tailor-made: Adapting and Extending CA(A)D-Systems

Frank Petzold and Dirk Donath
The Bauhaus Universität Weimar
http://www.uni-weimar.de/jpai or http://infar.architektur.uni-weimar.de

Abstract. *CA(A)D systems are nowadays a part of everyday architectural practice and have completely replaced drawing with pen and T-square.*
The standard functions of such systems are (usually) sufficient for the everyday needs of practical work. The question is: can CA(A)D systems be adapted to improve work efficiency for specific users or applications such as those of the architect?
Most current CA(A)D systems already provide some level of support for individual adaptation however these are rarely exploited by the end users.
This paper discusses the teaching of skills for adapting and extending existing CA(A)D systems in current architectural education.
In addition to the teaching of programming and operational skills, the course also examines the critical examination of CA(A)D systems and the formulation of user requirements (analysis), the search for existing solutions, functions or extensions (research) and the technical adaptation or extension of the system (software development).
Using examples from current coursework, the paper illustrates both process and results of teaching practice.

Keywords. *Digital Design Education, 2D Representation, Design Process, 3D Modeling ,Education in CAAD.*

CA(A)D – Ideal and Reality

Today, most CA(A)D systems support all common work processes. The functions provided enable one to work efficiently, to save time and to further optimize project coordination. Building models, auto-generation of plans, sections and elevations as well as photorealistic visualizations are just some of the advantages over existing manual methods.

These and other advantages are propagated at trade fairs, exhibitions and on the manufacturer home pages – CA(A)D as ‘universal remedy’!

However, when one examines architectural practice with CA(A)D systems in the workplace, it is not unusual to find that 3D is only used in the design phase and that the majority of work is undertaken (drawn) in 2D. Even here, the possibilities of CAD (simple functions such as copy, duplicate, blocks and layer structuring) are often not used to their potential.

Why is this? Are the systems too complicated? What has changed since CAD's simple beginnings? Why is CAD used only as a 'computer aid to design'?

A brief look back at the beginnings of CAD in the 1960s helps us to answer some of the question. In 1962, Ivan Sutherland's demonstrated with his Sketchpad that it is possible to draw simple drawings (sketches) with a degree of interactivity on a computer-driven radar screen (light pen, keyboard) (Davies, 1997). In 1965 the first attempt to design a commercial CAD system for 2D technical drawings was undertaken by Lockheed for the aviation sector. At the end of the 1960s the first research activities at the University of Cambridge were undertaken into the possibilities of 3D for representing complex systems (here piping for chemical plants).

As the first home computers began to become more commonplace in the home and workplace at the beginning of the 1980s, the first CAD programs for the market place began to be developed.

A CAD-boom began as the cost of workstations sank and software performance and functionality increased. Systems with powerful specialized functions began to be used.

The step into three dimensional geometric modeling was made possible by increasing hardware performance towards the end of the 1980s, which made digital modeling and assessment from all sides a reality for even smaller companies.

In the years that followed a large number of CAAD-systems were developed to support the architect in his or her planning activities.

Today, CAAD also encompasses building data modeling and life-cycle modeling. The systems have become ever more complex, particularly with regard to the man-machine interface and it is this dialogue, i.e. the availability of clear functions and information provision for the respective work stages, that is of central importance for the computer-aided support of the entire design process.

CA(A)D in university architectural education

CAD is already an integral component in the education of architectural students – either as part of the normal design process or in the form of specific courses.

At present one could divide the budding architects into several groups: the classic architect who uses software, the interested architect who is interested in working more efficiently with his or her software, and the specialist architect who develops his or her own functions in the CA(A)D environment.

For architectural education, this means that it should encompass not only learning and applying CAD systems but also a critical assessment of systems, their performance, potential and functionality, as well as an analysis and understanding of the needs of the user – the architect. Instead of the architect adapting him or herself to the computer, the computer or software system should be a tool for supporting the development of a design idea and the subsequent professional realization thereof.

In addition to existing CAAD courses which focus on the spectrum of computer-aided support in the architectural environment – modeling, animation, interactive systems through to VR/AR systems – we also offer programming courses oriented around the adaptation and extension of commercially available systems.

