On the computation of well-structured graphic representations in architectural design

Henri Achten
2004
© 2004 Henri Achten
Technische Universiteit Eindhoven
Faculteit Bouwkunde
Bouwstenen 88

All rights reserved. No part of this book may be reproduced in any form by any electronic or mechanical means (including photocopying, recording, or information storage and retrieval) without permission in writing from the publisher.


CIP-DATA LIBRARY TECHNISCHE UNIVERSITEIT EINDHOVEN

Achten, H.H.

NUR 955
Trefwoorden: computer graphics / computer-aided architectural design / architectonisch ontwerpen – dataverwerking
Subject headings: computer graphics / computer-aided architectural design / architectural design – data processing
for
Gabriela
# Contents

Chapter 1  
Introduction  
1.1 Outline  

Chapter 2  
Well-Structured Graphic Representations in Architectural Design  
2.1 Apparent structure of graphic representations  
2.2 Graphic units and generic representations  
2.3 Related work  

Chapter 3  
Identification of Graphic Units and Generic Representations  
3.1 Analysing graphic representations  
3.2 Identified graphic units and generic representations  
3.3 Hypothetical generic representations  
3.4 Overview and scope  

Chapter 4  
Typological Knowledge and Generic Representations  
4.1 Typological knowledge  
4.2 Sequences of generic representations  
4.3 Knowledge application in the sequence  
4.4 Overview and scope  

Chapter 5  
Analytical Technique of the Design Process  
5.1 Methodology for analysis of a design process  
5.2 The case: house design  
5.3 Classification of design actions captured in drawings  
5.4 Overview and scope  

Chapter 6  
Pen Plus: Building a Drawing Tool  
6.1 Graphic units as basis for a sketch tool  
6.2 A set of graphic unit sketch tools  
6.3 Structural Sketcher  
6.4 Overview and scope  

Chapter 7  
Paper Plus: Building a Graphic Unit Recogniser  
7.1 Multi-agent system approach  
7.2 Case-based reasoning approach  
7.3 A hypothetical Case-Based Design Aid System  
7.4 Overview and scope  

Chapter 8  
Summary  
8.1 Graphic units and generic representations  
8.2 Theoretical utility  
8.3 Practical utility  
8.4 Future work  

Acknowledgements  

About the author  

References  


Chapter 1  Introduction

The main subject in this document is graphic representation: the drawings that the architect makes during the design process. Vitruvius already mentions that among the skills an architect has: “he must be a good draughtsman and have a command over geometry, in order to make correct perspective drawings and plans” (Kruft 1994:24). During the history of architectural design, a sophisticated set of graphic techniques has been developed by which design ideas are represented. Such techniques have become conventions of encoding (hatching patterns, line types, icons, etc.) and conventions of depiction (plan, section, perspective, etc.) that are shared within the professional community. If it is possible to capture graphic representations such that computers can start to understand them, then an important bridge to facilitate design has been built.

The work presented in this book brings together my research over the past ten years on this question, starting with my PhD research in 1993. The book is in the spirit of a report, summarising my findings. Contrary to producing a PhD-thesis, the past period was not uniquely focuses on (this) one subject. So for me it was a genuine pleasure to see how in this period I was able to work on this theme along so many different avenues. The view offered here is my own, but it could only come about through the cooperation and discussion with a great many talented people. Throughout the book, and in the acknowledgements, I endeavour to give proper acknowledgements to work by others that have been informative for my research. However, this is not a textbook in the sense that I dwell much beyond the immediately related research.

From the many areas of application of computers in design, graphical design support has enjoyed attention for a very long time (likely since Ivan Sutherland’s Sketchpad from 1963 (Sutherland 1963)). Research on graphical design support has been performed from many different perspectives by a great number of people. The current work hopes to make a modest contribution to the field. It begins with the unremarkable observation that drawings are content-bearers for architects, not only between professionals and other participants in the building process, but also for the architect personally during the design process. The major question is: “How can drawings capture content, and can we describe drawings and content in such a way that both become accessible to the computer?”
There is an important reciprocal relationship between Computer Aided Architectural Design (CAAD) and theoretical inquiry in architectural design. The domain of architectural design has specific requirements it imposes on computer support, and thus it is necessary to understand this. The other way around, theoretical notions in architectural design can be tested computationally and this leads (almost by definition) to design support. Therefore, this work draws its foundations from three disciplines, namely: design research, design theory, and design computing.

- Design research engages in the general study how man designs. It investigates the nature and scope of (cognitive) structures, how these can be described and analysed, and how they are applied in design (Cross 1984). Design research therefore, aims to inform us what is actually happening in design. Findings from this field are particularly relevant to understand the cognitive structures and processes that play a role in design, and how these are related to graphic representations.

- Architectural design theory provides both the reflection (as termed by Schön 1983) of the architect on his profession as well as the normative positions to which architects would like to hold themselves (Rowe 1987). In the current theme of graphic representations, we can learn from design theory how graphic representations are used as a means of discourse, what the perceived role of graphic representations is, how the design process evolves by means of sketches, and how knowledge is captured in particular through the use of diagrams. Despite the wildly varied nature, form, subject matter, and approach of architectural theories, there is general consensus among architects and theorists alike that graphic representations play an important role in architectural design – although there is also substantial disagreement about the exact nature of that role.

- The computer is a device that can implement any given formal system of rules (the so-called Turing-Church thesis, see among others Franklin 1995:77-79). This position is explicitly acknowledged in the fields of Artificial Intelligence (AI) and Cognitive Psychology, both of which assert that human thought is mental computation performed on internal states (Cummins 1988:14-18). It is possible to extend this position to architectural design, and to use the computer to implement findings from design research and design theory. In this way, one can test whether theoretical notions are consistent (do they not lead to
contradictions in their description), valid (do they actually do what they are supposed to do), and productive (are they useful and fast enough in the practice of design). It goes without saying that once implemented, such work can also be used for actual design support.

Put very concisely, it has been found that graphic design support is an intuitively appealing technique that allows architects to design with the computer. Most CAAD systems offer basic functionality for the production of drawings and are now being developed to more sketch-like design support. It appears that the provision of drawing tools as such is efficient for the production of drawings, but this does not lead to understanding of content in such a way that a computer can reason about the produced drawings. In order to achieve this, it is necessary to build more structure in the computer representations of drawings, and to build inference mechanisms on top of that, which can deal with the captured information. This can lead to much improved CAAD:

- A natural interface that very much resembles the way architects work.
- Automated inference of design intentions from drawings, either while the architect is working or from finished drawings.
- Implementation of more versatile drawing tools for architects.
- Well-founded understanding of the scope of expressiveness, consistency, and clearness of graphic representations in architectural design.

1.1 Outline

The main thesis in this book is that there exist conventions of encoding in graphic representations, in particular in the plan representation, which are meaningful and useful to architects. These well-structured representations can be distinguished from each other and formally described in such a way that they can be implemented in a computer system, leading to better design support.

**Part One: Graphic units and generic representations**

Part One is the theoretical part that poses the existence of well-structured graphic representations in architectural design. The theoretical notions underlying the main thesis are introduced in Chapter 2, and the concepts of graphic unit and generic representation are defined. In Chapter 3, the methodology to identify graphic units and generic representations is outlined and the results are discussed.
Part Two: Theoretical utility of graphic units and generic representations

Part Two deals with the theoretical utility of graphic units and generic representations. It is demonstrated how knowledge used in the design process can be captured and utilised (Chapter 4) and how the design process can be analysed by means of graphic units for structured description (Chapter 5). The theoretical work from Part One is expanded in two main directions: further structuring of the set of identified generic representations, and the application of Feature-Based Modelling for a formal description of graphic units and generic representations in architectural design.

Part Three: Productive utility of graphic units and generic representations

There are two principally different ways in which the research findings can be implemented in a CAAD system. One way is to build in graphic unit recognition in whatever the architect is drawing: in other words, to put graphic units “in the paper.” The other way is to build tools that generate graphic units: to put them “in the pen,” so to speak (see Figure 1).

Top: Paper plus: implementation of graphic units as recognition “in the paper.” This approach leaves intact the sketchy and personal appearance of hand drawings. Lines that are drawn are generally conceived as strokes. Recognition may build on the drawing order of strokes, speed of singular strokes and sequences of strokes, style of the stroke, and later on aggregates of strokes into shapes or constellations.

Bottom: Pen plus: implementation of graphic units as tool “in the pen.” This approach provides the user with a comprehensive and consistent set of drawing tools that quickly enable graphic development of the design.
Both strategies have been pursued in the past, and are in varying degrees of development. This has been addressed in four different tracks:

1. An expert system for a building type (pen plus approach).
2. A sketch tool that creates graphic units (pen plus approach).
3. A Case-Based design aid system (paper plus approach).
4. Automated recognition of graphic units in drawings (paper plus approach).

The pen plus approach is presented in Chapter 6; the paper plus approach is presented in Chapter 7.

Part Four: Summary

Part Four provides a brief summary of the presented work and points out the main contributions that have been made over the past ten years.

About notation and references in the work

In order to distinguish between various kinds of concepts that are discussed in the work, a number of notation conventions are used. Table A summarises these.

Table A: Notation conventions.

<table>
<thead>
<tr>
<th>Kind of concept</th>
<th>Notation convention</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graphic unit</td>
<td>(1) <em>Simple contour</em></td>
</tr>
<tr>
<td>Generic representation</td>
<td><em>Proportion System in Contour</em> (14)</td>
</tr>
<tr>
<td>Hypothetical generic</td>
<td><em>Element Vocabulary and Function Symbols</em> (H19)</td>
</tr>
<tr>
<td>Variable in computer code</td>
<td><em>Number_of_people</em></td>
</tr>
<tr>
<td>Feature representation</td>
<td><em>complex Geometry.shape.2DShape {}</em></td>
</tr>
</tbody>
</table>
Part One

Graphic Units and Generic Representations
Chapter 2  Well-Structured Graphic Representations in Architectural Design

The drawing is a medium of great expression to the architect. It allows in various techniques (pen, pencil, etching, paint, etc.) much sensibility in the presentation of a design during all stages of the architectural design process (for examples, consult Klotz 1988:398-418, Pfeiffer 1990, Paatero 1993, Lawson 1994, Shoshkes 1989). All these manifestations of the drawing can be captured under the group of graphic representations, including current-day two-dimensional computer renderings or computer-edited images. In this work, a specific class of graphic representations is studied: namely, the plan-based line drawings that architects make of the building design.

2.1 Apparent structure of graphic representations

In a broader sense, graphic representations are external representations. They are those entities that are physically accessible to our senses, and which are expressly constructed by us to stand in for an object or concept. Examples of external representations are photographs, graphic representations, text, formulas, etc. The role of external representations from a cognitive psychological perspective is to aid the limited resources of short-term memory to keep track of a multitude of issues. This becomes particularly evident when we are engaged in intensive mental processes such as design. Akin (1986) distinguishes five properties that characterise external representations. These properties are:

1  Multiplicity: it is possible to make multiple different representations of the same reality.

2  Consistency: a representation has some constant way of depicting things, which enable the viewer to interpret the representation.

3  Functionality: a representation has a specific kind of use. A perspective projection for example, is useful for showing the spatial appearance of an object or interior, but is less useful for measurements to the drawing.

4  Abstraction: a representation contains a subset of all the properties the thing that is depicted in reality has.

5  Organisation: A representation functions in a particular way, depending on how it is cognitively perceived by the viewer.
The applicability of these concepts to architectural graphic representations can readily be verified. Architectural sources in the thirteenth century (Bucher 1979) and seventeenth century (Serlio 1611) already contain major conventions of depiction used in architecture today: diagrams (for illustrating geometric principles), plans, sections, elevations, perspectives, and details. The property of consistency is demonstrated by the fact that graphic elements are recognised to depict columns, capitals, vaults, walls, etc. The property of abstraction is demonstrated by the fact that the drawing depict the appearance of building elements, and not, for example, their material or construction details.

In later sources, in particular in Durand’s work (Durand 1804) the properties of functionality and multiplicity are used to depict in a series of diagrammatic drawings how the design of a building progresses through an increasing elaborate system of axes (Figure 2). The property of functionality limits the kind and appearance of elements that are used to communicate the purpose of the drawing. The property of multiplicity allows the representation to focus on the use of axes as one way to depict the building design.

The property of organisation can be illustrated from a series of more analytical architectural studies. These studies organise drawings through the careful selection and use of a limited set of entities, such as outlines in a raster (National Building Agency 1965), contours with markers (Sherwood 1979), figure-ground drawings (Herdeg 1967; Zevi 1974), or schematic plans (Wittkower 1973).

The following observations are of particular interest here:

- Compared to other conventions of depiction (perspective, section, façade, etc.) the plan representation almost always appears in (early) design since it can comprehensively depict organisation of the design.
- Under a particular convention of depiction (plan, perspective, section, etc.) a graphic representation does not need extensive textual elaboration in order to produce a correct interpretation. The constituent graphic elements provide strong
clues about the interpretation, and these elements do not change much over a long period of time.

- For identifying what a graphic representation is about, the elements that are used are not the most basic elements of a drawing (vertices, lines, planes, etc.) but aggregates of these elements with a particular interpretation (e.g., a closed polygonal shape with constant thickness and particular hatching that indicates a wall).

- A ‘correct’ graphic representation – a representation that makes sense to a viewer – presents a feasible and well-balanced whole. This means that it is not easily possible to give many different interpretations of a given graphic representation without leading to inconsistencies between elements. Here we face the most distinguishing difference with sketches, where due to the developmental and explorative nature inconsistencies are unavoidable.

The observations above lead to the hypothesis that there exist widely shared ways of depicting particular objects or concepts in architectural design, and that it is possible to identify these. This way of looking at graphic representations is very domain-dependent, and thus quite different from general approaches that aim to be domain independent, such as in the work of Anderson and McCartney (2003). In order to find widely shared ways of depicting particular objects or concepts in architectural design, it is necessary to define what constitutes such a “way of depicting,” and to find suitable aggregates of graphic elements. In this work, two levels of such aggregation are defined to structure the investigation: graphic unit and generic representation.

2.2 Graphic units and generic representations

Throughout the design process, the architect makes numerous drawings (McCall et al. 1997). These drawings form part of the output during a design process, and often it is possible to track the development of a design process throughout the production of drawings. This does not mean that all of the reasoning is captured by means of drawings, or that the whole process can be completely reconstructed only on the basis of drawings. It is rather the case, as Koutamanis (1990) states, that although graphic representations do not explicitly encode all design considerations, ultimately all results are reflected in the drawings. Rodgers et al. (2000) track the design process by means of sketches, which implies a stepwise perspective to the analysis of a design process. Likewise, in
the research work presented here, the design process is conceived as a stepwise process, of which each step can be followed through the production of drawings (Figure 3).

The key element in Figure 3 is a graphic representation, which depicts the state of the design object. In order to make the graphic representation, it is necessary to take design decisions (for example, drawing a closed shape requires deciding on the topology of the shape, relative proportions of the layout, and tentative surface area). Design decisions require knowledge of the design task, the brief, site, design participants, etc. Since the graphic representation is concerned with the state of the design, the required knowledge is declarative knowledge. Each graphic representation therefore, encodes declarative knowledge of the building design.

A sequence of graphic representations establishes a sequence of design decisions. In each step of the sequence the state of the design is defined. The transitions from one state of the design object (graphic representation encoding particular design decisions) to the next state of the design object (graphic representation encoding other particular design decisions) can be defined on the basis of possible transitions from one graphic representation to the next. Such transitions are about the sequence of design decisions, and therefore require decisions about the design process. This type of knowledge is usually termed procedural knowledge. In this work, it is proposed that the set of procedural knowledge and declarative knowledge are captured to some extent in the graphic representation. Such a whole of graphic representation and associated knowledge is termed a generic representation.
The notion of stepwise development, or transition from state to state of the design is well established in computational notions about problem solving (Newell and Simon 1972) and design (Akin 1986:13-17, 19-20). In this perspective designing is likened either to search or to exploration in an abstract design space containing all solutions to a design problem. The exact wording relates to the user’s belief whether design can be tackled through problem solving and some kind of optimisation (design as search), or whether design requires a freer activity within this space (exploration) – see for a formal approach within these assumptions Chang et al. (2003). An important aspect in such approaches is the definition of legal operations that cause state transitions. Here the difficulty surfaces what the correspondence is between operations that can be defined formally correct for search or exploration techniques, and the above mentioned declarative and procedural knowledge associated with generic representations. Although there clearly is a state-transition between generic representations, to assume that thus it is possible to comprehensively describe everything that is relevant to this transition seems unrealistic (Chapter 7 provides a focus limited to low-level changes that can be described on the basis of the drawings). It is important therefore, to determine which part of the transition can be described, and furthermore, which part actually matters to understand what happens in these state transitions.

**Definition of graphic units**

On some level of aggregation, regularly re-occurring objects in drawing seem to depict concepts that have everyday usage in architectural design. Examples are: the “grid,” which consists of orthogonal organised lines at a modular distance; the “axial system,” which consists of lines that set up a number of axes along which a design can be organised; or the “zone,” which consists of (filled) closed polygonal shapes that represent areas with specific properties. These objects are often drawn in very similar manners, and their names are known in the design community. In this research, such objects are termed “graphic units.” The definition of a graphic unit is as follows:

*Graphic unit: a specified set of graphic entities and their appearance that has a generally accepted meaning within the design community.*
Two aspects on which graphic units can be identified are their geometry (graphic primitives and relationships) and appearance (line weight, hatching pattern, line types, etc.). However, since the graphic primitives that make up drawings are rather limited, it is not always possible to distinguish purely on geometry or appearance between various sets of graphic primitives. The issue of “generally accepted meaning” then becomes crucial to distinguish between graphic units. For example, to determine whether a set of lines depicts a schematic subdivision (general principle of ordering subdivision) or a partitioning system (particular way of subdividing a space) cannot be settled purely on the constituent lines themselves. It is necessary to recognise that the schematic subdivision is less specifically developed towards a design than the partitioning system. The distinguishing mark between these drawings concerns the kind of decision taking that is involved when the designer makes these drawings. It is therefore necessary to look also at the decision making aspects that are involved to determine whether one particular set of graphic primitives differs from another, apparently similar, set of graphic primitives.

**Definition of generic representations**

In a drawing there may be several instances of one graphic unit. For example, a drawing with graphic unit “contour,” may contain many contours, and a drawing with graphic unit “grid,” may contain many grids. In the same way, a drawing can contain several instances of different graphic units. For example, instances may occur of the graphic unit “grid” and “contour.” In such cases, there is additional decision-making aside from the decisions related to the grid and contour. The contours are coordinated with respect to the grid, involving decisions about grid-module, and modular size and position of the contour(s). Therefore we can introduce a level of aggregation of graphic units. A graphic representation that consists of instances of graphic units, is called a “generic representation.”

**Generic representation:** a graphic representation that consists of one or more instances of one or more graphic units, but no other graphic elements.

In this work, it is proposed that generic representations are widely used conventions shared among design professionals. In Chapter 3, such generic representations are identified.
2.3 Related work

The current work takes its position in the field of design support by means of graphic representations, and is related to research on:

1. Diagrams.
2. Sketch recognition.
3. Shape recognition.
4. CAAD and drawings.

The work on diagrams and diagrammatic notation by Gross et al. (1988) and Gross (1990) has been informative for the theoretical development of generic representations. These authors investigated the implicit constraint-definition that occurs when architects draw diagrams while designing. Based on these observations, they showed how providing graphic techniques for constructing design drawings could generate constraint-models. Such constraint-models can be used to keep a design consistent with requirements, to help the designer identify problematic areas in the design, and to find similar cases of designs that display the same constraint structure. In the current work, the hypothesis of implicit constraint-definition has been reversed, in such a way that given a drawing, it must be possible to derive the points of design decisions making that have lead to that particular drawing. This reversed hypothesis has lead to the notion of the graphic unit and the generic representation above. In later work, Do (1997), and Do et al. (2000) have shown what kind of graphic shorthands designers use in design. They demonstrated through various implementations the potential use of a computational representation of icon-like images. These shorthands are not similar to graphic units or generic representations because they often have a symbolic rather than an architectural interpretation.

More detailed studies on the mechanics of sketching have revealed that architects tend to construct drawings ‘part after part,’ which can help identify the structures present in a drawing (Kavakli et al. 1998). Versteijnen (1997) shows that sketching aids the designer in particular in the mental simulation task when the consequences of a design have to be thought through. This implies that the drawing is interpreted in an analytical way, and that the architect is looking for structures that are present in the design. There are only a limited number of studies about the depicting techniques in drawings. Koutamanis (2001) investigates and categorises both depiction technique and meaning of sets of graphic elements that are close to graphic units in the current study. His interest
lies in the way graphic elements can perform as coordinating devices for architectural elements such as doors, walls, furniture, etc. In his work the aggregation level of graphic elements is much smaller than those of graphic units, which typically deal with larger elements of a building design. Earlier, Koutamanis has studied automated plan recognition (Koutamanis 1990). Much of automated recognition work takes a drawing after it has been finished, and processes it with a number of techniques taken from machine vision (Marr 1982, Leibovic 1990, Humphreys 1992, Anderson and McCartney 2003). The goal in machine vision is to approximate human vision in machines. For this reason, much of the work that takes place in machine vision has founding assumptions and goals that lie far outside the current investigation. Findings from machine vision are related but very difficult to apply directly. Work in the field of CAAD often is domain-specific, that is, working from the assumption that the representations are architectural in some way (e.g. Pellitteri 1997, and Park and Gero 2000). Such processing often is time-consuming, since it has to take into account additional domain-specific requirements. Nevertheless, this can also be turned into an advantage because the domain limits the search for interpretation.

Sketch-like drawing support by means of CAAD has seen an increase of interest but there has not been much report of follow-up work recently. The work on diagrams mentioned above has been implemented in a number of applications for design support, mainly for retrieval of images or earlier designs: the Electronic Cocktail Napkin (Gross 1996, Do 1998) and GIDA (Do 2001). Applications by other researchers on sketch-support are SketchBoX (Stellingwerff 1999), Piranesi (Richens 1999), and the transparent sketch tool (Trinder 1999).

