LOWERING THE THRESHOLD FOR COMPUTERS IN EARLY DESIGN: SOME ADVANCES IN ARCHITECTURAL DESIGN

Dr. Ir. Henri Achten,
Design Systems
Faculty of Architecture
Eindhoven University of Technology, Building and Planning
PO Box 513, 5600 MB Eindhoven
Netherlands

ABSTRACT
The design drawing is an important medium for establishing design support by means of computers. Architects intensively use graphic representations to communicate their design ideas personally, between professionals, and others. In this study, we consider line drawings such as sketches or drawings. Based on previous investigations, we propose that there exist well-structured graphic representations termed graphic units. Examples of graphic units are: grid, zone, axial system, contour, and element vocabulary. Associated to graphic units are specific kinds of design information that is relevant for this kind of representation (for the grid: module size, modular coordination, dependent grids; for the zone: definition of functional elements, dimensions of zones and margins, etc.) Therefore, understanding graphic units forms a basis for computer-based interpretation of drawings during the early phase of the design process. In this paper we present two principally distinct applications: "paper plus" and "pen plus." The "paper plus" approach features automated recognition of graphic units as the architect is drawing. Work in this area has been based on techniques from multi-agent systems and Case-Based Reasoning. The "pen plus" approach features drawing tools based on graphic units. Work in this area has been based on techniques from expert systems and computer graphics. The "paper plus" and "pen plus" approaches show how an earlier understanding of graphic representations in architectural design is possible, thus lowering the threshold for the use of computers in the design process.

KEYWORDS
Graphic representations, graphic units, automated drawing recognition, drawing tools

INTRODUCTION

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 traditional drawing techniques used by architects.
- Automated inference of design intentions from 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.

In this study, we consider line drawings such as sketches or drawings. In order to make drawings accessible for computers, it is necessary to understand the conventions of depiction and encoding of drawings in line drawings. Conventions of depiction concern the projection techniques such as plan, section, and perspective. Conventions of encoding concern the graphic means by which additional information is added to the line-shapes such as hatching patterns, line types, and line width. In an analytical study of some 220 drawings taken from an extensive period of time (13th to 20th century), 24 specific kinds of drawing elements that have a well-established use and meaning to architects were identified (Ref [1]). These kinds of drawing elements are termed "graphic units" (Table 1).

Table 1: Graphic units

Graphic Unit	Description
Simple contour	Regular shape showing an outline.
Contour	Any irregular shape showing an outline.

Measurement device	Measure for establishing (relative) dimensions.
Specified form	Contour with specified dimensions.
Elaborated structural contour	Outline with structural detail.
Complementary contours	Composition of outlines.
Function symbols	Textual indication of function.
Zone	Area with specific use or function.
Schematic subdivision	Schematic depiction of principal subdivision.
Modular field	Irregular subdivision of area along coordinating lines.
Refinement grid	Grid with smaller module coordinated in other grid.
Schematic axial system	Schematic depiction of organisation of axes.
Axial system	Organisation of axes applied to building design.
Grid	System of modularly repeating coordinating lines.
Tartan grid	Double grid based on two alternating modules.
Structural tartan grid	Tartan grid with structural elements.
Element vocabulary	Set of simple shapes depicting (interior) elements.
Structural element vocabulary	Set of simple shapes depicting structural elements.
Functional space	Outline combined with function indicator.
Partitioning system	Schematic depiction of more detailed subdivision.
Proportion system	Diagram showing how proportions are derived.
Combinatorial element vocabulary	Precise relationships between elements.
Circulation system	Principle layout of circulation.
Circulation	Layout of circulation applied to building design.

The following observations can be made:

1. Half of the 24 identified graphic units represent structuring devices (e.g., grid, zone, and axial system) rather than concrete building elements (e.g., contours, functional space, and circulation). This implies that architects have an extensive set of representations for organising the design.
2. 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).
3. Drawings that contain the same graphic units can be considered to deal with the same kind of design decisions. We term such drawings “generic representations.”
4. Graphic units encode things such as composition, layout, modularisation, circulation, and interior in a graphic way.

Graphic units convey information that is generally shared by the architectural community. Much of this information can be derived by examining the drawing. We propose therefore, that graphic units form the basis for a visual language on which to build more sophisticated design support.

