of the reference model BPM have been implemented in PMshell. For the product data, the concretisation mechanism is
implemented, including decomposition following the design process. For activities and resources, the relation with the product and its states is implemented. The implementation of the BPM in PMshell is used to implement a project type model for a specific type of precast concrete structures.

The technical implementation of the integration concept for CA DfC is evaluated on the following issues:

- PMshell,
- implementation of the Building Project Model,
- implementation of the project type model, and
- implementation of constructability knowledge.

PMshell proved to be the right kind of tool for the implementation of the general resource models, the BPM, and the project type model. Especially PMshell's support has been very helpful for the concurrent development, implementation and application of different conceptual models (or domains), in which also the relations between the different domains can be implemented gradually.

The STEP modelling languages and the mechanism available to relate conceptual models with different scopes, have good corresponding notions in database technology and object-oriented programming. These mechanisms were therefore relatively easy to implement and apply. However, decomposition of objects has no corresponding notions in the implementation paradigms. Therefore, object decomposition had to be implemented, including functionality to define decomposition relations in conceptual models and functionality to apply decomposition in project models.

Working with PMshell showed that bridging the gap between specification (by the developers) and implementation (by the programmers) is very difficult. Bridging that gap means more than just passing information smoothly between the two. It also requires another way of thinking: conceptual models developed for implementation have to be more concrete than the developers want them to be (or are able to define). This means that implementation considerations restricts the conceptual modelling freedom. Meta-functionality can be used to overcome (parts of) this problem. With meta-functionality developers can make modelling decisions on a conceptual level, which are then translated automatically into implementation details by the system. For example, meta-functionality proved to be useful for the implementation of object decomposition. With the possibility to define meta-functionality, the system has the potential to evolve from a generally applicable modelling tool, to a tool that supports the development of specific project type models.
Experience with the prototype version of PMshell also showed that the current implementation is not yet ready for commercial use\(^1\). For commercial use the robustness, performance, and portability are too low, and the hardware requirements are too high. PMshell lacks a graphical interactive interface for end-users that is adaptable to specific needs. In addition, the user interfaces are not clear and user-friendly enough. Nevertheless, it showed that it is technically possible to bring conceptual modelling, object-oriented programming, and communication via project models together into one system.

Implementation of the BPM proved to be a major effort. The prototype implementation demonstrated the value of modularisation of implementations of conceptual models. Modularity was necessary to divide the large effort into conquerable parts and over several people. Due to the immature state of the implementations of the general resource models, still much attention had to be paid to graph and shape functionality. Therefore, only some representative parts of the BPM could be implemented. This emphasises the need for modules (or, domains as they are called in PMshell) that are easily extendible, without interference with other modules.

Implementation of a project type model in the case study showed that it is possible to implement a modelling system that integrates building information in a project model. However, using that information for the integration of existing applications was not easy. Not all processes involved were supported with computer applications that use and generate information on the semantic level of project models. Other systems were not open enough to access their information easily. Therefore, the implementation of the project type model not only had to store and manage project information, it also had to generate information. The implementation had to provide means to input and manipulate information like a design system. Because developing a design system was not the goal of this research, the resulting system became very simple (and limited). This made an extensive demonstration of information exchange via project models impossible.

Implementation of constructability knowledge in rules in the project type model demonstrated that knowledge rules can be defined and applied. The object-oriented programming language enables implementation of more complex rules than can be formalised in EXPRESS. However, the shortcoming of only qualitative validation per rule is the same.

\(^1\) At the moment, TNO is developing a new version of PMshell that will overcome many of the problems of the prototype version.
8.3 Organisational Implementation

Chapter 6 proposed a migration route for the organisational implementation of CA DfC and project modelling. In chapter 7 these ideas were used to choose the subject for the case study. The following summarises and evaluates the findings.

In current practice, organisational developments are observed that might be supported with CA DfC. If participants want to adopt CA DfC, they have to use the semantic level of project models for communication between their applications from the beginning. Additional translators are needed that generate information at a lower semantic level from the project model for communication with participants that do not support the level of project models. For organisational implementation it is important that the building participants are actively involved and that the scale of communication grows step-by-step. External organisations such as umbrella organisations and the software industry might speed up this implementation of CA DfC. The umbrella organisations could work on the standardisation of conceptual models. The software industry could provide the building industry with modelling tools and computer applications that supports the semantic level of project models.

Based on experiences in developing, implementing, and applying the concept and computer support for CA DfC, some of the suggestions for the organisational implementation of CA DfC can be evaluated.

The first step in the first stage proposed in the migration route seems to be fruitful. Starting the implementation of project modelling in a demonstration project with a completely standardised product is already very difficult. The same holds for starting the introduction of CA DfC in a company that both designs and constructs part of the building. Nevertheless, it demonstrated people in practice some of the potentials of project modelling.

Developing the PiM and the constructability rules together with the end-users was not very difficult. However, the realisation of the conditions for their implementation were. The effort spent in the research on developing project modelling tools confirmed the need for commercially available project modelling tools. The effort spent on development and implementation of generally applicable conceptual models confirmed the need for standardisation of conceptual models and commercial computer applications that support these conceptual models.
9 Conclusions and Recommendations

Based on the findings of the thesis and their evaluation, this chapter draws conclusions on the value of the proposed strategy for the introduction of CA DfC in the building industry. It also formulates some recommendations for further development of the strategy.

9.1 Conclusions

The research reported in this thesis proposes a strategy for CA DfC in the building industry. In section 1.2, the strategy is looked at from three angles, which resulted in three research questions. The research proposes answers for these questions. Based on the evaluation in chapter 8 this section draws conclusions on the value of the three answers for CA DfC.

Research question 1: integration concept

The first research question is: "What are the characteristics of an integration concept that supports CA DfC?" The answer proposed in chapter 4 is to improve the integration of information and knowledge using project modelling. Project modelling is seen as an extension of the STEP product modelling approach. The kernel of the integration concept is the Building Project Model (BPM).

Chapter 2 characterised CA DfC by six interactions between design and construction. The following draws conclusions about the value of the developed integration concept by identifying how it might support these interactions.

The first abstraction mechanism of the BPM, the product-activity mechanism, supports the formalisation of the relations between product, activity, and resource information in a building project. This enables communication about these relations between project participants, which supports the feedback on the building design from construction (interaction 2) and the backward exchange of constructors' information (interaction 3). It also enables internal integration of project information, which can be used to integrate design and construction applications in a
company when design and construction tasks are shifted (interaction 5 and 6).

