LOCAL VALUES in a NETWORKED DESIGN WORLD

ADDED VALUE OF COMPUTER AIDED ARCHITECTURAL DESIGN

DUP Science
scientific committee

HENRI ACHTEN                      TU-EINDHOVEN
ALEXANDER ASANOWICZ               TU-BIALYSTOK
ANDY BROWN                        UNIVERSITY OF LIVERPOOL
ADAM JAKIMOVICZ                   TU-BIALYSTOK
JONAS AF KLERCKER                 TU-LUND
GERNOT PITTIONI                   INGENIEURBÜRO DR. PITTIONI MÜNCHEN
MARTIJN STELLINGWERFF            W&K, SINT-LUCAS / TU-DELFT
JOHAN VERBEKE                     W&K, SINT-LUCAS

DUP Science is an imprint of
Delft University Press
P.O. Box 98
2600MG Delft
The Netherlands
Phone    +31.15.2785678
Fax      +31.15.2785706
E-mail   info@library.tudelft.nl

Editors
Stellingwerff, Martijn           Hogeschool voor Wetenschap & Kunst, Sint-Lucas Architecture
Verbeke, Johan                   and TU-Delft, Faculty of Architecture, Form & Media Studies
                                 Hogeschool voor Wetenschap & Kunst, Sint-Lucas Architecture

Keywords
Architecture, Local values, Globalisation,
Computer Aided Architectural Design.

ISBN 90-407-2507-1
DUP Science

Copyright © 2004 by Hogeschool voor Wetenschap & Kunst, Sint-Lucas Architecture,
Brussels. All rights reserved. No part of this book may be reproduced in any form, or by
any means without written permission of the author.

Printed in the Netherlands.
A Critical Evaluation of Early Stages Software in its Capacity of Coping with Contextual Issues

AUTHORS
Anna Maria Chrabin,
Herman Neuckermans
Katholieke Universiteit Leuven
Faculteit Toegepaste Wetenschappen, Dept.ASRO
Heverlee
BELGIUM

Jaroslaw Szewczyk
Bialystok University of Technology, Faculty of Architecture
Bialystok,
POLAND

Abstract
In this paper we analyse critically early design stages software in its capacity of coping with contextual data at large (i.e. representing cultural, aesthetical context, etc.). We identified 5 categories of early stages software: geometry based graphic editors, evaluation architectural software, generative and shape-grammar based systems, evolutionary systems and other systems. Calling the object under creation during of the early stages a CAD conceptual model, we will investigate to what extend this software allows the architect to experience and represent the context in which a design is situated. Especially we will focus on its capacity to allow interaction, playful interaction on our way to the design. Designers, and particularly architects, interact with the local context similarly to interacting in a game: the context influences the users’ decisions, surprises them and causes permanent changes to their ways of thinking. On the other hand, architects permanently shape and reshape the context, and reduce the context to a protean point of reference. Such behaviour characterises creative thinking that is crucial for the early stage of design.

The investigation led us to the conclusions that the effective interactivity with the context needs simple rules, a plain interface and data reduced as simple as possible, especially when interaction with the context is performed during the early stages of a design process. The findings can be used in organising computer environments for early-stage design.
1. Scope

Good architecture always carries particular values. Within these values the place, the local context is very important. The context plays a significant role in the development of the whole design, because it determines what the newcomer has to be with regard to what is already there. The context plays a double role in the game: it imposes constraints and at the same time offers opportunities. This means that the site has capabilities! The local context can be perceived in two ways: firstly as the physical and tangible environment of the site, secondly as the cultural or conceptual context, which amongst others relates to the actual currents and tendencies in architecture.

Two schools of looking at the local context can be distinguished: the first one states the local context is important; the second one states that the local context is not important. We consider contextual information to be important for architecture, because we think it is important to respect and reinforce the local identity and the character of the place. Besides, the context offers interesting clues to start from.

