SCEMA

Scema, P.O. Box 59, SF-02601 Espoo, Finland. Visiting Address: Upseerinkatu 3 A. Phone +358-0-513 022, telefax +358-0-513 544

Knowledge-Based CAD

Since their introduction in the eighties, Computer-Aided Drafting (CAD) systems have become ubiquitous among design engineers. From costly systems with simple interactive geometric capabilities, CAD-systems have evolved into affordable, but sophisticated tools which automate many aspects of design documentation and retrieval. The development of even routine designs, however, has still been a manual - and often tedious - process. These first generation CAD tools have provided gains in drafting productivity of 50% to 200% in repetitive design applications.

This article describes the history and evolution of a new kind of knowledge-based CAD system, Design^{++}. By providing significant new capabilities to support both the development and documentation of design solutions, Design^{++} increases productivity in design projects by ten to fifteen times, freeing engineers to make fuller use of their skills and creativity.

Design^{++} allows storage and re-use of engineering knowledge, supports engineering teamwork and concurrent engineering, enables "what-if" experimentation, integrates engineering disciplines, and automates generation of documents, including drawings, parts lists and specifications.

Keywords: Knowledge-Based CAD, Intelligent CAD, Knowledge-Based Engineering System, Engineering Expert System, Design Automation.

INTRODUCTION

Common to most engineering tasks is the need to effectively retrieve data, perform analysis and synthesis based on the data and store the results for later use. The results of these operations are eventually documented as drawings, bills of materials, manufacturing instructions, technical documents and many kinds of reports.

The engineering analysis and synthesis process incorporates a great deal of know-how which we call expertise. The key to successful engineering automation - versus drafting - is the ability to store and utilize engineering expertise in a computer.

Traditional CAD systems provide some means to express engineering knowledge, like:

- ▲ parametric programs
- ▲ graphical symbols
- ▲ existing drawings and geometric models

In addition, some domain knowledge has been implemented by CAD software vendors in application packages.

Although computer aided tools provide great potential for automating engineering design, traditional CAD systems have brought major productivity improvements mainly to drafting work in the detail design phase. They have provided only limited support for conceptual and preliminary design.

This article describes the development of ideas and their implementation for an object-oriented, knowledge-based engineering design software product, which provides tools for automating not only detail design, but also conceptual and preliminary design phases.

BRIEF HISTORY

The history of the Design^{++} system began in 1985 at Nokia Information Systems in Finland. The CAD/CAM and Knowledge-based Systems departments started to look for a joint project in mid-1985. After evaluating the potential of several pilot projects among a few large Finnish customers, Tampella Power Industry and Nokia discovered a mutual interest in developing a prototype of a knowledge-based engineering design-support tool.

The project began in February 1986. By February 1987, the project team had developed two prototypes and delivered a working version of a system to support the design and manufacturing of boilers to Tampella's engineering department.

In September 1988 three key people from Nokia's project team left Nokia and founded an independent company, Scema Oy, to commercialize the technology. Scema's first pro-duct, running on a Unix workstation, was announced in April 1989. By 1990, Scema had been applied to a number of other engineered products, including paper machines, paper coating machines, telecommunication networks, cable factories, etc., and about 10 licenses had been sold.

The successful project triggered considerable international interest. This led to the founding of Design Power, Inc. in California in July 1989. In January 1990 Design Power, Inc. (DPI) took over the responsibility for commercializing and marketing the product of this Finnish research and development effort worldwide, except Scandinavia, which is covered by Scema Oy. The product is called Design^{++}.

THE DRIVING FORCES FOR KNOWLEDGE-BASED CAD

While traditional CAD systems focus on automating drafting, i.e., drawing production, the focus of knowledge-based CAD is to utilize unified object models to store and re-use engineering knowledge (Levitt, 1990).

Design synthesis for many engineered products involves selecting, connecting and locating standard or parametric components to form the subsystems and systems of a complex product.

Today, engineers do much repetitive work when designing customized solutions. They must draft each drawing manually or with CAD, execute the same kinds of calculations and select appropriate components for each project. By using knowledge-based systems, engineers can describe all the necessary calculations, the bases for selecting components and the underlying relationships determining layout in product knowledge-bases, called "intelligent" libraries.

Supported by knowledge-based systems, engineers can better utilize knowledge of the product, its design process, manufacturing and operating requirements than ever before. With utilizing knowledge-based engineering systems it is possible to create, maintain and share product know-how, which is the main task of engineers. As a result, new design projects can be carried out rapidly and with less effort. In the words of the R&D manager of Tampella Power Industry, "Routine engineering is automated; engineers now serve an R&D role in our company" (Riitahuhta, 1990)

The objective of a knowledge-based CAD system is to change focus from automating the drafting to automating the broad decision making and the repetitive engineering tasks in the actual design process (Dym and Levitt, 1991).

The challenge is: what is the most successful way to implement such a system for the mass market ?

The approach taken in Design^{++} is to integrate an object-oriented, knowledge-based design system Design^{++} with existing commercial CAD systems, SQL-based relational databases and with desktop publishing software.

IMPLEMENTATION

Initial Ideas

Design can be viewed as an iterative series of "synthesize, analyze and evaluate" cycles (Dym and Levitt, 1991). The initial approach that was explored in the Tampella prototype system was to use the system as an intelligent design critic, which would analyze and evaluate the design decisions synthesized by the engineers, compare those decisions to the stored knowledge and advise if there would be conflicts.

An experimental system was developed in five months to monitor and advise in the placement of the auxiliary components of a boiler furnace, like sootblowers (Figure 1). Feedback from Tampella's engineers implied emphasis on synthesis rather than analysis. Their opinion was that instead of using the knowledge-based technology to monitor what the engineers were doing wrong it would provide more value by assisting in the repetitive decision making.

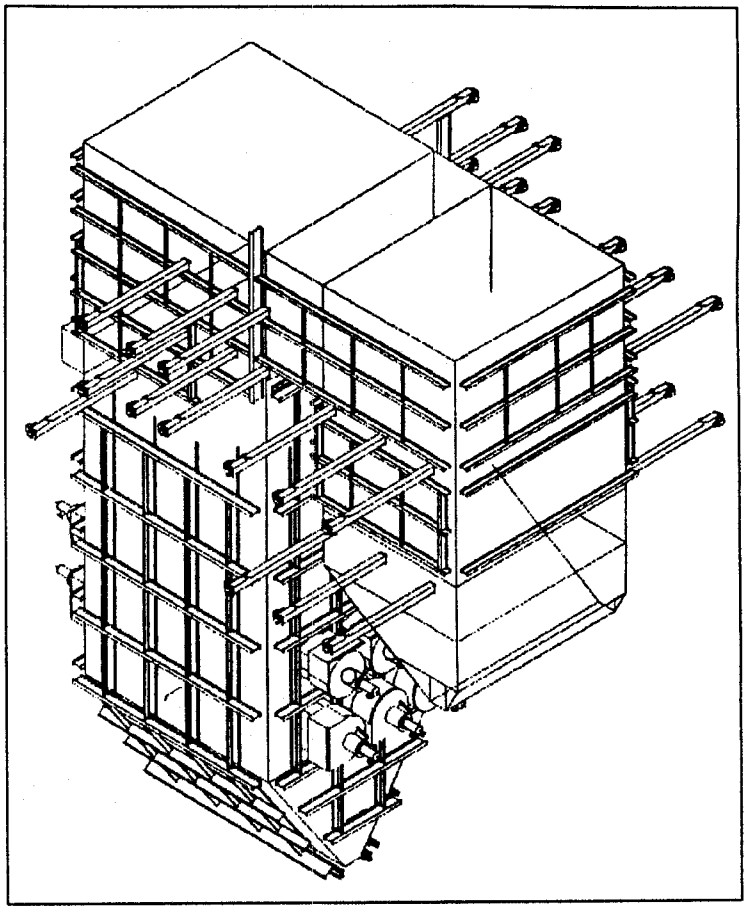


Figure 1: Auxiliary components of the boiler furnace.

Two additional months was spent to specify the second prototype, which proved to be successful implementation. The essential change in thinking was that the system was allowed to synthesize instead of analyze and evaluate. In other words, the system was instructed to incrementally build up design solutions. This approach put the evaluation task back in the engineers' hands. Engineers became able to evaluate different design alternatives quickly and choose from them.

Basic Concepts

The object system of Design^{++} is based on an object-oriented frame-based tool, KEE. Frame-based representation is very suitable for describing physical as well as abstract entities. A frame is a computer data structure which can contain an arbitrary number of attributes describing both static and dynamic properties of a "thing", e.g., a valve (Figure 2).

Valve

ATTRIBUTE	VALUE CLASS	VALUE
Type	Symbol	**Ball_valve**
Actuator_type	One.of: *Pneumatic Manual Electric*	**Pneumatic**
Body_material	Symbol	Unknown
Connected_to	List	Unknown
Flowrate	Real	Unknown
Fluid	Symbol	Unknown
Line_number	Integer	Unknown
Nominal_size	Real	Unknown
Pressure	Real	**500.0**

Figure 2: Frame representation of the properties of a valve.

Each attribute in a frame can contain additional properties (facets), such as its default value, value class and a comment. Value class provides a mechanism to control that the data types and ranges of attribute values are legal. In addition to static properties, objects can contain dynamic properties, which enable to incorporate engineering expressions in objects. Engineering expressions can be arithmetic, algebraic, standards based or case based procedures, which determine how the objects are selected, connected, located and sized. These design rules, which are attached to attributes as facets (Figure 3), are automatically executed whenever the value of an attribute is requested, if the value is unknown.

When the value of the valve's *body_material* attribute is requested, either by the user, or by a design rule, the system discovers that its value is unknown, and that it has a rule associated with it. This causes the rule to "fire". The rule contains a reference to the *fluid* attribute of the same object. The system discovers that the *fluid*'s value is unknown and that it has a rule attached. This rule then fires. It contains a reference to the parent component of the valve. The system finds the parent object and requests the value of its *service* attribute. That value, *black_liquor*, provides the value of the valve's *fluid*. Finally, the value of *fluid* is returned to the rule of *body_material* and that rule determines that the appropriate material is 316SS (a type of stainless steel valve material).

The system creates dependency links among related attributes dynamically during the execution of its rules, e.g., from the parent component to *fluid* and from *fluid* to *body_material*. These dependencies are then stored in corresponding attributes. Similarly, the references from rules to other attributes, i.e., the rule "arguments", are determined dynamically during rule execution and stored in corresponding attributes. This is how a valve "knows" that its *body_material* depends on its parent's service. This information is called a dependency network.

Valve

ATTRIBUTE	FACETS	VALUE
Type	**Value.class:** Symbol	**Ball_valve**
Actuator_type	**Value.class:** One.of: Pneumatic Manual Electric	**Pneumatic**
Body_material	**Value.class:** Symbol **Design.rule:** (case (:? My Fluid) (Black-liquor '316SS) (Water 'Bronze))	**316SS**
Connected_to	**Value.class:** List	Unknown
Flowrate	**Value.class:** Real	Unknown
Fluid	**Value.class:** Symbol **Design.rule:** (:My-parent Service)	**Black_liquor**
Line_number	**Value.class:** Integer	Unknown
Nominal_size	**Value.class:** Real	Unknown
Pressure	**Value.class:** Real	**500.0**

Figure 3: Frame representation of a valve with design rules.

