GENO-OBJECT-CLASSES IN CONSTRUCTION IT

Kristian Agger, The Aarhus School of Architecture.

ABSTRACT

The Geno project intends to participate in the development of the next generation of construction IT systems. Goals for this research should be: to loosen the design process from the production of design documents, to free the geometry from orthogonal projection, to make possible a full, variable, complete detailing without loosing consistency, to move the development of building component specific IT modelling tools closer to the end user and to improve the efficiency and capability of these modelling tools.

The Geno project works with three developer / user layers: 1. GenoObjectClasses, the basic standardized data and functional structure, developed by IT specialists in a close dialogue with the IAI IFC development. 2. ProtoObjectClasses: IT tools for modelling spaces, construction elements and parts. Developed by IT specialized architects, engineers, on the bases of Genotypes. Made available to the end user through Internet by component vendors. 3. PhenoObjects: spaces, construction elements and parts, specified, dimensioned and placed and interrelated by the designer, to be analyzed and supply project information for all participants in the construction and management process. The idea of this structure is to improve dynamic and user influence on IT modelling tool development.

The standardized class structure for this, the GenoObjectClasses has to support three concurrent models, namely the: 1. SpaceModel, an interrelated surface model, a non-detailed division of the project space in functional spaces (living room, kitchen, bath etc.) and construction spaces (foundation, wall, roof, slap, etc.). 2. ComponentModel, a successive partitioning or filling the SpaceModel with building elements, components (facing wall, inner wall, insulation, window, door, ceiling, roof construction, inventory, furniture, etc.), interrelated and related to the SpaceModel. 3. EntityModel, a similar fill to the Componentmodel with building parts (brick, joint, plaster, fitting, gutter, etc.) to make a complete consistent product model possible.

The "three model structure" to be filled out successively, add flexibility to the design process. When calculations and visualizations are performed the detailed model is used, but in areas with no detailing the model on the lower detailing level is used. This means that the total model will be "complete", if only the SpaceModel has been modeled. The development of GenoObjectClases will build as close as possible on IFC, and seek to expand IFC where it is necessary.

Status for the Geno project is that implementation has been started with AutoCAD ObjectARX.

Keywords, product model, design process, generic object classes, design detailing .

INTRODUCTION

Next step in the development of the IT system for construction must concentrate on much better modeling efficiency and access to full (but variable) detailing. This will make it possible to build a model that can support analyses and generation of project documents at all levels of detailing. Modelling efficiency can be obtained through the development of specialized modelling tools for each single part of the building. These tools should be implemented by specialist designers or by construction industry vendors as part of component documentation available on the Internet.

A precondition for a tool development, independent of traditional CAD system development is the existence of a standard object class kernel, a comprehensive product model. This GenoModel should co-ordinate 3D geometry, attributes, construction and inter-connection of functional spaces and building parts in one structure. All participants in the design and construction and management process will model into and gain information from this object database, and the traditional project documents will disappear.

Next generation of construction IT-system, built on these premises will:

- loosen the design process from the production of design documents,
- free the geometry from orthogonal projection
- make possible a full, variable, complete detailing without loosing consistency
- move the development of building component specific IT modelling tools closer to the end user and
- improve the efficiency and capability of these modelling tools

SYSTEM DEVELOPMENT

None of the CAD systems in use in design offices today are fully optimised for building modelling. The data models are not structured to hold a connected building product model, and that is why the modelling tools available only offer efficient drawing edition and modelling of simple 3D models for visualisation (AutoCAD, MiniCAD, Arris etc.). Still, most building oriented systems (AutoCAD-Point, ArchiCAD, AllPlan etc.) only have capacity for an integrated model that can support none detailed drawing production and calculation. These CAD systems imitate the traditional design process and documentation, without the profound renewal and improvement of the design process that is the potential of IT, as seen in for example the car industry.