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 that works on whatever the architect is drawing: to put graphic units “in the paper” (“Paper Plus”). The other way is to build drawing tools that generate graphic units: to put graphic units “in the pen” (“Pen Plus”): see Figure 1.

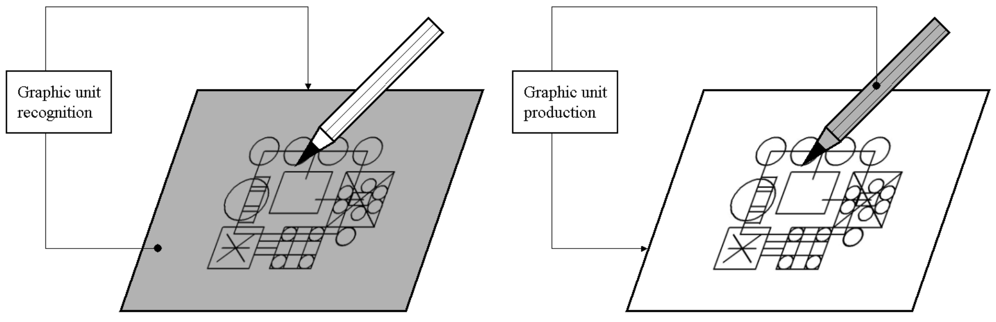

Figure 1

Both strategies have been pursued, and are in varying degrees of development:

1. An expert system for a building type (Pen Plus approach).
2. Sketch tools that create graphic units (Pen Plus approach).
3. A Case-Based design aid system (Paper Plus approach).
4. Automated recognition of graphic units in sketches (Paper Plus approach).

In the paper, we will first briefly discuss related work, and then provide an overview of the work developed in the Pen Plus approach, followed by the Paper Plus approach. Finally, the paper ends with a brief discussion and conclusions.

RELATED WORK

The goal of the research work is to provide design support to the architect while he is working. We want to achieve this through understanding of the graphic units that are used. We look at 'drawing in action' for two reasons: (1) use drawing actions as clues for recognizing graphic units (Ref [2], [3]); and (2) to provide support through the system as the designer is working, in a very short time span after the drawing actions.

Related work falls in the category of CAD systems and sketch analysis. Computational work on interpreting drawings has focused mainly on bottom-up analysis from primitives to larger constructs, for example in facades (Ref [4]), or more complex shapes in plans (Ref [5]). Examples for graphic design support are the Electronic Cocktail Napkin (Ref [6]), Hypersketch and PHIDIAS (Ref [7]), Netdraw (Ref [8]), and EsQUIsE (Ref [9]). The sketch functionality and interpretation of EsQUIsE is particularly close to the current work. What is missing is a well-founded basis of elements that can be considered for analysis and computational interpretation.

Related research in sketching, aimed at identifying pervasive structures is less common: Ref [10] look at commonalities in sketches, in particular short-hands for drawing the same concepts; McFadzean's Computational Sketch Analyser (Ref [3]) takes several sketch-acting clues as indicators what the current status of the design process is; Ref [11] proposes a taxonomy of elements in sketches, breaking them down into organizational units not unlike graphic units. Ref [12] note a number of mechanisms between sketches without providing a more refined set of criteria to track design development.

PEN PLUS APPROACH

Graphic units offer 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 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 specialized elements for each graphic unit.

PEN PLUS: EXPERT SYSTEM

From the research presented in [1], it appears that drawings can contain one to four different graphic units (and of course multiple instances per graphic unit). For example, there are many different drawings possible that display multiple spaces (graphic unit contour) arranged in a grid (graphic unit grid), controlled by axes (graphic unit axial system). All these drawings fall under the generic representation "Axial System in Contour in Grid" and deal with the same design decisions. Sequences of generic representations represent the development of a design process by adding more graphic units, by following up more general graphic units with more specific graphic units (this termed "successive graphic units"), or by adding different themes addressed by the generic representations. A possible sequence of 24 generic representations has been demonstrated in Ref [1]. The first seven steps of this sequence have been applied to the office building type and implemented in a Frame-Based design aid system. This work has been performed in a course on expert systems and implementations of the sequence were made by students (Ref [13]). The main knowledge structure underlying all systems is the so-called frame (Ref [14]), 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 (see Figure 2).