The dependency network provides the means to maintain design consistency when modifications to the models are made. If the parent component of the valve is changed, the value of the valve's *fluid* becomes obsolete and is set back to unknown. This causes the value of the valve's *body_material* to become obsolete and also be set to unknown. This automatic removal propagation resulting from design changes ensures that a design stays consistent.

System Architecture

The objects in Design^{++} are called components and they are stored in libraries. They are the warehouse where components are created and stored. Libraries can contain both domain and product knowledge. In libraries the components are presented in a "is-a" hie-rarchy, which allows to inherit properties from higher level abstract component descriptions to more specific components. The benefit of this arrangement is that the information is maintained only in one place and the inheritance mechanism takes care to distribute the information down the "is-a" hierarchy, i.e., class-subclass hierarchy.

The use of a frame-based object representation for storing the object-level knowledge and data is straightforward. Other concepts, however, are needed to illustrate the function of components and their interrelationships. The concept of product structure was adopted from the mechanical engineering world. In a product structure, the components are arranged in a tree hierarchy in which more detailed components reside at lower levels and more abstract components at higher levels. For example, the component *boiler_plant* could be at the highest level and a *beam* or *pipe* could be at the lowest level.

The components in a product structure represent a specific kind of engineered project. Once the components have been exported from libra-ries they become project dependent and they are

called instances of the library components. The library components, which are called classes, are project-independent, while the project dependent components, instances, are stored in product models (Figure 4).

Skeletal product structures describe aggregations, how something consists of something else. For example, a pre-heating section consists of two headers and a variable number of tubes. This could be described in a skeletal product structure file in the following manner:

```
(Pre_heating_section
      (Upper_header)
      (Lower_header)
      (:N
       Pre_heating_tube))
```

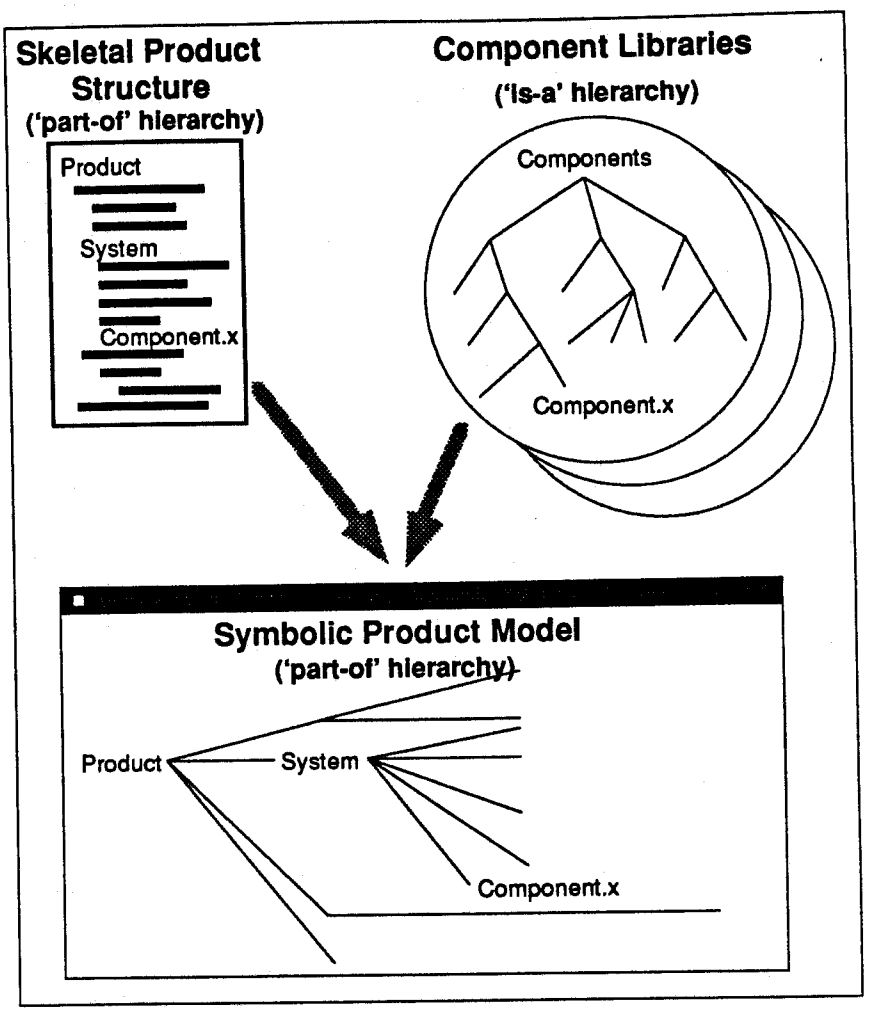


Figure 4: Knowledge organization.

A project-independent rule can be attached to the *Pre_heating_section*, which allows it to determine the appropriate number of *Pre_heating_tubes* in each product model.

Interfaces

In order to be acceptable in a production engineering environment, a knowledge-based engineering system needs to integrate seamlessly with other software tools, like CAD systems and data bases (Figure 5).

Design^{++} libraries contain two categories of components; parts and assemblies. The difference is that parts have geometric properties and assemblies do not.

Integration with a CAD system requires that parts in the symbolic product model have geometric attributes, which are used to parametrically describe the component geometry as well as to define its location and rotation. These attributes are automatically inherited from a system-level geometry library where the geometric types of components are defined. One of the inherited attributes contains a rule which communicates with CAD system by sending the geometric description of the component. The result can be seen in the geometric window of the CAD system.

Another important aspect in CAD integration is the mapping between the symbolic model and the geometric model. Mapping is based on identifiers, which are stored both in symbolic components and in geometric entities. These identifiers are cross-wired and the CAD system is thus integrated by a bi-directional link, which enables user entries in both graphic and symbolic models.

Graphical schematic diagrams, such as process and instrumentation diagrams (P&ID's), are widely used to represent topological knowledge (how components are connected to one another) in a manner easily understood by engineers. The bi-directional CAD - knowledge-base link allows schema-tics to be used for both input and output of topological information in Design^{++}.

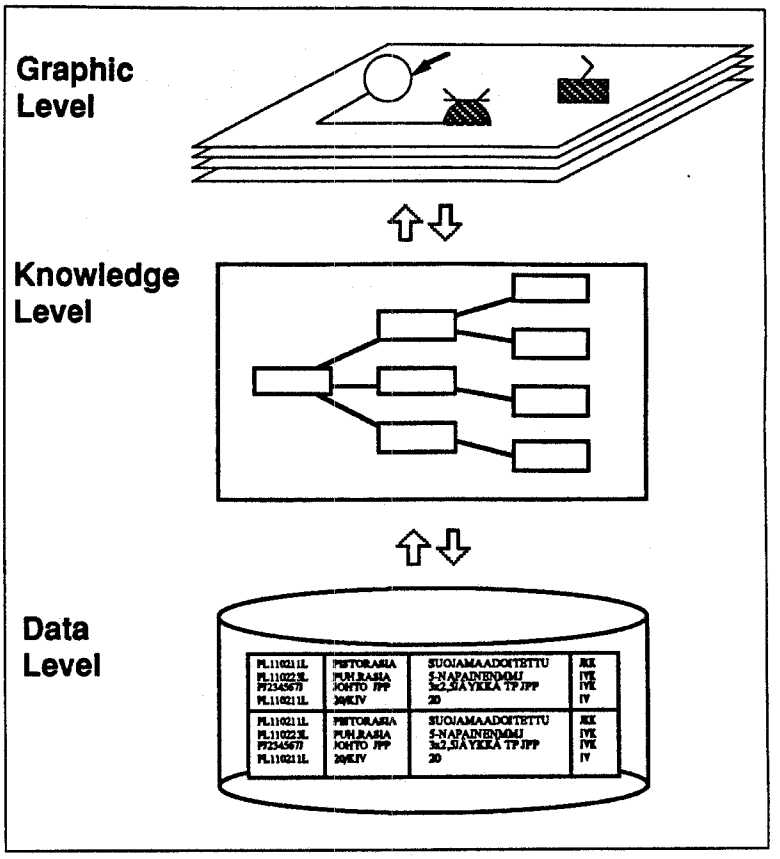


Figure 5: Systems integration.

Relational data bases are also essential to integrate in an automated engineering environment, because utilization of SQL-based databases in engineering companies is becoming increasingly popular. Databases are used to store standards, such as AISC (DIN) steel data and ASME (DIN) piping data. Also manufacturers' product catalogs are beginning to appear in the market in electronic format, which can easily be converted to relational data bases.

The design rule language in Design^{++} provides functions, which can make SQL-queries to databases. For example, the diameter of a connection tube of a pre-heating section of a boiler furnace is calculated by the following design rule:

```
2 * (Sqrt ((:? Boiler_plant Steam_capacity) /
           (:? My Flow_speed) /
           (Rdb-sql "Select Distinct Value from Physical_factors where
            Key = Water_density" 'STD)))
```

The *steam_capacity* is retrieved from the *boiler_plant* object and *water_density* from the table *physical_factors* in a relational database. If the steam capacity changes later on, the system automatically maintains consistency through the model's dependency network and re-determines the tube's diameter.

EARLY APPLICATIONS

Tampella Boiler Design

The first Design^{++} system was delivered to Tampella Power Industry in April 1987. The system was integrated with Tampella's Computervision CAD systems and a VAX-based engineering analysis system. Today, Tampella is using five workstations equipped with Design^{++} and the main application is the 3D layout of boiler plants.

The process for building a knowledge-based application usually starts by identifying those modules or subsystems which are critical for the design process and which can increase productivity when their design is automated, for example, the pre-heating section of a boiler plant.

The first task is to create a library of components which describe the components used in the module. In our example, the pre-heating section consists of two headers and a variable number of tubes connecting them (Figure 6).

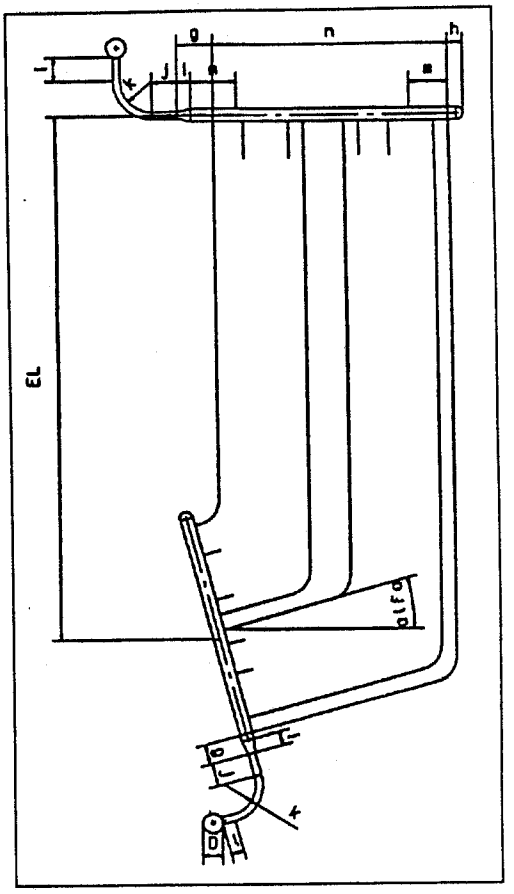


Figure 6: Pre-heating section of a boiler.

Next step after creating the library component, is to describe its attributes (Figure 7). Domain or product knowledge can be expressed as default values and design rules. The system can use several alternatives when trying to determine a value of an attribute during the design process. The sequence by which Design^{++} is trying to determine attribute values is following; 1) local value, 2) design rule, 3) inherited (or default) value, 4) external data file and finally if non of above was able to give the value, it is asked from the user.