Different tools deal – or not – with different meanings of the context. We found that existing computer tools, assisting architects with their work, do not help them during the conceptual phases of the design efficiently, because of their lack to support contextual issues. Software tools, enabling the users to experience the surroundings through ‘conceptual interplaying’ with the local condition, have to help them with creative conceptual design more successfully. We are convinced that computer-aided design is capable of enabling such interactivity.

We investigated some aspects of interaction with the context in the design process asking the following questions:

1. How does the existing software serve and control contextual data?
2. How can the different types of contextual data be mapped to the users’ activities and software tools?
3. How is the computer capable of assisting the architect in evaluating and experiencing the context?
4. How can a computer allow a playful and exploratory interaction with the context most successfully?
5. How to enhance the use of the context?

Software for conceptual design presents a great variety of approaches to the design problem, which only proves the complexity of the design process. But, successful interaction with conceptual data during the early stages of design requires detailed analysis of all design requirements, especially the most underestimated in CAD: the contextual data, which seems to be crucial for the development of a design idea.

We made a few working assumptions for the further investigation:

- Designers, and particularly architects, interact with the local context similarly to interacting in a game: the context influences the users’ decisions, surprises them and causes permanent changes to their ways of thinking. On the other hand, designers permanently shape and reshape the context, considering it a protean reference. Such behaviour characterises creative thinking.
- We can consider here each stage of the design process as an entirely distinct “game”. This assumption is rather made for simplification of the analyses and does not result from the design and spatial reasoning internally, because in real design these different stages are overlapping each other.
• We focus especially on the early stages of design, for a number of reasons:
  - The most important decisions are taken during the early stages of the design process. The early decisions can determine the idea of the project and they influence the development of the model;
  - In the early, conceptual stages of design, objects are created with aesthetical qualities and values, subsequently developed in the next design stages;
  - Early stages of design tend to focus on architectural notions rather than on construction or cost data.

• We focus on contextual data. The role and methods of interacting with contextual data result from the importance of the context:
  - Contextual data seem to be important because of their significance in decision making in the conceptual phases of the design process;
  - Contextual data seem to be important because the context carries unique information and values, which are valuable during the whole design process;
  - The local aesthetical and cultural context remains extremely important as it leads to aesthetical diversity.

2. Early design stages software

To set the stage for our investigation, we propose the following classification of architectural software according to the main methods of interacting with data in 5 categories:

**Category I: Geometry – based vector graphics editors**, such as drawing boards, 3D modellers and VR environments

Information in these systems, which present geometry-centric approach, is organised around pure geometry. We can map some old and new graphical software to this category. The examples are: 3D modellers, VR environments, vector graphic software, various general CAD environments (AutoCAD, MicroStation, etc.), sketching boards.

**Category II: Evaluative architectural software based on the “Single Building Model” (SBM) paradigm**

Information in SBM-based CAD systems that present construction-centric approach is organised around the virtual building data or the facility data. Most of the latest CAAD systems fall into this category, especially if presenting “Single Building Model” approach. The examples of the older software were GDS, RUCAPS and Sonata (developed at the end of the 70’s). The newest examples are object-oriented parametric modellers such as Autodesk Revit, Archicad from Graphisoft and Allplan FT from Nemetschek.

**Category III: Generative systems for conceptual design**

Information in these systems, which present conceptual, topology-centred approach, is organised around graphical (or, sometimes numerical) programmable data representing the main architectural spaces, maintaining the topology and hierarchy of these elements. Generative systems were mainly space allocation systems, developed for generating plans automatically. An example was ‘General Space Planner’ [Eastman, 1973].
Category IV: Evolutionary systems including advanced knowledge-based technologies

The experimental, architectural evolutionary systems are often based on the most advanced adaptive models, genetic algorithms, parallel, evolutionary and adaptive search procedures etc. We can map a few experimental tools for conceptual design to this category. An example can be Drawing Analogies [Gross, M. and E. Do, 1995].