But why should students of architecture learn to program CAD systems? The didactic aims of the course address several levels:

1. Future architects should be able to make use of the possibilities that current CAD systems offer such as integrating new line styles, patterns, existing macros (such as AutoCAD Lisp, VBA or ARX routines, or GDL for ArchiCAD) in order to react more flexibly to the everyday planning requirements.

2. Designing can be viewed as "programming

at a large scale". It schools logical and precise thought. The course aims less to teach programming and more to develop a more structured design methodology using individual electronic tools.

3. Software developers (also known as software architects) can be compared to architects. They develop plans for the construction, function and form of programs. Programming itself is the process of translating a specification into an understandable language for the computer.

The aim is not only to open up niche areas for architects but also to enable future architects to not only be users but also co-designers of future CAD-systems and their functionality. The skills learnt enable architects to tailor their work environment to their needs and therefore to work more efficiently.

"tailor-made" – a course for adapting and extending CA(A)D systems

Almost all current CAAD systems offer the ability to adapt the user interface, to extend the palettes (line types and patterns) as well as to add new functions via extensions. Many students and practicing architects are, however, unaware of these possibilities and equally unaware of the availability of already-developed extensions and add-ins on the internet.

The "tailor-made" course therefore addresses not only the ability to extend and adapt the user interface or functionality through programming interfaces, but also a more general examination of "computers in architecture". The course therefore follows two main areas:

1. Teaching of skills to adapt existing systems (programming)
2. Formulation of add-ins (for CAD systems) to support particular design and planning tasks.

The first area begins with the adaptation of AutoCAD©, its user interface and the definition of new line types and hatchings and goes on to examine the possibilities offered by AutoLISP to write simple additional functions. The theoretical possibilities are explored in seminars via small practice-oriented exercises. Suitable approaches to solving such problems (software technology) are therefore also an integral part of the course.

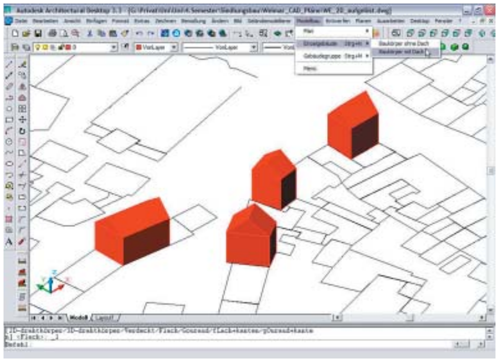
In the second part of the course, students examine either a given or a self-chosen CAAD-related task. The kind of tasks ranged from "digital (urban) modeling" to "topography modeling", "additional layout tools" as well as "from sketch to CAD-element."

The topics are specified (or chosen) only as general headings, in order that the students first examine the range of possibilities and potentials involved.

As a result of this process, the topic specifics and their position within the design process are defined in more detail. Based upon this, research (the research?) was conducted into existing IT-solutions as well as into planning requirements from the viewpoint of the architect. Research was also undertaken into solutions available in parallel disciplines e.g. mechanical engineering or shipbuilding.

The research and problem definition is then used to formulate a concept for the user – i.e. from the viewpoint of the architect. Selected aspects of the concept are then realized in a final phase as both user-interface prototype and small LISP-routines. Students are encouraged to make use of existing LISP routines available on the net, to integrate or adapt these where necessary and to write new LISP routines.

The following section shows student projects which illustrate this approach:



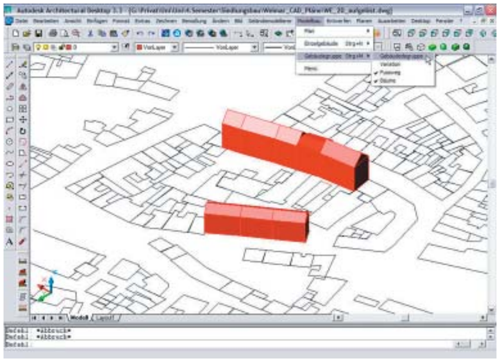


Figure 1. Generating of individual buildings, with and without roofs

Figure 2. Generating of building groups based upon variations of an individual building type

Example 1: CAUD: Computer Aided Urban Draw
(A. Genca, L. Rothstein & A. Schwarze)

The group chose to examine the modeling of urban environments.