The likely sequence for determining the values of the attributes in figure 7 is following:

- ▲ *Header_bending_radius* (k) is a constant in all boilers.
- ▲ *Nominal_length* (EL) of the pre-heating section will be retrieved from an external data source, which contains results of the proprietary analysis programs.
- ▲ The value of *Nominal_width* is needed when the design rule of *Lifting_lug_increment* is executed; the value will be retrieved from an external data source and if not found will asked from the user.
- ▲ *Lifting_lug_increment* (m) has a design rule m = 0.2071 * n, where n is the nominal width of the pre-heating section.
- ▲ *Isolation_thickness* is asked from the user.

When knowledge is documented in library components, the maintenance of engineering knowledge becomes more organized and easier. The system organizes components, their attributes and

PRE_HEATING_SECTION

ATTRIBUTE	FACETS	VALUE
Header_bending_radius		300.00
Nominal_length		Unknown
Nominal_width		Unknown
Lifting_lug_increment	**Design_Rule:** (0.2071 * (:? My Nominal_width))	Unknown
Isolation_thickness		Unknown

Figure 7: A library component representing the pre-heating section of a boiler.

design rules and assists in the maintenance of the information in various ways. For example, if the rule for the lug increment changes, the existing value of that attribute in active design models will automatically be removed (the attribute value becomes unknown). When the values is next time requested, it is calculated using the new rule.

When the modeling of the pre-heating section is complete, which takes only about two days, the knowledge-based system is ready be used to automatically generate design solutions for this module in all new boilers. The engineers in charge of this module then become responsible for describing and maintaining their knowledge in the system. In this way, engineering work becomes more like product R&D. What used to be routine design has been automated with productivity gains of over an order of magnitude in Tampella's case.

The layout design problem is solved by developing layout rules for the plant sections and their individual components (Pernu, 1988). For example, the wall columns of a recovery boiler building (Figure 8) are the outermost columns around the plant. In the lateral direction of the boiler, the distance of the columns from the plant centerline is determined by the furnace width and the length of the sootblowers and service platforms, if any. The service platforms are of standard width, but the length of the sootblowers depends on the furnace dimensions and the furnace dimensions depend

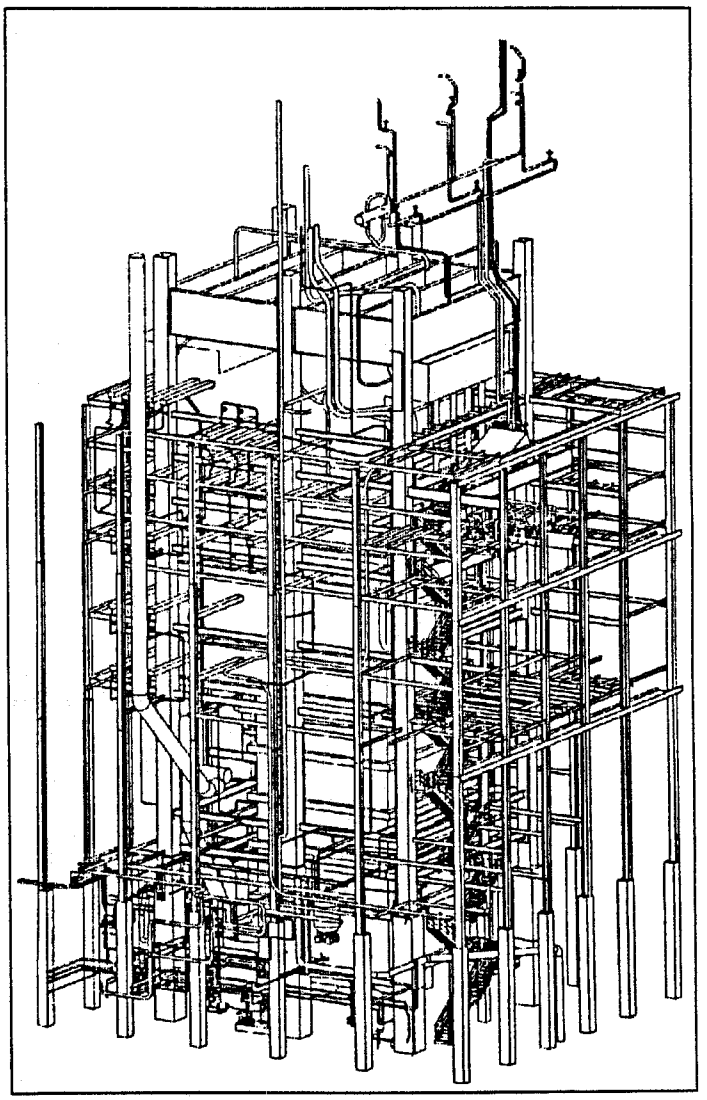


Figure 8: Boiler plant.

on the quantity of dry solids in the black liquor.

The column distance is obtained as the sum of the furnace width, sootblower length and service platform width. Rules for furnace width and sootblower length are obtained in a similar way.

If the properties of the black liquor are changed, a new solution is obtained immediately. The idea is that the rules are established for each component, and the furnace "knows" its own width, the sootblower "knows" its own length and service platforms "know" their own widths. The column requests the required data from each component and calculates its position in the layout. If some component can not answer, i.e., the attribute in question has no design rule or the value cannot be obtained any other way, the system directs the request to the designer.

An example of consistency control during design changes is a change of the density of dry solids in the liquor. As a consequence of this change, the attribute values which are dependent on it, i.e. furnace dimensions, sootblower length and column position, will be removed and recomputed. In the interest of efficiency, all calculations are based on demand-driven realization, so that the removed values are not automatically recalculated until some other object or the user requires it.

Air duct design

The air ducts of a recovery boiler in different plants differ so much that it was found be impossible to carry out the task using parametric CAD software.

Tampella made a comparison of the design of air ducts (Figure 9) with conventional methods and with the knowledge-based system (Riitahuhta 1988). The examination covered the project design group leader's work, process design, layout drawings and detail drawings for manufacture and installation.

The study shows that high productivity has been achieved with Design^{++}. The design time of the ductwork has decreased by 75%, 5 times fewer drawing versions have been produced and the design has been completed 4.5 months earlier.

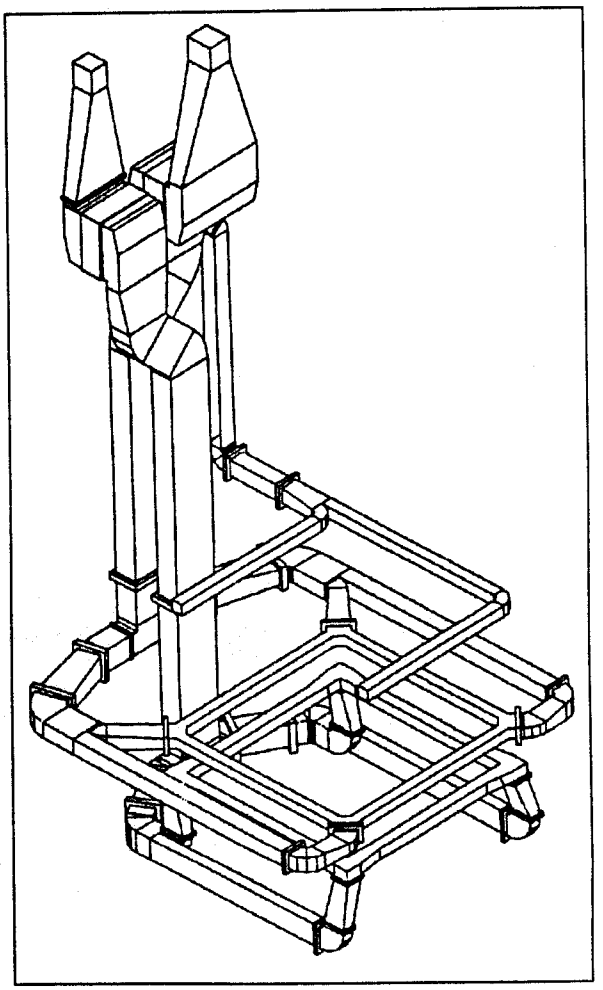


Figure 9: Boiler ductwork.

Pipework design

Another application study made by Tampella was the design of upper circulation piping (Figure 10). Here the upper circulation pipes refer to the water collecting pipes of the boiler furnace walls which return the boiler water to the drum.

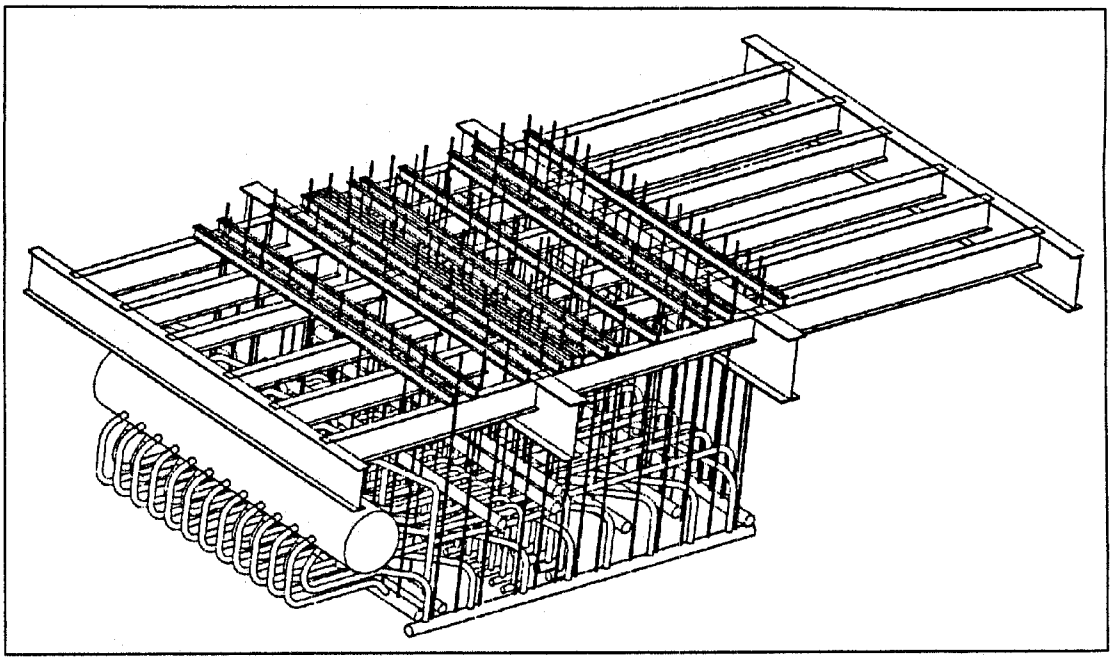


Figure 10: The upper circulation piping.

Modeling is divided into three stages:

- ▲ defining the positions of the drum connections
- ▲ defining the positions of the connections of the wall collecting headers
- ▲ defining the pipelines from the drum connections to the corresponding connections of the wall collecting headers

The layout of the drum connections is determined by standard construction criteria, but the number and positions of the connections are determined separately for each project. The positions of the connections are determined by the obstacle areas formed by the downcomers.

There is a standard procedure for the layout of the wall connections. Solutions for a particular project are provided by the different obstacle areas caused by boiler sizes. Tampella has not tried to standardize the structures according to the obstacle areas; rather the structure is determined in each project by the rules.