Category V: Other systems

These categories have been identified according to their approach to data management. For each category we have listed a number of representative software tools which are published and/or available on the market. There are several experimental tools supporting conceptual design, and not many commercial ones. In the following paragraphs we will analyse these ordered by these 5 categories.

2.1 Review of Tools Enhancing Early Stages Design

Category I

Architectural Studio from Autodesk is a conceptual, commercial drafting tool and a 3D study modeller with a pure, simplified interface. Architectural Studio stores both sketches and models with emphasis on their background and layout, thus enabling user to create and retrieve very basic information about the spatial context. It does not support any enhanced relations between the context and the model.

SketchUp from @Last Software and D-Board from Nemetschek are commercial software similar to Architectural Studio, but the first supports 3D modelling only, and the second supports only 2D drafting. The context data is not supported anyway.

DDDoolz [De Vries, 2000] is a three-dimensional sketcher for mass study and conceptual design, developed at the Eindhoven University of Technology by Bauke de Vries and Henri Achten. The DDDoolz has a reduced interface and allows easy visual evaluation of created models. The context data is not supported.

ErgoDESK is a 3D/VR modeller supporting 2D gesturing for 3D model creation. It enables two handed interaction and seamless transitions between 2D/3D, and utilises hardware: 6 D.O.F. trackers with stereoscopic cameras, Lightpen for 2D input, and ActiveDesk (made by Fakespace Systems). ErgoDESK does not support the context data.

Harold [Cohen 2000], developed by Jonathan Cohen’s team at the Brown University in Providence, encourages playing with images through the idea of a billboard displaying projections of strokes drawn by the user. The projections are viewpoint-dependent, and they can contain much reduced spatial context data. The application enables designers to test the spatial relationships between objects and their contexts quickly and intuitively, but in a very simple manner. The program also allows the user to model the shape of the terrain.
Moderato [Mase 2000], developed by Jitsuro Mase at the University of Cambridge, allows direct sketching within a digital three-dimensional space, without the need of sketch-recognition. The context data is not supported.

Pegasus [Igarashi, 2000] developed by Takeo Igarashi at the University of Tokyo, Institute of Technology, is a very simple application, based on the idea of supportive interface. It allows conceptual drawing of pure 2D shapes according to two operations: beautification (or rather enhancing precision) and prediction user’s decisions. The programme works regardless to the context information, except of the context understood as a very close relation of the two-dimensional strokes. The programme has nothing to do with other local values.

Quicksketch [3] was developed by the Computer Graphics Group at the Technical University of Ilmenau, to be used with pressure-sensitive LCD screens and allows for three-dimensional modelling from two-dimensional strokes. It was created as a front-end to more sophisticated modelling programmes. Drawing starts from the freehand simple stroke. The application checks the local context and tries to find the geometrical relationship in the shape. In other words, the programme tries to recognise the intentions of the user and translate the shape into a more geometrical representation. Quicksketch
can apply also other parameters specified by the user and related not only to the context of the drawing. Thus, users have possibilities to define various simple “contexts”, called modes.

![Figure 5. Quicksketch automatically recognises adjacencies and right angles and subsequently cleans up the drawing [3].](image)

**Category II**

There are not typical construction-centric software for conceptual design, although we can partially map few pieces of software to this category.

**Sketch 3D** [Do, 2000] developed by Ellen Yu-Luen Do at the University of Washington, is a pen-based tool recognizing simple 2D shapes and their spatial relationships. According to those relationships it generates models, utilising the Virtual Reality Modelling Language. Thus Sketch 3D utilises idea of context data, although in a very limited way.

![Figure 6. Left – A floor plan drawn with Sketch 3D. Right – 3D VRML world with walls and columns [Do, 2001].](image)

**Sculptor** [Kurmann & Engeli, 1996] was developed by David Kurmann’s team (Architecture and CAAD, ETH Zürich). It allows for direct and intuitive conceptual modelling using closed solid objects, voids and negative volumes, which can be grouped.
hierarchically. Models can represent real buildings or have only pure, functional and behavioural characteristics. Two main constraints, gravity and detection of collision, give the user the feeling of natural, physical modelling, a bit like playing with children’s blocks. The user finds spatial relationships and reciprocal influences of objects very easily.