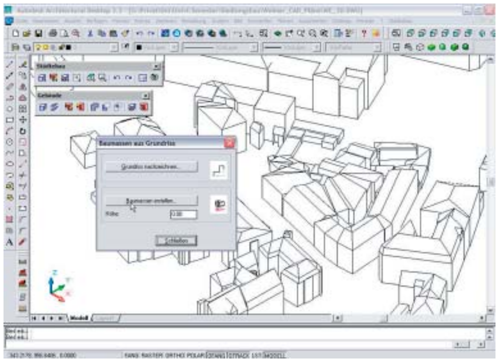
(defun c: programmieren stat(d)t sägen!()) - (defun c:city programmed not sawn!()) was both starting point and aim. Beginning with research into existing program systems as well as "classic modeling techniques" (i.e. in the workshop), a concept and prototypical implementation for the efficient production of abstract digital urban models was developed which uses basic pre-defined building envelopes.

The functionality covered three main areas: the import and scaling of a pixel-vector graphic as the basis for the model, the creation and positioning of individual pre-defined buildings and the creation and positioning of building constellations based upon the previous individual building blocks. The functions were developed and programmed by the students themselves.

Example 2: Digital Urban Model (F. Leibe, M. Reißig & C. Wambach)

The aim of this project was to develop a simple and quick tool to generate abstract urban models using the standard AutoCAD functions. Using a series of abstract building volumes, digital urban models can be quickly created to help visualize urban environments during the design process.

Through the simple modifications of the build-



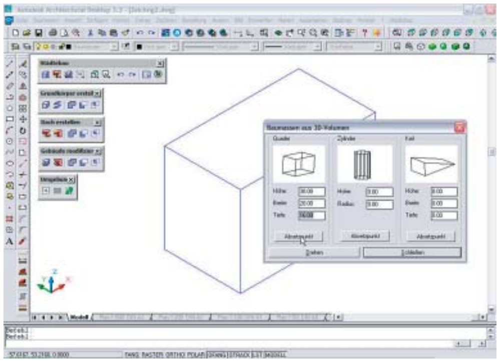
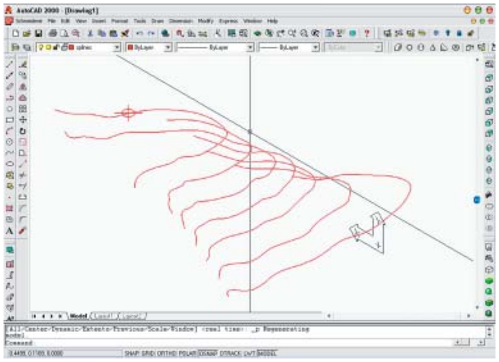
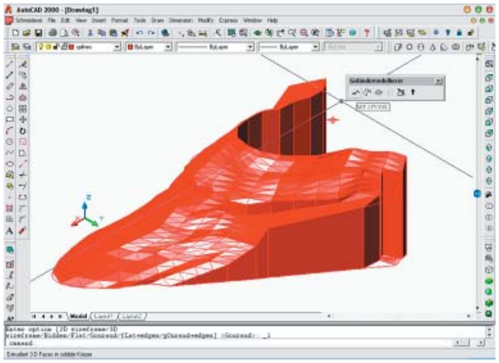


Figure 3. Generating of urban massing based upon 2D data

Figure 4. Generating of building groups using variants of individual building volumes



```
[All/Center/Dynamic/Extents/Previous/Scale/Window] <real time>: _p Regenerating
model.
Command:
```



```
Enter option [2D wireframe/3D
wireframe/Hidden/Flat/Gouraud/fLat+edges/gOuraud+edges] <Gouraud>: _l
Command:
```

Extrudiert 3D Faces in solide Körper

Figure 5. Generating of simple topography models based upon the contour lines

Figure 6. Generating a complex topography by means of meshs and volumetric objects

ing volumes, the user can quickly generate and examine the impact of different design alternatives. The tools combine existing AutoCAD functions such as extrusion, rendering and shading into combined urban modeling tools.