The second abstraction mechanism, the concretisation mechanism, provides means to support information needs during problem solving processes, as shown for preliminary design of precast concrete structures in the case study. When the information needs of the various related design disciplines is supported in project models, designers can use project modelling during the design process. Then, project information is available on a high semantic level and in a computer-interpretable format, which makes forward exchange of the building design possible (interaction 1). The separation of problem and solution allows the designer to exchange his requirements to the constructor that designs his own part of the building, which may make the downstream shift of design tasks possible (interaction 6).

As shown in the case study, in which constructability rules are formalised as EXPRESS rules, exchanging constructability knowledge to the designer and applying it during design is possible (interaction 4). Though EXPRESS supports the formalisation and exchange of knowledge rules, it is not very suitable for the knowledge evaluation process itself, because it allows evaluation of one rule at the time only.

The integration concept developed describes how formalisation and exchange of information and knowledge can be improved. As argued in section 2.3, this formalisation and exchange are important for the six interactions of DfC. This means that the proposed integration concept of project modelling with a reference model for building projects is a workable way to support DfC. Evaluation showed that the framework used lacks mechanisms for: data reduction in project models, relating views, and decomposing objects. It also showed that practical application of general conceptual models such as BPM requires a mechanism to define and use subsets of conceptual models. Finally, it showed that complete evaluation of the constructability of a design, requires means to formalise complex constructability rules and to compare conflicting rules.

**Research question 2: computer implementation**

The second question is: "How can the technical implementation of the integration concept for CA DfC be supported with computers?" The answer proposed in chapter 5 is: with a project modelling system that supports three groups of users: the developers of conceptual models, the programmers, and the end-users in practice. PMshell is a prototype version of such a project modelling system that fulfils most requirements of these users as specified in section 5.1.

The conceptual model developer formalises information structures and knowledge rules. In PMshell he can use the conceptual modelling languages NIAM and EXPRESS and he can switch between the two. He is also
able to add (simple) object-oriented functionality to the conceptual model for testing its applicability. He can relate conceptual models with different scopes and develop these models in separate modules. Moreover, PMshell also offers the developer meta-functionality to guide and help the developers of specialised models.

The programmer implements the conceptual models of the developer in a computer environment. This means that he develops interfaces with applications and custom-made user-interfaces for the end-users. For this he can use the object-oriented language in which PMshell is implemented. He is able to build on the work of the conceptual model developer and to develop prototypes rapidly.

The end-user in practice wants to use the conceptual models of the developers and the implementation of the programmers for managing and exchanging information. PMshell enables the import and export of STEP physical files, it has an open database, and application procedures can be called from within the system. The end-users are able to use the functionality defined by the developers and programmers. They are also able to check their project model against the knowledge rules defined by the developers.

The project modelling system developed showed that it is possible to support the implementation of the integration concept for CA DfC with computers. Development and implementation of the BPM and the PtM showed that integrating the CASE tools for the various groups of users in one system is mandatory for smooth transition between the users. Implementation of the system in an object-oriented programming language showed the value of the object-oriented paradigm for implementing conceptual models. Evaluation showed that PMshell has the potentials to become a true project modelling system, but that it needs some major improvements before it can be used in practice. It is clearly still a prototype. Because not all building processes are supported with applications that use information on the semantic level of project models, project modelling systems have to provide means for direct input and manipulation of information and for generating information at lower semantic levels.

**Research question 3: organisational implementation**

The third question is: "How can the integration concept for CA DfC be implemented organisationally?" The answer proposed in chapter 6 is: (1) by starting small scale information exchange, directly on the semantic level of project models, (2) by providing translators that are able to generate information at a lower semantic level from the project model, and (3) by step-by-step enlarging the scale of the information exchange. This resulted in a proposed three stage migration route required to implement CA DfC.
Because in this research CA DfC is only applied on a small scale in a test case, no conclusions can be drawn on the impact of CA DfC on organisational developments observed in current practice. Some global conclusions can be drawn on the first stage of the proposed migration route and the possible influence of external organisations.

Starting CA DfC for a relatively simple and standardised product, in a company that both designs and builds was already difficult, but confirmed its possibilities. The recommendation to start the first stage with a small demonstration project to convince the people involved, seemed correct.

In the case study, developing the conceptual models and the knowledge rules together with the end-users appeared not very difficult. However, their implementation was hindered by the large efforts that had to be put in providing conditions such as availability of modelling tools and standardised general models. This confirms the observation that external organisations can influence the implementation of CA DfC significantly by providing these conditions.

9.2 Recommendations

The recommendations for further development of the proposed strategy for CA DfC follow from the evaluation of chapter 8 and conclusions in the previous section: the fruitful issues have to be worked out in more detail and the missing issues have to be added. For each strategic viewpoint some recommendations are given.

Integration concept

Elaboration of the integration concept as presented in this thesis is needed before it can be fully applied in practice.

The framework is not yet complete. It lacks several mechanisms that should be available. A view mechanism is needed to relate the different views of the various building participants. An object decomposition mechanism is needed to define and apply decomposition structures. A mechanism that relates identical objects is needed to decrease redundancy in project models and to improve support of the design process. Finally, a mechanism is needed that enables the use of subsets of conceptual models.

A complete reference model for the building industry could be useful to be able to extend the application of project modelling to other stages than design and construction. The BPM could serve as basis for a more comprehensive reference model that includes all life cycle stages of the product. Probably, view models have to be developed to support the various participants in the total life cycle of a product. The present abstraction mechanisms, concretisation and product-activity, should be part of
the reference model to support respectively the information requirements during design and construction management, and the integration of product, activity, and resource information.

The integration concept reaches the practical level when project type models for all kinds of product are developed, based on the same reference model. To enable open communication and reuse of software, these project type models should build on the standardised reference model and the resource models as much as possible.

Formalisation of (constructability) knowledge is another issue that deserves attention. Though EXPRESS can be used to formalise and exchange knowledge rules, it only enables qualitative conformance checking of separate rules. A complete evaluation of the constructability of a design can probably better be done by dedicated constructability evaluation applications, e.g., based on knowledge-based techniques.

It is possible to identify groups of participants in the total building process that may profit from the recommendations. General issues of the integration concept (such as the framework and a neutral format for knowledge formalisation) could probably best be dealt with by the research community. Practical issues of the integration concept (such as a reference model, project type models, and formalisation of specific knowledge) could probably be dealt with by the building participants, or by their umbrella organisations. Standardisation bodies such ISO-STEP could intermediate between research and practice.

**Computer implementation**

The evaluation and conclusions showed that a project modelling system like PMshell has potentials for supporting CA DfC, but that it needs improvements before it can be used in practice.

