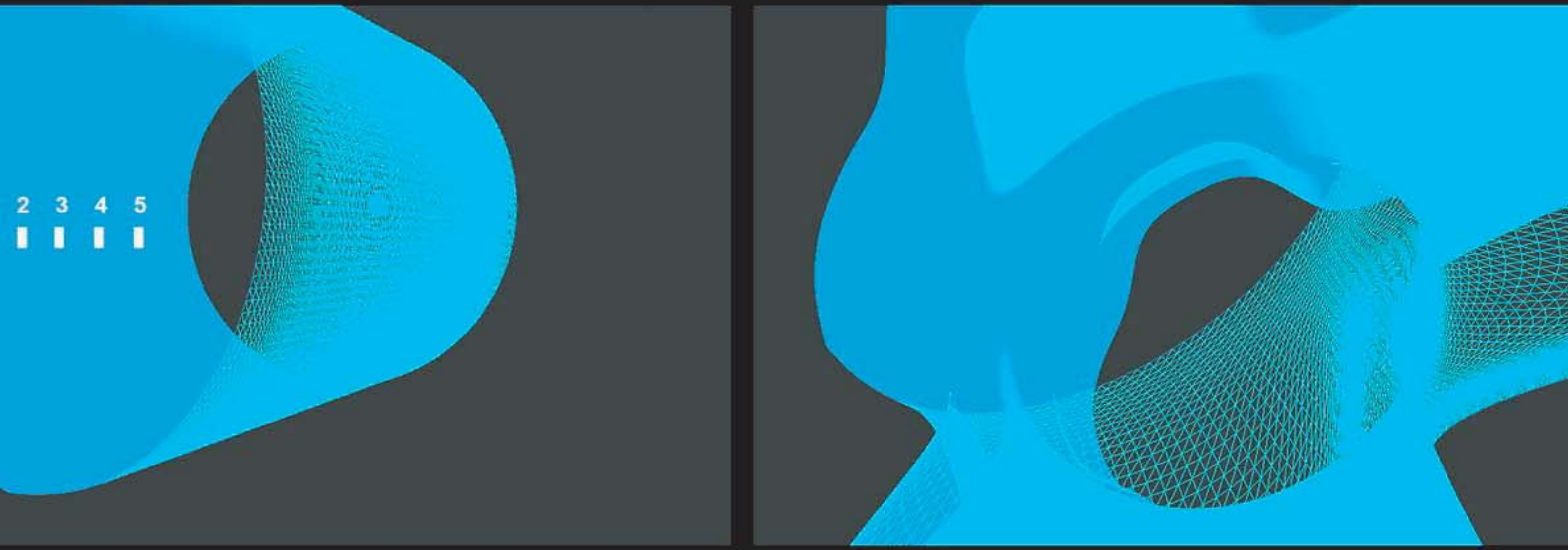
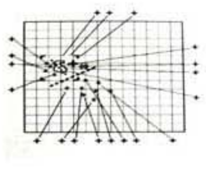


SYSTEM-EMBEDDED INTELLIGENCE IN ARCHITECTURE

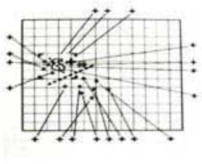
H. Bier: Founded on the imperative to understand, evaluate and consciously decide about the use of digital media in architecture this research not only aims to analyze and critically assess computer-based systems in architecture, but also proposes evaluation and classification of digitally-driven architecture through procedural- and object-oriented studies. It, furthermore, introduces methodologies of digital design, which incorporate Intelligent Computer-Based Systems proposing development of prototypical tools to support the design process.

SYSTEM-EMBEDDED INTELLIGENCE IN ARCHITECTURE

H. H. Bier/ TU Delft



SYSTEM-EMBEDDED INTELLIGENCE IN ARCHITECTURE



Proefschrift ter verkrijging van de graad van doctor aan de Technische Universiteit Delft, op gezag van de Rector Magnificus Prof. dr. ir. J. T. Fokkema, voorzitter van het College voor Promoties, is het openbaar te verdedigen op woensdag, 21 mei 2008, om 12:30 uur door Henriette Hildegard BIER, Diplom-Ingenieur TU Karlsruhe, geboren te Arad, Roemenie.

Dit proefschrift is goedgekeurd door de promotoren: Prof. ir. S. U. Barbieri en Prof. ir. K. Oosterhuis

Samenstelling promotiecommisie:

SUMMARY

Tanslation from English into Dutch by P. J. Teerds - SAMENVATTING:

Dit onderzoek richt zich op een kritische beoordeling van computergestuurde systemen in het domein van de architectuur, met betrekking tot de verwerving van aspecten van intelligentie. Digitaal aangestuurde architectonisch ontwerpprocessen worden geclassificeerd op basis van procedurele en objectgeoriënteerde studies. Verder worden methoden van digitaal ontwerpen geïntroduceerd, die gebaseerd zijn op intelligente computergestuurde systemen. Deze methoden impliceren de ontwikkeling van software-prototypen, om het ontwerpproces te ondersteunen.

Het eerste hoofdstuk geeft een overzicht over de stand van zaken in het veld van de computergestuurde instrumenten en processen. Het tweede hoofdstuk gaat in op de invloed hiervan op ontwerpconcepten en -methoden. Dit onderzoek poogt een antwoord te geven op de vraag hoe aspecten van intelligentie in ontwerpsystemen opgenomen kunnen worden en op welke manier dit het proces en het ontwerp kan beïnvloeden.

De intelligente software-prototypen, die in dit onderzoek ontwikkeld en in de praktijk en in ontwerpstudio's getest zijn, worden door de TU Delft verder ontwikkeld in ontwerp- en onderzoeksprojecten. Obser-

vaties, hypothesen en theorieën zijn in dit onderzoek methodologisch beproefd door middel van praktische experimenten in internationale workshops. De software-prototypen zijn voornamelijk ontwikkeld en getest op de TU Delft.

Aan de experimentele beoordeling en classificering van de computergestuurde systemen in relatie tot de opgenomen aspecten van intelligentie, voegt dit onderzoek een beschrijving van de ontwikkeling van prototypen toe, zowel in tekst, als ook in afbeeldingen, tabellen en diagrammen. De software-prototypen, die binnen het onderzoek ontwikkeld zijn, zijn geïmplementeerd in Java door middel van het gebruik van de open-source 'Integrated Development Environment Eclipse' in combinatie met de bibiliotheek en 'OpenGL Application Programming Interface for 3D rendering' van Processing; deze kunnen geraadpleegd worden op website www.hen-ro.eu.

Het onderzoek richtte zich op de aspecten van intelligentie, die opgenomen zijn binnen de computergestuurde instrumenten en processen en hun invloed op het architectonisch ontwerp. De nadruk lag daarbij niet op de representatie, maar op de capaciteit van de computersystemen om complexe berekeningen binnen het domein van de architectuur uit te voeren. De focus lag op complexe systemen, waarbij de complexiteit gedefinieerd wordt door de veelvoud aan interacties binnen architectonische sub-systemen, zoals de fysische en sociaaltechnische systemen.

De software-prototypen, samengevat in een systeem met de naam SpaceCustomizer [SC], functioneren als een Expert System [ES] en tonen verschillende gradaties van intelligentie, afhankelijk van de complexiteit van de specifieke taken van de prototypen. Het basisniveau van intelligentie is de intelligentie van de op automatische bewegingen gebaseerde SpaceGenerator [SG]. Verder gevorderde niveaus van intelligentie dragen aan de ene kant bij aan interactieve ruimtelijke veranderingen, door de toepassing van Artificial Intelligence [AI], met name door het gebruik van 'computer vision and sensor-actuator' technologieën. Aan de andere kant vergemakkelijken de prototypen de automatische ruimtelijke opzet, waarbij gebruik wordt gemaakt van Boolean Satisfiablity [SAT].

De ontwikkelde software-prototypen zijn specifiek gerelateerd aan geometrie, beweging en functie. De kwesties die gerelateerd zijn aan de vorm, structuur en materiaal zijn alleen in de context betrokken, waarmee de bredere context van de geometrie, beweging en functie, zoals deze onderzocht en verkent zijn, weerspiegeld wordt.
De geometrische aspecten duiden aan de ene kant op de omzetting van een op 'NURBS' gebaseerde in een 'voxelized' geometrie en aan de andere kant op de transformatie van een door NURBS oppervlakte naar een in driehoeken opgedeelde oppervlakte.

Geometrie, met name de niet-euclidische geometrie, en beweging worden waargenomen als onderling verbonden fenomenen omdat beweging speelt binnen dit onderzoek een rol als ruimte-gerenerende kracht. Verder is de functie onderling verbonden met de beweging. Beweging in een ruimte kon daarom vastgesteld worden door de met haar samenhangende functie en is opgenomen in functionele objecten. Een etenswaren-kiosk, die in dit onderzoek als casus is genomen, vraagt bijvoorbeeld om een specifiek aantal toepassingen om het op deze manier te laten functioneren. Deze toepassingen zijn echter alleen bruikbaar als de ruimte die nodig is voor de operatie ook beschikbaar is.

De ontwikkelde software-prototypen verstrekken elkaar daarom op een fundamenteel niveau de benodigde informatie. De functie beïnvloedt immers de beweging, de beweging beïnvloedt het gebruik van de functie en beide beïnvloeden op hun beurt van binnenuit de geometrie van de ruimte.

De prototypen opereren semi-automatisch. Complete automatisering van ontwerpproces, van het idee tot het daadwerkelijke bouwen, is niet toepasbaar:. Allereerst omdat dit even complexe berekeningen vergt, als het proces zelf en bovendien is de totale automatisering, als het al geïmplementeerd kan worden, onuitvoerbaar. Niet alle taken worden door computers beter opgelost dan door mensen. Computers hebben bijvoorbeeld geen 'common sense'. Daarentegen zijn ze in staat complexe berekeningen in relatief korte tijduit te voeren. Neem bijvoorbeeld Combinatorial Optimization, zoals in dit onderzoek geïmplementeerd is. Dit impliceert niet alleen de berekening van alle mogelijke, maar ook de best mogelijke ruimtelijke combinaties van functies in programmatisch ontwerp. Terwijl architecten het misschien moeilijk vinden een overzicht over de functies, hun bijbehorend volume en hun juiste positionering te houden, kunnen de functionele delen gemakkelijk computergestuurd uiteengelegd worden, in overeenstemming met de van te voren opgestelde regels een voorwaarden.

Computers zijn daarom in dit onderzoek vooral gebruikt in de domeinen waar ze het ontwerpproces verrijkten door machinematige intelligentie op te nemen. Hiermee compenseerden ze de menselijke besluitvorming op de momenten dat deze begrensd is of overvraagd wordt.

Intelligentie, die opgenomen is in computer systemen, maakt de continue interactie tussen de ontwerper en de semiautomatische intelligente ontwerpinstrumenten mogelijk; Deze instrumenten kunnen daarom alleen als ondersteuning voor de ontwerper gezien worden.

De software-prototypen, die in dit onderzoek ontwikkeld zijn, zijn getest in de casus van de eerder genoemde etenswaren kiosk. Voor deze kiosk zijn niet-euclidische ruimten ontwikkeld, gebaseerd op de patronen van beweging van een menselijk lichaam door de ruimte. Bovendien is de functioneel ruimtelijke opzet geoptimaliseerd.

Door het testen van de protoptypen zijn de meest relevante aspecten van intelligentie vastgesteld. Deze zijn opgenomen in de zoekmachine, die ontwikkeld is voor de geautomatiseerde driedimensionale indeling van functies in dubbelgekromde 'voxelized' ruimten .

De drie prototypen die in dit onderzoek ontwikkeld zijn, bevatten zowel statisch-dynamische als ook interactieve concepten. Terwijl Space-Customizer : FunctionLayouter [SC : FL] functies plaatst en SC : SpaceGenerator [SG] ruimte dynamisch genereert door de beweging van het lichaam in de ruimte te volgen, is de resulterende ruimte statisch. SC : FL en SC : SG zijn daarom alleen in het ontwerpproces gebruikt, terwijl SC : Space Interactivator [SI] niet alleen gebruikt is gedurende het ontwerpproces, maar ook als de dynamische, interactieve motor voor het controleren van interactie van gebouwcomponenten en gebouwen.

SC : SG laat het statische tweedimensionale gezichtspunt op de relatie tussen het lichaam en de architectonische ruimte, zoals breschreven in de Modulor [Le Corbusier, 1948], achter zich. Het vestigt de aandacht op de dynamische en driedimensionale beweging van een lichaam door de ruimte, die ruimte-genererende karakteristieken oplevert.

SC : SI laat de statische blik op de architectuur achter zich, door toepassing van interactieve architectuurprincipes [Oosterhuis, 2006]. Deze verschaffen een prototypisch instrument, dat de interactieve verbinding legt tussen de beweging en dubbelgekromde gebouwenveloppen.

SC : FL verbetert het voorgaande tweedimensionale functional-ruimtelijke onderzoek, door nieuwe benaderingen voor de zoektocht naar optimale oplossingen voor de plaatsing in drie dimensionale ruimten voor te stellen. SC : FL genereert, in vergelijking met andere prototypen, zoals die van Loos en Wright [Flemming U. et al, 1992] een functionele plaatsing op een vergelijkbare schaal en realistische toepasbaarheid. Loos en Wright, echter, handelen slechts met de plaatsing van functionele objecten in twee dimensies, terwijl SC : FL de plaatsing in drie dimensies voorstelt. SC : FL verkavelt de functies binnen een complexe, vrij-gevormde 'voxelized' geometrie, in plaats van in een simpele orthogonale ruimtelijke geometrie.

Toegepast op het ontwerpen van complexe driedimensionale structuren behoeft deze software prototypen echter verdere ontwikkeling. Somige delen van de prototypen zijn slechts conceptueel ontwikkeld, terwijl anderen als software al zijn geïmplementeerd. Verder de toepassing op verschillende niveaus is nog niet uitputtend onderzocht.

Toekomstperspectieven voor het inregelen van intelligentie in computergestuurde systemen in architectuur duiden daarom aan de ene kant op een intensiever onderzoek naar de ontwikkeling en het gebruik van intelligente computergestuurde systemen in architectuur, en aan de andere kant bewuste differentiatie tussen de problemen die opgelost kunnen worden bij dergelijke middelen, afhankelijk van hun natuur en schaal. Verder vereist het inregelen van de intelligentie in computergestuurde systemen in architectuur, zoals uit het onderzoek blijkt, de samenwerking tussen architecten en computerdeskundigen op een vergelijkbare manier als de manier waarop architecten samenwerken met constructeurs. Dit zal plaatsvinden onder de voorwaarde dat architecten niet slechts inzicht in het ontwerpprobleem hebben, maar ook een basiskennis hebben van de automatiseringsproblematiek, die relevant is voor het oplossen van de ontwerp- en constructieprobleem in samenwerking met de computerdeskundigen.

PREFACE

Founded on the imperative to understand, evaluate and consciously decide about the use of digital media in architecture this research not only aims to analyze and critically assess computer-based systems in architecture, but also proposes evaluation and classification of digital-ly-driven architecture through procedural- and object-oriented studies. It, furthermore, introduces methodologies of digital design, which in-corporate intelligent computer-based systems proposing develop-mentof prototypical tools to support the design process.

Vast references included in this research attempt to make this work-self-contained: However, material which is too lengthy to explain, but-which has been well documented elsewhere, has been either onlyshortly mentioned or completely left out.

While the first chapter enumerates state-of-the-art computer-based-*tools and processes*, the second chapter engages in interpretative-work dealing with the influence of those computer-based tools andprocesses on design *concepts and methods*. The third chapter in-tro-duces *innovative* work describing prototypical tools - developed withinthis research - incorporating aspects of machinic intelligence.

This research atempts to answer questions regarding how aspects of *intelligence* are incorporated in design systems and how these influ-ence the design process and the design: Generative Design, for instance, has been focus of current design research and practice largely due to the phenomenon of *emergence* explored within self-organizing-systems, generative grammars and evolutionary techniques.

In this context, system-embedded intelligence has often been reducedto the mechanics of working with these systems.

This research, in response, not only critically reveals what these techniques offer architectural design, but also addresses challenges intheir application and development. Its relevance as reference for developing an understanding for computer-based systems in relation totheir incorporated aspects of intelligence has been meanwhile confirmed in projects implemented internationally.

Furthermore, intelligent software-prototypes developed within this research have been tested in practice and in design studios and will befurther developed in prospective design and research projects undertaken at TU Delft.

Methodologically seen, observations, assumptions and theories havebeen verified in practical experiments implemented in internationalworkshops, whereas, software-prototypes have been mostly developed and tested at TU Delft.

In addition to the experimental assessment and classification of computer-based systems with respect to their incorporated aspects of intelligence, this research describes software-prototypes development. This is presented in textual descriptions incorporating images, tables, and diagrams as well as a self-starting presentation incorporatingmovies delivered on an additional DVD.

Implemented in Java using the open source Integrated Development-Environment Eclipse in combination with open source libraries and OpenGL Application Programming Interface for 3D rendering of the programming environment Processing, software-applications developed within this research are not included as machine-readable executables on this DVD, but will be published on the website www.henro.eu.

This research represents the result of 3 3/4 years intensive work supported and in part implemented by experts in programming such as R.Schmehl, J. Galle, H. van Maaren, M. Heule, A. de Jong, N. Brouwers, and G. van der Hoorn, T. Nakata, K. Steinfeld, D. Rutten and A. Feldman.

In addition, this research has benefited from the conceptual input ofcollaborators such as K. de Bodt and O. Vazquez- Ruano, as well as the constructive critique of fellow researchers such as L. Sass, R. Oxman, and N. Biloria.

It also has benefited from the support of A. Graafland, L. van Duin, M. Schoonderbeeck, F. Geerts, P. J. Teerds, A. Kilian, M. Male-Alemany,

P. Marone, and D. Spencer. Last but not least, this research could not have been implemented without the support and critical input of U. Barbieri, K. Oosterhuis, and T. Knight as well as the active participation of students from TU Delft, TU Innsbruck, TU Karlsruhe, Roma Tre, and HIAS Antwerpen.

FIGURES

2004] lead by H. Bier at TU Innsbruck and TU Karlsruhe.

F2.03 - Image showing a student project from the master-class on Digital Design [2003-2004] lead by H. Bier at TU Innsbruck and TU Karlsruhe.

F2.04 - Denari's project THE WALL described in Gyroscopic Horizons - Prototypical Buildings and Other Works [1999].

F2.05 - Denari's project THE WALL described in Gyroscopic Horizons - Prototypical Buildings and Other Works [1999].

F2.06 - Denari's project THE WALL and its replica developed by a student participating in the master-class on Digital Design [2003-2004] lead by H. Bier at TU Innsbruck and TU Karlsruhe.

F2.07 - Mutant - THE KIOSK - developed by students from TU Vienna participating in the workshop on Digital Design [2004] lead by H. Bier at Roma Tre.

F2.08 - Physical model of the mutant developed by students from TU Vienna participating in the workshop on Digital Design [2004] lead by H. Bier at Roma Tre.

F2.09 - Image showing student projects from ADSL-workshop [2005] lead by H. Bier, O. Vazquez-Ruano, and K. de Bodt.

F2.10 - Image showing student project from ADSL-workshop [2005] lead by H. Bier, O. Vazquez-Ruano, and K. de Bodt.

F2.11 - Translation diagram developed by H. Bier and O. Vazquez-Ruano for the ADSL-workshop [2005].

F2.12 - Sound sample from Audacity filtered through three different Photoshop-filters.

F2.13 - Image shows student project from ADSL-workshop [2005] lead by H. Bier, O. Vazquez-Ruano, and K. de Bodt.

T2.01 - Table originating from The Metapolis Dictionary of Advanced Architecture by Gausa, M. et al. [2003] modified by H. Bier.

F2.14 - Muscle-Tower project [2004] developed by Hyperbody available online from http://www.protospace.bk.tudelft.nl.

F2.15 - Kaisersrot project developed by CAAD - ETHZ available online http://wiki.arch.ethz.ch/twiki/bin/view/Kaisersrot/ParzellenP.

F2.16 - Images from Samitaur by E. O. Moss available online from http://www.ericowenmoss.com/index.php?/projects/project/samitaur.

F3.16 - Image showing the interface developed for SC : IS [2006] by K. de Bodt and J. Galle.

F3.17 - Image showing principle of adjustment of a double-curved *slave*-surface to a computer-generated *master*-surface [H. Bier, 2005].

F3.18 - Images showing Thon's procedure of voxelization of polygonal objects [2004].

F3.19 - Composite showing the data-connectivity envisioned for SC : MS : AS : VG.

F3.20 - Image showing the voxelization-model developed by H. Bier and K. Steinfeld for SC : MS : AS : VG [2006].

F3.21 - Voxelization for polygonal objects by Thon et. al. [2004] using a *quad-tree* partitioning of the triangle-mesh in order to speed up the ray-casting procedure.

F3.22 - Virtools interface built by Hyperbody-students for the BR-project [2006].

F3.23 - Dynamically Interactive Virtools model showing BR-project lead by H. Bier and K. Oosterhuis [2006].

F3.24 - Change of movement of functional units within different building volumes as simulated by Hyperbody-students in BR [2006].

F3.25 - NURBS-based geometries tested Hyperbody-students for the BR-project [2006].

F3.26 - Data-exchange system developed by Hyperbody-students for the BR-project [2006].

F3.27 - Voxelization resolution ranges from 5-90 cm enabling an almost accurate representation of the curved geometry - Figure shows voxelization resolution 30/30/30 and 15/15/15 cm developed by H. Bier and R. Schmehl [2007].

F3.28 - Figure shows one of the four optimal layouts for 90/90/90 voxelization resolution as developed by H. Bier, A. de Jong, N. Brouwers, G. van der Hoorn, M. Heule, and H. van Maaren [2007].

F3.29 - Figure shows best possible layout solution for 30/30/30 voxelization resolution as developed by H. Bier, A. de Jong, N. Brouwers, G. van der Hoorn, M. Heule, and H. van Maaren [2007].

D3.01 - Diagram: SC incorporates three sub-tools MS, IS, and FL, which incorporate sub-tools such as AS, which in turn incorporate sub-sub-tools such as VS, AV, AVM and GNV.

INTRODUCTION

Four decades ago Chermayeff and Alexander [1963] claimed that architectural design can be automated with computers; follow-up research proved automation in design to be inefficient, because it required extensive input, and ineffective, because the output was diagrammatic: The concept of automated design was, therefore, replaced with Computer-Aided Design [CAD]. It seemed that rather simple not complex design tasks could be solved with computer techniques [Tzonis, 1990].

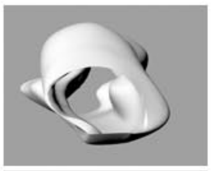
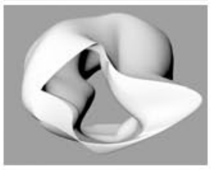
Since then, CAD-systems have gradually been developing towards incorporating aspects of knowledge about the designed object: Ranging from autonomous design tools to design support systems, *Intelligent Computer-Aided Design [IntCAD]* systems have replaced meanwhile CAD-systems [Brown, 1998]. N0.01

Considering that scientists might be able today to develop a computer model representing the *whole* universe but this computer model would have a sophistication corresponding to the complexity of the universe itself [Zuse, 1967; Schmidhuber, 1997] and knowing that according to the concept of computational equivalence formulated by Wolfram [2002] any process natural or artificial can be viewed as a computation of equivalent sophistication, this research puts forward the argument that *complete* automation in architecture is for the time being not relevant because it requires computer models of equivalent sophistication. N0.02

If the computation takes as much effort as the real process of design, what do architects gain from computation except additional insights, which they can not derive from practice?

Architects gain from computation - in addition to insights - support in the design process not only in form-finding, but also in functional, mechanical, structural, constructive problem-solving.

The shift from mechanical to digital, forces architects to reposition themselves: [1] They do not produce *merely* drawings but produce digital data, which becomes single source of design and fabrication [Kolarevic, 2003]. [2] Furthermore, digital systems not only inform the design process but also the fabrication process, challenging the MODERNIST concept of standardization, introducing the concept of mass-customization, which implies, as Slessor [1979] put it, that 'uniqness is now as easy and economic to achieve as repetition'. And [3] *complete* automation from idea to building might not be relevant at the time being, because it requires computation as complex as the process itself, but, semi-automation based on *system-embedded intelligence*, which addresses possibilities for architecture based on algorithmic techniques, is of great relevance. N0.03

This research, therefore, explores system-embedded intelligence, which informs the design and fabrication process, based on the premise that only at the moment it incorporates *intelligence,* the computer becomes more than a tool, it becomes a *reasoning machine*. Purpose of this research in system-embedded intelligence is critical assessment and evaluation of state-of-the-art computer-based methods and tools in architectural design with respect to their incorporated aspects of machinic intelligence, and furthermore, development of prototypical tools, which improve the design process by incorporating *reasoning* capabilities.

In this context, machinic intelligence refers only in part to *Artificial Intelligence [AI]* as described in Computer Science. It also denotes computer-based concepts and systems that utilize what appear to be reasoning capabilities: These imply computations employing strategies which exploit *heuristic* knowledge [Brown and Chandrasekaran, 1989] described as either embedded in existing tools and procedures or newly developed and tested within this research. N0.04 N0.05

While the first two chapters focus on state-of-the-art computer-based *Tools and Processes*, and their influence on design *Concepts and Methods*, the third chapter describes *Visions and Perspectives* incorporating frontier research in system-embedded intelligence.

Prototypical tools developed within this research - described in the third chapter - operate as an *Expert System* by providing analysis of specific design problems, and, depending on their programming, generating alternative designs or *recommending* corrections. N0.06

Even though Expert Systems are not new, the system developed within this research represents an INNOVATION from the point of view of its components, their linkage to each other and their application to architectural design. The proposed system, named SpaceCustomizer [SC], is an experimental software platform able to execute semi-automated tasks such as the development of [1] Motion Spaces based on *NURBS*-geometries, [2] Interactive Spaces, and [3] Functional Layouts. N0.07

[1] Motion Spaces [MS] are double-curved 3D-spaces generated by tracing the movement of the human body in space, whereas, the motion map defines boundaries of the volume within which architecture can emerge. The volumetrical outlines of the body in motion establish an initial framework to develop Motion Spaces employing movement studies based on ergonomics.

The initial fundamental question in the development of MS has been: HOW to control designs based on NURBS-geometries? On the one hand it is easy to manipulate NURBS-surfaces by pulling control points, on the other hand, the question is how to control their manipulation, which rules and design methodologies can be developed to control designs based on NURBS-geometries?

If in this context can be talked about a paradigm shift based on the influence of digital technologies, than this shift can be described in the methodology: In opposition to modular, repetitive architecture developed by using grids and proportions based on functional and formal rules, curvilinear architecture is being developed within this research by generating space through following the movement of the body in space based on ergonomic principles.

SC : SG [SpaceGenerator] generates space following the movement of the body in space and can be seen as the Modulor [Le Corbusier, 1948] of the Digital Age since it establishes relationships between the human body and the architectural space: As a system of proportions Modulor uses measures of the human body in architecture by partitioning it in modules according to the Golden Section and to two Fibonacci Series. It puts, basically, man as measure of architectural spaces, which SC : SG does as well in a more drastic manner, since it generates 3D-space through following the movement of the body in space based on ergonomic principles; While Modulor applies a 2D proportioning system, SC : SG employs a 3D DYNAMIC space-generating system.

[2] Interactive Spaces [IS] are double-curved spaces generated interactively by the movement of the body in space: The input - movement - is being electronically processed in such a way that the output represents a continuous, real-time modification of the space. For this purpose an on-site-built InterFace employing sensor-actuator technology enables translation of the recorded movement into spatial configurations, whereas the *InterAction* between the body and the N0.08

architectural space gives insight into, how the human body shapes space.

As a software prototype, SC : SI [SpaceInteractivator] is different from SC : SG not only from the point of view of its interactive nature but also from the point of view of its application: While SG is exclusively applied in the design process to generate movement patterns, which influence the design, SI is applied in the design process as well as in the implementation of dynamic building components and buildings as interactive engine controlling dynamics on the software level.

[3] Functional Layouts [FL] are 3D space-configurations generated by applying top-down and bottom-up problem-solving procedures to optimally place functions in 3D-space.

SC : FL [FunctionLayouter] generates functional layouts exhaustively and enables the designer to develop and consider more alternatives than by means of conventional sketching methods mainly because architectural space-planning is highly combinatorial, and therefore, difficult to conceive exhaustively by human search-means.

Instead of one, SC : FL generates multiple designs and enables informed choices by departing from a *singular*-design principle, that represents a potentially prejudiced position of the *master*-designer. SC : FL generates ALL functional designs within the spectrum of an *optimal* solutions field.

Considering intelligence, more specifically reasoning, an ability to operate with concepts and form conclusions based on facts, the developed software-prototypes exhibit different degrees of intelligence depending on the complexity of their particular tasks, the most basic
degree of intelligence being the intelligence of an automated move- N0.09
ment-based space-generator. N0.10

More advanced degrees of intelligence enable on the one hand inter- N0.11
active spatial transformations by employing AI, namely, *computer vision* and *sensor-actuator technologies*; on the other hand, they facilitate spatial layout optimization by means of *Boolean Satisfiability [SAT]* used, inter alia, in Automated Reasoning.

While these software-prototypes address exclusively issues related to geometry, function and movement, they are interrelated as function influences movement, movement influences the use of function, and both influence the geometry of space.

Obviously, practice-relevant, these software prototypes generate automatically alternative designs confirming the assumption that semi-automation based on system-embedded intelligence is of great importance: They enhance the design process by incorporating machinic intelligence, compensating where human decision-making might be limited or overextended.

Furthermore, their placing within the framework of contemporary computer-based systems with respect to their incorporated aspects of intelligence enables conscious decision about their use in architecture as well as informed envisioning of their further development, which is relevant for research as well as teaching.

NOTES: Notes to this section explain concepts and notions difficult to extract from context.

N0.01 - Intelligent Computer-Aided Design is a term used by Brown [1993] to describe computer-aided design systems incorporating aspects of knowledge about the designed object.

N0.02 - Automation implies the use of computers to control industrial machines and processes.

N0.03 - System-Embedded Intelligence denotes intelligence incorporated in a computer-based system such as intelligence exhibited by a computer program, which is demonstrably capable of *intelligent* action and/or interaction, i.e. automated functional 3D-layout, interactive spatial responsiveness, etc.

N0.04 - Artificial Intelligence as described by McCarthy [1956] implies the use of intelligent machines for task automation requiring intelligent behavior. In general, this refers to actions and interactions regarded as intelligent when carried out by a human being.

N0.05 - Heuristic is a methodology of approaching problems and formulating solutions to them. Polya [1945] specifically describes heuristics such as: [1] Simplify and abstract: Draw a diagram of the problem, [2] Work backward: Assume a solution and see what can be derived from that, [3] Find a concrete example for an abstract problem, etc.

N0.06 - Expert Systems [ES] in architecture are - according to Brown [1993] - design systems able to check design decisions for conformance with standards and guidelines, extract features, select materials, outline deficiencies and suggest corrections, as well as evaluate the manufacturability or constructability of the design.

ES are, basically, programs employing computations that underlay intelligent behavior implying the use of general search strategies which exploit qualitative heuristic knowledge about the problem domain [Brown and Chandrasekaran, 1989]. ES, therefore, can be seen as systems operating with knowledge of experts in whole or in part.

N0.07 - NURBS - non-uniform, rational B-splines - are mathematical representations for curves and curved surfaces.

N0.08 - Interactivity implies a two-way action-reaction exchange, which is implemented in architecture by means of sensor-actuator technologies.

N0.09 - Computer vision implies technologies able to extract information from images and multi-dimensional representations similarly or complimentarily to biological vision.

N0.10 - Sensor-Actuator Technologies imply devices receiving and responding to signals such as sound, image, etc.

N0.11 - SAT refers to Boolean Satisfiability, which implies determining if the variables of a given *formula* can be assigned in such a way that the formula evaluates to a Boolean-value: TRUE or FALSE [Een and Sorensson, 2003].

BIBLIOGRAPHICAL REMARKS: Research in computer-based systems in architecture incorporates a wide range of topics and sub-topics addressing issues of representation, simulation and generation of architecture.

Slessor addressed [1979] the potential of computer-based techniques to generate economically uniqness in architecture, whereas an overview on state-of-the-art concepts and methods on computer-based digital design and fabrication in architecture has been compiled two decades later by Kolarevic [2003].

Issues of incorporated knowledge in the designed object and automation were addressed by Tzonis [1993] and Brown [1998]. Whereas issues of computability, have been addressed by Schmidhuber [1997] who reinterpreted Zuse´s assumption [1967] that the history of the universe is computable: While Zuse suggested that the computer could be a Cellular Automaton, Schmidhuber proposed an universal Turing-machine.

