Howe, A.S. (1998). A new paradigm for life-cycle management of kit-of-parts building systems. (Doctoral Dissertation, University of Michigan, 1998). University Microfilms International, No. 9909905.

A NEW PARADIGM FOR LIFE-CYCLE MANAGEMENT OF KIT-OF-PARTS BUILDING SYSTEMS

by

Alan Scott Howe

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Architecture)
in The University of Michigan
1998

Doctoral Committee:

Professor James Turner, Co-chair
Professor Michael Parsons, Co-chair
Professor Harold Borkin
Professor Lynn Conway

To my ancestors who paved the way,
and to my posterity who will inherit the work.

ACKNOWLEDGMENTS

I would like to thank my wife Eiko who has been the unseen partner in this work, bearing through long hours and overnighters and almost singlehandedly raising our six children while I was mentally in other worlds. I value the support and input from the only real peer group I will ever know: my brothers and father who see the world as I do.

Thanks be to another family at Wolff / Lang / Christopher Architects for teaching me how buildings go together. My heartfelt gratitude goes out to Gordon Hashimoto, John Sugden, and Bob Hermanson for helping me rise above mediocre design tendencies. I am eternally indebted to Mikio Shoji for believing in me even when it wasn't necessarily the popular thing to do. I would like to thank Edward Smith for opening my eyes to research and for introducing me to the outstanding doctoral program at Michigan. And the doctoral program at Michigan! A world of opportunities at my fingertips was made available by the various departments and fields.

Finally, I would like to thank my real father, that eternal architect of the universe, who stood by me in his wisdom and power, and made all the machines work when my bungling would have ended in perfect failure.

TABLE OF CONTENTS

LIST OF FIGURES

Figure

LIST OF TABLES

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

This research introduces a new paradigm in design which attempts to integrate design, fabrication, construction, maintenance, and facility management processes. The paradigm is based solely on the use of pre-designed and pre-engineered kit-of-parts systems and depends entirely upon computer-based tools and a geographically distributed networking environment such as the Internet. The paradigm includes a real-time linkage of a virtual design environment to design information sources, manufacturing and construction environments, and the actual final product itself. Instead of static artifacts, the paradigm encourages the design of dynamic systems of standards and interfaces. In order to show the feasibility of such a paradigm, an experimental design environment is developed and tested. A series of case studies supports the approach for possible application of similar design environments.

1.1 Prelude

Increasingly, powerful computer-aided design tools have enjoyed greater roles in the design process. In parallel, the Internet with its World Wide Web has proved to be a revolution in information dissemination, providing real-time access to sources located around the world. Since information is the ultimate substance from which designs are conceived, a logical question could be: how can a design process be enhanced by direct links to information sources? Considering the increased proliferation of information-based automated manufacturing processes, a second question would logically follow the first: how would direct instantaneous access to manufacturing processes affect the design process? Finally, assuming instant access to performance data from the constructed object itself, how would an information feedback loop affect current use and future design improvements to objects of a similar type? While these questions will probably remain unanswered for many years, the development of an experimental environment which sets the stage for linking design, manufacturing, and use can be demonstrated.

The work discussed herein is mainly concerned with technology and design. Though the principles being applied are not necessarily new or novel in themselves, merging them together in the way described may be considered cutting edge. Though this work in itself is crude and preliminary, it paints a picture that is exciting to some and unsettling to others. How would our society change if entire processes from the procurement of raw materials to the final production of buildings and products were automated, not requiring the attention of a single person? Even though this work does not cover such a wide range, the small collection of processes which are associated with fabrication, assembly, and erection represent a sufficient portion to perhaps raise such questions.

This work does not attempt to answer doubts and concerns about the properness or implications of the technology, but only addresses the feasibility of it. Major inspiration for this work begins with writings from a previous century. Over a hundred years ago [Smith 1833] discussed principles of design which advocated the creation of systems rather than static objects. Individual elements in the system are given rules and bound by laws which define their behavior individually and interfacing with other elements. Smith used the term "spheres" to refer to these rule-defined realms. As long as the elements do not exceed their bounds or spheres they are allowed to govern themselves in a dynamic system. Smith also described hierarchies of systems, where collections of these element-rule sphere sets form higher orders that become spheres in themselves with their own bounding laws. In the author's literature search, this is one of the earliest references to object-oriented philosophy found.

Smith's philosophy has proved to be a profound influence on the author that inspired early ideas of self-governing kit-of-parts building systems. Other influences included observation of nature. As will be discussed later, these same object-oriented principles can be found in nature. Though seemingly antihumanistic, automation, robotics, and information technologies represent the very essence of humanism. Nature itself employs these principles. Even while maintaining an objective stance in the context of this work, the appropriate words of a historical religious figure come to mind. Two millenia ago Jesus Christ made a profound statement that relates to this discussion. He said, "Consider the lilies of the field, how they grow... even Solomon in all his glory was not arrayed like one of these" [The Bible, Matthew 6:28-29]. If we take his advice and consider a simple flower, with all its sophisticated mechanisms for pollination, opening and closing in response to sunlight (or the lack thereof), and the complex web of distribution and waste disposal infrastuctures. Each individual cell is a complete factory having the ability to construct copies of itself. Cell structure and overall design are dictated by systematic DNA coding. A simple lily of the field could undoubtedly be one of the most hi-tech machines to which the human mind has ever been exposed.

So what must we do, simply observe the lily and say it's pretty because it's blue? Which is the humanistic part, the fact that the appearance of the flower invokes emotion and appreciation for beauty, or the idea that such a simple thing exists at all and is able to propogate and perpetuate itself year after year, century after century? In this work, the author attempts to take some of these principles and bring them into the field of architecture and design, albeit in a much grosser form.

1.2 Purpose and scope

The purpose of this study is to encourage the acquisition and use of knowledge pertaining to the design of pre-engineered building systems, and establish a new paradigm for the design, fabrication / construction, and use of buildings and other artifacts based on an object-oriented approach to life-cycle management. The paradigm calls for the establishment of a virtual representation of the artifact which will be linked to the actual constructed object to affect lifecycle management. In addition, the paradigm calls for the virtual representation to be connected to real-time information sources via a network such as the Internet.

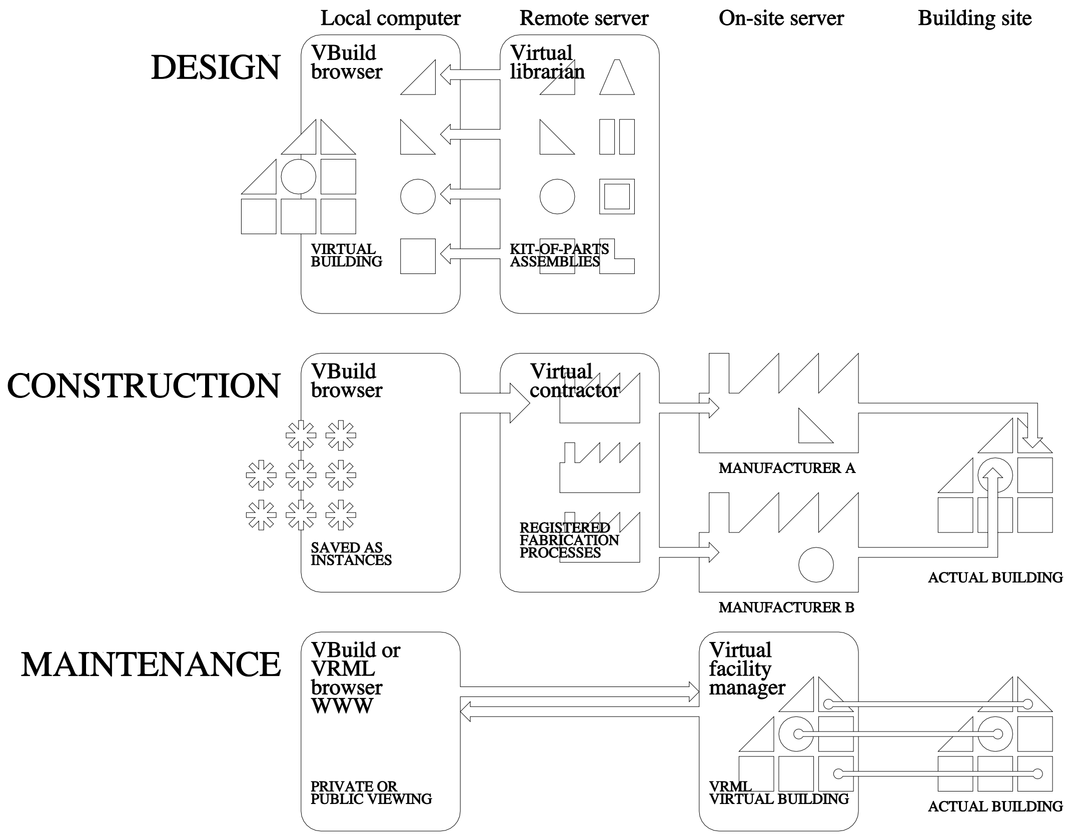


Figure 1: Design to construction to maintenance: life-cycle management

Designing dynamic systems rather than static artifacts provides a set of rules and behaviour that encourages easy redesign in the event of changing needs, and lends itself toward green concepts such as recycling and environment-friendly disposal (this is discussed later in conjuction with the Life Cycle Architecture System). Using an object-oriented approach, various factors, parameters, and environments are self-contained and isolated in a digital as opposed to analog fashion, simplifying research, aquisition of knowledge, and design and development of building and construction systems.

1.3 Research outline

This research begins with a study of various design philosophies which advocate the systemization of form generation. Chapter 2 provides an overview of the standardization of building components through prefabrication, form generation through design philosophies of the past such as *Metabolism* and *Dutch Structuralism*, and recent trends in the hi-tech design world. Brief description of pertinent design projects and investigations by the author are included. In chapter 3, technologies relevant to the current work are discussed, with short descriptions of experiments and simulations conducted in conjunction with this work by the author. Chapter 4 describes the computer framework of the VBuild browser and explains the various functions. In chapter 5 an experimental model kit-of-parts building system is described. This was designed, fabricated, assembled, and remotely controlled using the technology VBuild incorporates.

Chapters 6, 7, and 8 introduce case studies which reinforce this work. Chapter 6 discusses the Life Cycle Architecture System project as a case study of three different variations of a joint-based kit-of-parts concept. In chapter 7 a hybrid joint / module-based kit-of-parts system is introduced which applies the various technologies discussed to residential construction. Chapter 8 briefly discusses the application of kit-of-parts concepts and life-cycle management to other fields, including automotive design and naval architecture.

This work is concluded in chapter 9, where implications for architectural research, practice, and education are discussed, and directions for future research are outlined.

CHAPTER 2
KIT-OF-PARTS CONCEPT

A kit-of-parts is a collection of discrete building components that is pre-engineered and designed to be assembled in a variety of ways. When assembled, a kit-of-parts can define a finished building or single artifact. The components fit together according to rigorously designed interfaces which provide for flexible configurations. Components are sized for convenient handling or according to shipping constraints. Since a well designed component can be mass produced and used over and over again, fabrication processes can be worked out in advance for real-time manufacturing at time of need.

Following a brief discussion of design philosophies and building techniques that have contributed to the kit-of-parts concept, examples of designs that have been conceived in preparation for this work are described.

2.1 Base philosophy

The use of kit-of-parts technology is an important factor in this research. Many of the concepts are found in nature, and the object-oriented concept lends itself to applications that utilize new information technologies and maximize the possibilities for computer usage.

2.1.1 Nature’s kit-of-parts

As a design concept, the kit-of-parts philosophy is based on ideas found in nature, which can be called nature’s kit-of-parts system. One of the most basic of all building blocks in the system is the atom. An atom can be thought of as a very simple micro-electronic device which runs on electricity and light energy. Atoms receive and transmit similar to miniature wireless transceivers, communicating to each other by electromagnetic waves of disturbed media and orbiting and jostling each other in a sea of self-generated ether. They consist of two parts: things that act and things that are acted upon. The nucleus of the atom provides mass and substance which can be used to build things. Electrons are automated around the nucleus in an orbit or cloud and provide a means of intercourse with other nuclei.

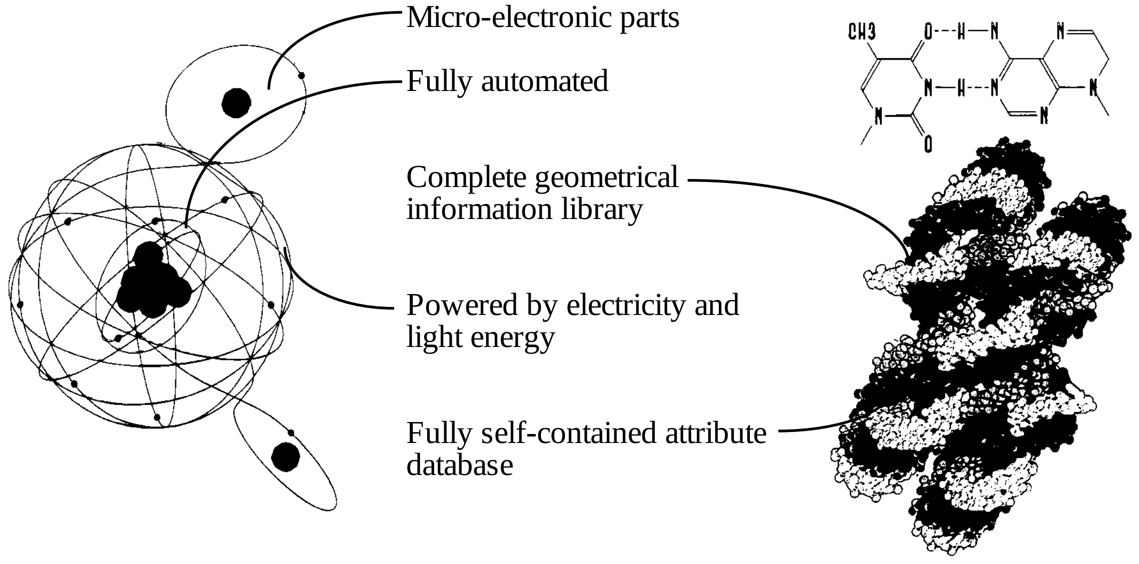

Micro-electronic parts
Fully automated
Complete geometrical information library
Powered by electricity and light energy
Fully self-contained attribute database
CH3
O
H
N
N
O

Figure 2: Nature's kit-of-parts

This very simple system, the most basic being hydrogen, is the kit-of-parts from which the entire universe as we know it is constructed. The universe is fundamentally digital in its very nature. Each electron, proton, neutron, etcetera can be thought of as being programmed to react a certain way when bonded with others of its kind. In this way the same atom responds to temperature differently when by itself or coupled with other atoms in molecules.