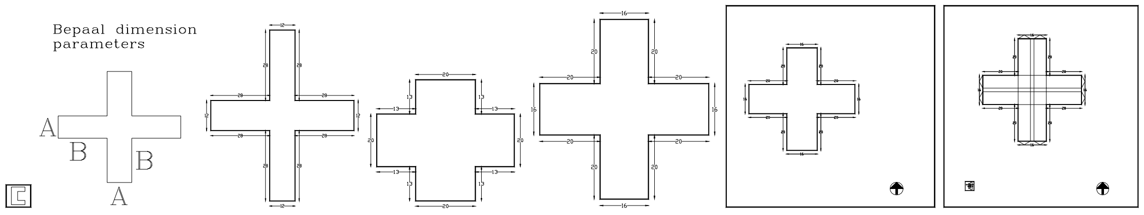

Figure 2

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. Although the systems are rather limited, 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.

PEN PLUS: STRUCTURAL SKETCHER

In a recently concluded PhD-thesis in Computer Graphics by Slava Pranovich (Ref [15]), a sketching system called the Structural Sketcher was developed for architects, which utilises graphic units as basic elements for creating designs. This approach is more versatile than the expert system mentioned above, because any combination of graphic units can be made by the designer. Therefore, what the system should be able to do is:

- Track the instantiation of each graphic unit that is made by the architect.
- Interpret the relations between the various instances of the graphic units.
- Allow personal changes within the relations.
- Maintain consistency of the relations over various manipulations of the graphic units.

From all 24 graphic units, 14 graphic units have been implemented in the system:

- Five graphic units dealing in some way with spatial shapes 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.
- Four graphic units dealing with grids are considered as grids and defined as sets (N_{GC}, p_0 .. p_{NGC}) with N_{GC} is the number of grid components, p_0 defines the origin of all grid components, and p_1 .. p_{NGC} defines separate grid components.
- The graphic unit axial system is defined as (p_1, p_2, M) with p_1 and p_2 defining the axis, and M is {(g_i, g_j), ..} with g_i and g_j identifying the mirrored objects.
- The graphic unit zone is defined as (N_p, p_1 .. p_n) with N_p the number of vertices that define the area of the zone, and p_1 .. p_n the vertices of the zone.
- Three graphic units that deal with standard objects in regular layouts are defined as images (*bm*, p_1, p_2, p_3, p_4) with *bm* a bitmap image, and p_1, p_2, p_3, p_4 the vertices of the parallelogram in which the bitmap fits.

Transformations such as translation, rotation, skew, and scaling over graphic units amount to applying the transformation first to the anchor points relative to which a graphic unit is defined, and then to all vertices that define the graphic units. Transformations are engaged via the KITE user interface (Ref [16]).

The complication lies in recording the relations between graphic units and maintaining these relations when one or more graphic units are transformed. All relations are defined as vertices in a directed graph on the basis of the anchor points, which are the nodes. The use of generic constraint satisfaction techniques (Ref [17]) to maintain relations is complex and difficult to understand for a user. A simpler solution is to choose one transformation at a time and send it across the edges in the graph. The transformation starts at a point selected by the user and is propagated through the anchor points using relations, affecting the graphic units that are associated to the anchor points. Different types of constraints can be taken into account via transmission properties that are stored in relations and anchor points. To prevent ambiguities of multiple possible paths, we derive a spanning tree when transformations are propagated. A spanning tree of a graph is a tree that connects all vertices (Ref [18], [19]). This ensures that every connected graphic unit is affected only once. Every action of the user with the manipulator is processed as follows:

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.
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. Anchor points are updated.

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

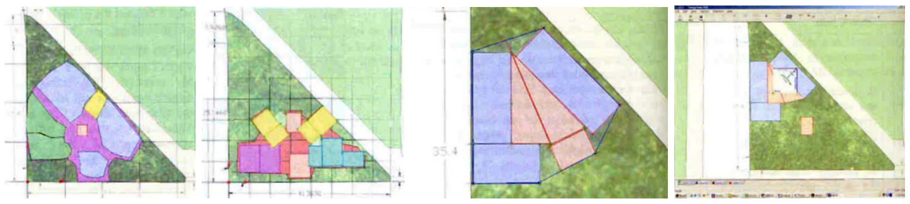

Figure 3

The research resulted in a working sketch prototype, called the Structural Sketcher, which enables architects to sketch with structured elements based on graphic units. In comparative tests against traditional pen and paper, several generic drawing software, and CAD software for architects, Structural Sketcher usually performed as second best to pen and paper.