Wolfram [2002] goes back to Zuse's assumption and undertakes research in Cellular Automata in order to understand and model complexity.

Complexity and complex processes in architecture have been studied by Alexander [1963] and have been implemented in space allocation algorithms by Seehof and Evans [1967], Mitchell [1974], Simpson [1980], Baykan and Fox [1997], Michalek et al. [2002].

1. TOOLS AND PROCESSES

Digital media change the relationship between architecture and its means of production: Digital fabrication allows production of complex non-Euclidean geometries, and architecture, therefore, is no longer defined by modernist principles of standardization and repetition.

Not only that the design process merges seamlessly into fabrication - in an iterative process of transferring data from a 3D modeling software to a 3D-printer or a CNC - Computer Numerically Controlled - machine, but double-curved geometries become as feasible as planar geometries [Slessor, 1997].

In an attempt to classify and evaluate state-of-the-art computer-based devices utilized in architecture with respect to their incorporated aspects of machinic intelligence, this chapter describes Digital Design and Fabrication:

By departing from Kolarevic's [2003] summary on digital technologies and their use in architecture, this chapter proposes a revised classification and introduces additional computer-based tools and processes.

1.1 Digital Design

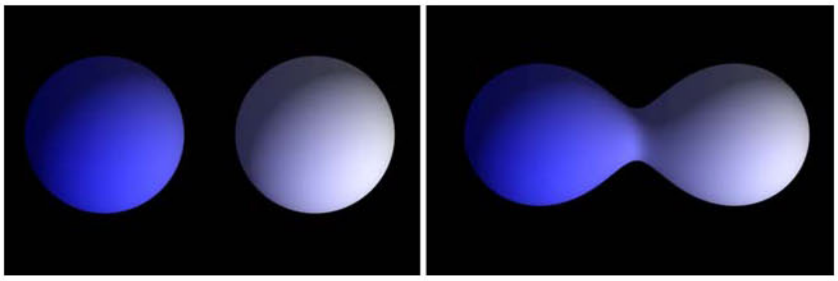
1.1.1 Computational Tools: Topological Spaces, Isomorphical Surfaces, Motion Kinematics and Dynamics, Parametric and Generative Designs

According to Kolarevic [2000] Digital Design implies computational devices such as: Topological Spaces, Isomorphical Surfaces, Motion Kinematics and Dynamics, Key Shape Animation, Parametric Design and Genetic Algorithms.

Considering Genetic Algorithms as being part of Generative Design, which also incorporates Self-organizing Systems and Generative Grammars, this research proposes a revised classification.

[1] Topological Spaces: While classical geometry deals with faces, sides and vertices looking at isolated elements, TOPOLOGY deals with the whole, focusing on the connection between those elements. It, basically, studies properties of objects independent of their size and form [Gausa and Soriano, 2003] and incorporates, inter alia, the geometry of continuous curves and surfaces mathematically described as NURBS: Non-Uniform Rational BSplines.

NURBS-geometries introduce double-curved surfaces in architecture allowing not only for design but also for fabrication of curvilinear geometries changing fundamentally our understanding for architecture: Architecture is not anymore about planar elements developed in a repetitive, modular way but it is rather about complex, digitally designed and fabricated *free*-form geometries [Kolarevic, 2003]. N1.01



F1.01

[2] Isomorphic Surfaces imply blobs or metaballs, which are amorphous objects with internal forces due to mass attraction: When two metaballs are close enough to affect one another they connect gener-

ating an isomorphic surface. The figure shows two spheres having equal radii and equal fields of influence, whereas more complex configurations are generated when metaballs of different size are interacting with each other - as for instance in Franken's BMW Pavilion.

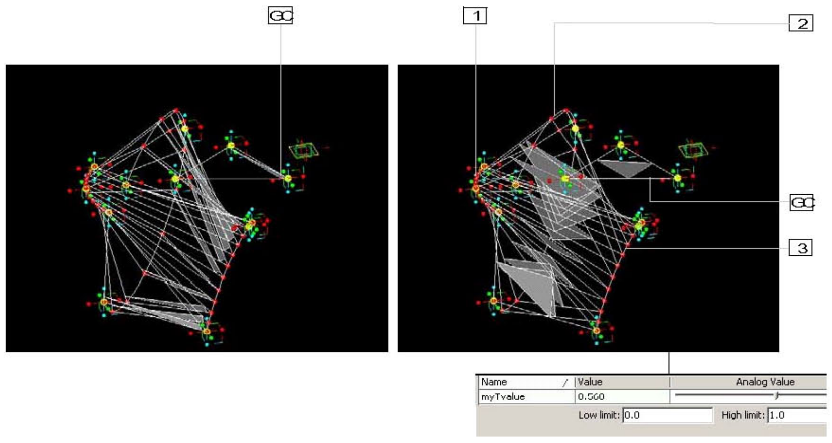
[3] Motion Kinematics and Dynamics: Animate Architectures as described by Lynn [1998] rely on motion-based modeling techniques such as forward and inverse kinematics and dynamics to generate initial architectural forms. While kinematics studies motion without consideration given to mass or external forces, dynamics takes into consideration physical properties such as mass, elasticity and physical forces such as gravity, inertia.

[4] Key Shape Animation generates designs by deforming the modeling space, while the intermediate states of the transformation process are computed accordingly. In this context, morphing as a process of blending dissimilar forms in order to generate hybrid forms, represents an additional deformation and transformation technique involving a time-based design strategy.

[5] Parametric Design implies development of *parametric models* by establishing dimensional and geometric *constraints* between the parts of the model. These models are, basically, topological descriptions specifying the parts that constitute them and the relations between those parts.

In this context, parametric modification can be accomplished with spread-sheets, scripts, or by manually changing parameters such as dimensions, relative distances, and angles in the digital model. These are linked to geometry in such a way that when values for parameters change, the part associated with them changes accordingly.

EXAMPLE: A 3D-model developed with a parametric modeling software, Generative Components, exhibits an apparent complexity - Its geometrical structure changes according to the movement of the slider, which corresponds to changes in the numerical value from 130 to 460 to 880.



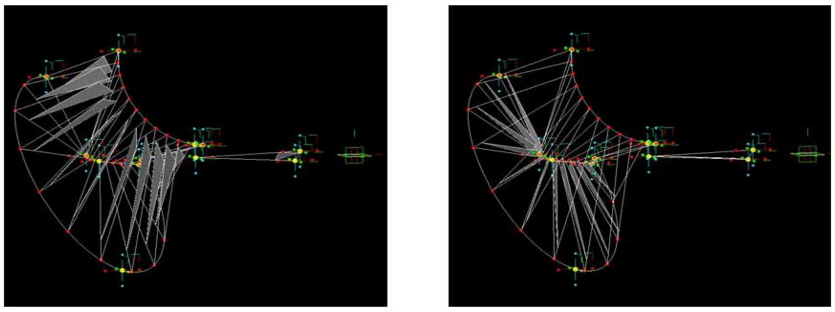
Name	Value	Analog Value
myTvalue	0.560	

Low limit: 0.0 High limit: 1.0

F1.02

Even though the structure seems to be complex it has been generated by modeling three BSplines in 3D-space and by generating a so called Generative Component [GC], which is - in this case - a triangle within a triangle.

The curves have been divided in regular intervals by points and are linked with the Generative Component in a way that between the points of the three curves, similar to their unique context, *adaptive* components are generated. The changing value determines the shape of the inner triangle by moving the corners along the lines of the outer triangle.



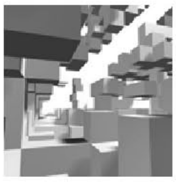
F1.03

That means, there is a relationship established between the components of this model:

Changing a curve, for instance, implies that this change will be propagated to all parts of the system. There are, basically, two connections, one of them between GC and the curves, and the other one between the corners of the inner triangle and the outer triangle, which lets the corners slide along the curves according to a numerical input.

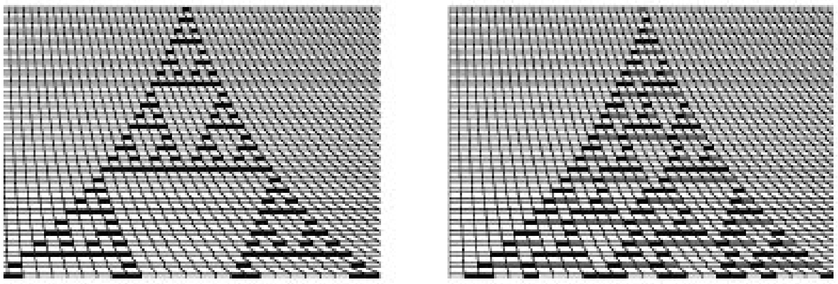
[6] Generative Design incorporates computational tools such as Self-organizing Systems, Genetic Algorithms, and Generative Grammars.

Self-organizing Systems based on Cellular Automata and Swarm Intelligence have been employed in architecture under the premise that conceptualization has been shifting - as Kuhn [1996] describes it - from object-generation to development of interacting components, systems and processes, which in turn generate objects.



Swarm Intelligence is addressed within this research in section 3.3.1.1. This section, therefore, only introduces Cellular Automata:

Cellular Automata [CA] are discrete models consisting of grids of regular cells, wherein each cell assumes one state from a number of possible states. Rules determine the state of the next cell depending on its immediate neighbors' state [Wolfram, 2002]. Even though this underlaying structure is simple, CAs are able to perform sophisticated computations: Their application in architectural design ranges from ornamentation producing visual pattern variations to automated volumetric building generation [Coates, 1996].


F1.04

Genetic Algorithms [GA] use mechanisms inspired by biological evolution and are, basically, search and optimization tools: Once the objective is encoded in a structure called *genome*, the genetic algorithm creates a population of genomes then applies crossover and mutation to the individuals in the population in order to generate new individuals.

These GA-techniques are applied in architectural design not only to generate functional diagrams [Elezkurtaj and Franck, 2002] but also to generate designs [Loomis, 2003].

Looking for generative systems of organization, architects re-discover principles of organization in nature and their mathematical correspondence: The sequence 1, 2, 3, 5, 8, 13,... is known as the Fibonacci series, in which each number is the sum of the previous two. The Fibonacci numbers can be observed in the arrangement of the branches or leaves around the stem of many plants.

This knowledge has been incorporated in Lindenmayer-systems, also called L-systems, which are a mathematical formalism proposed by the biologist Lindenmayer [1968] as a foundation for an axiomatic theory of biological development.

L-systems have found application in computer graphics including generation of fractals and modeling of plant-growth: Chu's [1999] approach to Digital Design is based on applying the generative rules of the L-system to architecture as shown in the project X Phylum.

Generative Grammars, in particular Shape Grammars [Stiny, 1975] have been also applied in architecture and design: They consist of shapes, shape-rules and transformation-rules which enable generation of designs as tested within this research and described in section 2.2.2.

Architects are using computational tools in an unconstrained way applying them whenever helpful in the design process: Lynn's design process in the Cardiff Bay Opera House project, for instance, relies, initially, on fractal organization.

This spatial organization is subsequently superimposed with a functional organization, where the volumes are re-arranged according to functional requirements. There is not one organizational concept determining the design but many.

In contrast to Lynn, Franken's BMW-pavilion is based exclusively on isomorphic surfaces: Being designed and fabricated exclusively by digital means, this project can ultimately be seen as a representative example for computer-based architecture.

However, the exclusive use of digital techniques from idea to materialization seems to be rather exceptional. Instead there is a combination of non-digital and digital tools and procedures being used in architecture:

Gehry's design, for instance, implies a sculptural approach; only its implementation in built architecture incorporates computational techniques. Gehry's initial hand-made physical model is digitized in a 3D scanning process. The obtained digital data is subsequently transferred to 3D modeling programs, where the actual process of analysis and optimization is taking place. The form-finding process is, therefore, not based on computational tools. Only the subsequent processing involves reverse engineering and parametric definition in order to allow rationalization of geometry and engineering of structure.

1.1 Digital Design

1.1.2 Computational Processes: Collaborative Systems, Building Information Modeling

Computer programs influence design, whereas their use in different phases of the design process establishes a MODUS OPERANDI new to architecture:

Architects generate digital information, which can be used in fabrication and construction to directly drive the computer-controlled machines, and therefore, produce building components by *CAD-CAM* . N1.02

CAD-CAM processes establish a direct link between conception and production and are supported by Collaborative Systems [CS] which allow team members such as architects, engineers and manufacturers to work on projects at any place, while the files are available from a common database on the Internet.

CS rely, basically, on the concept of shared *virtual space*, wherein collaborating participants work synchronously and/or asynchronously on the same project. *Intelligent* engines incorporated in this shared space, support activities of all participants by providing automated services, such as detecting design changes and automatically notifying the participants about them [Castle and Pollalis, 1999].

In this context, Building Information Modeling [BIM] as a digital representation and communication model for the building process, facilitates exchange of information in digital format: It connects architecture, engineering, and construction industry by incorporating data about geometry and spatial relationships, quantities and qualities of building components, number and nature of systems, etc.

So far, the described computational processes incorporate aspects of *intelligence* in the way systems store and share incorporated knowledge about the designed object:

The knowledge itself is generated with help of computational devices. These reflect the ability of digital media to frame questions and interrogate issues pertaining to conceptualization, representation and simulation of architectural design. However, with exception of Parametric and Generative Design, they do not incorporate aspects of knowledge about the designed object going beyond its representation.

1.2 Digital Fabrication

1.2.1 Computer-Numerically Controlled Processes: CAD-CAM

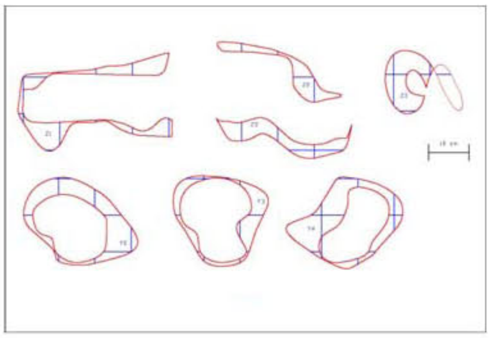
Digital Fabrication refers to Computer Numerically Controlled [CNC] technologies implying data transfer from a 3D modeling or a 2D drafting program to a CNC-machine. These allow semi-automated production of small-scale models and full-scale building components directly from 2 - 3D digital models.

As described by Kolarevic [2004] and others, Computer-Aided Manufacturing [CAM] incorporates three main CNC-fabrication techniques: Subtractive, Additive, and Formative Fabrication.

[1] Subtractive Fabrication refers to material removal processes as for example milling and cutting. In both cases the geometry is translated into a CNC-program, which operates the milling device.

CNC-milling has been applied, for instance, in the fabrication of molds for the double-curved glass panels of Franken‘s BMW Pavilion, whereas the aluminum frames have been cut using computer-controlled water-jet technology.

As tested in a workshop [2005], where students from the university in Delft employed a laser-cutter to build a NURBS-based model, CNC-cutting technologies imply a basic level of machinic intelligence, since the machine *recognizes* cutting lines and scoring lines from a 2D-file and cuts and scores the sections, accordingly.



F1.05

[2] In opposition to removal of material, Additive Fabrication involves processes of layer-by-layer adding of material: 3D-Printing, as one of these techniques, uses ceramic powder and glue. The physical model is generated in a process of selective addition of glue layers to the powder-mass.

As a Rapid Prototyping technique, 3D-Printing is used to generate small-scale models not building components. More recently, experiments applying a similar concept of additive fabrication based on *sprayed* concrete technologies were introduced in the fabrication of building-components and buildings [Khoshnevis and Bekey, 2002].

The *contour crafting* prototype developed by Khoshnevis and Bekey is, basically, an oversized 3D-Printing device enabling automated layered fabrication of buildings and building components.

[3] Formative Fabrication implies shaping processes as, for example, straining metal beyond the elastic limit by heating and bending it. Formative Fabrication has been used in Gehry's recent projects such as the EXPERIENCE MUSIC PROJECT.

CNC-fabrication techniques, therefore, enable architects not only to generate and control complex geometries by establishing a feedback mechanism between conception and production through Rapid Prototyping but also to directly produce and construct building components.

EXAMPLE: ONL [Oosterhuis_Lenard] has a particular digital design and fabrication method, which implies scripting and programming:

The structure, for instance, of the double curved surface is generated not by modeling in a 3D-software but by scripting, which means the designer writes, a program which generates automatically the 3D-structure. This digital model represents not only a 3D-model of the building but incorporates all instructions needed in order to control the cutting machine, for instance.

ONL-projects rely mainly on three computational devices [Oosterhuis et al., 2004]: The first one implies the geometry of continuous curves and surfaces mathematically described as NURBS: Non-Uniform Rational BSplines. The second is based on parametric design, and the third implies scripting and programming.

[1] BSplines establish the geometry of the building in the project Acoustic Barrier: They follow the trajectory of the highway defining within the sequence of a relative smooth curve a bulge: the Cockpit.

In addition to NURBS-geometry, Acoustic Barrier and Cockpit also make use of parametric design and scripting as computational tools:

[2] Parametric design refers to parametric definitions such as $1 = x^2/a^2 + y^2/b^2$ for instance, to describe a 2D-curve. Each time a parameter changes, the curve regenerates to reflect the new value. The parametric model represents, therefore, the setup of a *meta*-design allowing for a reconfigurable design.

[3] Scripting and programming refer to the process of writing a program consisting of a set of coded instructions that enables the computer to perform a desired sequence of operations. In the case of Acoustic Barrier the geometry is based on several MAX-Script routines:

[3.1] The first script loads the DWG-files containing point clouds. It builds the axes of the steel profiles, that form the structure and projects the planar surfaces generated between the points, defining shape and position of the glass panels.

[3.2] The second script generates a detailed 3D-model of the steel-glass structure.

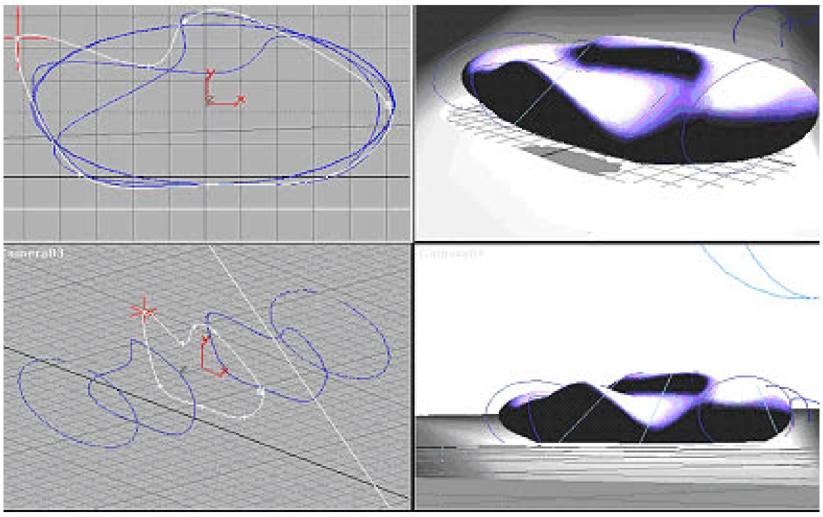
[3.3] The third script verifies the 3D-model resulting from the previous two scripts, applies constraints such as maximum tolerance, for instance, and effectively replaces deficient parts. Finally, the data generated by scripting is directly transferred to the steel and the glass manufacturer for further processing such as laser cutting.

Beyond generating small-scale models by means of Rapid Prototyping, ONL projects, as mentioned before, make use of CNC-technologies such as laser-cutting to fabricate building components.

Not only that ONL establishes direct communication and data-transfer with the manufacturers by accessing data from a common data base on the Internet but the digital data from the 3D-model controls the CNC-machine. Intermediate stages such as the production of 2D-drawings, for instance, are eliminated.

While digital fabrication allows for production based on local variation and differentiation, collaborative design and construction via Internet enable participants to communicate and collaborate in developing individual designs and produce *customized* buildings.

The process of collaborative design and construction proves to be effective because of the direct link established between client, architect and manufacturer: It skips several iterations such as generation of 2D drawings - construction documents - for the industrial production, which becomes unnecessary.



F1.06

In the design process, ONL not only makes use of NURBS-geometry and parametric design but introduces scripting and programming in the design process of complex shapes such as double-curved surfaces: This allows not only for automated 3D modeling of building components according to the script but also allows for automated generation of quantitative and qualitative data to control the CNC-fabrication of the components.

CONSTRUCTION STRATEGIES: In response to requirements of the double-curved surface ONL employs four specific construction strategies:

[1] Structural Skin, as a construction principle, implies skin geometries such as double-curved surfaces capable to serve as structure, therefore capable of self-support. The constructive concept of structural skin opposes the modernist concept of separation of skin and structure.

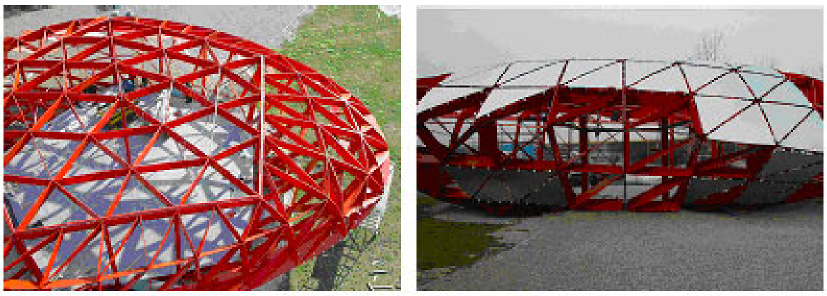
[2] Polygonal Tessellation: Implies the transformation of the surface from curved to facetted enabling subsequent extraction of 2D, planar surfaces from those double-curved surfaces. In general, the transformation of a NURBS-surface into a facetted surface refers to automat-

ted tessellation processes based on surface subdivision algorithms providing several computer generated tessellation alternatives.

ONL develops a different strategy: It intersects in a first step the NURBS-surface with a generic structure in order to create point clouds. These points resulting from the intersection between the equidistant lines defining a pattern of identical triangles and the NURBS-surface establish a spatial matrix. From these points the structure and the envelope are developed by means of scripting, whereas the point cloud establishes general conditions for the development of a generic detail.

[3] Generic Detail: The constructive concept of the Generic Detail is based on the premise that there is no separation between building components such as vertical and horizontal elements - walls and floors - as known in modernist architecture. From the Generic Detail developed according to the requirements of the whole, specialized details are developed according to local rules.

For instance, the triangulation of the skin of the project WEB of North-Holland corresponds to the concept of generic detail, implying that every single hylite-aluminum panel although triangular has an individual size and shape. Even though the elements are the same, since they are all triangles, they are all different and therefore *specialized*.


F1.07

[4] Composite Material: Produced by combining different constituent materials, the composite displays properties of the different components representing an improvement in performance: The 2 mm thick hylite-aluminum panels employed in the project WEB of North-Holland consist of a polypropylene core placed between two ultra thin layers of aluminum. By fastening the triangle at the mid points of three sides and subsequently fastening of the edges of the triangle with omega shaped fasteners the panels are able to follow the geometry of the double-curved surface.

CONCLUSION: Architects develop increasingly their own tools - They customize their design tools by scripting and programming with the ultimate objective to develop a computer model containing all qualitative and quantitative data necessary for designing and producing a building.

While the single source of information is the digital model, architects establish increasingly platforms of collaborative production with numerous parties in the building process via Internet.

Architects have been, however, slow in implementing computer-based technologies: While the car industry, for instance, has been applying these technologies since the 1970s, architecturs only recently started to look into the use of computer-based technologies in design and construction.

CONTRIBUTORS: To this section have contributed K. Oosterhuis, S. Boer, and K. Aalbers.

NOTES: Notes to this section explain concepts and notions difficult to extract from context.

N1.01 - Free-form surfaces are used in CAD to describe double-curved surfaces described mathematically as NURBS; however, there are other methods such as Gorden- and Coon-surfaces [v. Dam et al., 1996] to decribe free-forms.

N1.02 - CAD-CAM processes imply the direct transfer of data developed by means of Computer-Aided Design [CAD] to Computer-Aided Manufacturing [CAM] firms in order to directly produce building components from digital data.

BIBLIOGRAPHICAL REMARKS: Computer-based digital design and fabrication tools in architecture - with exception of Generative Design - have been outlined, inter alia, by Kolarevic [2003]. His overview on state-of-the-art computer-based systems in architecture served within this research as framework for evaluation and further development.

As co-author of Architecture in the Digital Age: Design and Manufacturing [2003] Kolarevic describes recent developments in the application of digital design and fabrication technologies in architecture.

With respect to Generative Design, Shape Grammars have been developed by Stiny [1975], while issues of self-organization dealing with Cellular Automata have been employed in architectural design experiments by Coates [1996]. Furthermore, Evolutionary Design has been implemented by Elezkurtaj and Franck [2002] in floor plan design.

Building Information Modeling has been extensively discussed at the ACADIA Conference [2004] and has been described and published published in the respective conference proceedings edited by Eastman et al.

Digital design and fabrication exemplified in ONL-projects constitutes a key-reference within this research mainly because of its relevance in this research field and Oosterhuis' direct influence on content and implementation of this research.

2. CONCEPTS AND METHODS

Concepts and Methods [2] describes ideological and methodological changes occurred in architecture through the influence of digital technologies such as changes in typology, morphology, methodology and ideology.

2.1 Typology

Changes in typology are announced by Ito [1997] in his observation regarding the development of a new hybrid-building type, the mediatheque. He describes how electronic media surpasses other media such as paintings and books, and therefore, established typologies for museum, library, and theater, are challenged by a new building concept in which typo-morphological differences are blurred and reconstructed as mediatheque.

2.2 Morphology [of the Double-Curved]

2.2.1 De-Formation and De-Construction: F.O.Gehry and Coop Himelblau

In terms of changes in morphology, the morphology of the double-curved is well represented in Gehry's architecture. Thinking of Gehry's architecture in Kipnis' terms [1993] as an architecture of De-formation and Coop Himmelblau's architecture as an architecture of De-construction the differences in morphology seem obvious: Surface characteristics such as continuos-smooth and angular-facetted can be traced back to their corresponding SOFTWARE-INTRINSIC MORPHOGENETIC FEATURES. These refer to attributes and software characteristics related to generation and control of geometries pre-determining design. For instance, curvilinear surfaces are based on the geometry of continuous curves mathematically described as NURBS. N2.01

Does this curvilinearity, though, represent a break with the past? Contemporary computer-based architecture seems to reject the notion of urban and structural typology suggesting an ideological, conceptual and formal break. However, biomorphic forms based on curvilinear geometries are not new: There is a range of precedents from Gaudi‘s Casa Batlo [1905] to Mendelsohn‘s Einsteinturm [1921] and Domenig‘s Zentralsparkasse [1975].

Contemporary computer-based architecture addresses notions of biomorphology and deformation, exemplified in buildings such as Gehry’s Peix [1992] and Guggenheim Museum [1997]. Kipnis [1993] considers this architecture an architecture of De-formation, since it emphasizes form, regards modernist language historical reference and rejects Euclidian-geometries.

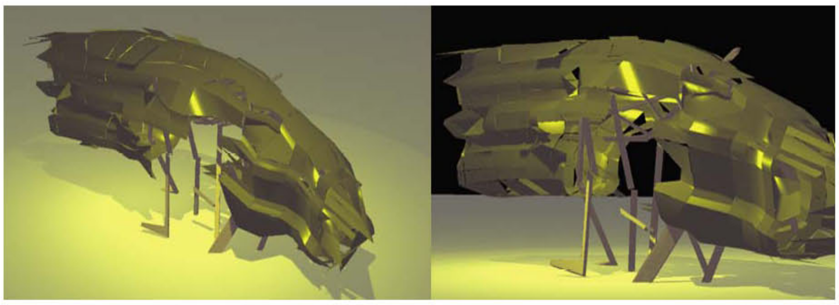
2.2 Morphology [of the Double-Curved]

2.2.2 Software-Intrinsic Morphogenetic Features: Digital Design Strategies - The Curvilinear and the Facetted

An experimental approach to Digital Design has been implemented within this research in three workshops [2003-2005] at the universities in Innsbruck, Rome, and Antwerp reflecting critical analysis and assessment of 3D modeling software as design tools.

As a field of studies in architecture Digital Design introduces the use of computer to students not only pragmatic, but also conceptual, as an instrument to explore complex systems of organization. A case study - Denari's project The Wall - served as a framework for a 3D-modeling survey with emphasis on implementation of 3D-construction and editing operations as well as examination, interpretation and evaluation of 3D-modeling tools and their intrinsic morphogenetic features.

PROGRAM INTRINSIC MORPHOGENETIC FEATURES refer to attributes and software characteristics related to generation and control of geometries pre-determining design. Rhino's morphogenetic features are based on the geometry of continuous curves and surfaces, mathematically described as NURBS: Non-Uniform Rational BSplines. The ability to control effortlessly their shape by manipulating control points implies rigor in understanding and applying methods to generate and transform architectural space.



F2.01

In contrast to Rhino, which is a surface modeler, FormZ operates with solids, allowing a wide range of additive and subtractive operations. FormZ's shape-generating features rely on Boolean-operations such as union, intersection, and difference applied to solids.

Questions of HOW these computer programs are determining and influencing the design became relevant on the level of experimentation: Students were asked to develop and implement software knowledge in a design process, which implied the TRANSFORMATION of an architectural object designed by N. Denari: THE WALL [1990].

The transformation process defined by operations such as repetition, segmentation, differentiation, diversification, deformation and rules such as symmetry-asymmetry, regular-irregular, repetitive-nonrepetitive, implies not only software-determined geometric properties such as curvilinear-angular, but also qualities of space such as soft-sharp.

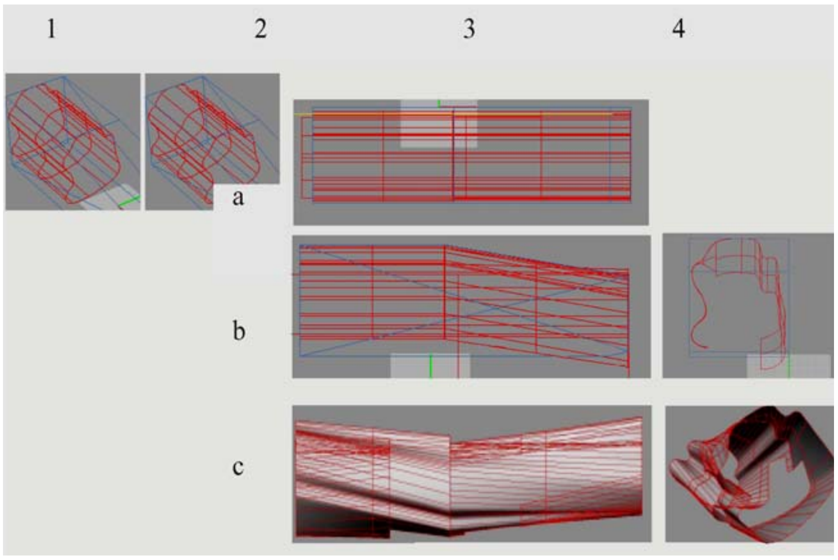
In this context, the transformation of a NURBS-surface into a polygon mesh - as illustrated in F2.01 - represents a transformation from curvilinear to facetted. Polygon meshes defined as sets of connected planar surfaces are used not only to represent polygonal objects but also aproximate curved surfaces by increasing the number of polygons in order to create smooth transitions. Decreasing the number of polygons creates angular discontinuities in the surface accentuating its angular-facetted character in opposition to a continuous curvilinear surface.

Considering the transformation process as consisting of a series of evolving spatial configurations, and the choice of a stadium in accordance with targeted formal and programmatic conditions as being a feasible methodology of design development and evaluation, the workshop gave insight into the influence of computer-based design techniques on design.

In this context, Shape Grammars provided a systematic approach to the generation of designs: According to Stiny [1980] Shape Grammar is defined as a set of rules based on shape, used to generate designs by applying rules such as *a->b* where *a* and *b* denote shapes, and a rule is applicable only if shape *a* is part of shape *b*.