First of all, PMshell has to outgrow the prototype stage. Its robustness, performance, portability, and flexibility have to increase and its hardware requirements have to decrease. A subtyping mechanism has to be found that retains the possibility to define meta-functionality, but that improves the robustness, performance, and flexibility. PMshell is implemented in Eiffel (version 2.3), which supported many object-oriented features, but also caused low performance and portability, and high hardware requirements.

Secondly, the user-friendliness of PMshell has to improve. This can be done by improving the presentation and the graphical possibilities. The user-interface of PMshell is mainly directed to developers and programmers. For end-users there is only a standard menu-driven user-interface. A graphical user-interface has to be developed that is adaptable to the needs of specific end-users.

Of course, when the integration concept is extended with new mechanisms, PMshell also has to support these new mechanisms.
Developing robust and flexible project modelling systems applicable in the practice of the building companies is obviously the task of the software industry. Their systems should at least have the functionality of the PMshell prototype. The research community can show how these systems could be extended by developing new prototypes.

**Organisational implementation**

The case study made clear that implementation of CA DfC needs standardisation of conceptual models, a project modelling system, and semantic-rich applications. Only when international standardisation is possible, it makes sense to develop abstract conceptual models such as resource models and reference models. And only when international project modelling standards are available, it makes sense to develop open, semantic-rich applications based on resource, reference or project type models. Therefore, efforts in development and implementation of CA DfC should be integrated via international standardisation organisation such as ISO-STEP.

The organisational consequences of the implementation of CA DfC are only discussed globally in this thesis and need further research.

Organisational developments in the building industry and other industries that might affect and might be affected by CA DfC must be watched closely. The building participants should develop a vision on how they want to use CA DfC in their companies. Together with participants that have the same vision, they should develop a strategy to reach that vision, e.g., by detailing the migration route sketched in this research. Important issues are the financial and legal aspects of improved integration of design and construction.
References

[AMICE Consortium 1989]

[Andreassen 1987]

[Andreassen, Kahler, et al. 1988]

[ANSI/SPARC 1985]

[Appleton Company 1985]

[ARTB 1993]

[Autodesk 1990]

[Balkenende 1990]

[Ballast Nedam 1993]

[Bedell and Kohler 1992]

[Bennenk and Boom 1992a]
Bennenk, H.W. and Boom, P. van, “EKON-systeem geeft ruimte aan gebruikers en ontwerpers (I),” Cement, vol. 44 (issue 2, 1992a), pp. 16-21 (in Dutch: The EKON system gives space to users and designers (I)).

[Bennenk and Boom 1992b]
Bennenk, H.W. and Boom, P. van, “EKON-systeem geeft ruimte aan gebruikers en ontwerpers (II),” Cement, vol. 44 (issue 5, 1992b), pp. 21-28 (in Dutch: The EKON system gives space to users and designers (II)).

[Björk 1991]
[Björk 1993]

[Björk and Hannus 1992]

[Björk, Penttilä, et al. 1989]

[Boender, Bruijn, et al. 1989]

[Boender and Pronk 1991]

[Böhm 1991]

[Booch 1991]

[Boothroyd and Dewhurst 1987]

[Brodie 1986]

[Brodie, Mylopoulos, et al. 1986]

[Bruijn 1990]

[Choi and Kim 1993]

[Dale 1991]

[Dekker and Vreeze 1985]
Dekker, K.H. and Vreeze, N. de (ed.), *Aanbesteding, bouwteam van brochure plan. Wat bepaalt de keuze?*, vol. 124, SBR, Foundation for Building Research, Rotterdam, the Netherlands, 1985 (in Dutch: Tendering, building team and brochure plan methods of project preparation. What determines the choice?).

[Design Power 1991]
[Dupagne 1991]

[Eastman, Bond, et al. 1991]

[Eijs 1992]

[Eijs 1993]

[Fenves, Hendrickson, et al. 1990]

[FIP 1986]

[Fischer 1989]

[Fischer 1991]

[Froese 1992]

[Gelder 1989]

[Gieilingh 1988]

[Gieilingh and Suhm 1993]

[Hannus 1987]

[Haselhoff and Rijlaarsdam 1988]

[Hoek 1991]


[ISO/TC184 1993f]

[ISO/TC59 1993]

[Joosten and Werven 1991]


[Kolountzakis and Fischer 1991]

[Koskela 1992]

[Krom and Tolman 1992]

[Kuiper 1989]

[Kuiper, Boender, et al. 1992]

[Kuiper, Tolman, et al. 1992]

[Laitinen 1992]

[Lucardie 1992]

[Lucardie 1994]
[Luijten and Luijten 1993]

[Luijten, Froese, et al. 1992]

[Meyer 1988]

[Meyer and Fenves 1993]

[Mischgofsky 1991]

[Miyatake, Yamazaki, et al. 1992]

[Miyiwaka and Toshirjo 1986]

[Mooij 1993]

[Morrel and Smies 1991]

[Newell 1981]

[Nijssen and Halpin 1989]

[Penttilä, Talonpoika, et al. 1990]

[Pohl 1991]
Pohl, J., ICADS working model version 2 and future direction, California Polytechnic State University, School of Architecture and Environmental Design, USA, 05-91, 1991.
[Poyet, Debras, et al. 1993]

[Prasad, Morenc, et al. 1993]

[Ranyak 1991]

[Ray-Jones and McCann 1971]

[Reed 1991]

[Siemens Nixdorf 1992]

[Smith 1986]
Smith, B.M., PDES Initiation Activities, NIST, 1986.

[Smith and Smith 1977]

[SofTech 1981]

[Sol 1982]

[Sowa 1984]

[Spanbeton B.V. 1991]

[Sriram, Logcher, et al. 1989]
Sriram, D., Logcher, R.D., Groleau, N., et al., DICE, an object oriented programming environment for co-operative engineering design, MIT, Dep. of Civil Engineering, USA, IESL-89-03, 1989.

[TeleSoft AB 1990]

[Toepoel 1992]
Toepoel, L., Prefabricage en Automatisering, Delft University of Technology, 1992, (part 1; in Dutch: Prefabrication and Automation).

[Tolman 1989]
[Tolman 1990]

[Tolman 1991]
Tolman, F.P., Integration Core Model for Product Modelling, CAM-I, C90-PMP-02, 1991, (also published as TNO report B-91-0885).

[Turk 1992]

[UN/ECE 1990]

[Vogel 1991]

[Vos and Elkhuizen 1992]

[Waard 1992]

[Willems 1988]

[Willems 1990]
Willems, P.H., The Road Model Kernel, Version 0.2, TNO Building and Construction Research, B-89-831, March 1990.