PAPER PLUS APPROACH

As stated in the previous section, the disadvantage of specialized tools for producing drawings is that the architect needs to learn new techniques. Because architects are already very familiar with graphic representations, and skilled in their production, another approach is to build tools that can understand drawings rather than force the architect to learn new tools. In this section, such recognition based on the theory of graphic units and generic representations is presented. The main functionality of any 'Paper Plus' system is the automated recognition of drawings. Two tracks have been developed to work on 'Paper Plus' tools: automated recognition by multi-agent systems, and the use of techniques from Case-Based Reasoning for identifying graphic units.

PAPER PLUS: MULTI-AGENT SYSTEM APPROACH

Automated recognition of sketches and 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, etc). Within these assumptions, and usually within a narrow 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 widen the scope for recognition systems.

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. The recently developed notion of multi-agent systems (Ref [20], [21], [22], [23]) 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.
- Agent-systems can function in dynamically changing environments, where resolution is not always possible. Drawing constitutes such an environment.

The work on multi-agent systems proceeds 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.

We have established an agent framework for developing a multi-agent system. An agent in the framework has input, output, and an internal state and processes that are closed to the outside world. The input part senses the world environment and receives broadcast messages. The output part manipulates the world environment and broadcasting messages (Figure 4).

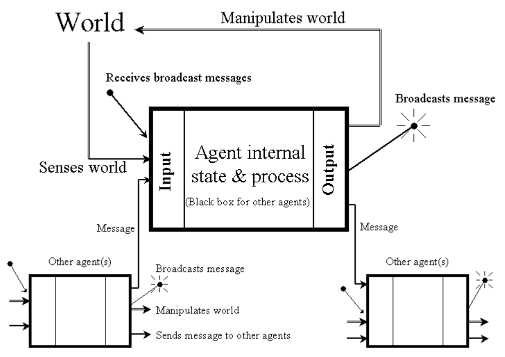

Figure 4

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. We establish implicit control 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 (Ref [24]: p. 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 (if the agent can manipulate).
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 (Ref [25]). The second track, extending the multi-agent system to graphic unit recognition, has yet to be started (the next section provides an approach for this). From the Mah Jong systems, we learn how to coordinate decision-making between agents, how the principles of communication by broadcasting in the multi-agent system apply, and how combined search- and recognition techniques for heuristics function in a dynamic setting. Finally, we gain some insight 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, and so forth. These issues need to be resolved in further development.

PAPER PLUS: CASE-BASED REASONING APPROACH

A system which has a graphic unit recognition mechanism such as the one described above, delivers as output a ranked list of candidate graphic units that are detected in the drawing. Combinations of graphic units form generic representations and therefore it is possible to provide the architect with relevant information on the design decisions that are related to generic representations. In the Expert System example, this was demonstrated for the specific domain of office buildings. The approach with multi-agent based recognition of graphic units has the advantage that it follows the architect rather than prescribes his design process. In order to assist in the recognition of graphic units, we have established a decision tree that classifies graphic units. Furthermore, we have established a query composition algorithm that can deal with cases of near- or mismatching graphic units, and a look-up table for generic representations (Ref [26]).

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 (Ref [27]: p. 52-53). The decision tree in our case provides a question-answer mechanism that leads to identification of a graphic unit in a drawing. The nodes are not bifurcal to decrease the number of questions. The process starts when the designer prompts the system for case retrieval. Each pass through the tree identifies one graphic unit (Figure 5).