Sculptor is able to implement into the objects various types of knowledge, such as objects behaviour, and relationships between architectural elements. The behaviour of objects is related to their context, and it can be reactive, interactive and autonomous. All objects interact with themselves, the user can analyse the influence of some objects on another or on the whole context.

Teddy [Igarashi et al., 1999] was developed by Takeo Igarashi at the University of Tokyo. The programme recognises two-dimensional strokes and transforms them into three-dimensional models. It does not support analysis, evaluation and communication. Teddy hardly deals with the regular architectural elements in favour of free soft-edges shapes. The context data is not supported at all, because Teddy works only with one object at the time.

Figure 7. Sculptor: Acting with space [Kurmann & Engeli, 1996].

Figure 8. Teddy in use [Igarashi et al., 1999].

Category III

SEED (Software Environment to support the Early phases in building Design) [38] divides the preliminary design process into the following phases: architectural programming, schematic layout design, and schematic configuration design. Each SEED module, supplying one of the design phases, offers designers a broad range of form generation capabilities that range from stepwise construction under the designer's control to the automatic generation of design alternatives and case-based design. Context information is not supported intentionally, but it can be represented in a simple way.

EsQUIsE [Leclercq & Juchmes 2002] developed by the team of Leclercq at the University of Liège in Belgium, is a sketching tool captures functional relations and topology of the created objects. No relations between elements are forced in advance. However, while the sketch is drawn, a predefined tool recognises the whole context of the drawing and spatial topology between objects. The computer interprets the plan, synthesises strokes, extracts textual marks and analyses the contextual relationships, generating symbol representation of the functional relationship between spaces.
ADDED VALUE OF COMPUTER AIDED ARCHITECTURAL DESIGN

Category IV

The **Electronic Cocktail Napkin** and **Drawing Analogies** [Gross & Do, 1996] were developed by a team of Ellen Yu-Luen Do (Georgia Institute of Technology Atlanta) and Mark Gross (University of Colorado). The Electronic Cocktail Napkin supports sketching, diagramming, storage and retrieval of sketches and diagrams. This tool was used as an interface for other software, especially retrieving information from visual databases (such as Sketchbook and Drawing Analogies - a sketch-based, ‘shape borrowing’ visual reference programs for automated visual indexing of image collections to support creative designing). It retrieves images indexed on conceptual design features, functions, and on spatial relations including adjacencies (such as right of, left of, directly above, below), containments (such as concentric, overlap and contains), and relations among lines (connect, intersect, tee). Such approach makes the tools applicable for acting with the spatial, conceptual context. The software utilises context data for shape recognition, and it builds the pattern definition – a case-based library, in which the same objects and their graphical equivalents can have different meanings in different contexts.

---

**Figure 9.** Left – The Capture Window – the sketch made by the user, Right – The Synthesis Window – the computer interpretation of the sketch [8].

**Figure 10.** The Topology Window – the graph of the functional spaces [8].
The blob-like shape is taken as an example, to show how the software works. The Napkin cannot determine whether it is a square or a circle. The problem can be solved in two ways: either the user overdraws the existing drawing using a more accurate representation, or the programme finds another known local configuration of the objects, which allows choosing the correct context. An example with a wheel wagon shows that the analysed shape should be interpreted as a circle, not a box, just because of the wagon context.

Figure 11. (a) Shape that can be seen as circle or square; (b) The user resolves a problem by overdrawling the shape as a box; (c) The Electronic Cocktail Napkin resolves as a circle from its position in the context [Gross & Do, 1996].

