28. XNET2 - METHODICAL DESIGN OF LOCAL AREA NETWORKS IN BUILDINGS

AN APPLICATION OF THE A4 INTELLIGENT DESIGN TOOL

Hartmut Ayrle

Institut für Industrielle Bauproduktion
Universität Karlsruhe
7500 Karlsruhe, Germany

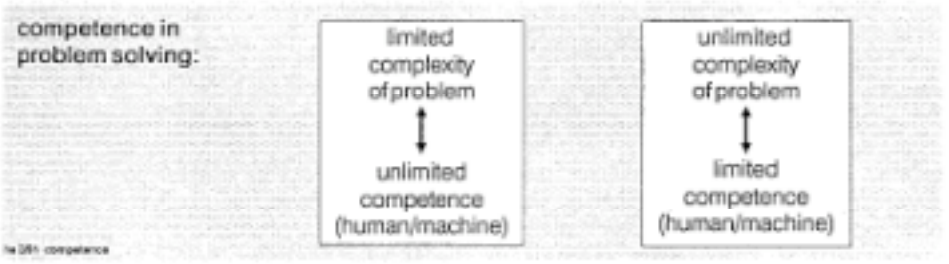
XNET2 is a prototype program, that helps network planners to design Ethernet-conform data-networks for sites and buildings. It is implemented as an example application of the ARMILLA4 Intelligent Design Tool under Knowledge Craft. It is based on a knowledge acquisition phase with experts from DECsite, the network-branch of DEC. The ARMILLA Design Tool is developed on the basis of Fritz Haller's ARMILLA, a set of geometrical and operational rules for the integration of technical ductwork into a buildings construction.

Our Perception of the Design Process

A discussion of intelligent software tools to support design processes must start with formulating the underlying assumptions on the design process. During the development of A4 we picked up the following ideas:

Limited Competence

In a „problem-space“ of limited complexity we can have unlimited competence in solving the problem - whereas in a problemspace of unlimited complexity, we can always only have limited competence in problem solving.



The complexity of a problemspace rises with the number of objects and the number of relations between the objects in the problem space. We think that real-world design problems tendencially contain unforeseeable many objects with unforeseeable relations. Thus we do not claim neither for a human nor an automatic planner something like total competence in solving design problems.
In more detail, we state that the design of a building is normally incomplete in three aspects:

- time
- representation
- functions and related design knowledge

Temporal Incompleteness

If we regard a building as a system during its total lifetime and wish to support conscious design of its construction and all the following changes, then the design process is not completed until the final dismantling of the building. The initial design is only a special case of the permanent redesign. To limit design to the initial construction means to deny the consequences initial design has for the operation and quality of a building during its lifetime.

Representational Incompleteness

As every discipline in the design process looks at, designs and operates a building from a different point of view, several worlds of symbolic descriptions of a building exist simultaneously. This statement is just another perception of the architect's coordination task and in our case of the design tool's coordination task of different models. In fact we are trying to establish a program that supports the handling and interaction of different descriptions of one identical building by different users.

As well, the representation of a building in analogous or digital data is principially incomplete. The representations we use are always only symbols for the real-world objects - only the building itself may be regarded as an analogous representation of a certain state in the design process. This leads to a new assessment of the contents of design documents. The statements they convey are always symbolic and thus may undergo more detailed consideration at some unforeseeable moment in the design process. A natural limitation of symbolic representation appears when the need for analogous physical distinction by using built models rises. CAD in this context is nothing more than an editor for the graphic parts of the symbolic representations of building models.

Incompleteness of Design Knowledge - Functional Incompleteness

Symbolic building models do not only contain descriptions of real-world objects but also knowledge about the appropriate way of designing and operating these objects. As there exist many differing descriptions of a building, the knowledge for design and operation is primarily distributed within the multiple descriptions and initially cannot be seen by every participating discipline. This is one source of design conflicts.