A key to life is DNA, which is an assembly of molecules. DNA is a virtual existence of a being, a plan for how the actual being will be shaped and constructed. The DNA contains coding which represents all the attributes and information required to describe the being and its functions. In possession of DNA, each cell "assembly" knows its function and also what it needs to know of all the other cells in the entire being. There are bounds set within which each cell is free to operate according to the instructions of the DNA. The actual shape of the cell is determined by the virtual shape held within the DNA.

In this very simple way, parts make up assemblies which make up entire systems, such as the body's structural system, mechanical system, exterior envelope system, and power and information systems, among others. The systems all work together in perfect integration for the well being of all. External influences from the environment, such as rain falling or vibration on the floor of a moving train are sensed by certain of the body's systems. Information is passed to the central processing unit (the brain) where data is assessed and reactions determined. Then automated machinery (muscles) carry out the reactions according to plan, via information received on the neuro network.

2.1.2 Artificial kit-of-parts

The most basic elements in buildings and other man-made objects, like atoms in nature, are its separate parts. One such part, a simple bolt, may have many, many attributes. Not only is its shape important, but the type and strength of the material, its cost, the manufacturer, the knowledge of how it fits together with other parts (such as a nut), its overall place in the whole system are all attributes of a bolt which allow it to be used as a part of the building.

A collection of parts with their associated attributes becomes an assembly, much similar to cells in the body. Many of these cells assembled together become an organ or system such as the building's skeleton (structural system), circulation system (mechanical), skin system (external envelope), and neuro network (power & information system) among others. Ideally the “organs” all work together in good integration for the well being of the building.

Kit-of-parts architecture involves organizing the millions of individual parts in a building into assemblies of standard easy-to-manufacture components. The construction of the building is generally carried out on the assembly level as opposed to the part level. The architect defines a parts library describing every major assembly in the building. The library could be similar to an interactive LEGO set, which has many standard parts already available but is open for additions by creative users trying to meet new needs. The assemblies are conceived in a systematic way, perhaps based on a certain increment or size or shape grammar. Standard connections between the assemblies are defined, allowing greater freedom in the form itself: anything is possible as long as the connection rules are observed. After rules of connection and increment are established, the number of possible shapes and appearance the parts could take is limitless. Rules of connection and increment may include requirements for materials, structural strength, force reaction paths, center of gravity, insulative qualities, transparency.

Kit-of-parts philosophy lends itself toward advanced manufacturing, and computer and information technologies. Handling multiple identical components as instances of a master element is an efficient use of the computer. In addition, spaces can also be divided up and thought of as components as well, where digital blocks of spatial volume have their own attributes and enclosing structure (similar to an atom which generates its own space around itself). Plug-in ducting systems and package HVAC units could be designed around zones based on the digital blocks of space.

In traditional construction methods, buildings are erected in what can be called “final line” construction, where raw materials, tools, labor and such are all gathered to the site and processed on the spot. The measuring, cutting, and processing of raw materials constitues the majority of labor in manufacturing and construction. Assembling these cut and processed parts into major assemblies such as walls or floors consumes even more time and labor on the building site. The use of prefabrication and kit-of-parts construction turns the entire building process into "assembly line" style, where many different manufacturing events can occur in parallel in safe, controlled environments, and the actual assembly on site of the resulting components is fast, clean, and safe.

2.2 Prefabrication

Kit-of-parts construction is a special form of prefabrication. The major difference between the two is that kit-of-parts components can be assembled and dismantled over and over again according to need. The majority of prefabrication produces a set of components that generally are permanently connected at the time of construction and can become damaged and unreusable when dismantled. Nevertheless kit-of-parts construction and prefabrication are similar in many ways and share the same history.

Prefabrication techniques arose from three basic needs that traditional construction methods could not meet: the need to create shelter quickly with little effort, the need for unusually large numbers of buildings at a low cost, and the need for periodic renewal. Since prefabrication lends itself toward the "assembly line" model more than the "final line" process, large amounts of identical or similar components can be mass produced quickly and cheaply. The "assembly line" technique allows the processing of raw materials to be performed in advance, thus reducing the amount of time and labor needed to erect the structures on site.

2.2.1 Early prefabrication attempts

Prefabrication and kit-of-parts construction has been around since the dawn of civilization. The first clay bricks of ancient civilizations prepared in advance with the intent of fitting them together on site fall well within the definition of prefabrication. To this limited extent, prefabrication has persisted in all cultures in some form or other, most likely due to the need for reducing the “bottle neck” of time and labor while waiting for bricks to cure or stones to be dressed. In ancient Japan wood kit-of-parts systems for the assembly of shrines produced a whole range of snap-together / take-apart joints that have been standardized over the centuries. This was partly borne out of the Shinto tradition that shrines be renewed and rebuilt every twenty years.

In more recent history the need for quick or movable shelter led to the use of prefabricated buildings. [Kelly 1951] described about how in 1624 English fishermen brought a panelized house to the Americas that was assembled and disassembled several times. Some use of prefabrication was used by settlers in the 1700’s to provide shelter immediately upon arrival to the New World. Cast iron was used in early attempts at prefabrication from around 1830 and resulted in the spectacular mass-produced Crystal Palace in 1851 by Joseph Paxton.

Certain historic events created the need for prefabrication. The California Gold Rush of 1848 stimulated a boom for prefabricated houses that were shipped from all over the world. The mentality of the time was not to create a lasting community, but to get in, mine the gold, and move on without worrying too much about substantial shelter. Though the end of the Gold Rush produced many ghost towns, the need for portable, quick, easy to assemble shelter necessitated the use of prefabrication techniques. Another historical event was the American Civil War. In 1861, lumber dealers Skillings and Flint sold a prefabricated shelter that could be erected in three hours. With the mobile military needing quick shelter, Skillings and Flint did much business with the Union armies.

By the early 1900’s prefabrication moved to more complex systems. [Sheppard 1946] describes prefabricated plumbing and service stacks that were completely standardized. [Diamant 1968] speaks of early British heart units that were prefabricated and placed on the site, whereupon the rest of the building was built around it using conventional or prefabricated construction techniques. In addition to panel systems and prefabricated structural members, [Bruce and Sandbank 1943] tell of entire dwelling unit-sized modules. One firm LeTourneau produced welded, all-steel houses in 1937 that were moved and placed in one piece, including the garage. Some LeTourneau houses were moved from plant to site by water, floating on their own bottoms. Le Corbusier experimented with manufactured modules that plugged into a main superstructure. Similar work by Frank Lloyd Wright attempted to link with the development of the mobile home which is so popular in the United States today [Swaback 1970]. In 1948, the Lustron Corporation began producing steel prefabricated houses in a surplus aircraft factory. The company was backed by a government loan and produced approximately 2,500 houses before folding in 1950. The enameled house modules could be washed down with a garden hose and never needed painting.

[Gloag and Wornum 1946] discussed early designs of package kitchens and baths that culminated in the development of Buckminster Fuller's Dymaxion House. In the world of prefabrication, Buckminster Fuller was clearly ahead of his time and his research has provided a great deal of basic concepts and knowledge that influenced Archigram, the *Metabolists*, and the hi-tech movement.

Today, prefabrication is a given. It can safely be said that there is probably no construction project that does not use some form of prefabrication, whether it is the use of precast or precured masonry units, precut lumber, or presized insulation batts (and other building materials). Since the use of these materials have become so common place, they are no longer considered to be within the realm of prefabrication. Today "prefabrication" means pre-cut or pre-assembled components designed to fit together in a certain way such that no other processing of the raw materials is necessary. In the simplest application, many commercial applications can be found that produce pre-cut logs, lumber, or metal parts prepared for mail-order homes and industrial buildings with follow-the-directions, do-it-yourself assembly, such as Neville Log Homes, Topsider

Building Systems, and Butler Buildings. In more advanced systems, entire modules and major assemblies are produced using mass production techniques, such as Sekisui, Misawa, or Toyota Homes. In Japan, almost the entire housing industry has come to use these advanced prefabrication and systematic assembly techniques. Those construction companies that do not have the capability for advanced prefabrication techniques are being pushed out of the market because they cannot meet competitive prices and time schedules.

Some innovative, long-lived kit-of-parts systems have come out of surprising corners of the construction industry. Construction support systems, such as container-type job site office structures and snap-together scaffolding systems have come to wide use. Japanese companies like Hory Corporation have developed adjustable span trusses and girders that can be used over and over again. Japan Steel has developed a series of deployable stairways that fold compactly for shipping and can be hung tens of stories off of scaffolding. European firms like Plettac and SGB International have special slip joints for scaffolding that can be connected or taken apart by hand and require no special tools. An Italian firm, Alpi, has designed an attractive but functional bolt-together system that can be used for scaffolding or for formwork construction. While the container job shacks are generally spartan and dull, Fuji Sash has developed a system with plug-in toilet or kitchenette capsules and snap-in cladding parts. Using certain window components, the modules can be made quite attractive.

2.2.2 Major prefabrication system types

Kit-of-parts and prefabricated systems fall into four main category types: joint-based, panel-based, module-based, and special construction which includes deployable and pneumatic inflatable structures. With exception of the special construction types, the systems have each evolved out of traditional construction methods. In some cases it is even hard to distinguish the difference between traditional and prefabricated structure, since the only difference is where and how the raw materials were processed.

JOINT-BASED: On the simple end, a common prefabrication technique is to pre-cut materials for later assembly and construction. These individual pieces may look the same as their traditionally prepared counterparts, except that they are prepared in advance. More advanced applications which fall into the joint-based category are systems which have clear distinctions between the members and joints, and often celebrate the joint with some special design or connection technique that either enhances the ease of assembly or speeds erection time. Upper-end joint-based systems are often moment frame structures that connect together via a systematic connector and receive lateral support from fasten-on panels or diagonal braces. These upper-end systems do not have a structure which is integral with the enclosure or cladding, but have exposed structural members which support the cladding system at minimal points. A local example of this is curtain wall construction, where the curtain wall contains its own self-supporting structure that hangs from the overall building structure by only a few support points. Highly refined joint-based systems are good candidates for kit-of-parts concepts since the connectors are so clearly defined and can be designed for easy demounting as well as easy assembly. Joint-based systems are easy to represent virtually in a computer. A well designed joint-based system can also lend itself toward automated construction techniques (discussed in later chapters).

PANEL-BASED: Some of the earliest prefab systems were panel-based. In the West where shear panel construction gained popularity (as opposed to wood post and beam construction in Japan and other Eastern countries), prefabricated panel concepts naturally evolved. The first systems were wood frame, but today materials range from wood to metal to precast concrete. Panel-based systems essentially incorporate structure and wall / floor cladding and decks into one-piece assemblies. The collection of individual parts become a discrete component which works as a single structure or cladding member. Upper-end panel-based systems also often have special joints which ease the construction process. Panel-based systems can also be good candidates for kit-of-parts concepts as long as joints and connectors are designed properly for disassembly as well as assembly. Panel-based systems are also easy to represent geometrically in the computer. In general, components in panel-based systems can be bulkier, but fewer in number than joint-based systems. For this reason, panel-based systems may be more suited for automated construction technology than joint-based systems. Of course, this would depend on how well the system was designed for ease of automated assembly.

MODULE-BASED: Some of the most advanced prefab systems are module based. Modules are entire portions or blocks which are assembled in advance and set into place at the site. Because of the size and scope of each component, the number of necessary modules required in a construction is usually much less than panel or joint-based systems. As was pointed out earlier, module-based construction can represent an entire self-contained building with a single unit. This would be impossible with panel or joint-based systems. Module-based systems are also ideal considerations for kit-of-parts concepts, and might be the easiest to represent in virtual form. Considering large blocks or modules, designing appropriate robotic construction systems for site assembly would be even more natural than those designed for panel systems. Modules themselves could be manufactured in off-site factories using similar technology as that which the automotive industry uses for the assembly of cars. In fact, Toyota’s housing division already uses automotive-style assembly plants for their housing modules, as does Sekisui, Misawa, and other Japanese housing manufacturers. The use of pre-manufactured modules can decrease construction time. Sekisui and other Japanese firms can construct a home in a single day and complete finish work in a week -- a process that traditionally takes months.

SPECIAL CONSTRUCTION: Special construction systems consist of deployable structures (including folding trusses and swing-open modules), and inflatable structures.

[Candela et.al. 1993] describes research in deployable folding truss systems. For the purposes of maintaining a compact and / or lightweight profile for shipping, various ingenious truss designs, domes, space trusses and folding vaults are discussed. The appropriate use of these structures is often difficult to define. Since they are so perfect and balanced, and the removal of even a single member destroys the integrity of the entire frame, it is difficult to provide openings for entrances and other holes. Still the technology has great potential for quickly deployable structures as is called for in many kit-of-parts construction environments.

Another form of deployable structures may also be classed as furniture. [Ambasz 1972] introduced a group of Italian designers that have produced swing-open modules and capsules that range from self-contained structures to package kitchen units. Alberto Seassaro created a “central block” in 1968 that is a simple cuboid when folded up, and when deployed functions as the core of a residence. Bed, table, wardrobe, toilet, and shelves all expand from the cuboid. Joe Colombo designed a similar structure that included kitchen appliances and entertainment electronics such as television. Another of Colombo’s designs was the Mobile House, which was a compact package that could be transported on the back of a truck and dropped off on the site, to expand to two or three times the stored size. In yet another project Colombo devised a series of block capsules with different functions that could be stacked or rearranged beside or on top of each other to form a custom dwelling. The capsules were targeted for use during disasters when emergency housing becomes necessary, or for use by workers on large scale construction and civil works sites. This form of deployable capsule concept was the basis for the Qamel specialty vehicle project which is discussed in later chapters.

Other examples include military barracks, bridges, and other structures which are meant to be deployed, stored, and relocated in a minimum amount of time. With constantly moving front lines during war, the movement and housing of troops and the maintenance of supply lines has necessitated the development of compact folding bridges and structures that can be transported on the back of a truck.

[Herzog 1976] discusses the development of inflatable modules and structural systems which also are designed to be lightweight, compact, and portable during shipping and storage, but expand to appropriate-sized volumes when inflated. Some interesting projects include inflatable tube bridges which are quite compact for shipping but are sturdy enough when deployed to become a footbridge across lakes and streams. Inflatable structures in themselves can be considered kit-of-parts systems, but are especially so when combined with joint, panel, or module-based systems.