For the recognition of architectural drawings, in particular sketches that are made in the form of plans, it is necessary to understand how sketches are drawn, and what they depict. Leclercq (2001) has implemented a sketch system, called EsQUIsE, which interprets rooms and their functions from a sketch in real-time. In terms of the current work, EsQUIsE works with the generic representation Functional Spaces (7), because it analyses directly from the sketch the given stroke-input in terms of spaces (the lines and how they make contact, their nearness, etc.) and function indication (lines depicting text). Based on this, EsQUIsE reasons about a given plan layout and can infer some properties of the preliminary design in terms of thermal behaviour, daylighting, and so forth. EsQUIsE constitutes a very convincing example that a more thorough
understanding of architectural representations may have great benefits for design support.

Compared to the work presented above, the current discussion on graphic units and generic representations provides a more comprehensive framework that defines which sets of graphic entities need to be supported in a CAAD system. This is also where the notions of graphic unit and generic representation make their proper contribution to the field. A deeper understanding of the conventions of encoding and depiction may significantly boost developments in graphical design support. The related work provides much information about alternative applications, user interface, and application areas in the field of graphical design support that is not addressed in this book.
Chapter 3 Identification of Graphic Units and Generic Representations

Graphic units and generic representations have been defined in the previous Chapter. In this Chapter, the research methodology for identifying graphic units and generic representations is described, the results of the analysis on a body of historical graphic representations, and the scope and possible omissions in the work.

3.1 Analysing graphic representations

In order to identify graphic units, an inventory was taken of architectural drawings as they have been published in a wide range of sources in a long time span. Sketches are excluded because since sketches are produced under time-pressure, often for personal use, they are more ambiguous than regular drawings, and by nature they can contain inconsistencies in what they depict. Sketches pose additional problems of interpretation caused only by the way they are produced. The difference between sketches and drawings principally lies not that they depict things differently, but more in the precision of the drawing. The main aim of the research is to verify whether graphic units exist, and if so, how they are defined. Because this question can be settled with fewer problems when drawings are used for the analysis than with sketches, the analysis has been limited to architectural drawings.

The body of analysis

Drawings have been selected from a large historical sample of architectural representations. By doing so, the notion of graphic units is substantiated by a long period of architectural practice, which thus forms a sound basis for their existence. It was not always possible to draw graphic representations from original, facsimile, or reprint documents. In those cases, the images were taken as they were published in other sources that explicitly refer to the original sources. In total, 211 graphic representations from twenty-eight sources were used. These sources range from the thirteenth to twentieth century (Table B: next page).
Table B: Graphic representations and their sources (year, author (source), and number of images).

<table>
<thead>
<tr>
<th>Year</th>
<th>Author (source)</th>
<th>Number of images</th>
</tr>
</thead>
<tbody>
<tr>
<td>1215-1235</td>
<td>Villard de Honnecourt (Bucher 1979)</td>
<td>3</td>
</tr>
<tr>
<td>1494-1496</td>
<td>De Sangallo (Vidler 1990)</td>
<td>1</td>
</tr>
<tr>
<td>1521</td>
<td>Cesariano (Tzonis and Lefaivre 1986)</td>
<td>5</td>
</tr>
<tr>
<td>1560</td>
<td>Master WG (Bucher 1979); Cousin (Tzonis and Lefaivre 1986)</td>
<td>1 ; 7</td>
</tr>
<tr>
<td>1590</td>
<td>Rusconi (Tzonis and Lefaivre 1986)</td>
<td>1</td>
</tr>
<tr>
<td>1611</td>
<td>Serlio (Serlio 1611)</td>
<td>2</td>
</tr>
<tr>
<td>1728</td>
<td>Gibbs (Tzonis and Lefaivre 1986)</td>
<td>1</td>
</tr>
<tr>
<td>1738</td>
<td>Palladio (Palladio 1738)</td>
<td>3</td>
</tr>
<tr>
<td>1797</td>
<td>Bentham (Vidler 1990)</td>
<td>1</td>
</tr>
<tr>
<td>1804</td>
<td>Durand (Durand 1804)</td>
<td>59</td>
</tr>
<tr>
<td>1847</td>
<td>Ledoux and Ramée (Vidler 1990); Milizia (Milizia 1847)</td>
<td>3 ; 8</td>
</tr>
<tr>
<td>1957</td>
<td>Zevi (Zevi 1974)</td>
<td>2</td>
</tr>
<tr>
<td>1960</td>
<td>Vitruvius translated by Morgan (Vitruvius 1960)</td>
<td>14</td>
</tr>
<tr>
<td>1965</td>
<td>National Building Agency (National Building Agency 1965)</td>
<td>2</td>
</tr>
<tr>
<td>1967</td>
<td>Herdeg (Herdeg 1990)</td>
<td>14</td>
</tr>
<tr>
<td>1971</td>
<td>March and Steadman (March and Steadman 1971)</td>
<td>4</td>
</tr>
<tr>
<td>1973</td>
<td>Wittkower (Wittkower 1973)</td>
<td>1</td>
</tr>
<tr>
<td>1974</td>
<td>Boekholt et al. (Boekholt et al. 1974)</td>
<td>2</td>
</tr>
<tr>
<td>1976</td>
<td>Carp and van Rooij (Carp and van Rooij 1974); March (March 1976)</td>
<td>1 ; 9</td>
</tr>
<tr>
<td>1979</td>
<td>Ching (Ching 1979)</td>
<td>30</td>
</tr>
<tr>
<td>1980</td>
<td>Netsch (Schmertz 1980)</td>
<td>1</td>
</tr>
<tr>
<td>1985</td>
<td>Clark and Pause (Clark and Pause 1985)</td>
<td>12</td>
</tr>
<tr>
<td>1990</td>
<td>Herdeg (Herdeg 1990); Bailey (Bailey 1990)</td>
<td>18 ; 5</td>
</tr>
<tr>
<td>1991</td>
<td>Mitchell and McCullough (Mitchell and McCullough 1991)</td>
<td>1</td>
</tr>
</tbody>
</table>
Images from the sources were selected when they depicted:

- Plan representation (no sections, facades, or perspectives).
- Level of complexity similar to preliminary design phase or earlier (no shop drawings, structural details, or presentation drawings).
- Variety in graphic techniques.

The images were coded according to the source from which they were taken, the page number, and the order number of the image on the page starting from the top left to the bottom right (in case of multiple images on a page). These then form the body of analysis.

**Methodology of the analysis**

Each drawing is analysed in terms of its constituent elements, to see whether meaningful aggregates can be determined that can be discriminated from other aggregates on the aspects of geometry, appearance, and design decision. Because such aggregates usually do not appear in isolation in a graphic representation, they have to be distinguished among other graphic entities.

When it is possible to determine such an aggregate, and it has not been identified as a graphic unit before, it is tentatively defined as a new graphic unit by naming it, and given a description of the aspects geometry, appearance, and design decision. The aspect geometry describes the drawing in terms of the graphic primitives that are used; the basic shapes, lines, and elaborations of these forms. The aspect appearance describes how the geometry is depicted: line type, weight, hatching, and so forth. The aspect design decision describes what factors are involved when someone decides to make a drawing of that particular kind (a contour implies decisions about shape, topology, and relative proportion; a grid implies decisions about module, and so forth).

If the aggregate has been identified before, the graphic representation where it occurs is compared with all previous instances where the graphic unit was found. It is checked whether the definition of the graphic unit still holds, whether it needs further refinement, or has to be changed or removed. The resulting graphic units in the list should be mutually exclusive and maximally comprehensive. Because this is not achieved in one run of the analysis, the process above occurs in a number of cycles to reorder the found set of tentative graphic units. The following example illustrates the method (Figure 4).
Three drawings taken from various sources. The images are redrawn after original sources (Ching 1979, Mitchell and McCullough 1990, Zevi 1974).

The left and middle images have the following differences:

- The circle, triangle, and square in the left image are instances of the platonic shapes. Under the assumption that the graphic representation depicts a building, the single line of the shapes defines the building envelope. The shapes that make up the layout in the middle are composite forms of rectangles, circles, and lines. Under the assumption that the graphic representation depicts a building, the lines of the shapes define borders between spaces.

- The shapes of the layout are not always platonic, but in all cases they are contours. Therefore, it is possible to distinguish between “simple contour” and “contour.” As a result, two graphic units are defined: “simple contour” for all instances of platonic shapes, and “contour” for any other shape. The distinction is made since drawing a simple contour involves a conscious decision by the architect to use that specific form. Both graphic units get a number, and they are denoted as (1) Simple contour, and (2) Contour.

The layout in the middle and the filled-in black and white drawing on the right could be considered as multiple instances of the graphic unit (2) Contour. However, the following differences occur:

- In the layout, lines depict the contours. In the filled-in black and white drawing, a graphic distinction is made by the filling pattern in the complex shapes. There is no indication how the shape is built up from smaller elements. The black and white drawing indicates the mass-space distribution of the building by means of a figure-ground technique.
The design decisions in the layout deal with composition of contours and their relationships, relative size, and position. The design decisions involved with the figure-ground image deal with the space-mass distribution of the design.

As a result a new graphic unit is defined: (6) *Complementary contours* for all those instances where the figure-ground technique indicates a space-mass like distribution.

In a similar manner, all graphic representations from the body of analysis have been analysed for their constituent graphic units. In many cases, the drawings consist of more than one graphic unit. Drawings that have one or more graphic units (but no other elements) are termed generic representations. The names of generic representations are built up of the names of the constituent graphic units. For example, the generic representation that has graphic units (2) *Contour* and (16) *Grid*, is named *Contour in Grid*. Generic representations have a different notation to distinguish them from graphic units: *Contour in Grid* (15). The number is given for reference purposes, and does not indicate any order among the generic representations.

On the results of the analysis presented here, and later work, a decision tree has been established to distinguish between graphic units (section 7.3:103-105). Work on automated recognition is presented in section 7.1:93-99.

### 3.2 Identified graphic units and generic representations

The 211 graphic representations of the survey resulted in 24 graphic units (Table C) and 50 generic representations (Table D). The following sections provide the summary of the results. A more comprehensive analysis is presented in Achten (1997b).

**Graphic units identified in the analysis**

The left column displays the name and source of the graphic unit, an example image from which it is identified, and a simplified, iconic image. Each graphic unit has a number for abbreviated reference – the number does not signify any order among graphic units. In the right column, the graphic unit is described by its geometrical features, appearance, and design decision aspects.
Table C: Graphic units identified in the analysis.

<table>
<thead>
<tr>
<th>Graphic unit</th>
<th>Description of the graphic unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1) <em>Simple contour</em> (Ching 1979)</td>
<td>Regular n-sided (n=3, 4, 5, etc.) polygonal shapes, including the circle, represented by single lines. On the level of scale of the building, the simple contour represents the building envelope. Decisions concern the basic shape of the building, in particular the limitation to a regular n-sided shape.</td>
</tr>
<tr>
<td>(2) <em>Contour</em> (Mitchell and McCullough 1991)</td>
<td>Closed polygonal shapes, represented by single lines. On the level of scale of the building, the contour represents the major spaces of a building. If the object is much smaller – indicated by relative size with other objects – it may part of the graphic unit element vocabulary (see below). Decisions concern the actual shape of objects.</td>
</tr>
<tr>
<td>(6) <em>Complementary contours</em> (Zevi 1974)</td>
<td>Composition of outlines through figure-ground technique. The boundaries of the shapes are represented by single lines. All shapes that have the same colour represent either mass or space of the building. Decisions concern relations between mass-space, public-private, or inside-outside in building layout.</td>
</tr>
<tr>
<td>(3) <em>Measurement device</em> (Serlio 1611)</td>
<td>Subdivided line, arrows, and numbers. A measurement device provides a unit of measurement of the drawing or the dimensions of parts of the drawing. It is an explicit means of establishing dimensions of elements in the drawing. Decisions concern the dimensions of related objects. The measurement device usually occurs with other objects in a drawing.</td>
</tr>
<tr>
<td>(4) <em>Specified form</em> (Palladio 1738)</td>
<td>Contour with specified dimensions. Single line representation plus some dimensioning indicator. Decisions concern the accurate size and shape of objects. The objects in the drawing are annotated by remarks about dimensions, thus establishing their size.</td>
</tr>
<tr>
<td>(5) <em>Elaborated structural contour</em> (March and Steadman 1971)</td>
<td>Complex articulated shape depicting an outline with structural detail. Building envelope with details of structural elements (columns, walls, and such). Decisions concern the elaboration of building structure.</td>
</tr>
</tbody>
</table>
Table C: Graphic units identified in the analysis – continued.

<table>
<thead>
<tr>
<th>(7) Function symbols (Boekholt et al. 1974)</th>
<th>Textual or iconic indication of function by use of letter symbols, text elements, or icons. Decisions concern the location of functions in the design. The function symbols are the text elements “W,” “S3,” and “K1/E” in the example image. Function symbols usually occur in combination with other objects.</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Function symbols" /></td>
<td>F</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(21) Functional space (Herdeg 1990)</th>
<th>Contour combined with specific hatching and line pattern to indicate different functions. Closed polygonal shape indicating space. Decisions concern the differentiation of spaces on the basis of their function, without explicit function indication (function symbol).</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Functional space" /></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(8) Zone (Boekholt et al. 1974)</th>
<th>Area with specific use or function. Closed polygonal shape with some marking to distinguish from other zones. Decisions concern broad functional indications where spaces, functions, or objects may be located in the design. A zone does not specify the contours that can be realised in it. A zone is usually accompanied either by some measurement device or specified form to give it dimensions, or annotated with function symbols to indicate the purpose of the zones.</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Zone" /></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(10) Schematic subdivision (Wittkower 1973)</th>
<th>Schematic depiction of principal subdivision. Single lines representation. Simple indication how the design should be subdivided, without looking at the concrete building design.</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Schematic subdivision" /></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(22) Partitioning system (Clark and Pause 1985)</th>
<th>Lines indicating general way of partitioning without specifically indicating where actually partitions will occur. Single lines representation. Decisions concern the main means of subdividing the building design. In the example, the partitioning system occurs in combination with the building envelope.</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Partitioning system" /></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(13) Schematic axial system (Durand 1804)</th>
<th>Schematic depiction of organisation of axes. Single line representation. Basic organization principle along which parts of the building are arranged. Axes usually represent the main axes of spaces, but this can vary. Decisions concern the basic composition of the design.</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Schematic axial system" /></td>
<td></td>
</tr>
</tbody>
</table>
Table C: Graphic units identified in the analysis – continued.

<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
<th>Additional Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>(15)</td>
<td>Axial system</td>
<td>Organisation of axes applied to building design. Single lines representation. Precise location of axes and placement of elements along the axes. Decisions concern the elaboration of axes in the building design. The applied axial system is the central cross-lines in the example image. As the axial system is always specific for a building design, it occurs in combination with other objects.</td>
</tr>
<tr>
<td>(11)</td>
<td>Modular field</td>
<td>Irregular subdivision of area along coordinating lines. Single lines representation. Decisions concern the structure of the coordinating lines that rule where elements should be placed.</td>
</tr>
<tr>
<td>(16)</td>
<td>Grid</td>
<td>System of parallel lines on modular distance. Single line representation. The grid structures where elements can be placed along the coordinating lines. Decisions concern the measure of the module. A grid as such usually occurs in combination with the elements it coordinates; in the case of the example, a series of columns.</td>
</tr>
<tr>
<td>(17)</td>
<td>Tartan grid</td>
<td>System of parallel lines using two alternating modules of different size. Single line representation. Two different bands are used for placing kinds of elements – usually mass-elements and space-elements. Decisions concern the measure of the two different modules and their proportion. A tartan grid usually occurs in combination with the elements it coordinates. In this case it is combined with an axial system and a circle in the centre.</td>
</tr>
<tr>
<td>(18)</td>
<td>Structural tartan grid</td>
<td>The structural tartan grid is a tartan grid (see above) with one band is reserved specifically for structural elements. In the example, the tartan grid coordinates structural elements such as walls and columns. Therefore, the grid lies under the walls.</td>
</tr>
<tr>
<td>(12)</td>
<td>Refinement grid</td>
<td>Parallel lines on modular distance coordinated in grid with larger module. Single lines representation. Structuring where elements should be placed using a smaller module, and coordinating it with larger grid. The refinement grid is located in the example image in corners where the large 4x4 grid is subdivided in another 4x4 smaller grid (leading in effect to a 16x16 grid).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td><strong>(19)</strong> Element vocabulary (Durand 1804)</td>
<td>Set of simple shapes depicting (interior) elements. An element vocabulary defines a coherent set of matching elements. Elements are represented by closed polygonal shapes and additional line elements. Decisions concern the set of elements and their relative sizes.</td>
<td><img src="image1" alt="Diagram" /></td>
</tr>
<tr>
<td><strong>(20) Structural element vocabulary (Durand 1804)</strong></td>
<td>Set of simple shapes depicting structural elements. Closed polygonal shapes. Set of related elements for structural purposes (columns, walls, and such). Decisions concern composition, and rules for combination and placement of the elements.</td>
<td><img src="image2" alt="Diagram" /></td>
</tr>
<tr>
<td><strong>(24) Combinatorial element vocabulary (Milizia 1847)</strong></td>
<td>Precise relationships between particular elements. Closed polygonal shapes indicating elements organised with respect to each other on specific locations. Decisions concern regular relationships between elements.</td>
<td><img src="image3" alt="Diagram" /></td>
</tr>
<tr>
<td><strong>(23) Proportion system (Bucher 1979)</strong></td>
<td>Diagram showing how proportions are derived (e.g. Golden Section). Single lines representation, organised in a geometric relationship. Decisions about relative proportions of objects.</td>
<td><img src="image4" alt="Diagram" /></td>
</tr>
<tr>
<td><strong>(26) Circulation scheme (Ching 1979)</strong></td>
<td>Principal layout of circulation. Single or double line representation. Decisions concern the principle indication how circulation in the building design should be achieved, without looking at the concrete design itself.</td>
<td><img src="image5" alt="Diagram" /></td>
</tr>
<tr>
<td><strong>(27) Circulation (March 1976)</strong></td>
<td>Layout of circulation applied to building design. Single or double line representation. Decisions concern the particular way how circulation is achieved in the building design.</td>
<td><img src="image6" alt="Diagram" /></td>
</tr>
</tbody>
</table>
As the above presentation shows, in those cases where the graphic primitives are very similar to each other (e.g. (13) Schematic axial system and (15) Axial system; (19) Element vocabulary and (20) Structural element vocabulary; and (17) Tartan grid and (18) Structural tartan grid), it really is necessary to distinguish by the meaning of the drawings to establish a graphic unit.

**Generic representations identified in the analysis**

Graphic units are the smallest set of graphic elements that have a meaning in architectural design. Drawings that consist of combinations of one or more graphic units are called generic representations. In the body of analysis, the generic representations occur as particular combinations of graphic units. Table D lists all generic representations found in the analysis.

<table>
<thead>
<tr>
<th>Table D: List of all generic representations found in the analysis.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Generic representations with one graphic unit</strong></td>
</tr>
<tr>
<td>Simple Contour (1)</td>
</tr>
<tr>
<td>Combination of Contours (2)</td>
</tr>
<tr>
<td>Complementary Contours (3)</td>
</tr>
<tr>
<td>Modular Field (4)</td>
</tr>
<tr>
<td>Proportion System (5)</td>
</tr>
<tr>
<td>Multiple Grids (6)</td>
</tr>
<tr>
<td>Functional Spaces (7)</td>
</tr>
<tr>
<td><strong>Generic representations with two graphic units</strong></td>
</tr>
<tr>
<td>Proportion System in Contour (14)</td>
</tr>
<tr>
<td>Contour in Grid (15)</td>
</tr>
<tr>
<td>Zone in Specified Form (16)</td>
</tr>
<tr>
<td>Function Symbols in Combination of Contours (17)</td>
</tr>
<tr>
<td>Axial System in Specified Form (18)</td>
</tr>
<tr>
<td>Schematic Subdivision in Grid (19)</td>
</tr>
<tr>
<td>Schematic Subdivision With Function Symbols (20)</td>
</tr>
<tr>
<td>Schematic Subdivision in Contour (21)</td>
</tr>
</tbody>
</table>
3.3 Hypothetical generic representations

The concept of the graphic unit is the most significant contribution of the research work. In importance graphic units take precedence above generic representations. The introduction of a new graphic unit has a much larger impact (through the combinatorial effect with the other graphic units) than the introduction of a new generic representation (which may consist of a previously unseen combination of existing graphic units). This is so because the combinatorial possibilities of 24 graphic units are very large. In order to quantify this, we can use the observation that in the set of identified generic representations none has more than four graphic units.

Table D: List of all generic representations found in the analysis – continued.

<table>
<thead>
<tr>
<th>Generic representations with two graphic units - continued</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partitioning System in Contour (22)</td>
</tr>
<tr>
<td>Specified Elaborated Structural Contour (23)</td>
</tr>
<tr>
<td>Elaborated Structural Contour in Grid (24)</td>
</tr>
<tr>
<td>Circulation Scheme in Elaborated Structural Contour (33)</td>
</tr>
<tr>
<td>Structural Element Vocabulary in Structural Tartan Grid (34)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Generic representations with three graphic units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proportion System in Elaborated Structural Contour in Tartan Grid (35)</td>
</tr>
<tr>
<td>Zone in Contour in Grid (36)</td>
</tr>
<tr>
<td>Axial System in Contour in Grid (37)</td>
</tr>
<tr>
<td>Axial System in Contour in Tartan Grid (38)</td>
</tr>
<tr>
<td>Axial System in Specified Form in Structural Tartan Grid (39)</td>
</tr>
<tr>
<td>Schematic Subdivision in Grid and Refinement Grid (40)</td>
</tr>
<tr>
<td>Structural Element Vocabulary in Structural Tartan Grid and Refinement Grid (46)</td>
</tr>
<tr>
<td>Schematic Subdivision and Schematic Axial System in Contour (41)</td>
</tr>
<tr>
<td>Elaborated Structural Contour and Function Symbols and Axial System (42)</td>
</tr>
<tr>
<td>Element Vocabulary in Zone and Contour (43)</td>
</tr>
<tr>
<td>Circulation in Contour in Grid (44)</td>
</tr>
<tr>
<td>Structural Element Vocabulary in Contour in Modular Field (45)</td>
</tr>
<tr>
<td>Structural Element Vocabulary in Axial System in Contour (47)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Generic representations with four graphic units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Element Vocabulary and Function Symbols and Grid in Specified Form (48)</td>
</tr>
<tr>
<td>Schematic Subdivision in Zone in Contour With Function Symbols (49)</td>
</tr>
<tr>
<td>Structural Element Vocabulary in Axial System in Contour in Grid (50)</td>
</tr>
</tbody>
</table>
If four graphic units is the maximum number of graphic units, then all possible generic representations are those that are the result of combinations of one graphic unit (24 combinations), two graphic units (276 combinations), three graphic units (2024 combinations), and four graphic units (10626 combinations). This makes a total of 12950 possible generic representations. Thus it is not surprising that in the current body of analysis many of the possible generic representations have not been found.