Furthermore, in building practice it is normal for planning agents a) to change their view of their design task during work, b) not to be integrated in the design team appropriately, or c) not to be integrated at all. This means that the multiple symbolic models of the building and the contained knowledge are changing due to functional redefinitions during the design process with unforeseeable effects. Again the design process is potentially incomplete, here as to the design and operation knowledge.

In summary, we see the design process as an effort to handle design conflicts that inevitably appear in a human, real-world context. A software tool to support design tasks must treat design conflicts as constitutive parts of the design process. The main service it may offer is to unreveal and handle design conflicts and only in second range to possibly resolve them. Thus, intelligent design tools, which may contain expert systems as automatic planners, have to be designed and used as assistants to the human planner. The final responsibility for design decisions will stay with the human as long as the competence encoded can only be limited.

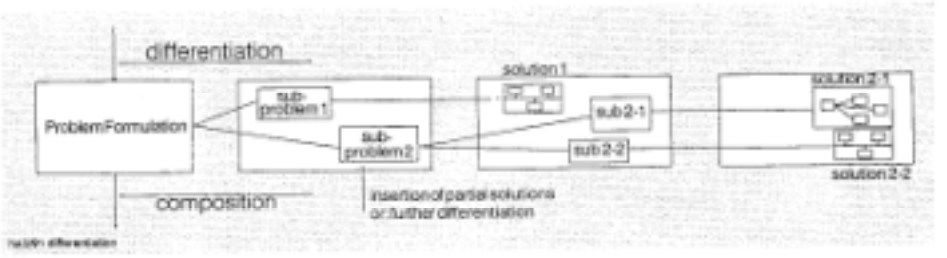
The Intelligent Design Tool A4

How do we handle these uncertainties in our "intelligent" design tool A4 ?

As it seems to be impossible to formulate a complete description of the manifold knowledge to be taken into account during the design process, we hand the formulation of semantic knowledge back to the different users, and focus on the possibility of coordinating them through a common syntax for the design and operating process. Only on such a platform can semantic knowledge be usefully formulated.

Design Methodology

We propose a common design methodology for all involved disciplines, that initially contains no knowledge about objects or procedures of the design task but is itself an aid to dynamically establish and handle symbolic representations of objects and procedures in multiple planning modules. It transfers the principles known from Methodical Construction in Mechanical Engineering to the field of Architecture and accomplishes the well-known incremental functional decomposition with the possibility to formulate constraints and constructive knowledge for every decomposition phase/module.

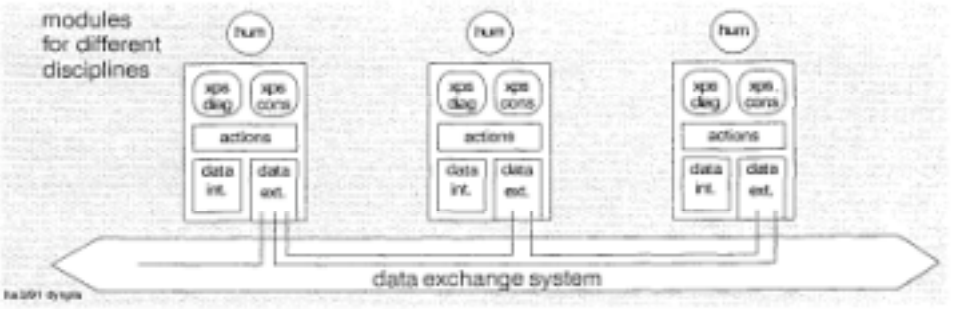


A planning agent starts to build a planning module with a graphic sketch of what he wants to plan - that is what he wants to define more precisely - on an arbitrary scale of detail. He gives name, space and time for the problem he formulates, be it an object or process to be designed or operated. Then he starts detailing the problem. Every detail he gives can again be another problem to be solved later, or it can be a partial solution, that he creates himself or that is introduced from a prototype solution catalogue by him or by a planning automaton. The problem he started with is solved, when all details are decomposed so far that a solution for them can be found. Having solved a problem he can incorporate it into a larger problem formulation module, where it in turn may serve as partial solution.