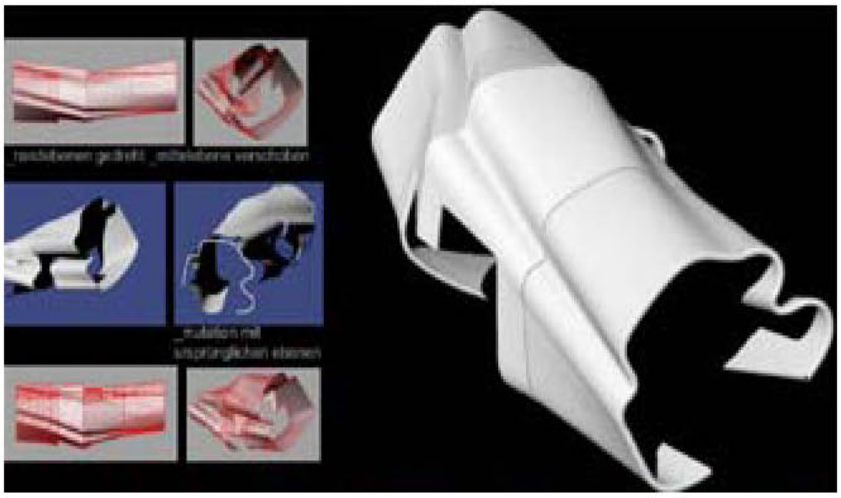
In terms of Shape Grammar, the vocabulary used in the workshop contained one initial shape: A cylindrical object - *1a*. The spatial transformations applied to this object were multiplication, translation, reflection, rotation, whereas the rules represented a selection of spatial transformations chosen in accordance to the constraint that shape *a* is part of shape *b*, which in turn is part of shape *c*. The rules consist of three consecutive operations involving two spatial transformations - multiplication and translation.

F2.02 shows the primitive *1a* and its duplicate *2a* in an axonometric view. The extended duplicate is translated and positioned adjacent to the primitive *3a*. Operations such as parallel translation of the front-face are shown in *3b* and *4b* in side and front views, whereas front face translation, rotation, and torsion of primitive and duplicate are shown in *3c* and *4c* side and front view, respectively.



F2.02

Basically, the grammar development starts with shapes as basic components and spatial relations between shapes. The definition of a rule involving a spatial transformation *a->b->c*, as described, leads to the development of a series of shapes. Derivations of the rule defined as a sequence of designs, where each design is generated from the previous design by applying rules, implies an ordering of the shapes into a hierarchy, which provides a proliferous design-generating method.



F2.03

Furthermore, decomposing Denari's object into a hierarchy of subshapes allows for any parts of the shape to be transformed within a system of rules. Denari's object can be seen, therefore, as a template to study curvilinear geometries in architecture.

Based on experiments in Rhino and FormZ, this workshop introduced students to digital design, offering them clues to evaluate, compare and consciously decide about the use of 3D-modeling software in the design process.

The experimental approach to digital design implemented in this Rhino-FormZ workshop implied three pre-defined conditions: [1] Students had no preliminary knowledge of the 3D-modeling software, [2] Design task was restricted to the transformation of an object designed by Denari, and [3] Workshop established a link between design and software on the geometrical-formal level as well as explored how Shape Grammar informs the design.

Students implemented 3D-modeling experiments revealing intrinsic morphogenetic features of computer programs by generating complex geometries using modification and assembly of simple geometrical objects resulting in complex, composite-objects.

Aside from 3D-modeling experiments, transpositioning digital design strategies from practicing architects such as N. Denari in the academic environment ensured exploration of methods and techniques, which found implementation in internationally relevant architectural practices.

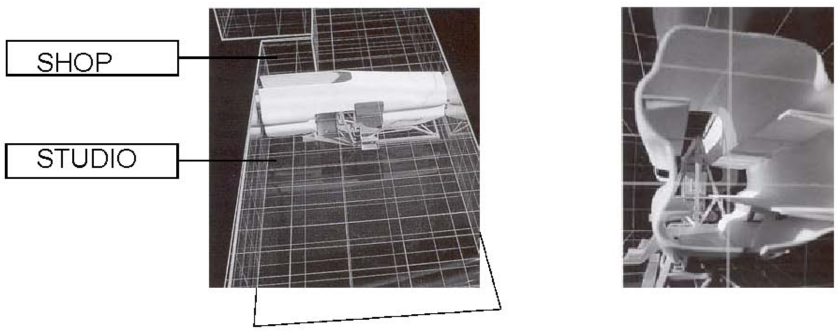
The follow-up workshop has been organized [2004] at the University of Studies Roma Tre in Rome and has been focusing on the implementation of digital models into physical prototypes:

2.2 Morphology [of the Double-Curved]

2.2.3 Architectural Avant-Garde: Curvilinearity as Negation of the Past - Euclidean and Non-Euclidean Geometries in Architecture

The workshop at Roma Tre exploring aspects of curvilinearity in architecture has been structured in three parts: [1] Other Geometries, [2] Objects, and [3] Spaces.

[1] Other Geometries started with the analysis of the Denari-project, which has been designed to separate a space in two areas - studio and shop - serving simultaneously as space subdivisioning element and storage space.

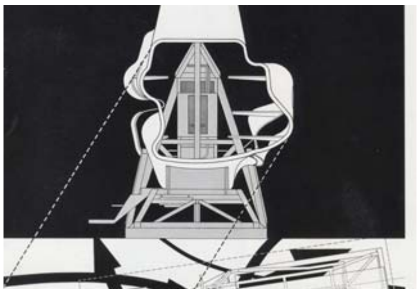


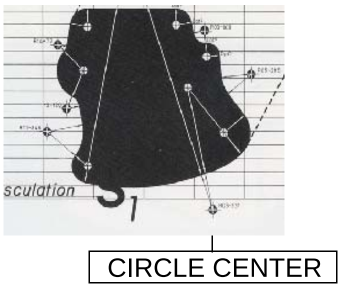
F2.04

The workshop started with the development of design skills by using 3D-modeling tools based on NURBS-geometries, whereas the computer has been employed not as a representation but as a design tool.

Even though Denari did not use NURBS and developed the curvilinear shape from circles - meaning that every time the curvature changes, every single circle center, radius, and tangent have to be re-designed and re-positioned - the workshop focused on NURBS, specifically emphasizing the difference between curves from circles and NURBS.

[2] Objects explored methodologies of digital design: While Denari`s object can be seen as a template for studying curvilinear geometries in architecture, its decomposition into a hierarchy of sub-shapes allows for any parts of the shape to be transformed within a system of rules - as described in section 2.22.

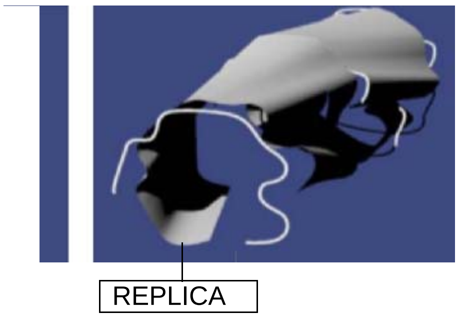




F2.05

From analysis of the original to development of a *replica* and a *mutant*, the process is based on the exploration of NURBS-geometries and development of alternative designs through trans-FORMATIONS of the digitally modeled object into so called MUTANTS.

Considering the mutant a *new* organism resulting from a *genetic* change in the *parental* type, the gene of the Denari-object can be seen as the CURVE.



F2.06

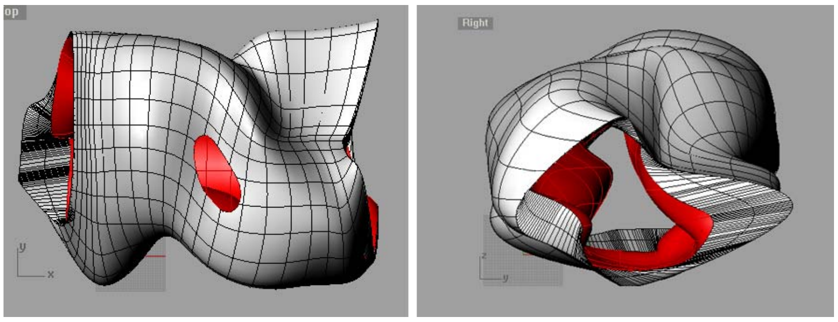
Furthermore, assuming that curvilinear architecture is being developed by generating space through following the movement of the body in space based on ergonomic principles, the workshop implemented intuitively generation and modeling of the double-curved skin in accordance to estimated movement within the proposed program. Basic idea has been to generate space by following the movement of the human body, whereas the volumetrical outlines of the body in motion establish an initial framework for spatial development.

The third part of the workshop focused on the implementation of the digital into physical models:

[3] Spaces implied the development of a coherent architectural space and generation of physical models by means of contouring, whereas contouring is a process in which supporting primary and secondary structure is generated by sectioning the object.

This process, basically, emulates the CAD-CAM fabrication process and informs students about the implementation of complex non-Euclidean geometries in architecture: While the physical model has been considered a working model, which means that input coming from the model has been taken in consideration to eventually change the design, the digital model contained all data and became source of fabrication.

Students employed intuitively principles of *space-customization* for the development of a *mutant* named THE KIOSK.



F2.07

The space accommodates activities related to the operation of a food-kiosk, wherein equipment and movement patterns have been estimated with conventional sketching methods. The inner-space accommodates movement and operation of equipment components, which are placed in the space located between the outer- and the inner-skin on two levels accessible for users: 0.00-0.90 m and 0.90-1.80 m.

The NURBS-based space has a degree of spatial complexity, which proves to be difficult to control on the operational level - how much space is available for equipment components and their operation can only be roughly estimated.

The workshop showed that computer programs and their intrinsic morphogenetic features are determining and influencing not only the design but also the design process: While experiments in digital design offer clues to HOW software influences design and design processes, *new* design strategies can be established by gaining expertise in the creative use of software.

The workshop employed a two-phase procedure: [1] DESIGN: 3D-modeling, rendering, visualizing, and [2] FABRICATION: Contouring, which implied construction of a physical model by extracting sections from the digital model.

While Denari developed his design based on circles, the workshop emphasized the use of NURBS: The workshop departs, therefore, from modernist language as historical reference and rejects Euclidean-geometries, in order to explore NURBS.

Based on the premises that students had no preliminary knowledge of the employed 3D-modeling computer program and the design task was restricted to the transformation of an object designed by Denari, the workshop established a link between design and software on the geometrical-formal level.

Students implemented 3D-modeling experiments revealing intrinsic morphogenetic features of computer programs and their influence on design: The developed experiments implied generation and control of complex double-curved geometries, wherein deformation has been the major focus. Control of deformation and manipulation of the NURBS-geometry has been achieved by establishing a methodology in which the volumetrical outlines of the body in motion generate architectural space. Even though applied intuitively this methodology established rules to generate and transform space.

This workshop not only assessed 3D-modeling tools and their intrinsic morphogenetic features but redefined strategies of design and ensured their implementation in education.

F2.08

The use of a case study, such as Denari‘s design has been supporting the development of an initial understanding not only for design based on curvilinear shapes but also for the difference between curves based on circles and NURBS-geometries. It offered a primitive to start the 3D-modeling with, connecting the learning of the software with the development of the replica, reinforcing the idea, that software taught from within a design task is beneficial for the associative learning process.

2.3 Methodology

2.3.1 Diagrams > Patterns > Computations

Computer-based design methodologies can be traced back to, inter alia, Alexander’s *pattern language* [1977], Newell’s *information processing* theory [1957], and Wolfram’s concept of *computational equivalence* [2002] that views any process natural or artificial as a computation of equivalent sophistication.

In this context, design is conceived as emerging from a process of defining design problems by diagrammatically assessing recurring problem characteristics and identifying problem solutions, which are correspondingly encoded in *patterns*. These, according to Alexander, not only describe design problems but also incorporate generic solutions, which describe spatial arrangements *customizable* to particular contexts.

Within this research, patterns are encoded in *algorithms* in order to compute best possible solutions for the formulated problems. Instead of one, this methodology enables generation of multiple solutions by departing from a *singular*-design principle, that represents a potentially prejudiced position of the *master*-designer. This methodology implies generation of all possible designs within the spectrum of an *optimal* solutions field.

For Alexander [1977] diagrams have a generative capacity enabling development of patterns from which *proto*-building and *proto*-urban patterns can be abstracted.

Patterns, however, do not determine the design but rather narrow down design choices to a relevant range:

Alexander's PATERN LANGUAGE offers, basically, a space planning methodology - It not only defines planning problems but also structures and frames them by specifying the elements of those problems and their relationships to each other.

In this context, problem spaces can be seen as sets of *knowledge states* containing potential solutions [Newell et al., 1957]. Taking those knowledge states as input, new knowledge states can be generated by applying operations defined as *generative processes*. N2.02 N2.03

The newly generated knowledge states are verified by applying *test procedures*, whereas choice of generative processes and test procedures are determined on the basis of the information contained in the problem space definition.

Within this research, computer programs incorporating these problem-solving mechanisms generate, inter alia, solutions for layout problems:

These involve placement and adjustment of functional units in respect to one another and with respect to a whole - the building. The *best* solution is achieved when no further improvement can be observed in the overall arrangement.

However, *best* solution does not necessarily coincide with *best possible* solution, and therefore the problem is reformulated as an optimization-problem. In this context, search spaces are narrowed down to optimal solution spaces in a sequential yet iterative three steps procedure: diagrams > patterns > computations.

Considering computations computer-based information processors employing *algorithms*, the following design methodology has been asserted:

[1] Find and describe diagramatically design problem, [2] Frame problem by defining problem-intrinsic patterns, and [3] Solve problem by generating and computing appropriate algorithms. N2.04

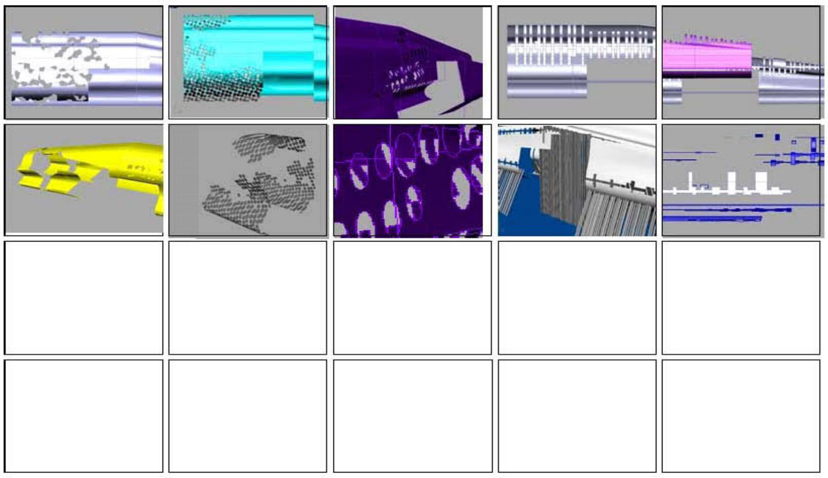
This three-steps procedure has been implemented in SC : FL described in section 3.3.1.

2.4 Ideology

2.4.1 Principles of Digital Media Revisited

Departing from Manovich's definition of principles of new media this research adds to his fifth principles a sixth one:

SINCE DIGITAL, ALL MEDIA BECOME ONE - According to Manovich [2001] digital media can be described formally by using mathematical functions, and therefore, it becomes subject to algorithmic manipulation. All media - sound, image, movie, text, etc. - are represented as machinic code, which means that the sound code can be used to generate a drawing and vice-versa.



F2.09

Manovich defines five principles, which distinguish new media from old: Numerical Representation, Modularity, Automation, Variability, and Transcoding.

[1] Numerical Representation implies that media is numerically represented, and therefore, it is subject to algorithmic manipulation.

While algorithms are expressed in *programming languages* instructing a computer about what steps to perform in what order so that specified tasks are being errorless executed, computations employ machine-readable code. This consists of a system of instructions and data structured as patterns of *bits* - *bi*nary digi*t*s - such as 00 01 00 00.

[2] Modularity - Media elements such as image, sound, text, are represented as individual objects, which can be assembled into larger-scale object-compounds such as Power-Point files, but they continue to maintain their separate identity.

[3] Automation - Discrete representation of *information* and its numerical coding allow automation of operations involved in media creation, such as access, editing, and manipulation.

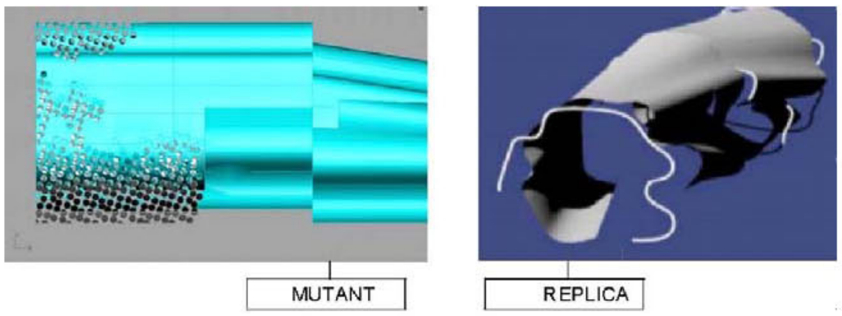
[4] Variability: A new media object such as a Web-site is not something fixed but might exist in different, potentially, *infinite* versions. N2.05

[5] Transcoding, Manovich's fifth principle, reflects the ability of digital media to represent both, image, and machine-readable code. Digital media, therefore, are characterized by those two distinctive layers.

Reinterpreting Manovich's TRANSCODING-principle as a process of translation and conversion from one digital medium to another, this section presents the results of a workshop held at the Higher Institute of Architectural Sciences Henry van de Velde [HIAS] in Antwerp [2005].

The workshop explored design concepts based on Numerical Representation, Modularity and Transcoding, whereas transcoding has been reinterpreted as translation from one medium to another, by applying graphical rules to sound compositions and vice versa. It introduced students to software they did not previously know: A surface modeler - Rhino - and a sound editing tool - Audacity.

Beyond learning to use these programs as design tools, the targeted outcome of the workshop has been the generation of designs, so called replicants and mutants, based - once again - on Denari's project THE WALL.



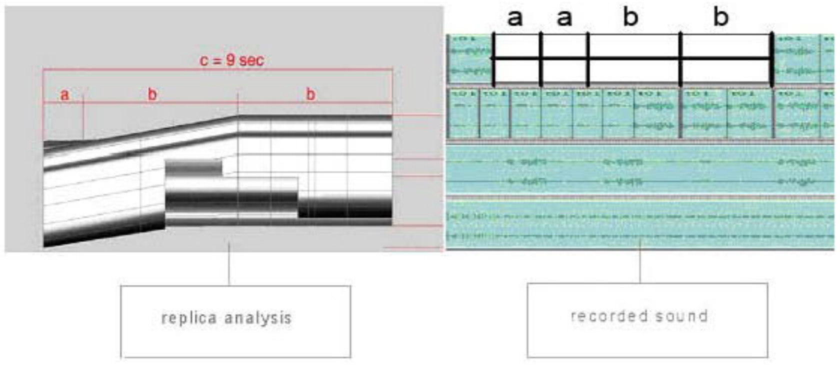
F2.10

From the original to the replica and the mutant, the transformation process involved interpretation of the Denari replica as a *new* media object by means of transcoding: The sound-analysis provided the diagram from which rules could be abstracted to transform the replica into a *mutated* media object.

In a first step the replica has been analyzed in regard of proportions - spatial and geometrical relations of the parts to the whole - in order to generate a diagram, with a translation scale of length equaling 9 seconds. This diagram represents a template, which has been generated and used for the development of a sound-composition.

In a second step samples of sound such as the tick-tack of a watch, the murmur of a conversation are imported in Audacity, which is a music sampling software introduced to the workshop participants. A sound-composition has been generated in a cut-and-paste procedure according to the diagram representing the proportions of the replica developed in the previous step. These proportions represent, the *rhythm* of the new sound composition.

The sound-composition has been, in a following step, pixelized and, finally, translated in a sound-no sound diagram from which a pattern template has been developed. The developed pattern template has been used to perforate the skin of the replica generating a mutant, which could be read, according to Vasquez-Ruano [2005], as a test of the *sound* of the space.



F2.11

Technically speaking, the mutants have been generated in a process of transfer from Rhino to Photoshop and Audacity, while the mutation process itself implied operations such as subtract/cut/puncture applied to the double-curved surface.

The cut-out patterns correspond to compositions, which have been generated from sound. The sounds have been, therefore, re-composed in Audacity, filtered in Photoshop, drafted over, extruded and intersected with the replica in Rhino. The puncturing, cutting, intersecting, and subtracting operations are conform to the patterns developed from the sound composition.

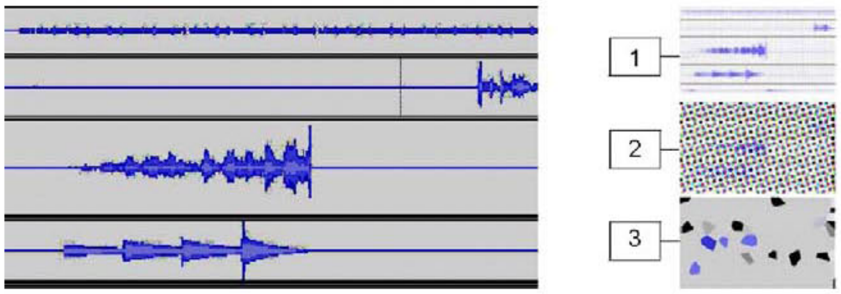
With respect to filters, three different filters - F2.12 - have been applied: One is generating a rectangular pattern, the second is generating a circular pattern and the third a mosaic pattern.

These three filters have been applied to the sound composition, which has been generated in Audacity:

The resemblance between diagram and sound-composition as graphical representations is obvious. However, the sound-compositions themselves, suggest a rather random sequence of different sounds, since they have been generated not by hearing them but by composing them according to graphical considerations.

Projects developed by students applying a pattern based on the rectangular sample - F2.13 - that generated bizarre contours in the double-curved surface, employed the filter, which resembles very much a code language or a punch-card.

This exercise in transcoding in digital lingo across media, as Vasquez-Ruano [2005] puts it, has been proven to be an alternative design mechanism based on Manovich's and Kipnis' definition of *new media and new architecture*, respectively. Kipnis [1993] suggests that architects need to establish criteria for a new architecture by not repeating the mistakes of modernism, which were erasure and replacement, instead using recombination and experimentation.

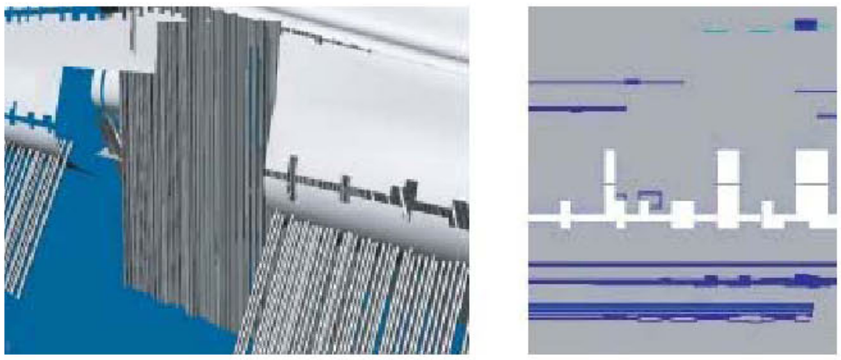


F2.12

Considering modernist language historical reference and rejecting Euclidean-geometries the workshop developed an experimental approach based on non-Euclidean geometries. It, furthermore, employed recombination and experimentation with forms: Its aim has been to consciously test the concept of transcoding across media in architectural design in order to explore their creative potential.

Although Manovich mentions *hypermedia* as a variant of his principle of variability, he misses to elaborate on its interactive nature: N2.06

INTERACTIVITY, which can be seen as a sixth principle, is based on the concept of digital linking or hyper-indexing and has been addressed within this research in the third chapter.



F2.13

Section Contributors: This workshop is the result of a collaboration with O. Vazquez-Ruano and K. de Bodt.

Acknowledgements: This section has benefited from the contribution of students participating in ADSL 2005.

2.4 Ideology

2.4.2 *Hyper-* and *Super-Modernity*: Digitally-Driven Architecture N2.07

Similarly to the way industrial fabrication with its concepts of standardization and serial production has been influencing modernist architecture, digital fabrication influences contemporary architecture:

While standardization focused on processes of rationalization of form, mass-customization as a new paradigm, which replaces mass production, focuses on uniqness, which becomes as easy and economic to achieve as repetition [Slessor, 1997].

As described in the first chapter, mass-customization is being implemented in CAD-CAM processes, which are computer-driven design and fabrication processes. These enable generation of complex designs based on NURBS-geometries:

While modernist architecture has been developed in a modular, repetitive way by using grids and proportions based on Euclidean-geometries, contemporary architecture is experimenting with non-Euclidean geometries, which describe both hyperbolic and elliptic geometries [Greenberg, 1993].

In this context, a series of *new & old* concept pairs establish a framework within which the influence of digital tools on architecture finds definition:

[1] InfoAesthetics & Aesthetics: Analog to the modernists' beliefs that the aesthetics of industrial society emerged in the industrial realm, INFO-AESTHETICS [Manovich, 2005] suggest that the *new* aesthetics already exists in computer interfaces and tools, and the information society, might not need *new* visual languages and forms because it can re-configure the *old* ones by digital means.

In this context, one of the relevant computer-based reconfiguration means is morphing:

[2] Morphing & Collage: Considering collage as an assemblage of different entities into a whole, whereas the parts maintain their own identity, morphing changes an entity into an other in a seamless computer-implemented transition.

MODERNIST	CONTEMPORARY
<RE>: RELATIVE	**<IN>: INTERACTIVE**

STATIC	DYNAMIC
COLLAGE	MORPHING
IMAGE	**USER INTERFACE**
INDEX	HYPERINDEX
PATTERNS	COMPUTATIONS
MECHANICAL	DIGITAL
MODULAR	FREE-FORMED
REPETITIVE	UNIQUE
STANDARDIZED	MASS-CUSTOMIZED

MODULOR	**SPACECUSTOMIZER**

T2.01

[3] User Interface & Image: The User Interface [UI] changes our understanding of Image, since it does not represent something to only be gazed at, but rather something to *interact* with. A desktop of a computer might have a *passive* background and *active* icons, which establish connections to other documents and software via *hyperlinks*.

[4] HyperIndex & Index: The HyperIndex represents a form of idexicality, employing a trans-referential system existing in more than three dimensions, *hyperspace*, as an instrument to link and arrange non-sequentially references in databases.

Summarily speaking, hyperindexing involves the generation of metadata - *data* about data - by establishing primary connections and relations, while storing secondary, for the query redundant information. N2.08

On a large scale the shift from mechanical to digital brings with it a change from modular, repetitive, and standardized to free-formed, unique, mass-customized architecture:

[5] Electronics & Mechanics - The shift from mechanical to digital forces architects to reposition themselves: since they do not produce *merely* drawings but produce digital data, which becomes single source of design and fabrication [Kolarevic, 2003].

Furthermore, digital systems not only inform the design process but also the fabrication process, challenging the MODERNIST concept of standardization, introducing the concept of mass-customization, which implies, as Slessor [1997] put it, that uniqness is now as easy and economic to achieve as repetition.

If modernist architecture is about Euclidean geometries, contemporary architecture looks into the potential of non-Euclidean geometries:

[6] Non-Euclidean & Euclidean geometries - For over two thousand years only Euclidean geometry has been known, when Einstein's theory of general relativity has shown that Euclid's axioms are only an approximate description of the physical space: The fifth axiom, the axiom on parallels, has been proven to be incorrect [Greenberg, 1993].

Non-Euclidean geometries have shown that there are infinitely many parallel lines in the elliptic space and there are no parallel lines in hyperbolic space. Even though Euclid believed that his axioms were self-evident statements about physical reality, Einstein's theory of general relativity has shown that the geometry of space-time is non-Euclidean.

Non-Euclidean geometries have been applied in architecture in a rather experimental way - examples show often the juxtaposition of Euclidean and non-Euclidean spatial principles. Gehry's architecture, for instance, employs a spatial structure based on Euclidean geometries, while the building envelope follows curvilinear rules. Non-Euclidean geometries require, therefore, a *new* discipline in architecture:

[7] Spacecustomizer & Modulor: While Modulor [Le Corbusier, 1943] is a system of proportions, which uses measures of the human body partitioned in modules according to the Golden Section and two Fibonacci Series, SpaceCustomizer uses measures of the human body in architecture by tracing the movement of the body in space. The volumetrical outlines of the body in motion establish an initial framework to develop spaces employing movement studies based on ergonomics.

Modulor applies a 2D proportioning system, while SpaceCustomizer employs a 3D, dynamic, space-generating system. Both put man as measure of architectural space: However, Modulor targets modular space structures and standardization, while SpaceCustomizer aims for non-standard, mass-customized spatial configurations.

In addition to non-Euclidean geometries, contemporary architecture incorporates aspects of dynamics and interactive-kinetics:

F2.14

[8] Dynamic & Static Architecture: Projects such as Decoi's Aegis Hypo-Surface and Hyperbody's MuscleTower are prototypes for not only dynamic but also interactive architecture:

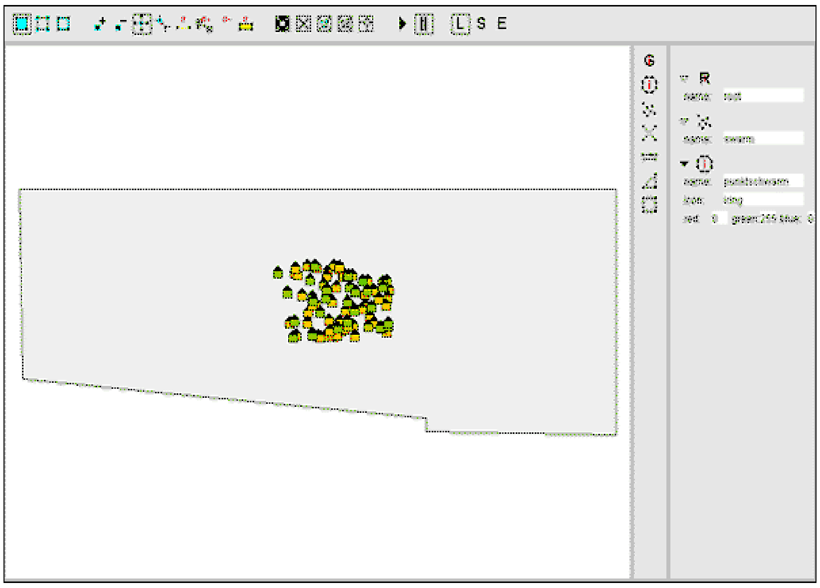
While the Hypo-Surface transposes an up-and-down movement of pistons into undulating movements of a triangulated surface, the MuscleTower is a structure rotating, bending and twisting in the 3D-space.

In addition to being dynamic, these prototypes are *interactive,* which implies that they *respond* to external inputs; They incorporate sensor-actuator technologies enabling them to *interact* with their surrounding in a *self-organized* manner. The MuscleTower, for instance, incorporates movement-sensors. As soon as movement is detected in the neighboring area the tower responds by twisting and bending in this direction.

In this context, interactivity denotes responsiveness, and can be seen as a communication process in which each message is related not only to the previous messages exchanged, but also to the messages preceding them.

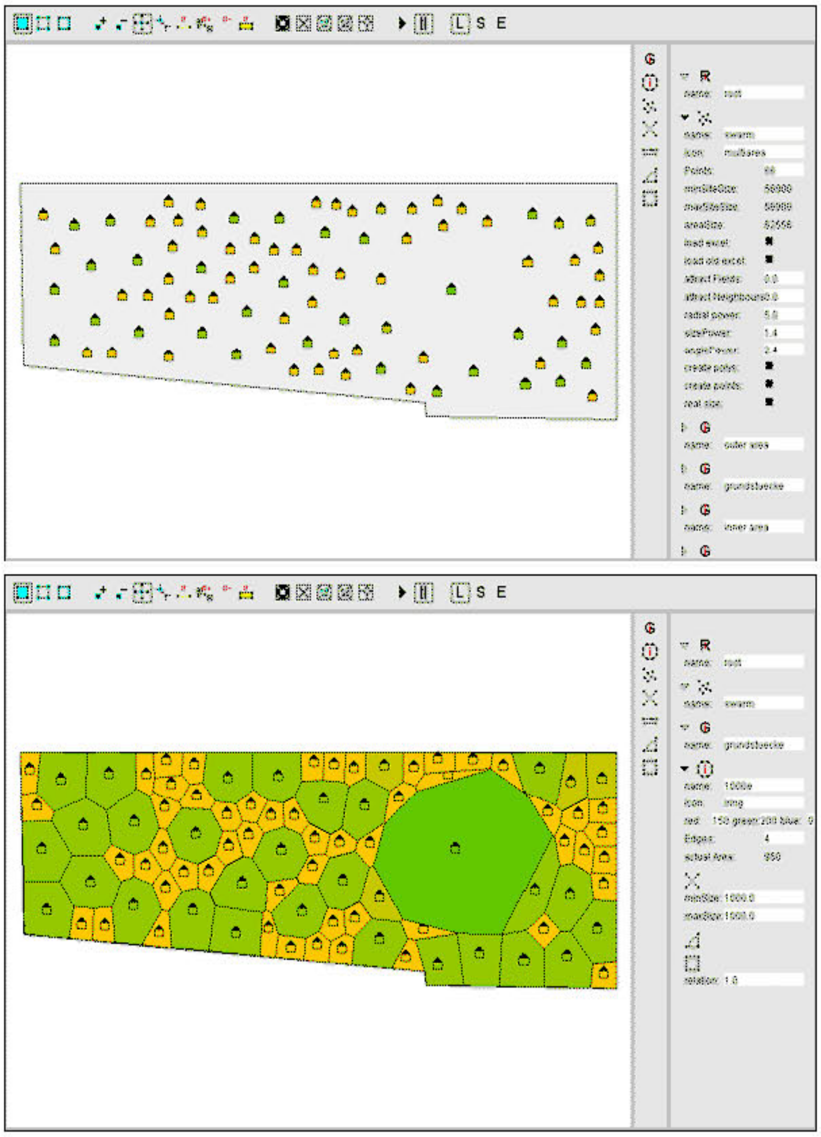
If modernist architecture is about top-down control, digitally-driven architecture is about integrating bottom-up organization in the design process.