There is a big difference between all possible generic representations and the actual 50 generic representations that are found in the body of analysis. Does this mean that the body of analysis is much too small? It is difficult to tell without further analysis what combinations of graphic units constitute viable generic representations and which combinations are less useful. Generic representations are drawings that convey a specific meaning, such as the organisation of a plan, the subdivision of a contour, the way elements should be located in a grid, and so forth. A limited number of combinations of graphic units may be sufficient to capture such meanings. It is important to understand which combinations these are, so that they can lead to a better understanding of architectural drawings. This is attempted in the next section.

**Addition and subtraction of graphic units**

On the basis of the current work, it is impossible to hypothesize about a graphic unit that has not been identified in the research. Neither the analysis nor the theory provides clues about additional graphic units. It is however, very well possible to reason about generic representations, given the current set of 24 graphic units. The main mechanism that helps us to do so is the addition or subtraction of graphic units to generic representations.

The addition of a graphic unit to a generic representation means that to a generic representation a graphic unit is added which it does not have at the moment. For example, the addition of graphic unit (16) Grid to generic representation *Structural Element Vocabulary in Axial System in Contour* (47) results in generic representation *Structural Element Vocabulary in Axial System in Contour in Grid* (50). Both are part of the current set of generic representations. It is also possible, for example, to add the graphic unit (16) Grid to the generic representation *Elaborated Structural Contour and Function Symbols* (27), leading to generic representation *Elaborated Structural Contour*
and Function Symbols in Grid. This is not part of the current set of generic representations and therefore would be a hypothetical generic representation.

Subtraction of a graphic unit from a generic representation means that a graphic unit is taken away from a generic representation. For example, subtraction of the graphic unit (16) Grid from the generic representation Circulation in Contour in Grid (44) leads to Circulation in Contour (32), both of which are already identified. It is also possible, for example, to subtract the graphic unit (16) Grid from the generic representation Axial System in Contour in Grid (37), which results in a generic representation Axial System in Contour that is not part of the current set of generic representations. Thus, the latter generic representation would be a hypothetical generic representation.

Addition and subtraction are allowable mechanisms to search in this manner for hypothetical generic representation, because they are the simplest mechanisms to explain differences between generic representations within the current set of graphic units. Three strategies can tell more about the existence of generic representations not identified in the body of analysis: (1) right-bound subtraction of graphic units from existing generic representations; (2) left-bound addition of graphic units to existing generic representations; and (3) right-bound subtraction combined with left-bound addition to existing generic representations.

All generic representations resulting from right-bound subtraction of Element Vocabulary and Function Symbols and Grid in Specified Form (48), which has four graphic units: (19), (7), (16) and (4). Subtracting graphic unit (7) Function symbols leads to a generic representation with three graphic units: (19), (16) and (4), which is hypothetical Element Vocabulary in Specified Form in Grid (H19). In the next step, subtracting graphic unit (4) Specified Form leads to generic representation Element Vocabulary in Grid (28), which already has been found. In the last step either graphic unit (16) Grid or (19) Element Vocabulary can be subtracted, leading to Element Vocabulary (11) and Grid (H45).
Right-bound subtraction of graphic units

Right-bound subtraction of graphic units works by taking any generic representation from the current set of generic representations, and to subtract step-by-step one graphic unit from this generic representation. This leads to both existing generic representations and hypothetical generic representations.

Take for example the generic representation *Element Vocabulary and Function Symbols and Grid in Specified Form* (48) (see Figure 5). There are four graphic units that may be subtracted: (19) *Element vocabulary*, (7) *Function symbols*, (16) *Grid*, and (4) *Specified form*. Therefore, the first subtraction leads to the following generic representations: *Function Symbols and Grid in Specified Form*, *Element Vocabulary and Grid in Specified Form*, *Element Vocabulary and Function Symbols in Specified Form*, and *Element Vocabulary and Function Symbols in Grid*. All of these do not occur in the current set of generic representations, so they are all hypothetical generic representations. In the next step, from each of these hypothetical generic representations one graphic unit is subtracted. In total this leads to 15 generic representations, of which there are three in the current set, and twelve hypothetical generic representations. Right-bound subtraction of all generic representations leads to 58 hypothetical generic representations.

Left-bound addition of graphic units

Left-bound addition of graphic units works by adding to any generic representation of the current set a graphic unit that it does not have, and to continue adding step-by-step until there are four graphic units in total. Although the mechanism of addition is complementary to subtraction, this leads to a different solution than the right-bound subtraction, because the starting set of generic representations with one graphic unit is much larger than is the case in the found generic representations that have four graphic units.

To the generic representation *Proportion System* (5), for example, 23 graphic units can be added in the first step, leading to one generic representation from the current set; *Proportion System in Contour* (14) and 22 hypothetical generic representations. In the next step, to each of these generic representations 22 graphic units can be added, leading to 506 hypothetical generic representations. In the final step, 21 graphic units can be added, leading to 10626 hypothetical generic representations. As can be imagined, doing this for all generic
representations leads to almost the same large number as the logical combination of all graphic units.

**Right-bound subtraction and left-bound addition**

The final possibility is the combination of both right-bound subtraction and left-bound addition. This means that any addition of graphic units ultimately must end in a generic representation of the current set. This severely constrains the very large number of possibilities of the left-bound addition. In total, it leads to 9 hypothetical generic representations.

**List of right-bound hypothetical generic representations**

Both the logical combination of all graphic units and the left-bound addition of graphic units yield a too large number of hypothetical generic representations to be informative about possible generic representations that have not yet been identified. The combination of left-bound addition and right-bound subtraction leads to nine hypothetical generic representations, which seems too small. For this reason, the right-bound subtraction is chosen to hypothesize which generic representations are also possible. Table E presents a list of all 58 hypothetical generic representations. Hypothetical generic representations are distinguished from generic representations with the letter ‘H’ added to the number.

**Table E: All hypothetical generic representations based on right-bound subtraction.**

<table>
<thead>
<tr>
<th>Hypothetical generic representations</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Structural Element Vocabulary in Contour in Grid</strong> (H1)</td>
</tr>
<tr>
<td><strong>Structural Element Vocabulary in Axial System in Grid</strong> (H2)</td>
</tr>
<tr>
<td><strong>Schematic Subdivision in Zone in Contour</strong> (H3)</td>
</tr>
<tr>
<td><strong>Schematic Subdivision in Zone with Function Symbols</strong> (H4)</td>
</tr>
<tr>
<td><strong>Schematic Subdivision in Contour with Function Symbols</strong> (H5)</td>
</tr>
<tr>
<td><strong>Zone in Contour with Function Symbols</strong> (H6)</td>
</tr>
<tr>
<td><strong>Element Vocabulary in Zone</strong> (H29)</td>
</tr>
<tr>
<td><strong>Element Vocabulary in Contour</strong> (H30)</td>
</tr>
<tr>
<td><strong>Function Symbols and Axial System</strong> (H31)</td>
</tr>
<tr>
<td><strong>Schematic Subdivision and Schematic Axial System</strong> (H32)</td>
</tr>
<tr>
<td><strong>Schematic Axial System in Contour</strong> (H33)</td>
</tr>
<tr>
<td><strong>Schematic Subdivision in Specified Form</strong> (H58)</td>
</tr>
</tbody>
</table>
**Table E: All hypothetical generic representations based on right-bound subtraction – continued.**

<table>
<thead>
<tr>
<th>Hypothetical generic representations - continued</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Element Vocabulary and Function Symbols in Grid (H7)</td>
<td>Schematic Subdivision in Refinement Grid (H34)</td>
</tr>
<tr>
<td>Element Vocabulary and Function Symbols in Specified Form (H8)</td>
<td>Grid in Refinement Grid (H35)</td>
</tr>
<tr>
<td>Element Vocabulary in Specified Form in Grid (H9)</td>
<td>Axial System in Structural Tartan Grid (H36)</td>
</tr>
<tr>
<td>Function Symbols in Specified Form in Grid (H10)</td>
<td>Specified Form in Structural Tartan Grid (H37)</td>
</tr>
<tr>
<td>Structural Element Vocabulary in Grid (H11)</td>
<td>Axial System in Tartan Grid (H38)</td>
</tr>
<tr>
<td>Structural Element Vocabulary in Contour (H12)</td>
<td>Contour in Tartan Grid (H39)</td>
</tr>
<tr>
<td>Structural Element Vocabulary in Axial System (H13)</td>
<td>Zone in Grid (H40)</td>
</tr>
<tr>
<td>Axial System in Grid (H14)</td>
<td>Proportion System in Elaborated Structural Contour (H41)</td>
</tr>
<tr>
<td>Schematic Subdivision in Zone (H15)</td>
<td>Proportion System in Tartan Grid (H42)</td>
</tr>
<tr>
<td>Zone in Contour (H16)</td>
<td>Elaborated Structural Contour in Tartan Grid (H43)</td>
</tr>
<tr>
<td>Function Symbols in Zone (H17)</td>
<td>Structural Element Vocabulary (H44)</td>
</tr>
<tr>
<td>Function Symbols in Contour (H18)</td>
<td>Grid (H45)</td>
</tr>
<tr>
<td>Element Vocabulary in Specified Form (H17)</td>
<td>Axial System (H47)</td>
</tr>
<tr>
<td>Function Symbols in Grid (H20)</td>
<td>Zone (H48)</td>
</tr>
<tr>
<td>Function Symbols in Specified Form (H21)</td>
<td>Function Symbols (H49)</td>
</tr>
<tr>
<td>Specified Form in Grid (H22)</td>
<td>Specified Form (H50)</td>
</tr>
<tr>
<td>Axial System in Contour (H23)</td>
<td>Refinement Grid (H51)</td>
</tr>
<tr>
<td>Structural Element Vocabulary in Refinement Grid (H24)</td>
<td>Structural Tartan Grid (H52)</td>
</tr>
<tr>
<td>Structural Tartan Grid in Refinement Grid (H25)</td>
<td>Circulation (H53)</td>
</tr>
<tr>
<td>Structural Element Vocabulary in Modular Field (H26)</td>
<td>Tartan Grid (H54)</td>
</tr>
<tr>
<td>Contour in Modular Field (H27)</td>
<td>Measurement Device (H55)</td>
</tr>
<tr>
<td>Circulation in Grid (H28)</td>
<td>Partitioning System (H56)</td>
</tr>
</tbody>
</table>
Utility of hypothetical generic representations

Hypothetical generic representations have two uses: the first is to assess the scope of the current work and how well the current set covers all possible generic representations. The second use enables reasoning about transitions between generic representations. This is useful in the theoretical and practical utility of the work (see in particular Chapter 4 and Chapter 7).

3.4 Overview and scope

The categorisation of graphic representations in terms of graphic units leads to the identification of genuinely different kinds of drawings that have different utility in the design process. Half of the 24 identified graphic units represent structuring devices (such as grid, zone, and axial system) rather than concrete building elements (such as contours, functional space, and circulation). This implies that architects have an extensive set of graphic representations for organising the design. Graphic units vary from being schematic, indicating global organisation or intention (e.g., contour, schematic subdivision, and schematic axial system), to being very specific and precise about location and dimension of the elements that are depicted (e.g., elaborated structural contour, functional space, and circulation). This implies that graphic units are tuned to the level of information that is available in the design process.

Graphic units encode things such as composition, layout, modularisation, circulation, and interior elements in a graphic way. Therefore, even without additional explicit textual information, they convey information that is generally shared by the architectural community. Much of this information is encoded implicitly, but can be derived by examining the drawing. This implies that graphic units can form the basis for a visual language on which to build more sophisticated design support.

Generic representations found in the survey have up to four graphic units. This is partly the result of the limitation to fairly simple graphic representations taken from the sources. Nevertheless, it also shows that the aggregation level of the graphic unit is not too small and does not lead to a great number of rather meaningless sets of graphic elements, nor that the aggregation level is too big leading to a very limited number of graphic elements.

In terms of research methodology, the distinction between graphic units on the basis of geometry, appearance, and design decision is productive to identify a number of aggregates of graphic entities that have a specific meaning to the
architect. The distinction by means of geometry and appearance is rather straightforward, but it cannot discriminate between two drawing elements that look the same, but have different interpretation. For example, a set of parallel lines placed orthogonal may either indicate a grid or a set of closely packed squares. The distinction on the basis of decision-making therefore is required, but is problematic in the sense that it requires understanding of the context of the drawing and its purpose. This leads to less clearly stated criteria on which to differentiate between graphic units.

The aim of the survey was first to establish a wide as possible selection of kinds of images, rather than large repetitive numbers of similar images. The analysis has revealed in the first place the breadth of scope of graphic units (what they represent and how they are used). By means of right-bound subtraction of graphic units a quantified statement about the number of feasible generic representations can be made. This exercise does not provide a definite answer, because it necessarily can only build on identified graphic units. So although this argumentation to some extent is circular, it does provide a clue about the comprehensiveness of the analysis – in all cases on internal consistency. If we may assume that right-bound subtraction actually identifies likely combinations of graphic units, then in case of a good survey there should not be many more hypothetical generic representations than those actually found in the analysis. However, the number of 58 hypothetical generic representations would double the total amount of generic representations. This indicates that the scope of the analysis is limited. A more comprehensive study of (historical) sources with architectural graphic representations is required to identify more generic representations. If the current broad historic perspective is taken, this may involve several hundreds of sources with many thousands of images. Based on current experience, this will not lead to a dramatic increase in the number of graphic units: perhaps up to three new graphic units will be thus identified. The number of new generic representations is very likely to be much greater, but will still remain some orders of magnitude away from the total number of generic representations possible by simple combination. Such a survey is worthwhile to establish a firm base for these elements and for improved understanding of architectural graphic representations.
Part Two

Theoretical Utility of Graphic Units and Generic Representations
Chapter 4  Typological Knowledge and Generic Representations

In this Chapter, the procedural and declarative knowledge content of generic representations is investigated. For this purpose, attention is given to a specific class of knowledge content associated with building types, in particular the office building type. The declarative knowledge of the office building is established by means of a literature survey on office buildings. The procedural knowledge is established by making a sequence of generic representations that represent a design process of an office building. Based on this work, the declarative knowledge is applied to the subsequent generic representations that form the sequence.

4.1 Typological knowledge

The building type is a (knowledge) structure that is both recognised as a constitutive cognitive element of human thought and as a constitutive computational element in CAAD systems. Despite general consensus about the relevance of the concept, design theorists such as Rowe 1987, Heath 1984, Habraken 1985, Lawson 1997, and Schön 1988 have various approaches to type that differ with respect to terminology, place in design, role in design, and significance for architecture. In the following section, two basic roles of type and two basic approaches to type definition are outlined.

Two roles of building type

The concept of building type has a number of roles in architectural theory and design theory. Basically it can be considered either as a theoretical construct for practice or as a cognitive knowledge structure. As a theoretical construct, the building type is a comprehensive means for organising large amounts of knowledge related to classes of buildings. It is recognised within the professional community and used in architectural theory; see for example de Quincy 1825, Argan 1963, and Colquhoun 1967; to establish norms for building types; e.g. De Chiara and Callender 1981, and Neufert and Neufert 1992; for descriptive purposes; e.g. Sherwood 1979, and Polyzoides et al. 1982; and for architectural history (Pevsner 1979 provides a good example). Such approaches
capitalise on the human ability of categorisation. The mechanisms underlying such distinctions between types are rarely made explicit.

In order to deal effectively with the design process, the architect needs relevant internal representations of knowledge (Akin 1986, Lawson 1980). The building type as a form of knowledge seems to associate declarative knowledge (the kind of building) with procedural knowledge (designing that kind of building), which is a critical aspect of a cognitive knowledge structure (Markman 1999:5-10). There seem to be only suggestions or propositions that the psychological schema may be the mental representation of a building type (Hamel 1990:11, 34, Coyne et al. 1990, and Sowa 1984:128). Work in the field of AI has shown the productivity of the notion, but it still faces notorious problems, in particular the required specialisation of reasoning mechanisms that can deal with real-world complications (Russell and Norvig 1995:207).

**Two approaches to building type**

A second distinction focuses on basic assumptions on the nature of building type. These are the ambiguity versus explicitness approach, and the idealistic versus procedural position. The ambiguity approach states that the building type is inherently ambiguous (de Quincy 1825, Argan 1963, Colquhoun 1981, Rossi 1982:40-41, Habraken 1985:27-28). Type cannot be defined in an explicit manner but only through its instances. The ambiguity accounts for the creation of related-yet-different instances of the building type by appealing to the indefiniteness of type. This approach offers in fact no mechanism or principle that can be studied more carefully. The explicitness approach states that the building type can be defined explicitly, and that it instructs designers to create instances of the type (this approach is generally associated with Durand: see Perez-Gomez 1983:4, Vidler 1977a, 1977b, and Westfall and van Pelt 1991:146-148). The explicitness approach clarifies the creation of instances of the type by identifying procedures and principles of instantiation.

The idealistic position of type contains the pervasive notion that there is some abstract entity as a ‘building type object’ of which the architect has knowledge in some way. It poses the existence of ideal types of which instances (concrete buildings) are imperfect examples. De Quincy (1825) was the first to explicitly state this view, and versions of it still have currency (e.g. Mitchell 1990:86-94). This position influences research on building types. It emphasises implementation of type-like structures that are taken to encompass the building
type. Making the general structure specific creates instances. The approach has a lot of attractiveness, not the least of which is the fact that it can be computationally modelled (in particular by means of the prototype: Gero 1990, Coyne et al. 1990, Oxman 1990, and Rosenman and Gero 1993). However, the focus on a prototype-like structure tends to downplay the importance of the design process, which is required for instantiation. The logical opposite to the idealistic position is that there is no such thing as a ‘building type object,’ but rather a pre-structured process of instantiation and articulation in which knowledge is applied as the means to create instances. This point of view puts emphasis on the process rather than the type object. It is related to the problem of defining natural species in biology and theories of evolution (as Dennett 1995:202 points out: “evolution does not need [to draw lines]; the real world can get along just fine with the de facto divergences that emerge over time.”)

**The building type in this study**

In this work, the building type is considered as classes of buildings that have major characteristics in common. Types are often identified by their function. Examples of building types are hospitals, offices, and airports. A building type encompasses a significant form of knowledge in design. When dealing with a theatre for example, the architect already has relevant information of theatres by virtue of knowing the theatre type. Thus, knowledge of building types aids in the design of buildings that belong to that type. Because it comprises common knowledge, the building type enables easy communication between design participants. Questions that seem unresolved up to now about computational approaches to building types are the relationship between the various instances that are generally recognised as belonging to a particular building type, the way a type can deal with varying briefs (or with mixed functional use), and how a type can accommodate different sites. Approaches that aim to model building types as data structures of interrelated variables face problems clarifying these questions.

Knowledge of the building type is used during the design process. It informs both the order of decisions taken in the design process (in other words, it provides procedural knowledge) and the outcome of these decisions (in other words, it provides declarative knowledge). Rather than focusing on a ‘building type object’ it seems necessary to investigate the role of knowledge in the design process and to find a representation which:
• Supports design decisions.
• Supports the changes and transformations of the design during the design process.
• Encompasses knowledge of the design task.
• Relates to the way architects design.

As outlined in section 2.2, the design drawing seems to be well suited for such a representation. We therefore propose a structured process by means of generic representations (that hold procedural knowledge of building type instantiation), and to which declarative knowledge of the building type is applied per step from one generic representation to the next, leading in effect to instantiation of designs that belong to that particular building type. Rather than saying that this particular process is the general way in which architects work, it works more like a recommended procedure, or in other words, a design method. Distinctive of this approach is that the design method can be described completely in a graphic way by means of a sequence of generic representations. Such a descriptive way seems to lie closer to design sensibilities of architects than purely verbal methodologies.

The office building type

The office building type is chosen for application of the theory of generic representations. Although it mainly occurs in the 20th and 21st Centuries, its roots lie as far back as the 16th Century in the case of the so-called ‘Uffizi’ administrative buildings in Florence of the de Medici family (Pevsner 1986:47, 213, Staal 1987, Chapter 1). This study is limited to the main kind of office building as produced in the period 1970-1990, which is still in currency in the early 21st Century. Many sources on office buildings are based on this type and provide relevant information for knowledge acquisition.

Knowledge of the office building type is acquired from existing literature on office buildings. Each source must present to some extent a comprehensive account of office buildings in quantitative statements. This ensures some consistency in the knowledge extracted from the source. The range in time of the sources is from 1973 to 1992, with origins from the Netherlands, Germany, United Kingdom, and the United States. Since the subject is the mainstream office building of the period 1970-1990, which in matter of style is relatively context-independent, both range in time and country are not problematic for the

Knowledge acquisition of the office building type

Information in sources about a building type is presented in various manners: text, graphics, diagrams, tables, nomograms, calculations, etc. The format depends on the subject, the expected audience, and required level of detail (e.g. does it present rules of thumb for first estimates, or rather precise information for fine-tuning design decisions). Statements are extracted from the sources when they can be formulated in a sentence stating some state of affairs or quantity. Such statements are instances of declarative knowledge. Example statements are:

- “The conference rooms should be centrally located to the users” (De Chiara and Callender 1980).
- “Office buildings typically have a Gross Area/Net Area ratio of 1.35” (Hoke 1988).
- “The planning module and the exterior wall module must be reconciled with the structural module or column bay. If all these modules coincide, then the wall or window units adjacent to the column must be smaller than the intermediate units” (De Chiara and Callender 1980).

Declarative knowledge aids the architect when making design decisions such as “where to position the conference room,” “what total area can be expected from this brief,” and “how to coordinate grids.”

The generic representation determines which information is required. This is based on the properties of the generic representation.

4.2 Sequences of generic representations

The set of generic representations does not yet have a structure that determines how a sequence of generic representations can be established. There are three mechanisms for putting generic representations in a sequence: by means of successive graphic units, addition of graphic units, and themes of generic representations.
**Successive graphic units**

In a design process, one can identify sequences of design decisions. For example: before the particular length of a building wing is decided upon in a design, the decision has been taken that the shape of the building actually constitutes a number of wings. In terms of graphic units this means that the graphic unit: (2) *Contour* (a shape with no particular dimensions), is established before the graphic unit: (4) *Specified form* (a contour with particular dimensions). The (4) *Specified form* forms the precondition for (5) *Elaborated structural contour* (a shape with particular dimension which edge has been detailed). Therefore, it is possible to define the sequence of successive graphic units: (2) $\rightarrow$ (4) $\rightarrow$ (5).