Dynamic Planning Model

During an initial decomposition process, the planning agent builds a series of program modules which represent his way of solving/decomposing the specific design task. Each module comprises the objects and knowledge for one decomposition step. One agent's modules form together his planning world and exist in parallel to other agent's modules.

As different planning agents build up their symbolic representations in a series of planning modules, we allow them to "publish" those parts, that may be relevant for other planners. Each agent can decide to declare "interests" that he has on other agent's objects or that other agents should have on some of his objects. That way a common planning world is gradually established alongside the multiple specialized planning worlds. The different planning agents will then use objects and procedures of multiple interest from the common planning world in their own process of detailing. Thus a network emerges, that reveals the dependencies of the different disciplines through the multiple use of objects or processes in the differing planning worlds. If one planning agent A changes the internal structure of an object or process, this change is passed by a data-exchange-system (rather than an inflexible database-system) directly to all the planners B, C etc. who use this element in their decompositions.



Conflicts may then be resolved in one of the following ways:

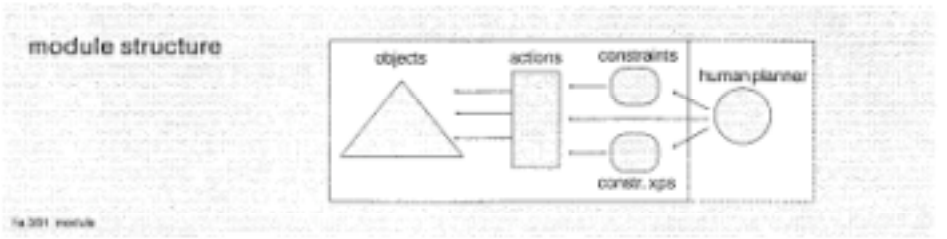
- The program notifies other users of the changed element, that a change occurred which might possibly cause a conflict; some notified users state no conflict with their current design and accept the change; some other users do state a conflict and try to resolve it by re-

design of their internal plannings; if this is not possible, the originator is notified and the conflict has to be resolved either on the level of analogous face-to-face discussion or by means of a decision hierarchy.

This means that our program initially fulfills the task of coordinating different disciplines by bringing conflicts to the surface rather than trying to resolve them with principially incomplete design knowledge.

Design Constraints and Automatons

As we have seen, our program A4 initially only offers two automatisms: a) the operational support for the decomposition of a problem which keeps track of multiple uses of elements and b) the notification of planners when conflicting designs become possible. So far, our system does not deserve the name of "intelligent" tool yet. How do we encorporate the human planner's knowledge into our system?
Our A4 system thus offers possibilities: a) to define constraints on the modification of object attributes that have to be fulfilled within a specific design subtask/module b) to define typical (partial) solutions for design subtasks in form of prototype objects that represent elements which need no further detailing/decomposition c) to define knowledge of constructing solutions by finding and inserting partial solutions.



In our system, design knowledge is always appended to a certain design phase/module. In every design phase a planner works on, he can use the constraints, strategies or partial solutions that are formulated as relevant for it. He may examine this knowledge, lock its usage or add new knowledge through different appropriate editors, textual or graphical. Thus he has the choice between three work-modes with our design tool: a) pure manual design of building elements, with conflicts being shown for all elements, that he publishes to other planners; b) manual design controlled by the encoded constraints for the current design phase;
c) automatic design driven by encoded strategic knowledge and prototype solutions.

Distinct series of design phases, that predefine appropriate levels of detailing/ decomposition for a certain design task, are defined by a user's initial formulation of a design subtask. They as well represent some knowledge of the design task, especially on a promissing decomposition strategy. Nevertheless, the modules representing such a predefined process can be altered at any time for specific projects.