[9] Self-organization & Control: Self-organization refers to a process, in which the organization of a system is not only generated but also increases automatically without being controlled from outside. In architecture, the exclusive control of the architect has been in part replaced by emergent design processes based on swarms, cellular automata, and genetic algorithms.



name: root
name: swarm
name: puddlesSwarm
icon: img
red 0 green 255 blue 0

F2.15



```
R
name: root
name: swarm
icon: mulbares
Points: 88
minSiteSize: 56900
maxSiteSize: 56900
areaSize: 82568
load excel:
load old excel:
attract Fields: 0.0
attract Neighbours0.0
radial power: 5.0
sizePower: 1.4
anglePower: 2.4
create polys:
create points:
real size:
G
name: outer area
G
name: grundstuecke
G
name: inner area
G
```

```
R
name: root
name: swarm
G
name: grundstuecke
name: 1000e
icon: img
red: 150 green:200 blue: 0
Edges: 4
actual Area: 950
minSize: 1000.0
maxSize: 1000.0
relation: 1.0
```

F2.15

Swarms are employed in generative design processes, which deal with ample amounts of data featuring sometime conflicting attributes and characteristics. Those attributes and characteristics are incorporated in *behaviors* according to which design components *swarm* towards targeted spatial configurations.

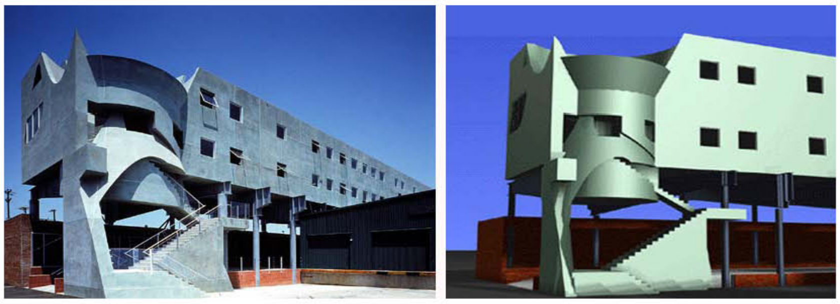
In this context, architectural design becomes procedural instead of object orriented:

[9.1] ARCHITECTURE AS PROCESS: Architectural form can be seen as emerging from a process of self-organization, in which the dynamics of all parts of a system determine the result, and therefore, the architect becomes the designer of a process instead of a result.

Kaisersrot [Braach et al., 2002] is a software prototypes developed to generate urban and architectural structures based on Swarm Intelligence: Urban and architectural components swarm and organize themselves according to predefined rules towards a targeted spatial configuration. These rules pertain to function, structure and geometry, while formal aspects have been neglected.

Similarly, BuildingRelations [BR], developed with Hyperbody-students within this research, employs Swarm Intelligence for design development and is being described in the third chapter.

[9.2] ARCHITECTURE AS RESULT: Compositional rules specific to computer-based designs emerge from software-intrinsic morphogenetic features, which can be traced back to operations such as Boolean intersection, difference and NURBS-manipulation. Moss' Samitaur, for instance, reveals characteristics resulting from Boolean operations, while Gehry's architecture is informed by NURBS-geometries.



F2.16

Samitaur, is the result of in FormZ intersected geometrical objects such as cones, cylinders and bars:

These have been interpenetrated, fragmentally deleted and recomposed in an apparent random way. Openings have been punched through the conglomerate without consideration for resulting, perhaps, contradictory complexities.

While Moss' intersections of geometrical objects seem to stay in the realm of deconstructivist principles based on Euclidean geometries, Gehry's deformation principles employ non-Euclidean geometries.

Both approaches involve a sculptural notion of spatial composition and are pre-determined by software intrinsic morphogenetic features - as described in section 2.2.2.

In this context, software intrinsic morphogenetic features refer to shape-generation properties and characteristics inherent design software such as FormZ and Rhino:

Both approaches involve a sculptural notion of spatial composition and are pre-determined by software intrinsic morphogenetic features - as described in section 2.2.2. These refer to shape-generation properties and characteristics inherent design software such as FormZ and Rhino:

FormZ is a solid-modeler enabling Boolean operations, such as intersection and difference between solids as shown in the Samitaur project, while Rhino is a NURBS-modeler allowing for deformation as shown in Music Experience Project.

F2.17

Even, though, Gehry has mostly employed inverse engineering - see section 1.2.1 - without relying on software to generate designs, more recent designs such as the Music Experience Project reveal morphogenetic characteristics stemming from NURBS-modeling software such as Rhino.

In opposition to the projects employing swarm principles described previously, these two projects reflect a top-down approach to design:

The architect designs the result by digital and non-digital means with no or little consideration to potential input stemming from some form of *machinic intelligence*. More specifically, knowledge about the designed object is exclusively incorporated in parametric models developed in Gehry's *post*-engineering process. In addition, Gehry's construction process involving BIM as a digital representation and communication model, which facilitates exchange of information in digital format, incorporates aspects of intelligence as described in section 1.1.2.

CONCLUSION: While non-Euclidean geometries influence form, the knowledge about the designed object is incorporated at the level of its connectivity with data stemming not only from its geometry but also from its *content* and *behavior*. Digitally-driven architecture implies, therefore, on the one hand, digitally-designed architecture, on the other hand, it implies architecture generated and controlled by digital means.

Beyond, digitally-driven design processes, digitally-driven architectures such as interactive architectures are of interest, since they not only incorporate knowledge about the designed object but also involve *intelligent* interaction between user and architectural space. This interaction is based on data-exchange between building components and users as described in the third chapter.

NOTES: Notes to this section explain concepts and notions difficult to extract from context.

N2.01 - Morphogenesis originates from the Greek *morphe* meaning shape and *genesis* meaning creation - http://en.wikipedia.org/wiki/Morphogenesis.

N2.02 - Information Processing implies processing of data into information by means of computation, which utilizes mathematics, namely, algorithms to generate information from data [Newell, 1990].

Problem finding, framing and solving are parts of Information Processing: While problem finding might require vision and creative thinking, problem framing involves, mainly, critical thinking.

N2.03 - Knowledge is structured information of theoretical and/or practical content.

N2.04 - Algorithms describe tasks and sequences in which those tasks are to be implemented by a computer.

N2.05 - Information is the result of data processing, in which data is manipulated and organized in such a way that it increases knowledge [Newell, 1990].

N2.06 - Hypermedia is a term introduced by Nelson [1965] used to describe hyperlinked graphics, audio, video, and text in order to generate a non-linear medium of information. PowerPoint presentations and Web-sites are hypermedia.

N2.07 - Hyper- and Super-Modernity: In contrast to Modernity, which dealt with *moderate* changes while being embedded in a historical context, Hyper- and Super-Modernity view history as an unreliable guide, since changes take place at an accelerated pace [Charles and Lipovetsky, 2006].

N2.08 - Data are numbers, characters, pixels, which are usually processed into information by means of manipulation and organization of incorporated data in order to increases knowledge [Newell, 1990].

BIBLIOGRAPHICAL REMARKS: Questions on how digital media have developed in time are addressed, inter alia, by Manovich [2001]. He describes the concept of Info-Aesthetics in a *semi-open source* book presented on the Internet - http://www.manovich.net/. His overview on state-of-the-art computer-based concepts and methods served - within this research - as framework for evaluation and further development.

Issues of information processing and problem solving have been addressed, inter alia, by Newell et al., [1957], whereas design specific problem solving has been addressed by Alexander [1977]. He developed a design methodology based on diagrams from which patterns were developed in order to generate designs.

3. VISIONS AND PERSPECTIVES

Visions and Perspectives [3] describes frontier research on computer-based systems incorporating relevant aspects of intelligence including prototypes developed and tested within this research.

Implemented in environments such as Rhino, MaxMSP, Virtools, Processing, and MiniSat the developed software prototypes address issues of motion-based 3D-space generation, NURBS-geometry voxelization, and automated functional layouting. They incorporate aspects of Artificial Intelligence [AI] based on computer vision and sensor-actuator as well as constraint solving techniques.

3.1 Organic-Inorganic Relations

3.1.1 Machinic Reasoning: Artificial Intelligence, Human-Computer Interaction

3.1.2 Cybernetic Organism

Generally speaking, Artificial Intelligence [AI] is intelligence exhibited by a computer [McCarthy, 1956]. It incorporates methods classified as machine learning, such as Expert Systems [ES], which process known information and provide solutions based on them [Jackson, 1998]. The Microsoft Office paperclip, for instance, is a well known ES: It recognizes specific typed-in features and makes suggestions for corrections accordingly.

AI also involves learning-based methods employing Neural Networks, which are systems with pattern recognition capabilities, Fuzzy Systems, which apply techniques for reasoning under uncertainty, and Evolutionary Computations, incorporating Swarm Intelligence and Genetic Algorithms [McCarthy, 2004].

From the perspective of their incorporated aspects of intelligence, prototypes developed within this research can be considered *hybrid-intelligent* as they combine methods relying on Expert Systems, Swarm Intelligence and Constraint Satisfaction.

Since they function under human supervision, these systems can be seen as being *interdependent* and requiring Human-Computer Interaction [HCI] devices and interfaces.

Incorporated in the human body, HCI-devices such as integrated pacemakers, intraocular lenses, cochlear implants, and robotic prostheses, contribute to blurring borders between organic and inorganic matter, transforming humans into Cybernetic Organisms [Haraway, 1991]. Similarly, wearable HCI-devices enveloping the human body, and ubiquitous computing systems integrating computers into the living environment intermingle increasingly organic-inorganic relations. The following sections describe, however, prototypes employing HCI-interfaces and devices developed within this research, addressing issues of architectural design-development and computer-integration into architecture.

3.2 Semi-Automation

3.2.1 Semi-Automated Design and Fabrication Processes in Architecture: SpaceCustomizer - GeometryTriangulator and Unfolder

Considering that scientists might be able today to develop a computer model representing the *whole* universe but this computer model would have a sophistication corresponding to the complexity of the universe itself [Zuse, 1967; Schmidhuber, 1997] and knowing that according to the concept of computational equivalence formulated by Wolfram [2002] any process natural or artificial can be viewed as a computation of equivalent sophistication, this research puts forward the argument that *complete* automation in architecture might be for the time being not relevant because it would require a computer model of equivalent sophistication as architecture itself.

If the computation takes as much effort as the real process of design, what do architects gain from computation except additional insights, which they can not derive from practice?

Architects gain from computation - in addition to insights - support in the design process not only in form-finding, but also in mechanical, structural, constructive problem-solving.

The shift from mechanical to digital, forces architects to reposition themselves: [1] They do not produce *merely* drawings but produce digital data, which becomes single source of design and fabrication [Kolarevic, 2003]. [2] Furthermore, digital systems not only inform the design process but also the fabrication process, challenging the MODERNIST concept of standardization, introducing the concept of mass-customization, which implies, as Slessor [1997] put it, that uniqness becomes as easy and economic to achieve as repetition. And [3] *complete* automation from idea to building might not be relevant at the time being, because it requires computation as complex as the process itself, but semi-automation based on system-embedded intelligence, which addresses possibilities for architecture based on algorithmic techniques is of great relevance.

ALGORITHMIC TECHNIQUES such as scripting and programming enable automation of design and fabrication processes:

The script - UfoldNurbsSurface - developed within this research triangulates and unfolds *NURBS*-surfaces, and implies a degree of *intelligence*, since the script distributes points on the surface and attributes each point a specific *behavior*, according to which triangles generated between the points, translate and rotate in such a way that the complete surface is, finally, flattened and laid out on the xy-plane. N3.01

The intelligence and degree of deterministic organization lies in the way one point organizes itself spatially in relationship to the next one. In this case the *point-cloud* organizes itself into triangles: N3.02

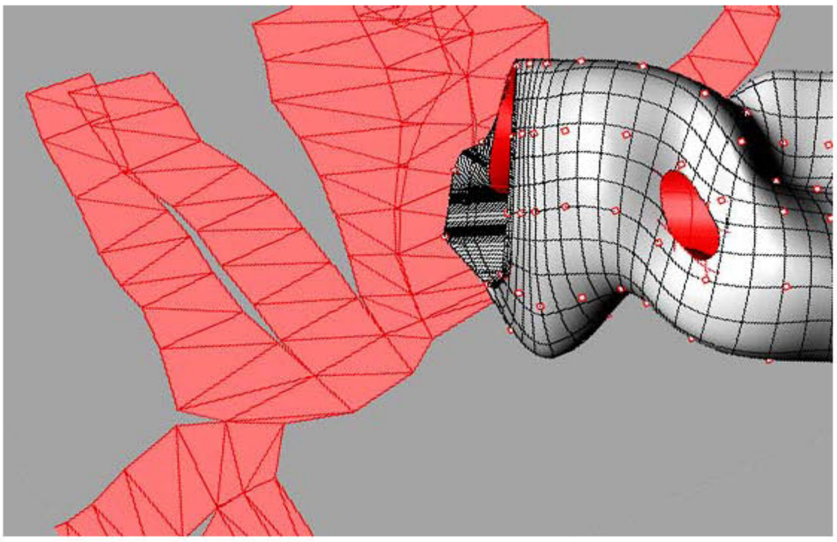
These, translate and rotate achieving a certain order, which can be seen as an emergent order, since it implies interaction between the parts of a system, such as points and triangles, in a way that this interaction leads to a specific order. The emergent order, therefore, arises not from the coexistence of the parts but from their interaction.

The rules according to which the points and triangles, respectively, interact are local, affecting the points and triangles in next vicinity, generating a global order, only after the last point and triangle, has found its place.

BACKGROUND: Even though commercial software such as FormZ have an *unfold* function, this function is limited to the unfolding of a limited range of NURBS-surfaces.

Furthermore, the unfolding of complex double-curved geometries is difficult to control: Fragments of the triangulated surface are often placed somewhere in 3D-space without any connectivity to the main triangulated surface, and therefore, finding their position within the whole is very much similar to a puzzle-game.

The script developed within this research not only triangulates and unfolds ANY double-curved surface but also places the unfolded strips as defined by the user.



F3.01

CONTENT: As an interdisciplinary field of expertise incorporating [1] Mathematics, [2] Scripting and Programming, and [3] 3D Modeling and Design, scripting-based design enables, in this case, the triangulation and unfolding of NURBS-surfaces.

[1] Mathematics: Mathematical descriptions are used in this exercise to enable translation and rotation of points and triangles, respectively. They are based in vector and tensor analysis [Borisenko and Tarapov, 1968] and inform the scripting of the unfolding process. While mathematical formulas are generic, scripts employ those generic formulas in specific configurations. Furthermore, scripts establish the sequence in which those formulas are to be applied.

[2] Scripting and Programming languages are used to develop com-

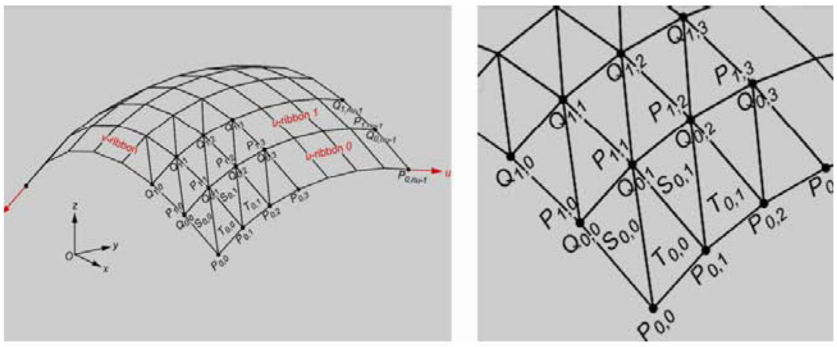
puter programs.

Scripting languages are often implemented with interpreters; They are usually stored in text form - ASCII - and are interpreted prior to being invoked. Interpreted, scripting-languages tend to be slower and use more memory while running, however, it is usually faster to program in a scripting language, and script-files are smaller than e-quivalent program files.

The scripting language used to develop Ufold- and TriangulateNurbs-Surface is Visual Basic Script: VBScript interpreted in Rhino has enabled the development of Scripting-Based Design Methods for Space-Customizer - GeometryTriangulator and Unfolder - which is a tool-kit [Bier and Schmehl, 2006] developed to implement, inter alia, tasks such as the triangulation and unfolding of a NURBS-surface.

UfoldNurbsSurface is a script incorporating several sub-routines: [1] Sub-routine ArrayPointsOnSurface generates points on *UV*-curves according to the density required by the user, [2] Sub-routine Evaluate-PointCoordinates, [3] Sub-routine AssignNodeNumbers, [4] Sub-routine ArrayTriangles, and [5] FlattenTriangles. N3.01

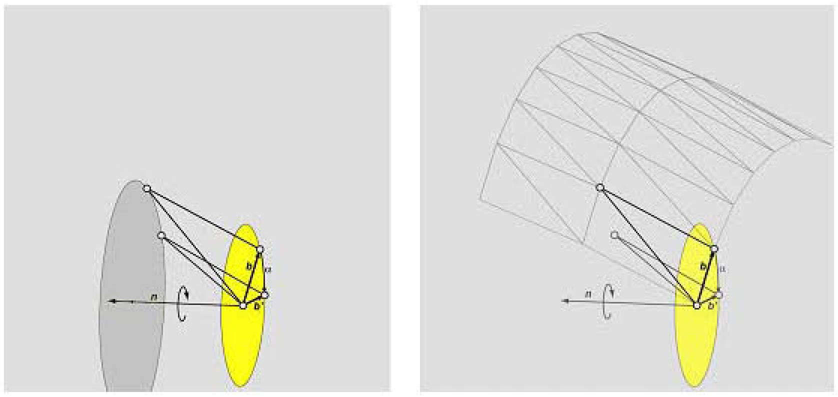
PROCEDURE: The script distributes a user-defined number of points along the UV-curves on the NURBS-surface. In a second step, it generates triangles between those points. These group into ribbons, which finally rotate, translate, and flatten into the xy-plan.



F3.02 Triangulation of the UV-mesh: According to the original UV-parametrization of the surface, the user has to specify direction, along which the ribbons are defined. Using this re-assignment of the parameter directions, the node-points of a uniform UV-mesh are generated on the NURBS-surface.

As illustrated in figure F3.02 each one of the i = 0, ..., nv-1 u-ribbons is specified by an array of bottom-node points Pi,j with j = 0, ..., nu-1 and array of upper node points Qi,j with j = 0, ..., nu-1. Two triangles are formed per surface element: An upper triangle Si,j defined by points Pi,j,Qi,j,Qi,j+1 and a bottom triangle Ti,j defined by points Qi,j+1, Pi,j+1, Pi,j.

Rotation of the first triangle edge into the plane: To prepare this step, the base point P0,0 with the complete triangulated surface attached to it is translated to the origin O. Then the first triangle T0,0 and the complete triangulated surface attached to it, is rotated around the x-axis in such a way, that the triangle edge connecting points P0,0 and P0,1 is positioned in the xy-plane. This configuration is the starting point for the iterative unfolding process.



F3.03

[3] 3D Modeling and Design: NURBS-surfaces are easy to manipulate by puling control-points. Questions regarding how to control this manipulation, which rules and design methodologies can be developed to control designs based on NURBS, are focus of the SpaceCustomizer project, not of this section, though. This section describes, exclusively, the scripting-based triangulation and unfolding of NURBS-surfaces, more specifically of ANY double-curved surface, which means:

There are no constraints and rules in the design of a NURBS-surface coming from the unfolding-process.

DISCUSSION: The presented exercise in scripted-based unfolding of a double-curved surface requires expertise in mathematics, programming and 3D-modeling. The programmed *behavior* for points and triangles is based in mathematics, which defines rules according to which the points and triangles relate to each other in xyz-space.

The established order is dynamic at least until the system reaches *equilibrium* and is exercised on the global and the local level:

While the connectivity between the parts of the system is established and applied on the global level, the rotational operations are local. Each point and triangle positions itself in relationship to the previous one. The script establishes *behavior*-rules for points and triangles similar to the rules within a swarm:

[1] Keep a certain user-defined distance to the neighbor-point, [2] Span triangle surface by connecting to two neighbor-points in the fol-

lowing sequence [P0,0; Q0,0; Q0,1] and [Q0,1; P0,1; P0,0] respective-ly, [3] Rotate [P0,0; P0,1] into xy-plane, [4] rotate [P0,1; Q0,1] into xy-plane, and so on.

While the sequence of *behaviors* is determined by the script, each *be-havior* such as translation, rotation, is determined by the mathematics behind them:

The rotation of the first triangle and with it of the whole geometry into the xy-plane, for instance, requires three mathematical *formulas* per-taining to the rotation of an arbitrary vector in the 3D-space: N3.03

[1] $\boldsymbol{R} = \boldsymbol{I} - \sin\alpha(\boldsymbol{E}\cdot\boldsymbol{n}) + (1-\cos\alpha)(\boldsymbol{E}\cdot\boldsymbol{n})^2$, [2] $\boldsymbol{b}' = (\boldsymbol{n}\cdot\boldsymbol{b})\boldsymbol{n} - \boldsymbol{n}\cdot(\boldsymbol{n}\cdot\boldsymbol{b})$, and [3] $\boldsymbol{b}' = \boldsymbol{R}\cdot\boldsymbol{b}$

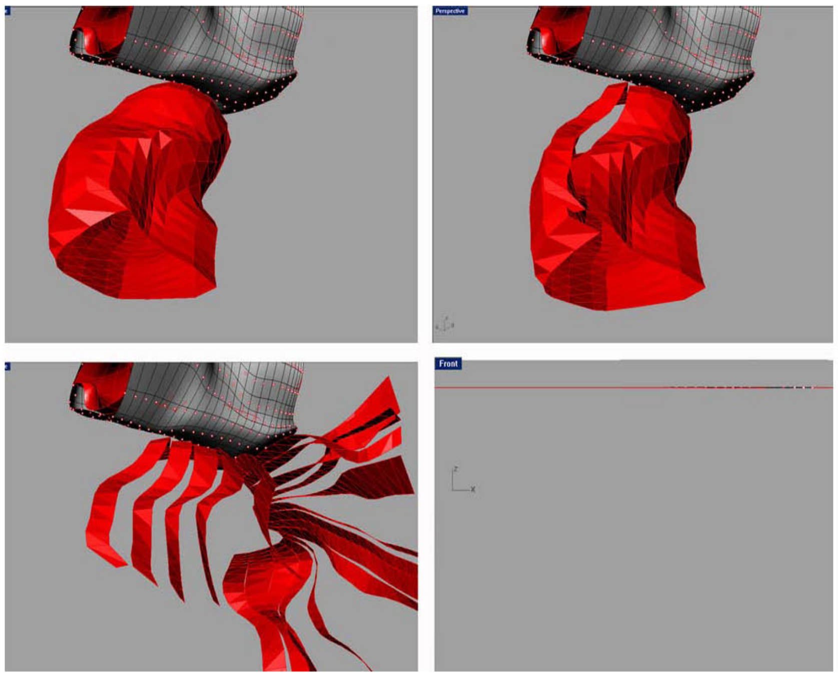
In this context, *b* and *b'* are the original and the rotated vector, respec-tively, *n* is the axis unit vector, and *alpha* is the rotation angle, whereas the implementation of the mathematics in VB-scripting is as follows:

```
Function AssignRotationMatrix (mrot, vn, alpha)
''''''''''''''''''''''''''''''''''''''''''''''''''
Rotation tensor mrot from axis unit vector vn & an-
gle
''''''''''''''''''''''''''''''''''''''''''''''''''
  Dim salpha: salpha = Sin ( alpha )
  Dim calpha: calpha = Cos ( alpha )
  mrot(0,0) = (1.0 - calpha)*vn(0)*vn(0) + calpha
   mrot(0,1)  =  (1.0  -  calpha)*vn(0)*vn(1)  -  sal-
pha*vn(2)
   mrot(0,2)  =  (1.0  -  calpha)*vn(0)*vn(2)  +  sal-
pha*vn(1)
   mrot(1,0)  =  (1.0  -  calpha)*vn(0)*vn(1)  +  sal-
pha*vn(2)
  mrot(1,1) = (1.0 - calpha)*vn(1)*vn(1) + calpha
   mrot(1,2)  =  (1.0  -  calpha)*vn(1)*vn(2)  -  sal-
pha*vn(0)
   mrot(2,0)  =  (1.0  -  calpha)*vn(0)*vn(2)  -  sal-
pha*vn(1)
   mrot(2,1)  =  (1.0  -  calpha)*vn(1)*vn(2)  +  sal-
pha*vn(0)
  mrot(2,2) = (1.0 - calpha)*vn(2)*vn(2) + calpha

End Function
```

The unfolding process is represented in figure F3.04: From left to right, the triangulated surface separates in U-ribbons. These stay connected with the primary V-ribbon, while rotating in space.

This dynamic process stops when all ribbons are flattened into the xy-plane, shown in the front-view - right-left image - as line.



F3.04

Basically, the ribbon-structure corresponds to a global order, which can be changed according to the user's needs: Separate, numbered, and flattened triangle-strips or triangles are, for instance, easy to CNC-fabricate.

In any case, the order emerges from the global structure, while the rotation and translation movements are determined locally:

This order is based on organizational principles such as feedback, which are expressed in programming as if/then/else statements.

Understanding feedback as an evaluative response used to control the behavior of the system, its correspondence in programming can be seen as the if/then/else statement, which conditionally, executes a group of statements, depending on the value of the feedback - positive/negative. Does the ***if*** part evaluate to true, which means the feedback is positive, the process continues with an attempt to match the ***then*** part. Otherwise, the ***else*** part is attempted instead.

The UnfoldNurbsSurface script employs in order to distribute a user-defined number of points along the UV-curves on the NURBS-surface the following if/then/else statement:

```
'' Specify discretization in new u-direction of
surface
   nU = Rhino.GetInteger("Specify discretization of
new u-direction", 2, 2)
  If IsNull(nU) Then Exit Sub
```

```
   '' Specify discretization in new v-direction of
surface
   nV = Rhino.GetInteger("Specify discretization of
new v-direction", 2, 2)
   If IsNull(nV) Then Exit Sub
   ''
   '' Get the domain (dimension of parameter space)
of the surface
   If (itype = 1) Then
     '' u-slices
     U = Rhino.SurfaceDomain(strObject, 0)
     V = Rhino.SurfaceDomain(strObject, 1)
   Else
     '' v-slices
     U = Rhino.SurfaceDomain(strObject, 1)
     V = Rhino.SurfaceDomain(strObject, 0)
   End If
   If Not IsArray(U) or Not IsArray(V) Then Exit Sub
```

CLARIFICATION: Considering interaction as a reciprocal action between the parts of a system the UnfoldNurbsSurface script implies interaction between the user and the script in the moment, when the user chooses number of points to be distributed on the NURBS-surface, or UV-direction of the surface discretization.

In this context, the rotation of triangles belonging to the triangulated NURBS-surface into the xy-plane can be seen as an example of emergence, where complex global behavior arises from the interaction of simple local rules:

Based on rules such as [1] Rotate into xy-plane, and [2] Take attached geometry with, the rotation paired with the progressive splitting into ribbons enables flattening of a complex 3D-geometry into a 2D-plane.

REASONING: The flattening of 3D-geometries into 2D is of particular interest for the implementation of digital into physical models. In this context, adaptive triangulated meshes [Delaunay, 1934] allow approximate representation of complex geometries based on NURBS:

While triangles are simply described by three numbers representing the corners between which the triangle spans, NURBS are described by knot vectors, uniform/ non-uniform degrees, etc. and are, therefore, more difficult to operate with mathematically.

NOTES: Notes to this section explain concepts and notions difficult to extract from context.

N3.01 - Adaptive triangulated meshes [Delaunay, 1934] are approximate representations of double-curved surfaces. While triangulated meshes are described as a series of points, NURBS-surfaces are described by UV-points, whereas UV-points represents intersection points of length and width , respectively.

N3.02 - Point-clouds are sets of points in 3D-space describing surface features of 3D-objects.

N3.03 - The three mathematical formulas pertaining to the rotation of vectors in space are part of a complex system of formulas. They serve in this case as an example for math-implementation in scripting.

Section Contributor: R. Schmehl - Scripting in VBScript for SpaceCustomizer.

Acknowledgements: This section has benefited from the contribution of students from the University in Vienna.

3.2 Semi-Automation

3.2.2 *Robotics* N3.04

Not only fabrication but also assembly of building components relies increasingly on digital technology:

Electronic surveying and laser positioning allow for precise placement of building components and robots are successfully used in the construction and assembly process - Shimizu Manufacturing System by Advanced Robotics Technology [SMART] is an automated construction system allowing for erection and welding of structural steel frames, positioning of concrete floor panels, exterior and interior walls [SHIMIZU, 2003].

Interactive architecture, described in the second chapter, relies on robotics as well.

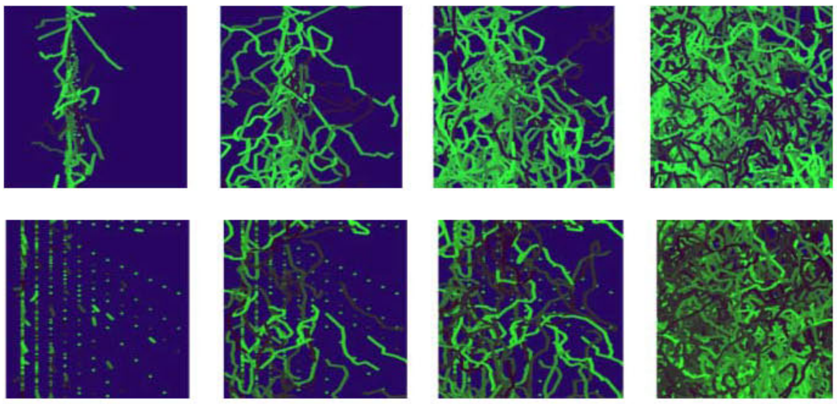
3.3 System-Embedded Intelligence

3.3.1 Software Prototypes: SpaceCustomizer - SpaceGenerator, SpaceInteractivator, GeometryVoxelizer, FunctionLayouter

Based on the premise that only at the moment it incorporates *intelligence* the computer becomes more than a tool, it becomes a reasoning machine [Tzonis, 1993] this research focuses on the development of prototypical tools, which improve the design process by incorporating aspects of intelligence. These refer to Artificial Intelligence [AI] as described in Computer Science implying, inter alia, multi-agency, computer vision, and machinic reasoning using true/false logic to evaluate data.

Multi-Agent Systems [MAS] are in Computer Science distributed Artificial Intelligence systems consisting of several agents capable of reaching collectively goals [Ferber, 1999]. More recently, these systems have become the focus of interest within the discipline of architectural design - largely due to the phenomenon of emergence, which can be seen as the formation of a complex whole from simple constituent parts. Multi-agency, however, has often been misunderstood within the design community, and the principles of working with these systems have remained unknown to many designers:

Within a course on multi-agent systems at TU Delft [2007] with invited guests from Massachusetts Institute of Technology [MIT], students have been introduced to the mechanics of a number of multi-agent design [MAD] procedures with the aim to critically reveal what this technique may offer architectural design, and what challenges remain in its application.



F3.05

Students were introduced to a multi-agent approach by starting with an analog, hands-on approach, later moving into programmatic exercises implemented in the programming environment *processing*.