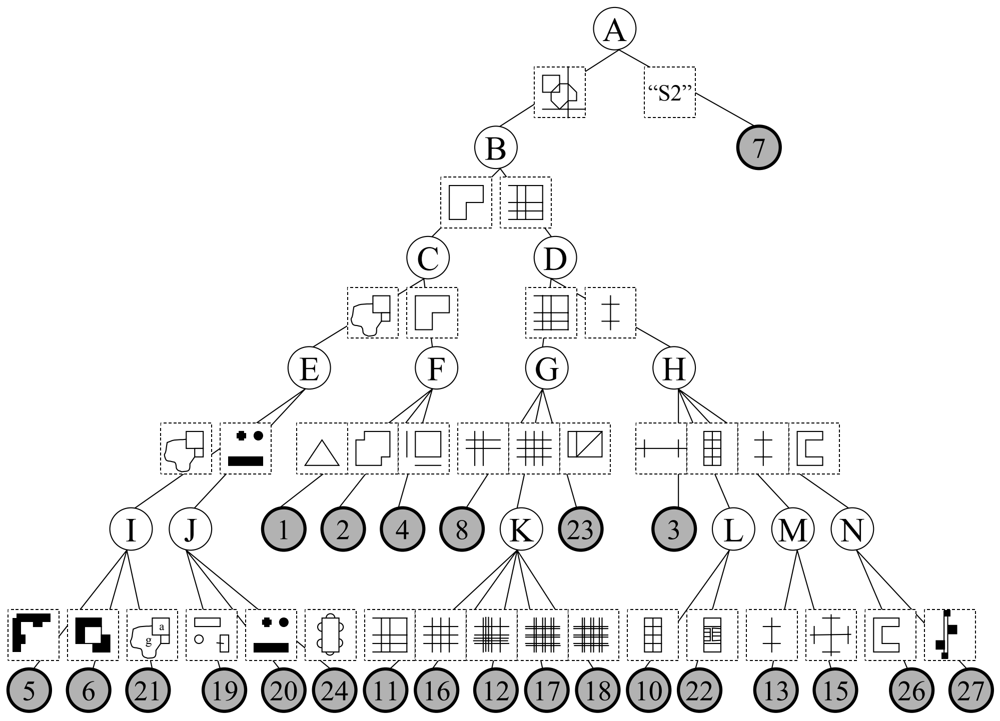

Figure 5

The result of identifying graphic units is a list L: L_{gu} {gu_1, gu_2, …, gu_n} with gu_i an identified graphic unit. Such a set of graphic units may have one perfect match as a generic representation or none. However, reasonably similar generic representations may also be relevant to the architect. A generic representation is reasonably similar when it has graphic units that either are one step more or one step less developed than one of the graphic units in L_{gu}, when it has a subset of the graphic units in L_{gu}, or a combination of both. Rather than taking L_{gu} and finding matches in the list of generic representations, we expand L_{gu} with all possible relevant variations and thus construct a query Q {{gu_1, gu_2, …, gu_n}, w_i}, …} with {gu_1, gu_2, …, gu_n} an expanded set of graphic units, and w_i the weight of that set. Finding the matching generic representations then simply is a matter of checking the table of generic representations. Since the query Q is already rank-ordered by weight, the result is also rank-ordered by weight. From the generic representations that are found, we can then determine the associated design decisions and provide the architect with relevant knowledge in a particular domain concerning those design decisions.

CONCLUSIONS

Design support tools that do not deal with highly specialised questions and (life-)critical systems, should be least obtrusive and distractive as possible, and determine the content of the design implicitly, not by continuously asking the designer. Understanding drawings is an important step to achieve this goal. In the work above, we have demonstrated that a sound understanding of the conventions of depiction and conventions of encoding in the form of graphic units and generic representations is the basis for developing design support tools that interpret and follow the architect's moves. Such systems are capable to infer design intentions and provide the necessary knowledge for the architect to make design decisions. Design drawings therefore, although they look fuzzy and ambiguous in the early design phase, can be machine interpreted and thus the threshold for using computers in early design can be lowered.