[Willems 1992]

[Willems 1993]

[Willems, Kuiper, et al. 1991]

[Wilson 1992]

[Wilson and Pratt 1986]

[Wirth 1977]
[Womack, Jones, et al. 1990]

[Woodruff 1993]

[Wright 1963]
Author Index

AMICE Consortium 4
Andreasen 3, 22, 140
ANSI/SPARC 27
Appleton Company 28
AR TB 44, 109, 111
Autodesk 13, 24, 113
Balkenende 103
Bedell 33
Bennenk 125
Björk 26, 33, 35, 54, 56
Boender 33, 88, 104
Böhms 36
Bond 40
Booch 40
Boom, van 125
Boothroyd 21, 140
Brodie 47
Bruijn, de 33, 41, 88
Choi 134
Dale 33
Debras 42
Dekker 44
Design Power 42, 116
Dewhurst 21, 140
Dupagne 26
Eastman 40
Eijs 88
Elkhuizen 110, 117
Fenves 40, 128
FIP 141
Fischer 2, 40, 140
Froese 55, 56
Gelder, de 33
Gielingh 27, 31, 33, 34, 35, 53, 67, 68, 74
Halpin 28, 32, 47
Hannus 40, 56
Haselhoff 44, 112
Hendrickson 40
Hoek, van den 9, 134
Intergraph Corporation 24, 41
IOP-Bouw 9
ISO/IEC 29
ISO/TC184 25, 27, 28, 29, 36, 62, 79, 80, 113, 153
ISO/TC59 24, 70
Jones 45
Joosten 44
Kähler 3, 140
Kato 41
Kim 134
Kita 41
Kohler 33
Kolountzakis 40
Koskela 45
Krom 44, 110
Kuiper 40, 88, 94
Laitinen 111
Logcher 40
Lucardie 2, 25
Luijten 88
Luiten 56, 88
McCann 24
Meyer, B. 40, 88, 91
Meyer, S. 128
Mischgofsky 44, 118, 119
Miyatake 110
Miyiwaka 22
Mooij 88
Morenc 45
Morrel 124
Mylopoulos 47
Newell 2
Nijssen 28, 32, 47
Penttilä 26, 40
Pohl 40
Poyet 42
Prasad 45
Pratt 23
Pronk 104
Ranyak 22
Ray-Jones 24
Reed 13, 24
Rijlaarsdam 44, 112
Siemens Nixdorf 42
Smies 124
Smith, B. 27
Smith, D.C.P. 47
Smith, J.M. 47
SofTech 7, 52
Sol 40
Sowa 99
Spanbeton B.V. 125
Sriram 40
Suhm 33, 35, 53, 68, 74
Talonpoika 40
TeleSoft AB 96
Toepoel 123
Tolman 8, 27, 31, 85, 88
Toshirjo 22
Turk 40
UN/ECE 116
Vogel 88
Vos 110, 117
Vreeze, de 44
Waard, de 33, 36, 120, 141
Werven, van 44
Willems 33, 34, 59, 68, 94, 104
Wilson 23, 41
Wirth 31, 96
Womack 45
Woodruff 88
Wright, von 49
Yamazaki 110
Subject Index

abstraction mechanism 47
Activity 48, 60, 138
Actor 58
ANSI/SPARC 31
Application Protocol (AP) 29
assemblability evaluation method 21, 22
assembly process 137, 145, 146
Ballast Nedam 8, 9, 111, 124
Basic-Graph domain 99, 106, 187, 188
BPM 47-75, 97, 105, 127, 152, 155,
157, 160, 187
Building Information Model (BIM) 9
building process 9-12
building team 2, 14, 45, 116
CIM framework 36-38, 84, 97, 152, 160
classification 24, 47, 70
co-makership 2, 44, 118
combining abstraction mechanisms 73
communication 12
complex shape parts model 33
Computer Aided Conformance Checking
36
Computer Aided Design for Construction
(CA DfC) 4
Computer Aided Systems Engineering
(CASE) 41
Computer Integrated Construction (CIC) 8
Computer Integrated Manufacturing
(CIM) 36
conceptual model 28
concretisation 71, 73, 106, 143, 152,
158, 160
concretisation. 64
concurrent engineering 2, 14, 45
Configurable Graphical Editor 88
constructability knowledge 15, 22, 36,
39, 45, 78, 140, 145, 153, 155, 158,
161
construction automation 110
construction robots 110
constructor 11
Cost 60
decomposition 38, 48, 66, 106, 134,
152, 154, 160
Delft University of Technology 8
Design for Assembly (DfA) 3, 21, 140
Design for Construction (DfC) 3, 9-19,
158
Design for Manufacturing (DFM) 21
design-and-build 2, 44, 111
designer 11
developer 77, 84
divide-and-conquer 34, 66, 152
domain 89
domain structure 90, 97
DXF 13, 24
EDI 116
Eiffel 88, 95, 161
EKON 125, 127-149, 193
end-user 77, 78
EXIS 41, 88
expert system 25
EXPRESS 28, 31, 36, 37, 41, 48, 84, 89,
96, 141, 145, 152, 153, 193
EXPRESS-G 28, 85
feature modelling 23
Flexible Production Automation (FPA) 22
Flyover Model Kernel 33
formalisation 12
Functional Unit 32, 34, 67
GARM 31, 34, 67
GenCOM 55
general resource 32, 85
Generalised-Topology domain 104, 190
graph layer 98
IDEFO 7, 9, 52
IDEFIX 28
IMPPACT reference model 33, 35, 53, 68
information 2
information management 83
Information Resource Dictionary System
(IRDS) 29
integrated resource 28, 32
integration 2
interactions of DfC 11, 12-16, 157
IRMA 56, 58
ISO-STEP 25, 27-29, 39, 120, 161, 162
kernel of PMShell 90, 91-96
knowledge 2
knowledge-based (KB) 25, 36, 161
layer in PMShell 89
lean production 45
learning-by-doing 112
life cycle cost model for buildings 33
logic of change 49-52
Material 59
meta-functionality 86, 94, 95, 101, 107, 143, 154
migration route 113, 123, 156, 162
modelling languages 29-31, 84, 152, 154
modularity 85, 155
network functionality 99, 128
NIAM 28, 37, 47, 48, 49, 73, 84, 96, 152, 187
object type 28
object-oriented 40, 43, 88, 155, 159
PDES 27
performance contract 2, 16, 73, 119
plate girder bridge model 33
PMSGQT domain 103, 145, 189
PMShell 77-108, 142, 154, 158, 161, 187
problem solving 64
Product 48, 59
product design 127, 142
Product domain 106, 143, 192
product model 28
product modelling 25, 26-36, 39
product type model 32
Product-Activity domain 108, 145, 192
programmer 77, 86
Project 49, 107
project modelling 35, 38, 109
Project Object 49
project type model 38, 85, 98, 155, 161, 162
precast concrete structures 123-149, 193
ProMod 40, 103
Proposed Object 71
Proposed Product 128, 143
rapid prototyping 87
RATAS 33, 40, 70
Realised Object 71
reference layer 98, 105
reference model 32, 38, 85
Required Object 71
Required Product 128, 143
residential building model 33
Resource 48, 60, 138
resource layer 97, 103
Road Model Kernel 32, 34
rule
  global 92
  local 92, 145
  scope 31-33, 85, 97