Modern system implementation is object oriented, based on object classes, a code module defining both data (attributes) and methods (functions) of the object. The advantage of this structure, is that the Object class can be redeveloped without affecting the rest of the system, as long as the naming of data and methods referenced by other objects is not changed. An object can represent a very concrete thing, a building part, a window with dimension attributes, parameters for framing, and methods for placing, calculation, etc. Object classes can also be something more abstract, for instance a relation between a window object and a wall object, securing that the window follows the wall during modelling and that the area of the window is subtracted from the wall area. An editing (modelling) command is also implemented as an object. A modeling object calls the relevant functions in the objects that take part in the modelling operation.

Traditional CAD systems are developed by the CAD vendor. Most systems allow the advanced user to adjust and expand the system to a more specific use, with a programming tool. In AutoCAD it is ObjectARX that gives access to add new classes on top of existing classes. MiniCAD offers MiniPascal, that is without object structure. And ArchiCAD has a graphic programming language GDL, to build parametric components from, but without the facilities to expand with new objects or functions.

The GenoProject tends to offer a developer / user structure with 3 layers:

- *GenoTypes*, the basic object (data / function) structure developed by IT specialists, in accordance and dialogue with IFC (Industry Foundation Classes developed by IAI). GenoObjectClasses hold attributes: the geometrical form, material and surface characteristics and relations / bindings to other objects, and methods: to visualise the object in different projections and cut, to calculate quantities, area, volume, point of gravity, weight. Methods to receive and transfer forces and heat in the static or thermal systems. Methods to model geometry and topology. General methods to parameterise and to group objects. This basic kernel system is to be developed by traditional CAD vendors.
- *ProtoTypes,* are the tools to model spaces, construction elements and parts, developed by specialized architects/engineers for design offices and building article producers. Based on GenoObjectClasses by heritage of methods and attributes, ProtoObjectClasses will be developed with its own set of data and functions, making the ProtoType a modelling tool specialised for designing different space types and different building parts: walls, windows, doors, columns, slabs roofs, panels, stairs, boarding, covering, etc.. ProtoTypes can be distributed via Internet as part of building article information.
- *PhenoObjects*, are spaces, building elements and parts, specified, dimensioned and placed by the designer, to establish a project model that allows every participants in the building process to generate analyses and all kinds of documentation.

This structure will bring the development of modelling tools closer to the user. As each element type of the building needs its own tools to be built, each part of the building needs its own modelling tools to be modelled effectively. New building articles need new modelling tools, ProtoTypes, so the development will be a continued process. As ProtoTypes are based on GenoTypes they will "plug in" to every Geno based kernel IT system.
The GenoProject define GenoObjectClasses, and develop ProtoObjectClasses to test and refine the kernel.

GENO-OBJECT-CLASSES

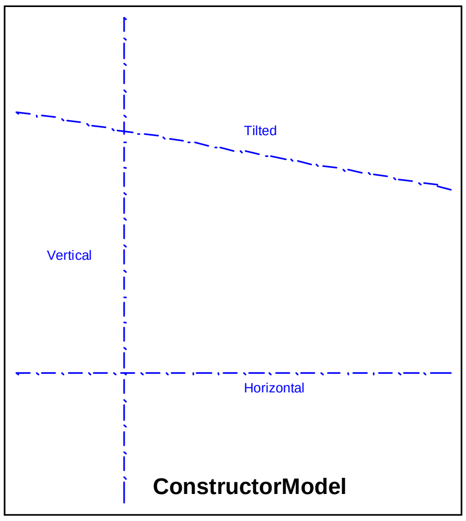
In the search for the essence of the GenoTypes, it has been important to find a structure that will support the design process. No single theory describes the designers methods. Registrations of the process often find a pattern where the designer relatively fast jumps to a conclusive idea to be evaluated and refined. Another observation is that some details are developed to a conclusion very early in the process. The reason for this could be that the detail influences the rest of the geometry, that it visually plays an important role, or just that it is nice to record ideas immediately. These considerations have led to a structure, where the GenoObjectClasses must support 3(4) concurrent models, the:

- ConstructorModel (hardly a model), consists of points, curves, planes, that the same way as gridlines, support and guide the construction of the:
- SpaceModel, an interrelated surface model, a non-detailed division of the project space in functional spaces (living room, kitchen, bath etc.) and construction spaces (foundation, wall, roof, slap, etc.).