REFERENCES

[1] Achten, H.H.: *Generic representations - an approach for modelling procedural and declarative knowledge of building types in architectural design*. Ph.D. Thesis, Eindhoven: Eindhoven University of Technology, 1997.
[2] Kavakli, M. – Scrivener, S.A.R. – Ball, L.J.: *Structure in Idea Sketching Behavior*. Design Studies, **19**(4), 1998, p. 485-517.
[3] Fadzean, J. Mc.: *Computational Sketch Analyser (CSA)*. In: "Proceedings of the 17th eCAADe Conference" (Editors A. Brown, M. Knight and P. Berridge). Liverpool: The University of Liverpool, 1999, p. 503-510.
[4] Pellitteri, G.: *A Tool For a First Analysis of Architectural Facades*. Automation in Construction, **5**(5), 1997, p. 379-391.
[5] Park, S-H. – Gero, J.S.: *Categorisation of Shapes Using Shape Features*. In: Artificial Intelligence in Design 2000" (Editor J.S. Gero). Dordrecht: Kluwer Academic Publishers, 2000, p. 203-223.
[6] Gross, M.D.: *The Electronic Cocktail Napkin - A Computational Environment for Working With Design Diagrams*. Design Studies, **17**(1), 1996, p. 53-69.
[7] Call, R. Mc – Johnson, E. – Smith, M.: *Hypersketching: Design as Creating a Graphical Hyperdocument*. In: "CAAD futures 1997" (Editor R. Junge). Dordrecht: Kluwer Academic Publishers, 1997, p. 849-854.
[8] Qian, D. – Gross, M.D.: *Collaborative Design With Netdraw*. In: "Proceedings of the CAADfutures '99 Conference" (Editors G. Augenbroe and C. Eastman). Dordrecht: Kluwer Academic Publishers, 1999, p. 213-226.
[9] Leclercq, P.P.: *Programming and Assisted Sketching – Graphic and Parametric Integration in Architectural Design*. In: "Computer Aided Architectural Design Futures 2001" (Editors B. de Vries, J.P. van Leeuwen, and H.H. Achten). Dordrecht: Kluwer Academic Publishers, 2001, p. 15-31.
[10] Do, E.Y-L. et al.: *Intentions in and Relations Among Design Drawings*. Design Studies **21**(5), 2000, p. 483-503.
[11] Koutamanis, A.: *Prolegomena to the Recognition of Floor Plan Sketches*. In: "Design Research in the Netherlands 2000" (Editors H. Achten, B. de Vries, and J. Hennessey). Eindhoven: Eindhoven University of Technology, 2001, p. 95-105.
[12] Rodgers, P.A. – Green, G. – McGown, A.: *Using Concept Sketches To Track Design Progress*. Design Studies **21**(5), 2000, p. 451-464.
[13] Achten, H.H. et al.: *Knowledge-Based Programming for Knowledge Intensive Teaching*. In: "Proceedings of the 13th European Conference on Multimedia and Architectural Disciplines" (Editors B. Colajanni and G. Pellitteri). Palermo: Universita di Palermo, 1995, pp. 139-148.
[14] Coyne, R.D. et al.: *Knowledge-Based Design Systems*. Reading, Massachusetts: Addison-Wesley Publishing Company, 1990.
[15] Pranovich, S.: *Structural Sketcher: A Tool for Supporting Architects in Early Design*. PhD-Thesis, Eindhoven: Eindhoven University of Technology, 2004.
[16] Pranovich, S.S. – Wijk, J.J. van – Overveld, C.W.A.M. van: *The Kite Geometry Manipulator*. In: "CHI2002 vol. 4, issue 1, Minneapolis (USA), April 20-25", ACM SIGCHI, 2002, p. 764-765.
[17] Hower W. – Graf, W.H.: *A Bibliographical Survey of Constraint-Based Approaches to Cad, Graphics, Layout, Visualization, and Related Topics*. Knowledge-Based Systems, **9**(7), 1996, p. 449-464.
[18] Bellman R.E.: *On a Routing Problem*. Quarterly of Applied Mathematics **16**(1), 1957, p. 87-90.
[19] Moore E.M.: *The Shortest Path Through a Maze*. In: "International Symposium on the Theory of Switching" 1959, pp. 285-292.
[20] Russell, S. – Norvig, P.: *Artificial Intelligence: A Modern Approach*. Upper Saddle River: Prentice-Hall Inc., 1995.
[21] Wooldridge, M. – Jennings, N.: *Intelligent Agents: Theory and Practice*. The Knowledge Engineering Review **10**(2), 1995, p. 115-152.
[22] Nilsson, N.: *Artificial Intelligence: A New Synthesis*. San Francisco: Morgan Kaufmann Publishers Inc., 1998.
[23] Weiss, G.: *Multiagent Systems – A Modern Approach to Distributed Artificial Intelligence*. Cambridge: The MIT Press, 2001.
[24] Franklin, S.: Artificial Minds. Cambridge: The MIT Press, 1995.
[25] Achten, H.H. – Jessurun, J.: *A Multi-Agent Mah Jong Playing System: Towards Real-Time Recognition of Graphic Units in Graphic Representations*. Acta Polytechnica **43** (2), 2003, p. 28-33.
[26] Achten, H.H.: *Design Case Retrieval by Generic Representations*. In: "Artificial Intelligence in Design '00" (Editor J.S. Gero). Dordrecht: Kluwer Academic Publishers, 2000, p. 373-392.
[27] Mitchell, T.M.: Machine Learning. New York: The McGraw-Hill Companies, Inc., 1997.