Another example shows the recognition of the curlicue. How the programme recognises the shape depends once again on the context data. In different contexts the same shape can mean something different. For example, in the context of a head, curlicue means hear; in the context of a circuit diagram, a coil; and in the context of a mechanical diagram, a spring. Recognition takes place by analysing the context. If the user does not specify the context explicitly, the Napkin has to look for other elements on the drawing that are represented more explicitly. Summarising: “Context plays a dual role: once the Napkin knows a drawing's context, it can better interpret elements and configurations. On the other hand, a single unique element or configuration can often determine context” [Gross & Do, 1996].

Figure 12. A curlicue maps to different domain interpretations [Gross & Do, 1996].

Figure 13. Left – Cocktail Napkin drawing area. Right – Sketchbook for storing interesting sketches [Gross & Do, 1996].

Figure 14. The Electronic Cocktail Napkin shows context-sensitive relations [Gross & Do, 1996].
RT2 – The Right Tool at the Right Time [Do, 1996], developed by the same team as the software described above, is a freehand sketching environment, developed at the Georgia Institute of Technology and the University of Colorado at Boulder, based on the idea of automatically invoking different tools related to the performed actions. It can recognise a drawing with its context and simultaneously offers support for different activities, such as exploring ideas, finding references, analysis, modification and manipulation. Designing starts on the whiteboard from rough bubble-like sketch. RT2 recognises the drawing context and chooses the appropriate tool for analysing and further developing an idea. This is performed by calling the case-based library containing examples of similar solutions for relations between shapes as well textual information about cases. Before further developing the project, designers would like to see and analyze existing buildings, corresponding to the design requirements. The programme automatically finds references to the examples, which are stored in The Great Buildings Collection CD ROM drive.

RT2 serves as an interface for many knowledge-based design tools, which always refer to the actual context. RT2 recognises the sketch and its context, and offers appropriate tools dependently on the task, without obscuring the design flow. RT2 can be perceived as a more developed and flexible version of Electronic Cocktail Napkin, created by the same team. The idea of working is similar, but the difference lies in invoking the appropriate tools. In the Electronic Cocktail Napkin, the need for calling the appropriate application has to be explicitly stated, whereas RT2 invokes it automatically. Thus a coded system for recognising the context is very important. The authors of the programme noticed that designers use a limited collection of signs, which are characteristic for the particular representation and context.

The Visual Interaction Platform (VIP and VIP-3) [Aliakseyeu et al., 2001] developed at the Eindhoven University of Technical, is a system of hardware and software, based on Augmented Reality technology. It consists of a video projector displaying data on the horizontal “action-perception space”, a digital pen, the enhanced paper prop and the brick elements used for selecting and positioning the virtual elements. The movements of these elements are tracked by a camera located above the table. Sketches made with a
digital pen can be digitally projected onto paper, or can be created on paper with an ink cartridge. A second vertically oriented projection space can be used for communication with other participants of the project or to supply the user with more extensive feedback. The digital pen allows for focusing the designer’s attention directly on the paper. Interaction with the virtual paper is very intuitive: the projected image contains, for example, a library with different architectural objects, which can provide inspiration for the designer. With the use of one brick element the designer can move and manipulate those projected objects. The other dominant brick is used for making annotations, sketching and so on.

The VIP works with domain-specific applications specially devoted to it, such as Architectural Shift+F7, developed as a combination of two other systems: DYNAMO and Idea Space [Heylighen & Segers, 2002]. Idea Space is a network of ideas, combined in different configurations and being reconfigured depending on the context. Each idea, which is a node in a net, is connected with other ideas by their relations. The designer states the first relations, and the new ones are generated by the system, which examines lexical references of the net. By uncovering some important relations, the system gives suggestions for further development of the design. DYNAMO allows for retrieving information from a case base [Schank, 1982; Heylighen and Neuckermans, 2000]. Architectural Shift+F7 enables accessing that case library directly from the VIP design environment. The usage of context data is limited to lexical references connected to case library.