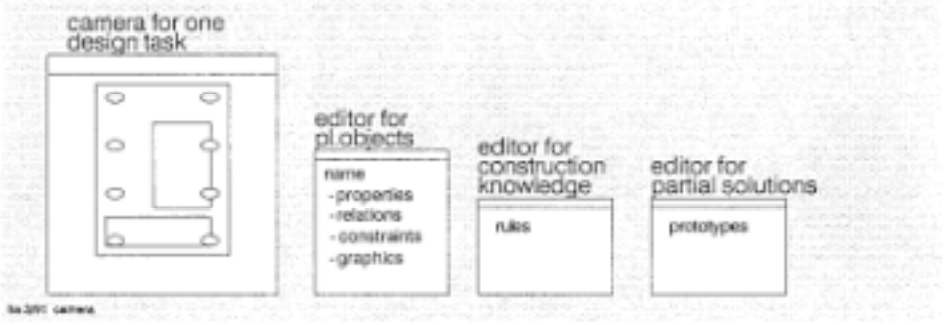
The representation of such knowledge in a software system has to be considered under software engineering aspects and I cannot discuss this subject here. It may be said however, that our experience in implementation points towards complex multi-language software development-tools. They make it possible to represent different types of knowledge in different, appropriate languages - using only one language (LisP, Prologue, OPS etc.) brings too many problems in adopting the manifold ways of real-world reasoning to one single method of evaluation. We use the KnowledgeCraft development shell with an X-Windows based graphic interface builder.

The Role of the Graphic Interface

The graphic interface of our program is of special interest for us, first because we concentrate on a methodology for design, which produces questions on the realization of the methods. Second because we believe, that expert systems are only usable to the degree in which they are controlable. To give control over the software's influences on the design process, it is necessary to "show" what exists and what happens. As "a picture says more than a thousand words", we suggest that every object or operation to be planned, every design-process to be represented and every kind of knowledge to be incorporated in the program is best described and can best be investigated in graphic form. Of course there are things that are most efficiently described in words or formulas, but our preference lies towards graphic expressions.

Subsequently our program offers graphic and textual editors for several uses:

a) a CAD-like editor for graphic input and detailing of objects and graphic representations of processes;
b) a graphic editor for the relation network between the elements in the planning world;
c) an alphanumeric editor for textual definition of planning world elements or design knowledge;
d) a graphic editor for design knowledge is under development.



An Application Example: LAN-Design with XNET2

To give a more concrete picture of our program, I want to finish by giving you an impression of an expert system we are about to implement. In cooperation with Digital Equipment Germany we build a design tool to support planning of Local Area Networks (LAN) for buildings under the name of XNET2.

The software is based on a thorough knowledge aquisition phase (about 1.5 man-years) which revealed important parts of the design knowledge of DECsite network planners. It uses the concepts of the A4 kernel explained earlier. The system predefines 4 design phases and 4 detailing levels for the LAN-planning process.

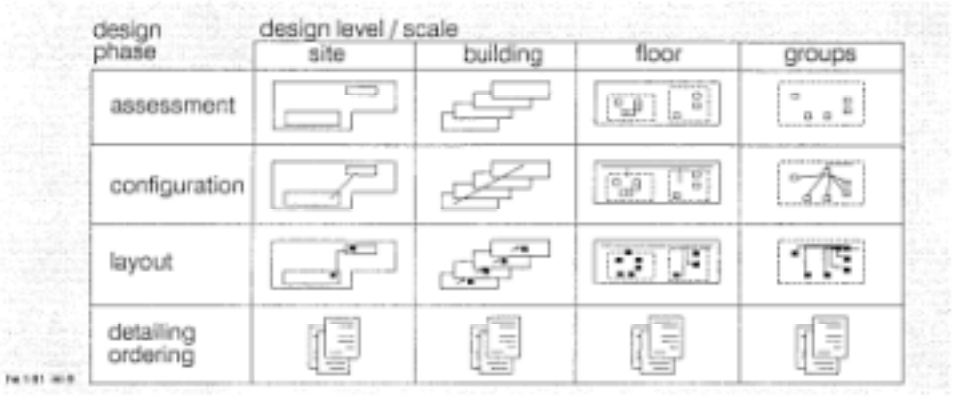
The phases of the LAN design process are:

a) an initial assessment of the site, its buildings and duct-possibilities and the existing and intended computer hardware installations to be connected;

b) the production of configuration alternatives that work on rough spacial and technical connection possibilities;

c) the accurate layout planning, that takes into account the exact measurements of existing rooms, possible ducts and permissible cable-lengths;

d) the definition of details for duct installation and the production of ordering lists.