One final type of special construction kit-of-parts could be hybrid systems which include two or more of the main types mentioned earlier. In reality, hybrid systems are the superior of the lot. Though panel-based systems can stand on their own, joint-based systems almost always include some form of panel hybrid mix. The use of core modules for difficult construction such as restrooms or kitchens, with panel-based systems filling in the rest of the structure is quite common. Hi-tech architects (discussed later) have combined the use of joint-based systems with inflatable or membrane systems for extremely sucessful buildings.

2.3 Kit-of-parts and design

Using prefabrication and kit-of-parts techniques as a superior construction technique has its advantages, but using the methods in an aesthetic way may be a little more difficult. In the United States, the mobile home industry is flourishing. Mobile homes are perfect examples of module-based construction. Still, very few design professionals would agree that mobile homes have any architectural aesthetic value. Just using the technology is not enough. The kit-of-parts theme must be taken as the center of design investigation in order for it to have any merit in design methods research. Since before the 1930’s manufacturing, prefabrication, and kit-of-parts concepts have occasionally been the subject of excellent design-related research. Starting with the Bauhaus in Germany and work by Le Corbusier and others, the foundations of the Modern movement was partially based on this theme. Though this chapter is not intended to be a historical review of such work, some discussion of various movements and their contribution to the kit-of-parts concept is in order.

2.3.1 Metabolism and Structuralism

[Jencks 1971] discusses some early attempts at linking good design investigation with kit-of-parts systems. In 1950, Charles Eames designed his own home almost entirely from catalog parts. By the fact that the catalog parts existed in the first place, the possibility of designing a structure using them was not unheard of. What was exceptional was that the home had the quality and beauty one would expect in a custom built home. Such a result can only come through rigorous design investigation, which is mostly lost on modern manufactured homes. The Eames home took commonly available consumer products and used creative expression to transform them into something exceptional. In the same post-war era, other designers were interested in linking the consumer with commonly available building products. The Plug-in City by Peter Cook in 1964 proposed an environment that recycled itself based on consumer needs. Self-contained pre-manufactured modules could be “clipped-on” to a basic structure and dissected or discarded when their need no longer existed. The Archigram group, of which Cook was a member, had radical ideas, and their design research with plug-in modular building concepts provided inspiration to designers around the world.

One of the most important design philosophies to employ the use of module-based kit-of-parts building concepts was the *Metabolism* movement. Starting in 1960, Kenzo Tange led a group which began defining architectural volumes and building systems conceptually based on natural systems which occur in nature, such as the cell, skeletal structure, and circulation systems. Traditional Japanese design concepts such as “ma,” which means the void or space in between, combined with new ideas about prefabrication and industrialization produced exciting new design investigations. The *Metabolists* not only explored module-based building systems, but considered the spatial volumes themselves as celular modules that could be added for expansion in any direction as needed. Tange’s plan for Tokyo in 1960 defined a spine bridge that spanned Tokyo bay that had cellular modules of city structure growing out of it in the middle of the bay. The modules were made up of smaller modules, which were made up of smaller units, on and on in the *Metabolistic* way.

[Ross 1978] discusses two Tange projects based on a *Metabolistic* kit-of-parts: a cylindrical core column and plug-on floor modules. The projects were based on a design investigation which envisioned a system that could expand and fill space in any direction. The projects were the Yamanashi Communications Center of 1966 and the Shizuoka Press and Broadcasting Office of 1967. Similar designs were proposed by Arata Isozaki, Kisho Kurokawa, and others.

[Kurokawa 1977] explained the *Metabolism* movement’s use of the capsule through his Nakagin Capsule Tower of 1972 and other projects. The Nakagin capsule apartments are still in high demand to this day. The Takara Beautillion erected for the Expo ‘70 in Osaka, Japan was a prime exemplar of using a skeletal structure to define space (or inversely, to be defined by modules of space). The system used a truss which was fabricated from 90 degree bent pipes, where the bends welded back to back formed an orthogonal node with the legs reaching outward. The legs, when bolted to legs of other trusses, became beams and columns in a volume-generating cuboid. The legs that were not engaged reached out in space as if waiting to clasp future truss additions. Unfortunately this wonderful example of the *Metabolist* movement was dismantled after the expo ended. Still, the Takara Beautillion remains one of the best examples of a module-based kit-of-parts, and can be considered an exemplar of how refined and extensive design investigations have gone in regards to kit-of-parts research. Designed by the author, the Virtual Building I project discussed at the end of this chapter was partly inspired by Kurokawa's Takara Beautillion project.

Another post-war movement was the *Dutch Structuralism* movement. As explained by [van Heuvel 1992], *Structuralism* also dealt with modular volume generation, and resulted in exciting spatial transitions between interiors and immediate surroundings. Indeed, the term "complexes" may be more appropriate than "buildings." In the early days, a group of architects worked together as editors for an architectural periodical called *Forum*. Frustrated with unliveable row houses, *Forum* members Aldo van Eyck and Herman Hertzberger took several matchboxes and stacked them in a row. By offsetting the arrangement, a zigzag structure resulted, which naturally gave birth to private corners. Taking the idea to multiple levels, a dynamic structure with private terraces, gardens, and exciting volumes developed. These ideas were published and gained much popularity in the Netherlands. Some outstanding examples include Hertzberger's Head offices Central Beheer of 1972 which had dynamic interiors and exteriors, the skeletal box structures cantilevering in some areas and protruding in others. Another exemplar is PEN offices of 1982 by Abe Bonnema which took a sloped shed roof element and effectively repeated it in a dynamic composition. Structuralism's main characteristics included modular blocks of space that were defined by bold concrete structure systems. Wall and floor panels were inserted within the structure as needed, leaving the impression of expandable cells.

[Moshe Safdie 1970] presents his Habitat and other projects as an impressive use of *Structuralism-like* methodology using actual kit-of-parts apartment modules. The modules themselves were precast and finished off site, then lifted into place in a *Structuralist-type* dynamic supervolume. Though quite an expensive project, the attractiveness of the design has made the Habitat an exemplar of kit-of-parts construction applied as a central theme in rigorous design investigation. To this day there is a waiting list of would-be tenants.

The efforts of this era were widespread. In 1970, the United States Department of Housing and Urban Development sponsored Operation BREAKTHROUGH which attempted to provide an arena for research and development in design and construction techniques in residential construction. In response to the program, over a thousand firms participated, 244 of which submitted proposals that dealt with the design, testing, and evaluation of complete housing systems. Though most of the proposals had some form of prefabricated construction, there were some notable examples that were module-based kit-of-parts systems likely influenced by movements such as the *Metabolists* or *Structuralists*. Alcoa and others proposed a high-rise core to which fully premanufactured concrete dwelling modules could be lifted and connected into place. The Commodore Corporation, National Homes, and others devised systems that resembled mobile homes stacked in multistory configuration in the *Structuralist* style. Moshe Safdie participated in the program with a proposal for bilevel concrete modules stacked similar to the Habitat. Spuntech, a company that had its beginnings at The University of Michigan and the sound research of the Architectural Research Laboratory (ARL), proposed a module-based system that consisted of cocoon-type spun fiberglass capsules which borrowed technology from the aerospace industry [Paraskevopoulos, Borkin et.al. 1968]. The capsules could be used as single family dwellings or cantilevered from a central structure for highrise construction (see Capsule Hotel later in this chapter).

The *Structuralism* movement has a modern day grandchild called the *Skeleton / Infill* (SI) construction concept. The SI concept started as early as the 1960’s when John Habraken talked about using permanent superstructures with less permanent infill in mass housing schemes. Delft University has done much research on an SI concept called Open Bouwen Ontwicklungs Model (OBOM) which has a permanent concrete superstructure and a joint or panel-based kit-of-parts system making up interior partitions and other infill. The Japanese have joined in and applied centuries of wood kit-of-parts construction culture to several modern SI projects. The Kodan Experimental Housing Project (KEP) of 1974 utilized an exterior fixed wall or shell with catalog prefabricated parts for the inside. The Century Housing System (CHS) of 1980 called for a systematic replacement life of various elements in the residence, which meant that elements with longer life spans could not be damaged when short lifetime elements were replaced: a form of demountability. The Next21 project sponsored by Osaka Gas in the 1990's provided a concrete framework within which several architects were given commissions to design different flavored residences using modular rules.

Today an international research group, Intelligent Manufacturing Systems (IMS), continues SI and kit-of-parts module-based research in the IF7 project, of which the author is a member (discussed in later chapters). The IF7 project attempts to address the application of automated construction technologies to the object-oriented world of kit-of-parts concepts, using rigorous design investigations and design methods research.

2.3.2 Hi-tech movement

The *Hi-tech* movement is a special group of architects that, through extensive design investigations and research, have brought the aesthetic of the building to the point of celebrating the kit-of-parts / manufactured concept in its very appearance. Inspired by Buckminster Fuller, Archigram, the *Metabolists* and others, *Hi-tech* was born as a British movement. It is often characterized by exposed structural and mechanical systems, and plug-in utility modules, surrounding a simple orthogonal plan. The emphasis was not on designing spaces as much as allowing for flexibility. On an imaginary scale of "space-generates-structure" on the left and "structure-generates-space" on the right, Hi-tech would be all the way to the right. *Hi-tech* epitomizes the combination of design investigation and kit-of-parts construction technology. Among *Hi-tech* architects, the cardinal rule of kit-of-parts design for demountability was strictly observed in appearance if not in functionality.

[Davies 1988] explains that the *Hi-tech* movement started out slowly and painful, but finally received needed momentum when Richard Rogers and Renzo Piano won the design competition for the Pompidou Center in 1977. From that time the movement became world class, with the four major players (Rogers, Norman Foster, Nicholas Grimshaw, and Michael Hopkins) producing a wide range of impressive works. The group also had peripheral members such as Piano from Italy [Lampugnani 1995], with works that included ships and cars in addition to buildings, Helmut Schulitz of California, and many other exceptional architects and designers. Highlites in the movement include Rogers' Lloyd's of London and Foster's Hong Kong Bank, both in 1986.

Though the main movement insisted on the use of glass and steel, Piano also managed to use finely worked timber on many projects including the IBM Traveling Technology Exhibition of 1984-1985 [Brookes and Grech 1990]. Materials and manufacturing methods research was common among the *Hi-tech* group, and often the architects went straight to manufacturers in order to achieve custom results on thier exacting kit-of-parts standards. [Russell 1985] explained how Rogers experimented with different types of glazing designed in-house. Rogers worked closely with manufacturers in the castings development of many of his projects.

Though the *Hi-tech* aesthetic give the impression of plug / unplug assembly and demountability, in reality it was not always that simple. Many times, such as in the case of Lloyds and Hong Kong Bank utility modules, the appearance of unplugability was preserved, but the actual capability was never achieved. The actual technology of *Hi-tech* was never really high technology at all, but only sought for the appearance of it in an industrial aesthetic. The real high technology projects like those designed by Jan Kaplicky of Future Systems rarely got built do to lack of interested clients [Pawley

1993]. One example of *Hi-tech* that actually became a standard mass-produced kit-of-parts is Michael Hopkin's Patera Building System designed in 1981. The system, which could be ordered in different sizes, was never really popular because of the high cost.

Built or unbuilt, real or apparent, cheap or expensive, the important point is that structured design investigation and research was applied to kit-of-parts concepts as a central theme and produced exceptional results. The manufactured home which blindly used superior prefabrication technologies without doing the necessary design investigations may have given kit-of-parts and prefabrication a bad name, but the *Hi-tech* movement may have helped recover the reputation, even if it is not as fully mass-produced as it tries to convey.

It is hoped that proper research and design investigations may provide knowledge and proof of feasibility so that low-cost, high-profile kit-of-parts design and manufacturing can actually be achieved.

2.4 Previous kit-of-parts design research

From 1985 to 1994, a series of architectural design projects conducted by the author presented opportunities to incorporate kit-of-parts design philosophy into practical and conceptual building solutions. Though the number of projects designed were well over a dozen, a selected number are described in this work. As exemplars, the Experimental Media Center, Prototype Dwelling, and Dwelling & Workplace projects (all 1987) were conceived as conceptual design solutions as part of the Master of Architecture course at the University of Utah. Urban Slalom (1991, 1992), Commercial kit-of-parts studies (1993), Virtual Building I (1993), and Expo Pavilion (1993) projects were designed through practice working with Kajima Corporation in Tokyo, Japan. The Capsule Hotel (1995) was completed while doing research at The University of Michigan.

2.4.1 Experimental Media Center

The Experimental Media Center was a 16000sf building devoted to experimental film production located in Reservoir Park, Salt Lake City, Utah. Six stainless steel core towers were the only permanent structure in the building. Structural steel space frame floors and cladding were conceived as a pre-manufactured joint-based kit-of-parts system that stretched between the towers. Three of the towers functioned as vertical circulation, one tower for mechanical services, and two for video projection and multi-media.

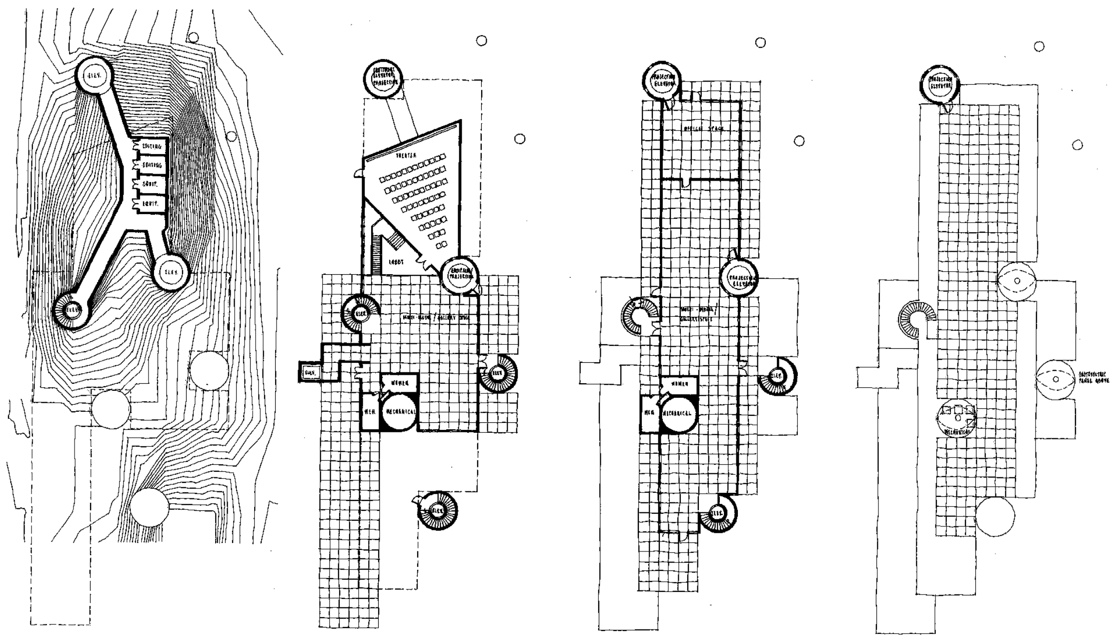


Figure 3: Experimental Media Center floor plans

In addition to the six core towers, an additional network of smaller stainless steel towers expand into the park area to allow a flexible structure for screens, mists, or projection and laser equipment. This additional facility was for the purpose of community involvement. The kit-of-parts structure was designed to be easily assembled and demounted using a minimum of labor and simple tools. The entire building was to function as flexible studio space that could be reshaped and redesigned by the film industry students and professionals that use the building, according to requirements of film production.