This principle of moving from less specific to more specific can be used to delineate successions of graphic units as they become more developed. In this way, 12 sequences of successive graphic units are identified (Figure 6).

The names of the successive graphic units are:

- (2) *Contour* $\rightarrow$ (4) *Specified form* $\rightarrow$ (5) *Elaborated structural contour*.
- (1) *Simple contour* $\rightarrow$ (4) *Specified form*.
- (2) *Contour* $\rightarrow$ (6) *Complementary contours*.
- (7) *Function symbols* $\rightarrow$ (8) *Zone* $\rightarrow$ (21) *Functional space* $\rightarrow$ (19) *Element vocabulary*. 
Since moving to a successive graphic unit means developing the graphic unit into more specific terms, such a movement requires decision making by the architect. Given the current set of ‘instruments’ in this research, it is possible to infer decision making in terms of changes in graphic units and generic representations. This will surely be quite different from any possible sequence or combination of considerations that may lead the architect from one phase to the next (he or she may be browsing through a magazine, have a discussion with an associate, focus on quite different aspects, and so forth). The position taken here is not that all of this design reasoning can be captured, but rather that it is possible to limit analysis to capturing the design reasoning that results in graphic changes in the design drawing, and even more so, that this is sufficient to lead to good design support.

Later we will also see how the notion of successive graphic units can be used when matching generic representations to the current design drawing in a Case-Based Design context (section 7.3).

**Addition of graphic units to a generic representation**

The addition of a graphic unit to a generic representation results in a new generic representation. For example, adding the graphic unit (18) *Structural tartan grid* to the generic representation *Axial System in Specified Form* (18) leads to the generic representation *Axial System in Specified Form in Structural Tartan Grid* (39). For a more comprehensive discussion, see section 3.4.

**Themes of generic representations**

Generic representations that deal with the same subject can be grouped. For example, the generic representations *Simple Contour* (1), *Combination of*
Contours (2), and Complementary Contours (3) are the only generic representations that exclusively deal with the shape and place of the building edge. They constitute the theme “Shape.” In the set of generic representations that have been identified, there are two other themes. These are “System” and “Structure.” Combinations of the themes also occur (“Shape and System,” “Shape and Structure,” “System and Structure,” and “Shape and System and Structure”).

The themes as such are internally organised following the previous two principles of successive graphic units and addition of graphic units. In the broad outline of a design, the themes can be used to characterise whether the architect starts with Shape, System, or Structure.

A sequence of generic representations

By making sequences of themes, and sequences of generic representations within themes, possible sequences of generic representations are defined. From these, a particular sequence can be defined which forms the basis for knowledge acquisition of a building type. Achten (1997a, 1997c) presents a sequence of 23 generic representations.

1 Simple Contour (1). The first generic representation of the theme “Shape.” Establish a building envelope.
2 Combination of Contours (2). Tentatively define major parts of the building envelope.
3 Specified Form (H50): Tentatively define the dimensions of the building envelope.
4 Complementary Contours (3): Locate the building shape in the site. Step one up to four deal with the building envelope exclusively. The next steps in the sequence deal with “shape and structure.”
5 Zone (H48): Establish organizational zoning of the building plan.
6 Schematic Subdivision in Zone (H15): Establish general layout within the zone.
7 Schematic Subdivision in Zone in Contour with Function Symbols (49): Mark functions in the schematically subdivided zone and apply it to the building envelope.
8 Zone in Specified Form (16): Determine the zoning in the tentative dimensions of the building.
9 **Schematic Subdivision** (9): Come up with a subdivision principle to distinguish main parts of the building.

10 **Schematic Subdivision in Contour** (21): Apply the subdivision principle to the contour of the building.

11 **Grid** (H45): Establish a grid module.

12 **Schematic Subdivision in Grid** (19): Coordinate the main parts of the building to the grid.

13 **Schematic Subdivision in Specified Form** (H58): Apply the subdivision principle to the tentative building layout.

14 **Schematic Axial System** (8): Establish a set of axes for the building organisation.

15 **Axial System in Specified Form** (18): Apply the axes to the tentative building layout.

16 **Contour in Grid** (15): Apply the building envelope to the grid.

17 **Zone in Contour in Grid** (36): Apply the zoning to the building envelope and coordinate it with the grid.

18 **Partitioning System in Contour** (22): Establish the partitioning within each of the main parts of the building.

19 **Circulation Scheme** (13): Establish a circulation principle.

20 **Circulation in Contour** (32): Apply the circulation principle to the building envelope.

21 **Element Vocabulary** (11): Establish an element vocabulary to indicate functional use in the building.

22 **Element Vocabulary in Contour** (H30): Apply the element vocabulary to the building.

23 **Element Vocabulary and Function Symbols and Grid in Specified Form** (48): Establish the functional layout of the building design.

### 4.3 Knowledge application in the sequence

By following the sequence as outlined in the previous section, a series of design decisions is established, including the knowledge required for those decisions (Achten et al. 1998). The first generic representation of the sequence is *Simple Contour* (1). By establishing the contour of the building, the following parameters are defined: `number_of_floors`, `surface_area`, and `story_height`. 
In order to make decisions about `surface_area`, it is necessary to know about aspects that influence floor area, in this case: minimum feasible surface area, gross area/net area ratios, and installations ratios. The `surface_area` is related to the `number_of_floors`, which is influenced by: definitions of low-, medium, and high-rise buildings, `story_height`, and structural principles. In this manner, the generic representation determines which knowledge from the office building type needs to be acquired in the particular stage of design termed `Simple Contour` (1).

The following two examples demonstrate how declarative knowledge is applied to a generic representation: *Simple Contour* (1) and *Specified Form* (H50). What is important to observe is not the specific application of knowledge. This is rather straightforward and can be represented in many other ways. What is interesting to note, is how all of the following reasoning per generic representation is ‘encapsulated,’ as it were, simply by making that particular drawing.

**Knowledge application in Simple Contour (1)**

Suppose the following simple brief for an office building: a single-purpose, low-to medium-rise office building, which poses no special requirements on structural or installation features. The future tenants are anonymous. The site is a rectangular area measuring 75x75 m², oriented north-south. Nearby buildings pose no special circumstances with respect to obstruction, shading, distance from site boundaries, etc. The useful floor area to be realized is 5500 m². The building is to be rented out for office space; therefore, each floor is destined for office use.

The *Simple Contour* (1) starts with drawing the shape of the office building. Suppose for example, that a T-shape layout will be used. The following parameters are defined: `surface_area`, `number_of_floors`, and `story_height`. The shape requires design decisions about the surface area of a floor and the number of floors.

**Surface area of a floor**

Relevant statements for office buildings about the surface area of a floor are:

- The minimal economically feasible surface area of an office floor is 600 m².
- Office buildings typically have a Gross Area/Net Area ratio of 1.35.
• Two percent of the building floor area served by the boiler and chiller will provide a room large enough for the chiller and boiler and their accompanying pumps.

• A room large enough for the air handler will be provided by 4% of the building floor area served.

The surface area of each floor of the T-shape must exceed 600 m². With respect to the gross area of the office building, the functional area stated in the brief can be multiplied by 1.35 which means 1.35x5500 m² = 7425 m². HVAC takes up 2% of this (148.5 m²) for boiler and chiller and 4% of this (297 m²) for air handling. The surface area of a floor is related to the number of floors and the maximum dimensions of the envelope within the site.

**Number of floors**

Relevant statements about office buildings concerning the number of floors are:

• A high-rise office building is a building with the top floor 22 m above the site.

• Floor height usually has the dimension of 3.00 m, 3.10 m, 3.40 m, 3.70 m, or 4.20 m.

• Buildings with floors more than 18.3 m above ground level require at least one fire-fighting stair which must have direct access to open air at ground level.

• From a structural point of view, the following classes of number of floors can be distinguished: (1) 1-4 floors, (2) 5-7 floors, and (3) 8-10 floors. For each class, a number of structural systems are advisable for stability.

Given the minimum requirement of 600 m² and the gross surface area of 7425 m² it is possible to establish a range of possible floor areas: 8 x 928 m², 7 x 1061 m², 6 x 1238 m², 5 x 1485 m², 4 x 1856 m², 3 x 2475 m², 2 x 3713 m², en 1 x 7425 m². More than eight floors are not possible within the low- to medium rise building constraint. A tentative choice is made for 4 x 1856 m², and for a story height of 3.70 m.

In *Simple Contour* (1) the following characteristics of the building design are established: `surface_area (1856 m²), number_of_floors (4 stories), and story_height (3.70 m)`. The dimensions of the building envelope are not established, although the shape indicates a building with three wings (T-shape).
Knowledge application in Specified Form (H50)
The form of the simple contour determines a number of tentative dimensions. Parameters defined in this generic representation are orientation, perimeter, and dimensions. Decisions implied by establishing a Specified Form (H50) are orientation, length of the wings, and depth of the wings.

Orientation
Relevant statements about office buildings concerning the orientation are:
• Orientation of the main axis usually is E-W in the USA and S-N in Europe. A south-north orientation is chosen for the building, positioning the T-shape as a ‘T’ in the site.

Length of the wings
Relevant statements about office buildings concerning the length of the wings are:
• A circulation point (stairs) may be no further than 25 m from the end facade of a wing and no more than 50 m from another circulation point.
• Dead ends (no exit provision at the end) may be no deeper than 6.10 m.
• The maximum distance between workplace and egress is 30 m.
• An office space typically is 4.50-6.00 m deep.

The positioning of circulation points (stairs and elevators) forms an important factor in determining the dimensions of the building. A first estimate of the length of the wings can be made by assuming equal dimensions for length and width of the wings. The length of the building then is 64.8 m, and the depth is 43.2 m. Since in this case one circulation point is always over 25 m distance from one end facade of a wing, at least two circulation points are required along the long side, and one circulation point in the short wing of the building.

Depth of the wings
Relevant statements for office buildings about the depth of the wings are:
• The floor space within 7.62-9.14 m of the facade provides premium rentals, resulting in slab-like office buildings, usually some 18.3-21.3 m wide and 46 m long.
• There are four basic depths of space: shallow space (4-5 m), medium depth space (6-10 m), deep space (11-19 m), and very deep space (over 20 m).
• From 1969-1980, the average area of floors in new offices starting construction, dropped from ca. 3500 m² to ca. 1000 m². The typical depth dropped from ca. 21 m to ca. 14 m with a highest peak of ca. 30 m in 1972.

• The pressures from users, office electronics, energy conservers and in consequence the rate-paying tenant of office buildings, all point towards medium depth buildings (14-17 m across) as an attractive depth for both speculative and custom-designed developments.

• Daylight can be used up to a space depth of 7 m. Working at the facade is desirable with respect to daylight provision. If wing depth is chosen smaller, the building becomes longer. Given the trends towards more shallow space, and the limitation of the site of 75 m, the most shallow depth of the wing can be 18,0 m. Therefore, within the parameters of the building shape, the wing depth is 18,0 m or more, and the wing length is 28,5 m or less. If this class of wing depth is unsatisfactory it is necessary to either go back to Simple Contour (1) and choose more floors with a lesser amount of surface area (5 x 1485 m², 6 x 1238 m², or 7 x 1061 m²), or to choose another shape for establishing Simple Contour (1).

Specified Form (2) establishes the dimensions and orientation of the office building’s perimeter. The orientation chosen is north-south. The dimensions are: wing length (21 m), wing depth (22 m), and floor area (1870 m²), resulting in a perimeter of 214 m.

An expert system for office buildings
The first seven steps of the sequence of 23 steps in section 4.2 have been applied to the office building type and implemented in a Frame-Based design aid system. This work has been performed in the context of an expert systems course, where students implemented the sequence (Achten et al. 1995). The main knowledge structure underlying all systems is the so-called frame (Coyne et al. 1990), which encodes the sequence of generic representations by assigning each slot to a single generic representation. For the implementation it was not required to have a flexible order of generic representations, thus the sequence of slots provided the order in which the system was executed (Figure 7: next page).
Top: A sequence of steps for an office building in an implemented expert system utilising generic representations (work by student B. de Haan).

Bottom: Continuation of development of office building plan in expert system.

The systems were programmed in AutoLISP in an AutoCAD environment. Each slot of the frame runs a subroutine that solves that particular generic representation and loads the results in the frame. The reasoning steps involved in two of these slots (Simple Contour (1) and Specified Form (H50)) are outlined above. Although the systems have limited functionality, they quickly assist the architect to establish a basic building envelope, tentatively establish its dimensions, locate it in the site, decide upon a zoning principle, and apply it to the envelope. The various implementations demonstrate quite clearly that establishing a generic representation requires multiple cycles of considering alternatives and variants, which are all expressed in the same graphic manner.

4.4 Overview and scope

The notions of graphic unit and generic representation make it possible to identify design decisions and related required knowledge of a building type. Furthermore, they enable support of the design process in a manner which is much related to the way architects work. Generic representations allow structured knowledge acquisition of a building type, and can establish a sequence of design decisions for that particular type.

By showing how generic representations encode design issues that are helpful in the early stage of design, the work points to directions to support architects in a more architectural fashion in design aid systems. A CAAD system that operates via generic representations or that could identify generic representations in a drawing is able to identify the knowledge required at that
particular stage in the design process (computational approaches to these questions are addressed in Part Three).

At this point, the knowledge base has the following characteristics:

- A broad scope of knowledge of the office building type.
- Variables, terms, and units are informally used without explicit definition.
- Statements have varying degrees of complexity.
- The form of the statement is not fixed in a precise way.
- There is no explicit check on consistency.

Many statements depend on assumptions such as the nature of organizations, general requirements posed by employees, architectural style, economical relationships between structural span and flexible workplace, etc. General design issues such as composition and style are not covered in the knowledge base. The application of generic representations is limited to the early stages of design where graphic representations considerably aid in concept formation of the design. They seem less powerful in the later, more detailed, stages where complexity of the drawings is much larger.
Chapter 5  Analytical Technique of the Design Process

In this Chapter, it is demonstrated how generic representations form the framework for analysing a design process. Such an analysis proceeds on the drawings that are produced during the design process. Generic representations are used to assess the design decision-making aspects involved per drawing. On this basis the development of the design process is described. In order to make the description more precise, a formal language called Feature-based modelling is used to describe the design steps. The method of analysis is presented, after which a concrete case of a house design is analysed using this methodology. The analysis leads to a classification of changes in the design process.

5.1 Methodology for analysis of a design process

As outlined in section 2.2, a phase in the design process is captured by the drawings made in that phase. The office building example in Chapter 4 has shown that analysis of the graphic units present in the drawing can help identify design decisions that are involved with these graphic units. In order to precisely describe what happens in drawing, a formal language is required. Before describing the methodology of the analysis itself, Feature-based modelling is introduced as the formal language of choice.

Feature-based modelling technique

Feature-Based modelling (FBM) originates from areas of Mechanical Engineering. Early papers on FBM provide the background and history of these techniques (Cunningham 1988, Shah 1991, 1994, and Bronsvoort and Jansen 1993, and Bronsvoort et al. 1996). FBM has been reviewed for its relevance to architectural design in van Leeuwen et al. (1996, 1997) – see also Hendricx (2000) for a related approach. Based on this review, the technique has been modified and extended for information modelling in architectural design, which requires a more open-ended approach. The following definition of the term Feature is employed (van Leeuwen and Wagter 1998; van Leeuwen 1999):

A Feature is a collection of high-level information, possibly emerging during design, defining a set of characteristics or concepts with a semantic meaning to a particular view in the life-cycle of a building.
This definition reflects four important aspects of Feature modelling:

1. A Feature has high-level information with semantic meaning.
2. Both physical and non-physical characteristics and concepts can be defined.
3. Definition and use of emerging Features during design is supported.
4. A Feature relates to a particular view in the life-cycle of a building.

Van Leeuwen (1999) provides a Feature modelling framework for the development of information modelling systems for support of architectural design. On the highest level, there are classes of Feature Types that form the main kinds of information: Simple FeatureType, Enumeration FeatureType, Geometric FeatureType, Complex FeatureType, Constraint FeatureType, and Handler FeatureType. Feature Types are instantiated into Feature Instances. Feature Instances contain actual building information of a design.

Feature Types are conceptual pieces of knowledge, and Feature Instances represent the actual state of a concept in the design. Take for example, a Feature Type called ‘UnitPrice,’ defined to represent information about cost. It is a Simple FeatureType, having a basetype ‘real,’ and a unit ‘Euro/month.’ A Feature Instance of the Feature Type UnitPrice could be Rent_Office1, with a value of 600 Euro/month. A design has many spaces, so it can have instances called Rent_office1, Rent_office2, Rent_office3, Rent_office4, etc.

Feature models are flexible by supporting alteration of specific Feature Types during the design process. They are extensible because it is possible define new Feature Types and Feature Instances. Also, it is possible to define relations between Feature Instances that have not yet been foreseen at the Feature Type level. This dynamic character of Feature modelling seems to be in accordance with the dynamic nature of design. A so-called “Feature Tool” has been developed in the group to define Feature Types and Feature Instances and to manipulate them. In this Chapter, Features are described in a textual representation for its brevity (Van Leeuwen and Wagter 1998).

The analysis has been done in collaboration with van Leeuwen (Achten and van Leeuwen 1998; 1999). It has to be noted that the Feature-Based modelling approach used in this research has now been superseded by Concept modelling. It leads too far from the present work to discuss the details of this change: the interested reader is referred to van Leeuwen and Fridqvist (2002). The main approach of modelling information however, holds for both techniques.
Analysis of material

The analysis of the design process is based on the drawings made by an architect for an architectural assignment that has been realised in practice. Each single drawing is taken as a step in the design process for which a Feature model can be established. The transition from one step to the next therefore, represents the design decisions taken from that phase to the next, similar to the outline discussed in section 2.2. A phase therefore, is defined as: one single drawing, with notation: phase n. A step is defined as: the transition from phase n \(\rightarrow\) phase n+1. Since the brief is a textual document, and not a drawing, the brief is defined as phase 0.

The drawings made during the design process were delivered by the architect’s office. As the drawings were not time-stamped, the sequence of drawings had to be established first, in consultation with the architect. In the reconstructed sequence, each drawing was described on the basis of its constituent elements. This description was based on the same method as the analytical method described in Chapter 3 for identifying graphic units and generic representations. In an interview, the architect was asked to verify the sequence and to make changes where appropriate. It appeared that analysis through the sequence of drawings and description per drawing is a productive way to extract design information. It has to be noted that to ask for verification after the design process has been concluded, may lead to post-rationalisation (in the words of Darke 1979), and to unwarranted agreement if the description seems plausible enough. These aspects can be met when such comments are made during the actual process, using protocol-analysis, and by time-stamping the material.

After the sequence of drawings is established, the analysis of the design case is carried out as follows:

- In phase n, identify the elements that have to be defined as Features. These elements are concepts used in the design, and typically are nouns in the text or graphic units in the drawings.

- If the elements are new, define a complex Feature Type or simple Feature Type for each new element. An element is defined as a complex Feature Type when it cannot be defined as a simple Feature Type (string, integer, real, Boolean, or enumeration). If the element in question already has been defined as a Feature Type in any previous phase, record which changes have taken place and
determine whether these should be included at the Type level or the Instance level.

- In the case of new elements, establish Feature Instances based on the new Feature Types defined in step 2. In the case of existing Feature Types, record changes in Feature Instances. Proceed to phase n+1.

Example: the Feature Type Space
The brief provides a lexicon of design elements that play a role in the building design. A major element in the brief are the required spaces in the house, such as “hall,” “toilet,” and “wardrobe.” First, they have to be defined as Feature Types, after which Feature instances can be made.

Rather than defining a Feature Type for each kind of space (such as a Feature Type Hall for the hall, Feature Type Toilet for the toilet, etc.), a more general Feature Type for the concept of space is defined, of which the various spaces in the brief are instances.

Elements of the Feature Type
The text in the brief notes the following aspects of a space:

- Function (such as “bedroom” and “bathroom”).
- Contained elements (such as “stair” and “toilet”).
- Visual relationship (such as “kitchen closed with respect to living”).
- Access relationship (such as “doors to garden and bathroom”).
- Daylighting (such as “daylighting in kitchen”).
- Adjacency (such as “scullery between garage, kitchen and bathroom”).
- Rooftype (such as “no glass roof”).
- Number of persons (such as “guest room for two persons”).

Determining which aspects are to be included in the definition of the Feature Type Space and which aspects are to be defined in other Feature Types is not straightforward. If the aspect concerns only the space itself, and does not refer to other elements, then it can be included in the type. Following this guideline, constraint-like relations such as visual relationship are better defined outside the Feature Type. Function, contained elements, daylighting, rooftype, and number of persons are within the particular space and therefore are included in the type definition. The Feature Type space is defined accordingly, and results in the following:
complex BuildingElement.space.Space {
    Has BuildingElement.space.Space contains[0..?];
    Spec User.value.Daylighting daylightIsUsed;
    Spec User.value.Function function;
    Spec BuildingElement.structure.Rooftype kindOfRoof;
    Spec User.value.NumberOfPersons numberOfPersons;
}

Identifier of the Feature Type Space
The first line identifies the Feature Type class, which is a ‘complex’ class because it is composite in character and contains many different kinds of other Feature Types. The text BuildingElement.space.Space is the Feature identification in the context of a Feature Type library.

Aspects of the Feature Type Space
The next five lines define the aspects of the Feature Type Space as identified above. They are the contained spatial elements Space, and the aspects Daylighting, Function, Rooftype, and numberOfPersons respectively.

Definition of an aspect
Each line has a three-part structure: relation, FeatureID, and role. Four of the relations are specifications (Spec) since they further define the space. The contains relation is a decomposition since the contained spaces are part of the space. The FeatureIDs refer to Feature Types that are related to the Feature Type Space. They are defined below. The role describes the role of the Feature in the definition. The numbers in brackets (for example [0..?]) indicate cardinality of the relation: how many instances of this role are allowed or required in a Feature Instance.