semantic level 13, 17
separation of definition, representation,
  and presentation 27, 103, 113
separation of problem and solution 34,
  65, 70, 72, 152, 158
Shape 59
Shape domain 105, 106, 143, 191
sheet metal parts model 33
shift of tasks 12
Shop Floor Production Model 62
Spanbeton 8, 124
specialisation 48
specialisation-by-instantiation 94
Standard Data Access Interface (SDAI)
  28, 31, 79
standardisation 18
State 57, 137
steel structures model 33
STEP physical file format 28, 31, 96
Sub-Super-Graph domain 102, 106, 189
supplier 109
Technical Solution 32, 34, 67
Time 60
TNO Building and Construction Research
  8, 88
Total Quality Management 2
umbrella organisation 161
umbrella organisations 120
Unified Approach Model 35, 54
Waco-Liesbosch Beton 9
Wirth Syntax Notation (WSN) 31, 96
Samenvatting

De laatste decennia is de bouw geconfronteerd met stijgende eisen vanuit de maatschappij. Er is vraag naar hogere kwaliteit, kortere produktietijden, gebouwen op maat, meer aandacht voor het milieu en betere werkomstandigheden. Het is waarschijnlijk dat deze eisen zullen blijven toenemen.

Een mogelijk antwoord van de bouwindustrie op deze eisen is het verbeteren van de integratie van ontwerp en uitvoering. De integratie van deelprocessen van een bouwproject kan worden gedefinieerd als "het continuïteit en interdisciplinair delen van doelen, kennis en informatie tussen alle deelnemers" [Fischer 1989]. Design for Construction (DfC) heeft betrekking op de integratie van ontwerp en uitvoering. Binnen de bouw is DfC de tegenhanger van Design for Assembly en Design for Manufacturing uit de werktuigbouw.


Dit proefschrift bekijkt de strategie voor CA DfC vanuit drie gerelateerde gezichtspunten:

- integratieconcept,
- computer-implementatie en
- organisatorische implementatie.

Het integratieconcept gaat uit van de veronderstelling dat voor integratie van de bouwprocessen, hun computerprogramma's geïntegreerd moeten worden. Om de programma's te integreren moeten de gebruikte informatie en kennis uitwisselbaar zijn. Hiervoor moeten het formaliseren en uitwisselen van informatie en kennis worden standaardiseerd.

Technische implementatie van het integratieconcept vereist ondersteuning door de computer. Er is onder andere ondersteuning nodig voor het ontwikkelen van standaards en voor het gebruik van deze standaards voor uitwisseling tussen computerprogramma's in de praktijk.

Voordat bouwbedrijven CA DfC zullen toepassen, willen ze weten wat het te bieden heeft en hoe zij ernaar toe kunnen groeien.
Als eerste beschrijft dit proefschrift de stand van zaken van CA DfC. Vervolgens worden de drie gekozen strategische gezichtspunten uitge- werkt en getest in een case-study. Uiteindelijk worden de resultaten van het onderzoek geëvalueerd.

Stand van zaken van Design for Construction

De graad van integratie van ontwerp en uitvoering in het huidige bouwproces kan worden gekarakteriseerd door de graad van integratie van zes "interacties" tussen ontwerpers en uitvoerders:

1. voorwaartse overdracht van het bouwontwerp,
2. tegerkoppeling op het bouwontwerp vanuit de uitvoering,
3. achterwaartse overdracht van de informatie van de uitvoerder,
4. achterwaartse overdracht van algemene uitvoeringskennis,
5. verschuiving van uitvoerings-managementtaken naar voren en
6. verschuiving van ontwerptaken naar achteren.

Interacties 1, 2 en 3 beslaan de uitwisseling van informatie, interactie 4 de uitwisseling van kennis en interacties 5 en 6 de verschuiving van taken tussen ontwerp en uitvoering. Bij elke interactie wordt betere integratie van ontwerp en uitvoering zowel op technisch als organisatorisch gebied gehinderd. Algemeen gezien heeft de introductie van computers geled tot optimalisering van de deelprocessen. Echter de communicatie tussen deze zogenaamde eilanden van automatisering, is niet wezenlijk veranderd: meestal communiceren de deelnemers van een bouwproject nog steeds met papieren documenten die door mensen geïnterpreteerd moeten worden.

Integratieconcept voor CA DfC

Het eerste gezichtspunt van de strategie voor CA DfC is het integratieconcept. Dit concept standaardiseert de formalisering en uitwisseling van informatie en kennis in een bouwproject.

Het integratieconcept voor CA DfC is gebaseerd op verbetering van communicatie door gebruik te maken van projectmodellen. Projectmodelleren richt zich op formalisering en uitwisseling van projectinformatie in bouwprojecten, en wel zodanig dat computerprogramma's de informatie direct kunnen "begrijpen" en gebruiken. Op basis van verwant onderzoek is het integratieconcept gestructureerd met een raamwerk dat zoveel mogelijk gebruik maakt van de produktmodeel-aanpak van STEP [ISO/TC184 1993b].

Onlangs heeft STEP modelleertalen gestandaardiseerd, die kunnen worden gebruikt om informatiestucturen te formaliseren (EXPRESS) en om data tussen programma's uit te wisselen (STEP physical file-formaat).
Deze talen zorgen ervoor dat computers kunnen communiceren met de-zelfde syntax.

Standaardisatie van de syntax is echter niet voldoende voor betekenisvolle communicatie. Ook over de woorden en hun betekenis (de semantiek) moet overeenstemming worden bereikt. Dit gebeurt in conceptuele modellen. Conceptuele modellen kunnen worden ontwikkeld voor elk kennisgebied. Tolman [1991] stelt voor drie soorten conceptuele modellen te gebruiken om deze modeller-inspanningen te structureren:

- resource-modellen,
- referentiemodellen en
- projecttypemodellen.