The upper headers, the weld joints and the lifting lugs of a header, which are needed for hanging the header and the furnace wall supported by it from the upper support structures, are obstacle areas. Boiler downcomers, boiler suspension rods, parallel circulation pipes and other piping also form obstacle areas. The pipeline locations are computed, using a layout in which the obstacles are avoided on the basis of the design rules. Consideration of several alternative layouts - previously unthinkable - is now routine.

The power of the system is based on the large amount of domain knowledge. When the amount of knowledge increases, knowledge management becomes the primary problem. In Design^{++} the knowledge management is supported by inheritance in library components, which allows to define and maintain knowledge only in one place, and by providing tools to locate, edit and store design rules.

The benefits obtained through automation in the design of the upper circulation pipes can be described as follows:

- ▲ using manual design, the design work took **2 months**
- ▲ with Computervision's plant design software it took **2 weeks**
- ▲ using Design^{++}, it takes **2 days.**

Working experiences

Tampella has made extensive studies of the possibilities of automating the design process with the following results (Kelhä, 1988 and Pernu, 1988):

"Speeding-up and improving preliminary design was set as a target. So far plant layout design has been carried out on the drafting board, since applying CAD at this stage has not brought sufficient benefits."

"From experience we know that the design problem cannot be solved with parametric programs. In order to automate the design of a boiler plant pressurized body, we have carried out in our CAD system parametric programs which typically become 20.000 to 30.000 rows in length. According to our experience, maintaining programs of this size becomes overwhelming when design rules are changing."

"In regards to the design of an entire boiler plant, the problem is far more complex and so no conventional parametric program can be considered. On the other hand, the latest technique, the expert system, promises to give an exact plant layout already in the stage of a proposal."

"The knowledge-based system gives new belief in design automation possibilities and opens up new prospects for the designer, in whose role the emphasis is being transferred from detail design for each project to the project-independent planning of design rules. Design will be more of an R&D nature and project design will become automated design control and supervision."

THE FUTURE

Design^{++} product development is today located in Cupertino, California. Silicon Valley is an ideal environment for further commercialization by providing the necessary technological infrastructure as well as the tradition of bringing new high technology products to the market. Many of the leading database, object-oriented software and CAD companies are located in the same area.

Design Power Inc. received venture capital financing in June 1990. This deal provides not only financial resources but also additional management support for making Design Power the leading supplier of services and software for knowledge-enhanced engineering automation.

Industry analysts predict that in 10 years, 80% of CAD products will incorporate knowledge-based features. Several major CAD-vendors are now looking ways to enter in the knowledge-based CAD market. Design Power envisions them more as potential partners than competitors.

We see a totally new market developing, what we call knowledge publishing. Knowledge publishing products allow engineering software developers to publish domain specific, public engineering knowledge in a generic and open form as libraries of intelligent components.

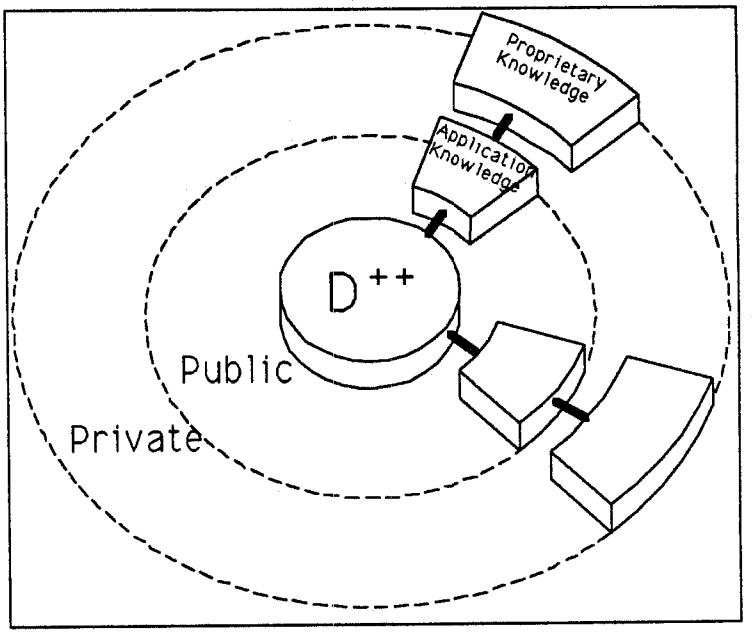


Figure 11: Knowledge organization.

In addition, component and equipment vendors are able to publish not only component and standards catalogs but also the engineering knowledge required to incorporate those products into designs. Vendors of motors, pumps and fans are able to deliver their product catalogs on optical disks, complete with the engineering knowledge to choose the right components, size them, and connect them to other components in a design.

Almost as a side effect, this capability enables end users to properly configure an order for a complex component, reducing errors and costs of a vendor's products.

End-users can further tailor the public libraries to their individual needs by adding their proprietary knowledge about selecting, connecting and laying out components on top of the "public" knowledge (Figure 11).

CONCLUSIONS

Knowledge-based engineering automation tools have put an enormous new source of power in the hands of engineers to manage increasingly complex designs and accomplish their tasks more rapidly and with less effort.

Today, knowledge-based CAD introduces an opportunity for advanced construction and manufacturing companies to dramatically increase productivity, reduce costs and improve quality in their engineering functions.

Design^{++} provides capabilities, which are especially valuable in proposal engineering and preliminary design phases. It is a complementary, not competing, tool to CAD systems, and it is not limited to any particular application. Design^{++} is a generic tool, applicable to virtually any kind of engineering problems.

Since its introduction, Design^{++} has been used, among other areas, for the following applications:

- ▲ ductwork design
- ▲ equipment configuration
- ▲ factory process planning and layout design
- ▲ computer network configuration
- ▲ process and instrumentation design
- ▲ process plant layout design.

The vision of a powerful knowledge-based CAD has become true. If we just place one question: what is the next major technological step in computer aided design? The answer is inevitably knowledge-based CAD.

REFERENCES

Dym C. and Levitt R. (1991). Knowledge-Based Systems for Engineering. McGraw-Hill, New-York.

Kelhä S. (1988), Boiler Plant Application Using CAD and Expert System. Computervision European Users´ Conference. Rome, Italy.

Levitt R. (1990). Merging AI with CAD: Intelligent, Model-Based Engineering. Shaw Memorial Lecture, North Carolina State University, Raleigh, N.C.

Pernu H. (1988), The Expert System in Boiler Plant Design, Computervision European Users Conference. Rome, Italy.

Riitahuhta A. (1988), Systematic Engineering Design and Use of an Expert System in Boiler Plant Design, International Conference on Engineering Design, ICED ´88. Budapest, Hungary.

Riitahuhta A. (1990), Enhancement of the Boiler Design Process by the use of Expert System Technology, CIFE Symposium Proceedings, Technical Report No. 24. Stanford University, Palo Alto, USA.

Spang Robinson (1989), The Spang Robinson Report on Artificial Intelligence, December 1989. John Wiley & Sons, New-York.

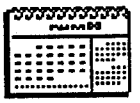
SCEMA

Scema, P.O. Box 59, SF-02601 Espoo, Finland. Visiting Address: Upseerinkatu 3 A. Phone +358-0-513 022, telefax +358-0-513 544

23.5.1991

Prototype for Haka Oy

BACKGROUND



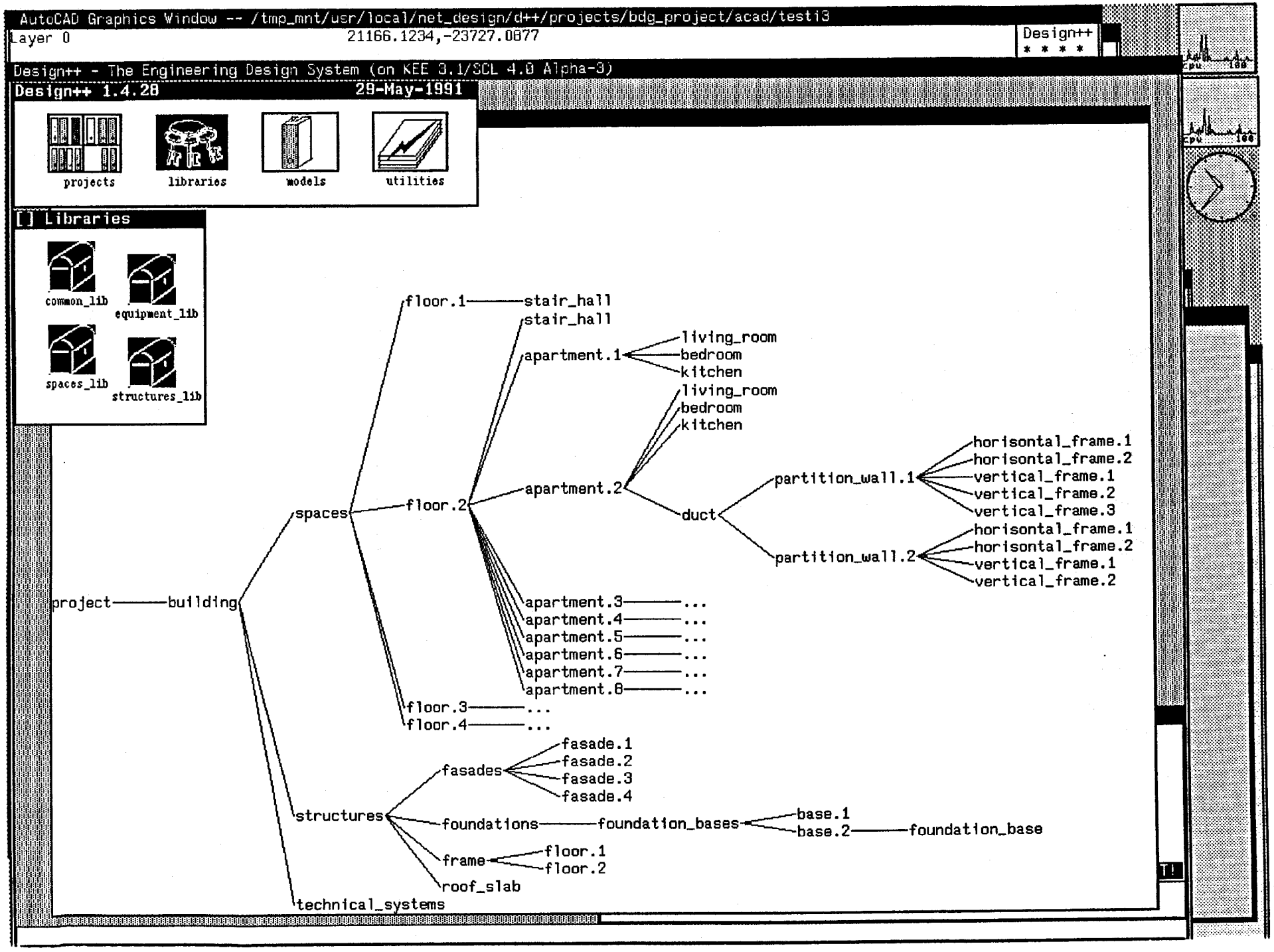
- Engineering and productstructure prototype. March 15 - May 23, 1991