Research and design investigations included space and volume studies, full-sized component mock-ups, and computer-based design exercises. In one mock-up, full-size joints were bolted to structural members to investigate the assembly of the mountable / demountable space frames.

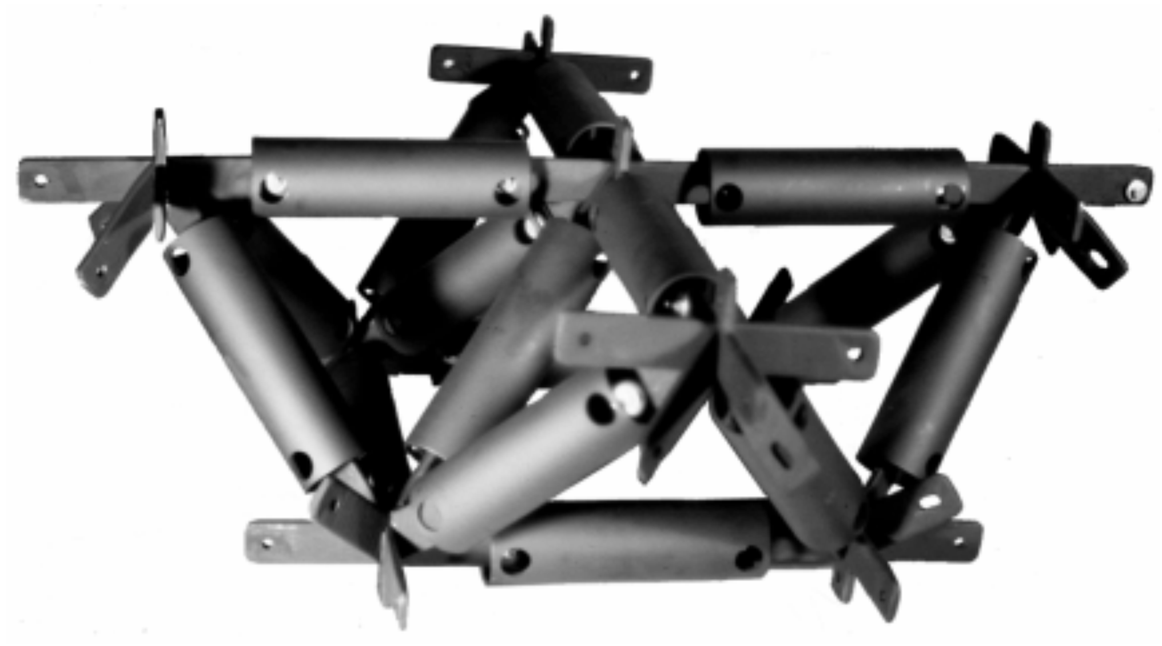


Figure 4: Media Center truss mock-up

2.4.2 Prototype Dwelling

The Prototype Dwelling was a hybrid module / panel-based kit-of-parts system intended for use in low to medium density (25 units / acre) apartment complexes. Three module types were designed: a core module that included bath, kitchen, laundry, and entrance halls; a space module that could be adapted to various purposes; and a stair module for exterior vertical circulation. The modules were square and could be arranged adjacent to each other in a variety of configurations. Used as an entrance, one wall of the core module was connected with the stair module. Space modules could be placed adjacent to the other three walls, or connected to other space modules to produce hundreds of variations of floor plans. A two-bedroom dwelling could be set up nine different ways, with the possiblity for single, three, and four-bedroom apartments as well.

The modules were skeletal in nature, with a panel-based kit-of-parts system that could be snapped into place to form walls, windows, doors, and louvers as needed. A welded steel roof pan component was devised that could hold soil for roof gardens and landscaping. The overall configuration of the modules followed a *Structuralist* view that demonstrated a dynamic play of interior / exterior spatial arrangments. Multi-floor apartment complexes became mountain-shaped with spacious roof gardens and lawns possible at all levels.

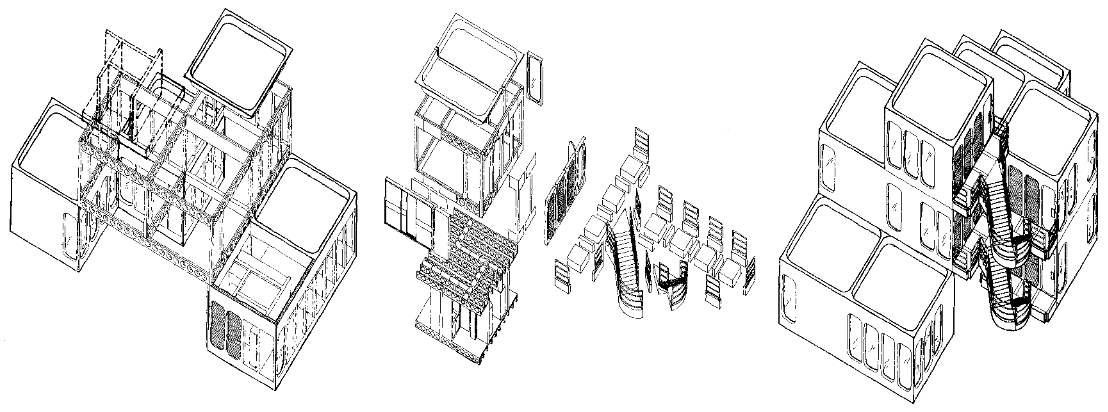


Figure 5: Prototype Dwelling assembly

The research completed for the Prototype Dwelling project included design investigations for module dimensions, panel systems, and structure systems, as well as detailed joint and seam studies.

2.4.3 Dwelling & Workplace

As the name connotates, the Dwelling & Workplace project involved devising a residence type that combined apartment / office units catering to professionals and small businesses. Nine apartment units at 1600sf each and office units at 2000sf (making a total of 32400sf) were grouped into an apartment complex on a high alpine mountainside overlooking Emigration Canyon in Utah.

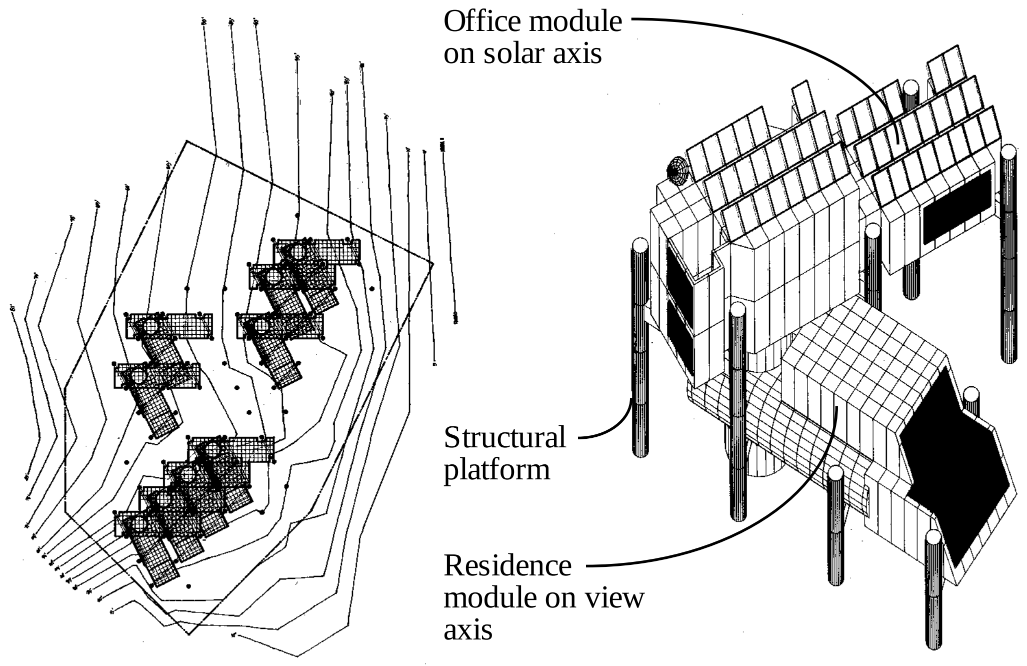


Figure 6: Dwelling & Workplace

The project involved the development of a hybrid module / joint-based kit-of-parts system that could be constructed with minimal damage to the mountain site. A skeletal post and beam structural system was devised that cantilevered out and provided elevated platforms for the placement of apartment or office modules. A single cylindrical core linked the two modules as vertical circulation. The skeletal structural system was designed such that the apartment module could be oriented on an optimum view axis, and the office modules could be oriented on a solar axis due to the arrays of solar electric and water heating panels placed on the roof. The structure system also allowed for slope, where units were stepped higher up the mountain.

The research conducted in the Dwelling & Workplace project included extensive geometric and volume studies tailored to sloping sites in order to generate an optimum kit-of-parts system. Additional studies included design investigations for the purpose of fitting residential and office functions into the modules.

2.4.4 Urban Slalom

Conceived as the next generation facility after the Spring / Summer / Autumn / Winter Snow (SSAWS) facility in Chiba, Japan, the Urban Slalom project was a 63,600m2 (684,700sf) indoor ski resort (including commercial and residential facilites) to be located on a landfill man-made island in Osaka. Early designs called for a figure-eight artificial ski slope supported by 20 masts.

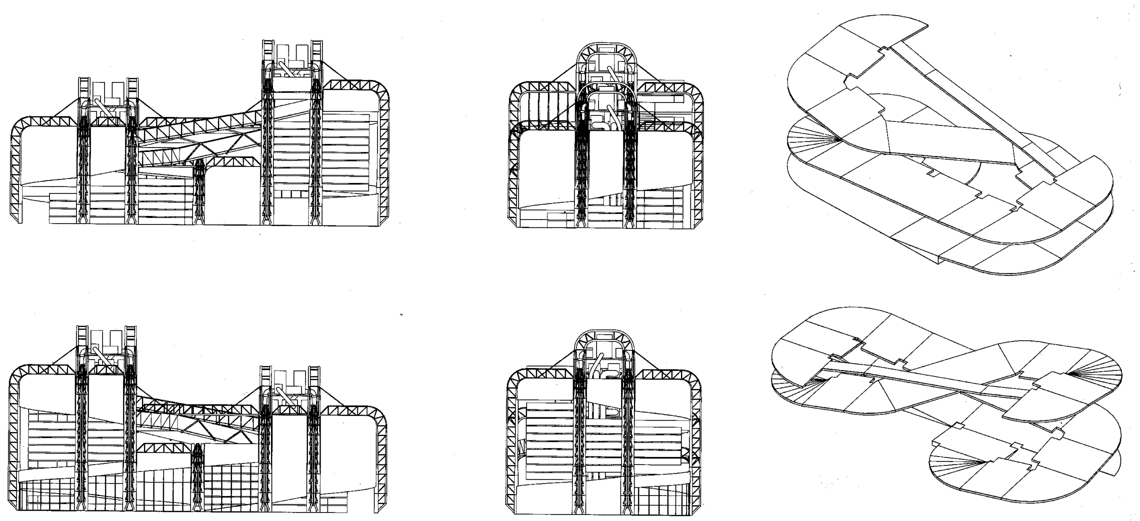


Figure 7: Figure-eight indoor ski slope (early version)

Eight masts formed a pair of core structures that housed the unusually large amount of mechanical equipment needed to turn the facility into a “mountain-sized refrigerator.”

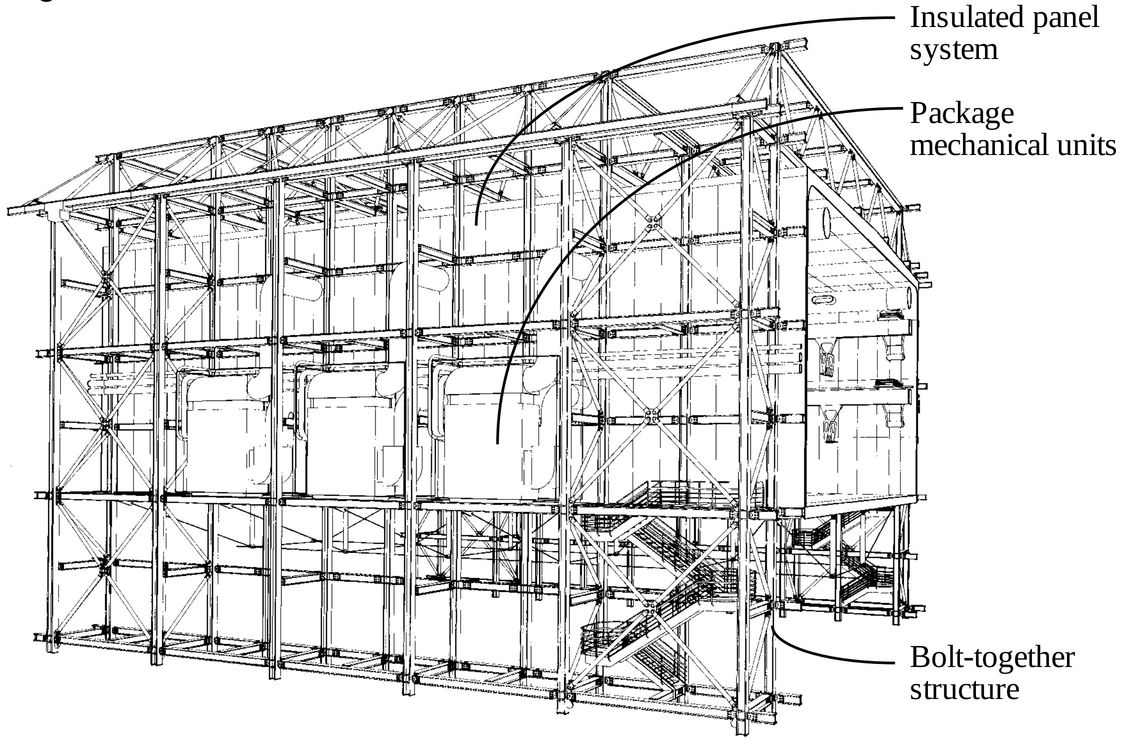


Figure 8: Indoor ski resort construction (later version)

Later versions had a U-shaped artificial ski slope suspended between two sets of truss walls that were constructed from a joint-based kit-of-parts steel structure. Research investigations included extensive joint studies for steel wide-flange construction and tension members, and integration of mechanical equipment and highly insulated cladding systems.