- ComponentModel, a successive partitioning or filling of the SpaceModel with building elements, components (facing wall, inner wall, insulation, window, door, ceiling, roof construction, inventory, furniture, etc.), interrelated and related to the SpaceModel.
- EntityModel, a similar fill to the ComponentModel with building parts (brick, joint, plaster, fitting, gutter, etc.) to make a complete consistent product model possible.

The idea of more concurrent models to be filled out successively, add flexibility to the design process. When calculations and visualizations are performed the detailed model is used, but in areas with no detailing the model on the lower detailing level is used. This means that the total model will be "complete", if only the SpaceModel has been modelled.

ConstructorModel



Purpose

The ConstructorModel is the first rough outline of the site and the possible building volume. The “model” is established successively when it is needed and functions as a guide during design of the SpaceModel, so that the surfaces and edges of the SpaceModel can be related to one or more of the elements, “constructors”, in the ConstructorModel. A move or turn of an element in the ConstructorModel will cause the related elements from the SpaceModel to follow. Thus the constructors can be used to control the superior geometry.

Object Classes

The elements of the ConstructorModel, the *Constructor*, can be a point, a curve (straight or curved) or a plane (flat, curved, sculptured).
The ConstructorObject defines type (point, curve, plane), geometry (form, placing) and relations to other ConstructorObjects.
Examples: site border, building line, height controlling plane defining the exterior limits for the project and, grid line, and floor, ceiling and roof plane controlling the interior geometry.

Modelling

The constructor plane can be defined by:

- Intersection curve with another constructor plane, in a given angle.

- Parallel connection to another constructor, in a given distance.
- Tangential connection in a point.
- Three or more 3D points.
- Lofting, extrusion, point interpolation, etc.

Single curved constructor curve is defined on a flat plane, double curved constructor curve on a curved plane, and constructor points by co-ordinates.

Relations

The constructor can be defined as fixed or related to other constructors by angular, parallel displacement or tangential relations. The floor plane constructors can for instance be interrelated with parallel displacement. If one floor to floor distance is changed all interrelated floor plane constructors will adjust.

Visualisation

The visual appearance of the constructor depend on the type and whether is to be shown in the projection plane or just somewhere else in the view cone.

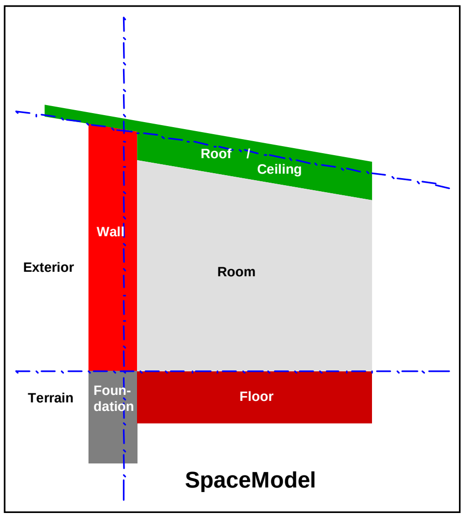
placing/type (colour)	Point (red)	Curve (green)	Plane (blue)
In the projection plane	*Bold cross*	*Bold line*	*Bold intersection line*
Elsewhere	*Cross*	*Line*	*Grid*

Analyses and Documentation

No analyses can be generated solely based on constructors.

The constructor can be shown on drawing documents for example as grid lines.

SpaceModel



Purpose

The SpaceModel is a superior subdivision of the total project space into functional rooms and constructional spaces. A sketch model, able to deliver data for rough calculations and visualisations. The SpaceModel is a surface model, where the surfaces can be related to Constructors and to each other, and where the surfaces constitute the relations between the spaces, so that generation of calculation data is possible. The degree of detailing is like a cartoon model, but it is possible to successively fill the SpaceModel with components, building elements, inventory, furniture, etc..