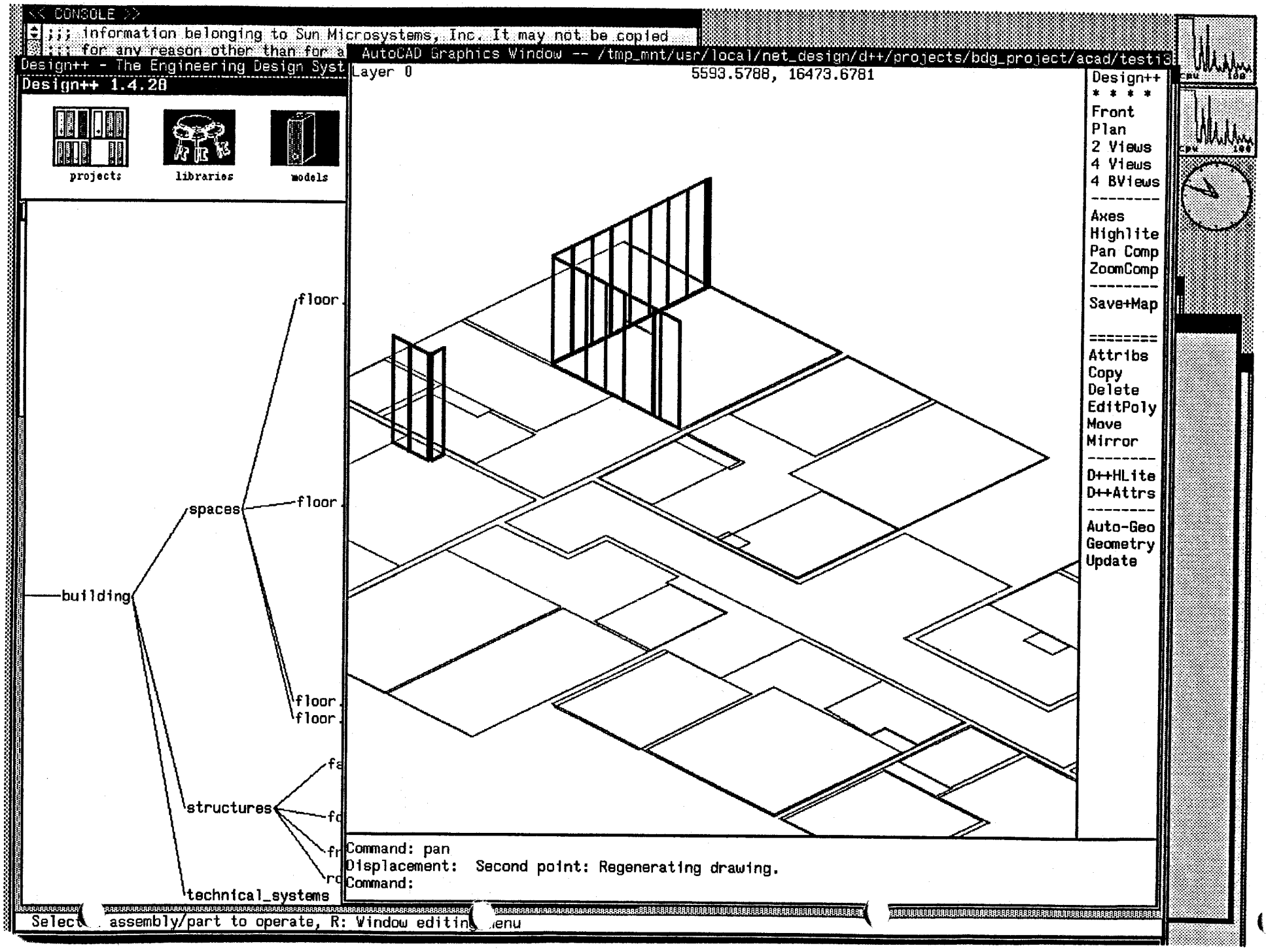
CHARACTERISTIC NUMBERS

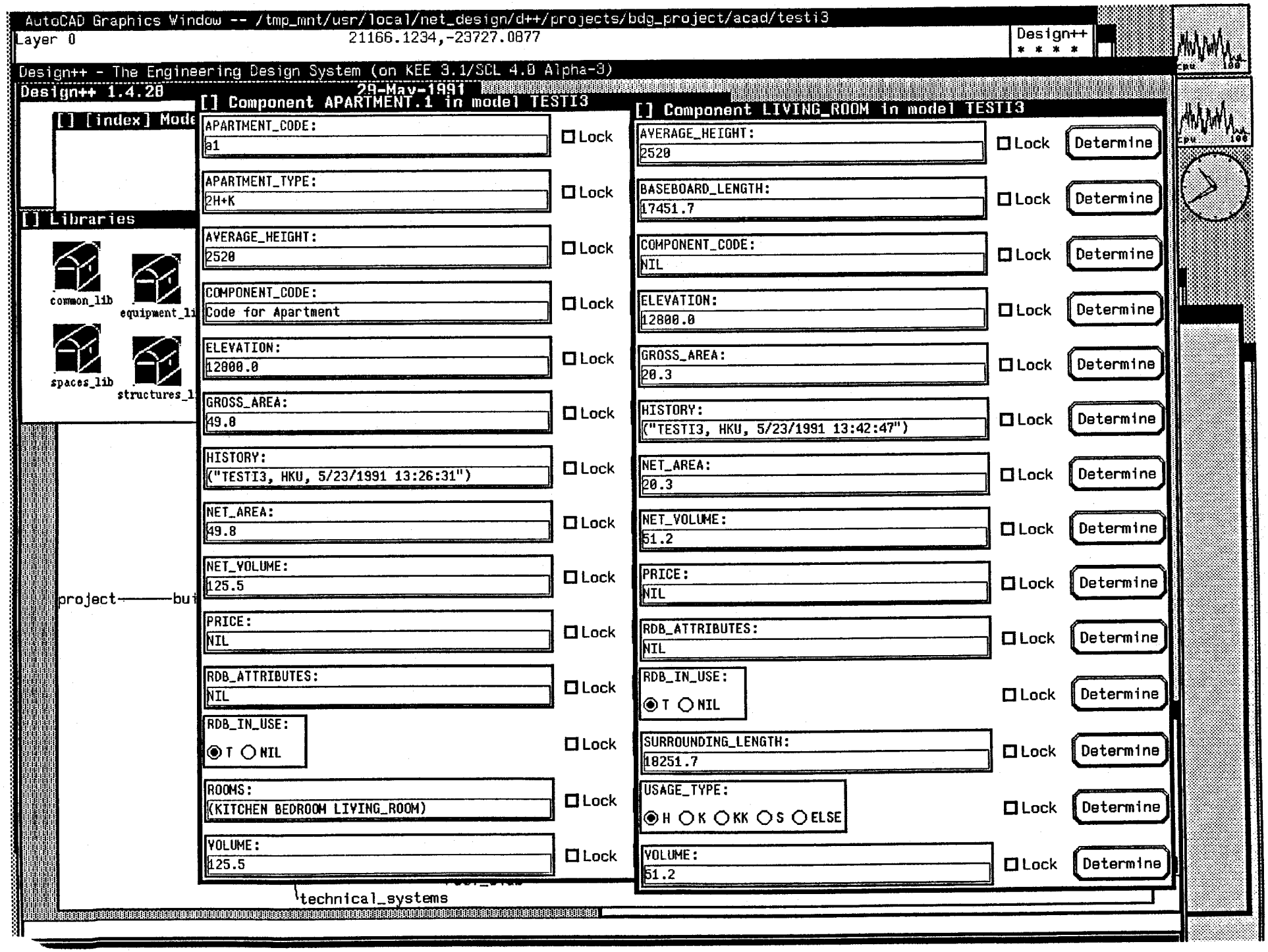
- Customer work 25...50 man-days
- Scema work 25 man-days

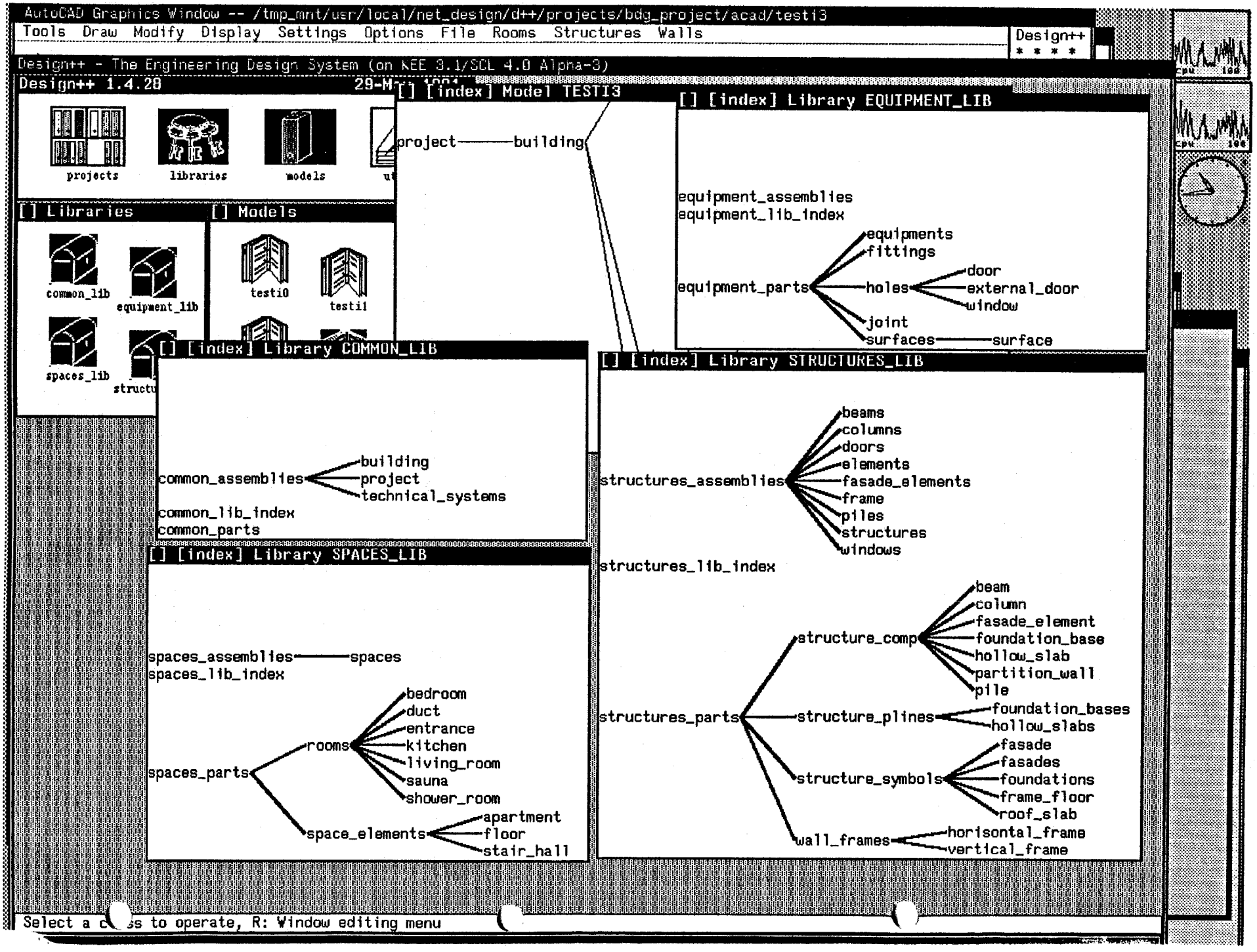
- 66 library components
- 1623/708 attributes