Definition of implied Feature Types
In order to complete the Feature Type definition of space in this phase, the Feature Types User.value.Daylighting, Function, and NumberoOfPersons, Rooftype must be defined as well. Since at this point they are only mentioned in the brief, the Feature Types are simply defined by naming the concepts:

string BuildingElement.structure.Rooftype {
}

boolean User.value.Daylighting {
    TypeDefault {True}
}
Feature Type Space in other Feature Types

The Feature Type Space is not only used to make Feature Instances of spaces such as hall and kitchen, but also in all other cases where in the brief of phase 0 a space is mentioned. This is the case for example, in the access relationship between spaces. Constraint.access.Space_Space is the corresponding Feature Type:

```plaintext
complex Constraint.access.Space_Space {
    Spec BuildingElement.space.Space access[2..2];
}
```

Example: feature instances of Feature Type Space

The Feature Types have to be instantiated into the particular Features Instances used in the design. In this way, the Feature instances describe the design, and the Feature Types the concepts used in the design. In the case of Feature Type Space, this means making instances of spaces such as hall, living, kitchen, veranda, etc. The Feature instance for the living is made as follows:

```plaintext
BuildingElement.space.Space Living = {
    contains[1] = Dining
    contains[0] = Sitting
    function = FunctionLiving
}
BuildingElement.space.Space Dining = {
    function = FunctionDining
}
BuildingElement.space.Space Sitting = {
    function = FunctionSitting
}
User.value.Function FunctionDining = {
    Value {"Dining"}
}
User.value.Function FunctionLiving = {
    Value {"Living"}
}
User.value.Function FunctionSitting = {
    Value {"Sitting"}
}
```
5.2 The case: house design

The case is a design for a small house, executed by an architect’s office. The case was taken from a real design task, of a project that was realised as a building. The project architect in question uses the CAAD software AutoCAD from the very start of the design work. Drawings during the design process are made as new copies rather than changing or revising old drawings. In this way, all the drawings that are produced during the design process are easily available for analysis. Furthermore, since the drawings are produced with CAAD software, they are also clear for analysis and expression in terms of features. The drawings obviously are not of the sketch-kind that is generally associated with architectural design. The reason for using these drawings is very pragmatic: for the Feature-based description of the design process, it is more important to show how the Feature-definitions come about, and how they change during the design process, than that the material taken from the case conforms to general expectations what the drawings look like. Sketches would involve too much uncertainty in terms of interpretation of the drawings.

The material available for analysis consists of: one print size A4 with the brief termed Brief, four prints size A4 with drawings made during the design process termed Compilation 1 through 4, and four prints size A2 termed Print 1 through 4. The Brief document is the result of the architect’s interview with the client about his wish to have a house built. The prints Compilation 1-4 form the basis of the analysis. They contain 30 drawings made during the design process. The documents Print 1 through Print 4 show the finalised drawings of design proposal. Since these latter documents are about the final design, they are not considered in the analysis of the design process.

The sequence of drawings was first established, and after that the architect was asked to make corrections on this sequence so that it reflected the actual development of the design process. Based on the corrected sequence, we asked the architect what the design intentions in each step of the sequence were (Achten and van Leeuwen 1998; 1999). Based on this sequence, the design process was described in terms of Feature Types, Feature instances, and changes in the Feature model. Table F shows the first eleven drawings of the design process. In the left column the drawing is shown, and in the right column the architect’s intentions.
**Table F: The first eleven phases of the house design case.**

<table>
<thead>
<tr>
<th>Drawing</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Phase 0" /></td>
<td>Phase 0: The architect interviews the client and summarises the brief on a A4 sheet which also contains some first small sketches indicating the desires of the client.</td>
</tr>
<tr>
<td><img src="image2" alt="Phase 1" /></td>
<td>Phase 1: The brief is translated into spaces positioned in the site, giving an indication of the required space and the organisation of the ground floor plan. The main mass is located in the northwest, leaving space for the garden. The module used is 1.20 m. It is a ‘sketch-module’ used by the architect. Graphic units in the drawing are (2) Contour, (7) Function symbols, and (3) Measurement device.</td>
</tr>
<tr>
<td><img src="image3" alt="Phase 2" /></td>
<td>Phase 2: Most spaces are placed on the ground floor. The facade study shows the building as it is realized in the vertical plane, including window position and roof shape.</td>
</tr>
<tr>
<td><img src="image4" alt="Phase 3" /></td>
<td>Phase 3: The architect uses a grid from the start, establishing space dimensions. The use of a second grid is based on the location of the double sized garage in the upper left corner. Access to the garage in a straight line would make this element visually too dominant, which is why part of the design is rotated on the basis of a second grid. The first grid = grid 1. The second grid = grid 2. Graphic units in the drawing are (2) Contour, (16) Grid, and (3) Measurement device.</td>
</tr>
<tr>
<td><img src="image5" alt="Phase 4" /></td>
<td>Phase 4: The centre part of the building is oriented towards grid 2. Spaces are placed according to the new grids to see how they work out. Graphic units in the drawing are (2) Contour, (7) Function symbols, and (3) Measurement device.</td>
</tr>
</tbody>
</table>
**Table F: The first eleven phases of the house design case – continued.**

<table>
<thead>
<tr>
<th>Phase 5</th>
<th>In this phase, the basic variant is introduced. The building mass consists of two squares between which a centre part is defined, which is rotated relative to the squares. This basic variant is used throughout the following design process. Upper left square = square A. Bottom right square = square B. The centre piece = centre piece. Graphic units are (1) <em>Simple contour</em> and (2) <em>Contour</em>.</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Phase 5 Diagram" /></td>
<td><img src="image2" alt="Phase 5 Diagram" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Phase 6</th>
<th>Sub variant of phase 5. Square B is lifted from the ground floor (dots indicating columns). The center part has acquired an L-shape lying against square A. The L-shape is dominant with respect to the other squares A and B. The bottom-side of the L is aligned with bottom right corner of square B. In that point, lines of grid 1 and grid 2 intersect. Square A is kept intact. Graphic units are (1) <em>Simple contour</em>, (2) <em>Contour</em>, and (5) <em>Elaborated structural contour</em>.</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image3" alt="Phase 6 Diagram" /></td>
<td><img src="image4" alt="Phase 6 Diagram" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Phase 7</th>
<th>Facade study of phase 6, showing particular roof shape for this design.</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image5" alt="Phase 7 Diagram" /></td>
<td><img src="image6" alt="Phase 7 Diagram" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Phase 8</th>
<th>Facade study of phase 6, showing particular roof shape for this design.</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image7" alt="Phase 8 Diagram" /></td>
<td><img src="image8" alt="Phase 8 Diagram" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Phase 9</th>
<th>Sub variant of phase 5. Square B becomes subordinate to the centre piece, which in turn becomes subordinate to Square A. The L-shape becomes more implicit in the contour of Square A which follows both the square and the centre piece. The two rectangles to the right are for indication of the size of the kitchen and scullery. Graphic units are (1) <em>Simple contour</em> and (2) <em>Contour</em>.</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image9" alt="Phase 9 Diagram" /></td>
<td><img src="image10" alt="Phase 9 Diagram" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Phase 10</th>
<th>First concept design in which all spatial elements are organised as in the final plan. The utilitarian spaces are located in the centre piece, Square B contains the garage, and Square A the living. In the remainder of the design process, this concept will be further refined and elaborated, but there will occur no more great changes in the basic functional layout. Graphic units are (1) <em>Simple contour</em>, (2) <em>Contour</em> and (19) <em>Element vocabulary</em>.</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image11" alt="Phase 10 Diagram" /></td>
<td><img src="image12" alt="Phase 10 Diagram" /></td>
</tr>
</tbody>
</table>
Feature-based description of phase 0

In subsequent phases the Feature model is established on the basis of the previous phase, which makes it possible to track changes of the design during the design process. In phase 0, the brief is translated into a Feature model. The brief consists of elements of the design (spaces and objects such as bathtub, toilets, etc.) and of relations between elements.

Constraints in the Feature-based model

The relations can be viewed as constraints. Some of these constraints concern spatial relations. Adjacency constraints between two Feature Types A and B can be expressed as: A _adjacent_ B, the contains constraint as A _in_ B, and the vertical constraint as A _above_ B. Their logical opposites are A _NOTadjacent_ B, A _NOTin_ B, A _NOTabove_ B.

Other constraints concern access from one element to another. These access constraints can be expressed as: A _access_ B and the opposite A _NOTaccess_ B. Since this relation is reciprocal, only one _access_ constraint has to be defined.

The third type of constraint is expressed in the statement ‘kitchen is closed with respect to the living,’ meaning that there is no visual relation between the kitchen and living space. The constraints A _visual_ B and the opposite A _NOTvisual_ B are reciprocal, and can be defined in the same manner as the _access_ relation.

The ‘fireplace in living’ constraint can be expressed in two ways: as a Fireplace_in_Living instance of a Heating_in_Space constraint Feature Type, or by establishing an association-relation in the Feature Instance Living with the Feature instance Fireplace. The first option allows more flexibility because heating elements are bound to occur in more spaces.

Elements and constraints in the Feature-based model

The tables below show the Feature Types (Table G) and Feature Instances (Table H) for elements and constraints respectively.
<table>
<thead>
<tr>
<th>Feature Type (Supertype)</th>
<th>Feature instance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Space</td>
<td>Hall, Toilet, Wardrobe, Living, Sitting, Dining, Kitchen, Veranda, Scullery, Garage, Bedroom, Bathroom, Shower, GuestRoom</td>
</tr>
<tr>
<td>Door (ElementInWall)</td>
<td>DoorBathroom_Bedroom, DoorBathroom_Garden</td>
</tr>
<tr>
<td>Floor</td>
<td>FloorLiving</td>
</tr>
<tr>
<td>Material</td>
<td>MaterialFloorCovering</td>
</tr>
<tr>
<td>Roof</td>
<td>MaterialRoofVeranda</td>
</tr>
<tr>
<td>Stair</td>
<td>Stair</td>
</tr>
<tr>
<td>Garden</td>
<td>Garden</td>
</tr>
<tr>
<td>Chair (Furniture)</td>
<td>Chair</td>
</tr>
<tr>
<td>Table (Furniture)</td>
<td>Table</td>
</tr>
<tr>
<td>Fireplace (Heating)</td>
<td>FireplaceLiving</td>
</tr>
<tr>
<td>Bathtub (Sanitary)</td>
<td>BathtubBathroom</td>
</tr>
<tr>
<td>ToiletPot (Sanitary)</td>
<td>ToiletPotHall, ToiletPotGuestroom, ToiletPotBathroom</td>
</tr>
<tr>
<td>WashBasin (Sanitary)</td>
<td>WashBasin1_Bathroom, WashBasin2_Bathroom, WashBasinGuestroom</td>
</tr>
<tr>
<td>Daylighting</td>
<td>DaylightingBedroom, DaylightingKitchen, DaylightingVeranda</td>
</tr>
<tr>
<td>Function</td>
<td>FunctionBedroom, FunctionHall, FunctionDining, FunctionKitchen, FunctionSitting, FunctionLiving</td>
</tr>
<tr>
<td>Storey</td>
<td>StoreyGroundFloor, StoreyFirstFloor</td>
</tr>
<tr>
<td>NumberOfPersons</td>
<td>NumberOfPersonsGuestroom</td>
</tr>
</tbody>
</table>
Table II: Feature Types and Feature Instances of constraints in phase 0.

<table>
<thead>
<tr>
<th>Feature Type (constraint type)</th>
<th>Feature instance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Space_adjacent_Space (spatial)</td>
<td>Kitchen_adjacent_Living, Veranda_adjacent_Living, Scullery_adjacent_Garage, Scullery_adjacent_Bedroom, Scullery_adjacent_Kitchen</td>
</tr>
<tr>
<td>Space_adjacent_Garden (spatial)</td>
<td>Kitchen_adjacent_Garden, Bedroom_adjacent_Garden</td>
</tr>
<tr>
<td>Storey_above_Storey (spatial)</td>
<td>Storey1_above_Storey0</td>
</tr>
<tr>
<td>Furniture_NOTin_Space (spatial)</td>
<td>Furniture_NOTin_Kitchen</td>
</tr>
<tr>
<td>Heating_in_Space (spatial)</td>
<td>Fireplace_in_Living</td>
</tr>
<tr>
<td>Stair_NOTin_Space (spatial)</td>
<td>Stair_NOTin_Living</td>
</tr>
<tr>
<td>Space_Space (access)</td>
<td>Bedroom_access_Bathroom</td>
</tr>
<tr>
<td>Space_Garden (access)</td>
<td>Bedroom_access_Garden, Kitchen_access_Garden</td>
</tr>
<tr>
<td>Space_NOTvisual_Space (visual)</td>
<td>Kitchen_NOTvisual_Living</td>
</tr>
</tbody>
</table>

With these Feature Types of elements and constraints, all the statements in the brief in phase 0 are formally described.

**Feature-based description of phase 1**

In phase 1, the brief is transformed to a set of spaces in a drawing. Significant changes with respect to phase 0 consist of assigning shapes and their dimensions to spaces, and locating them in the site by means of a grid.

**Geometry in the Feature-based model**

The notion of shape can be implemented as part of the existing Feature Type Space or by defining a new Feature Type for Shape, which is associated with the Feature Type Space. Because shape is very basic in architectural design, and many other kinds of shapes may occur later in the design, a supertype 2DShape is defined of which Rectangle is a subtype. For the rectangle and its position in
a drawing, the dimensions and position need to be defined as Feature Types as well (resulting in Length, Point, and Coordinate).

```plaintext
complex Geometry.shape.2DShape {
    TypeDescr {"General shape definition with point of reference"}
    Spec Geometry.topology.Point referencePoint[1..1];
}
```

This description follows the same principle as outlined above in the example of Feature Type Space.

```plaintext
complex Geometry.shape.Rectangle(Geometry.shape.2DShape) {
    TypeDescr {"Rectangular shape with dimensions and reference point"}
    Spec Geometry.dimension.Length length[1..1];
    Spec Geometry.dimension.Length width[1..1];
}
```

```plaintext
real Geometry.dimension.Length {
    TypeDescr {"Linear dimension in m"}
    TypeDefault {1}
    TypeUnit {"m"}
}
```

Length is expressed in decimal numbers, so the Feature Type Length is a “real.” The line “TypeDef {1}” means that if we don’t say anything, an instance will have value 1. The line “TypeUnit {“m”}” means that the length will be measured in metres, and that a metre is expressed with “m.”

```plaintext
complex Geometry.topology.Point {
    TypeDescr {"Point in orthogonal axial system with x, y, z co-ordinates"}
    Spec Geometry.topology.Coordinate xcoordinate[1..1];
    Spec Geometry.topology.Coordinate ycoordinate[1..1];
    Spec Geometry.topology.Coordinate zcoordinate[1..1];
}
```

The point that was used earlier for 2DShape, is now defined as a Feature Type. The three lines with specialisation state that any point has exactly 1 x-coordinate, 1 y-coordinate, and 1 z-coordinate. Again, the coordinates are also Feature Types, which also have to be defined:

```plaintext
real Geometry.topology.Coordinate {
    TypeDescr {"Coordinate along an axis in an axial co-ordination system"}
    TypeDefault {0}
    TypeUnit {"m"}
}
```

Since a coordinate can be any decimal number along an axis, it is a “real.” For the rest the default value is defined as 0, and the coordinates are expressed in metres.
All Feature Types are available now that are necessary to describe what happens to the Feature Type `Space` when we move from the list of spaces in phase 0 to a first schematic drawing in phase 1. The Feature Type `2DShape` is associated with the existing Feature Type `Space` (bold line shows addition to the old Feature Type):

```plaintext
complex BuildingElement.space.Space {
    TypeDescr {"Space element within which activities can take place"}
    Spec BuildingElement.space.Space contains[0..?];
    Spec User.value.Daylighting daylightIsUsed[1..1];
    Spec User.value.Function function;
    Has BuildingElement.structure.Rooftype kindOfRoof;
    Spec User.value.NumberOfPersons numberOfPersons;
    Assoc Geometry.shape.2DShape shape;
}
```

In the current state of phase 1, the following is known about spaces: they can contain zero to any number of other spaces (line 3: “Spec BuildingElement.space.Space contains[0..?]”), they either use daylight or not (line 4), they have a function (line 5), they have some kind of roof (line 6), and they can accommodate a number of people (line 7). What happens in the transition from phase 0 to phase 1, furthermore, is that by means of an association-relation the space is given a particular shape.

**Grid in the Feature-based model**

The shapes are drawn in a grid that is used for co-ordination. The grid is defined by its module and a point of origin (this also accommodates the use of multiple grids). Both the origin of the grid and the position of elements in the grid require a set of co-ordinates. On the instance level therefore, a Feature Instance `Origin` (instance of `Geometry.topology.Point`) is defined, relative to which measures can be taken and grids positioned. The left-bottom corner of Grid_1 is placed on the `Origin`.

```plaintext
complex Geometry.Topology.Grid {
    Descr {"Origin and module of a grid"}
    Spec Geometry.topology.Point originOfGrid[1..1];
    Spec Geometry.dimension.Length moduleOfGrid[1..1];
}
```

The positive x-axis is oriented horizontally and to the right of the `Origin`. The positive y-axis is oriented vertically and above the `Origin`, as is customary in architectural design. For the Feature Type `Rectangle`, the reference point is defined as the most left-bottom corner of the rectangle, width and length being measured in the orientation of the positive x and y-axis.
Changes in the Feature-based model

These Feature Types and Instances suffice to describe the state of phase 1. The Feature Instance Kitchen, for example, is changed because of the additional association to the Feature Type Space of which it is an instance, and the definition of its location and dimensions in the associated Rectangle. In phase 1, the Kitchen has dimensions 3.60 m x 3.60 m, and is located on co-ordinates (6.0, 6.0, 0):

```
BuildingElement.space.Space Kitchen = {
    Descr {"Kitchen"}
    daylightIsUsed = DaylightingKitchen;
    function = FunctionKitchen;
    shape = Rectangle_Kitchen;
}
```

```
Geometry.shape.Rectangle Rectangle_Kitchen = {
    Descr {"Rectangular shape for kitchen"}
    length = Length_Kitchen;
    referencePoint = ReferencePoint_Kitchen;
    width = Width_Kitchen;
}
```

```
Geometry.dimension.Length Length_Kitchen = {
    Value {3.6}
}
```

```
Geometry.dimension.Length Width_Kitchen = {
    Value {3.6}
}
```

```
Geometry.topology.Coordinate Coordinate_X_Kitchen = {
    Value {6}
}
```

```
Geometry.topology.Coordinate Coordinate_Y_Kitchen = {
    Value {6}
}
```

```
Geometry.topology.Coordinate Coordinate_Z_Kitchen = {
    Value {0}
}
```

In phase 1 not all spaces mentioned in phase 0 are present, and there are four spaces that have not been assigned a function name by the architect. The older instances of spaces that are included in phase 1 change in the same manner as the kitchen example. Space_0 through space_4 are instantiated directly according to the new Feature Type definition.
5.3 Classification of design actions captured in drawings

The case provides a new way to look at design processes. In particular, changes from one phase to the next can be expressed in terms of changes in the Feature model. In this way, design actions can be matched to Feature model alterations (Achten and van Leeuwen 1998, 1999). What is important to note is, that these changes are captured on the basis of the drawings that are produced in the design process. The extent of the Feature-Based model shows how comprehensive the implications can be that are visible in the drawings.

From the interview with the architect based on the sequence of drawings (Table F: pp. 68-69) we can also see which part of the reasoning is not captured by graphic units or changes in the Feature model. This concerns in particular inferences such as “access to the garage in a straight line would make this element too dominant.” This kind of reasoning may be characterised as design rationale (Moran and Carroll 1996). Capturing such design rationale is quite difficult, and up to now only possible by explicitly asking the designer (e.g. Cerulli et al. 2001).

In the design case, the architect switches between plan-based drawings and studies of the façade to assert the appearance of the design, and to judge which kind of roof-organisation yields a satisfactory composition. Every time the architect makes a move to a different kind of representation than plan-based, analysis by means of graphic units is not possible.

The case demonstrates that graphic units are instrumental to direct the Feature modelling of the steps in the design process. By pointing out the different occurrences of graphic units in the drawings, such as (2) Contour, (7) Function symbols, and (3) Measurement device in phase 1, attention is directed quite straightforward to the associated aspects of the graphic units, such as size and location of the contours, identification of associated functions, and the modular size of the objects.

The changes in the Feature model are based on the case study. The descriptions of the Feature model alterations are very specific for the case. Therefore, it is necessary to classify them into more general descriptions of design actions and associated changes in the Feature model. The classification provides a vocabulary for talking about drawings, the design process, and changes in the design process. The following table presents the classification and the definition of the terms for changes in the Feature model.
Table I: Design actions and changes in the Feature model.

<table>
<thead>
<tr>
<th>Design action</th>
<th>Changes in the Feature-Based model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generalisation</td>
<td>When a group of objects share common properties, define the specific objects as Feature Types, and define a Feature Type (super type) of which they are sub types. The super type functions as generalisation.</td>
</tr>
<tr>
<td>Concept identification</td>
<td>Terms in the brief that are relations or spatial-, material-, and functional elements are defined as Feature Types.</td>
</tr>
<tr>
<td>Element creation</td>
<td>Terms in the brief that are actual parts in the design (“hall”, “floor”, “fireplace”) can be instantiated directly on the basis of the corresponding Feature Types.</td>
</tr>
<tr>
<td>Constraint creation</td>
<td>Terms in the brief that are relations in the design can be instantiated on the basis of constraint Feature Types.</td>
</tr>
<tr>
<td>Concept extension</td>
<td>Adding an association relation to a Feature Type in order to include more characteristics.</td>
</tr>
<tr>
<td>Shaping</td>
<td>Giving shape to the spatial elements involves element creation of the Feature Types Shape and of Feature Types position and dimension.</td>
</tr>
<tr>
<td>Assignment</td>
<td>On the Instance level make an association relation between Feature Instances.</td>
</tr>
<tr>
<td>Move</td>
<td>The co-ordinates that define position have been changed in a Feature Instance.</td>
</tr>
<tr>
<td>Substitution</td>
<td>An existing association between Feature Instances is broken and that one of the Feature Instances is replaced.</td>
</tr>
</tbody>
</table>

This table is not complete since obvious design actions such as the other translations rotate and scale are missing. Further work will have to identify such changes and describe them in terms of changes in the Feature-model.