In addition to buildings, other urban 'objects' such as the generation of trees were also realized.

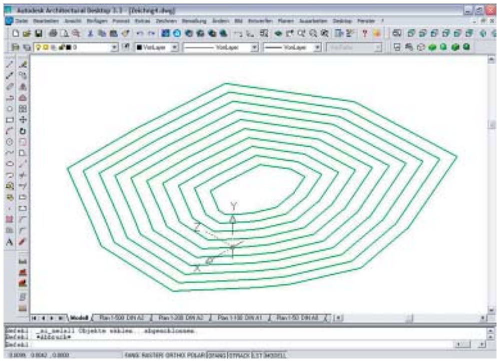
Example 3: VR2L LANDSCAPER (C. Quiatkowsk & G. Siarov)

The project group chose to analyze the generation of topography modeling. The context of a building is an essential aspect of a building and often generator of design decisions. Commercial CAD systems often support this only with extra tools, primarily for visualization purposes.

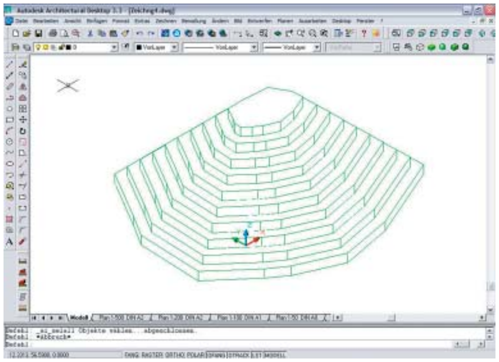
VR2L LANDSCAPER is a program module for the simple and rapid generation of virtual topographies. Basis for the model is either a scanned plan or measured 'point clouds' derived from a digital survey. Using defined functions, contour lines can be generated for use in layer models or topography nets.

The project group took a dual approach, restructuring existing tools (grouping of landscape relevant tools) on the one hand and conceiving of and implementing specialized landscape modeling functions on the other.

LISP routines obtained from the internet e.g. Net2Vol.lsp (Jesse, 2001) and Flatten.lsp (Middlebrook, 2001), were used, adapted and integrated within the application.



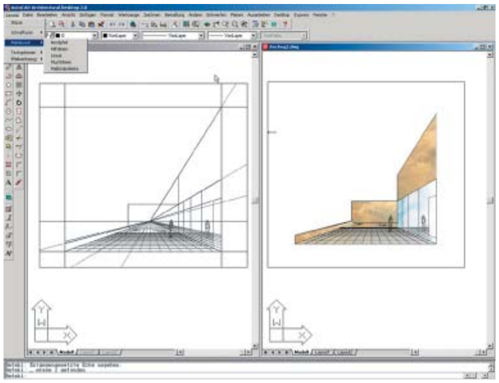
```
Befehl: _ai_selall Objekte wählen: abgeschlossen.
Befehl: *Abbruch*
Befehl:
```



```
Befehl: _ai_selall Objekte wählen: abgeschlossen.
Befehl: *Abbruch*
Befehl:
```

Figure 7. Semi-automated generation of contour lines

Figure 8. Generating of a volumetric topography model



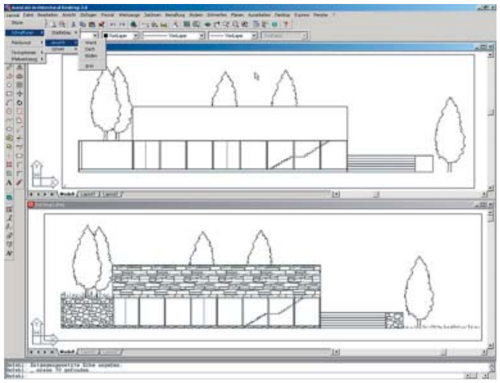


Figure 9. Vanishing point tool and sketch tool

Figure 10. Layout tools, grass and hatching

Example 4: Landformer (A.Knor, S. Maciolek & M. Häfner)

The project group also chose to examine the generation of simple topography models. The aim was to produce a simple model quickly and to make this available for further in editable form as a volumetric model. The focus lay on the user-system interactions. Based upon simple terms used in conventional workshop modeling techniques, the user need only enter limited information. The system is therefore simple, fast and intuitive. The implementation made use of existing LISP routines (Antkowia, 2004).