Object Classes

The most important object classes in the SpaceModel is the SpaceObject and the SurfaceObject:

- The SpaceObject, the room, contains data about the room type (functional rooms: living room, kitchen, bath, workshop, concert hall, exterior, etc. or spaces for construction: exterior wall, interior wall, horizontal division, ceiling, roof stair, etc.) and attributes (Functional activity descriptions: Proportion limits, orientation, access relations, etc. or preliminary building element specifications: Cost, U-value, density, construction duration, maintenance cost, lifetime cost, etc.), visual appearance on sections (colour, raster), relations to other SpaceObjects, and relations to SurfaceObjects, that constitute the shape, relations to ComponentObjects (se later) filled into the space.
- The SurfaceObject, planes that separate the spaces, has data describing, geometry: type, placing and contour, visual appearance: contour line, colour, texture and references to the two spaces it separates. It has relations (eventually) to Constructors and other SurfaceObjects, to control placing and contour. The SurfaceObject will be related to the ComponentObjects through their EnvelopeObjects (see later under ComponentModel definition). These relations cause the EnvelopeObjects to “make a hole into” the SurfaceObjects, so that when the models are visualised the components will be shown where they are present.

Modelling

Modelling the SpaceModel includes the creation of the SpaceObjects directly during the space programming phase or indirectly when modelling the SurfaceObjects.
Modelling the spaces can take the following forms:

- Modelling one single SurfaceObject with references to two spaces it separates, and relations to other SurfaceObjects and Constructors, and graphical 3D input on constructor plan or through snap to existing points
- Modelling a complete room or an extension to a space (and thereby reduction of neighbouring spaces) by extruding a SurfaceObject to a given height or to a constructor plane. This includes a process where the generated surfaces are properly related to surrounding spaces and surfaces.
- Editing the SpaceModel through changes to the ConstructorModel or to changes in parameters in relations to other SurfaceObjects or Constructors.

Relations

Relations that are internal to the SpaceObjects as part of the spatial program are: distance in between (max. min.), access (direct, indirect or none), daylight (access and orientation), view.
SpaceObjects have relations to their bounding SurfaceObjects, which are pointing back to the two SpaceObjects they are common to.
The placing relation between SurfaceObjects and Constructors and other SurfaceObjects can be angular, parallel displacement or tangential. SurfaceObjects can have a common contour with another SurfaceObject. And there can be a “hole making” relation to an EnvelopeObject of a ComponentObject.

Visualisation

The SpaceModel can be visualised as a wire frame model with or without hidden lines or as a surface model with coloured, shaded and shadow casting surfaces. The model can be shown in section of every orientation and in parallel, isometric or perspective projection. Each SpaceObject, or a selected group of objects can be separately visualised. When the SpaceModel and the ComponentModel and EntityModel are visualised together, the SpaceModel only shows where no components or entities has been modelled.

Analyses

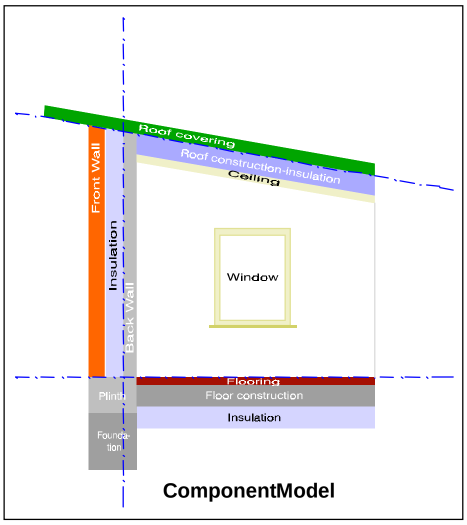
The highly integrated SpaceModel can generate data for a row of sketchy analyses:

- Economic estimates.
- Area overview and analyses.
- Thermal calculations, heat loss and gain.
- Static calculation, loads and stability.
- Acoustics and sound insulation.
- Time planning and logistic.
- Running costs, cleaning and maintenance.
- Ecological lifetime accounting.