5.4 Overview and scope

Feature-based models provide a formal language with which to describe generic representations and their internal structure (as stated above, the work has evolved in the meanwhile into Concept-based modelling). The technique of inheritance can be applied to create hierarchies of information-structures that build on top of each other. In this way, low-level definition of very basic concepts such as geometry, shapes, and meaning-labels are accessible to Feature Types that represent more complex entities. On a middle level, Feature Types can represent graphic units, and one step higher, Feature Types can represent generic representations that are built up from graphic unit Feature Types.
As can be seen from the classification of changes in the Feature model, there are many fine-grained operations that do not directly have a bearing on graphic units or generic representations. Changes on instances of graphic units, for example changing the shape of a contour, does not lead to a different graphic unit, but the changes still have to be captured and described. Changing the shape of an element involves element creation and shaping for creating the new form, and substitution for assigning this new form to the former concept that receives it. The mechanisms of change in sequences of generic representations (successive graphic units, addition and subtraction of graphic unit) are captured as element creation, concept extension, shaping, and substitution.

It has to be stressed that Feature-based modelling has been developed for a wider purpose than presented in this document. The goal of Feature-based modelling is to provide an open-ended information structuring formal language that can represent any kind of information relevant in the design process, and that can be changed during the design process. For this reason, part of the approach discussed above may seem quite laborious for the research presented here. The main advantage of this approach is that the results are compatible with other work utilising Feature-based modelling (see for example de Vries and Jessurun 1998b).
Part Three

Productive Utility of Graphic Units and Generic Representations
Chapter 6   Pen Plus: Building a Drawing Tool

The theory of graphic units and generic representations offers many angles to implement intelligent tools that help the architect to structure his design thinking by means of drawing. Such approaches have the substantial advantage that from the very beginning the drawing is built up by well-defined tools, and therefore the internal representation of the design is well-structured, consistent, and accessible for machine reasoning. Disadvantages of this approach are that the architect is required to learn new tools, and that any drawing style that falls outside the scope of the tools cannot be supported, or in all cases, not interpreted by the system. User interface issues and handling become important issues in order not to distract the architect from work at hand.

Two tracks have been developed to build such “Pen Plus” tools: an expert system that assists the designer through the design process by means of generic representations, and a sketch tool with specialised tools for each graphic unit. The expert system approach has been covered in section 4.3. In this Chapter, the focus will be on the sketch tool system.

6.1 Graphic units as basis for a sketch tool

Current drawing systems in the domain of architectural design offer primitives such as lines, rectangles, circles, etc., and enable the user to manipulate them via operations such as transformation, grouping, and aligning. A CAAD system offers primitives tailored to the domain, such as walls, doors, and windows, and has built-in rules to simplify manipulations. These systems have reached a high level of sophistication, and are perfectly suited for the production of the final technical drawings in the closing phases of the design process. However, they do not offer support for the early phase of the design process when concept formation is important and typically supported by sketch-like drawing activity.

In this Chapter, a system called “Structural Sketcher” is outlined that is based on graphic units and generic representations (Pranovich et al. 2002a, Pranovich 2003). The presented work is the PhD-project by Slava Pranovich (Pranovich 2004). The system has tools for making graphic units and also provides a way to define and maintain changes in the relations between graphic units. This is implemented on the basis of a graph representation where the nodes represent
graphic units and the edges relations between graphic units. Changes in the graphic units are propagated through the graph.

Graphic units can be considered as a medium to express the ideas in an architectural design. In the early phases the architect is trying to resolve basic issues of composition, layout, structure, circulation, and preliminary indication of arrangements. As stated before in section 3.3 many graphic units are concerned with structuring the design (e.g. (16) Grid, (8) Zone, (27) Circulation system), rather than describing the design itself (e.g. (2) Contour, (6) Complementary contours, (19) Element vocabulary).

For example, a designer uses an (15) Axial system to show the symmetry between two contours. The (15) Axial system does not only describe the design, but also structures the information in the design. Current drawing systems offer options to mirror graphic objects, but when the operation has been done, the information that two contours are symmetric is lost. Furthermore, whereas an architect often explicitly denotes an axial system, the operation of mirroring often involves only a temporary indication of the mirror action. The presence of an axial system as a separate structuring device and object in the drawing is ignored in such an approach.

A (16) Grid is another good example in this respect. Drawing systems offer grids to position elements more easily, but only as a tool and not as a meaningful element on its own. There is however a strong link between a grid and the objects that coordinated on this grid (Figure 8), and changing one or the other should have consequences for either the grid or the objects that relate to the grid.
Another example concerns the simultaneous use of multiple grids, which structure the design and relate to high-level concepts behind the final design (for example Bax 1985). The architect experiments with different orientations and scales, coordinates objects and grids relative to each other, and observes how the grids influence and enhance each other.

A design system has to incorporate not only the pure graphical and geometric aspects of each individual graphic unit, such as position, orientation, colour, but also the conceptual and abstract level of information representation based on the designer’s understanding of graphic representations. Current drawing and design systems fall short in this respect. The concept of graphic unit and generic representation can be instrumental in this development since they combine graphic information with design content. This implies a design aid system, where all graphic units have the same status: all graphic units are objects that can be manipulated and changed, and have a suitable visual representation.

A second key point concerns the mutual interaction between graphic units, which almost by definition is crucial. Using again, when a designer changes something in one contour, then the changes must be reflected in the other contour also. When the designer changes the position of the axial system, the position of the contours also has to change.

Summarizing, in the Structural Sketcher the user can define a design in terms of graphic units and relations between them, such that during subsequent object manipulation all objects are active and influence each other and react to user actions in a meaningful and predictable way. This should enable an architect to define a design space and to explore this space in an effective, efficient, and hopefully creative way.

6.2 A set of graphic unit sketch tools

The Structural Sketcher system evolved by implementing graphic units, defining the possible relations between graphic units, and handling both graphic units and relationships. Fourteen graphic units have been implemented in the system:

- Five graphic units dealing in some way with spatial shapes (in particular: (1) Simple contour, (2) Contour, (4) Specified form, (5) Elaborated structural contour, and (6) Complementary contours) are considered as contours and defined as sets of vertices \((N_p, p_1 .. p_n, c)\) with \(N_p\) is the number of vertices, \(p_1 .. p_n\) vertices of a contour, and \(c\) is a Boolean value for an open or closed contour. By generalising these graphic units as contours, their distinctive differences by
appearance for the time being is ignored. For the proof of the Structural Sketcher system this is allowed, but a full-fledged sketch support system will have to address these differences.

- Four graphic units dealing with grids (in particular: (16) Grid, (17) Tartan grid, (18) Structural tartan grid, and (12) Refinement grid) are considered as grids and defined as sets \((N_{GC}, p_0 \ldots p_{NGC})\) with \(N_{GC}\) as the number of grid components, \(p_0\) defines the origin of all grid components, and \(p_1 \ldots p_{NGC}\) defines separate grid components. With this definition all kinds of grids can be established, but not distinguished in meaning from each other (for example, a grid has \(N_{GC} = 1\) and a tartan grid has \(N_{GC} = 2\), but this does not further effect behaviour of this grid).

- The graphic unit (15) Axial system is defined as \((p_1, p_2, M)\) with \(p_1\) and \(p_2\) defining the axis, and \(M = \{(g_i, g_j), \ldots\}\) with \(g_i\) and \(g_j\) the mirrored objects.

- The graphic unit (8) Zone is defined as \((N_p, p_1 \ldots p_n)\) with \(N_p\) the number of vertices that define the area of the zone, and \(p_1 \ldots p_n\) the vertices of the zone.

- Three graphic units that deal with standard objects in regular layouts (in particular: (19) Element vocabulary, (20) Structural element vocabulary, and (24) Combinatorial element vocabulary) are defined as images \((b_m, p_1, p_2, p_3, p_4)\) with \(b_m\) a bitmap image, and \(p_1, p_2, p_3, p_4\) the vertices of the parallelogram in which the bitmap fits. This definition, to be honest, is more like a ‘shortcut’ to enable quick insertion of standard layouts. Ideally, it would be implemented in the same form as the contours (structured sets of polygons).

Instances of graphic units and their relations can be added and changed by direct manipulation. Furthermore, each graphic unit and relation has several properties that can be set by the user and that influence how manipulations affect other graphic units. This gives the user much flexibility in how he wants to explore design possibilities, but it also significantly increases interface load.

**Interaction between similar graphic units**

Consider two contours A and B, where A controls B (see Figure 9). Suppose we rotate A, what should happen to B? The answer is not unique. In this case, nothing could happen (in the case when B has to react only to translation), B can be rotated around the centre of A (Figure 9b), the centre of B can be rotated around the centre of A (Figure 9c), or B can be rotated around its own centre (Figure 9d). In the design aid system the user can specify which behaviour is desired by setting attributes of the relations and graphic units (if he does not the
system will choose one of the possibilities). In the user interface, the relations themselves are defined by selecting points of different graphic units and pressing a button, upon which the relation is visualized as an arrow.

**Top:** Interaction between graphic units. *a:* Initial positions of contours. *b:* Rotation around A. Images are screen dumps of the first Structural Sketcher prototype system (A and B added).

**Bottom:** *c:* Rotation with the rotation for contour B constrained. *d:* Rotation with the rotation for relation A-B constrained.

### Interaction between dissimilar graphic units

One graphic unit can have multiple parents. A parent is a graphic unit with a supervising relation to this graphic unit. Such parents influence the behaviour, where the form of influence depends on the type of graphic unit. The previous example concerned two graphic units of the same type. The situation becomes more complex and also more challenging when types are mixed. Let’s take the grid as an example. How does a grid influence other objects? Normally a grid helps to determine dimensions and locations of objects:

- **Determine dimensions:** The lines in a grid are spaced at a particular distance, called the module. This module is a basic unit of measurement. Every element that is placed in the grid, and that adheres to the grid, has dimensions that are a whole multiple of the module. So, if the module is 30 cm, then any dimension of the object on the grid is 0, 30, 60, 90, 120 cm, etc.

- **Determine locations:** The intersections of the lines in the grid determine begin- and endpoints of elements that are placed in the grid. In this way, constant dimensions of the element and coordination of the objects in the grid is ensured.
Other objects can influence a grid. The generic representation *Contour in Grid* (15) consists of graphic units: (2) *Contour* and (16) *Grid*. There are two types of behaviour possible for such a generic representation. Firstly, a contour is bound to a grid. This situation occurs for instance when a designer positions a building in the space defined by a grid. The points and lines of the contour must share the orientation and position of the grid. Secondly, a grid can be bound to a contour. This situation can happen when a designer wants to define a new measuring system to match new objects within a specific contour. The properties of the grid are now derived from the position and orientation defined by the contour.

Consider a grid G1, which influences a contour ABCDE, to which a second grid G2 is attached (Figure 10). The local influence of the grid G2 is visualized by fading the grid off a distance from the parent object. When a user translates the contour ABCDE (one of the points or the whole contour), the system helps the user to position the contour within the grid G1: the positions of points are rounded to the nearest grid point. Also, the position of the grid G2 will be affected. Extra flexibility is offered to define the relation between a grid and other graphic units. The following levels of geometric influence are offered:

- **Line-Line level**: the orientation of the supervised object must coincide with the orientation of the grid.
- **Point-Line level**: points of the supervised object must be located on one of the grid lines.
- **Point-Point level**: points of the supervised object must be located at intersections of grid lines.

In this way, the prototype system maintains not only the shapes of the contours, and coordinates these with the grid, but it also maintains the internal relationships between the grids and the contour. These relations are meaningful to the architect when he is working with grids and contours.
6.3 **Structural Sketcher**

The considerations above have lead to Structural Sketcher, which is based on graphic units. Internally, a design is described as a graph, with graphic units as nodes and relations as edges. Relations can be either uni-directional (one graphic unit supervises another) or bi-directional (two graphic units influence each other). The current prototype however imposes constraints on the graph: between any two nodes in the graph at most one path may exist. Hence, cycles are not allowed. The graph does not have to be connected. Multiple independent sub-graphs can be used.

The graph is used by the system to propagate geometrical transformations from one node to another. All graphic units and relations are shown as graphic objects that can be manipulated and changed by direct manipulation. The user is enabled to use translation, scaling, rotation, and skewing.

The use of a hierarchical structure to model geometric or graphic objects is standard in computer graphics (Foley et al. 1990). However, in standard drawing packages this mechanism is used only in a limited way: the user is allowed to hierarchically group elements. The current prototype system offers more flexibility, in the sense that for each kind of transformation and for each graphic unit and relation a separate decision can be made how transformations are dealt with.

The use of points and relations to model interactions between graphic units is partly based on work done earlier in the context of interactive simulation (van Wijk and van Liere 1997). Again however, more flexibility is offered here, and also the supervision of graphic units such as by means of a grid is novel.

A relation between two graphic units is presented graphically as an arrow between two points, which belong to related graphic units. Multiple connection points can be defined per graphic unit. Such a point defines the origin of the local coordinate system for subsequent transformations of a graphic unit. A geometric transformation of a relation changes the position of these connection points. The graphic unit itself (in this case the shape of a (2) *Contour* or the spacing and direction of a (16) *Grid* is defined by another set of points. These points are defined in a local coordinate system attached to each graphic unit. A geometric transformation of a graphic unit changes its local coordinate system, and hence the position of the points in global coordinates.
The propagation of transformations starts at a point selected by the user. The transformation is propagated through the graph affecting all the graphic units and the relations on its path. This is demonstrated by a simple example for the propagation of the geometrical transformation. The contours A, B, C are sequentially connected by relations A → B and B → C. The connection points are the local origin of the graphic units (see Figure 11).

11

Propagation of the rotation transformation.

When the user applies a rotation of 45 degrees to the geometrical centre of contour A, this transformation is propagated as follows through the graph:

1 The transformation of the relation A → B affects the sub-graph B-C: the local coordinate systems of contour B and contour C are translated to reflect the 45 degrees rotation of O_B around O_A. The contours B and C are translated as a result of this (see Figure 11b).

2 The transformation of the relation B → C affects the local coordinate system of contour C: the position is translated according to a 45 degrees rotation of O_C around O_B. Contour C is translated as a result (see Figure 11c).

3 Finally the rotation around the origin is applied to the local coordinate systems of contour A, B, and C in succession (Figure 11d). The contours change their orientation together with the local coordinate systems.

In this scheme all transformations are done independently and the final transformation of the graphic units is equal to the product of these transformations.

The user can set attributes for each graphic unit and relation to define which transformations in the graph have to be performed or not, such that a large variety of geometric relations is available to the user. For example, if the user in the previous case constrains the processing of the geometrical transformations to relations only (step 1 and 2), then the result is that only the contours are translated (Figure 11c). If only transformations of units are enabled, the result is
that all contours rotate around a local origin. This latter option is not available in other design systems. Also, each combination of these can be used.

The user can set the transformation attributes for each type of transformation individually. A click on the right mouse button on a relation or graphic unit gives a context menu from which all attributes of an object can be changed.

Another important aspect of the processing of geometrical transformation in the graph is the influence of parent objects. If an object that is manipulated by the user has an incoming relation from a grid, the influence of this grid has to be taken into account. The processing of influence takes place before the transformations are propagated through the graph. In the current version of the system, only the parent influence (and not for instance the grandparent influence) is taken into account. The scheme of the user action processing is presented in Figure 12.

1. Transformation calculation: the geometry manipulator produces the transformations given by the user.
2. Transformation correction: these transformations can be corrected with respect to the gravity field (for example, if a grid is active, then objects adhering to the grid will ‘snap’ to grid points).
3. Spanning tree calculation: the spanning tree for the propagation of a transformation is calculated starting from the current anchor point.
4 Anchor points transformation: the transformation is propagated via the spanning tree and subsequently, anchor points are updated.

5 Graphic units transformation: transformations are propagated from anchor points to the graphic units. Graphic units are updated.

6.4 Overview and scope

The first instance of the Structural Sketcher system offered total freedom in the relations between graphic units that are present. From the usage of the system both with researchers and architects, it appeared that so much freedom was not productive since the user had to maintain all meaningful relations. Based on this finding, we looked more carefully at the relationships between graphic units that hold for all generic representations, and implemented these in the last version of Structural Sketcher.

The user interface is critical to an unobtrusive use of the system. It is not that difficult to offer many options and much flexibility, but a major challenge is to define interaction techniques and visual metaphors such that the user can define what he wants in an intuitive way. In the last version of Structural Sketcher, many options that apparently were not used much were removed, and concentrated on the instantiation of graphic units and their manipulation, rather than control of relationships. Most of the direct manipulations of the objects were accommodated in a single device called the “Kite” (explained in detail in Pranovich et al. 2002b and Pranovich 2004:67-79).

What Structural Sketcher clearly demonstrates, is the amount of internal relationships that are called into existence simply by creating graphic units in a drawing. This is similar to some extent to the findings of Gross et al. (1988) and Gross (1990) who were looking at implicit constraint definition in diagrams. Again, we would like to point out that to a trained observer, such as an architect, these relations are obvious from looking at the drawing itself. Nevertheless, it is far from obvious how to actually encode it. With Structural Sketcher, we have an underlying graph structure that can encode, maintain, and propagate changes to related graphic units in a drawing in a manner that conforms to the properties of these graphic units.

Structural Sketcher was tested with ten subjects with an architectural background. A test consisted of a brief introduction about the purpose of the system, a tutorial session in which the subjects performed a few tasks in order to get used to the new software, a design task for a small health care centre located...
at the university campus (see Figure 13), and finally a questionnaire to fill in. Apart from that, the amount of user actions that are required to successfully manipulate objects were compared between Structural Sketcher and AutoCAD, 3DStudio Viz, and Powerpoint.

From the user test, it appeared that Structural Sketcher scored rather well in terms of ease of learning, ease of manipulation, efficiency, and subjective satisfaction. Points of improvement concern appearance of the system, and suitability for the early phase of design. The aspect of appearance of the system can lie in the fact that shapes are defined as polygons rather than the freehand shapes of a hand-drawn sketch. Substituting this definition for a stroke-based approach may improve both aspects.

The comparison test showed that Structural Sketcher substantially reduces the number of required user actions to successfully manipulate objects. Over three tasks, compared to AutoCAD it required respectively 65%, 59%, and 55% less actions, compared to 3DStudio Viz this was respectively 40%, 43%, and 54% less actions, and compared to Powerpoint this was respectively 13%, 16%, and 54% less actions. Apart from anything else, this indicates that working with Structural Sketcher simply is faster than with the compared systems (Pranovich 2004:90-91).

Structural Sketcher, although far from a completed sketch-supporting system, nevertheless provides a first confirmation that graphic units offer an attractive foundation for an interactive design system for the early phases of the architectural design process.
Chapter 7  Paper Plus: Building a Graphic Unit Recogniser

As stated in the previous Chapter, building specialised tools for producing drawings may have the disadvantage that the architect needs to learn new techniques. Because architects are very familiar with graphic representations and skilled in their production, it seems better to build tools that can understand drawings rather than force the architect to learn new tools. In this Chapter, such recognition based on the theory of graphic units and generic representations is presented.

The main functionality of any ‘Paper Plus’ system would be the automated recognition of drawings. The first track in this Chapter outlines recently started work on this (section 7.1). When such functionality is available, a system may offer the architect real-time advise based on the analysis of the drawing. This is presented in section 7.2 – 7.3.

7.1 Multi-agent system approach
Automated recognition of drawings relies on an understanding how drawings are constructed in general, and on the domain in which the drawings are made. Most recognition systems assume that a particular convention of depiction is used (such as plan, isometric projection, perspective, etc.) and that the drawings deal with a particular domain (such as interior architecture, architecture, mechanical engineering, and so forth). Within these assumptions, and usually within a well-defined area of application, such systems operate reasonably well. If the scope of analysis is to include any plan-based drawing however, most systems run into problems because of the limiting character of the basic assumptions. The work of graphic units may provide a comprehensive framework to found analysis of produced drawings.

Specialised agents for graphic units
Architects use a great variety of graphic elements to structure and organise their drawings, which have a clear meaning for them. There is currently no system that can interpret drawings with a graphic “vocabulary” as comprehensive as the identified graphic units and generic representations.
In order to achieve real-time recognition of graphic units in a drawing system, we are looking at the field of multi-agent systems – a recently developed discipline stemming from the area of Distributed Artificial Intelligence. Researchers in this area acknowledge two basic observations: (1) most of intelligent activity can be considered as distributed in one way or the other, and (2) the isolated symbol-processing approach seems to have reached the limits of what can be achieved.

Based on this, the notion of an “agent” as a situated and autonomous entity capable of interacting with the world and other agents has gradually developed (Russel and Norvig 1995, Nilsson 1998, Weiss 2001). Since the capabilities of such a widely defined agent range from the very simple to complex systems (Müller 1998), there is as yet no single definition what an agent is (Wooldridge and Jennings 1995, Wooldridge 2001), nor a theoretical basis for multi-agent systems (Luck 1999).

Although the mainstream of work on multi-agent systems usually conceives agents in human-like terms, a number of researchers have proposed to apply the multi-agent approach to the more cognitive functions of intelligent behaviour (Minsky 1988, Maes 1989, Brooks 1990). Franklin (1995) sums up such directions and identifies agents as a necessary building block for complex and intelligent behaviour.

Research on multi-agent systems focuses not only on the capabilities of the agents themselves, but moreover on reasoning within groups as cooperating or competitive individuals. It is generally found that conceived this way, multi-agent systems can function robustly in highly dynamic and unpredictable environments, removing some of the brittleness of previous AI systems.

Two stage research setup
The track develops in two phases: first the development of a stable and sufficiently fast platform for a multi-agent system; and second the implementation of graphic-unit specialised agents that will interpret a drawing. The work is still fairly new, with the first phase nearly completed. It has not produced yet any substantive results in the area of recognition.

Agents and recognition of graphic units
Important issues to address in graphic unit recognition in drawings concern ambiguities and inaccuracies in the drawing, and resolving conflicting
interpretations between candidate graphic units in the drawing. Multi-agent systems seem appropriate for tackling these issues. To summarize:

- An agent can specialize in recognition of one particular graphic unit, building on other agents that recognize more primitive graphic elements (systems approach).
- Agents may engage in conflict identification and resolution; this is necessary to deal with ambiguity in a drawing.
- Functionality is built piecemeal on top of existing agents, so that the system can be developed incrementally.
- Multi-Agent-systems can function in dynamically changing environments, where resolution is not always possible. Drawing is such an environment.

**Multi-agent framework**

We have established an agent framework for developing a multi-agent system. An agent in the framework has input, output, and an internal state with processes that is closed to the outside world. The input part senses the world environment and receives broadcast messages. The output part manipulates the world environment and broadcasts messages (Figure 14).