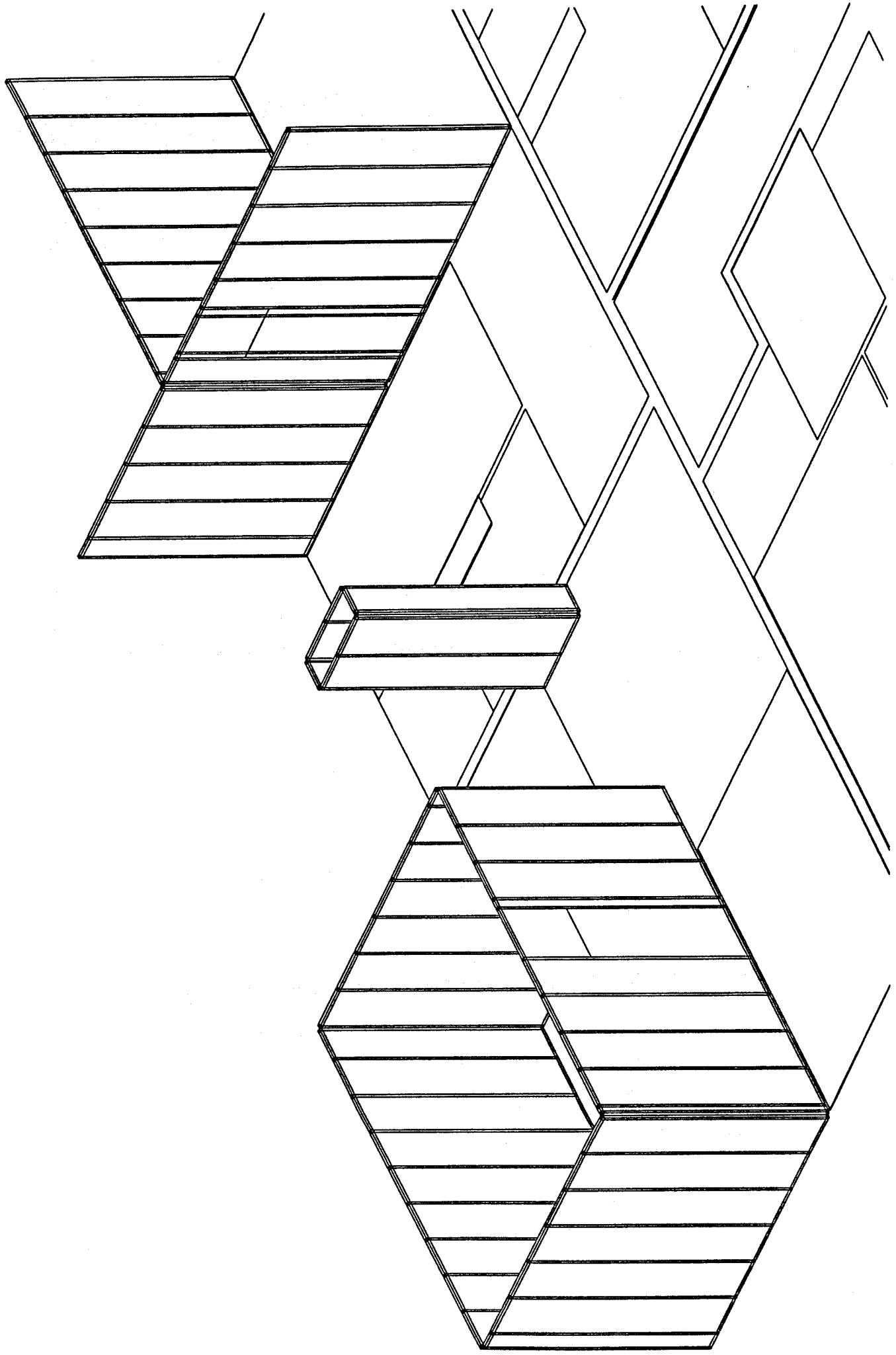
- 455/140 rules
- Proposal model had appr. 120 components, 3862/3159 attributes and 1708 rules


AutoCAD Graphics Window -- /tmp_mnt/usr/local/net_design/d++/projects/bdg_project/acad/testi3
Layer 0
21166.1234,-23727.0877
Design++
Design++ - The Engineering Design System (on KEE 3.1/SCL 4.0 Alpha-3)
Design++ 1.4.28
29-May-1991
projects
libraries
models
utilities
[] Libraries
common_lib
equipment_lib
spaces_lib
structures_lib
project
building
spaces
floor.1
stair_hall
stair_hall
apartment.1
living_room
bedroom
kitchen
floor.2
apartment.2
living_room
bedroom
kitchen
duct
partition_wall.1
horisontal_frame.1
horisontal_frame.2
vertical_frame.1
vertical_frame.2
vertical_frame.3
partition_wall.2
horisontal_frame.1
horisontal_frame.2
vertical_frame.1
vertical_frame.2
apartment.3
apartment.4
apartment.5
apartment.6
apartment.7
apartment.8
floor.3
floor.4
structures
fasades
fasade.1
fasade.2
fasade.3
fasade.4
foundations
foundation_bases
base.1
base.2
foundation_base
frame
floor.1
floor.2
roof_slab
technical_systems


CONSOLE
;;; information belonging to Sun Microsystems, Inc. It may not be copied
;;; for any reason other than for a
Design++ - The Engineering Design Syst
Design++ 1.4.28
projects
libraries
models
building
spaces
floor
floor
floor
floor
structures
technical_systems
AutoCAD Graphics Window -- /tmp_mnt/usr/local/net_design/d++/projects/bdg_project/acad/testi3
Layer 0
5593.5788, 16473.6781
Design++
* * * *
Front
Plan
2 Views
4 Views
4 BViews
Axes
Highlite
Pan Comp
ZoomComp
Save+Map
Attribs
Copy
Delete
EditPoly
Move
Mirror
D++HLite
D++Attrs
Auto-Geo
Geometry
Update
Command: pan
Displacement: Second point: Regenerating drawing.
Command:
Select assembly/part to operate, R: Window editing menu


AutoCAD Graphics Window -- /tmp_mnt/usr/local/net_design/d++/projects/bdg_project/acad/testi3
Layer 0
21166.1234,-23727.0877
Design++
Design++ - The Engineering Design System (on KEE 3.1/SCL 4.0 Alpha-3)
Design++ 1.4.20
29-May-1991
[] [index] Mode
[] Libraries
common_lib
equipment_li
spaces_lib
structures_l
project
bui
technical_systems
[] Component APARTMENT.1 in model TESTI3
APARTMENT_CODE:
a1
APARTMENT_TYPE:
2H+K
AVERAGE_HEIGHT:
2520
COMPONENT_CODE:
Code for Apartment
ELEVATION:
12000.0
GROSS_AREA:
49.0
HISTORY:
("TESTI3, HKU, 5/23/1991 13:26:31")
NET_AREA:
49.8
NET_VOLUME:
125.5
PRICE:
NIL
RDB_ATTRIBUTES:
NIL
RDB_IN_USE:
T NIL
ROOMS:
(KITCHEN BEDROOM LIVING_ROOM)
VOLUME:
125.5
Lock
[] Component LIVING_ROOM in model TESTI3
AVERAGE_HEIGHT:
2520
BASEBOARD_LENGTH:
17451.7
COMPONENT_CODE:
NIL
ELEVATION:
12800.0
GROSS_AREA:
20.3
HISTORY:
("TESTI3, HKU, 5/23/1991 13:42:47")
NET_AREA:
20.3
NET_VOLUME:
51.2
PRICE:
NIL
RDB_ATTRIBUTES:
NIL
RDB_IN_USE:
T NIL
SURROUNDING_LENGTH:
18251.7
USAGE_TYPE:
H K KK S ELSE
VOLUME:
51.2
Lock
Determine


AutoCAD Graphics Window -- /tmp_mnt/usr/local/net_design/d++/projects/bdg_project/acad/testi3
Tools Draw Modify Display Settings Options File Rooms Structures Walls
Design++
Design++ - The Engineering Design System (on KEE 3.1/SCL 4.0 Alpha-3)
Design++ 1.4.28
projects
libraries
models
[] Libraries
common_lib
equipment_lib
spaces_lib
[] Models
testi0
testi1
[] [index] Model TESTI3
project
building
[] [index] Library EQUIPMENT_LIB
equipment_assemblies
equipment_lib_index
equipment_parts
equipments
fittings
holes
door
external_door
window
joint
surfaces
surface
[] [index] Library COMMON_LIB
common_assemblies
building
project
technical_systems
common_lib_index
common_parts
[] [index] Library STRUCTURES_LIB
structures_assemblies
beams
columns
doors
elements
fasade_elements
frame
piles
structures
windows
structures_lib_index
structures_parts
structure_comp
beam
column
fasade_element
foundation_base
hollow_slab
partition_wall
pile
structure_plines
foundation_bases
hollow_slabs
structure_symbols
fasade
fasades
foundations
frame_floor
roof_slab
wall_frames
horisontal_frame
vertical_frame
[] [index] Library SPACES_LIB
spaces_assemblies
spaces
spaces_lib_index
spaces_parts
rooms
bedroom
duct
entrance
kitchen
living_room
sauna
shower_room
space_elements
apartment
floor
stair_hall
Select a c ss to operate, R: Window editing menu



SITE INFORMATION

DATE : 24-May-1991
SITE : TESTI3

BDG volume	3393.3	m3	BDG volume/Gross floor area		2.52
Gross floor area	1346.4	m2	Net floor area/Gross floor area		0.99
Net floor area	1332.3	m2	Net floor area/Apartments		55.51
Apartments	24	pcs			

APARTMENT LIST

location	apartment type	pcs	Nm2
a1 / 2 floor	2H+K	1	49.8
a2 / 2 floor	2H+K	1	58.1
a3 / 2 floor	2H+K+S	1	63.7
a4 / 2 floor	2H+K	1	49.1
a5 / 2 floor	2H+K	1	54.6
a6 / 2 floor	3H+K+S	1	73.3
a7 / 2 floor	2H+K	1	58.3
a8 / 2 floor	2H	1	37.2
a9 / 3 floor	2H+K	1	49.8
a10 / 3 floor	2H+K	1	58.1
a11 / 3 floor	2H+K+S	1	63.7
a12 / 3 floor	2H+K	1	49.1
a13 / 3 floor	2H+K	1	54.6
a14 / 3 floor	3H+K+S	1	73.3
a15 / 3 floor	2H+K	1	58.3
a16 / 3 floor	2H	1	37.2
a17 / 4 floor	2H+K	1	49.8
a18 / 4 floor	2H+K	1	58.1
a19 / 4 floor	2H+K+S	1	63.7
a20 / 4 floor	2H+K	1	49.1
a21 / 4 floor	2H+K	1	54.6
a22 / 4 floor	3H+K+S	1	73.3
a23 / 4 floor	2H+K	1	58.3
a24 / 4 floor	2H	1	37.2
TOTAL		24	1332.3