In each of these phases decision have to be taken on four levels of detailing: i) considering the whole site; ii) considering each building as a whole; iii) considering each floor in a building; and iv) considering groups of computers within one floor.

Our program offers 16 cameras on the planning world that represent those differentiation tasks. Appended to every camera is the constraint and strategic knowledge that we - in this case as A4 users - extracted from the DECsite experts and encoded into the program.

The user may start to build up the project-specific planning world at any phase and any level - by giving graphic descriptions of the buildings or computers or duct solutions or by giving textual descriptions of the hardware to be connected or new constraints to be considered. He is offered catalogues of typical hardware and cables from where he derives individual objects for the planning world. Thus he can manually produce a network concept for his project. At all planning levels, the user can call up a control routine, that checks his decisions against the available constraints and marks obviously wrong situations.

We also offer a planning automaton, that is based on the same incremental decomposition concept the whole system is designed for: The user may mark a region for which the program is to produce solution proposals. The software then starts to decompose: a) It tries to define groups of computers within the given region, that are obviously easy to connect. In order to do so it checks the technical connectability (port and cable types) and the proximity relations (all within one room, closer than n meters, depending on the cable type, etc.) and formulates

a hierarchy of possible groupings. b) Then it starts to evaluate the existing prototype solutions for the current design phase and level and gives either possible alternative solutions or hands back the problem to the user as currently unsolvable.

The last statement must be understood correctly: the problem is currently unsolvable to the software, given the present state of the planning world and design knowledge. As soon as the user gives new informations or changes the planning world in some way, the same automatic process may be used again and may find a solution.

It is important to note, that we use collections of prototype solutions to select from rather than instructions for the generation of solutions. This takes more effort in evaluation but is necessary as we have encoded pure experience knowledge that up to now seems not to be expressable in generative form.

A generative method can be used when, during the layout planning phase, the system automatically tries to assign the roughly configurated cable-lines to existing or possible duct spaces. Here we use a modified Lee-algorithm to find possible cable and duct paths.

As XNET2 is completely based on the A4 system, it can be incorporated into a larger design tool, that holds modules for other design disciplines.

At our institute we run for example another expert system that supports planning and detailing of waste water duct systems in a building and shares parts of the symbolic building description with XNET2.

It is easy to think of a use of XNET2 for the maintenance of installed local area networks. As initial design is only a special case of redesign, most relevant data and knowledge for maintenance exists already and only few additional functions would have to be added, as for example timestamps for inspection intervals and test messages for hardware function checks.

Outlook

As we have seen, the A4 intelligent design tool models the design process as a decomposition process and supports the designer with a graphical interface and with means for constraints and constructive knowledge.

The critical point with the decomposition paradigm is however the process of decomposing a problem in exactly such a way, that partial solutions for sub-problems will be found. Expert excellence is the art of decomposing "correctly" and not towards a situation, where no solutions can be introduced. As pointed out in the XNET2 example, we are about to do more investigations in this field with the aim of producing an "automatic decomposition" support for A4 users.

References

A.Drach,L.Hovestadt, "Intelligente CAD-Systeme - Instrumente für die Planung und Verwaltung komplexer Gebäude", Proc. of the Intelligent Building Symposium 1989, Institut f. Industrielle Bauproduktion, Uni Karlsruhe

Peter Raetz, "XNET: ein intelligentes CAD-System für die Planung von lokalen Netzwerken in Gebäuden", VDI Verlag Fortschrittsberichte 20/19, 1989

VDI-Richtlinien zur Konstruktionsmethodik, Nr.2210-2217

D.T. Ross, "Structured analysis - a language for communicating ideas", IEEE Software Engineering 1/77

Marvin Minsky, "The Society of Mind", Simon & Schuster New York, 1985