2.4.5 Commercial kit-of-parts studies

A series of kit-of-parts system designs were investigated for a chain of men’s clothing stores (Aoyama) and department stores (Jusco). During the course of the investigations two major systems were developed. In one system, which was a joint-based kit-of-parts, a special low-cost joint was devised that consisted of eight cast sections. When assembled, tubular columns and trusses could be plugged into the joints. Roofing / cladding panels, floor decks, and handrails were simply bolted to the columns and trusses via metal straps. Adjustable foundation pads which the system plugged into were designed to be simple enough to sit on hard, untouched soil for temporary constructions, or set in excavated holes for more permanent solutions.

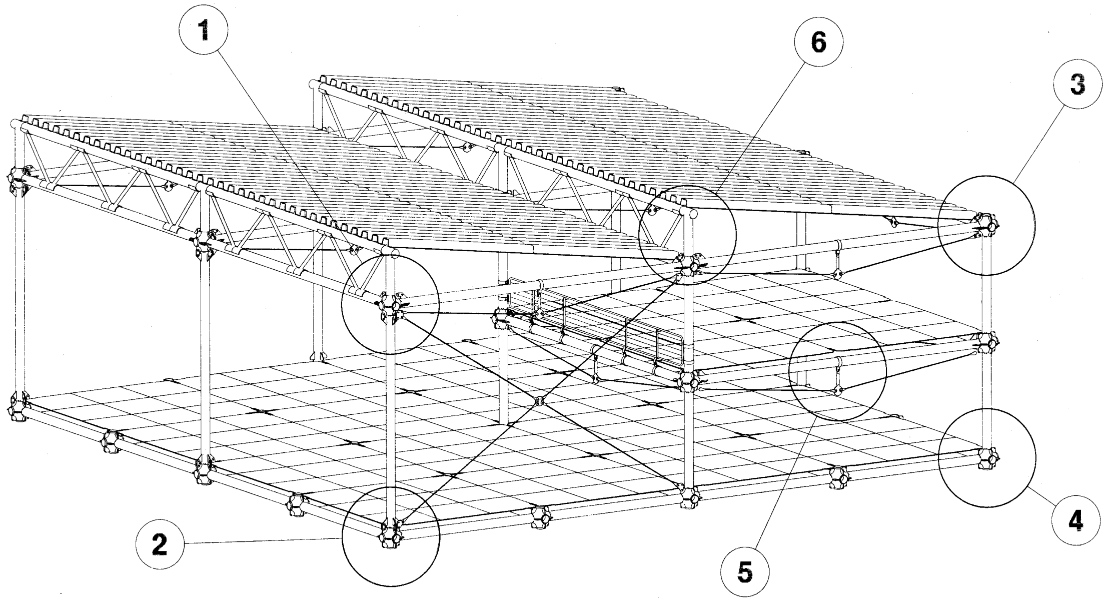


Figure 9: Tubular frame kit-of-parts

Used as is, the system was intended to be a low cost solution to chain store construction. As a slightly more expensive alternative, the hollow joints could be fitted with manual or electrically operated dampers, where the columns and beams became reroutable ducting systems. Lightweight heating, ventilating, and air conditioning units could simply be plugged into the nearest joint and after closing and opening dampers entire heating or cooling zones could be established. The columns were integral with small slots that could be fitted with rubber plugs, or removed to act as a return air / supply air register.

The second system was a module / panel-based kit-of-parts system based on a unit increment similar to shipping containers. Columns double as roof drains. Standard frames can be stacked in an infinite variety of arrangements and wall panels, glass panels, roof panels, and stair units are snapped into place as required. The modules were proportioned 1:3 to allow offset stacking in a Structuralism space arrangement.

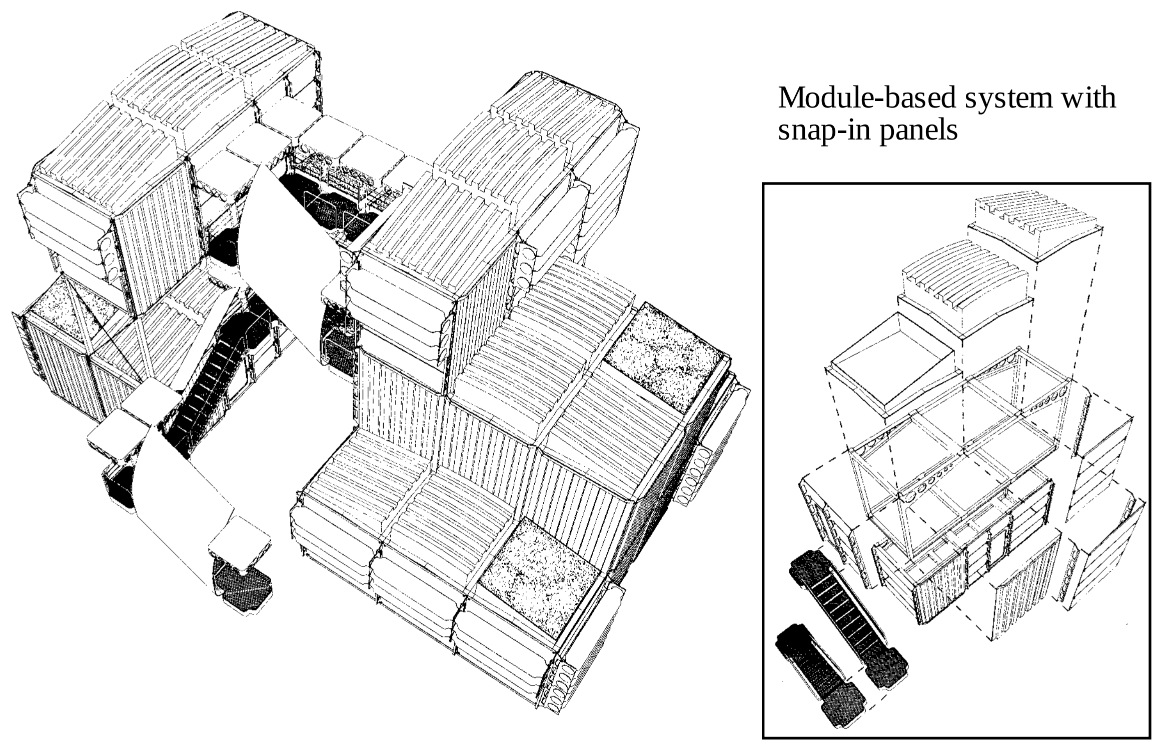


Figure 10: Container-based module kit-of-parts

In both systems (along with other minor systems and variations), research included design investigations for joint manufacture, structural / mechanical / cladding integration, and volume studies.

2.4.6 Virtual Building I

The Virtual Building I project was a major research effort to conceptualize the entire life cycle and virtual computer representation of a high-profile design, kit-of-parts building system. The Virtual Building I project could be considered a preliminary foundation to this current work. Discussions about computer representation of the Virtual Building I project will be conducted in later chapters.

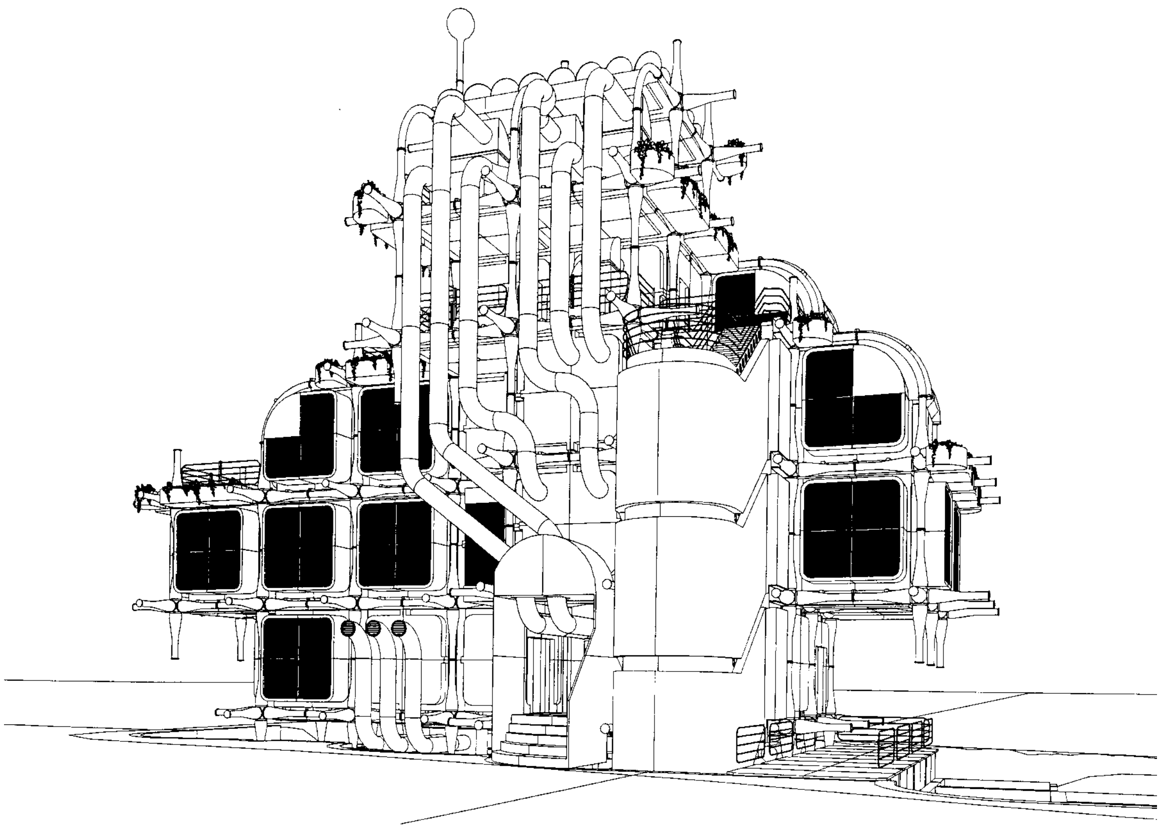


Figure 11: Virtual Building I

The Virtual Building I project was a conceptual 819m2 (8,800sf) office / commercial / residential building that could eventually expand an additional 1,300m2 (14,000sf) during a building's life cycle (or contract as needed).

The project was conceived as a fully integrated system along the entire life span of the building, from conceptual design to demolition. Conceptually the building is first designed and fully developed within the computer as a virtual building. The virtual model is used to run robots and construction machines to assemble the building on the site.

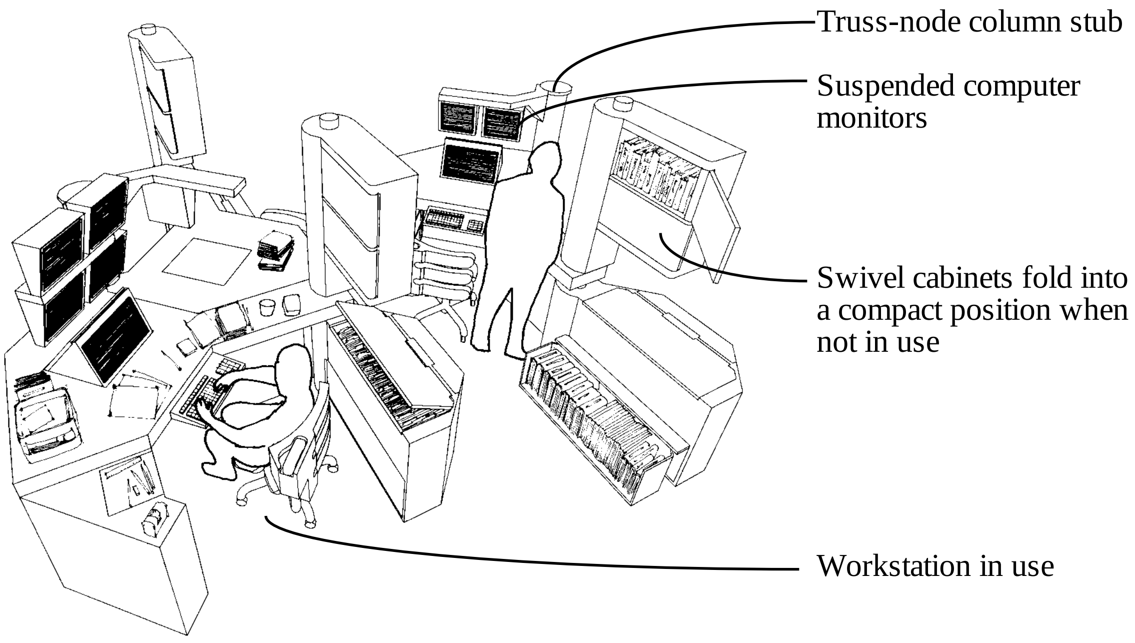

Truss-node column stub
Suspended computer monitors
Swivel cabinets fold into a compact position when not in use
Workstation in use

Figure 12: Virtual Building plug-in office furniture

All parts of the building have computer chips and various sensors that plug into factory assembled trusses. The trusses are single unit nodes similar to the structural elements Kisho Kurokawa developed for the Takara Beautillion project in 1970 as mentioned earlier, and act as a basic increment for form and space generation. The trusses, with their embedded computer chips, form an integrated network that could be governed through the virtual building the entire life span of the building. The kit-of-parts systems developed during the exercise were hybrid module / panel / and joint-based.

Research for the project included design exercises for the development of kit-of-parts systems, geometric computer representations, product modeling studies, and designs for plug-in integrated office furniture.

2.4.7 Expo Pavilion

The Expo Pavilion project was a 1,700m2 (18,000sf) pavilion for a major electric utility company. The concept consisted of a sphere-shaped double-walled space truss structure suspended above the ground. All structural and mechanical members were completely integrated in an easy-to-assemble joint-based kit-of-parts.