Documentation

- Reports from sketchy analyses.
- Sketch drawings and visualisations.
- Superior, non-detailed specification of rooms, constructions, building elements and surface materials.

ComponentModel



Purpose

The ComponentModel is a detailing of the SpaceModel. The construction space can be divided into building elements or components and functional space can be equipped with inventory and furniture.

The ComponentModel and the SpaceModel are created simultaneously. Contrary to the common SpaceObject surfaces of the SpaceModel a ComponentObject has a discrete geometry bound by a set of surfaces called EnvelopeObjects. The envelope can be equal to or close to the outwards geometry of the component. It is related to SpaceObjects, SurfaceObjects or to EnvelopeObjects of other components. Placing a ComponentObject is regarded as a subtraction of the component's volume form the SpaceObject it is placed into. An envelope of a component that is related to a SurfaceObject will (in calculations and in visualisations) be subtracted from the surface. If a component is placed in a domain where no spaces have been modelled there will be a duplication of the envelope so that it will function as a SurfaceObject too.

A component or building element is described geometrically by its envelope and detailed by the objects of the EntityModel or by its own internal geometrical structure. This structure has to be known by the kernel system's calculation and visualisation methods or the object must have internal methods for that. The components must have the ability to have their shape and dimension defined by parameters.

Object Classes

The primary objects of the ComponentModel are the ComponentObject, the EnvelopeObject and the JointObject:

- The ComponentObject contains data about: type (Sfb classification table 1), specification (material, construction, product, key to cost, U-value, density, construction duration, maintenance cost, life time cost, etc.), visual appearance on sections (colour, raster), relations to the container SpaceObject, and relations to its own EnvelopeObjects.
- The EnvelopeObject has data describing, geometry: (type, placing and contour), visual appearance: (contour line, colour, texture). It can have references to SurfaceObjects, that control placing and contour, and will be related to the ComponentObject and neighbouring EnvelopeObjects. There can be relations to envelope's of EntityObjects These relations "course a hole in" the EnvelopeObject of the component, so that ,when the models are visualised, the entities will be shown where they are present. All these relations can also go through a JointObject.
- The JointObject is a relation between two EnvelopeObjects belonging to two components. The data in the JointObject are: distance (tolerance, thickness), material, colour.

Modelling

Modelling the ComponentModel is basically:

- Creating the ComponentObject, with data and relation to the containing SpaceObject.
- Graphical input of a single EnvelopeObject.
- Modelling the full component envelope or an extension to a component by extrusion, and relating the EnvelopeObjects generated hereby.

But the detailing of spaces with components has to be done with modelling tools, ProtoTypes, specialised for each different type of component or building element.

Relations

The relation between the EnvelopeObject and the SurfaceObject is guiding the placing and forming of the envelope eventually with a minor offset. Envelopes are related to other envelopes in the same ComponentObject through common contours. And an EnvelopeObject can be related to an envelope in another component, eventually through a JointObject.

Visualisation

The ComponentModel can be visualised as a wire frame model with or without hidden lines or as a surface model with coloured, shaded and shadow casting surfaces. The model can be shown in section of every orientation and in parallel, isometric or perspective projection. Each ComponentObject, or a selected group of objects can be separately visualised. When the ComponentModel is visualised, the SpaceModel shows where no components or entities have been modelled.