![Figure 17. The VIP [Aliakseyeu et al., 2001].](image)

**Category V**

**SketchBoX** [Stellingwerff, 1999] was developed by Martijn Stellingwerff at the Faculty of Architecture in Delft, on the basis of the concept of physical paper represented by draw-able surfaces. SketchBoX - Variant I allows users to draw on the surfaces of three-dimensional models on semi-transparent wallpapers. SketchBoX - Variant II, an extension of the previous
one, allows for adjusting semi-transparent surfaces freely in space, independently of the 
existing objects.

All variants of the programme provide draw-able surfaces, yet differ insignificantly from 
each other. They allow users to analyse models in different contexts and relations. The 
first variant enables drawing in relation to the model. The second variant broadens the 
scope of analysis: it becomes possible to analyse the shapes in their surrounding context.

![Figure 18. A screen snapshots from SketchBoX [Stellingwerff, 1999].](image)

There are many less known other systems for conceptual design, such as Hand 
Sculpting (enables true 3D drawing), Redliner and Compadres (allow for Internet-
based collaboration), URP (Internet-based architectural model library). Most of them are 
not oriented towards contextual data.

### 2.2 Analysis and critical evaluation

Each category of CAD software has its own method of handling data and organising data 
structures, including contextual data. In general the assumption can be drawn, that the 
context was not a criterion developing these tools for conceptual design.

First, geometry – centred software enhances ‘playing’ with tangible geometry, but it 
usually lacks methods for describing a more complex syntax and semantics of data and 
also misses other non-visual information. Geometry-based vector graphics editors for 
conceptual design do not supply context at all or they present geometry-centric approach 
to context data, reducing it to the closest relations between the two-dimensional strokes 
or to the geometrical relationship inside the shape (Pegasus, Quicksketch), or sometimes 
showing the context through specific drawing layouts (Architectural Studio, Harold). 
Non-geometrical context is not supported.

Construction-centered (SBM-based) software enhances productivity in generating 
design documentation and when projects are repeated and evaluated in recurrent manner. 
The software shapes data containing “intelligent objects”: each object knows how to 
represent itself and follows rules of intelligent behaviour; but due to the great number of 
categories of building elements that the program can't predict, those systems become 
harder to use than generic systems [Richens, 1992]. On the other hand this software 
lacks conceptual data notations, and lacks methods to describe elements of cultural 
heritage connected to geometrical forms and often even lack contextual data at all. This
means that the best-known architectural software, belonging to this category, in fact is oriented towards cost-working applications, has problems with architectural meaning, such as intellectual property value, aesthetic information, cultural context, etc. It is relatively hard to use, and has not been optimised well for architectural data storage yet [Szewczyk, 2002]. The lack of construction-centred software for the conceptual design stage has made the investigation into how this category of software serves the contextual issues superfluous. The exception is the semi-construction-centric Sculptor, which creates spatial objects with their behaviour related to their spatial context.

**Generative systems for conceptual design** encompass very different approaches. One family of software is topology-centered and enables advanced management of relations between architectural spaces, without being able to cope with building or construction elements. The existing software systems have capabilities that support the rapid exploration of design alternatives, but in order to play a significant role in architectural design these approaches have to be upgraded seriously. In most cases they only take into account the so called transportation costs based on distances, journeys and relationships. They are based on sub-optimisation of traffic. We distinguish pure generative from ‘improvemental’ approaches. Till today they cope with such a reduced, almost caricatural model of architecture, so that in fact they still are not capable of coping with architectural design. [Neuckermans, 1975, 1978] They are most frequently used in real estate business context, ignoring the wealth of architectural qualities and of course the context.

Another approach tackles the design process by generating shapes in what is called process of **morphogenesis** [Lynn 2000, Koralevich 2000]. Apart from their power in generating new and ‘until before unknown’ shapes, they have initiated a new formal vocabulary in architecture, as it is the case in the Guggenheim Museum in Bilbao for example. The problem is how to link these shapes, blobs and others, into something which can become a real building. There are a few proposals emerging which try to marry this form generation process to the tectonics of architecture [Leach, 2002].