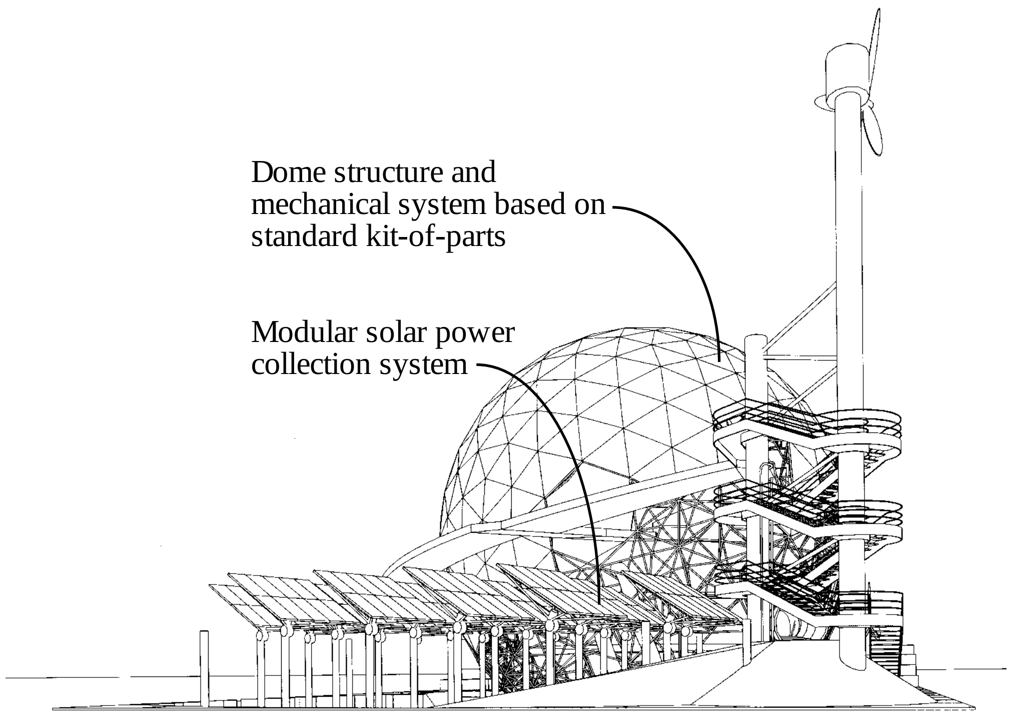

Dome structure and mechanical system based on standard kit-of-parts
Modular solar power collection system

Figure 13: Expo Pavilion project

Interior spaces included domed theater, waiting areas, and exhibit spaces. Energy generation towers were placed in a grid on the site and include windmills, hydroelectric generation, and solar panels which double as roofs for an exterior amphitheater / waiting area.

Research for the project included design investigations for structural / mechanical integration in the context of kit-of-parts systems, and explored viable methods for systematic disassembly when the exposition concluded.

2.4.8 Capsule Hotel

The capsule hotel is based on a module-based kit-of-parts library of components that can quickly be assembled or disassembled. The components are pre-manufactured and constructed into capsule hotels at train stations. Capsules can be coin operated similar to lockers, and accessed directly from the platform.

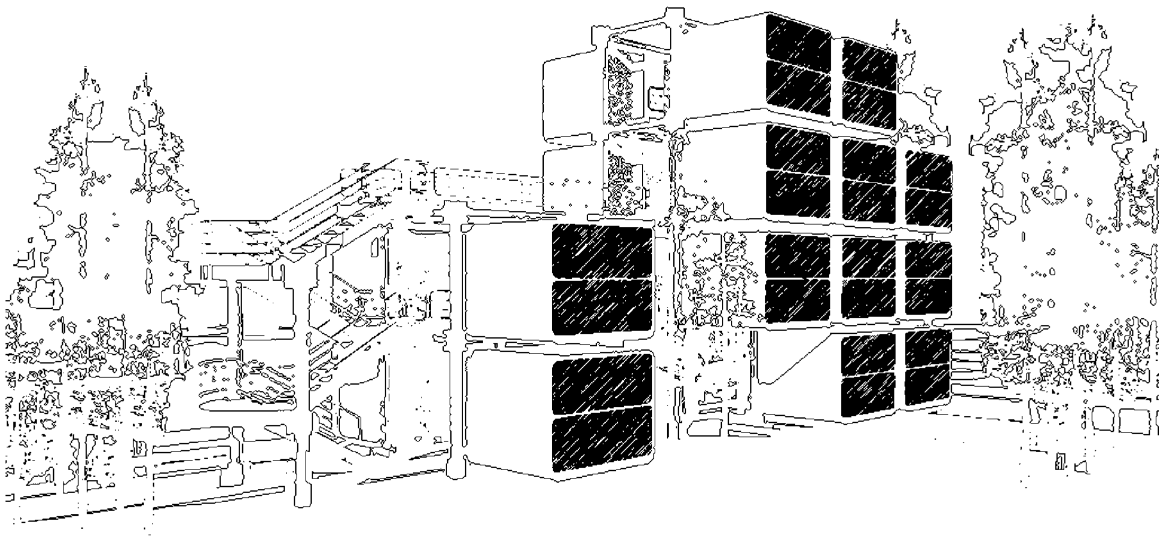


Figure 14: Capsule Hotel project

Each module is manufactured with a spun fiberglass shell and integrated structure that can be cantilevered from a central structure [Paraskevopoulos, Borkin et.al. 1968].

Each module has two full-height entrances which access upper and lower sleeping shelves. Only one sleeping shelf is accessible at a time from any one entrance, unless both entrances of the same module are rented and the removable block panel is unlatched from both sides. Each entrance converts to a small shower stall.

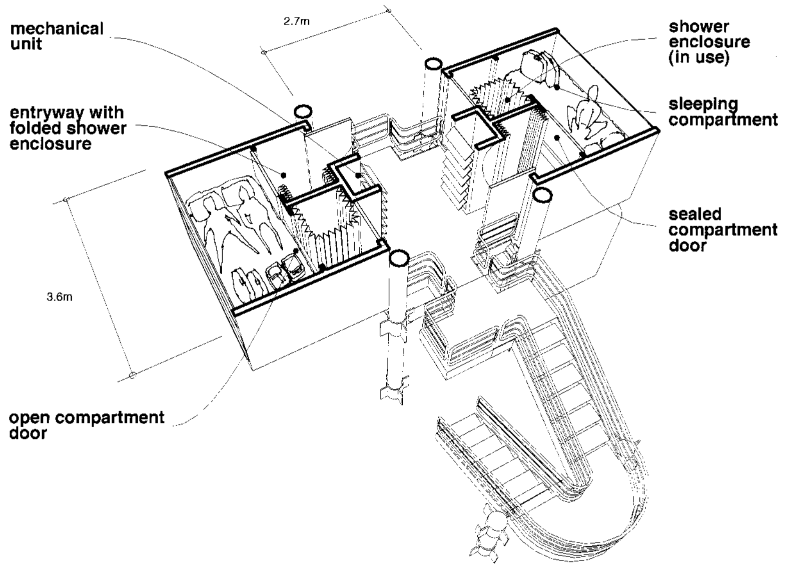


Figure 15: Capsule Hotel horizontal section

Each train station that is projected to maintain a capsule hotel would have a foundation grid constructed which reflects the maximum possible expansion for the hotel. The modular system would be assembled as needed on the foundation grid, leaving unused foundation sockets. When certain train stations show the need for more modules, capsules can be removed from another site, transported by train, and reassembled where needed in the waiting foundation sockets. A system such as this can also be used in emergency situations. Having a large supply of capsules installed at various locations which can be quickly dissassembled and relocated to disaster areas can provide temporary housing relief to stricken areas.

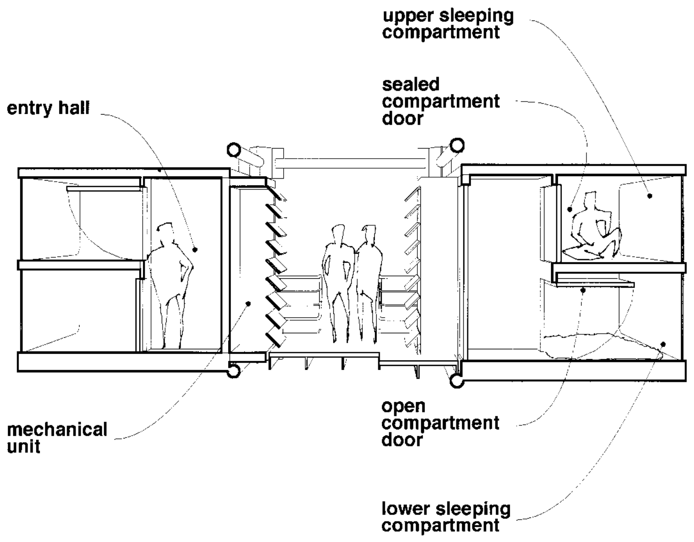


Figure 16: Capsule Hotel vertical section

Research for the Capsule Hotel included design investigations and space / volume studies for achieving the optimum size for sleeping chambers, structural increment, and station platform placement.

2.5 Summary

Prefabrication and kit-of-parts construction has been around for a long time, born out of the need to shorten the building timetable and reduce costs. Though the majority of prefabrication research and development has been initiated in response to urgent needs in connection to some historical event, a significant amount of top-down research has been conducted through design investigations. In the context of this work, kit-of-parts concepts provide a clean object-oriented approach to building which lends itself to the formulation of a life-cycle management paradigm. Previous to this work, some of the authors own design investigations have established a foundation upon which this work can rest.

CHAPTER 3

RELEVANT TECHNOLOGIES

Three major technologies which have contributed to this work are discussed: computer visualization, remote control / monitoring, and automated construction. In addition to surveys conducted to establish the state of the art of these technologies, several simulations, exercises, and studies were made which explored the feasibility of application in kit-of-parts design, construction, and life-cycle management techniques. These exercises are discussed in this and in subsequent chapters.

3.1 Computer visualization and virtual buildings

Architectural information can be divided into two major categories:

> 1) Information about the building artifact itself. A hierarchy of information types and elements can be used to describe a building. A building can be expressed in terms of its functions or activities. The functions or activities can be expressed in terms of their defining systems or containing spaces. Systems can be expressed in terms of their components, and spaces defined in terms of their enclosing structures. Components and Structures can be expressed in terms of their composition and materials. Materials can be expressed in terms of their defining attributes.
>
> 2) Methods for bringing the artifact into being. Methods for constructing the building include rules of how the elements fit together and procedures for their assembly. Often the design of the building is greatly influenced by the methods of construction, and the component size by their ability to be transported.

Constructing an accurate digital representation of an object in the computer, and having the ability to isolate each bit of information when needed for a specific task while filtering out enormous volumes of unneeded information is the two-fold purpose of information visualization.

3.1.1 Traditional methods of visualization

TWO-DIMENSIONAL EXPRESSION: The major part of an architect's training is concerned with visualization by way of drawings. Architects are trained to be proficient in the use of a variety of tools, ranging from a simple pencil to a complex computer system as a means of expressing the nature of the building they've conceived. Drawing is so important that in many countries, including the United States, it is a prerequisite for local government's approval of a building project. It is for this reason that the general thrust of the development of tools for architects has been to address this need to produce drawings.

THREE-DIMENSIONAL EXPRESSION: In architecture, the primary goal is not the drawings themselves, but the production of a dynamic three-dimensional collection of spaces, defined by structure, function, and aesthetics. Plans, sections, and elevations, which constitute the main types of views included in working drawing packages, are merely two-dimensional slices of the three-dimensional object. As a partial attempt to overcome this inadequacy, three-dimensional representational drawings (perspectives and axonometrics) and scale models have been a traditional spatial visualization technique.

STATIC INFORMATION: In the preparation of working drawings and specifications, it is often required that certain bits of information be repeated for clarity. For example, "calling out" material designations on an elevation may require the same information to be noted in various places on the same drawing, with references to the same material included in the specifications. In traditional drawings, problems occur when changes occur during the design process and the content of the information changes. In the example of the elevation material designations, the "call outs" are static and each of them would need to be changed if the material is changed.

DYNAMIC INFORMATION: The use of computers has greatly improved our ability to solve the problems associated with static information. Drawings and other means of architectural expression which have traditionally been conceived as separate entities can often be derived from core information. "Call outs" can be handled as keynotes which are dynamically linked to specifications such that redundancy becomes only a matter of appearance; repetition becomes instances of the same bit of information. In the dynamic use of information, changing a core piece of information will automatically change all of the instances.

Still, many problems remain. Reliance upon two-dimensional media for the representation of innately three-dimensional artifacts continues to be the primary function of architectural visualization. Information required for structural, environmental, daylighting, acoustical, energy, and other forms of analysis is for the most part still unlinked and static within the scope of the entire project. In addition, currently the role of the building representation (in other words drawings and computer models) is viewed as having ended its usefulness when the object it represents is built. This ignores the potential wealth of information that could be gleaned over the entire life span of the building to support building maintenance, re-design, and facility management.

3.1.2 Virtual buildings

Some of these problems can be solved with the Virtual Building concept. [Smith 1992] explained:

"A finished building is the best model of what the building will be...We don't usually have the luxury of constructing the complete building as a model or prototype. If we could do this, we could identify its deficiencies and our errors, and then tear it down and construct an improved version. What is needed is a Virtual Building that we could design and construct as many times as we please."

A Virtual Building is a three-dimensional representation of an innately three-dimensional artifact. All the types of information, from functions to spaces to structure to material to attribute, are fully represented and linked. A designer can manipulate simple volumes which are abstracted versions of the building's major form elements. In the same model, walk-throughs and fly-throughs can be accommodated in real time using Virtual Reality, such that materials, spaces, and sunlight can be experienced. Tools for performing complete analysis by filtering out only that data which is needed for the specific task would be linked to the same model. By keeping the Virtual Building active after the construction of the real thing, facility management could be facilitated effortlessly. Using sensors, the virtual building could be linked with the real one for the purpose of monitoring performance along its entire lifetime (for example, monitoring temperatures in numerous locations inside a space could give an accurate picture of heat loss / heat gain over time, in the quantified context of a three-dimensional model).

In virtual buildings, drawings would be incidental to the representation instead of being the primary objective. When a designer manipulates the simple volumes, any drawings that represent two-dimensional slices would be automatically updated. With the use of linked expert systems, elements that represent structural members would be linked to the volumes, and recalculation would occur along with automatic resizing of the members. Virtual buildings can be created with various tools available on the market today. However, they will require large databases and multiple data formats.