Agents operate independently: communication only takes place via broadcastings and the indirect effect of manipulation of the outside world. It is possible to instantiate any number of agents of a given type. The multi-agent system is multithreaded, having all the agents run continuously at the same time. Because in this way it is not possible to predetermine in which order which agents perform their actions, the design of the agents needs to take this into account.
account. Implicit control is established through the use of broadcasts. An agent reads the broadcasts and selects those messages that are relevant. The main motivation for this kind of emergent control lies in the assertion from AI that there is no explicit control structure when cognitive activity is perceived as the result of many agents interacting (Franklin 1995:17-18). The agent’s implementation has been made by Joran Jessurun, and runs basically as follows:

1. Wait for a message.
2. If the message is not interesting, go to 1.
3. Do something with the message.
4. Send messages.
5. Interact with the environment.
6. Go to 1.

An observer agent is implemented as follows:

1. Observe the environment.
2. Broadcast a message about important changes.
3. Wait for a while.
4. Go to 1.

This framework is applied in the first track of the work, by making a multi-agent system that can play Mah Jong (Achten and Jessurun 2003a; 2003b). The second track, extending the multi-agent system to graphic unit recognition, has yet to be started.

Mah Jong is a game consisting of 148 bones that are stacked in a specific pattern. The purpose of the game is to clear the board by taking away two bones that form a match in suit and number, and which can be taken away from the left, right, or both (Figure 15).
Mah Jong is a toy problem in comparison to design. Yet there are some characteristics that make it interesting to look at: (1) although the game is fully determined and finite, the player does not have full information because of the stacking of bones; (2) complications can occur that may be anticipated by studying the current situation of the board. These can impose constraints on possible draws; (3) the aim to take away a particular bone may lead to goal-driven search for other bones to be taken away before a match can be found; and (4) at many points during the game play, the player can choose between various draws of bones in a move. The choice has to be made with incomplete information, and competing arguments.

A multi-agent system for recognition of graphic units needs to balance search and recognition strategies: search for finding whether the graphic unit for which the agent is specialised is being drawn, and recognition for determining whether the graphic unit is actually present. This is similar to the problem of detecting which move to make. Furthermore, negotiation about best moves resembles negotiation between agents to settle into an interpretation of the drawing. In both cases, the environment changes dynamically through game moves of the player and drawing actions of the architect.

Towards graphic unit recognition

Let us consider how a multi-agent system may go about recognising graphic units in a drawing. Suppose we have a system that allows an architect to develop his design with an interface that mainly consists of a drawing area in the same way as pen and paper. Strokes are displayed as they are drawn; there is no beautification procedure that converts the lines. Let us also suppose we have agents that are able to recognise the primitive graphic elements such as lines, circles, and arcs (other types of primitives can be added). The line agent observes the world environment whether it sees lines, the circles agent whether it sees circles, and the arcs agent whether it sees arcs. While the architect is drawing, such shapes are created continuously (or something is created that the system simply cannot interpret). Each time one of the agents recognises the particular object it is specialised for, it broadcasts to the world its particular finding.

The broadcast is picked up by a second group of agents that are looking for more complex shapes: a contour agent is triggered on a broadcast of a line or arc and determines whether the new line or arc is a new shape or will complete a
new closed shape. If so, this shape is added to his list of shapes, and the contour agent broadcasts this message. A regular closed polygon will trigger the Simple contour agent, which looks for the graphic unit (1) Simple contour. An irregular closed polygon will trigger the Contour agent, which in effect searches for the graphic unit (2) Contour. The Grid agent, on the other hand, only likes parallel lines in two orthogonal directions, so it is interested in line segments. When a line segment has been broadcast, it will check whether the new line segment is running parallel to any other lines it already has stored as candidates for a grid, and if the new line segment will extend the set of parallel lines so that it may conclude there is actually a grid. The same reasoning applies for all basic graphic unit-agents (1) Simple contour, (2) Contour, (7) Function symbols, (11) Modular field, (16) Grid, (17) Tartan grid, (23) Proportion system, and (26) Circulation scheme.

Agents that build on the broadcasts of the basic graphic unit-agents determine more complex graphic units in the drawing. They are more complex because they have to compare more than one object to determine whether their specialisation is present in the drawing. The more complex graphic units are: (3) Measurement device, (4) Specified form, (5) Elaborated structural contour, (6) Complementary contours, (8) Zone, (10) Schematic subdivision, (12) Refinement grid, (13) Schematic axial system, (14) Axial system, (18) Structural tartan grid, (19) Element vocabulary, (20) Structural element vocabulary, (24) Combinatorial element vocabulary, (21) Functional space, (22) Partitioning system, and (27) Circulation.

It is very likely that some set of drawing elements in the course of a design process will be recognised at the same time by various graphic unit-agents. In order to resolve this ambiguity, some argumentative structure or scoring system is necessary that allows the agents to determine among themselves which interpretation scores the highest. Currently, it seems that creation-order of the primitive graphic entities, measure of completeness towards a graphic unit, and degree of specificity towards a graphic unit are important candidates for establishing a conclusion. Alternative interpretations that are made by the system are not discarded but simply ranked lower as candidate interpretations, since during the design process it may well appear that an interpretation has to be changed in favour for another.

The Mah Jong systems show how to coordinate decision-making between agents. Also, they demonstrate how the principles of communication by
broadcasting in the multi-agent system apply. Thirdly, they provide working experience with combined search- and recognition techniques for heuristics that function in a dynamic setting. Finally, some insight is gained in the effects of dynamically rank-ordering heuristics as a simple learning mechanism.

There are two main features that are lacking with respect to a system that can recognize graphic units: (1) there is no sophisticated learning mechanism; and (2) there is no user in the loop. Although the systems can optimise their internal rank-ordering of heuristics, they are not capable to establish new heuristics or modify existing heuristics. This may become an important issue when the system needs to tune in on the architects’ drawing style. The user in the loop can increase complexity of the environment of the multi-agent system, e.g., by erasing previously drawn objects, changing his mind about what he is drawing, starting a new drawing, or switching to a non plan-based representation.

7.2 Case-based reasoning approach

When a multi-agent system can recognise graphic units in a drawing, then there is still the question how this knowledge can be used to provide the architect with relevant information. Earlier, in Chapter 4 the relation between knowledge and generic representations was studied. If we conceive generic representations as a repository of drawings, then the question becomes how it is possible to access this repository to utilise knowledge related to the drawing. For this purpose, we investigate in this section the use of Case-Based Reasoning techniques.

Case-Based Reasoning (CBR) and Case-Based Design (CBD) have been proposed to utilize knowledge of previous design solutions to understand or solve current design problems. The main motivation for CBR-systems lies in the recognition that when solving a new problem, people often adapt an old solution for the current problem. CBR-systems are applied in AI in a variety of domains, such as law, cooking, and architecture (Kolodner 1993). Similarly, CBD-systems apply this principle in design (Maher and Pu 1997).

Routine design as a category is reasonably well covered both in research and existing CAAD applications. Innovative and creative design however, still face significant problems in the area of design support. In CBD a previous design episode or case that resembles the current design problem is retrieved from a case base and (slightly) adapted to fit the problem. In this way, CBD resembles the cognitive approach of designers and problem solvers who use previous design solutions when generating new solutions. This is in fact the basic
cognitive model underlying both Case-Based Reasoning and Case-Based Design.

A main assumption is that in case adaptation previous cases that resemble the current problem also have resembling solution principles that are relevant to the problem. The relatively complete solution then needs adaptation to fit the current task. Keeping the solution principle intact is a necessary but not trivial task. Issues in CBD therefore, are: problem definition, assembling an indexed description, matching the indexed description to the case base, retrieving matching cases, case adaptation, evaluation of the adapted case in the design problem, and storing the new case in the case base.

CBD claims not to be as knowledge-intensive as most production systems in AI because it relies on the structure present in a whole solution and presupposes that the relations that made up the good solution in the past can be kept intact for the new design problem. Nevertheless, case adaptation remains a notorious problem, and most systems halt at finding and presenting a relevant case. CBD is well established and has seen much research in the past years under varying assumptions. Examples are: Kolodner 1993, Lee and Lee 1995, Oxman 1995, Tan et al. 1995, Maher and Pu 1997, Chiu and Shih 1997, Heylighen 1998, de Groot et al. 1999, Grassi et al. 1999, and Mallory-Hill 2004. It has yielded a number of applications such as: CADRE, SEED, FABEL, ARCHIE and ARCHIE-2, IDIOM, PRECEDENT, EDAT, and DYNAMO.

Cases are retrieved through an indexing system. The indexing determines the features via which a case can be found. Index systems typically are text based. This means that finding the case requires verbalizing the design problem, which stands in contrast with the graphic way of working through drawings. When there is no explicit vocabulary for describing drawings, it is hard to verbalize thoughts that are formed through drawings. Furthermore, as was pointed out earlier, reasoning in design is often done on the basis of sketches (Goel 1995, Verstijnen 1997, McFadzean 1999), in particular in the formative early stages. Architects often associate during design via the images at hand through recollection of images of other projects.

A substantial improvement can be gained when the design drawings made during the design process can function as a (partial) query for a case-base. In order to do so, it is necessary to have techniques for image recognition, structuring cases in image-like fashion, retrieving cases from the case-base, and ways to incorporate the stored knowledge in the design. Graphic units can
provide a handle on these aspects. Three techniques are outlined: a decision-tree for graphic unit recognition, a query composition algorithm, and a look-up table for case retrieval. These three techniques are worked out in the context of a Case-Based Design Aid System (CBDAS).

Such a Case-Based Design Aid System provides the designer during the design process cases that bear relevance to the current design issues at hand. The work on the Fabel project (Coulon 1995; Schaaf and Voss 1995) and more recently SpaceScope (Hwang and Choi 2003) is most related to our approach here. It is important to notice that although we work within the terminology of CBD, it is not proper Case-Based Reasoning what we are doing here. Rather, we utilise some techniques and demonstrate how these can benefit from the use of graphic units.

7.3 A hypothetical Case-Based Design Aid System

Further elaboration of the theoretical work in this paper is done in the context of a hypothetical Case-Based Design Aid System (CBDAS). The hypothetical system has the following components (Achten 2000):

1 Interface for generating design drawings.
2 Establishing graphic units in the design drawing.
3 Composing a query for the index of generic representations.
4 Retrieving the appropriate set of generic representations.
5 The database of generic representations.
6 Presenting the retrieved set of generic representations and cases.

This outline provides a framework of reference for the following parts.

**Interface for generating design drawings**

The interface for generating design drawings may be a tailored system specifically for the CBDAS, or use an existing component or software on which the CBDAS builds. The interface is important in the sense of Human-Computer Interface aspects, but less important in the current discussion of a system that finds related cases on the basis of a drawing. The techniques for achieving recognition are outlined in the next section.
Establishing graphic units in the design drawing

Earlier, two different techniques were explored for utilising graphic units in automated drawing. In this Chapter, a third approach for establishing which graphic units are being drawn in the design is identified and outlined.

• Graphic units as drawing tools: Structural Sketcher, outlined in Chapter 6, shows graphic units as drawing tools: simply select a tool for (16) Grid, and you automatically create grids in the system. It shows how dependencies between graphic units can be defined implicitly and changed on the fly during the design process. In this approach, establishing graphic units that are made in the drawing come “free” with the application because they are simply identical with the tools that are used.

• Graphic unit recognition agents: the second approach, outlined earlier in section 7.1, discusses graphic units as a multi-agent system in which each agent is specialised in one kind of graphic unit, and tries to recognise this graphic unit while the designer is drawing. Settlement over ambiguities has to be resolved through negotiation between the agents in the system. In this approach, establishing graphic units that are made in the drawing is a computationally intensive process that nevertheless imposes the least amount of constraints on the designer and allows him to sketch freely without consideration of specialised tools that have to be used.

• Designer-assisted approach: relative to the two techniques discussed above, there is a third option for establishing graphic units in drawings, and that is simply by asking the designer. The designer-assisted approach can be either straightforward (ask the designer to identify graphic units), or assist the designer in identifying. The first approach assumes the designer to be knowledgeable about the theoretical work underlying the system, which usually is not the case. Such classification is prone to mistakes and distracts from the design work. The second approach guides the designer in a limited number of steps through a decision tree to determine which graphic unit is worked with at the moment.

Ultimately, automated graphic unit recognition needs to be implemented in a CBDAS. The designer-assisted decision-tree approach is useful because before we can automate anything, the various ambiguities and decision points in identification of graphic units need to be clearly defined. The decision-tree forms a starting point for this.
**A decision tree based on graphic units**

A decision tree is the representation of a classification mechanism for a phenomenon that differentiates attribute-value pairs into two groups (branches on a tree-structure), until a particular inference can be made (leaf of the tree) about that particular phenomenon (Mitchell 1997:52-53). In this section, a decision tree is presented to provide a question-answer mechanism that leads to identification of a graphic unit in a drawing. The nodes are not bifurcal in order to decrease the number of questions that need to be answered. The process starts when the designer prompts the system for case retrieval. Each pass through the tree identifies one graphic unit (Figure 16: next page).

Each node in the decision-tree either is a question or an identified graphic unit. The questions are identified with capital letters A, B, C, etc. The graphic units are identified with their reference numbers. At each decision point, a schematic drawing clarifies the decision that has to be taken. Textual clarification is also available, but for brevity’s sake this is not included. The next page shows the list of questions in the decision-tree.

Note that the questions first aim to isolate groups of graphic representations that have a meaningful resemblance (shapes, sets of lines, spaces, building elements, etc.), and then differentiate on the basis of their interpretation (grid, axial system, schematic subdivision, etc.) to derive the specific graphic unit. This method is not yet specific enough for automated graphic unit recognition such as would be required for a multi-agent system in section 7.1, but it already offers some structuring of a drawing analysis. As a question-answer mechanism for a designer it is adequate.

The decision-tree leads in a maximum of five questions to a graphic unit. This process needs to be reiterated until all the elements of the drawing have been identified, although a user may decide to have only part of the drawing interpreted.

**Composing a query**

The use of the decision tree results in a list \( L_{gu} \) of graphic units that have been identified in the graphic representation:

\[
L_{gu} \{gu_1, gu_2, \ldots, gu_n\}, n \in \mathbb{N}.
\]
A. Is it a graphic or symbolic element?
• If graphic element, go to B.
• If symbolic element, graphic unit is (7) Function symbols.
B. Is it a closed shape or a set of one or more lines?
• If a closed shape, go to C.
• If a set of one or more lines, go to D.
C. Is it a single shape or multiple shapes?
• If multiple shapes, go to E.
• If single shape, go to F.
D. Is it a coordinating system or not?
• If coordinating system, go to G.
• If not a coordinating system, go to H.
E. Is it spaces or building elements?
• If spaces, go to I.
• If building elements, go to J.
F. Is it a Platonic shape, a more irregular shape, or a shape that is specified?
• If Platonic shape, graphic unit is (1) Simple contour.
• If irregular shape, graphic unit is (2) Contour.
• If specified shape, graphic unit is (4) Specified form.
G. Is it a zone, grid, or proportion system?
• If zone, graphic unit is (8) Zone.
• If grid, go to K.
• If proportion system, graphic unit is (23) Proportion system.
H. Is it for measurement, subdivision, composition, or circulation?
• If measurement, graphic unit is (3) Measurement device.
• If subdivision, go to L.
• If composition, go to M.
• If circulation, go to N.
I. Is the contour elaborated and showing the structure, or are the contours complementary, or do the spaces have functional indications?
• If contour is elaborated, graphic unit is (5) Elaborated structural contour.
• If the contours are complementary, graphic unit is (6) Complementary contours.
• If the spaces have functional indications, graphic unit is (21) Functional space.
J. Is it a set of elements, structural elements, or rules for elements?
• If set of elements, graphic unit is (19) Element vocabulary.
• If structural elements, graphic unit is (20) Structural element vocabulary.
• If rules for elements, graphic unit is (24) Combinatorial element vocabulary.
K. Is it a modular field, grid, refinement grid, tartan grid, or structural tartan grid?
• If modular field, graphic unit is (11) Modular field.
• If grid, graphic unit is (16) Grid.
• If refinement grid, graphic unit is (12) Refinement grid.
• If tartan grid, graphic unit is (17) Tartan grid.
• If structural tartan grid, graphic unit is (18) Structural tartan grid.
L. Is it a schematic subdivision or a concrete partitioning?
• If schematic, graphic unit is (10) Schematic subdivision.
• If concrete, graphic unit is (22) Partitioning system.
M. Is it a schematic axial system, or a concrete axial system?
• If schematic, graphic unit is (13) Schematic axial system.
• If concrete, graphic unit is (15) Axial system.
N. Is it a schematic circulation scheme, or a concrete circulation system?
• If schematic, graphic unit is (26) Circulation scheme.
• If concrete, graphic unit is (27) Circulation.
This list is the basis for composing the query that will search the case base for matching generic representations. The matching is a simple look-up whether the generic representation exists, and then retrieval of the associated graphic representations follows. However, drawings evolve throughout the design process, and this simple matching procedure could miss cases that are related but do not belong to the same generic representation. By means of successive graphic units it is possible to identify drawings that are closely related to the current graphic representation. Also, drawings that are less complex may be relevant. The query for case matching therefore needs to be expanded.

**Query expansion with weighted successive graphic units**

As discussed in section 4.2, graphic units can have successive graphic units. In a sequence of successive graphic units, a graphic unit resembles to some extent its
predecessor or successor, as it becomes defined to a more or lesser degree. The relevance of this aspect for the CB DAS is the following. Consider \( L_{gu} \{gu_1, gu_2, \ldots, gu_n\} \) and \( M_{gu} \{gu_1, gu_x, \ldots, gu_n\} \) where \( gu_x \) is a successive graphic unit of \( gu_2 \). Then \( M_{gu} \) differs in one graphic unit from \( L_{gu} \). The different graphic unit \( gu_x \) is a more or less defined variant of \( gu_2 \). For example, \( gu_2 \) could be (13) *Schematic axial system*, and \( gu_x \) could be (15) *Axial system*. A case with an axial system can be informative for the designer, even when he is considering an axial system only in a schematic form. \( M_{gu} \) therefore, bears much resemblance with \( L_{gu} \), and should be included in the search for matching generic representations.

It is also possible to consider the cases where two elements of \( M_{gu} \) differ as successive graphic units with respect to \( L_{gu} \), where there are three successive graphic units, etc. Subsets of \( L_{gu} \) (where \( M_{gu} \) has a smaller number of graphic units) can be considered for matching as well. Since subsets resolve a lesser number of graphic units, they have reduced complexity with respect to the graphic representation under consideration. The relevance of subsets therefore, as matches in the query, will be less. In order to take note of the differences between matches, sets of \( M_{gu} \) need to be weighted against \( L_{gu} \). For this purpose, a list of weighted successive graphic units was established (Figure 17).

The weights have been established in the following manner. A modular field for example, does not differ much from a grid. Both consist of a system of lines that delineate where elements can be placed. A refinement grid however, adds an additional grid, thus increasing the complexity. Therefore, a succession from (11) *Modular field*, to (16) *Grid*, makes a lesser difference (weight 1) than a succession from (16) *Grid*, to (12) *Refinement grid* (weight 2).

![Weighted successive graphic units. Nodes are graphic units; weights are set on the vertices. The vertices are directed to indicate more elaborated graphic units.](image)
A subset of $L_{gu}$ is considered to deviate more than a set of $M_{gu}$ where two graphic units have been substituted by a successive graphic unit. The maximum weight in the case of two successive graphic units is 4 $(2+2)$. Therefore, a weight factor of 5 is set for each graphic unit of $L_{gu}$ that is omitted.

**Finding and rank-ordering all queries**

All possible combinations of $L_{gu}$ with successive graphic units that will form query $Q$, can be generated by the following method:

1. For each $gu_j$ in $L_{gu}$, substitute with its predecessor, successor, or itself. A predecessor $s_j$ is a graphic unit in the table of successive graphic units that precedes $gu_j$. A successor $s_j$ is a graphic unit in the table of successive graphic units that succeeds $gu_j$. Each single substitution in $L_{gu}$ forms a new list of graphic units $(s_1, s_2, ..., s_n)$. If $gu_j$ has been substituted by a predecessor or successor, then add the weight stated in the table of weighted successive graphic units to the current weight (initial value is 0). The total weight $W$ determines the rank-order of the new combination. Add the tuple $\{(s_1, s_2, ..., s_n), W\}$ to $Q$ (initial set is empty).

2. For each subset of $L_{gu}$ repeat step 1.

3. Rank-order the found list $Q$ by increasing weight. The result is stored in $Q$, which will be used for case retrieval.

The most computation intensive part of the whole process is the graphic unit recognition. Once a set of graphic units is identified, composing the query $Q$ is very simple. Therefore, multiple interpretations that may be generated in the manner described in section 7.1 can be included in the query as well.

**Example of a query**

Suppose in a graphic representation (Figure 18: next page) the following graphic units have been identified (possibly obtained through a multi-agent system as in section 7.1, by means of the decision tree above, by Structural Sketcher, or any other way): (10) *Schematic subdivision*, (13) *Schematic axial system*, and (6) *Complementary contours*. $L_{gu}$ then is: $\{10,13,6\}$ (the order of identified graphic units is irrelevant for the outcome).

Using the list of weighted successive graphic units (see Figure 17), we find that graphic unit (10) *Schematic subdivision*, has no predecessor, and one successor: (22) *Partitioning system*, with weight 1. Graphic unit (13) *Schematic axial system* has no predecessor, and one successor: (15) *Axial system*, with
weight 1. Graphic unit (6) *Complementary contours*, has one predecessor: (2) *Contour*, with weight 2, and no successor. Together with the subtraction of one or two elements from $L_{gu}$, this is everything necessary to generate $Q$.

$Q$ then, rank-ordered on increasing weights, is:

$$Q = \{\{(10,13,6),0\}, \{(22,13,6),1\}, \{(10,15,6),1\}, \{(10,13,2),2\}, \{(22,15,6),2\}, \{(10,15,2),3\}, \{(22,13,2),3\}, \{(22,15,2),4\}, \{(10,13),5\}, \{(10,6),5\}, \{(13,6),5\}, \{(10,15),6\}, \{(22,13),6\}, \{(22,6),6\}, \{(15,6),6\}, \{(22,15),7\}, \{(10,2),7\}, \{(13,2),7\}, \{(22,2),8\}, \{(15,2),8\}, \{(10),10\}, \{(13),10\}, \{(6),10\}, \{(22),11\}, \{(15),11\}, \{(2),12\}\}$$

The rank-order establishes the degree of potential match in query $Q$. Weight 0 indicates a complete match. Queries with weight 1 and 2 are partial matches that deviate because one graphic unit is a successive graphic unit. Higher weights indicate more differences with the graphic representation at hand (multiple successive graphic units or a lower number of (successive) graphic units.