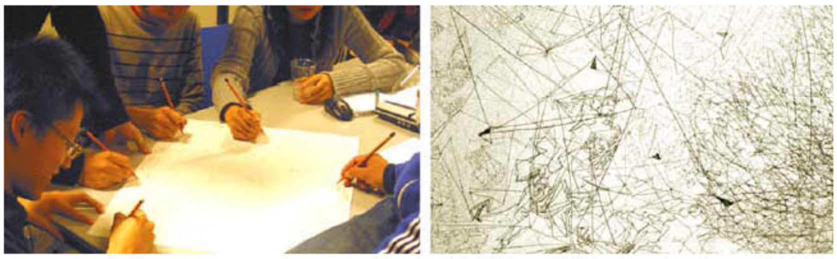
The developed models are simulations of moving particles leaving traces behind, while the employed system is a particle-spring system, which can be seen as a collection of point-masses in 3D-space potentially connected to each other by springs. The system obeys laws of physics, and forces acted on particles include gravitation and friction, whereas springs exert forces on particles according to spring-damping principles.

This system has been successfully employed in architecture for models such as dynamic, interactive hanging-chain models developed by a team of architects, computer scientists and engineers at MIT. Their *model* employs a particle-spring system for representing a structure by applying a gravitational field to it in order to generate its most efficient form. A3.01

According to the developers, MIT's virtual method is as straightforward as Gaudi's physical method for exploring hanging-chain models [Kilian and Ochsendorf, 2005]. In opposition to the MIT-model, the models developed at TU Delft employed the particle-spring engine embedded in *processing* not for developing physical simulations but for developing 2D designs.

The analog model, with which the course started, implied that students worked in groups:

While, one of the students, the programmer, defined the rules according to which each agent/student had to draft, the rest of the group would draft according to rules such as: [1] Put your pen to the center of the paper and draw a triangle; [2] When you finished drawing the triangle, choose a corner of that triangle, which is going to be the starting point of a new triangle. Draw a new triangle; [3] Repeat this step and [4] Vary in size and direction.


F3.06

After simulating analog-wise MAD-principles, students started working in *processing* on digital models. Similarly to the analog models, these implied definition of agents and rules, according to which agents behave. By changing parameters and behavior-rules for agents new systems were generated. Even restarting a simulation with the same parameters for agents and behaviors, would produce, however, different results due to emergence, which arises, obviously, not from the coexistence of the parts but from their interaction.

Section contributors: The workshop has been developed in collaboration with K. Steinfeld, S. Arida and K. DeBiswas.

Acknowledgements: This section has benefited from the contribution of international master students from TU Delft.

In addition to Multi-Agent Systems, Expert Systems [ES] have been explored and developed within this research:

SpaceCustomizer [SC], is an experimental software prototype able to execute semi-automated tasks such as the development of Motion, Mass, and Interactive Spaces, as well as Functional Layouts.

Functioning as an Expert System [ES], SC provides analysis and evaluation of specific design problems, and generates alternative designs. SC, therefore, can be seen as a system operating with knowledge and knowledge-based procedures of an expert in whole or in part [Brown and Chandrasekaran, 1989].

In this context, Motion Spaces [MS] are double-curved 3D-spaces generated by tracing the movement of the human body in space, whereas, the motion map defines boundaries of the volume within which architecture can emerge. The volumetrical outlines of the body in motion establish an initial framework to develop Motion Spaces employing movement studies based on ergonomics.

The initial fundamental question in the development of MS has been: HOW to control designs based on NURBS-geometries? On the one hand it is easy to manipulate NURBS-surfaces by puling control points, on the other hand, the question is how to control their manipulation?

If in this context can be talked about a paradigm shift based on the influence of digital technologies, than this shift can be described in the methodology:

In opposition to modular, repetitive architecture developed by using grids and proportions based on functional and formal rules, curvilinear architecture is being developed within this research by generating space through following the movement of the body in space based on ergonomic principles.

SC : SG

[1] SC : SG SpaceGenerator generates double-curved MS [Motion Spaces] by following the movement of the body in space and can be seen as the Modulor [Le Corbusier, 1948] of the Digital Age since it establishes relationships between the human body and the architectural space:

As a system of proportions Modulor uses measures of the human body in architecture by partitioning it in modules according to the Golden Section and to two Fibonacci Series. It puts, basically, man as measure of architectural spaces, which SC does as well in a more drastic manner, since it generates 3D-space through following the movement of the body in space based on ergonomic principles. While Modulor applies a 2D-proportioning system, SpaceCustomizer employs a 3D DYNAMIC space generating system.

In a time where AFNOR - Association Francaise de Normalisation - proposed standardization in building construction by getting a cross-section of methods used by architects, engineers and manufacturers, Le Corbusier created with Modulor a framework for architectural practice incorporating three intermingling areas - art, nature and mathematics.

In addition, Modulor attempted to facilitate assembly and construction by proposing a modular system to be used in the standardization of industrialized production of building components.

HUMAN SCALE AND ITS RELATIONSHIP TO ARCHITECTURE: Le Corbusier considered the metric system alienated from the dimensions of man, and credited it for the *dislocation* of architecture, implying according to Guerra [1999] that architecture is dislocated in relation to its object, which is to contain men.

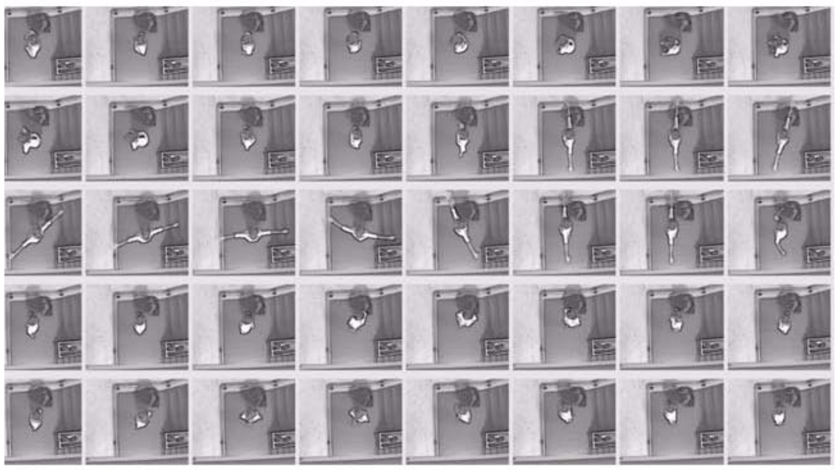
Le Corbusier's critique on renaissance and its abstract set of rules based on mathematics, dealing with icosahedrons instead of using vision, addressed the need to establish a relationship between mathematics, nature, and architecture, which he then implemented in Modulor.

ERGONOMICS: In opposition to Modulor, Neufert's *Architectural Data* [1943] does not propose a modular methodology for mass-production but instead informs about spatial requirements and offers planning criteria regarding function, structure and technical implementation.

However, this data, similar to Modulor's proportions, reduces dynamics of the moving human body in 3D-space to a static, 2D-representation. In contrast, SC departs from ABSTRACTION in order to SIMULATE the movement of the human body in 3D-space taking in consideration ergonomic principles.

Considering simulation as a computer-based representation of not only results but also processes, SC implements dynamic generation of movement and function-driven 3D-spaces in a case-study named THE KIOSK. Initially developed intuitively as a NURBS-model within a workshop at Roma Tre described in the second chapter THE KIOSK has been further tested and developed by means of *space-customization*:

The movement of the body in space has been recorded with digital cameras. From the recorded movies individual frames have been extracted, from which points and curves could be generated manually.

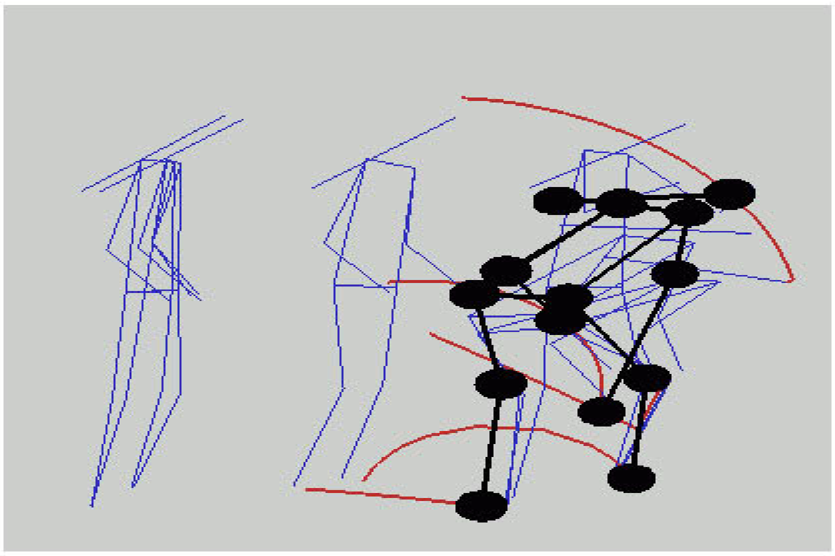


F3.07

The extracted points were, basically, flexure points at waist, neck, arms, shoulders, and elbows, pelvic joint, knees and ankles. These points have been listed by their coordinates in a text file, which later on has been imported into the 3D-space using Rhino.

An alternative method for movement mapping developed by Nakata [2003] has been additionally tested: It involved simulation of a digital human, which is an *information-compressed*, symbolic representation of human body-movement.

Employing an automated generator of movement representation, this simulation features red lines representing trajectories of the edges of the head, arms, and legs, and blue lines representing postures. It visualizes the movement of a digital human using a methodology for dance movement representation developed by Kandinsky [1926].



F3.08

Movements such as walking, bending, lifting, taking a seat, etc. have been simulated and represented digitally for the purpose of MS-generation: These movements have been chosen and developed in accordance to the movements tested in the case-study, THE KIOSK.

DESCRIPTION: The digital human developed by Nakata is defined by flexures - [1] Flexure of the waist joint determined by the angle between pelvis and spine, [2] Flexure of neck joint determined by the angle between spine and cervical vertebrate, [3] Flexures of both upper-arms and, [4] Flexures of the left and right shoulders given by the angle between spine and left/right upper arm. Furthermore, [5] Flexures of left/right elbow defined by the angle between left/right upper arm and left/right forearm, [6] Flexures of both thighs determined by the angle between directions of both thighs, [7] Flexures of left/right coxa: Angle between pelvis and left/right thigh, [8] Flexures of left/right knee determined by the angle between left/right thigh and left/right calf.

Based on Cohen's [1993] and Kestenberg's [1999] theories which describe the relationship between developmental and psychological states, and body movement, Nakata's motion simulator builds up on data gained from registered movements of torso, head and limbs with a VICON motion capture system [Nakata, 2003].

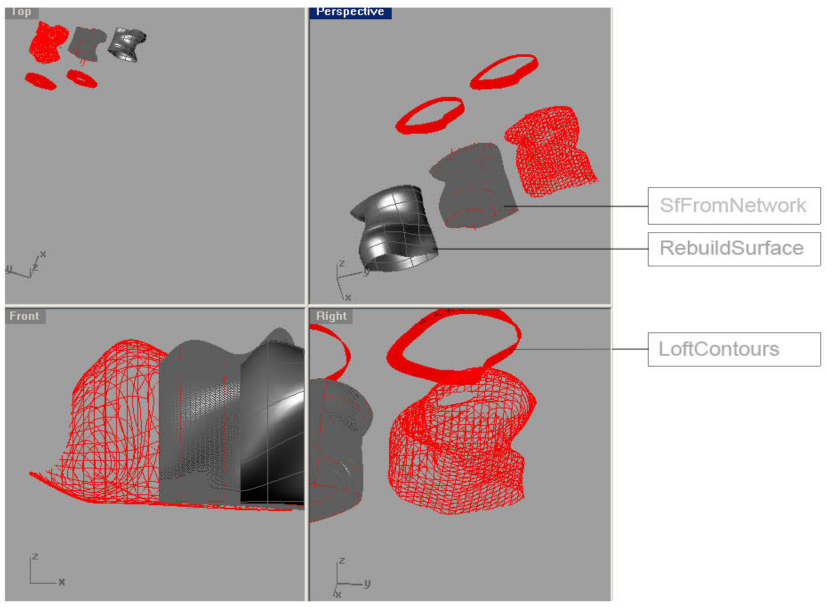
For the purpose of this research, movements were simulated with Nakata's simulator in order to generate curves from which Motion Spaces could be generated in SC : SG and Rhino.

SC : SG [SpaceGenerator] generates Motion Spaces [MS] based on NURBS-geometries. Conceptually speaking, MS are generated by tracking the movement of the human body in space, whereas, the motion map defines the boundaries of the volume within which architecture can emerge.

The SG-script uses, basically, the point-cloud generated with the two movement tracking methods - digital and real moving human. It employs, therefore, point-clouds generated from sequential images of the body in motion, which have been projected on a referential system.

In this context, several methods have been tested for the generation of a NURBS-surface from curves and points:

Rhino commands such as *GenerateSurfaceFromCurvesNetwork*, *Loft-Contours* and *RebuildSurface* have been tested in order to choose the most convenient implementation methodology. However, the use of existing Rhino-commands has proven to be ineffective.



F3.09

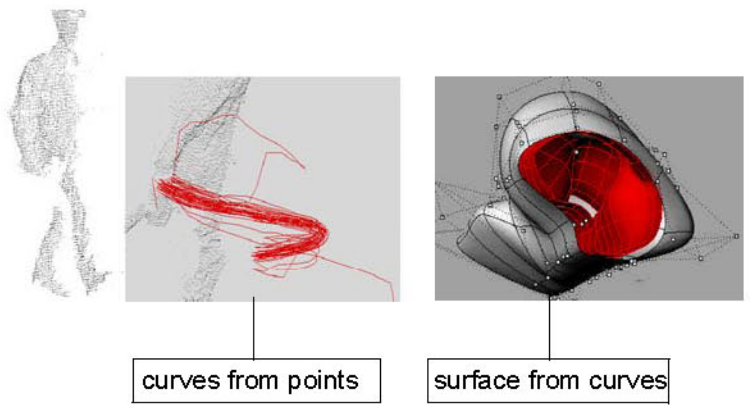
The MS-script has been developed in VBS to generate automatically NURBS-surfaces from a point-cloud imported from a text file. The MS-script has two sub-routines: [1] ImportPoitsFromTextFile, and [2] GenerateSurfaceFromPoints.

The first sub-routine imports the points from a previously generated text file, and the second generates the NURBS-surface by adding a surface from a point-grid.

The scripted MS is employed to adjust and reconfigure the intuitively designed NURBS-space. MS allows, therefore, testing of the 3D-space with respect to the human movement projected into it. The test reveals if and where the space requires reconfiguration in order to accommodate the projected movement.

```
'' Generate NURBS surface from points

'''''''''''''''''''''''''''''''''''''''''''''''''''
  Dim arrCount(1), j
  '' Mark second point as RED   point
  Rhino.ObjectColor arrNames(1), RGB(255,0,0)
  '' Specify discretization in u-direction of sur-
face
  arrCount(1) = Rhino.GetInteger("Discretization of
1st direction (marked by red point)", 2, 2)
  If IsNull(arrCount(0)) Then Exit Sub
  '' Specify discretization in v-direction of sur-
face
  arrCount(0) = Rhino.GetInteger("Discretization of
2nd direction", 2, 2)
  If IsNull(arrCount(1)) Then Exit Sub
  '
  Rhino.Print "Generate NURBS surface"
  Rhino.AddSrfPtGrid arrCount, arrPoints
  '
End Sub
```



F3.10

[2] Alternatively, SI [SpaceInteractivator] generates space interactively by following the movement of the body in space: The input - movement - is being electronically processed in such a way that the output represents a continuous, real-time modification of the space.

For this purpose an on-site-built InterFace employing sensor-actuator technologies enables translation of the recorded movement into spatial configurations. The InterAction between the body and the architectural space gives insight into, HOW the human body shapes space.

Interactive Spaces have been explored in the project SpaceCustomizer : SpaceInteractivator [SC : SI] within a workshop held 2006 at the Henry van de Velde Academy in Antwerp.

BACKGROUND: At the time being large scale architectural projects incorporate interactive systems focusing on light. Kunsthaus in Graz, for instance, has a light installation incorporating a matrix of fluorescent lamps integrated into the acrylic glass façade enabling display of movies and animations.

Small scale installations seem to target more complex configurations such as dynamic, interactive systems: Glynn's project Reciprocal Space [2005], for instance, is a room where the walls change shape in response to inhabitant's movements.

F3.11

Similarly, Poenisch's Dynamic Terrain [2005] is an interactive surface, which changes shape in correspondence to spatial and/or bodily requirements in real time. Other projects such as Decoi's Aegis Hyposurface work with triangulated surfaces in a similar way. However, Hyperbody's MuscleTower adds 3D-dynamics to the movement by enabling rotation and torsion of the structure.

All these projects work with input, processing and output tools, such as sensors, camera tracking systems, projectors, speakers, and software such as Macromedia Shockwave, Max/MSP, and Virtools.

SC : SI uses Max/MSP, camera tracking, and projection to study and implement double-curved space generation by following the movement of the body in space.

In opposition to the horizontal and vertical surfaces employed in the previously mentioned examples, SC : SI employs a double-curved cylindrical space surrounding the body in movement.

CONTENT: The interactive processes in SC : SI are controlled with software developed by K. de Bodt and J. Galle in Max/MSP, which is a graphical programming environment to create software using a visual tool-kit of objects.

The basic environment that includes MIDI, control, user interface, and timing objects is called Max. On top of Max are built objects such as MSP, which is a set of audio-processing objects enabling interactive filter design, hard disk recording, and Jitter, a set of matrix data-processing objects optimized for video and 3D-graphics.

F3.12

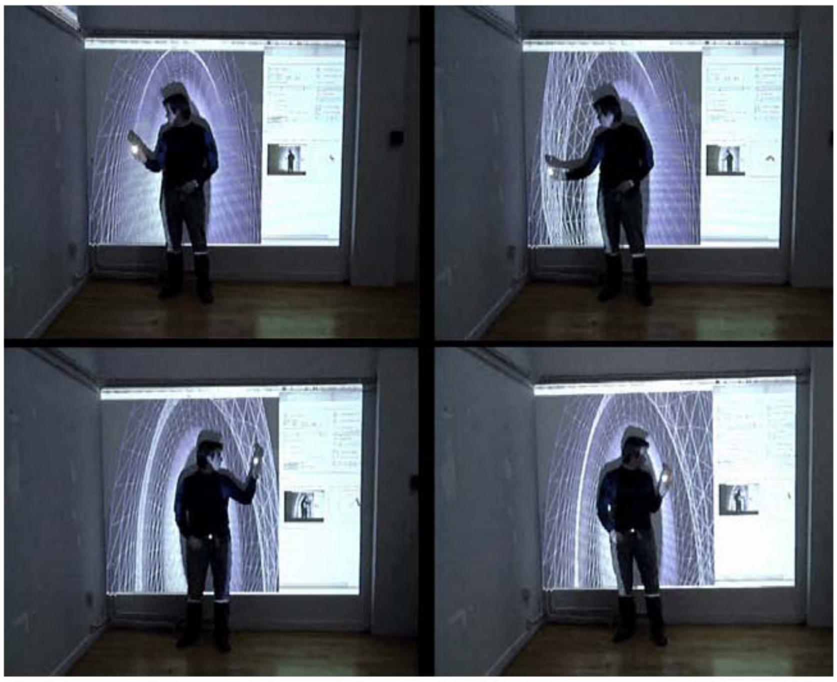
The interactive environment has been developed for transcribing the movement of the body into 3D-space based on SpaceCustomizer [Bier, 2006], which can be seen as the Modulor [Le Corbusier, 1948] of the Digital Age, since it establishes relationships between the human body and the architectural space.

SC : SI implements this concept on the level of transcription of movement into space by interactive means:

The initial space represents the minimum space a standing person needs, which is an ellipsoidal cylinder. The deformation of this cylinder follows accurately the movements of the human being in 3D-space.

IMPLEMENTATION: The ellipsoidal cylinder has been divided in five segments, while the ellipse itself in divided in eight sectors. Each of the eight sectors is being activated, when movement in this area is detected. This means the left/up movement of the arm triggers a deformation in the corresponding sector. The actual movement is being tracked by using a color/movement tracking technique, which involves several steps:

A camera captures body movements and generates image sequences from which movement data is being extracted. This movement activates the spatial deformation in a direct manner by inducing a proportional deformation of space. The space enlarges to accommodate the body in movement.



F3.13

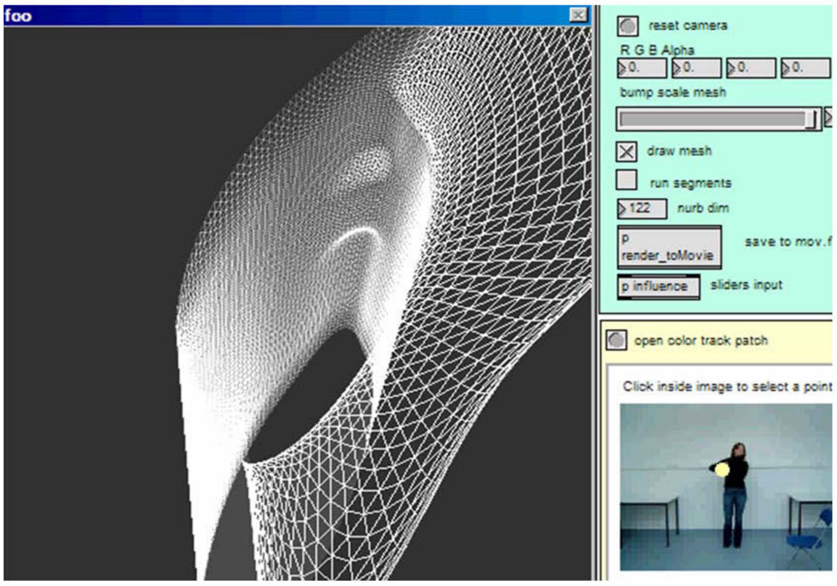
Geometrically speaking, the tracking of movement is based on the conversion of Cartesian coordinates of the tracked points into polar coordinates, while the deformation principle is based on NURBS, which is a mathematical model for generating and representing curves and curvilinear surfaces.

Editing NURBS-based curves and surfaces is easy: Control points are connected to the curves and/or surfaces in a way that their pulling or pushing induces a proportional deformation. While it is easy to manipulate NURBS-surfaces by puling control points, the question has been HOW to control this manipulation? SpaceCustomizer proposes a NURBS-manipulation based on the movement of the body through space.

The interactive manipulation of space is monitored on two interfaces: One of them is projected on a wall the other one is shown on the computer display. While the projected interface serves as an interactive representation and monitoring device, the interface on the computer screen enables control of spatial deformation by means of parametrical change of mesh resolution, color, and NURBS-dimension.

The computer screen interface has a display window for rendering, one main and several sub-patches, which contain programming packages. The user works mainly on the main patch. By starting the program, the processing of data, which is fed into the system, is initiated, and the NURBS-surface is being rendered in the display window.

Interface elements on the main patch are: [1] On/off switch, which is a toggle sending a trigger signal every 20 milliseconds; [2] RGB alpha - Controls the color of the NURBS-surface. [3] Bump-scale mesh - Slider to control the relative amount of the deformation. The input can be chosen for a smaller or larger deformation effect - The default is 1, which represents no scaling, while scaling corresponds to input multiplication up [>1] or down [<1]; [4] Draw mesh - Switches between shaded and wire mesh rendering; [5] NURBS Dim - Slider to reset manually the NURBS UV-mesh density; [6] Video window - 320 x 240 display for the camera image.



R G B Alpha: 0. 0. 0. 0.

[✓ draw mesh]

run segments

122 nurb dim

p render_toMovie — save to mov.f

p influence — sliders input

F3.14

PROGRAMMING: Max/MSP is a graphical programming environment for multi-media, used to design cross-platform programs and user interfaces. Programming takes place in the Patcher-window, where Max/MSP objects, represented as boxes, are connected with patch-*cords*. The program library includes several objects to perform a wide range of tasks, from adding two numbers together to wave-form editing.

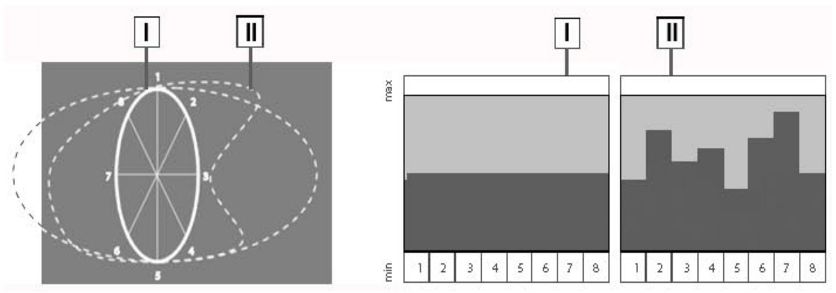
SC : SI consists of three patches: [1] 3D-Shape, [2] Deformation, and [3] Movement Tracking.

[1] 3D-Shape: This patch implements 3D modeling in openGL. It is, basically, a rendering patch, enabling NURBS-representation in real-time. The 3D-shape itself has been developed by following a several steps procedure: The *jit.gl.nurbs* object has been used to generate the cylindrical shape, from which the ellipsoidal cylinder has been derived by scaling it down to 1/3 in the y-direction.

An 8 x 5 *jit.matrix* has been mapped onto the control-points of the NURBS-surface, in a way that the cylinder is divided in five sections and each section is subdivided into eight sectors. This enables an accurate implementation of shape deformation according to the movement, as every subdivision can be addressed separately.

[2] Deformation: An initial displacement matrix establishes the way the movement is translated into shape-deformation - F3.15. The sections 1-8 of the ellipse are mapped into the displacement matrix in a way that a row represents the eight sections of the ellipse, while the degree of displacement of each section is shown in the corresponding min-max columns.

For instance, the initial ellipse - I - is represented in the displacement matrix as corresponding to a middle value, while the deformed ellipse - II - is shown as an alternation between middle and maximum values of displacement.



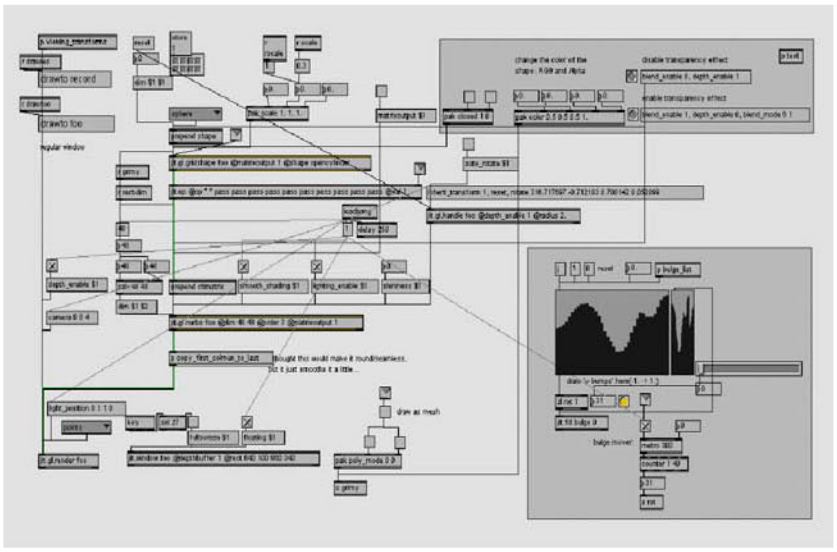
F3.15

[3] Color Tracking: The movement tracking in real-time has been implemented by means of computer vision, which employs color tracking performed with *cv.jit.track*, which is an external object for Max. It extracts *xy* coordinates from the movement and sends them to the Deformation patch, which in turn executes the shape deformation itself.

The color to be tracked is being selected by clicking with the mouse in the video frame window, which shows the real-time movements captured with the camera connected to it. The Cartesian coordinates of the tracked color/point are then converted into polar coordinates, which find their correspondence in the eight ellipsoidal sections.

CONCLUSION: This exercise in interactivity shows that the concept of responsive environments applied to architecture can be implemented in double-curved spaces, which dynamically react to the movement of the human body in 3D-space.

In this context, emergence and *self-organization* can be seen as principles on which interactive architectures can be based on, as building components dynamically adjust to their users' needs. N3.05



F3.16

Modes of emergence and self-organization in SC : SI are based on *space-customization*, which implies, inter alia, spatial transformations generated through human movement-transcription. These interactive spatial transformations are implemented by means of machinic intelligence implying computer vision, human-computer interaction, and parametric 3D-modeling.

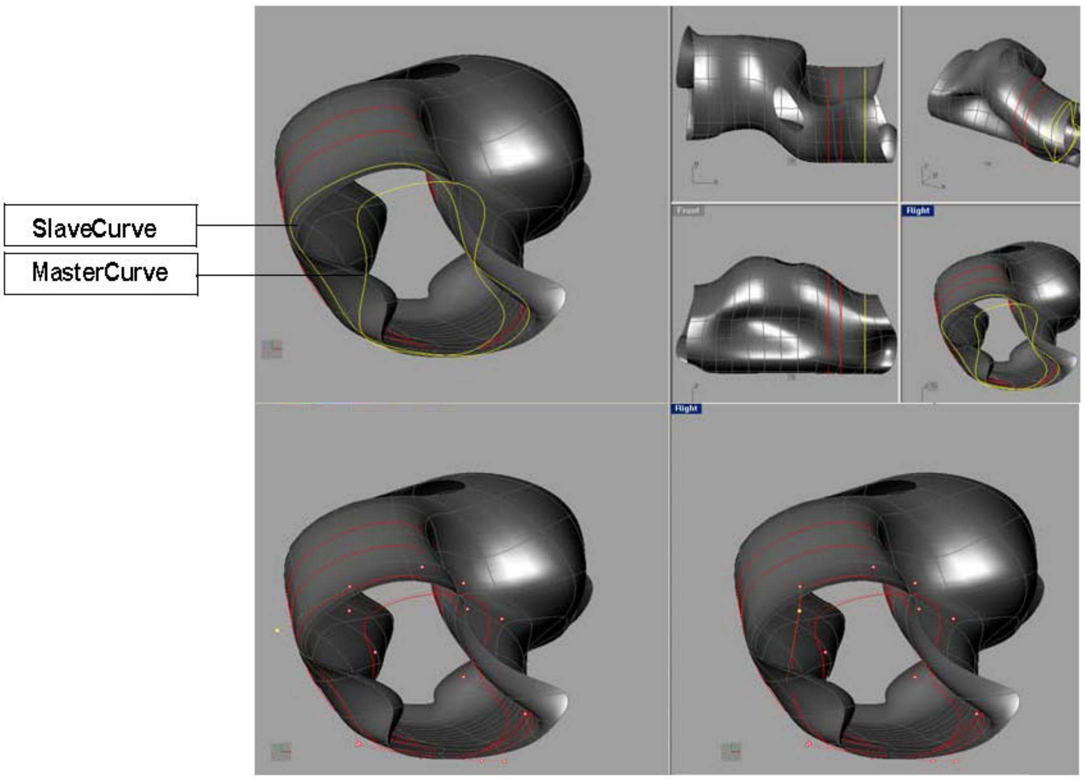
Section Contributors: J. Galle, K. de Bodt - programming in Max/MSP for SC : SI - and R. Schmehl - scripting in VB for SC : SG.

Acknowledgements: SC : SI has benefited from the contribution of students from HIAS in Antwerpen.

Beyond its interactive aspects SC : SI can be seen as a tool to adjust and reconfigure intuitively designed NURBS-based spaces:

Alternatively, the process of spatial adjustment from the intuitively to the computer generated NURBS-space and vice versa has been envisioned to be implemented with a sub-tool SpaceAdjuster [SA] incorporated in SpaceGenerator [SG] *acronymically named* SC : SG : SA. N3.06

N3.07 SA enables adjustment of a double-curved *slave*-surface to a computer-generated *master*-surface.



F3.17

SC : SG : SA has been implemented as a series of functions such as GeometryVoxelizer [GV], VolumeToFunctionAdjuster [VFA], and Nurbs-FromVoxelsGenerator [NVG], which are embedded in the alternatively developed Java-application:

Instead of adjusting NURBS, this application voxelizes the NURBS-space, it adjusts the voxelized space to functions and re-generates it subsequently as NURBS. This procedure has been assessed as most efficient in not only delivering means for volumetrical control of NURBS-based spaces but also facilitating spatial manipulation and adjustment.

CONSIDERATIONS: For the development of the software-prototype dealing with functional layout two alternatives have been taken into consideration:

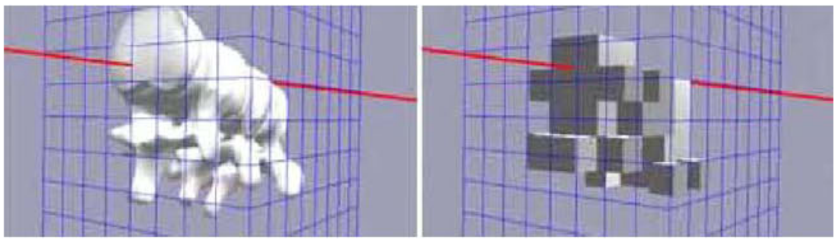
Development of a plug-in for Rhino, which is a commercial software, and development of a Java-application using *processing* library for rendering. The second has been chosen because of its independency from commercial software and on the Internet free available libraries for geometry, physics, and optimization.