PARTITION WALL PRODUCT STRUCTURE

code	name	amount	unit	work mk	material mk	total
4560	GYPSUM BOARD	30	m2	241.8	321.3	2275.02
4365	WOODEN DOORS	1	pcs		295.0	295.0
5480	TILING		m2			
6100	FURNITURE		m2			
8	USAGE		m2			
9	GENERAL		m2			
4561	SEAL JOINT	52	m	0.0	415.44	415.44
5800	LEVELLING	30	m2		600	600
5801	PAINTING	30	m2		450	450

```
(in-package 'kee)

;;;; Thu 18-Apr-91 11:07:13 by HKU
;;;  Forms surrounding length mm
(:! FOUNDATION_BASE FORM_SURROUNDING_LENGTH
    (ROUND-NUMBER (LENGTH-OF-POLYLINE (:? MY GEO_PLINE)) 1)
    )


;;;; Thu 18-Apr-91 11:10:03 by HKU
;;;  Area of form m2
(:! FOUNDATION_BASE FORM_AREA
    (ROUND-NUMBER
     (/ (* (:? MY HEIGHT)
           (:? MY FORM_SURROUNDING_LENGTH)) 1000000) 2)
    )


;;;; Thu 18-Apr-91 11:14:29 by HKU
;;;  Area against soil m2
(:! FOUNDATION_BASE CROSS_AREA
    (ROUND-NUMBER (/ (S::CALCULATE-AREA-OF-PLINE (:? MY GEO_PLINE))
                     1000000) 1)
    )


;;;; Thu 18-Apr-91 11:17:37 by HKU
;;;  Volume of concrete m3
(:! FOUNDATION_BASE CONCRETE_VOLUME
    (ROUND-NUMBER (/ (* (:? MY HEIGHT) (:? MY CROSS_AREA)) 1000) 2)
    )


;;;; Wed 15-May-91 10:26:18 by HKU
;;;  Amount of 1x4 wood m
(:! FOUNDATION_BASE 1X4_WOOD_LENGTH
    (ROUND-NUMBER
     (+ (* 3.0 (/ (:? MY FORM_SURROUNDING_LENGTH) 1000))
        (* 10.0 (:? MY FORM_AREA)))
     2))


;;;; Wed 15-May-91 10:32:33 by HKU
;;;  Amount of 60x2.5 nails kg
(:! FOUNDATION_BASE 60X2_5_NAIL
    (ROUND-NUMBER (* 0.004 (* 2 (:? MY 1X4_WOOD_LENGTH))) 2)
    )
```

MATERIAL COSTS

APARTMENT 2 FLOOR 2

Article	amount	unit	type	mk/unit	waste
Gypsymboard	2.65	pcs	RO	0.0	1.05
Gypsymboard	8.01	m2	RO	10.71	1.05
Upper horizontal frame	1.58	m	42*30	2.33	1.05
Lower horizontal frame	1.58	m	42*30	2.33	1.05
Vertical frame	14.0	m	42*35	2.88	1.03
Screws 25 * 4.2	98.0	pcs	25*4.2	0.03	1.05
Joint band	0.0	m		1.71	1
Elastic seam	14.37	m		8	1
Stoppers	7.0	pcs		0.4	1
Sealing band	3.17	m		1.71	1.02
Wood 30x40mm	11.2	m		2.1	1.05
Sheet moulding 40x40mm v		m		4.8	1.2
Sheet moulding 60mm		m		2.5	1.1
Corner moulding L 40x40m		m		3	1.1
Cupboard clamp h=150mm		m		2	1.1
Wash-basin clamp		pcs		20	1.1
Shower clamp		pcs		15	1
Chipboard 13mm		m		4.4	1
Insulation	4.45	m2		9.73	1.3
Wall area	4.45	m2			1.1
Room height	2.52	m			

APARTMENT 3 FLOOR 2

Article	amount	unit	type	mk/unit	waste
Gypsymboard	14.97	pcs	RO	0.0	1.05
Gypsymboard	45.27	m2	RO	10.71	1.05
Upper horizontal frame	8.98	m	42*30	2.33	1.05
Lower horizontal frame	8.98	m	42*30	2.33	1.05
Vertical frame	50.4	m	42*35	2.88	1.03
Screws 25 * 4.2	570.0	pcs	25*4.2	0.03	1.05
Joint band	39.2	m		1.71	1
Elastic seam	37.56	m		8	1
Stoppers	41.0	pcs		0.4	1
Sealing band	17.06	m		1.71	1.02
Wood 30x40mm	11.2	m		2.1	1.05
Sheet moulding 40x40mm v		m		4.8	1.2
Sheet moulding 60mm		m		2.5	1.1
Corner moulding L 40x40m		m		3	1.1
Cupboard clamp h=150mm		m		2	1.1
Wash-basin clamp		pcs		20	1.1
Shower clamp		pcs		15	1
Chipboard 13mm		m		4.4	1
Insulation	0.0	m2		9.73	1.3
Wall area	25.15	m2			1.1
Room height	2.52	m			

FOUNDATION BASE PRODUCT STRUCTURE

code	name	amount	unit	hours	work mk	material mk	total
2010	FORM WORK	4	m2	3.55	230.49	474.69	705.18
2021	REINFORCEMENT	6	kg	0.01	0.84	24.73	25.57
2022	CONCRETE WORK	0	m3	0.13	8.77	104.4	113.17

FORM WORK

material	theoretical amount	waste factor	amount to require	mk/ unit	use factor	mk
Wood 50x100mm	78.83	1,5	118.24	1.6	1/1	189.[illegible]
Wood 25x100mm	46.37	1,5	69.55	4.1	1/1	285.[illegible]
Nail 100x3x4	1.85	1,1	2.77	0.1	1/1	0.28
Nail 60x2x5	0.63	1,1	0.94	0.05	1/1	0.05

REINFORCEMENT

material	theoretical amount	waste factor	amount to require	mk/ unit	use factor	mk
Bar 8mm	0.0	1,5	0.0	2.56	1/1	0.0
Bar 12mm	6.44	1,5	9.66	2.56	1/1	24.73

CONCRETE WORK

material	theoretical amount	waste factor	amount to require	mk/ unit	use factor	mk
Concrete	0.27	1,1	0.3	351.5	1/1	104.4

Building Functional Heirarchy as an AND-OR graph

- basis for top-down design
- abstraction hierarchies
- data modeling notion of aggregation