Example 5: Tools for 2D plan generation and layout (Nicolás Mármora)

The project examined more efficient ways of generating 2D plans and the support of page layouts. The task was defined by the practical needs of the architect and chose to provide additional functions to support more efficient plan layout generation. Internet research lead to the assessment and selection of appropriate existing LISP routines and their integration into an integrated concept.

Résumé and Findings

For architectural students

To conclude, the students learnt how to examine and structure a problem, to develop strategies and to orient their developments towards a defined aim. For the fruitful and effective use of the potential of CAAD systems, the students need similar skills as those required for solving architectural design problems: conceptual clarity and the openness to test aim and result against each other in a directed, iterative and exploratory way.

The students gain an understanding of the structure and functional principles of CAD techniques, and learn a healthy skepticism of the universal promises propagated by marketing companies.

Their experience of testing the boundaries of CAAD systems enables them to adapt such systems in future more to their needs, both in their further studies and future work environment and to enable them and others to work more efficiently.

For architectural teaching

CA(A)D is already an integral part of teaching, both in design courses as well as in specific coursework. The “straightforward” use of such systems is usually the main aspect of such teaching.

The enabling of students to be more than ‘mere’ users of CAD systems i.e. to become contributors or co-designers of such systems, whether as critical users of system-architects is often neglected. For this reason we believe it is important to offer courses which examine the role of computer science in architecture and examine the conceptual basis of such systems.

For some of the students, the experience was so valuable that they can imagine it being a possible specialization route for their later career. Some students have gone on to undertake more intensive courses in our department and the parallel department “computer science in engineering”. These include interactive architectural presentation using graphic-oriented programming environments such as Quest 3D as well as other individual diploma projects.

For research

Today’s IT systems are growing increasingly complex and the danger is that they distract the (less well versed) user from the actual task as a result of the multiplicity of functions available or that the user simply ignores that which appears too complex. There is a need to simplify user interfaces and to streamline the usability of complex program systems with regard to the needs of the user, the architect, and his or her specific requirements. The less well versed user and the student can and should be involved more actively in this research process, so that we do not simply embellish upon what has been established in 10 years of office practice. After all, it is today’s students who will become future users of such CA(A)D tools in architectural practice.

References

Amsoneit, W.: 2003, CAAD PRAXIS - Computer Aided Architectural Design auf der Basis der HOAI, Akademic Verlag, Köln.

Antkowiak, A.: 2000, TAILORS, http://www.a.tu-berlin.de/TAD/download/lisp/tailors: Jan 2005.

Davies, D.: 1997, Internet Spin Doctors Notebook - Biography of a Luminary Dr. Ivan E. Sutherland, http://www.cc.gatech.edu/classes/cs6751_97_fall/projects/abowd_team/ivan/ivan.html: Jan 2005.

Jesse, H.: 2001, net2vol - Extrusion von 3D-Netzen, http://www.a.tu-berlin.de/TAD/download/lisp/autolisp_download.shtml: Jan 2005.

Middlebrook, M.: 2001, Freeware AutoLISP programs, http://markcad.com./mm_programs.htm: Jan 2005.

Püntener, P.: 2000, CAD in der Praxis: Leitfaden für Architekten, Designer und Ingenieure [Leitfaden für die optimierte und programmübergreifende CAD-Arbeit], Verlag CadForum Architektur und Gestaltung, Basel.

Ramage, K.: 1999, AfraLisp, http://www.afralisp.com/books.htm: Jan 2005.

Schmitt, G.: 1996, Architektur mit dem Computer, Viehweg Verlag, Braunschweig.