SC : SG : SA : GV [GeometryVoxelizer] enables voxelized representation of double-geometries, which can be seen as mass-models in architectural design.

BACKGROUND: Digital representation in the sixties and seventies was based on vector representation, which presented advantages since vectors address any point of the continuous space and exhibit, therefore, no *aliasing* effects. N3.08

The alternative approach, raster graphics, has been introduced in the late seventies and is based on a discretized representation of space namely *pixelized* representation. Similar to 2D-images represented as pixels, 3D-objects represented as voxels - *vo*lumetric pix*els* - have specific resolutions: While low-resolutions objects are facetted, high-resolution objects are smooth.

Voxelized representations have been successfully applied in medical visualization [Thon et. al., 2004] and computer gaming.

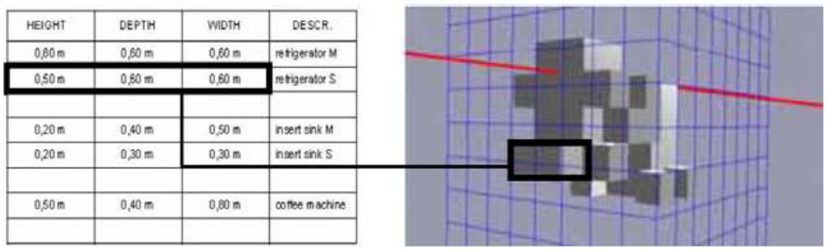

F3.18

In architecture voxelized spaces enable fluent transition from curvilinear-smooth to angular-facetted geometries and can be seen as *mass*-models used for volumetrical and functional studies:

In an iterative process volumes are assigned to functions and spatial relationships are established between the different functional volumes in order to generate 3D-layouts.

IMPLEMENTATION: GV has been implemented in two versions as VB-Script and Java-application.

The VB-scripted GV1 - GeometryVoxelizer1 - has been developed in collaboration with D. Rutten, a program developer for Rhino. GV1 transforms enclosed NURBS-based solids in voxelized geometries - as shown in appendix. The implementation of SC as plug-in for Rhino has been, however, abandoned in favor of developing a Java-application using *processing* for rendering. The first built-in function is the voxelization of double-curved geometries. A3.02



HEIGHT	DEPTH	WIDTH	DESCR.
0,80 m	0,60 m	0,60 m	refrigerator M
0,50 m	0,60 m	0,60 m	refrigerator S
0,20 m	0,40 m	0,50 m	insert sink M
0,20 m	0,30 m	0,30 m	insert sink S
0,50 m	0,40 m	0,80 m	coffee machine

F3.19

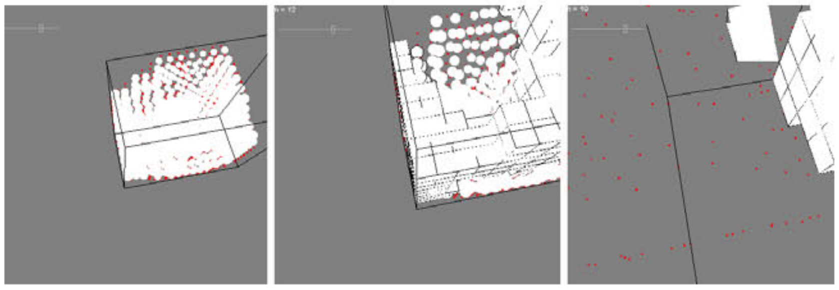
GV2 - GeometryVoxelizer2 - generates voxelized mass-models, which are linked to data-bases informing the space about the contained equipment components:

The SC-database is, basically, a table listing equipment components for a food-kiosk such as refrigerator, sink, dishwasher, coffee machine. An array of different types is allocated to each item so that the choice of a specific type induces automatically a corresponding change in the voxelized model.

Since voxels are quantifying the volume, the advantage of a voxelized geometry is obvious:

The established link between voxels and data-base allows for continuous, real-time update of the corresponding geometry - numerical changes in volume are automatically represented volumetrically and vice-versa.

GV2 is employed, therefore, in the process of adjustment and reconfiguration of functional spaces.



F3.20

DESCRIPTION: In its first implementation, a point-cloud has been exported from the NURBS-model, which has been used for the development of a voxelized 3D-model in *processing*. This voxelized model built in collaboration with K. Steinfeld has been, however, inoperable: Incremental change of resolution would result in inhomogeneous representation of voxelized space featuring gaps between voxels.

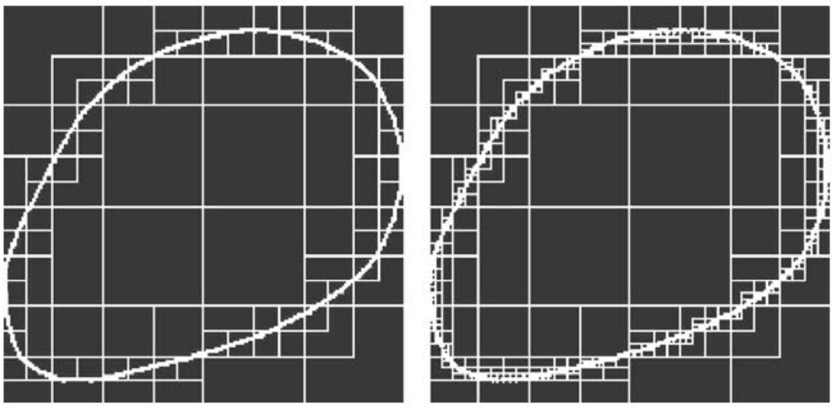
In its second implementation, the NURBS-geometry has been exported as a polygon mesh and has been voxelized in the, within this research developed, Java-application using *processing* for rendering.

[1] Input-geometry: The input-geometry is defined as a polygonal surface-mesh in *obj*-format. During *parsing* polygonal faces are automatically decomposed into triangle faces. This object is in the next step converted into a voxel-mesh in a process of voxelization for polygonal objects:

[2] Voxel-mesh: Resulting from the voxelization of the surface-mesh, the voxel-mesh provides a discrete 3D-space, which is later on popu-

lated with Functional Objects [FO]. The association of an FO with voxels can be maintained on object level - an object *knows* which voxels it is occupying - and on mesh level - a voxel *knows* by what object, if any, it is occupied.

GV2 - developed in collaboration with R. Schmehl - follows the description of a voxelization process for polygonal objects by Thon et. al. [2004] using a *quad-tree* partitioning of the triangle-mesh in order to speed up the ray-casting procedure. This procedure determines the inside/outside status of a voxel based on the location of the voxel center coordinate relative to the triangle-mesh. N3.09



F3.21

The quad-tree has been used to partition the space by recursively subdividing it into rectangular quadrants. This enables pre-computation of ray/triangle intersections and, therefore, accelerates the process of ray-casting by pre-determining, which triangles intersect with rays.

Rays intersecting the object determine, therefore, inside/outside conditions and generate voxels accordingly [Bier and Schmehl, 2008].

In its final implementation, GV2 incorporates voxel-resolution *sliders* and S/M/L buttons: N3.10

While sliders enable continuous, low-high resolution voxel-representation, S/M/L buttons show three resolutions representative for the study 30/30/30, 60/60/60 and 90/90/90 cm.

Voxelization resolution ranges from 5-90 cm enabling relative accurate representation of the curved geometry.

Section Contributors: R. Schmehl, D. Rutten, and K. Steinfeld - scripting and programming for GV.

Acknowledgements: Movement studies have been implemented with S. Bouten and A. Talic from TU Delft and the case study - THE KIOSK - has been developed with M. Gulyas, K. Orehounig and E. M. Streit from the University of Vienna.

SC : SG : SA : VMA [VolumeToMotionAdjuster] enables a voxelized geometry, which can be seen as mass-model, to update according to data-input from a database. It also enables adaptive voxel refinement based on local resolution-needs.

SC : SG : SA : NVG [NurbsFromVoxelsGenerator] generates, as its names suggests, NURBS-surfaces from periphery voxels.

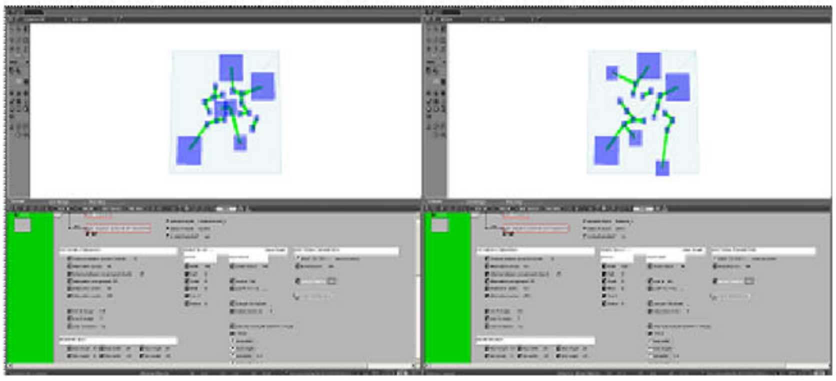
Both sub-tools, VMA and NVG have been, however, developed within this research only conceptually and have been not implemented into software prototypes.

The third SC-tool developed conceptually and implemented prototypically within this research is focusing on functional layout:

[3] SC : FL [FunctionLayouter] provides generation and optimization of functional 3D-layouts by employing constraint solving techniques.

SC : FL

PRELIMINARY STUDY: BuildingRelations [BR] has been developed with Hyperbody-students from TU Delft [2006]. It deals with generative and parametric design concepts as well as interactivity principles.



F3.22

Focusing on the development of an interactive design tool, which allows simulation of complex design processes, the project proposes an alternative design method based on swarm behavior:

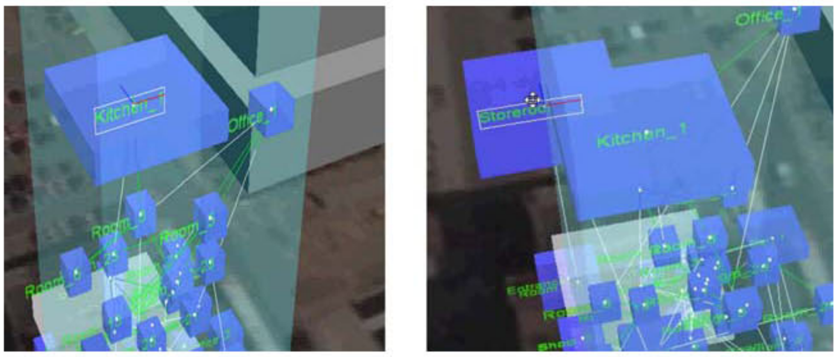
BR consists of *agents* interacting locally with one another and with their environment similarly to the way fish interact in a swarm and birds in a flock, respectively.

In the absence of top-down control dictating, how individual agents should behave, local interactions between agents lead to the bottom-up emergence of global behavior.

The rules according to which agents are interacting are simple: Reynolds' flocking simulation, for instance, is based on three rules according to which digital birds, named boids, are flocking - [1] Maintain a minimum distance to vicinity, [2] match velocity with neighbors, and [3] move towards the center of the swarm. While these rules are local establishing the behavior of one agent in relationship to its next vicinity, the flock behaves as a whole, coherently [Reynolds, 1987].

Similarly, all functional units pertaining to a building can be seen as flocking agents striving to achieve an optimal spatial layout. In this context, spatial relations between functional units can be described as rules, according to which all units organize-themselves into targeted configurations. This approach is particularly suitable for the functional layouting of complex structures:

While the architect might find it difficult to have an overview on all functions and their attributed volume and preferential location, functional units can easily *swarm* towards local optimal configurations.



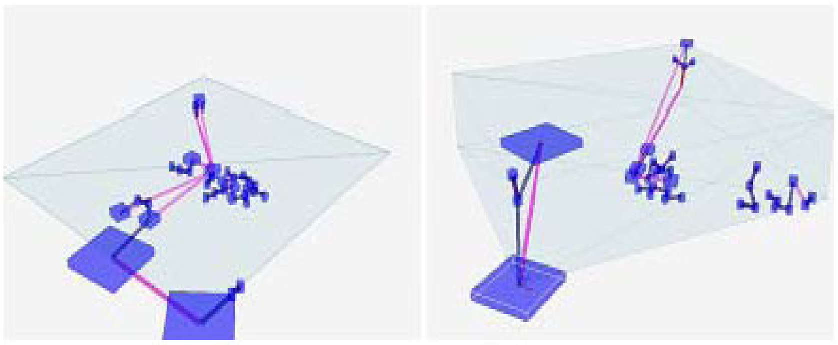
F3.23

Functional layouting in architectural design deals with the placement of functions in 3D-space, whereas building components such as rooms have no fixed, pre-defined dimensions, and are resizable.

Attempts to automate the process of layout incorporate approaches to spatial allocation by defining the occupiable space as an orthogonal 2D-grid and use an algorithm to allocate each rectangle of the grid to a particular function. Other strategies break down the problem into parts such as topology and geometry:

While topology refers to logical relationships between layout components, geometry refers to position and size of each component of the layout. A topological decision, for instance, that a functional unit is adjacent to another specific functional unit restricts the geometric coordinates of a functional unit relative to another [Michalek et al., 2002].

Based on a similar strategy BR generates solutions for complex layouting problems in an interactive design process. Furthermore, it operates in the 3D-space and therefore, it represents an innovative approach to semi-automated design processes.



F3.24

The developed software prototype consists of several sub-tools such as [1] SizeDefiner, [2] FunctionsDistributor, and [3] BoundingBox:

[1] SizeDefiner is a sub-tool, which establishes interactively dimensional relationships and constraints for building components. It is based on data originating from Building Regulations [DBR, 2003]. These define rules and restrictions regarding minimal floor-space areas per person.

The SizeDefiner script relies, therefore, on building regulations concerning the size of a space in relation to its occupants and its specific function. These define minimum required floor areas and occupancy numbers in relation to the allocated function.

In its first version SizeDefiner receives the input from the Building Regulations database and the user/designer, who defines number of people occupying the space; SizeDefiner generates than accordingly the space and scales it to fit the minimal size needed according to those regulations.

A more advanced version incorporates additional functions enabling the user/designer to adjust amount of floors, adjust floor heights and set the width and length of the space by overruling regulation constraints, if needed.

The autonomous working of the script requires aspects of intelligence, which in this case rely on a simple strategy:

Spatial units establish relationships with other spatial units by determining their in-between distance and automatically adjusting their width, length and height in order to prevent potential overlaps/collisions. Spatial units, therefore, adjust themselves to their surroundings. They are linked, therefore, to other units creating spatial relations defined and simulated with another sub-tool:

[2] FunctionsDistributor takes, basically, a program of requirements - number of specific spaces, their occupancy numbers, and building regulation classes - and translates it into an ordered spatial layout. This

order is achieved by defining distances between objects of the same type and objects of different type, meaning that same type objects are clustering while not related type objects disperse.

The development of this sub-tool involved several steps pertaining spatial configuration:

[2.1] Organization: Functional objects have configurable distances to each other enabling them to group together or to spread out depending on which group they belong to.

[2.2] Structure: Objects are square or spherical and their orientation is in a grid-like or radial structure, respectively. This enables development of specific layouts.

[2.3] Representation: Relations between objects are represented as magenta lines, when relations are established between grouped objects. Green lines indicate nearest object relations. This representation enables reading and evaluation of the dynamic 3D-diagram.

[2.4] Databases: FunctionsDistributor uses three arrays pertaining to the program of requirements, building regulations, which defines dimensions of the objects, and the master array, which is a dynamic array containing all data about functional objects.

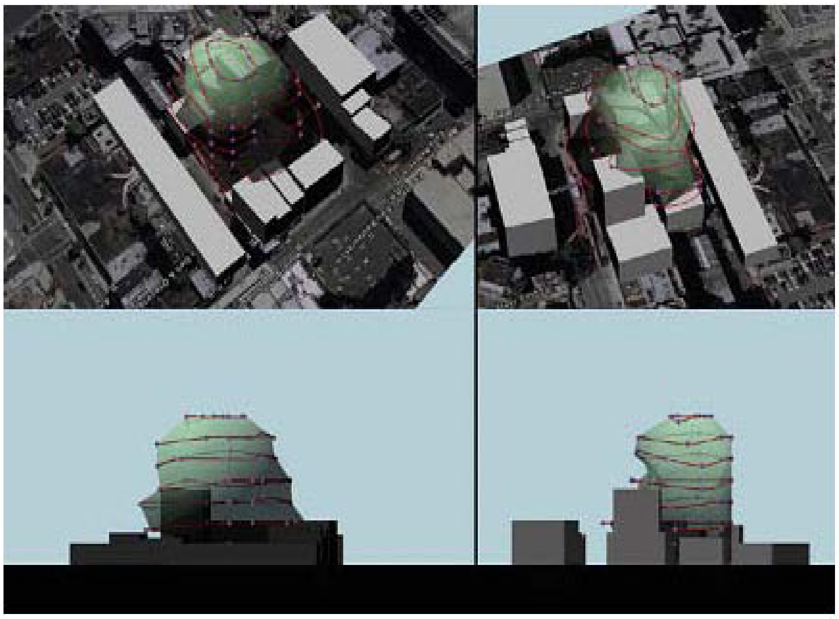
[2.5] Min-max component number: Several tests have been implemented in order to define a min-max number of spatial units. The maximum number of components has, however, dropped from 100-150 to 50-100 due to increasing complexity.

Placement of all functions in 3D-space is controlled by FunctionsDistributor:

SizeDefiner generates functions depending on their Building Regulation values as well as their external placement defined by FunctionsDistributor or by the user/designer.

This system enables adjustment of spaces to their surrounding spaces, whereas distances to other objects are defined by:

[1] Bounding-box, in which all functional objects are supposed to fit in, establishing 3D-boundaries for the building to be designed. [2] Preferred distance to the nearest functional object, preferred distance to the nearest object within its own functional group, and preferred distance to *center* and inside/outside boundaries follow the *elastic-cord* principle: The bigger the distance, the harder the object tries to get to the preferred distance. And [3] collision detection enforces at the moment an object touches another object that they both move for 0.1 second in the opposite vectorial direction.



F3.25

These self-organization mechanisms are complemented by interactivity: The layouting process does not take place outside of the influence of the user/designer. The user can select objects and move them to other places and the model re-adjusts to the new configuration. By clicking on an object, the user can *free* the object from a specific position enabling it to participate in the simulation all over again. In this way, the system and the user/designer search for a preferred layout of functions.

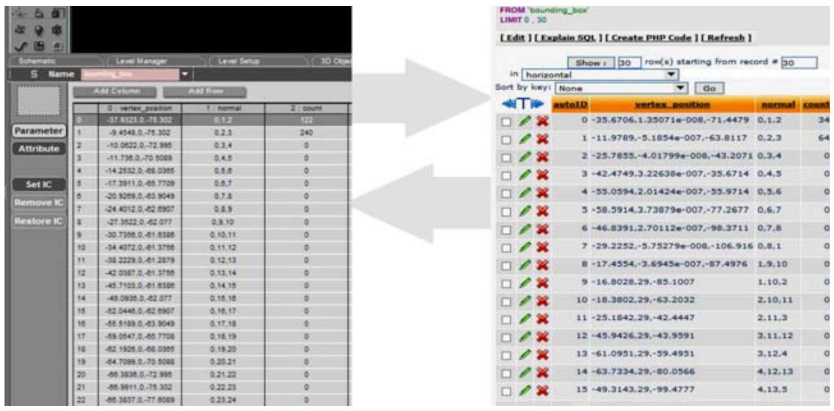
A third sub-tool, already mentioned, is the Bounding-Box [3], which establishes boundaries within which functional objects position themselves. It contains real-time editing features, which enable form-finding processes pertaining to surface definitions such as NURBS and polygon-meshes.

This sub-tool converts the data defining min-max areas, min-max floor heights, into a geometric model by creating a shape according to predefined square-meter requirements. By dragging control points the model is recalculated to stay within the pre-determined boundaries, while changing shape.

BR receives and sends continuously data from and to the database, which contains all information regarding which group the objects belongs to, to which other groups they may relate to, etc. Functional units are, therefore, described by their building regulation type, their name, scale and position, their occupancy, their number of floors, their condition such as active - free to move - or inactive - fixed.

These values or combinations of values can be used by other sub-tools running at the same time. Furthermore, they can be exported to other programs: Positions, dimensions and scale of objects can be exported to other 3D modeling programs such as Rhino, Maya, etc.

The database, therefore, establishes connectivities between different software and functions as a parameter-pool containing geometric and functional data: A 3D-model developed with the BoundingBox sub-tool could be saved to an online database, for instance, from which FunctionsDistributor would take data to generate a functional model within the parameters defined by the 3D-model.



	0 : vertex_position	1 : normal	2 : count
0	-37.9323,0,-75.302	0,1,2	122
1	-9.4548,0,-75.302	0,2,3	240
2	-10.0622,0,-72.995	0,3,4	0
3	-11.736,0,-70.5089	0,4,5	0
4	-14.2532,0,-68.0365	0,5,6	0
5	-17.3911,0,-65.7709	0,6,7	0
6	-20.9269,0,-63.9049	0,7,8	0
7	-24.4012,0,-62.6907	0,8,9	0
8	-27.3622,0,-62.077	0,9,10	0
9	-30.7366,0,-61.6386	0,10,11	0
10	-34.4072,0,-61.3756	0,11,12	0
11	-38.2229,0,-61.2879	0,12,13	0
12	-42.0387,0,-61.3756	0,13,14	0
13	-45.7103,0,-61.6386	0,14,15	0
14	-49.0936,0,-62.077	0,15,16	0
15	-52.0446,0,-62.6907	0,16,17	0
16	-55.5169,0,-63.9049	0,17,18	0
17	-59.0547,0,-65.7706	0,18,19	0
18	-62.1926,0,-68.0365	0,19,20	0
19	-64.7099,0,-70.5088	0,20,21	0
20	-66.3836,0,-72.995	0,21,22	0
21	-66.9911,0,-75.302	0,22,23	0
22	-66.3837,0,-77.6089	0,23,24	0

```
FROM 'bounding_box'
LIMIT 0 , 30
```

[Edit] [Explain SQL] [Create PHP Code] [Refresh]

Show : 30 row(s) starting from record # 30
in horizontal
Sort by key: None Go

autoID	vertex_position	normal	count
0	-35.6706,1.35071e-008,-71.4479	0,1,2	34
1	-11.9789,-5.1854e-007,-63.8117	0,2,3	64
2	-25.7855,-4.01799e-008,-43.2071	0,3,4	0
3	-42.4749,3.22638e-007,-35.6714	0,4,5	0
4	-55.0594,2.01424e-007,-55.9714	0,5,6	0
5	-58.5914,3.73879e-007,-77.2677	0,6,7	0
6	-46.8391,2.70112e-007,-98.3711	0,7,8	0
7	-29.2252,-5.75279e-008,-106.916	0,8,1	0
8	-17.4554,-3.6945e-007,-87.4976	1,9,10	0
9	-16.8028,29,-85.1007	1,10,2	0
10	-18.3802,29,-63.2032	2,10,11	0
11	-25.1842,29,-42.4447	2,11,3	0
12	-45.9426,29,-43.9591	3,11,12	0
13	-61.0951,29,-59.4951	3,12,4	0
14	-63.7334,29,-80.0566	4,12,13	0
15	-49.3143,29,-99.4777	4,13,5	0

F3.26

BR, therefore, is being used interactively and in combination with other software, to achieve non-deterministically designs. It is a design support system, since it supports the user/designer in the functional layouting process rather than prescribes a solution.

Its current implementation, even though diagrammatic, demonstrates an obvious capability to support functional layouting of large and complex buildings based on swarm principles, ON CONDITION THAT global optimization mechanisms are incorporated into it - which is the next step in the development of this software.

The inability to achieve a consistent functional layout with this software prototype goes back, in part, to the definition of local optima:

The software generates *endlessly* many local optima. It does not, though, generate a global optimum.

The further development of this tool aims, therefore, to incorporate global optimization mechanisms. It also aims to address issues related not only to functional organization of space but also issues of spatial coherency.

Section Contributors: D. Hoffers, M. Frederiks, and S. Korebrits scripted in Virtools SizeDefiner, FunctionsDistributor, and BoundingBox, respectively.

Acknowledgments: BuildingRelations has been developed and implemented with Hyperbody-students M. Frederiks, S. Korebrits, D. Hoffers, J. Pointl, M.C. van der Marel, R. Siemerink, T. Godbersen, and W.S. Slot.

N3.11 CONSIDERATIONS: Local search algorithms are *incomplete* algorithms, as the search might stop even if the solutions found are not optimal [Orlin et al., 2004]. While these algorithms move from solution to solution in the search space of candidate solutions until a solution is found, *optimal* solutions might lie far from the neighborhood of the solutions explored by the search algorithm.

In addition, design problems have large numbers of local optima. While, finding an arbitrary local optimum is relatively easy by using local optimization methods, finding the global optimum of such a function is challenging [Ghallab et al., 2004].

For purposes of optimization, such a function has to be defined, according to Ghallab et al. [2004] over the whole domain and must have a range of local optima, from which a global optimum can be selected. By contrast, a local optimum represents a solution from a selection of neighboring values.

Hybrid techniques, therefore, seem to be more efficient when applied to automated planning problems.

N3.12 PRECEDENTS: Two systems for automated 2D-layout design based on *Constraint Satisfaction* [CS] techniques have been compared by Fleming et al. [1992]. While one of the systems - LOOS - uses a form of *generate-and-test* constraint satisfaction and the other system - WRIGHT - uses *disjunctive* constraint satisfaction.

According to Fleming et al., both have an under-constrained general problem definition and, therefore, both produce an unmanageable large amount of feasible solutions. Loos adds objects sequentially, while Wright satisfies constraints incrementally. When tested and compared both generate similar solutions for the same problem.

According to the authors, disjunctive constraint satisfaction is more efficient, but less general than hierarchical generate-and-test constraint satisfaction regarding criteria it can incorporate. However, both can incorporate features of the other approach and overcome their limitations [Fleming et al., 1992].

IDIOM, a third constraint solver for 2D-layouts [Lottaz, 1998] aims to find globally consistent and complete solution spaces in an interactive design process.

Similarly, SC : FL - FunctionLayouter - solves layout problems by exploring large solution spaces and achieves this by reducing the search space, and by exploring it efficiently.

[1] The search space is reduced by applying *heuristics*: From previous spatial studies executed by conventional means such as sketching, it becomes obvious that certain parts of the available space are difficult to access or too small to accommodate Functional Objects [FOs]. This space has been deducted from the total amount of space enabling a search space reduction of almost 1/2. And [2] efficient exploration of the search space is ensured by employing search algorithms based on SAT-solving techniques. N3.13

IMPLEMENTATION: SC : FL relies on constraint-based modeling, which not only specifies geometry and content of objects but also generates object-layouts using constraint-solvers.

.
Constraint-based modeling systems - employed in parametric design tools - contain in addition to 2D-3D objects, constraints that are stored, displayed, edited, and solved correspondingly. In this context, persistent constraints relationships among modeled parts are maintained in the modeling process until further editing.

Constraint-solvers, therefore, exhibit an elevated level of complexity: SC : FL, for instance, generates solutions by applying constraint and optimization algorithms to the layout problem.

SC : FL incorporates aspects of intelligence: While global searches go systematically through all instances of solution-spaces, local searches go through some but not all instances of possible solutions. SC : FL, however, is not only a global but also a complete search engine, since able to find if an assignment is possible or not.

SC : FL employs constraints in order to direct the search for feasible and optimal solutions and it has been implemented in two versions SC : FL90 and SC : FlexFL. Both, address functional layout-problems for architectural designs based on curvilinear geometries:

While rather easy to manipulate formally, NURBS-based spaces are difficult to control with respect to allocation of functions in 3D-space. Therefore, in a first step the NURBS-based space is voxelized.

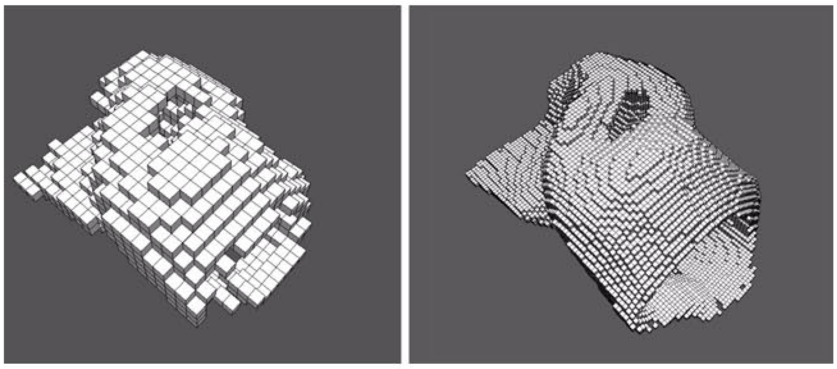
Voxelization within this project enables continuous, low-high resolution voxel-representation of the space within a 5-90 cm range.

SC : FL90 uses, basically, an abstracted problem definition. Equipment components are placed in a voxelized KIOSK-space with a resolution of 90/90/90 cm by applying heuristics such as:

Equipment is placed on two levels easily accessible for the kiosk-users: 0.00-0.90 and 0.90-1.80 m and the search space is, therefore, reduced to those two layers of voxels relevant for the search.

Furthermore, equipment is defined as 90/90/90cm units, ensuring a 1:1 mapping of FO to voxel, which is, as mentioned before, a simplified model.

In a second iteration, Flexible FunctionLayouter [FlexFL] drops the 1:1 mapping constraints, allowing FOs to span multiple voxels. This enables, inter alia, flexible voxel resolutions, which have been tested on a 30/30/30 cm resolution case study wherein FOs can span multiple voxels, increasing complexity in FO allocation with respect to geometry.



F3.27

In the original FL90 test-case, a food kiosk has been modeled, in which a total number of 14-26 functional objects have been assigned to 26 voxels. In this context, 11 functional object types have been differentiated: Refrigerator [RF], sink [SK], stove [ST], exhaust [EX], automat [AT], storage room [SR], trash bin [TB], dish-washer [DW], coffee machine [CM], micro-wave [MW], and cash desk [CD]. These FOs represent a typical equipment selection for a food-kiosk.

In addition, allocation constraints have been formulated implying definition of most effective functional spatial configurations:

These are in part based on empirical findings formalized by Neufert [2005] as well as the kitchen work-triangle defined by the Building Research Council [1993] at the University of Illinois, which specifies that SK, ST, and RF form, preferably, a triangle in which the closer the length of the triangle sides is to 200 cm, the better is the layout.

The FlexFL test-case uses a higher resolution model 30/30/30 cm, which implies a higher accuracy in geometrical representation, and therefore, a reduction of available space for placing FOs:

Instead of 14-26 FOs FlexFL30 allocates 10 FOs with sizes defined by their corresponding height/width/depth:

SK 60/60/30, ST 60/60/15, RF 60/60/60, AT 60/60/60, SR 60/60/60, MW 45/45/45, DW 45/45/45, TB 45/45/60, EX 45/45/15, CM 45/45/15, and CD 45/45/15. Sizes are, in this case, simplified but realistic assumption for usual FO-dimensioning. FlexFL30 deals, therefore, with a nearly realistic problem description, while FL90 deals with an abstracted one.

METHODOLOGY: The layout-problem has been addressed in SC : FL by master students at the Computer Science Department at TU Delft. They employed constraint propagation with separate optimization using *SAT*-solving techniques.

SAT refers to Boolean Satisfiability, which determines if the variables of a given *formula* can be assigned in such a way that the formula evaluates to a Boolean value: TRUE or FALSE.

The constraint solver used for solving the described layout-problem is MiniSAT+. This solver employs a systematic *backtracking* search procedure to explore a space of variable assignments looking for satisfying assignments [Sorensson, 2005] referred to as the Davis-Putnam-Logemann-Loveland [DPLL] algorithm. N3.14

MiniSAT improves the basic DPLL search algorithm by employing efficient *conflict analysis*, *clause learning*, *non-chronological backtracking,* and *random restarts* [Dechter, 2003]. N3.15

Without going into detail, backtracking - for the previously described layout problem - can be defined as an algorithm, which explores each possible combinations of FOs layout:

During the search, when a combination does not work, the search backtracks to the previous constellation, which presented possible alternatives, and explores them. When these alternatives are exhausted, the search returns to the previous constellation and explores all possible alternatives from there. When all constellations are explored, the search is complete [Gurari, 1999].