Dit onderzoek richt zich op het ontwikkelen van een referentiemodel voor de bouw. Resource-modellen worden reeds ontwikkeld binnen de STEP wereld. De case-study demonstreert de specialisatie van het referentiemodel tot een projecttypemodel.

Naast standaardisatie van betekenisvolle communicatie, is het met EXPRESS mogelijk eenvoudige uitvoeringskennis te formaliseren en uit te wisselen.

Het ontwikkelde referentiemodel is het Bouw Project Model (BPM) genoemd. Het gebruikt twee abstractie-mechanismen om de informatie in een bouwproject te structureren:

- een produkt-activiteit-mechanisme en
- een concretiseringsmechanisme.

Het produkt-activiteit-mechanisme relateert produktinformatie met activiteit-, toestands-, hulpmiddelen- en actoinformatie. Het model is gebaseerd op de logica van veranderingen van Von Wright [1963], de IDEF0 procesanalyse-methode en referentiemodellen uit verwant onderzoek.


De twee mechanismen samen vormen het BPM: het concretiseringsmechanisme is toegepast op produkt-, activiteit- en hulpmiddeleninformatie, welke zijn gerelateerd met het produkt-activiteit-mechanisme.
Computer-implementatie van het integratieconcept

Het tweede strategische gezichtspunt is de computerondersteuning van de technische implementatie van het integratieconcept. Verwant onderzoek leert dat ontwikkeling van conceptuele modellen en hun toepassing in de praktijk ondersteuning door computers vereisen. Dit onderzoek bekijkt de mogelijkheden om de ontwikkeling en toepassing van conceptuele modellen te ondersteunen met een geïntegreerd projectmodeller-systeem.

Op basis van een analyse van de behoeften van toekomstige gebruikers zijn eisen voor een projectmodeller-systeem opgesteld.

Het projectmodeller-systeem moet de STEP talen EXPRESS en het STEP physical file-formaat ondersteunen. Voor presentatie en de eerste ontwikkelingsfasen moet ook een grafische conceptuele modelleretaal zoals NIAM worden ondersteund. Bovendien moet het projectmodeller-systeem een mechanisme ondersteunen, dat de conceptuele modellen met verschillende gebieden kan relateren. Tenslotte moet het projectmodeller-systeem minimaal drie mogelijk gebruikersgroepen ondersteunen: de ontwikkelaars van de conceptuele modellen, de programmeurs en de eindgebruikers in de praktijk. Elke groep gebruikers heeft zijn eigen specifieke eisen. Om een goede samenwerking tussen de drie te waarborgen moeten de gaten tussen ontwikkeling van de conceptuele modellen, hun implementatie en hun toepassing worden overbrugd.

In nauwe samenwerking met TNO is een prototype van een projectmodeller-systeem ontwikkeld. Dit prototype is PMshell genoemd, naar ProjectModeller-omgeving.

Omdat het object-georiënteerde paradigma nauw aansluit op het conceptueel modelleren, is PMshell geïmplementeerd in de object-georiënteerde programmeertaal Eiffel. Het heeft een menu-gestuurde interface voor alle gebruikers en een grafische, op NIAM lijkende, gebruikersinterface voor de ontwikkelaars en programmeurs van conceptuele modellen. Het ondersteunt de STEP-talen. PMshell ondersteunt de ontwikkelaar en programmeur bij het specificeren van karakteristieken en gedrag van conceptuele modellen per interessegebied. Deze conceptuele modellen kunnen worden gerelateerd. Eindgebruikers kunnen de gedefinieerde eigenschappen en het gedefinieerde gedrag gebruiken om informatie over een specifiek project te modelleren. Ontwikkelaars en programmeurs kunnen niet alleen gedrag voor de eindgebruikers specificeren, maar ook voor ontwikkelaars van andere conceptuele modellen. PMshell geeft ontwikkelaars de mogelijkheid (uitvoerings)regels te definiëren en eindgebruikers de mogelijkheid hun projectmodel te testen op deze regels.

Representatieve delen van het referentiemodel BPM zijn geïmplementeerd met PMshell. Om deze implementatie mogelijk te maken zijn algemene zaken als netwerk-functionaliteit en vormdefinitie, -representatie en -presentatie, geïmplementeerd. Concretisering van het produkt gedurende
ontwerp is geïmplementeerd, inclusief een opdeling die het ontwerpproces volgt. Voor activiteiten en hulpmiddelen zijn de relatie met het produkt en zijn toestand gedurende de uitvoering geïmplementeerd.

**Organisatorische implementatie van het integratieconcept**

De organisatorische implementatie van CA DfC in de praktijk is het derde strategische gezichtspunt dat in dit proefschrift is geanalyseerd. In de huidige praktijk worden organisatorische ontwikkelingen onderscheiden die met CA DfC ondersteund zouden kunnen worden. Indien deelnemers aan het bouwproces CA DfC willen toepassen, moeten zij van het begin af aan communiceren op het semantische niveau van projectmodel len. Voor communicatie met bedrijven die niet op dit niveau communiceren, zijn additionele vertalers nodig die informatie op een lager niveau genereren uit het projectmodel. Voor de organisatorische implementatie is het belangrijk dat de deelnemers er actief bij betrokken zijn en dat de schaal waarop gecommuniceerd wordt stap voor stap wordt vergroot. Externe organisaties, zoals overkoepelende organisaties en de software industrie, kunnen de toepassing van CA DfC versnellen. De overkoepelende organisaties kunnen werken aan standaardisatie van conceptuele modellen. De software-industrie kan de bouw voorzien van modelleerhulpmiddelen en computerprogramma’s die het semantische niveau van projectmodellen ondersteunen.

**Toepassing van de CA DfC-strategie in een case-study**

In de case-study is de CA DfC-strategie toegepast voor een bepaald type geprefabriceerde betonconstructies voor (kantoor)gebouwen. Vanuit de drie in dit rapport beschouwde gezichtspunten wordt gekeken naar de integratie van ontwerp en uitvoering bij een prefab-leverancier.

Het onderwerp van demonstratie is gekozen op basis van de aanbevelingen voor de organisatorische implementatie, namelijk voorontwerp en assemblage op de bouwplaats van het in hoge mate gestandaardiseerde EKON systeem. Hoewel deze eerste demonstratie zo eenvoudig mogelijk is gehouden, laat het toch veel aspecten van CA DfC zien. Het BPM is gesignaliseerd in een projecttypemodel voor prefab-constructies. In het model zijn de tijdens het voorontwerp gebruikte produktinformatie, de informatie over de toestand van het gebouw gedurende de assemblage en de assemblage-activiteiten opgenomen. Ook zijn een aantal uitvoeringsregels gedefinieerd. Het projecttypemodel is geïmplementeerd door gebruik te maken van de BPM-implementatie in PMSshell. In een case is een projectmodel voor ontwerp en assemblage van een bepaalde prefab-constructie gegenereerd met deze implementatie.
Evaluatie en conclusies

De case-study demonstreerde een implementatie van CA DfC. Het liet zien dat de technische implementatie niet eenvoudig is, en dat er nog veel werk verricht moet worden voordat de bouw ten volle van CA DfC kan profiteren.