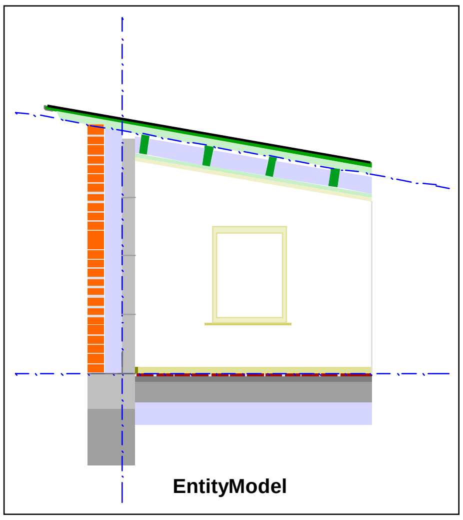
Analyses

The ComponentModel will be able to deliver data to the same analyses as the SpaceModel. The analyses are performed with greater accuracy as the components are modelled.

Documentation

- Reports from component based analyses.
- Drawings for the authorities and visualisations.
- Specifications for building regulations.

EntityModel



Purpose

The EntityModel is the final possibility of completely detailing the components of the ComponentModel. The entities fill out the component and have a set of enveloping surfaces that describe the outward geometry of the component precisely or nearby.
Placing an entity in a component can be regarded as a subtraction of the entity volume from the component and a visual take over. The EntityObjects can be interrelated via JointObjects. The entities must have the ability to have their shape and dimension defined by parameters.

Entity modelling will be used in parts of the building that are constructed on the site, by concreting, brick laying, covering, etc., whereas component based parts of the building are fully detailed by the component's internal geometry.

Object Classes

The primary objects of the EntityModel are the EntityObject and the two objects EnvelopeObject and JointObject defined for the ComponentModel.

- The EntityObject contains data about: type (Sfb classification table 2 and 3), specification (material, construction, product, key to cost, U-value, density, construction duration, maintenance cost, life time cost, etc.), visual appearance on sections (colour, raster), relations to the container ComponentObject, and relations to its own EnvelopeObjects.

Modelling - Visualisation – Analyses - Documentation

The EntityModel follow closely the ComponentModel in the definition of modelling, relations and visualisation. The main difference is that visualisation and analyses are more accurate.

Documentation covers a full detailed set of:

- Reports from entity based analyses.
- Working plans, sections, detail drawings and visualisations.
- Detailed specifications.

The GenoProject, as stated before, has no intention of imitating the traditional building project documentation, but is anticipating a situation where participants in the design, construction and facilities management process seek relevant information in the project object database.

MODELLING TOOLS

Modelling an integrated model effectively with creation, geometrically definition, specification and internal relations of ConstructorObjects, SpaceObjects, ComponentObjects and EntityObjects can only be done with highly specialised modelling tools, for constructors, for spaces, and for different component and entity types.

The specialised modelling tool, ProtoTypes, in the following called a Modellor, is an object class developed on the basis of GenoObjectClasses. The Modellors create GenoObjects and call GenoObject methods to write or read data values, relevant to the modelling process.

The Modellor is specialised for a specific part of the building, component, or element. Samples could be:

- Modelling functional spaces. Creating wall and ceiling surfaces by extruding the floor plan up to a Constructor. Or the same by horizontal extrusion of a wall surface.
- Modelling openings in a wall by drawing the opening profile on one surface and extrude it to the other.
- Modelling building elements in the construction space exterior wall. The space has to be divided into a front wall, an empty space and a back wall. The front wall has to follow the plinth and go to the base of the roof, say on the first floor. The back wall has to follow the plinth too, but is split where the floor division is.
- Mounting a window includes making an opening in the front wall, with room for the joint, and making an opening in the back wall with inclined side rabbets, extended to the front wall.
- Making piping for drinking water, with fittings, valves and cut through walls, etc..

- Covering a bath floor with tile. Lay out a board along the walls. Planning for slope. Choose lay-out with joint in the middle or tile in the middle. Experiment with board width to obtain less divided tiles, etc..
- Covering the roof with tile. Laying out felt underlay on the rafters, lathering, laying out tiles. Etc..

All known modelling techniques will be used, but adjusted precisely to the type of object.

IMPLEMENTING GENO PROJECT VERSION 1.