FL ARCHITECTURE: FunctionLayouter [FL] is based on a sequence of operations. Initially, the problem is read from an *XML* problem description. There, the problem is split into a number of rules and optimization targets. The problem is then translated into pseudo-Boolean constraints [Bier, et al., 2007] which are run through a modified version of MiniSat+. N3.16

The SAT solver's output is then parsed and translated into solutions. Based on the optimization target, new constraints are added, and MiniSat+ is invoked again. Finally, the solutions are displayed on a graphical user interface, where the user can request more information on a specific solution, again invoking the constraint translation system.

In this context, the main focus has been the translation of constraints from the specifications as given by user/designer, to the pseudo-Boolean constraints solvable by MiniSat+.

FL CONSTRAINTS: FL deals with a series of constraints such as cardinality, adjacency and design constraints.

Cardinality related constraints refer to max-min voxel occupancy in relationship to allocated functional objects:

- At least 2 voxels and at most 6 voxels should be assigned to AT.
- At least 2 voxels and at most 4 voxels should be assigned to SR and RF.
- At least 1 voxel and at most 2 voxels should be assigned to TB, EX, SK, and ST.
- Precisely 1 voxel should be assigned each DW, CM, MW, and CD.

Generic design constraints are not specific to a case but rather generic to all KIOSK-layouts:

- SK must be adjacent to DW and TB.
- If a voxel is assigned a ST at least one adjacent voxel must be

assigned a SR.
- RF and ST can not be adjacent; SK and ST can not be adjacent.

Triangle rules - Place RF, ST, and SK in a triangle, so that:

- Each triangle side ranges between 300-65 cm and triangle perimeter ranges between 865-400 cm.

Adjacency and occupancy rules describe constraints referring to neighboring relations such as:

- RF can be adjacent to RF, AT, DW, CM, MW, TB, CD, SR, EX, and can occupy voxels on levels 0.00-0.90 and 0.90-1.80.
- SK can be adjacent to SK, AT, DW, TB, SR, CD, and can occupy voxels on level 0.00-0.90.
- ST can be adjacent to ST, AT, DW, TB, SR, CD, and can occupy voxels on level 0.00-0.90.
- EX can be adjacent to RF, MW, AT, CM, SR, and can occupy voxels placed above ST on level 1.80.
- AT can be adjacent to AT, ALL FOs, and can occupy voxels on levels 0.00-0.90 and 0.90-1.80.
- SR can be adjacent to SR, ALL FOs, and can occupy voxels on levels 0.00-0.90 and 0.90-1.80.
- TB can be adjacent to ST, SK, RF, AT, DW, TB, SR, CD, and can occupy voxels on level 0.00-0.90.
- DW can be adjacent to ST, SK, RF, AT, TB, SR, CD, and can occupy voxels on level 0.90-1.80.
- CM can be adjacent to RF, MW, AT, CM, SR, SK, ST, and can occupy voxels on level 0.90-1.80.
- MW can be adjacent to RF, MW, AT, CM, SR, SK, ST, and can occupy voxels on level 0.90-1.80.
- CD can be adjacent to SK, ST, AT, TB, SR, DW, and can occupy voxels on level 0.90-1.80

FlexFL employs in addition to horizontal rules determining inside-outside orientation of FOs, vertical rules determining top-bottom orientation of FOs. In this context, *inside* describes an orientation towards the inner space of the kiosk, while *outside* describes an orientation towards the outer space. Furthermore, *top-bottom* relations specify location of FOs on top of other FOs such as the exhaust on top of stove.

In addition to cardinality, adjacency and design rules, FL employs optimization rules. These imply definition of most effective functional spatial configurations from an ergonomic point of view, which is based on empirical findings formalized by Neufert [1936] and on the *kitchen work triangle* introduced by the Building Research Council [1993] within the School of Architecture of the University of Illinois:

- The closer the length of the SK-ST-RF triangle sides is to 200 cm the better is the layout. In other words, the closer the SK-ST-RF triangle perimeter is to 600 cm the better is the kitchen layout.
- And maximize voxel occupancy.

A major difference between the FL90 and FlexFL is that a placement variable no longer represents the whole FO, but the anchor of an FO, i.e. the FO's voxel that is closest to the left, front, bottom corner of the FO. Most constraints are based on this *anchor-voxel*, including cardinality and occupancy constraints, as well as optimizations.

Overlap constraints are more complex: To prevent overlaps, rules are added to ensure that if a certain voxel is occupied by the anchor of an FO, the other voxels occupied by it are marked with *arrows* to such an anchor voxel. By ensuring that different arrows can not co-exist with each other or other occupying FOs in a single voxel, overlap is prevented. Extension to the 3D-case is done by adding a third, downwards pointer [Bier et al., 2007].

Adjacency rules are translated in a similar fashion to the FL90, however, neighbor list calculation is performed differently.

FlexFL's major contribution lies not necessarily in the automated horizontal placement of FOs, since this has been already addressed, inter alia, in Wright and Loos, but in the vertical placement of FOs, enabling 3D-layout of functions.

OPTIMIZATION: Once the search space has been reduced by allocation constraints, optimal solutions are generated from the number of valid solutions. In both FL90 and FlexFL, two optimization targets have been consecutively allowed, one to maximize the occupancy, and one to optimize spatial layout with respect to ergonomic aspects. The maximization goal is applied during an initial invocation of the SAT-solver [Bier et al., 2007].

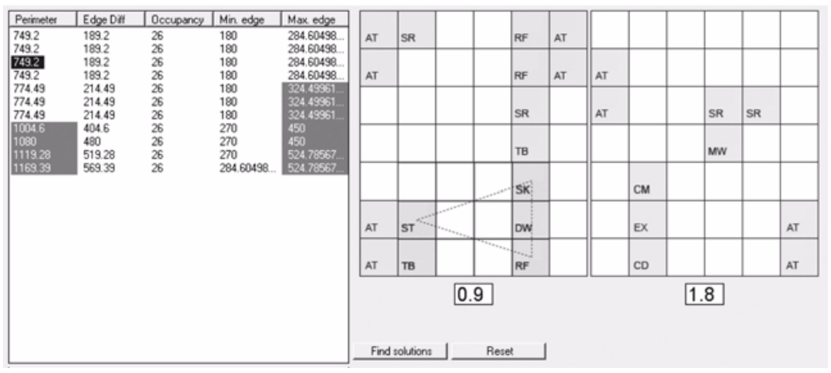
A second call is then made to the SAT-solver, with an additional constraint that fixates the occupancy to the previously found maximum value. During this second run, an ergonomic target is used, based on the empirical findings formalized, inter alia, in the kitchen work triangle by the Building Research Council.

The procedure that MiniSat+ uses to find the optimal solution is to find an initial solution while ignoring the minimization function. It then computes the value of the goal function, and adds a constraint that all new solutions should be better. The solver then continues its search with this new constraint. This procedure is repeated until no new results are found, where the last result is the optimal one.

Ergonomic Rules: Since translating the kitchen work triangle target to a pseudo-Boolean goal function would be too complex, a different approach has been chosen. An exhaustive search is performed for all different layouts of the sink, stove and refrigerator. This is implemented by running the solver repeatedly, while adding constraints to exclude sink-stove-refrigerator configurations that have previously been found.

RESULTS: FL90 generates 11 possible layout solutions from which 4 are optimal. All 4 optimal solutions satisfy occupancy maximization as well as SK, ST, and RF triangle optimization.

Generated solutions are shown as 2D-layouts at 0.90 m and 1.80 m relative space height. The triangle spanning between SK, ST, and RF shows gradient from possible to best possible solutions according to the principle the smaller the triangle, the better the solution.

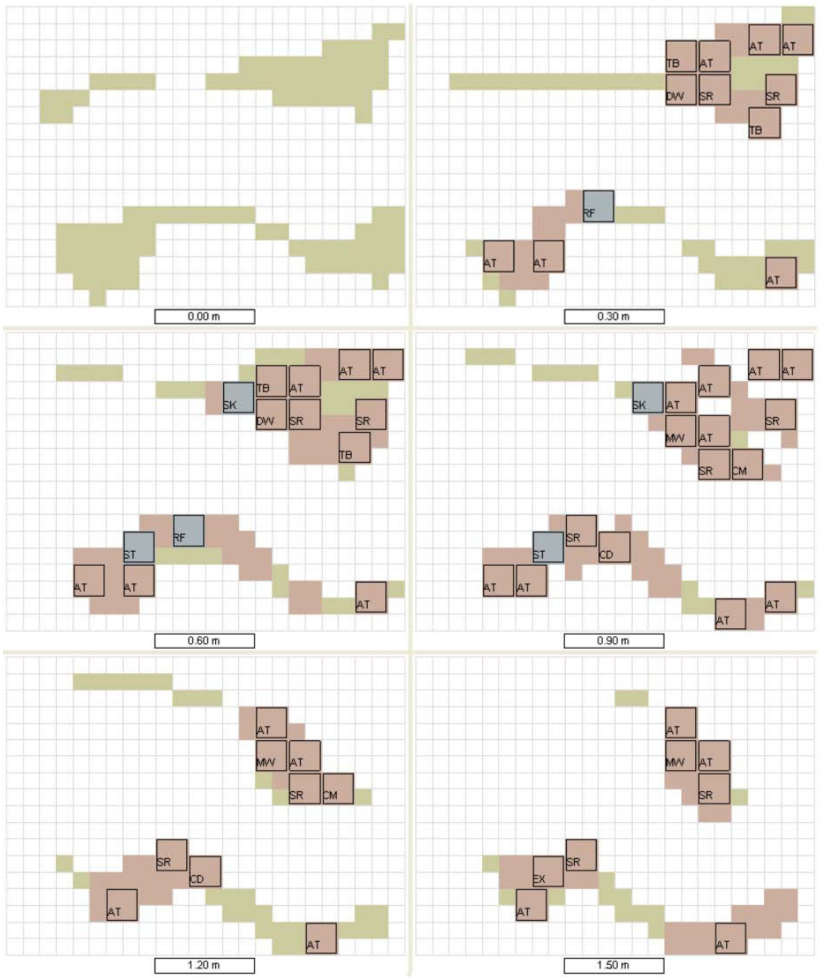


Perimeter	Edge Diff	Occupancy	Min. edge	Max. edge
749.2	189.2	26	180	284.60498...
749.2	189.2	26	180	284.60498...
749.2	189.2	26	180	284.60498...
749.2	189.2	26	180	284.60498...
774.49	214.49	26	180	324.49961...
774.49	214.49	26	180	324.49961...
774.49	214.49	26	180	324.49961...
1004.6	404.6	26	270	450
1080	480	26	270	450
1119.28	519.28	26	270	524.78567...
1169.39	569.39	26	284.60498...	524.78567...

F3.28

The figure - F3.28 - shows one of the four optimal layouts for 90/90/90 voxelization resolution as 2D-representation of the levels at 0.90 and 1.80 m relative space height. Sub-optimal solutions are marked in the table dark gray.

FlexFL30 generates 28 layout solutions incorporating about 23 different triangle configurations from which all are sub-optimal, since the maximum triangle edge length is exceeding 391cm. However, 9 triangle configurations are close to optimal.



F3.29

Solutions are represented in a sequence of 2D voxel-layers, where FO-positions are obviously more differentiated in response to not only the complex geometry but also to a nearly realistic problem definition. Each voxel-layer shows contained FOs:

TB for instance spans 2 voxel-layers, since it is 60 cm high, while SK occupies one voxel-layer, since it is only 30 cm high. FOs are placed in such a way that they accommodate the complex geometry except for one case, where an AT is not accessible - F3.29. This problem can be addressed by defining additional constraints describing in more detail spatial accessibility. This FOs layout satisfies, however, general requirements of not only accessibility but also optimal placement relative to each other as well as to the whole space.

The best possible layout solution for 30/30/30 cm voxelization resolution - shown in F3.29 - is generated corresponding to the following computing time-scheme: AllSolutionsTarget is found in 00:00:01:848 h:m:s:ms, while MaximiseOccupancyTarget and TriangleTarget are found in 00:00:00:793 and 00:00:00:460 h:m:s:ms, respectively. The search stops when all possible and all best possible solutions are found. And since the search even at 30/30/30 cm voxel-resolution is extremely fast - about 3 seconds - the software prototype does not need interactive features: A new search can be started almost every 3 seconds.

FL incorporates sub-tools such as SC : FL : SA : VFA [VolumeToFunctionAdjuster] and SC : FL : SA : NVG [NurbsFromVoxelsGenerator], which have been conceptually developed within the frame of this research but have not been implemented as software prototypes. A3.03

SC : FL : SA : VFA enables a voxelized geometry able to update according to data-input from a database. It also enables adaptive voxel refinement based on local resolution-needs.

SC : FL : SA : NVG generates, as its names suggests, NURBS-surfaces from periphery voxels.

CONCLUSIONS: FL generates functional layouts exhaustively and enables the designer to consider more alternatives than by means of conventional sketching methods mainly because functional space planning is highly combinatorial and, therefore, difficult to conceive exhaustively by human search means.

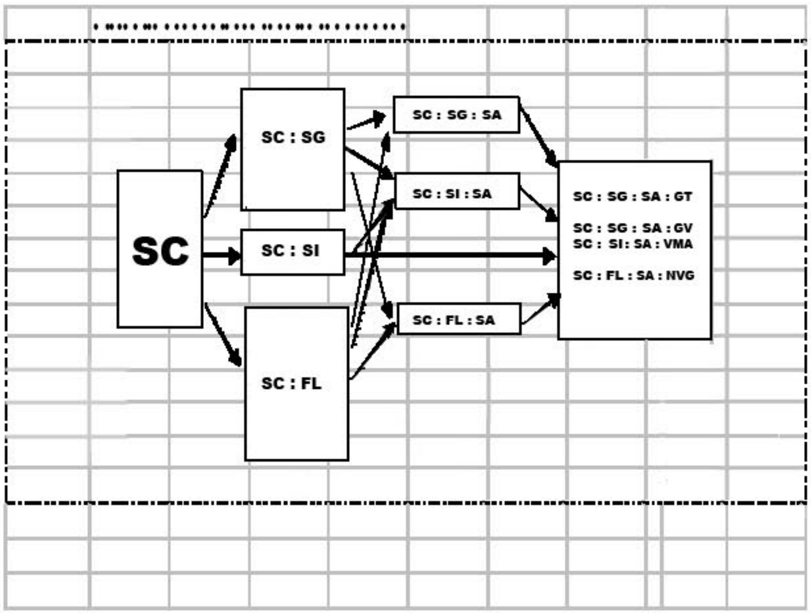
Instead of one, FL generates multiple designs and allows for critical choices by departing from a singular design principle, that represents a potentially prejudiced position of the *singular* designer: FL does not generate the *ultimate* design but instead offers alternative designs within the spectrum of an optimal solutions-field.

DISCUSSION: SC incorporates three sub-tools SpaceGenerator [SG], SpaceInteractivator [SI], and FunctionLayouter [FL], which incorporate sub-tools such as SpaceAdjuster [SA], which in turn incorporate sub-sub-tools such as GeometryTriangulator [GT], GeometryVoxelizer [GV], [VolumeToMotionAdjuster [VMA] and NurbsFromVoxelsGenerator [NVG] as shown in the diagram D3.01.

While GV and GT has been implemented as software prototypes, VMA and NVG have been only developed conceptually. Each of the three SC-tools SG, SI and FL incorporate their own specific SA sub-tools. GV, GT, VMA and NVG are, however, envisioned to operate together on the same platform using data from all three specific SA sub-tools.

Advanced degrees of intelligence enable on the one hand interactive spatial transformations by employing AI, namely, computer vision and sensor-actuator technologies [SC : SI].

On the other hand, they facilitate spatial layout optimization [SC : FL] by means of Boolean Satisfiability [SAT] used, inter alia, in Automated Reasoning.



Automated Reasoning as employed in SC : FL is, basically, logical reasoning, not common-sense or probabilistic reasoning since acceptable conclusions follow logically from the supplied facts such as 3D-space geometry, functional objects descriptions and rules for their positioning in the 3D-space.

For instance, optimization such as occupancy maximization is implemented in MiniSAT+ with a *goal* function, which finds a solution and takes the occupancy of that as a base value. It adds then an *inequality* stating that the occupancy must be larger than that for the next run and continues to run until no solution is found [Bier et al., 2007].

Once the occupancy has been maximized all possible SK-ST-RF triangles with that occupancy are found.

In comparison with Loos and Wright, FL generates functional layouts of similar scale and realistic relevance. However, Loos and Wright deal only with the placement of functional objects in 2D, while FunctionLayouter addresses the layout in 3D dealing with the allocation of functions within complex - at the origin curvilinear geometries - instead of simple - rectangular - geometries.

With respect to optimization, Loos constructs solutions incrementally, testing intermediate solutions on consistency and criteria relevant for architectural design, while an impromptu optimization is carried out according to these intermediate tests implying that no overall objective directs the search. However, without invoking a backtracking procedure, Loos' search is not complete: As presented in Fleming's paper it is neither exhaustive nor does it yield solutions with an overall optimal objective [Bier et al., 2007].

Wright is more similar to the approach implemented in FL: Wright uses constraint satisfaction, to implement a backtracking procedure that makes the search complete, while optimization is implemented afterwards and is, therefore, not used to direct the search.

Loos and Wright deal directly with the geometric aspects of both space and objects, while FL employs voxelization after which all geometric aspects are modeled through neighboring constraints. Furthermore, FL allows for a hierarchical optimization procedure:

Optimal occupancy is an overall objective directing the search while the triangle objective is done by inspection and selection [Bier et al., 2007].

As presented in Fleming's paper [1992] Loos and Wright are rather sensitive to scaling effects, while FlexFL30 indicates that this approach is less sensitive with respect to scaling. Furthermore, since FL is able to find if an assignment is possible or not, its search is complete.

With respect to incorporated aspects of intelligence, FL employs SAT-solving techniques based on an improved DPLL search algorithm:

While the search goes systematically through all instances, intelligent search mechanisms such as non-chronological backtracking and constrain learning are employed. The first one is a backtracking technique [Zhong P. et al., 1998] which increases search efficiency by jumping up - instead of one as backtracking does - more then one level in the search tree, while the second one is a technique for improving efficiency by recording and storing constraints whenever an inconsistency is found.

Section Contributors: H. van Maaren, M. Heule, A. de Jong, N. Brouwers, and G. van der Hoorn.

Acknowledgements: FL has been implemented with graduate students from the Faculty of Computer Science at TU Delft.

3.3 System-Embedded Intelligence

3.3.2 Spatial Prototypes: Motion, Interactive, Mass, and Functional Spaces

The developed prototypical tools generate spatial prototypes such as Motion, Mass, Functional, and Interactive Spaces.

[1] Motion Spaces are spaces generated by tracing the movement of the human body, whereas the motion map defines the boundaries of the volume within which architecture can emerge:

The volumetrical outlines of the body in motion establish an initial framework for architectural studies.

[2] Interactive Spaces are NURBS-based spaces generated interactively by the movement of the body in space, wherein the input - movement - is being electronically processed in such a way that the output represents a continuous, real-time modification of the space.

[3] Mass and Functional Spaces are spaces generated to not only study volumetrical and programmatic issues in architectural design but also to optimize functional layouts.

While the architect is able to find possible solutions for a functional layout, it is uncertain if those solutions are the *best* possible. An optimization algorithm as employed by SC : FL ensures finding best possible solutions for a functional layout as shown in the previous section.

CONCLUSION: The software-prototypes developed within this research address exclusively issues related to [1] Geometry, [2] Function and [3] Movement.

[1] Euclidean and non-Euclidean Geometries - In this research double-curved 3D-spaces are voxelized and double-curved surfaces are triangulated using GV and GT, respectively.

The reason for this conversion implies that adaptive triangulated meshes [Delaunay, 1934] allow approximate representation of complex geometries based on NURBS:

While triangles are simply described by three numbers representing the corners between which the triangle spans, NURBS are described by knot vectors, uniform/ non-uniform degrees, etc. and are, therefore, more difficult to operate with.

NURBS are optimal for intuitive 3D modeling, their implementation in built architecture require, however, discretized spaces and surfaces such as voxelized spaces and triangulated surfaces, respectively.

Methodologically seen voxelized 3D-spaces are employed in SC as mass-models in architectural design: The space is discretized in voxels allowing for input and control of each individual voxel.

Furthermore, the voxelized space is in its gradually refined discretization a volumetric correspondent for the double-curved space. The degree of voxel refinement determines not only formal affinity but increases significantly layout options:

In its ability to represent continuous space by means of high voxel-resolution the voxelized space becomes not only a conversion-tool from continuous to discretized space but also a tool to enable and study function- and mass-modeling for designs based on NURBS-geometries.

[2] Function - In an iterative process employing constraint-based techniques volumes are assigned to functions and spatial relationships are established between the different functional volumes in order to generate 3D functional layouts using FL.

[3] Movement incorporates several aspects: On the one side it requires minimum spaces for implementing specific movement-patterns as developed in SG, on the other side it potentially establishes an interactive connectivity between body and space as implemented in SI.

SG can be seen as a prototype for spatial explorations based on human movement. Even though employed within this research exclusively for NURBS-based spatial generation, SG develops movement patterns for non-orthogonal as well as orthogonal structures. In both cases movement patterns developed with SG allow a realistic estimate for movement/space relations.

Movement, space and function are interrelated, since movement influences the use of function, and both influence the geometry of space.

NOTES: Notes to this section explain concepts and notions difficult to extract from context.

N3.04 - Robotics imply feedback-driven relations between sensors and actuators embedded in a mechanical device, whereas control and feedback are provided by a software run on either an external or embedded computer or micro-controller.

Hyperbody's MuscleTower described in the second chapter can be considered a robotic mechanism.

N3.05 Self-organization for SC:SI has been defined as movement of control points of the NURBS-surface according to rules such as: [1] Keep a certain distance to moving body part, while not exceeding a max-distance, and [2] Go back to original position when no body part is at min-distance. Control points, therefore, organize themselves into specific spatial configurations by following the movement of the body in space.

N3.06 - Sub-tools are named by adding acronyms such as: [1] SC :

SG, [2] SC : SG : SA, and [3] SC : SG : SA : GT.

N3.07 - Adjustment of a double-curved slave-surface to a computer-generated master-surface is based on the principle that the master-device is controlling the slave-device, in such a way that, for instance, min-max distances defined by the master-surface are not exceeded.

N3.08 - Aliasing is a distortion exhibiting a jagged, stepped effect when representing high-resolution objects at low pixel/voxel resolution.

N3.09 - Quad-tree is a tree data structure in which each internal node has up to four children-nodes.

Tree data structures consist of ramified nodes emulating tree structures. In this context, tree-search implies that a problem is recursively split into sub-problems, forming a tree:

Sub-problems are simplified or eliminated using range reduction techniques, until none, one or all solutions are found - http://hpcc.engin.umich.edu/CFD/users/charlton/Thesis/html/node28.html.

The basic range reduction eliminates values that are forbidden by considerations of one to few constraints at a time.

N3.10 - Sliders are used as GUI-controls for choosing a value within a range of values.

N3.11 - Searches have degrees of rigor: [1] Incomplete - heuristics, [2] Complete - approximate global min/max is found, and [3] Rigorous - complete [Russell and Norvig, 2003].

N3.12 - Constraint Satisfaction Problem [CSP] is solved by finding solutions, which satisfy pre-defined criteria named constraints. CSPs are solved by means of search techniques such as backtracking, constraint propagation, local search [Tsang, 1993].

N3.13 - Heuristics consists of methods and approaches for directing the search for solutions.

N3.14 - Backtracking is a search algorithm exploring each possible solution until it finds the correct one [Tsang, 1993].

N3.15 - Non-chronological backtracking is a backtracking technique [Zhong P. et al., 1998] which increases search efficiency by jumping up - instead of one as backtracking does - more then one level in the search tree.

Constraint or clause learning is a technique for improving efficiency by recording and storing constraints whenever an inconsistency is found.

N3.16 - XML [Extensible Markup Language] is a general-purpose markup language facilitating the sharing of data across different information systems.

BIBLIOGRAPHICAL REMARKS: As Gropius' disciple Neufert [1943] published a systematic database consisting of spatial relationships and measurements describing the human body in relation to the occupied space.

Cohen [1993] is comparing body movement patterns of animals and determines the relationship between evolutionary state and mechanical constraints on body movement, whereas Kestenberg [1999] interprets body movement rhythms. From Cohen's and Kerstenberg's movement models Nakata [2003] develops an *automatic choreography* method to generate life-like body movements for humanoids and video game characters.

Reynolds [1987] introduced simple rules for Multi-Agent Systems and Self-organization in his paper on distributed behavioral models, whereas Travers [1996] addressed issues of programming with agents. Both dealt with particle systems, where particles/agents are programmed to have specific behaviors. Kilian and Ochsendorf's particle system, for instance, simulates hanging chain-models [2005].

According to Jackson [1998] and, Brown and Chandrasekaran [1989] ES apply general search strategies based on heuristic knowledge about the problem domain.

Automated Planning by Ghallab et al. [2004] gives insight into the use of combinatorial search in planning, whereas issues on constraint satisfaction are, inter alia, addressed by Tsang [1993], Dechter [2003] and Yokoo [2001].

Automation of functional 2D-layout problems in architectural design has been addressed, inter alia, by Elezkurtaj and Franck [2002], Michalek et al. [2002], Flemming U. et al. [1992], Baykan and Fox [1992], and Lottaz et al. [1998].

Global optimization is described in publications by Kearfott [1996] and Floudas [1999], while local search in combinatorial optimization is addressed by Orlin et al. [2004]. SAT issues have been researched by Davis and Putnam [1960], Davis et al. [1962], and Een and Sorensson [2005].

CONCLUSION

Aspects of intelligence incorporated in computer-based tools and processes and their influence on architectural design have been explored within this research with emphasis - not on representation but - on capabilities compute *complexity*, i. e. complex systems, in architecture, whereas complexity has been defined as arising from the multiplicity of interactions within architectural sub-systems such as physical, environmental and socio-technical systems.

While state-of-the-art computer-based devices have been discussed in the first chapter with little or no reference to how they might inform thinking, the second chapter addressed exclusively questions regarding their influence on design and design thinking.

In both chapters compute-based systems have been identified as incorporating aspects of intelligence not only on in the way systems store and share incorporated knowledge about the designed object but also in the way generated knowledge reflects the ability of digital media to frame questions and interrogate issues pertaining to conceptualization, representation and simulation of architectural design. However, with exception of Parametric and Generative Design, computer-based devices described in the first two chapters have been identified as not incorporating aspects of knowledge about the designed object going beyond its representation.

Parametric and generative design principles, however, have been identified as incorporating knowledge about the designed object at the level of their connectivity with data stemming not only from the object's geometry but also from its *content* and *behavior*. This specific data-connectivity described in the third chapter has been studied and implemented in software-prototypes developed within this research:

These prototypes operating as an Expert System [Brown and Chandrasekaran, 1989] have been exhibiting different degrees of intelligence depending on the complexity of their particular tasks, the most basic degree of intelligence being the intelligence of an automated movement-based SpaceGenerator. More advanced degrees of intelligence enabled on the one hand interactive spatial transformations by employing Artificial Intelligence, namely, computer vision and sensor-actuator technologies, on the other hand, they facilitated automated spatial layout by means of Boolean Satisfiability [SAT].

The developed software-prototypes have, specifically, addressed aspects related to geometry, movement and function. Issues related to form, structure, material have been addressed only contextually, as they reflect the broader context within which geometry, movement, and function have been examined and explored.

Geometrical aspects imply on the one hand conversion from NURBS-based to voxelized geometries on the other hand transformation from NURBS- to triangulated-surfaces. Both processes enable use of complex, non-Euclidean geometries in architecture by conversion to Euclidean geometries.

Geometry - more specifically non-Euclidean geometry - and movement have been conceived as interrelated, since movement acts within this research as space-generating force.

Furthermore, function has been conceived as interrelated with movement. Movement within a space is, therefore, determined by its associated function and incorporated functional objects. A food-kiosk - as shown in this research - requires a specific number of appliances in order to be operable as such. Those appliances, however, are only usable if the space needed for their operation is available.

The developed software-prototypes, therefore, inform each other on a fundamental level, since function influences movement, movement influences the use of function, and both influence intrinsically the geometry of space. They operate semi-automatically and demonstrate not only that *complete* automation from idea to building is not relevant for the time being, since it requires computation as complex as the process itself but also that complete automation is, if implementable, not applicable:

Computers solve specific not all tasks *better* then humans. Computers have, for instance, no common sense. They are able, however, to perform complex computations in relative short time:

Combinatorial optimization implemented in this research, for instance, implies computation of not only all possible but also all best possible spatial combinations of functional units for functional layout design.

While architects might find it difficult to have an overview on all functions and their attributed volume and preferential location, functional units are easily computer-based configured according to pre-defined rules and constraints.

Computers, therefore, have been employed - within this research - specifically in areas where they enhance the design process by incorporating machinic intelligence, compensating where human decision-making might be limited or overextended.

In this process the designer has been rendered as being in constant interaction with the developed semi-automated, intelligent design tools, which can be, therefore, only considered as supporting the designer not replacing him/her.

Software prototypes developed within this research have been tested in a case study, a food-kiosk, for which non-Euclidean spaces evolving from movement patterns and optimized functional layouts have been generated.

Prototypes' testing has identified most relevant aspects of intelligence as being incorporated in the search engine developed for the automated 3D-layout of functions in curvilinear voxelized spaces:

Since able to find if an assignment is possible or not SC : FL has been identified as a complete search engine.

This is based on the employed SAT-solving techniques, whereas Satisfiability can be seen as an *NP-complete* problem. This implies that there is no known algorithm able to determine in *polynomial* time - with respect to computation time - whether there is a satisfying assignment or what that assignment is [Garey and Johnson, 1979]. Only due to the recent development of SAT-solvers such as MiniSat [Een and Sorensson, 2003], RSat [Pipatsrisawat and Darwiche, 2007] and March [Heule and van Maaren, 2006] it is currently possible to solve problems with a large amount of clauses in reasonable time, which is what SC : FL does [Bier et al., 2007].

All three prototypes developed within this research incorporate either static-dynamic or interactive concepts:

While SC : FL places functions and SC : SG generates dynamically space following the movement of the body in space, the resulting space is static. SC : FL and SC : SG are, therefore, only used in the design process, while SC : SI is not only used in the design process but also as dynamic, interactive engine for controlling movements and interactions of building components and buildings on the software level.

All prototypes yield on some level innovative approaches enabling semi-automated design processes in which on the one hand movement patterns dynamically generate space, on the other hand double-curved geometries are voxelized in order to not only estimate volumetrical capacity but also to place optimally functions in 3D-space:

SC : SG departs from a static 2D-view on the relationship between body and architectural space as described in Modulor [Le Corbusier, 1948] in order to address dynamic 3D-movement and its space generative characteristics.

SC : SI departs from a static view on architecture by applying interactive architecture principles [Oosterhuis, 2006] offering a prototypical tool to establish interactive connectivity between movement and curvilinear building-envelopes.

SC : FL advances previous work in 2D, by proposing novel approaches for finding efficiently optimal layout solutions in 3D spaces. SC : FL generates, in comparison with other prototypes such as Loos and Wright, functional layouts of similar scale and realistic relevance. However, Loos and Wright deal only with the placement of functional objects in 2D, while SC : FL addresses the layout in 3D dealing with the allocation of functions within complex - free-formed geometries - instead of simple - rectangular - space geometries.

With respect to optimization, Loos constructs solutions incrementally, testing intermediate solutions on consistency and other criteria relevant for architectural design, while optimization is carried out according to these intermediate tests implying that no overall objective directs the search. However, without invoking a backtracking procedure, Loos' search is not complete. It is, therefore, neither exhaustive nor does it yield solutions with an overall optimal objective as SC : FL [Bier et al., 2007].

Applied in the design development of complex 3D-structures, these software prototypes require, however, further development:

Some prototype parts have been only developed conceptually, while others have been implemented as software. Also, their application at different scales has not yet been exhaustively explored.

Future perspectives for embedding intelligence in computer-based systems in architecture, therefore, imply on the one hand intensified research in the development and use of intelligent computer-based systems in architecture, on the other hand conscious differentiation between problems solvable by such means mainly depending on their nature and scale.

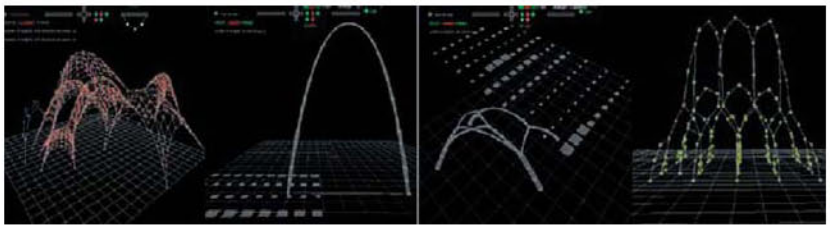
Furthermore, embedding intelligence in computer-based systems in architecture require, as shown in this research, co-operation between architects and computer scientist in a similar way architects co-operate with structural engineers:

This only takes place under the premise that architects have not only the insight into the design problem, which needs to be addressed, but also basic understanding for computational issues relevant for solving together with computer scientists design and construction problems.