Retrieving a case

The approach of generic representations has the major advantage that aspects on which the cases are stored (graphic units) are either present in the graphic representation or not. There is no problem with range of values, limit values, or fuzziness on which matching has to occur. The questions of similarity and process of design have been addressed via successive graphic units and subsets in the query.

Generic representations are listed as their set of graphic units (Table J). Query $Q$ matches its lists against Table J to retrieve all cases. Since $Q$ already is rank-ordered, the match also is rank-ordered in relevance.
Table J: Matching table for graphic units and generic representations.

<table>
<thead>
<tr>
<th>Left: generic representation (number)</th>
<th>Right: constituent graphic units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple Contour (1)</td>
<td>1</td>
</tr>
<tr>
<td>Combination of Contours (2)</td>
<td>1</td>
</tr>
<tr>
<td>Complementary Contours (3)</td>
<td>6</td>
</tr>
<tr>
<td>Modular Field (4)</td>
<td>11</td>
</tr>
<tr>
<td>Proportion System (5)</td>
<td>23</td>
</tr>
<tr>
<td>Multiple Grids (6)</td>
<td>16</td>
</tr>
<tr>
<td>Functional Spaces (7)</td>
<td>21</td>
</tr>
<tr>
<td>Schematic Axial System (8)</td>
<td>13</td>
</tr>
<tr>
<td>Schematic Subdivision (9)</td>
<td>10</td>
</tr>
<tr>
<td>Elaborated Structural Contour (10)</td>
<td>5</td>
</tr>
<tr>
<td>Element Vocabulary (11)</td>
<td>19</td>
</tr>
<tr>
<td>Combinatorial Element Vocabulary (12)</td>
<td>24</td>
</tr>
<tr>
<td>Circulation Scheme (13)</td>
<td>26</td>
</tr>
<tr>
<td>Proportion System in Contour (14)</td>
<td>2 23</td>
</tr>
<tr>
<td>Contour in Grid (15)</td>
<td>2 16</td>
</tr>
<tr>
<td>Zone in Specified Form (16)</td>
<td>4 8</td>
</tr>
<tr>
<td>Function Symbols in Combination of Contours (17)</td>
<td>2 7</td>
</tr>
<tr>
<td>Axial System in Specified Form (18)</td>
<td>4 15</td>
</tr>
<tr>
<td>Schematic Subdivision in Grid (19)</td>
<td>10 16</td>
</tr>
<tr>
<td>Schematic Subdivision with Function Symbols (20)</td>
<td>7 10</td>
</tr>
<tr>
<td>Schematic Subdivision in Contour (21)</td>
<td>2 10</td>
</tr>
<tr>
<td>Partitioning System in Contour (22)</td>
<td>2 22</td>
</tr>
<tr>
<td>Specified Elaborated Structural Contour (23)</td>
<td>3 5</td>
</tr>
<tr>
<td>Elaborated Structural Contour in Grid (24)</td>
<td>5 16</td>
</tr>
<tr>
<td>Elaborated Structural Contour in Complementary Contours (25)</td>
<td>5 6</td>
</tr>
<tr>
<td>Elaborated Structural Contour and Axial System (26)</td>
<td>5 15</td>
</tr>
<tr>
<td>Elaborated Structural Contour and Function Symbols (27)</td>
<td>5 7</td>
</tr>
<tr>
<td>Element Vocabulary in Grid (28)</td>
<td>16 19</td>
</tr>
<tr>
<td>Element Vocabulary in Multiple Grids (29)</td>
<td>16 19</td>
</tr>
<tr>
<td>Combinatorial Element Vocabulary in Grid (30)</td>
<td>16 24</td>
</tr>
</tbody>
</table>
Table J: Matching table for graphic units and generic representations – continued.

<table>
<thead>
<tr>
<th>Combinatorial Element Vocabulary in Specified Form (31)</th>
<th>3</th>
<th>4</th>
<th>24</th>
</tr>
</thead>
<tbody>
<tr>
<td>Circulation in Contour (32)</td>
<td>2</td>
<td>7</td>
<td>27</td>
</tr>
<tr>
<td>Circulation Scheme in Elaborated Structural Contour (33)</td>
<td>5</td>
<td>26</td>
<td></td>
</tr>
<tr>
<td>Structural Element Vocabulary in Structural Tartan Grid (34)</td>
<td>18</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>Proportion System in Elaborated Structural Contour in Tartan Grid (35)</td>
<td>5</td>
<td>17</td>
<td>23</td>
</tr>
<tr>
<td>Zone in Contour in Grid (36)</td>
<td>2</td>
<td>8</td>
<td>16</td>
</tr>
<tr>
<td>Axial System in Contour in Grid (37)</td>
<td>2</td>
<td>15</td>
<td>16</td>
</tr>
<tr>
<td>Axial System in Contour in Tartan Grid (38)</td>
<td>2</td>
<td>15</td>
<td>17</td>
</tr>
<tr>
<td>Axial System in Specified Form in Structural Tartan Grid (39)</td>
<td>4</td>
<td>15</td>
<td>18</td>
</tr>
<tr>
<td>Schematic Subdivision in Grid and Refinement Grid (40)</td>
<td>10</td>
<td>12</td>
<td>16</td>
</tr>
<tr>
<td>Schematic Subdivision and Schematic Axial System in Contour (41)</td>
<td>2</td>
<td>10</td>
<td>13</td>
</tr>
<tr>
<td>Elaborated Structural Contour and Function Symbols and Axial System (42)</td>
<td>5</td>
<td>7</td>
<td>15</td>
</tr>
<tr>
<td>Element Vocabulary in Zone and Contour (43)</td>
<td>2</td>
<td>8</td>
<td>19</td>
</tr>
<tr>
<td>Circulation in Contour in Grid (44)</td>
<td>2</td>
<td>16</td>
<td>27</td>
</tr>
<tr>
<td>Structural Element Vocabulary in Contour in Modular Field (45)</td>
<td>2</td>
<td>11</td>
<td>20</td>
</tr>
<tr>
<td>Structural Element Vocabulary in Structural Tartan Grid and Refinement Grid (46)</td>
<td>4</td>
<td>12</td>
<td>18</td>
</tr>
<tr>
<td>Structural Element Vocabulary in Axial System in Contour (47)</td>
<td>2</td>
<td>15</td>
<td>20</td>
</tr>
<tr>
<td>Element Vocabulary and Function Symbols and Grid in Specified Form (48)</td>
<td>4</td>
<td>7</td>
<td>16</td>
</tr>
<tr>
<td>Schematic Subdivision in Zone in Contour with Function Symbols (50)</td>
<td>2</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>Structural Element Vocabulary in Axial System in Contour in Grid (50)</td>
<td>2</td>
<td>15</td>
<td>16</td>
</tr>
</tbody>
</table>

**Example of retrieval result**

In the example where \( L_{gu} \) is \{10,13,6\}, the result of the retrieval is as follows:

1. Schematic subdivision and schematic axial system in contour (41), weight 2, graphic units (2,10,13).
2. Schematic subdivision in contour (21), weight 7, graphic units (2, 10).
3. Partitioning system in contour (22), weight 8, graphic units (2,22).
Schematic subdivision (9), weight 10, graphic units (10).

Schematic axial system (8), weight 10, graphic units (13).

Complementary contours (3), weight 10, graphic units (6).

There is no complete match in the resulting match. The first hit deviates with respect to the graphic unit (6) Complementary contours which has become (2) Contour in the generic representation. The case base (survey presented in Chapter 3) yields the following graphic representations belonging to the first two generic representations found in the match (Figure 19 and Figure 20):

---

**Case retrieval of graphic representations belonging to Schematic Subdivision and Schematic Axial system in Contour (41). Redrawn from Durand (1804), Parties, Planche 4 (left) and Planche 5 (right). The ‘+’ and ‘-’ are simplified notations for axes.**

---

**Case retrieval of graphic representations belonging to Schematic Subdivision in Contour (21). Redrawn from Tzonis (1986:31) (left) and Wittkower (1973:73) (right). The partial match omits the schematic axial system.**

---

The graphic representations that are found do not resemble the drawing that forms the basis of the query, but this is to be expected since generic representations are about the kind of drawing and not the particular shape. They do have a resemblance however on the level of design decisions that are taken. Nevertheless, the partial match does not retain much relevance to the current drawing. The level of shared design decisions can be formulated at most in terms of the graphic units that are shared. It seems that only when the current design task and the retrieved cases are in the same type domain (office building, hospital, municipality, etc.), and there is a perfect match in terms of graphic units, it is possible to attribute more specific forms of knowledge. The two examples from Figure 19 show a schematic subdivision of a contour, and a schematic axial system (‘+’ and ‘-’ signs). They differ from the drawing in not
having complementary contours. Still, they may prompt the architect to consider
the ways in which a schematic subdivision may be done on a shape, or how to
arrange his system of axes.

**Database of generic representations**

Each generic representation that is found in the retrieval procedure is a category
of a group of graphic representations that share common design decisions. The
generic representation forms the index through which all particular graphic
representations belonging to it are retrieved and presented to the user.

The cases that belong to the generic representation depict different designs.
Since they belong to the same generic representation, they share to some extent
the design decisions that the designer is working on in his particular design. This
information is stored with the generic representation.

The case base stores the particular graphic representations separately from
the generic representation, with an index to the generic representation. The
information on the graphic representations can be in any format, as at the
moment case adaptation is not an option.

In the example, there is no complete match for \( L_{gu\{10,13,6\}} \). Since the
constitutive graphic units are known, and the case base is built on graphic units,
the particular graphic representation of the example can now be added to the
case base, thus making it available for the next time.

**Presentation of retrieved set**

After zero, one, or more generic representations have been found to match the
set of graphic units in the design drawing they are presented to the designer. The
presentation shows the list of all found generic representations, as well as the
graphic representations that belong to a selected generic representation. Each
case is discussed in terms of the design decisions it addresses. This can then aid
the designer in the work at hand.

### 7.4 Overview and scope

The techniques outlined in this Chapter show how graphic units may be
recognised while the designer is drawing. The approach utilising multi-agent
systems should form a basic module for this purpose, on top of which other
applications can be developed, such as the CBDAS outlined afterwards. The
latter system can provide an architect with relevant design-content information
during the design process, based on a graphic indexing system that also takes into account some of the dynamics involved in design.

The procedure used here will identify all generic representations that have at least one graphic unit or successive graphic unit in common with the graphic representation. Typically, the first hits in the rank-order will have most relevance with the design situation at hand. The whole list of retrieved generic representations is showed to enable the designer to browse through cases that are not directly relevant, but that bear at least some resemblance to the project at hand. In this way, the CBDAS can become a context sensitive browsing tool. Furthermore, if alternative interpretations are present, for example from the multi-agent system component, then these alternative interpretations can be included in the retrieval of cases.

The results of this work can be generalized to other designing disciplines that have well-established graphic conventions in the design process. The fields of mechanical engineering and industrial design seem to share these features with architectural design. The main task in expanding the work to other disciplines first lies in identifying graphic units and generic representations in these fields.

The current system-outline supports assisted graphic unit recognition, it assembles a rank-ordered list of relevant generic representations, and it can address and retrieve cases of graphic representations that are related to the design problem at hand. A fully completed CBDAS has the following additional functionality: automated graphic unit recognition, a structured information model encoding the cases, some support of case manipulation and adaptation, and storage of new cases. Furthermore, such a system should be able to expand on its theoretical foundation: add new graphic units and generic representations.

The cases stored in the system need not be completely adaptable for use in the current design. Since they derive from completely different designs, it is not likely that they will have exactly the right elements and objects to make them easily adaptable to other solutions. Some manipulation must be possible however on the level of graphic units: for example, selecting the schematic axial system in a drawing, copying and pasting it in the current design, and then manipulating it for its new purpose. It is necessary therefore, to describe the cases in a formalism that allows such manipulation of parts (the graph representation of Structural Sketcher in Chapter 6 and the Feature-Based modelling approach in Chapter 5 point to possible directions).
Chapter 8  Summary

In this book an analysis of well-structured architectural graphic representations is presented. This investigation has been founded on a theoretical and historical analysis of graphic representations (Part One). The findings form a framework to think about the knowledge content role of graphic representations in architectural design (Part Two), and improved design support through the use of Computer Aided Architectural Design (Part Three).

8.1 Graphic units and generic representations

The analysis showed that it is possible to identify pervasive, re-occurring structures that have a general accepted meaning in the architectural design community. These structures are termed graphic units. From the number of 24 graphic units it appears that architects have at their disposal a well-developed set of graphic tools to develop the design. Half of the graphic units that have been identified deal with the structuring and organisation of the design, and the other half with describing the design itself.

The most significant contribution of Part One lies in the identification and definition of graphic units. I hope to have demonstrated throughout the book the important role that understanding drawings in terms of graphic units has for understanding the conventions of encoding and conventions of depicting by architects. In part One, analysing the existence of hypothetical generic representations assesses the scope of the current work. For a more thorough understanding of graphic representations a more substantial body of analysis needs to be established. This will lead to a greater number of repetitive instances of current generic representations and new generic representations, but not likely to a much larger number of graphic units. Such a survey will also provide a sizeable case-base of example drawings that a system may use to support the architect.

An important question is to which extent we can consistently and accurately identify graphic units in hand drawn sketches. I have stayed away from this avenue of research because it was first necessary to establish at all what graphic units and generic representations are. Now that we can move towards design support based on the notion of graphic units this question will become fundamental in order to assess relevance of the current work. The most
promising technology to address this question in my view is the use of multi-
agent systems. Based on the work so far, I firmly believe that those parts of
sketches that conform to the same conventions of depiction and encoding as
graphic units, can be identified in a hand drawn sketch.

8.2 Theoretical utility
Graphic representations record the development of the design process. By means
of the described design decisions associated with generic representations, it is
possible to identify procedural and declarative knowledge of a design task. This
principle has been tested on the basis of the office building type. Sequences of
generic representations define procedural knowledge, and each generic
representation defines declarative knowledge. This finding supports the notion
that graphic representations, and the way they develop sequentially, “embody”
design knowledge of the architect. Graphic units are not the ultimate answer to
implicit knowledge encoding. We have seen that they fall short when it comes to
capturing the design rationale, and when the architect uses non plan-based
representations. In order to provide relevant knowledge during a design task, we
also noticed that it is necessary that generic representations need to match very
close (perfect match is preferable) and come from the same task domain.

Generic representations have been described in a more formal way by means
of Feature-based modelling. This can lead to a low-level understanding of the
changes in the design process. What is new in this approach, compared to other
work, is the emphasis on sequences of generic representations, in particular by
means of successive graphic units.

8.3 Practical utility
It is possible to improve design support by means of Computer Aided
Architectural Design applications if these applications know better how to deal
with graphic representations. Looking at current sketch-based systems, it
appears that in terms of graphic units they often implement a limited set of
graphic units. This may be the reason why most sketch-based approaches have
difficulty of developing further because they cannot extend beyond their basic
assumptions of used graphic units (apart from the second obvious candidate
reason; the lack of proper inference mechanisms that can deal with the implicit
knowledge embedded in graphic representations).
In this book two strategies are presented for application of graphic units: through specialised tools that are based on graphic units, and by the recognition of graphic units while the drawing is being produced. The first approach, termed Pen Plus, yields the fastest results, and provides the architect with a set of drawing tools that help structure the design process. This approach is informative about the practical utility of graphic units for architects, but it is more limited from a scientific point of view, since it tells nothing of the principal correctness of the approach. The second approach, Paper Plus, poses major challenges in the area of real-time recognition. Multi-agent systems provide a distributed approach by specialised agents for each graphic unit. Through matching of present graphic units a simple lookup-technique for finding related graphic representations during the design process can be established. Both approaches can be used to build a Paper Plus system.

The implementation of the Paper Plus approach is informative about the consistency of the theory of graphic units and generic representations because it requires a complete specification by which these concepts are to be discriminated in a graphic representation. Both Paper Plus and Pen Plus approaches will tell us to which extent the work is productive because implementations can be applied and tested with architects in practice. On the basis of the results produced so far, I think it is fair to say that this is a worthwhile pursuit.

The notion of graphic units is particularly productive in the early stage of design when organisation of the design can be achieved very well through graphical means. Combined with other conventions of depiction such as sections, façades, and perspectives, graphic representations provide a powerful ‘toolkit’ for the architect to develop the design. Later on, non-graphical representations become increasingly apt for tasks such as simulation and calculation of the structural system, lighting and HVAC aspects, and so forth.

### 8.4 Future work

Much of the current work is now moving from theoretical considerations to practical applied work. I have been very fortunate to see the development of a Pen Plus system in the form of the Structural Sketcher by Slava Pranovich. This system demonstrates the added value that graphic units can bring to CAAD. Real-time recognition of graphic units in hand-drawn sketches remains the biggest challenge up to now, and it has the highest priority. The very general
question “can a CAAD system interpret a hand-drawn sketch” can now be reformulated into several smaller (but still by no means easy) questions: (a) can we recognise graphic units in a hand-drawn sketch – which translates to (a1) can we identify graphic primitives in a hand-drawn sketch; (a2) can we identify graphic units on the basis of these graphic primitives; and (a3) can we settle in a reasonable short amount of time any ambiguities about deciding which graphic units are in the drawing, and following on this, question (b) can we provide relevant knowledge to the architect based on the graphic units that we have identified?

Graphic units provide a means to understand the well-structured conventions that architects apply in their drawings. This is already an important step towards full-fledged sketch recognition, but it surely is not sufficient to reach that goal. There is a lot of additional activity going on in sketching that does not conform to shared conventions, but has more to do with doodling, thought processing, hand-eye coordination, aesthetic pleasure, personal style and expression, and simply the joy of drawing. Sketch-recognition work will on the one hand have to build substantially on findings from cognitive research on the role of sketches and how sketches are constructed, and on the other hand a deep appreciation of the subtleties involved in sketching if we are going to stand a chance in this area. This is the future path in which to further develop this work.

To be continued...
Acknowlegements

The work presented in this document summarises my research from the period 1993-2003 on well-structured graphic representations in architectural design. The work has been performed at the Eindhoven University of Technology, in the Department of Architecture, Building, and Planning. It started in the Design Methods Group, and continued in the Design Systems Group. Without the collaboration with these groups, much of the current work would not have been possible or more limited in scope and extent.

First of all, my PhD-work on graphic units and generic representations has been guided by Prof.dr.ir. Thijs Bax and Prof.dr. Robert Oxman. Without their enthusiastic and critical supervision, the PhD-thesis would not have been conceivable, nor its continuation of which the result now lies before you.

The work on Feature-based modelling and the design process has been performed with Jos van Leeuwen, who provided the theoretical foundations in his PhD-work and his continued research on Concept Modelling. Our collaboration on the design case analysis has been very informative for the extent of both generic representations and Feature-based modelling in design.

Jan Dijkstra introduced the area of expert systems to me when I had the occasion to assist him for an expert systems course. I was very happy – and anxious – that students engaged in their programming exercises with my work on generic representations: the course and its results formed the first real implementation of generic representations ‘in action.’

Structural Sketcher is the PhD-thesis by Sviataslav (Slava) Pranovich who worked under the guidance of Jack van Wijk from the Computer Graphics group of the Department of Mathematics and Computer Science of TU/e. Despite his non-architectural background, Slava was very keen to understand the domain of architectural design and the notion of graphic units. His questions invariably lead me to clarify my own ideas and challenge my preconceptions.

The track on multi-agent systems has much to thank to my colleague and room-mate Joran Jessurun for implementing the first basic multi-agent program. Our discussions on what agents ‘really’ are have been directive for our thoughts on the development of such systems. I don’t believe we have settled the dispute yet, but at least we have converged to some common understanding what agents are not.
The group members of Design Methods and Design Systems have always been actively engaged in many discussions that formed and sharpened ideas in the research. I thank them for lively debates and their interest in my work.

The foundations and related work of the current report comes from researchers in CAAD and design research all over the world. Their influence is acknowledged throughout this document and I hope to have given credit in the references where this is due. I would especially like to thank the reviewers of this manuscript for their thorough and thoughtful responses. They have helped much to clarify the argument and point out areas where improvement was possible. I am sorry I could not follow all their advices, as this would undoubtedly lead to a second – and quite different – book. Despite all the reviewing and repetitive screening of the text, there may still be mistakes in the text. The responsibility for these of course lies completely with me.

Finally, I want to thank my family and especially my wife, Gabriela, who is my unwavering support and to me a great source of inspiration: děkuji moc!
About the author

Henri Achten was born in 1967 in Venlo, the Netherlands. He studied architecture at Eindhoven University of Technology from 1986-1992, where he obtained his MSc. Degree in Architectural Design in the Design Methods Group under Prof.dr.ir. Thijs Bax. After these studies, he continued with PhD. degree studies in the Design Methods Group on generic representations – the subject of the current book. Promotors of the PhD. research were Prof.dr.ir. Thijs Bax and Prof.dr. Robert Oxman. The PhD. thesis was successfully defended in 1997, after which Henri Achten acquired a post-doc research position in the newly formed Design Systems group. The goal of this project was to provide the architectural-methodological underpinnings in the new VR-DIS research programme (Virtual Reality – Design Information System/Distributed Interactive Simulation). In 2000 Henri Achten became associate professor in the area of design theory and CAAD.

Since 2001, Henri Achten is vice-president of eCAADe, the foundation for Education and Research in Computer Aided Architectural Design in Europe. He is member of the editorial board of IJDC: the International Journal of Design Computing, editor of IJAC: the International Journal of Architectural Computing, and has acted as reviewer for the journals AIEDAM: Artificial Intelligence in Engineering Design, Analysis and Manufacturing, IJDC, IJAC, Design Studies, and IT-Con. He also participated in the European research projects on continuous lifelong learning AVOCAAD pilot project and AVOCAAD Multiplcator. He has lectured in Helsinki (HUT) and Prague (CVUT). He has organised three national events on design research and design education, participated in the organisation of the CAAD Futures 2001 conference in Eindhoven, and is program chair of the CAAD Futures 2005 conference to be held in Vienna. At the moment he is (co)author and (co)editor of five books and over fifty scientific papers.
References


Quincy, Q. de (1825). Type. Translated by Anthony Vidler. Oppositions 177 (8), pp. 147-150.