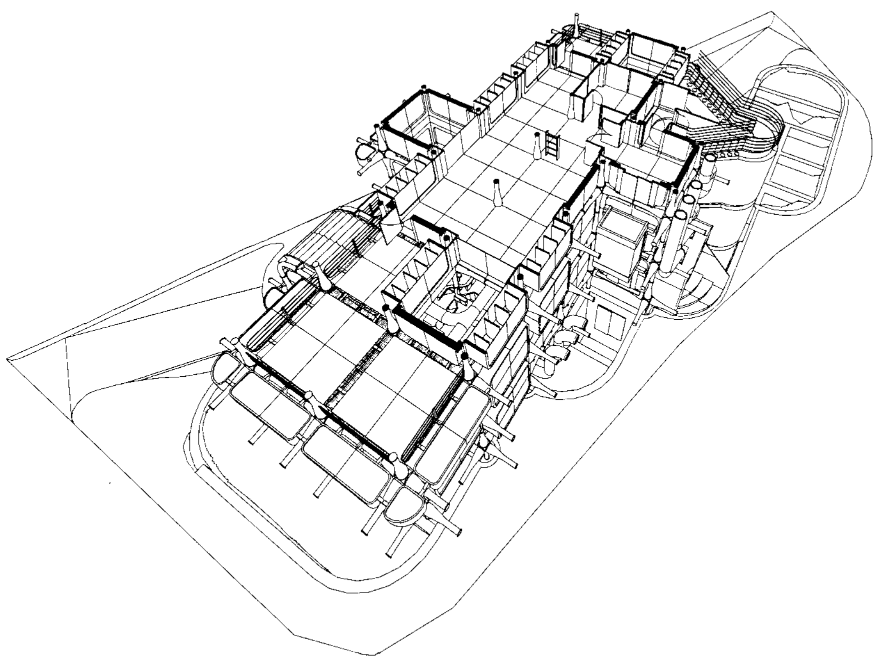


Figure 17: A slice of the three-dimensional Virtual Building I project...

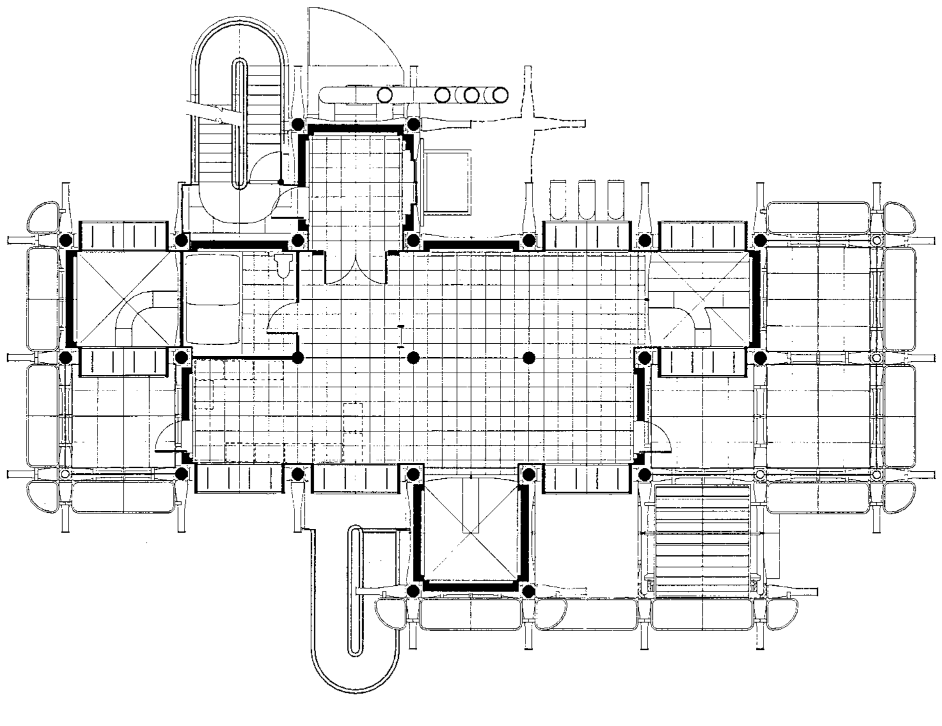


Figure 18: ...produces a two-dimensional floor plan

Actual implementation of virtual buildings as a product would require the development of an enhanced visualization system or browser which could be linked with databases and expert systems.

3.1.3 Specification for a visualization system

Due to size, especially with large projects, it may be impractical to contain all of the project database on a single system all at the same time. A one-minute animation alone could take up as much space as 800 megabytes depending on resolution. Instead, the use of networks such as the Internet with its potentially unlimited amount of disk space and information sources coupled with real-time visualization and filtering techniques is proposed. The assumption would be that in order to fully implement this system, building component manufacturers and materials suppliers would need to maintain servers on the Internet, and architects and designers would need to have access thereto.

It might be advantageous to develop a browser which will help the architect view the project database. Some desirable charactaristics may be as follows:

LEVELS OF ABSTRACTION & FILTERS: It would be necessary for the browser to have the capacity to view the project database in various ways, or at different levels of abstraction. Geometrical properties, textures, wireframe, textual descriptions, and diagramatical or symbolic views of the same database as well as the ability to apply filters would be necessary. Viewing all of the information at once would be unintelligible in most cases, but applying filters would select only that which is needed at the time. Other filters might include: hither and yon sectioning (conical, orthographic, and zoned) with automatic line width adjustment for the cut edge; dimensioning filters with automatic orthographic or perspective linked dimensioning; underlay and overlay filters which can handle linked images or linked text.

WINDOWS, SUBWINDOWS & BOOKMARKS: The browser would necessarily have the capacity to support multiple windows. Each window would be independent of each other in that different levels of abstraction or filters could be applied to different active windows. Within each window, the ability to insert a subwindow or lens with a different level of abstraction or filter would also be useful. Each window should be storable. Independent of the project database, a feature allowing casual visitors to the virtual building to save their own views or "bookmarks" (similar to the way one can save bookmarks on the Web) would be required.

INPUT & EXPORT: A function allowing the import and export of various data formats such as DXF, IGES, STEP, VRML, 3DMF, OBJ, PICT, TIFF, RGB, MOV, live video, and sounds as well as views, bookmarks, windows, subwindows, and filters would also be necessary.

SHEETS: The capacity to arrange and store sheets for the purpose of plotting output would be useful. A sheet would consist of an arrangement of windows (as defined earlier) that are either bordered or borderless and can overlap with adjustable degrees of transparency.

WORD PROCESSOR & SPREADSHEET: Having full word processing and spreadsheet characteristics in a fully linked manner would be required.

TEXT-BASED VISUALIZATION TOOL: For use with visualizing large volumes of textual data, it would be necessary to include the capacity to view certain parts of the database and have the remaining parts compressed "around the edges" using Fish-eye Views [Furnas 1986] and Butterfly visualization [Mackinley 1995], or data hierarchies with Tree Maps and Clustering [Mukherjea and Foley 1995].

PRODUCT MODEL VISUALIZATION TOOL: It would be necessary to have the capacity to represent the model in product model coding such as NIAM (see appendix) or EXPRESS, as proposed by the International Standards Organization.

MODELING: Modeling which supports manipulation of all of the geometric data elements would be necessary. Point manipulation, line & curve editing, surface operations, solid Boolean operations, as well as advanced modeling techniques such as mesh, fillet, sweep, rounding, parallel object, text editor would be needed.

RENDERING: Required would be the ability to apply texture mapping, bump mapping, fractal mapping, and decals. The full range of wire frame, hidden line, flat shading, Gouraud shading, Phong shading, Z buffer rendering, scan line rendering, radiosity, and ray tracing rendering is required.

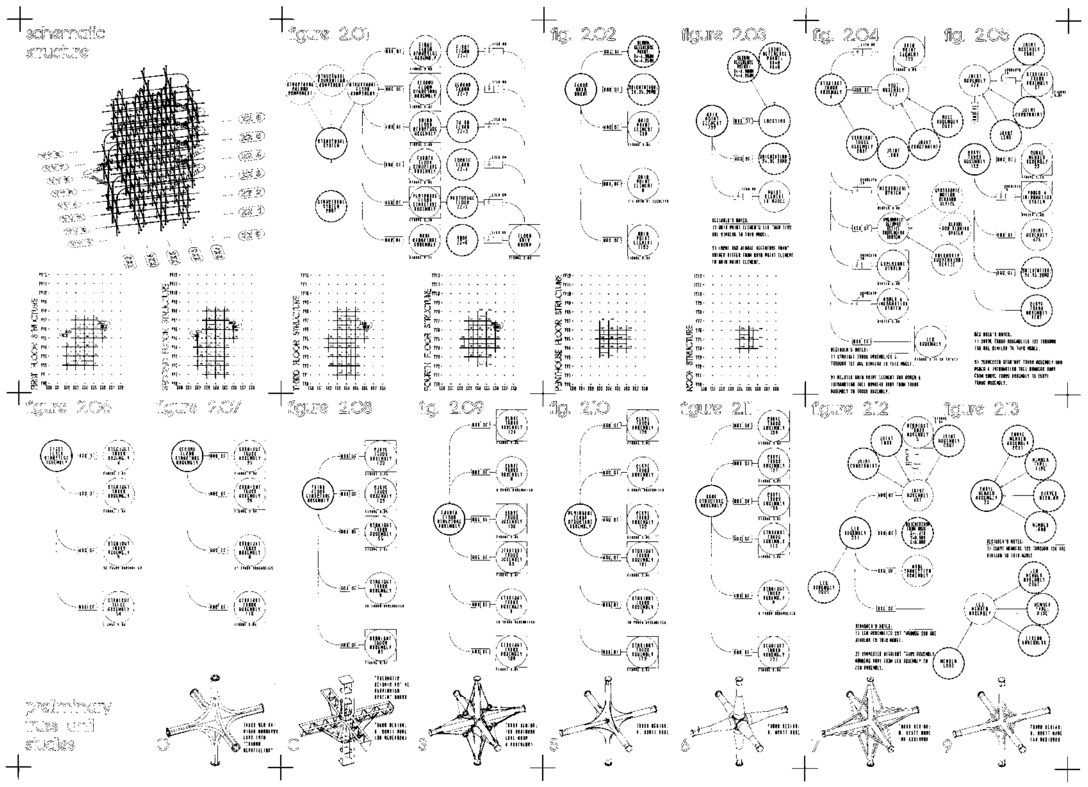


Figure 19: NIAM product modeling

ANIMATING: The ability to create walk-through paths, animations, Quicktime movies, and video input & output is necessary.

KINEMATICS: A feature for creating self-contained (behavior scripted) kinematic mechanisms, with the capacity for interactive operation or independent operation programming would be important.

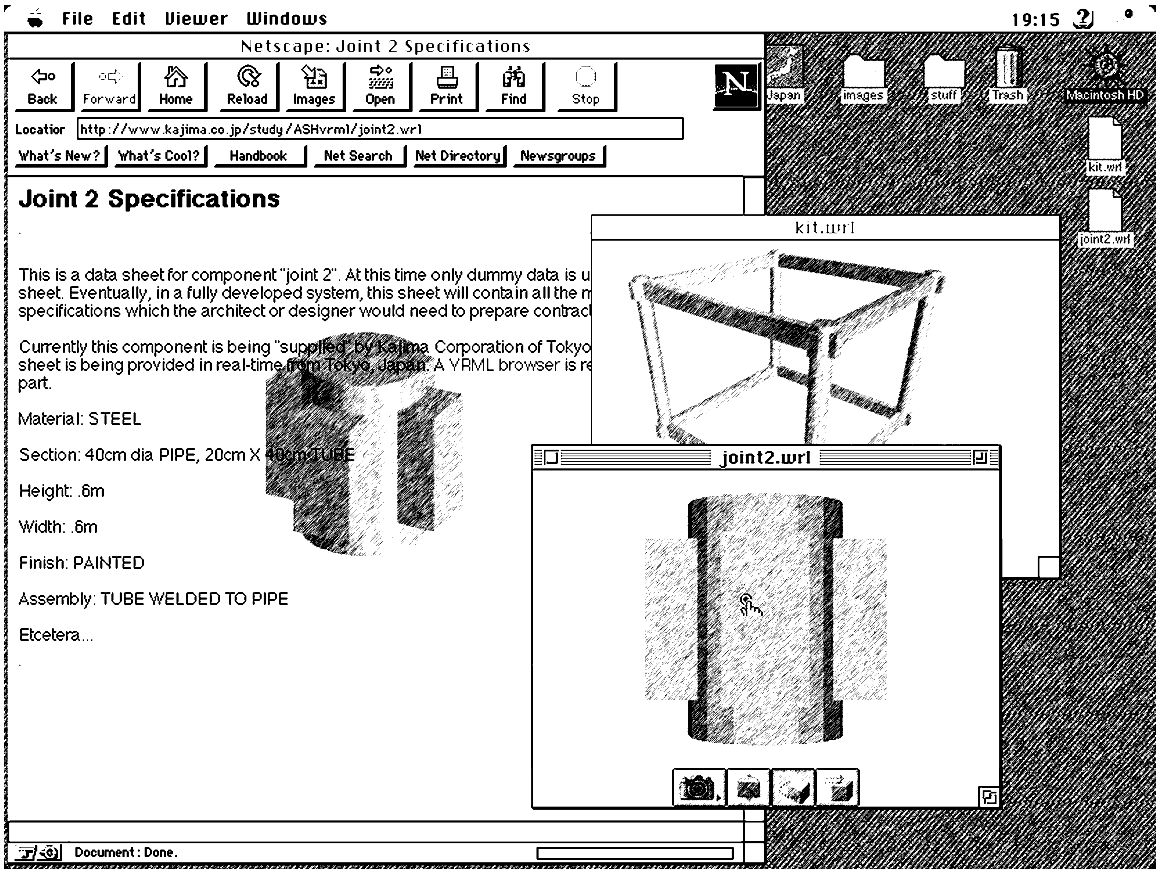
EXPERT SYSTEMS: Plug-in virtual designer, virtual structural engineer, virtual mechanical engineer, virtual electrical engineer, and virtual civil engineer could be developed.

3.1.4 Demonstration project

To demostrate the usefulness of such a browser, research has been done in the area of network-based modeling using hypothetical manufacturers of building components located at different areas in the world. The research reflects how a browser would function in its basic form and is the predecessor of the VBuild computer program discussed later.

The demonstration project conducted in 1995 consisted of a simple structure which used only three components: a column, a beam, and a joint [Howe 1996]. The models of the three components existed on Kajima Corporation's server in Tokyo, Japan and were stored in Virtual Reality Modeling Language (VRML) format, which is a powerful network-based modeling language. The parent model was located on a server at University of Michigan and had no components in it, only pointers which told the browser where to find the parts. Kajima represented a steel manufacturer located in Japan, and Michigan represented the project's architectural firm.