De case-study liet zien hoe enkele relevante aspecten van de interacties tussen ontwerp en uitvoering kunnen worden ondersteund door CA DfC. Het lijt erop dat de integratie van ontwerp en uitvoering kan worden verbeterd door de uitwisseling (en formalisering) van kennis en informatie tussen computerprogramma's te ondersteunen. Het raamwerk en het Bouw Project Model zijn een stap in de richting van CA DfC, maar behoeven extra aandacht voordat zij volledig toepasbaar zijn in de praktijk. De voorgestelde manier van formaliseren van uitvoeringsregels bleek nog van beperkte waarde. Evaluatie van de uitvoerbaarheid van een ontwerp kan waarschijnlijk beter plaatsvinden in toegespitste programma's zoals expertsystemen.

Het gebruik van PMshell maakte het aannemelijk dat de technische implementatie van het integratieconcept ondersteund kan worden door computers. Hoewel PMshell een goede stap in de juiste richting is, is het nog niet rijp voor toepassing in de praktijk. Onder andere moeten de robuustheid, de gebruiksvriendelijkheid en de flexibiliteit worden verbeterd.

Omdat de case-study gericht was op het integratieconcept en de technische implementatie ervan, kan niet veel worden geconcludeerd ten aanzien van de organisatorische implementatie. De problemen die naar voren kwamen bij de integratie van ontwerp en uitvoering binnen een bedrijf, wijzen erop dat integratie over bedrijfsgrenzen zeer moeilijk te realiseren zal zijn. Een meer gedetailleerd migratiedpad naar CA DfC in de bouw lijkt dus belangrijk en vereist daarom meer onderzoek.

Als algemene conclusie kan worden gesteld dat de voorgestelde strategie voor CA DfC kan bijdragen aan een betere integratie van ontwerp en uitvoering. Een aantal technische en organisatorische problemen die momenteel DfC in de zes interacties hinderen, kunnen, tenminste gedeeltelijk, overbrugd worden. Het ontwikkelde integratieconcept (met computerondersteuning) geeft middelen om informatie en kennis te formaliseren en uit te wisselen tussen de deelnemers in het bouwproces. Of deze verbeterde communicatiemogelijkheden tevens zullen leiden tot het delen van doelen, en uiteindelijk, naar betere antwoorden op de eisen vanuit de maatschappij, kan alleen de toekomst ons leren.
Curriculum Vitae

1965  Born at Harmelen, The Netherlands
1977-1983  Atheneum B, St.-Antoniuscollege at Gouda
1983-1984  Propaedeutics in Civil Engineering, cum laude
           Delft University of Technology
1984-1989  MSc degree in Civil Engineering
           Delft University of Technology
           Majored in numerical mechanics
           Advisors: prof.dr.ir. J. Blaauwendraad, prof.dr.ir. R. de Borst
           Thesis: The implementation of a three dimensional rubber element in a
           finite element method program
1989-today Research assistant at the Civil Engineering department of Delft
         University of Technology
         Working as a candidate for a doctorate, supporting graduate students,
         and lecturing
         The research was conducted in close relationship Ballast Nedam BV and
         TNO Building and Construction Research. In the course of the research
         congresses were visited and papers were published
Bibliography


Appendix A  The BPM in PMshell

This appendix contains the NIAM-like diagrams of the seven domains of the BPM that are implemented in PMshell.

- **PM OBJECT**: object type from a parent domain
- **PREFAB BEAM**: object type defined in current domain
- **A**: (uni-directional) attribute, no cardinality limitations
- **F**: function, minimum cardinality limited
- **RA**: redefined attribute, maximum cardinality limited
- **RP**: redefined function, minimum and maximum cardinality limited

**Figure A.1** special symbols of the NIAM-like user interface of PMshell

**Figure A.2** object types and subtype relations in Basic-Graph
Figure A.3 attributes and functions relating object types in Basic-Graph
figure A.4  object types, subtype relations, and attributes of Sub-Super-Graph

figure A.5  object types, subtype relations, attributes, and functions of PMSGQT
figure A.6 object types, subtype relations, attributes, and functions of Generalised-Topology
figure A.7  object types, subtype relations, attributes, and functions of Shape
**figure A.8**  object types, subtype relations, attributes, and functions of Product

**figure A.9**  object types and attributes of Product-Activity
Appendix B  Project type Model for EKON structural systems

This appendix describes a project type model for EKON structural systems in EXPRESS. The EXPRESS schema is generated with PMshell after implementation of the PIM.

SCHEMA ekon;

REFERENCE FROM bpm;

ENTITY structural_area
  SUBTYPE OF (required_product);
  shape: rectangular_block;
  building_type: STRING;
END_ENTITY;

ENTITY ekon_bear_struct
  SUBTYPE OF (proposed_product);
  shape: rectangular_block;
  fulfills: structural_area;
  bearing_direction: CHARACTER;
WHERE
  grid_conforms_variax: ....;
  available_crane_ok: ....;
  available_truck_ok: ....;
  slabs_hieght_ok: ....;
  no_columns_interconnected: ....;
END_ENTITY;

ENTITY column_row
  SUBTYPE OF (required_product);
  shape: rectangular_face;
  is_part_of: ekon_bear_struct;
END_ENTITY;

ENTITY floor_field
  SUBTYPE OF (required_product);
  charac_load: REAL;
  shape: rectangular_face;
  is_part_of: ekon_bear_struct;
END_ENTITY;
ENTITY ekon_column_system
  SUBTYPE OF (proposed_product);
  shape: rectangular_face;
  fulfills: column_row;
DERIVE
  vert_line_elements:
    LIST [0:?] OF vert_line_element := ....;
END_ENTITY;

ENTITY ekon_floor_system
  SUBTYPE OF (proposed_product);
  shape: rectangular_face;
  fulfills: floor_field;
DERIVE
  hor_surf_elements:
    LIST [0:?] OF hor_surface_element := ....;
  hor_line_elements:
    LIST [0:?] OF hor_line_element := ....;
END_ENTITY;

ENTITY vert_line_element
  SUBTYPE OF (required_product);
  shape: straight_line;
  max_N_force: REAL;
  max_M_force: REAL;
  max_Q_force: REAL;
  is_part_of: ekon_column_system;
END_ENTITY;