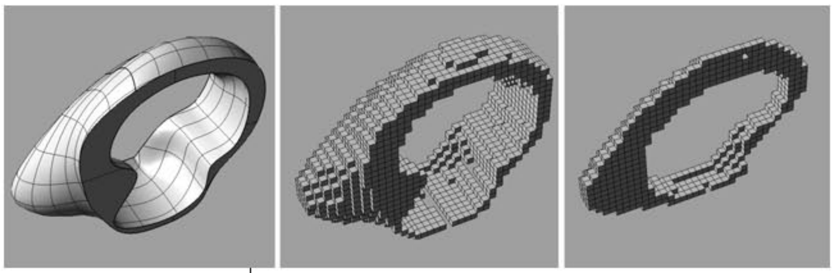
APPENDIX

A3.01 - The model developed at MIT employs a particle-spring system for representing a structure by applying a gravitational field to it in order to generate its most efficient form.

MIT's virtual method is considered to be as straightforward as Gaudi's physical method for exploring hanging-chain models [Kilian and Ochsendorf, 2005].



A3.02 - The VB-scripted function GeometryVoxelizer1 [GV1] has been developed in collaboration with D. Rutten, a program developer for Rhino. It transforms enclosed NURBS-based solids in voxelized geometries. The development of a plug-in for Rhino has been, however, abandoned in favor of developing a Java-application using *processing* for rendering.



A3.03 - FL-layout solutions are 2D represented as a sequence of voxel-layers, where FO positions are obviously more differentiated in response to not only the complex geometry but also to a nearly realistic problem definition.

FL employs voxelization after which all geometric aspects are modeled through neighboring constraints. Furthermore, FL allows for a hierarchical optimization procedure: Optimal occupancy is an overall objective directing the search while the triangle objective is done by inspection and selection.

Furthermore, FL is able to find if an assignment is possible or not, and therefore, its search is complete [Bier et al., 2007].

FlexFL's major contribution lies not necessarily in the automated horizontal placement of FOs, since this has been already addressed, inter alia, in Wright and Loos, but in the vertical placement of FOs, enabling 3D-layout of functions.

Once the search space has been reduced by placement constraints, optimal solutions are generated from the number of valid solutions. The maximization goal is applied during an initial invocation of the SAT-solver. A second call is made to the SAT-solver, with an additional constraint that fixates the occupancy to the previously found maximum value. During this second run, an ergonomic target is used, based on the empirical findings formalized, inter alia, in the kitchen work triangle by the Building Research Council.

A3.04

A3.04 - PROPOSITIONS are considered opposable and defendable, and as such have been approved by the supervisors, Prof. Ir. S. U. Barbieri and Prof. Ir. K. Oosterhuis.

01. Complete automation from idea to building is not be relevant at the time being, because it requires computation as complex as the process itself, but, semi-automation based on system-embedded intelligence is of great relevance.

02. System-embedded intelligence improves the design process by compensating where human decision-making might be limited or over-extended.

03. Architects gain from computational intelligence - in addition to insights - support in the design process not only in form-finding, but also in functional, mechanical, structural, and constructive problem-solving.

04. Embedding intelligence in computer-based systems in architecture requires co-operation between architects and computer scientist in a similar way architects co-operate with structural engineers.

05. Architects need to develop into computational issues relevant for solving together with computer scientists design and construction problems.

06. Future perspectives for embedding intelligence in computer-based systems in architecture imply intensified research and conscious differentiation between problems solvable by such means mainly depending on their nature and scale.

07. System-embedded intelligence renders the designer as being in constant interaction with the semi-automated, intelligent design tools, which can be, therefore, only considered as supporting the designer not replacing him/her.

08. Computers incorporating aspects of intelligence solve specific not all tasks better then humans.

09. Computers have no common sense: They are able, however, to perform complex computations in relative short time.

10. Computer supported functional layout development based on constraint-satisfaction techniques departs from a singular-design principle, that represents a potentially prejudiced position of the master-designer, in order to generate multiple designs within the spectrum of an optimal solutions field.

These propositions are considered opposable and defendable, and as such have been approved by the supervisors, Prof. Ir. S. U. Barbieri and Prof. Ir. K. Oosterhuis.

These propositions have been translated into Dutch by P. J. Teerds - STELLINGEN: Deze stellingen worden opponeerbaar en verdedigbaar geacht en zijn als zodanig goedgekeurd door de promotoren Prof. ir. S. U. Barbieri en Prof. ir. K. Oosterhuis.

01. Complete automatisering van het ontwerpproces, van het concept tot aan het bouwen zelf, is voorlopig niet relevant: het vraagt even complexe computer berekeningen als het proces zelf. Maar semiauto-

matisering, gebaseerd op in het systeem opgenomen computergebaseerde intelligentie, is van grote relevantie.

02. Computergebaseerde intelligentie verbetert het ontwerpproces door de menselijke begrensdheid op het gebied van het nemen van beslissingen te compenseren.

03. Architecten krijgen van computergebaseerde intelligentie - naast inzicht - ondersteuning in het ontwerpproces en bij het zoeken naar de juiste vormen, bij het oplossen van functionele, technische, structurele en constructieve problemen.

04. Computergebaseerde intelligentie in architectuur vereist een vergelijkbare samenwerking tussen architecten en computerdeskundigen als tussen architecten en constructeurs.

05. Architecten moeten kennis ontwikkelen van automatisering die relevant is voor het oplossen van ontwerp en constructie problemen in samenwerking met computerdeskundigen.

06. Het perspectief voor het implementeren van computergebaseerde intelligentie in architectuur veronderstelt een precies onderzoek naar en een beredeneerde differentiatie van problemen die opgelost kunnen worden door deze middelen, afhankelijk van hun karakter en schaal.

07. Intelligentie, die opgenomen is in computer systemen, maakt de continue interactie tussen de ontwerper en de semiautomatische intelligente ontwerpinstrumenten mogelijk; deze instrumenten kunnen daarom alleen als ondersteuning voor de ontwerper gezien worden.

08. Computers, die aspecten van computergebaseerde intelligentie in zich opnemen, lossen beslist niet alle taken beter op dan ontwerpers.

09. Computers hebben geen common sense, daarentegen zijn ze in staat complexe berekeningen in relatief korte tijd uit te voeren.

10. De ontwikkeling van de door de computer ondersteunde functionele organisatie, gebaseerd op constraint-satisfaction technieken, laat het simpele ontwerpprincipe van de meester-ontwerper achter zich, ten gunste van het mogelijk maken van verschillende ontwerpen binnen het spectrum van een veld optimale oplossingen.

BIBLIOGRAPHY

Alexander, C. et al.: 1977, A Pattern Language - Towns, Buildings, Construction, Center for Environmental Structure Series, Oxford University Press, USA.

Alexander, C.: 2004, The Phenomenon of Life - The Nature of Order - An Essay of the Art of Building and the Nature of the Universe, Routledge, UK.

Allen, S.: 1999, Points + Lines: Diagrams and Projects for the City, Princeton Architectural Press, New York.

Altshuller, G.: 1973, Innovation Algorithm, Technical Innovation Center, Worcester.

Barbieri, S. U.: 2003, Res Aedificatoria Fragments of a Contemplation, OASE 62 - Autonomous Architecture and the Project of the City, NAI Publishers, Rotterdam.

Baykan C. A. and Fox M. S.: 1992, Wright - A Constraint-Based Spatial Layout System, published in Artificial Intelligence in Engineering Design, Academic Press, Boston.

Baykan C. A. and Fox M. S.: 1991, Constraint Satisfaction Techniques for Spatial Planning in Intelligent CAD Systems III - Practical Experience and Evaluation, Springer-Verlag, Berlin.

Bier, H.: 2004, Digital Design Strategies, 8th International DESIGN Conference Proceedings, Dubrovnik.

Bier, H.: 2004, Digital Architecture, Architecture - Time, Space and People Magazine, 4/6, Mumbay.

Bier, H.: 2005, Esperimenti di Automazione Digitale - article published in the Journal for Architecture: Il Giornale dell'Architettura, Rom.

Bier, H.: 2005, Other Geometries_Objects_Spaces, published paper at the Conference Engineering & Product Design Education, Edinburgh.

Bier, H. and Schmehl, R.: 2006, SpaceCustomizer - Scripting Based Methods in Architectural Design - Game Set Match 2, Delft.

Bier, H. et al.: 2006, SC - Prototypes for Interactive Architecture, Lecture Notes in Computer Science, Springer, Berlin.

Bier, H. et al.: 2007, Prototypes for Automated Architectural 3D-Layout, Lecture Notes in Computer Science, Springer, Berlin.

Bier, H. and Schmehl, R.: 2008, GeometryVoxelizer - article submitted for publication in Lecture Notes in Computer Science, Springer, Berlin.

Boyer, C: 1996, Cibercities - Visual Perception in the Age of Electronic Communication, Princeton Architectural Press, New York.

Borisenko, A. I. and Tarapov, I. E.: 1968, Vector and Tensor Analysis, Dover Publications, New York.

Brown, D. C.: 1993, Intelligent Computer-aided Design, Encyclopedia of Computer Science and Technology available online from http://web.cs.wpi.edu/~dcb/Papers/EofCSandT.html.

Brown, D. and Chandrasekaran, B.: 1989, Design Problem Solving, Morgan Kaufmann Publishers, San Francisco.

Building Research Council: 1993, Kitchen Planning Standards, University of Illinois, Urbana-Champaign.

Castle, C. and Pollalis, S.N.: 1999, On-line Networks for Construction Projects, Internet-based Information Systems Group, Digital Design Information, Harvard Design School, Cambridge.

Charles, S. and Lipovetsky, G.: 2005, Hypermodern Times, Polity Press, Cambridge.

Chermayeff, S. and Alexander C.: 1963, Community and Privacy - Toward a New Architecture of Humanism, Doubleday, USA.

Childs, M. et al.: 2000, VBScript in a Nutshell: A Desktop Quick Reference, O'Reilly & Associates, Sebastopol.

Coates, P.: 1996, The Use of Cellular Automata to Explore Bottom Up Architectonic Rules, Eurographics Conference, Imperial College of Science and Technology, London.

Cohen, B. B.: 1993, Sensing, Feeling, and Action - The Experimental Anatomy of Body-Mind Centering, Contact Editions, Torornto.

Cook, S. A.: 1971, The Complexity of Theorem-proving Procedures, Proceedings of the Third Annual ACM Symposium on theory of Computing, ACM Press, New York.

Cormen, T. H. et al.: 2001, Introduction to Algorithms, MIT Press, Cambridge.

Chu, K. S.: 1999, X Phylum - The Turing Dimension, available online http://www.archilab.org/public/2000/catalog/xkavya/xkavyaen.htm.

Davis, M. and Putnam, H.: 1960, A Computing Procedure for Quantification Theory, Journal of the ACM 7/1.

Davis, M. et al.: 1962, A Machine Program for Theorem Proving, Communications of the ACM.

Dechter, R.: 2003, Constraint Processing, Morgan Kaufmann, San Francisco.

Dekkers, D. et al.: 2004, MVRDV: The Regionmaker, Wieland & Gouwens Hatje Cantz Publishers, Ostfildern-Ruit.

Delaunay, B.: 1934, Sur la sphère vide, Izvestia Akademii Nauk SSSR, Otdelenie Matematicheskikh i Estestvennykh, Nauk.

Denari, N.: 1999, Gyroscopic Horizons - Prototypical Buildings and Other Works, Princeton Architectural Press, New York.

Eastman, C.: 1973, Automated Space Planning, Artificial Intelligence, Elsevier.

Eastman, C.M., et al.: Fabrication: Examining the Digital Practice of Architecture, Editor Proc. 2004 ACADIA Conference, Cambridge.

Een, N. and Sorensson, N.: 2004, An Extensible SAT-solver, Lecture Notes in Computer Science, Springer Berlin / Heidelberg.

Elezkurtaj, T. and Franck, G.: 2002, Evolution Methods in Architectural Floorplan Design, Umbau, 19:129-37.

Ferber J.: 1999, Multi-Agent Systems - An Introduction to Distributed Artificial Intelligence, Addison-Wesley, Harlow.

Flemming U. et al.: 1992, Hierarchical Generate and Test vs. Constraint-Directed Search - A Comparison in the Context of Layout Synthesis published in Artificial Intelligence in Design, Kluwer Academic Publishers, Dordrecht.

Floudas, C. A.: 1999, Deterministic Global Optimization - Theory, Algorithms and Applications, Kluwer, Dordrecht.

Franken, B.: 2003, Real as Data - Architecture in the Digital Age: Designing and Manufacturing, Spon Press, London.

Friedberg, A.: 2006, The Virtual Window: From Alberti to Microsoft, MIT Press, Cambridge.

Garey, M. R. and Johnson D. S.: 1979, Computers and Intractability - A Guide to the Theory of NP-Completeness, Mathematical Sciences, W. H. Freeman, New York.

Gausa, M et al.: 2003, The Metapolis Dictionary of Advanced Architecture, Actar, Barcelona.

Gehry, F. O. and Ragheb, J. F.: 2003, Frank Gehry Architect, Solomon R. Guggenheim Foundation.

Gehry, F. O. and Gilbert-Rolfe, J.: 2002, Frank Gehry: The City and the Music, Routledge, New York.

Ghallab, M. et al.: 2004, Automated Planning, Theory and Practice, Elsevier, Morgan Kaufmann Publishers.

Glymph, J.: 2003, Evolution of the Digital Design Process - Architecture in the Digital Age: Designing and Manufacturing, Spon Press, London.

Greenberg, D. P. and Georgiades, P. N.: 1992, Locally Manipulating the Geometry of curved Surfaces, IEEE Computer Graphics and Applications, USA.

Greenberg, M. J.: 1993, Euclidean and Non-Euclidean geometries: Development and history, W. H. Freeman, New York.

Gurari, E.: 1999, Backtracking Algorithms CIS 680: DATA STRUCTURES available online from http://www.cse.ohio-state.edu/~gurari/course/cis680/cis680Ch19.html#QQ1-51-128

Haraway, D.: 1991, A Cyborg Manifesto - Science, Technology, and Socialist-Feminism in the Late Twentieth Century, published in Simians, Cyborgs and Women: The Reinvention of Nature, Routledge, New York.

Heule, M.J.H. and Maaren, H. van: 2006, March_dl: Adding Adaptive Heuristics and a New Branching Strategy published in Journal on Satisfiability, Boolean Modeling and Computation 2, Amsterdam.

Jackson, P.: 1998, Introduction to Expert Systems, Addison-Wesley, Harlow UK.

Johnson, P. et al.: 1996, Eric Owen Moss - Buildings and Projects 2, Rizzoli, New York.

Karthik, N.: 2003, A Simple Mathematical Approximation For Efficient and Faster Rendering of Blobs, available online http://www.metlin.org.

Kearfott, R. B.: 1996, Rigorous Global Search: Continuous Problems, Kluwer, Dordrecht.

Kestenberg, J. et al.: 1999, The Meaning of Movement, Gordon & Breach Publishers, USA.

Kilian, A. and Ochsendorf, J.A.: 2005, Particle-spring Systems for Structural Form-finding, Journal of the International Association for Shell and Spatial Structures, Madrid.

Kipnis, J.: 1993, Toward a New Architecture, AD: FOLDING AND PLIANCY, Academy Editions, London.

Knight, T.: 1998, Designing a Shape Grammar: Problems of Predictability, Artificial Intelligence in Design 98, Kluwer Academic Press, Dordrecht.

Kolarevic, B.: 2001, Designing and Manufacturing Architecture in the Digital Age, Architectural Information Management, ECAADE, Helsinki.

Kolarevic, B.: 2003, Digital Morphogenesis, Architecture in the Digital Age: Designing and Manufacturing, Spon Press, London.

Kramer, K. et al.: 1999, Tutorial - Mobile Software Agents for Dynamic Routing, ACM SIGMOBILE Mobile Computing and Communications Review, ACM Press, New York.

Kuhn, T.S.: 1996, The Structure of Scientific Revolutions, University of Chicago Press, Chicago.

Khoshnevis, B. and Bekey, G.: 2002, Automated Construction Using Contour Crafting - Applications on Earth and Beyond, University of

Southern California, 19th International Symposium on Automation and Robotics in Construction, Washington.

Le Corbusier: 1954, The Modulor - A Harmonious Measure to the Human Scale Universally Applicable to Architecture and Mechanics, Faber and Faber, UK.

Lieberman, H. et al.: 2003, Agents for the User Interface, Handbook of Agent Technology, MIT Press, Cambridge.

Liggett, R.: 1985, Optimal Spatial Arrangement as a Quadratic Assignment Problem, Design Optimization, Academic Press, Orlando.

Lintao, Z. and Sharad, M.: 2002, The Quest for Efficient Boolean Satisfiability Solvers, LNCS, Springer Berlin/Heidelberg.

Loomis, B.: 2003, User-Driven Genetic Algorithm for Evolving Non-Deterministic Shape Grammars, available online from http://architecture.mit.edu/descomp/works.htm.

Lottaz, C. et al.: 1998, Constraint Solving and Preference Activation for Interactive Design, Artificial Intelligence for Engineering Design, Analysis and Manufacturing, Cambridge.

Lynn, G: 1999, Animate Form, Princeton Architectural Press, New York.

Maher, M. and Zhang, D. M.: 1991, Case-based Reasoning in Design. AI in Design 91, Sydney.

Manovich, L: 2001, The Language of New Media, MIT Press, and INFO-AESTHETICS MANIFESTO from http://www.manovich.net/.

McCarthy, J. and Hayes, P.: 1969, Some Philosophical Problems from the Standpoint of Artificial Intelligence, Machine Intelligence 4, Edinburgh University Press, Edinburgh.

McCarthy, J.: 1956, Inversion of Functions Defined by Turing Machines, Automata Studies, Princeton University Press, Princeton.

McCormack, J.: 2000, Enabling the Use of Shape Grammars: Shape Grammar Interpretation through General Shape Recognition, Bennett Conference, Pittsburgh.

McCullough, M. et al.: 1999, The Electronic Design Studio - Architectural Knowledge and Media in the Computer Era, MIT Press, Cambridge.

Michalek, J. J. et al.: 2002, Architectural Layout Design Optimization, Engineering Optimization, Taylor and Francis, UK.

Mitchell, W.: 1995, City of Bits - Space, Place and the Infobahn, MIT Press, Cambridge.

Nakata, T.: 2003, Automatic Generation of Expressive Body Movement Based on Cohen-Kestenberg Lifelike Motion Stereotypes, Journal of Advanced Computational Intelligence and Intelligent Informatics, Fuji Technology Press, Tokyo.

Negroponte, N.: 1995, Being Digital, Alfred A. Knopf, New York.

Neufert, E.: 2005, Bauentwurfslehre - Handbuch für den Baufachmann, Bauherren, Lehrenden und Lernenden. Friedr. Vieweg & Sohn Verlag, Wiesbaden.

Newell, A. et al., 1957, Elements of a Theory of Problem Solving, Rand Corporation Report, USA.

Nocedal, J. and Wright, S. J.: 2006, Numerical Optimization, Springer-Verlag Berlin Heidelberg New York.

Novak, M: 1991, Liquid Architectures in Cyberspace, MIT Press, Cambridge.

Novak, M: 1996, Transmitting Architecture - The Transphysical City, article available online under WORD http://www.mat.ucsb.edu/~marcos/Centrifuge_Site/MainFrameSet.html

Oosterhuis, K.: 2002, Mission Statement, Programmable Architecture, l'Arca Edizione, Milano.

Oosterhuis, K.: 2003, Hyperbodies - Towards an E-motive Architecture, Birkhauser, Basel.

Oosterhuis, K. et al.: 2003 E-activities in Design and Design Education, Virtual Operation Room - A Time-based Architecture for the Augmented Body, Europia Productions, Paris.

Oosterhuis, K. et al.: 2004, Real Time Behavior in Architecture - paper published in the Conference Proceedigs CONVR04, Lisbon.

Oosterhuis, K. et al.: 2004, File to Factory and Real Time Behavior in Architecture - paper published in the Conference Proceedigs ACADIA, Cambridge.

Orlin, J. B. et al.: 2004, Approximate Local Search in Combinatorial Optimization, Society for Industrial and Applied Mathematics, 33/5, Philadelphia.

Pipatsrisawat, K. and Darwiche, A.: 2007, RSat 2.0: SAT Solver Description, Technical report D153, Automated Reasoning Group, Computer Science, Los Angeles.

Quigley, A. and Eades, P.: 2000, FADE: Graph Drawing, Clustering, and Visual Abstraction, Lecture Notes In Computer Science, Proceedings of the 8th International Symposium on Graph Drawing, Springer-Verlag London.

Reynolds, C.: 1987, Flocks, Herds, and Schools - A Distributed Behavioral Model, Computer Graphics, 21/4 SIGGRAPH '87 Conference Proceedings.

Rowe, P.: 1991, Design Thinking, MIT Press - reprint edition, Cambridge.

Sam-Haroud, D.: 1995, Constraint Consistency Techniques for Continuous Domains, PhD thesis, SFIT, Lausagne.

Sassen, S.: 2001, The Topoi of E-Space: Global Cities and Global Value Chains, Sarai Reader: The Public Domain, http://www.sarai.net/publications/readers/01-the-public-domain.

Schmidhuber, J.: 1997, A Computer Scientist's View of Life, the Universe, and Everything, Foundations of Computer Science: Potential - Theory - Cognition, Lecture Notes in Computer Science, Springer-Verlag, Berlin.

Simon, H. A.: 1996, The Sciences of the Artificial, MIT Press, Cambridge

Slessor, C.: 1997, Atlantic Star, Architectural Review, 102/12: 30-42, USA.

Smith, I. F. C. et al.: 1995, Spatial Composition Using Cases - IDIOM, ICCBR, Sesimbra.

Stephens, S.: 1997, The Samitaur Building, Architectural Record, New York.

Stiny, G.: 1980, Introduction to Shape and Shape Grammars, Environment and Planning B.

Stiny, G.: 1975, Pictorial and Formal Aspects of Shape and Shape Grammars, ISR, Interdisciplinary Systems Research, Birkhaeuser, Basel.

Yamazakia, Y. and Maedab, J.: 1998, The SMART system - An Integrated Application of Automation and Information Technology in Production Process, Computers in Industry, 1/35, 87-99, Elsevier.

Thon, S. et. al.: 2004, A Low Cost Antialiased Space Filled Voxelization Of Polygonal Objects, Proceedings of GraphiCon 04.

Toyo, I.: 1997, Image of Architecture in Electronic Age, in AA.VV. The Virtual Architecture, Tokyo University Digital Museum, available online http://www.um.u-tokyo.ac.jp/publish_db/1997VA/english/contents.html.

Travers, M.: 1996, Metaphors in Programming with Agents: New Metaphors for Thinking about Computation available online from http://alumni.media.mit.edu/~mt/diss/index.html.

Tsang, E.: 1993, Foundations of Constraint Satisfaction, Academic Press, Elsevier.

Tzonis, A.: 1993, Automation Based Creative Design, Current Issues in Computing and Architecture, edited in collaboration with Dr. Ian White, Elsevier, Amsterdam.

Vazquez-Ruano, O.: 2005, The Hand Stays in the Picture - Virtual Indexicality and Digital Transcoding, CATH Congress on The Ethics and Politics of Virtuality and Indexicality, Bradford.

Venturi, R.: 1996, Iconography and Electronics - Upon a Generic Architecture, MIT Press, Cambridge.

Vidler, A.: 2000, Warped Space: Art, Architecture, and Anxiety in Modern Culture, MIT Press, Cambridge.

Vivacqua, A.: 1999, Agents for Expertise Location, Proceedings AAAI - Spring Symposium on Intelligent Agents in Cyberspace, Stanford.

Watanabe, M.: 2002, Induction Design - a Method for Evolutionary Design, Birkhaeuser, Basel.

Wolfram, S.: 2002, A New Kind of Science, Wolfram Media, Champaign.

Yokoo, M.: 2001, Distributed Constraint Satisfaction - Foundations of Cooperation in Multi-agent Systems, Springer-Verlag Berlin Heidelberg.

Zhong, P. et al.: 1998, Using Reconfigurable Computing Techniques to Accelerate Problems in the CAD Domain - A Case Study with Boolean Satisfiability, Design Automation Conference, San Francisco.

Zuse, K.: 1969, Rechnender Raum, Friedrich Vieweg & Sohn, Braunschweig. English translation from 1970, Calculating Space, MIT Technical Translation AZT-70-164-GEMIT, Cambridge.

GLOSSARY

The following definitions explain not only used nomenclature but also describe tools developed within this research:

SC - SpaceCustomizer is a tool-kit developed within this research to implement, inter alia, tasks such as the triangulation and unfolding of a NURBS-surface, voxelization of a NURBS-based volume, and layouting of functions in a voxelized 3D-space.

SC : SG - SpaceGenerator is a SC-subtool generating NURBS-based spaces by following the movement of the body in 3D-space.

SC : SI - SpaceInterActivator is a SC-subtool generating interactively NURBS-based spaces by following the movement of the body in 3D-space.

SC : FL - FunctionLayouter is a SC-subtool placing functional objects voxelized 3D-spaces.

SC : SG : SA - SpaceAdjuster is a SG-subtool adjusting intuitively generated NURBS-based spaces to the movement of the body in space.

SC : SI : SA - SpaceAdjuster is a SI-subtool adjusting interactively NURBS-based 3D-spaces to the movement of the human body in space.

SC : FL : SA - SpaceAdjuster is a FL-subtool, which allows spatial adjustment according to functional layout.

SC : SG : SA : GV - GeometryVoxelizer is a SG : SA-subtool that enables voxelization of NURBS-based volumes.

SC : SG : SA : GT - GeometryTriangulator is a SG : SA-subtool triangulating NURBS-surfaces.

SC : SG : SA : GU - GeometryUnfolder is a SG : SA-subtool unfolding triangulated NURBS-surfaces.

SC : SI : SA : VMA - VolumeToMotionAdjuster is a SI : SA-subtool enabling adjustment of voxel-layout to the the movement of the body in 3D-space.

SC : FL : SA : NVG - NurbsFromVoxelsGenerator is a FL : SA-subtool generating as its names suggests, NURBS-surfaces from periphery voxels.

DVD SYNOPSIS

The attached DVD has autorun functionality for Windows, which implies that the PowerPoint presentation starts automatically as soon as the DVD is placed in the drive. The presentation starts with an integrated viewer, making the installation of PowerPoint, if not already available, unnecessary.

The presentation consists of text, images and AVI-format movies using the Cinepak Codec available by default on Windows. Forward/backward browsing is enabled by keyboard navigation with up/down arrows. Used nomenclature is described in the glossary.

On Linux or MAC systems, the PowerPoint file must be opened manually. Movies, however, can not be viewed, if Cinepak Codec not available.

Slides 1-2 introduce SC and SC : IS, which deal with the development of digital design strategies based on non-Euclidean geometries, whereas the body in movement generates interactively architectural SPACE.

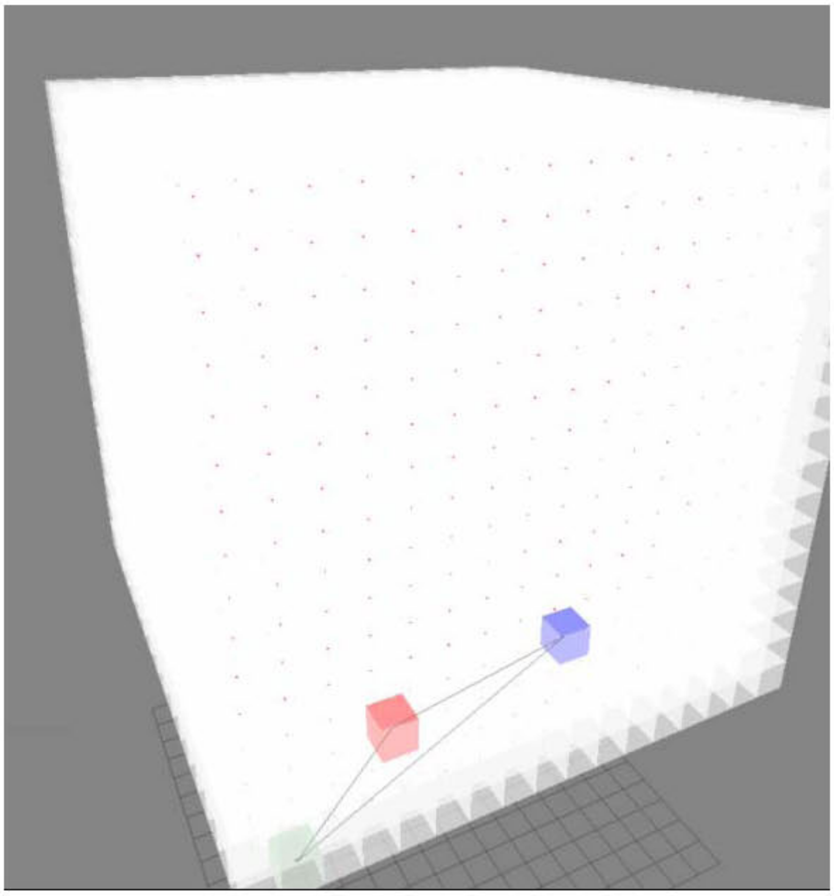
Slide 3 describes SC : FL, which generates and optimizes functional 3D-layouts for voxelized NURBS-based spaces by employing constraint solving techniques.

Slides 4-6 describe BR, which is a preliminary study implemented with Hyperbody-students in Virtools focusing on defining constraints for the 3D-placement of functions in a building.

Slides 7-9 describe SC : GV dealing with the voxelization of NURBS-based volumes.

Slides 10-14 describe SC : FL, whereas slide 10 shows how FOs - Functional Objects - are placed in the voxelized space according to functional and spatial rules described in detail in the third chapter of this research.

Slide 11 - B R U T E S E A R C H goes through all possible solution-instances.



Slide 12 - CSP : SAT-RULES - Constraint Satisfaction Problem : Satisfiability-Rules - and CSP : SAT-SOLVER - Constraint Satisfaction Problem : Satisfiability-Solver - imply search for all possible solutions. In this context INTELLIGENT SEARCH is an optimized search by means of, inter alia, non-chronological backtracking and constraint learning.

Slide 13 - FlexFL30 generates and represents 3D-layout solutions by showing each individual voxel-layer between 0.00-1.50 m and/or more.

Slides 15-22 describe SC : GT, whereas slide 21 shows the structure developed for the triangulation of the NURBS-surface:

Each one of the i = 0, ..., nv-1 u-ribbons is specified by an array of bottom-node points Pi,j with j = 0, ..., nu-1 and array of upper node points Qi,j with j = 0, ..., nu-1.

Two triangles are formed per surface element, whereas an upper triangle Si,j defined by points Pi,j,Qi,j,Qi,j+1 and a bottom triangle Ti,j defined by points Qi,j+1, Pi,j+1, Pi,j.

Slide 22 shows rotation based on rules such as rotate into xy-plane, and take attached geometry with:

Rotation paired with the progressive splitting into ribbons enables flattening of a complex 3D-geometry into a 2D-plane.

Slides 23-29 describe SC : SI, whereas slide 27 shows interfaces, which enable use of real-time and movie-based interaction:

Misuse of SI allows for spatial experimentation with NURBS-based designs.

CURRICULUM VITAE

After graduating in architecture [1998] from the University of Karlsruhe in Germany, Henriette Bier has worked with Morphosis [1999-2001] on internationally relevant projects in the US and Europe. She has taught computer-based architectural design [2002-2003] at universities in Austria, Germany, and the Netherlands and worked on a doctoral research at TU Delft [2004-2008].

Henriette Bier's research focuses not only on analysis and critical assessment of digital technologies in architecture, but also reflects evaluation and classification of digitally-driven architectures through procedural- and object-oriented studies. It defines methodologies of digital design, which incorporate Intelligent Computer-based Systems proposing development of prototypical tools to support the design process. Results of this research have been published in books, journals and conference proceedings.

Henriette Bier regularly leads workshops and lectures at universities in Europe and US. She teaches design studios within Hyperbody Research Group and Border Conditions at TU Delft. Currently, she is project coordinator of the workshop and lecture series on Digital Design and Fabrication within DSD [Delft School for Design] with invited guests from MIT [Massachusetts Institute of Technology] and ETHZ [Eidgenoessische Technische Hochschule Zuerich].