The following sketches what is to be implemented in Geno Project version 1., to try out and develop the GenoObjectClasses, described above. No EntityModellor will be implemented in ver. 1. as the technique of the EntityModellor is similar to the ComponentModellor. Before implementing an object it will be checked with IFC object classes to see whether there is some definitions that can be used directly, or to se whether it is necessary to prepare a suggestion for new IFC objects based on GenoObjectClasses.

ConstructorModellor

Ver. 1. will only have straight and plane constructors. It should be possible to establish constructor plane, line and point, with internal angular and “parallel with distance” relations. ConstructorObjects will be able to have a relation to the SurfaceObjects of the SpaceModel.

SpaceModellor

The SpaceModel ver. 1. is restricted to plane surfaces. The SpaceModellor include:

- Creating a SpaceObject directly in writing the space program or indirectly when modelling SurfaceObjects.

Space modelling can take the form of modelling a single SurfaceObject:

- That is, stating the two spaces and relations to constructors and/or other surfaces and drawing the surface parallel to a constructor or by snap to existing points.

Or modelling a full SpaceObject or space extension:

- By extruding a surface or constructor line to a given height/length or against a constructor plane. Followed up with automatic or semiautomatic relating of the surfaces generated by the operation to spaces, surfaces and constructors.
- Editing the SpaceModel by moving constructors or by changing relational parameters (angel, distance).

ComponentModellor

As mentioned the ComponentModellors are highly specialised to different building element types. In ver. 1. only a non-specialised generalised modellor is implemented with:

- Creation of the ComponentObject, stating data and relations to the containing SpaceObject.
- Drawing and relating a single EnvelopeObject or
- Modelling a full component or component extension by extruding an envelope plane or a constructor line to a given height/length or against one or more surfaces. Followed by stating relations.

Visualisation and Calculation

The principle of letting the ComponentModel overrule the SpaceModel in visualisations, if it is present, will be implemented in ver. 1..

A calculation of quantities of objects in the SpaceModel and the ComponentModel will be implemented in ver. 1..

System Development

AutoCAD ObjectARX used with Microsoft Developer Studio and Visual C++ has been choosed as development tools.

The time schedule for implementing GenoProject Version 1. Is:

- April-June. Learning programming tools.
- May-June. Detailed object specification and verification with IAI/IFC
- August-September. Implementation.

Exploring new concepts for the Construction IT System, as the GenoProject aims at, is a big task. But the integration with IAI makes a contribution to the development possible and ensure that the results will be used.

LITTERATURE

- Cesar Martinell y Brunet, Gaudi –his life, his theory, his work, Barcelona, (1967)
- Kristian Agger, Datastructures for function and geometrical description of buildings, paper, CAD76, London, (mar. 1976).
- Bryan Lawson, How Designers Think, Butterworth Architecture, (1980).
- Tom Søgaard Larsen, Projekt 90, rapport, PAR, København (1990).
- Laila Dybkjær, Computer-Aided Floor Plan Sketching –A study of Knowledge Representation in Architectural Design, Ph.D. Thesis, DIKU, København, (1991).
- Jan Karlshøj, Principper og metoder for opstilling af datamodeller til byggeteknisk anvendelse, rapport, DTU ABK, (1994).
- Informationsteknologi i byggeriet, -Fou-strategi, ATV, København (aug 1994).
- Per Galle, Product Modelling for Building Design, -annotated bibliography, DTU, Lyngby, (1994).
- Bo-Christer Björk, Requirements and information structures for building product data models, VTT, Espoo, (1995).
- Carsten Rode og Karl Grau, Implementation of a pragmatic system for integrated Building analyses, SBI, Hørsholm, (1996)
- ArchiCAD / GDL Reference Manual, Graphisoft, Hungary, (1997)
- Industry Foundation Classes, Release 1,5, IAI Members CD, IAI, (1997).
- Kristian Agger, Traditionel projektering eller projektering med IT, AbbNYT nr. 6 Taastrup (1997)
- Kristian Agger, IT modellering, -Status og Perspektiver, (Abb)NYT nr 1, Taastrup, (1998)