![Figure 19. Plan renderings of various House permutation examples [Lynn 2000].](image)

![Figure 20. Predator (vacuformed plastic structure – painting/architecture mutant hybrid) [Lynn, 2002].](image)

The shape grammars [Stiny, 1980] play an interesting role in research, but so far, have not been able to produce any significant result in design. One can distinguish formal, functional and structural shape grammars depending on the nature of the rules governing
the generation of the model. Apart from simplistic imitations and mutilated copies of the paradigmatic examples they start from the formal shape grammars have led to nowhere so far. On the contrary the functional shape grammars have been used successfully in generating models in different levels of detail [Mitchell 1975, 1977]. And also in recent times, the structural grammar based software produced for the generation of ‘at random’ structures by Shea [Shea, 2002] is worthwhile to be looked after. G. Schmitt stated in 1992 that “so far grammars have mainly proven useful for the partial analysis and reconstruction of past architecture rather than in the complete synthesis of new architecture” [Schmitt, 1992]. There are only few exceptions, which influences rather academic research than software market (SEED) excluding some software outside architecture, such as HVAC modellers and thermal simulators.

Evolutionary knowledge-based systems utilise case-based libraries containing examples of similar solutions for relations between shapes [Frazer, 1995]. They can retrieve information shaping what we call “conceptual context”, and automatically reference the model to the context examples. Thus, the evolutionary conceptual systems seem to serve the context more efficiently and flexibly than other systems, not limiting it to geometrical surrounding only [Hekkert and Dijk, 2001].

Other systems contain various tools and have not any common characteristics to be summarised, although a few (like SketchBoX) can handle the context in a limited way.

3. Conclusions and Future Work

The possibilities and the barriers related to context data management still need to be discussed. Gross and Do [1995] refer to context as an expansion of the search space. Gero [1998] suggests that expanding the space of design possibilities with larger knowledge bases can inspire creativity.

We tried to define and classify the context from the point of view of the architect using
CAAD, and our approach is naturally consistent with the views of Gross and Do [1995], Gero [1998] and others. But, many problems concerning contextual data are still untouched or they have only been analysed partially in the past, for example:

- how to read the context?
- how to derive primary generators from insights stemming from a careful analysis of the context?
- how to perform context-aware design with computers?
- how to store and manage contextual data digitally?

The sophisticated complexity of the present CAAD software (especially the unusual complexity of CAAD interfaces), which originated from a great number of design factors and rules, can be a barrier in implementation of context data and contextual rules for “contextual-and-conceptual gaming design” (compare: [Matthews et al., 2001]). This is because context-oriented software tools added to CAD systems (which are overloaded by interface data, [Jakimowicz and Szewczyk, 2001]), cannot achieve the goal for which they are implemented. Contextual data, if acting as an expansion of the search space, should be clear, intuitive and simply organised.

To conclude we suggest the following topics still to be researched more in depth:
1. Effective interactivity with the context needs simple rules, a plain interface and data reduced as simple as possible, especially when interaction with the context is performed during the early stages of a design process.
2. Architects need computer tools allowing an exploratory and playful interaction with the context.
3. Contextual data need to be included to common data model;
4. Further investigation upon the local contextual data is needed in order to create tools for interacting with the cultural information carried by the local context.

**Acknowledgements**

The research of Anna Maria Chrabin was supported by doctoral scholarship OE/02/25 from the Onderzoeksfonds of the KULeuven.

The contribution of Jaroslaw Szewczyk was a result of the investigation performed in 2001-2002 and supported by rector’s grant at Bialystok University of Technology (Faculty of Architecture), Grant No. W/WA/4/00, funded by Polish Research Council.

**Bibliography**


[38] Software Environment to support the Early phases in building Design, <http://seed.edrc.cmu.edu/OverView.html>, [01.02.2003].