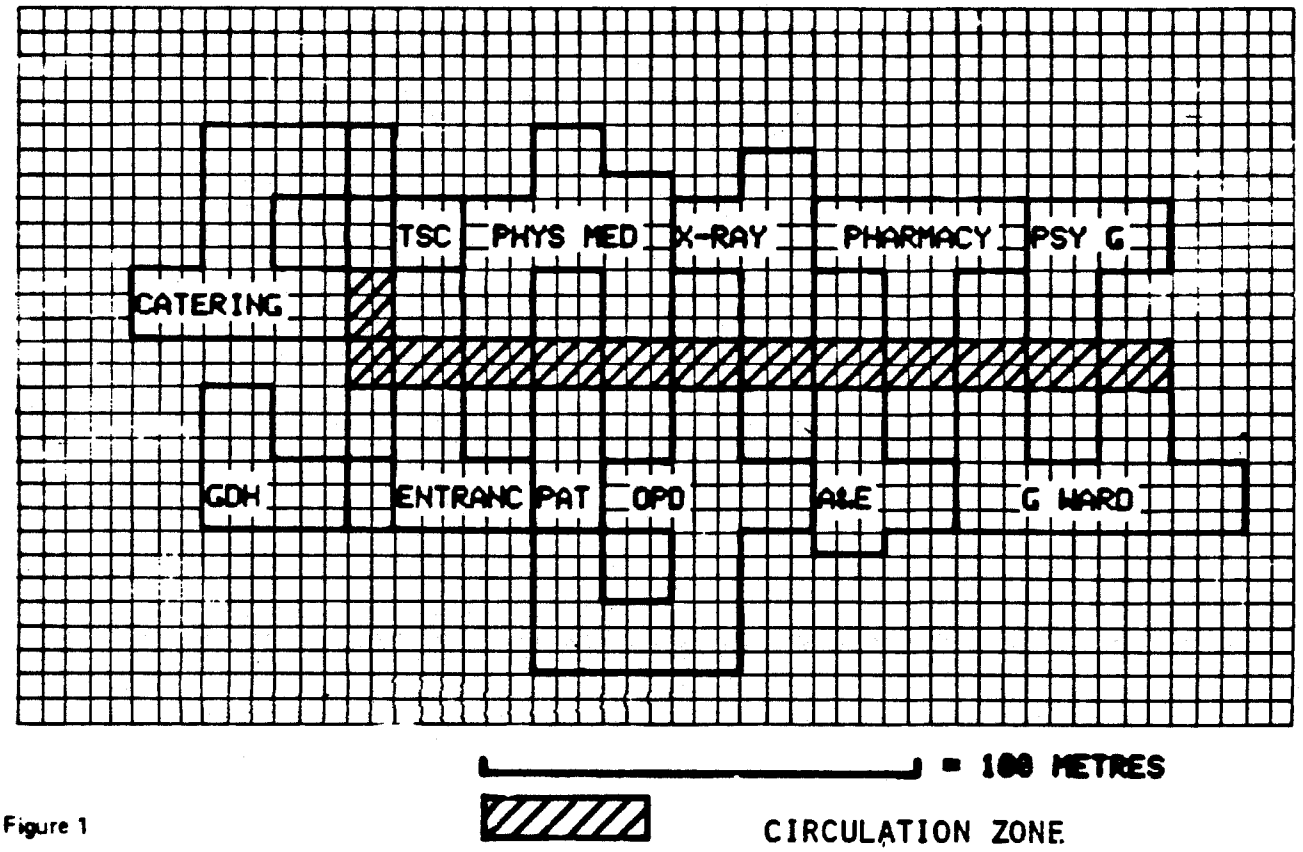

TSC
PHYS MED
X-RAY
PHARMACY
PSY G
CATERING
GDH
ENTRANC
PAT
OPD
A&E
G WARD
= 100 METRES
CIRCULATION ZONE

Figure 1

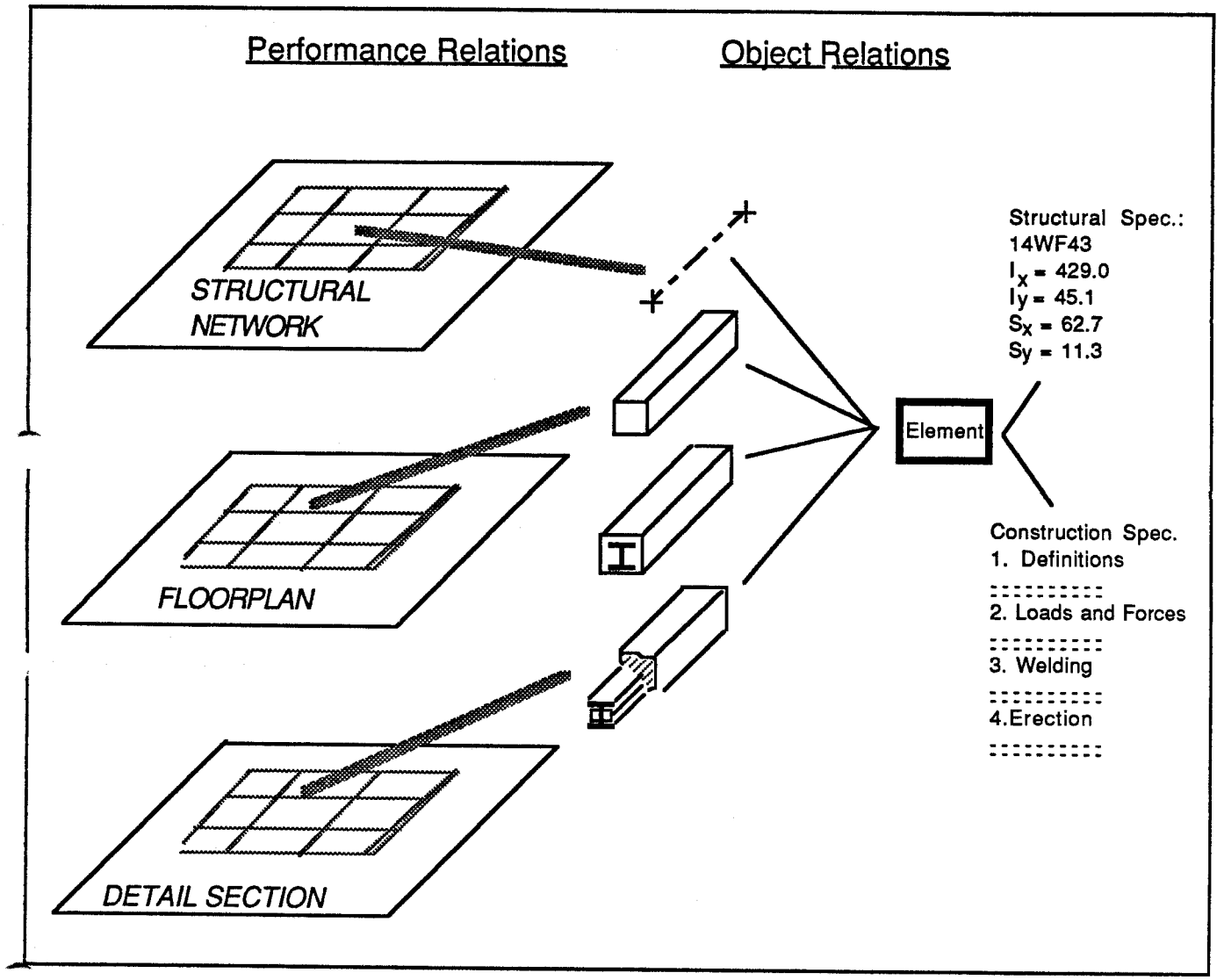
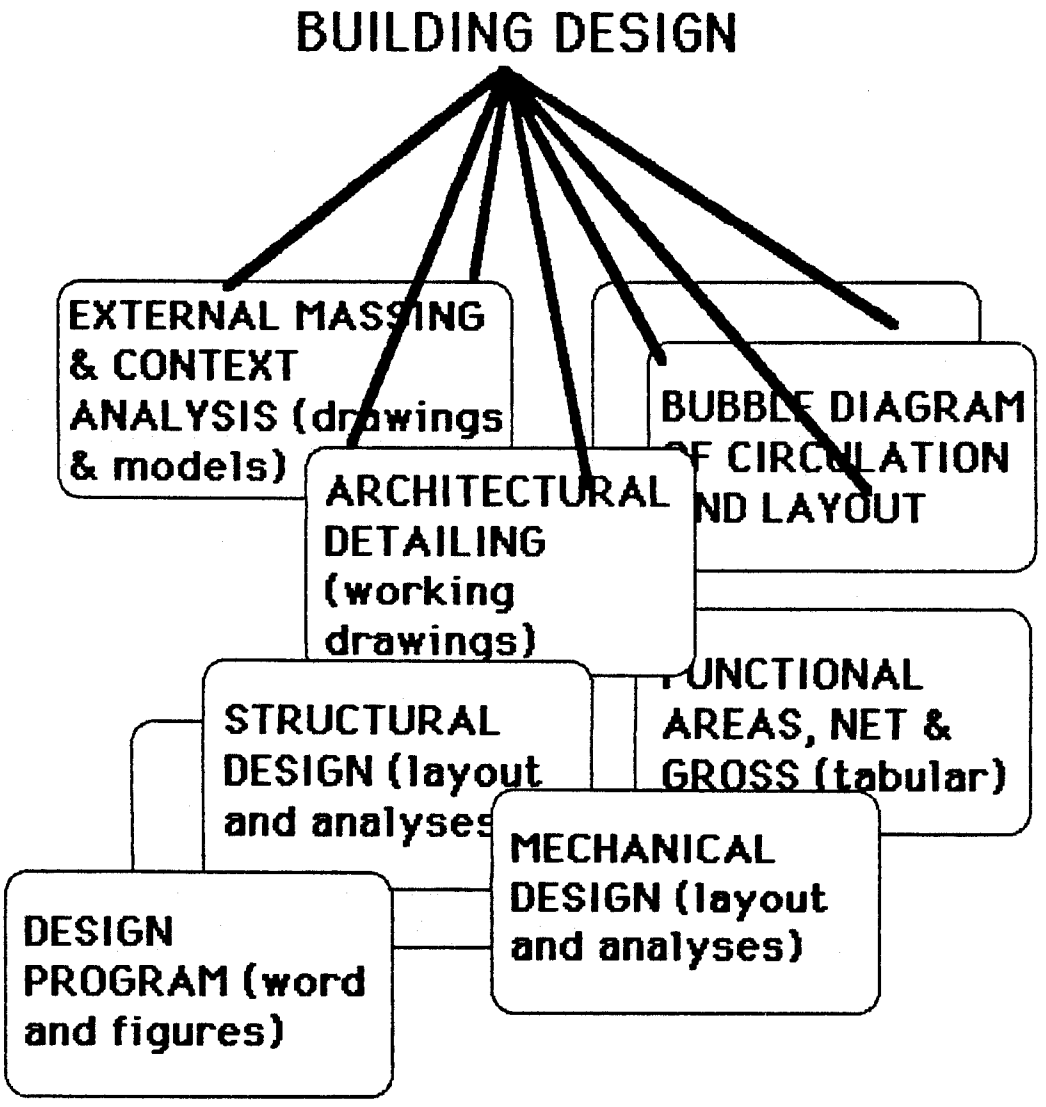


FIGURE THREE: Two classes of structure that are embedded within a modeling system.

A BUILDING DESIGN AS MULTIPLE DESCRIPTIONS



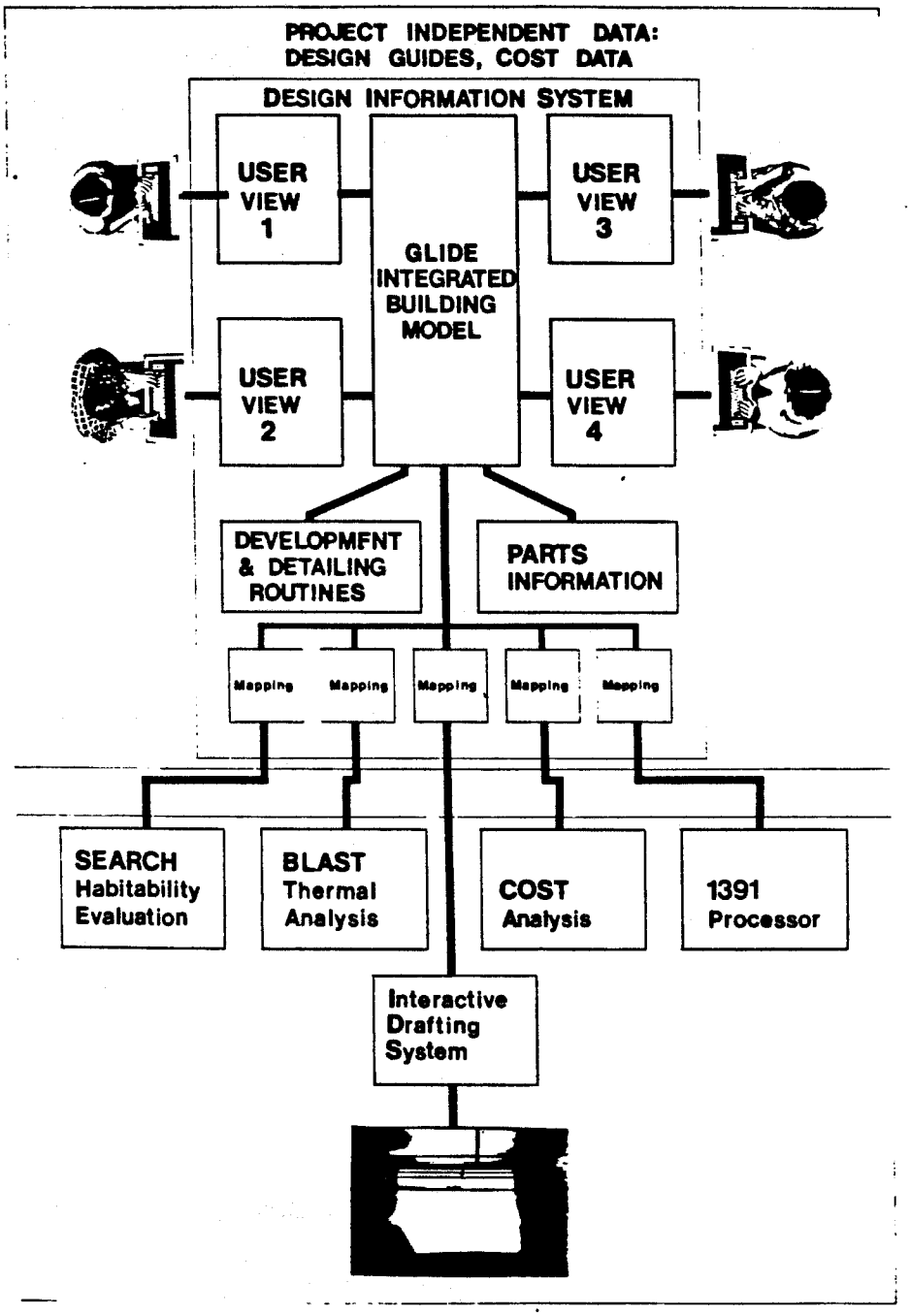
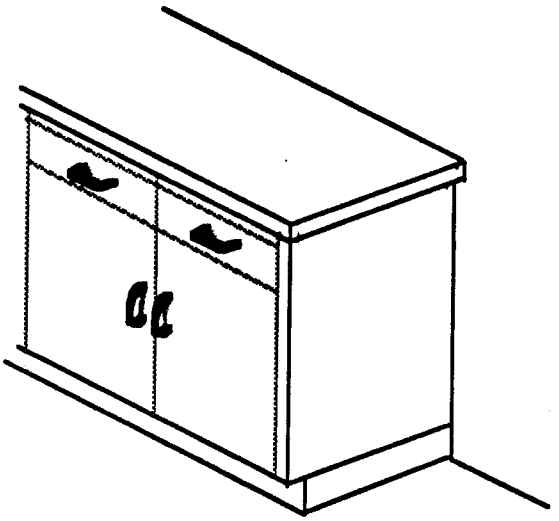
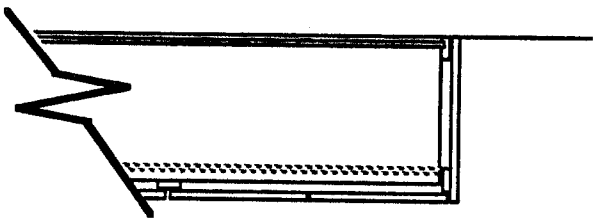


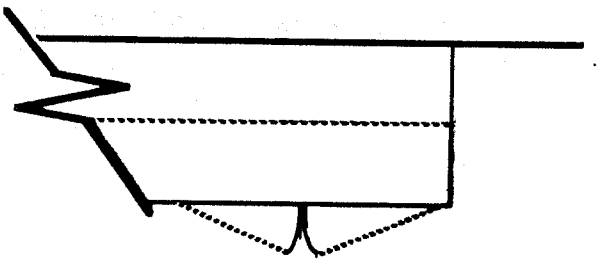
Figure 1: functional organization of the CAEADS system

DRAWING CONVENTIONS ARE NOT EQUIVALENT TO SECTIONS





HORIZONTAL SECTION



DRAWING CONVENTIONS