ENTITY hor_surface_element
  SUBTYPE OF (required_product);
  shape: rectangular_face;
  max_M_force: REAL;
  max_Q_force: REAL;
  is_part_of: ekon_floor_system;
END_ENTITY;

ENTITY hor_line_element
  SUBTYPE OF (required_product);
  shape: straight_line;
  max_M_force: REAL;
  max_Q_force: REAL;
  is_part_of: ekon_floor_system;
END_ENTITY;

ENTITY precast_element
  SUBTYPE OF (proposed_product);
  material: STRING;
  producer: STRING;
DERIVE
  supported_by:
    LIST [0:?] OF precast_element := ....;
  supports:
    LIST [0:?] OF precast_element := ....;
END_ENTITY;
ENTITY ekon_column;
    SUBTYPE OF (precast_element);
    width_side_1: REAL;
    width_side_2: REAL;
    length: REAL;
    fulfills: vert_line_element;
DERIVE
    supp_by_foundation: OPTIONAL foundation := ....;
    supported_by:
        LIST [0:1] OF ekon_column := ....;
WHERE
    standardised_dimensions: ....;
    grid_conforms_variax: ....;
END_ENTITY;

ENTITY ekon_beam
    SUBTYPE OF (precast_element);
    width: REAL;
    height: REAL;
    flange_height: REAL;
    flange_width: REAL;
    length: REAL;
    shape_type: CHARACTER;
    fulfills: hor_line_element;
DERIVE
    supported_by:
        LIST [2:2] OF ekon_column := ....;
    supports:
        LIST [0:?] OF variax_slab := ....;
WHERE
    standardised_flange_dimensions: ....;
    width_EQ_column_width: ....;
    height_EQ_slab_height: ....;
    left_and_right_same_slab: ....;
END_ENTITY;

ENTITY variax_slab
    SUBTYPE OF (precast_element);
    height: REAL;
    width: REAL;
    length: REAL;
    fulfills: hor_surface_element;
DERIVE
    profile_type: STRING;
    supported_by:
        LIST [2:2] OF ekon_beam := ....;
    weight_p_m: REAL := ....;
    weight_p_m2: REAL := ....;
WHERE
    standardised_height: ....;
    width_LE_1200: ....;
    width_corresponds_channels: ....;
    length_LE_9000: length <= 9000;
END_ENTITY;
ENTITY precast_connection
  SUBTYPE OF (proposed_product);
    reinf_material: STRING;
    steel_material: STRING;
    producer: STRING;
    fulfills: interface;
DERIVE
  connects:
    LIST [0:?] OF precast_element := ....;
END_ENTITY;

ENTITY foundation_column_connection
  SUBTYPE OF (precast_connection);
    nr_of_bars: INTEGER;
    diameter_of_bars: REAL;
    length_of_gains: REAL;
END_ENTITY;

ENTITY column_column_connection
  SUBTYPE OF (precast_connection);
    nr_of_beam_bars: INTEGER;
    diam_beam_bars: REAL;
    nr_of_column_bars: INTEGER;
    diam_column_bars: REAL;
    length_of_gains: REAL;
    WHERE
      preferably_not_used: FALSE;
END_ENTITY;

ENTITY bsf_connection
  SUBTYPE OF (precast_connection);
    type: INTEGER;
END_ENTITY;

ENTITY beam_slab_connection
  SUBTYPE OF (precast_connection);
    diam_of_bars: REAL;
    length_of_bars: REAL;
    nr_of_length_bars: INTEGER;
    diam_length_bars: REAL;
DERIVE
  length: REAL;
END_ENTITY;

ENTITY slab_slab_connection
  SUBTYPE OF (precast_connection)
DERIVE
  length: REAL;
END_ENTITY;
ENTITY ekon_project
  SUBTYPE OF (project);
  available_cranes: LIST [0,?] OF resource;
  available_trucks: LIST [0,?] OF resource;
  WHERE
    all_variax_mat_constant: ....;
    all_column_mat_constrant: ....;
    all_beam_mat_constant: ....;
    all_variax_height_constant: ....;
    all_column_dim_constant: ....;
    all_beam_dim_constant: ....;
    all_connection_concr_mat_constant: ....;
    all_connection_reinf_mat_constant: ....;
    all_connection_reinf_diameter_constant: ....;
    capacity_cranes_ok: ....;
    capacity_truck_ok: ....;
END_ENTITY;

ENTITY element_fabricated
  SUBTYPE OF (state)
    of_proposed_product: precast_element;
  WHERE
    correct_sequence: ....;
END_ENTITY;

ENTITY element_on_site
  SUBTYPE OF (state)
    of_proposed_product: precast_element;
  WHERE
    correct_sequence: ....;
END_ENTITY;

ENTITY element_on_position
  SUBTYPE OF (state)
    of_proposed_product: precast_element;
  WHERE
    correct_sequence: ....;
END_ENTITY;

ENTITY element_fixed
  SUBTYPE OF (state)
    of_proposed_product: precast_element;
  WHERE
    correct_sequence: ....;
END_ENTITY;

ENTITY element_realised
  SUBTYPE OF (state)
    of_proposed_product: precast_element;
  WHERE
    correct_sequence: ....;
END_ENTITY;
ENTITY connection_realised
SUBTYPE OF (state)
of_proposed_product:  precast_connection;
WHERE
correct_sequence: ....;
END_ENTITY;

ENTITY transport_element
SUBTYPE OF (proposed_activity)
starts_with:  element_fabricated;
ends_with:  element_on_site
END_ENTITY;

ENTITY lift_element
SUBTYPE OF (proposed_activity)
starts_with:  element_on_site;
ends_with:  element_on_position
END_ENTITY;

ENTITY fix_element
SUBTYPE OF (proposed_activity)
starts_with:  element_on_position;
ends_with:  element_fixed
END_ENTITY;

ENTITY realise_connection
SUBTYPE OF (proposed_activity)
starts_with:  element_fixed;
ends_with:  connection_realised
END_ENTITY;

ENTITY transport_element
SUBTYPE OF (proposed_activity)
starts_with:  connection_realised;
ends_with:  element_realised
END_ENTITY;

ENTITY truck
SUBTYPE OF (proposed_resource)
capacity_in_kN:  REAL;
length_cap_in_mm:  REAL;
width_cap_in_mm:  REAL;
height_cap_in_mm:  REAL;
END_ENTITY;

ENTITY crane
SUBTYPE OF (proposed_resource)
capacity_in_kN:  REAL;
END_ENTITY;

END_SCHEMA; -- schema ekon