The demonstration involved browsing the parent model over the World Wide Web. The information was scattered over the Internet at remote corners of the world, but using pointers the browser was able to assemble it all together into a coherent whole. Since the information is assembled in real time, all of the individual parts were current and up-to-date. Clicking the mouse on one of the parts would bring up a sample specification page.



http://www.kajima.co.jp/study/ASHvrml/joint2.wrl

Joint 2 Specifications

This is a data sheet for component "joint 2". At this time only dummy data is u
sheet. Eventually, in a fully developed system, this sheet will contain all the n
specifications which the architect or designer would need to prepare contrac

Currently this component is being "supplied" by Kajima Corporation of Tokyo
sheet is being provided in real-time from Tokyo, Japan. A VRML browser is re
part.

Material: STEEL

Section: 40cm dia PIPE, 20cm X 40cm TUBE

Height: .6m

Width: .6m

Finish: PAINTED

Assembly: TUBE WELDED TO PIPE

Etcetera...

Figure 20: A network-based modeling system

3.1.5 Summary

Though the actions performed in the demonstration are now commonplace, the study was one of the first attempts at assembling remotely located virtual building components in real time.

3.2 Automated construction

Architecture can for the most part be described as the creation of singular artifacts addressing case-specific needs for shelter. Architecture is not generally mass-produced, but is usually one-of-a-kind. This is one of the main reasons that the construction industry has been slow to adopt new manufacturing technologies which naturally come out of competitive mass production processes. In fact, there are many building construction methods and technologies in use today, such as masonry and rough carpentry, that can be traced back hundreds and even thousands of years.

Since the advent of the Industrial Revolution, architecture has seen slow, evolutionary progress in improvement of construction methods. Because of the one-of-a-kind nature of buildings, it has not been applicable (except in rare cases) to use mass-production techniques in construction. Still, improvement in the development of construction machines and tools has seen much progress, and the overall attempt to gradually adapt the new technologies to traditional building methods is taken for granted.

In recent years, however, another trend is emerging in parts of the construction industry which have to do with revolutionary changes in construction technologies. Instead of adapting new technologies to traditional methods, altogether new methods, conceived with the new technologies in mind and tailored to architecture, are being explored. One of the most dramatic of the new methods is automated construction, or the process of constructing (or manufacturing) buildings using robots and automated construction machines.

With the use of new methods of construction, our conception of how buildings go together will change. A knowledge of the new construction methods may influence the formulation of design concepts and encourage conscious effort on the part of architects to design for efficiency of manufacturability. "Design for manufacturability" could provide a healthy constraint around which creative designers can establish the aesthetic of the building in addition to pure function, much the same way building codes can provide healthy constraints which influence the design.

Automated construction technology will be explored through five topics: design, materials handling, application to traditional methods, progressive methods, and actual implementation & development.

3.2.1. Designing for automation

The majority of automated construction research and development has been bottom-up, from the construction / engineering side rather than top-down from the design end. The reason for this is obvious, since construction processes have no clear relationship with clients and users but everything to do with the production of the building. For this reason there are few papers available that address design for automation. When automated construction practices become more prevalent, it is assumed that more designers and architects will begin to address the issues that face their profession.

As an exemplar, [Bridgewater 1993] has discussed design considerations for automated processes. Two levels are described at which designers need to incorporate ideas of automation into their design concepts: component level and machine level. Neither one of the levels advocate radical changes in designs themselves, but only how the building is to be constructed and detailed. Essentially, designers need to consider the manufacture of larger parts in factories that can be assembled quickly on the site by appropriate construction machines.

At the component level, Bridgewater explains the concept of "strong axis of assembly." building construction usually involves assembly of components in a vertically downward direction. This is naturally due to gravity, and becomes what is called the strong axis of assembly. When the strong axis is identified, various tricks can be applied which simplify the assembly process and encourage ease of constructability. Those include beveling of joints and guides, cartesian orientation with minimal rotation, wiring harnesses, and snap-fit joints.

At the machine level, different tasks must be contemplated in order to simplify the assembly process. Designers having a knowledge of the machinery available can detail construction of joints and consider material handling methods which are optimally conceived for automation technologies. As stated earlier, there are only a few papers available on the subject of design for automation. Though Bridgewater attempts to explain how designers should consider how buildings are constructed and detailed in order to optimize automated processes, nothing is said about rethinking the design concepts themselves for even greater optimization. In fact, this area is weak overall and can be viewed as a large hole in design for automation research.

When designing for automated construction, the architect must remember that the range of area within which a robot can reach or perform work is called a work cell. Work cells differ depending on the design of the robot. In building construction, the multiple tasks required to complete the construction must be matched with work cells of corresponding appropriately designed robots. It may not be sufficient to use off-the-shelf robots for all tasks, but the design of construction machines in parallel with the building may be appropriate at times. For this reason it may be critical that design concepts of buildings be constructed using robots and construction machines be sensitive to work cells and automation concepts.

3.2.2. Material handling

In considering new construction processes utilizing automated construction, development of single-task, single-function robots in isolated environments or work cells is insufficient. An issue covering the handling of the materials and building components emerges, which seeks to provide smooth flow of the machinery and increase overall construction productivity.

[Skibniewski and Wooldridge 1992] discuss the use of material handling in high-rise construction. In material handling, building components are manufactured in factories, shipped to the construction site where they are temporarily stocked, and then delivered to the construction mechanisms for assembly into the building. Various automation technologies are utilized in the process, including Automated Guided Vehicles (AGV's), Automated Storage / Retrieval Systems (AS / RS's), robotics, and automatic identification. In automated construction, these technologies are applied to the three major tasks of building, warehousing, and manufacturing; and material handling systems must provide the link between the three.

Not all the processes involved with material handling can be automated as of yet, for the delivery of assembled components from the factory to the site, and from raw materials to the factory still require human intervention. Nevertheless a major portion of material handling can be automated. Skibniewski and Wooldridge describe the use of autonomous fork lifts, conveyors, automated lifts, and AS / RS warehousing in the construction of high-rise buildings in Japan. These systems used together make up a materials handling work cell, which physically overlaps all the work cells of the various robots and construction machines to the point that appropriate materials are delivered to each on schedule.

Another outstanding exemplar in material handling is work being conducted at Tokyo University [Sakao, Kondoh, Umeda, and Tomiyama 1996]. The cellular automated warehouse discussed begins to consider the robotic work cell as a module that fills up space. Conceivably, any material can be delivered or placed in any location in space, and the system can be reconfigured, expanded, or contracted in a minimal amount of time.

3.2.3. Traditional methods

A great majority of the work going on in automated construction research is in thinking of ways to apply the technology to traditional building methods. While there are a great many problems that need to be overcome in order to develop usable automated tools, studying possible applications to traditional methods is necessary because they are the most familiar to us.

[Bernhold, Abraham, and Reinhart 1990] advocate the application of automation technology in an evolutionary rather than revolutionary manner. In spite of the fact that buildings tend to be unique and prototypical, a careful analysis reveals that uniqueness can only be used to describe the condition at the higher levels of construction function: organization, project, and activity. Basic levels of function such as process, task, and motion are largely repetitive in nature, which is amenable to automation principles. Bernhold, Abraham, and Reinhart discuss flexible manufacturing system (FMS) concepts, meaning the use of highly adaptive machinery that can handle a variety of tasks rather than specialized equipment. These adaptive devices can be programmed to build everything from a brick wall to finished siding.

The work of Leonhard Bernhold was chosen as exemplary not only because of the concepts advocated in the FMS paper, but also for other work which attempts to apply automation to traditional building methods. This work includes the development of robotic masons, bridge painters, and excavation tools.

Another exemplar of the automation of traditional methods is Roozbeh Kangari. [Kangari and Halpin 1983] has researched the potential of robotization of individual tasks required in construction, assuming that the feasibility of automating the entire construction site would be dependent on need and would occur gradually. Tasks were rated by difficulty, repetitiveness, and safety among others.

3.2.4. Progressive methods

Instead of adapting automation technology to traditional methods, some research is being conducted which advocates kit-of-parts and prefabrication building techniques. These methods often use assembly techniques which are optimal for use in automation or attempt to apply as much of manufacturing technology as possible by finding ways to mass-produce building components. As was mentioned earlier, kit-of-parts methods have superb examples of development and implementation including the work of architects such as Fuller, Kurokawa, Foster, Piano, and Rogers who have hinted at the use of automation in manufacturing of components.

[Kurita, Tezuka, and Takada 1993] report about the development of a three-dimensional modular unit construction system tailored to the use of automated construction. Large building block components are assembled in a controlled factory environment (whether on site or off is irrelevant at this point) and shipped to the site for assembly.

Dante Bini's work is an exemplar of kit-of-parts development for the purpose of automated structure assembly. New space frame configurations and structural systems have been developed which assemble easily and lend themselves toward automation. Bini's work is a form of special construction kit-of-parts that folds out and deploys from a compact package.

3.2.5. Implementation and development

Automated construction techniques that have been fully implemented appear to fall into three major categories: collections of function-specific robots that work independently of each other, whose work cells are not necessarily laid out in a systematic way; robotic systems which form a systematic "factory" that is stationary or fixed in the context of the site; robotic systems which form a systematic "factory" that moves itself along as it completes portions of the building.

All of the examples are Japanese in origin. As we have seen so far, considerable research has been conducted in the United States, The Netherlands, and other countries but most of it has been at a university or research organization level. The majority of actual implementation has been in Japan. One reason for this is because of Japan's research infrastructure: large corporations conduct the majority of research under the encouragement of the Japanese Ministry of Construction and Japanese Ministry of International Trade and Industry, and thus have the funds to bring ambitious projects to reality.

The first category covers robots which were developed for inclusion into manned construction sites. Repetition, labor, or safety needs have justified the use of robots. These machines have mostly been developed for adapting automation techniques to traditional building methods. In addition, the first category includes robots and machines which are not necessarily part of construction, but may have some function to do with the maintenance of a building on a day to day basis. As an exemplar, Taisei's exterior wall painting robot can be cited. On a high-rise building (Shinjuku Core building in Tokyo, Japan), vertical gaps at each column line that run the entire height of the building contain hidden rails. The painting robot is positioned into any two of the rails and automatically moves up or down the entire building facade doing its job without assistance.

An exemplar for the second category is Kajima Corporation's AMURAD system. AMURAD sets up a stationary factory on the building site, with fixed material handling routes that deliver pre-manufactured building components to the robotic system. The factory is set up to automatically assemble one entire floor at a time, and to jack the floor up one level when it is finished. As a result, a finished high-rise building is "extruded" up from the site. Upon completion of the building, the ground-level factory is disassembled. The author was involved with early stages of the project, and conducted design investigations to develop a kit-of-parts that could be easily constructed using AMURAD equipment. The AMURAD equipment was used to construct two high-rise buildings located in Nagoya and Tokyo, Japan.

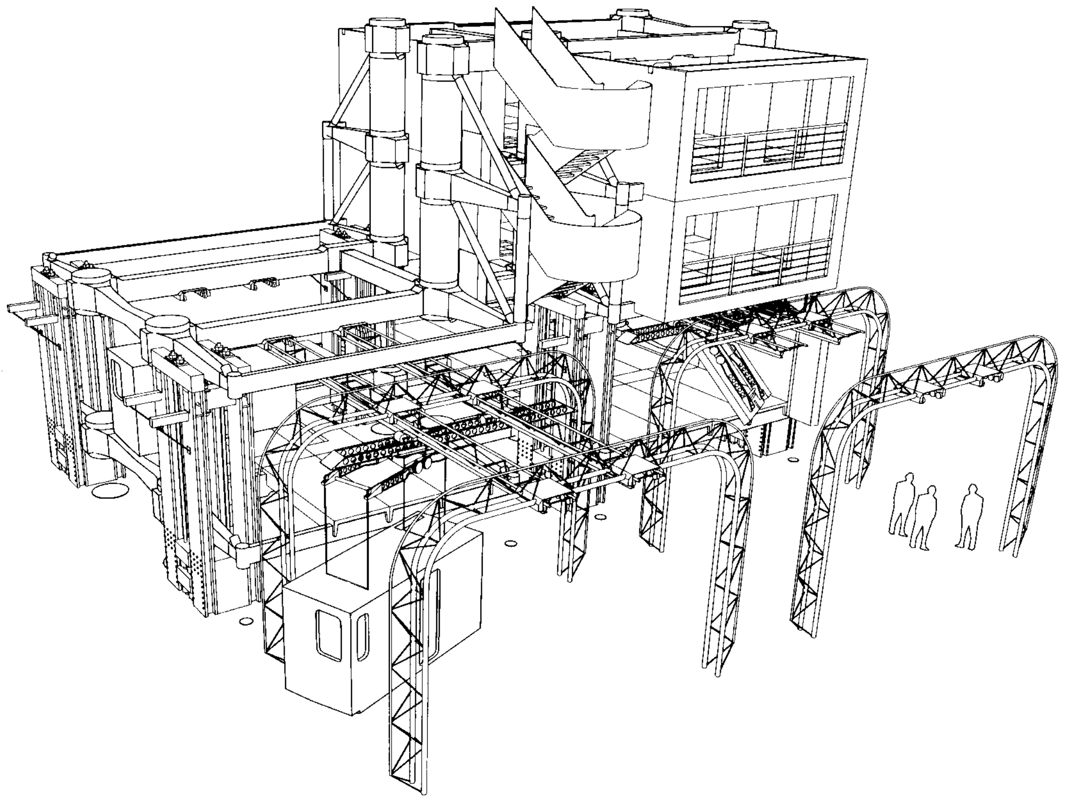


Figure 21: AMURAD automated construction site designed by author

In the third category, four of Japan's "big 5" construction firms (excluding Kajima) have some sort of automation implementation. The firms are: Shimizu, Taisei, Takenaka, and Obayashi. In addition to these, several other construction firms have also developed similar systems. In this case, Shimizu's SMART system is cited as an exemplar. The SMART system begins with a small core structure (which will become the building's core) being constructed conventionally to the height of a few stories. A platform the size of an entire proposed floor is attached to the top of the core. The platform contains a factory which proceeds to automatically assemble the entire ground floor, and in parallel constructs another level onto the core structure. When the ground floor is complete, the entire factory platform automatically jacks itself up (using the core structure) one level and repeats the process with the second floor, and so on. Material handling is done using automated lifts located in the core structure, where factory assembled components are delivered to the ground level and are automatically delivered to the factory. In effect, a finished high-rise building is "extruded" by the factory as it jacks itself up floor by floor. When the building is finished, the factory is disassembled and removed from the top of the building.

3.2.6 Summary

Automated construction appears to be a trend that will stay. Though revolutionary strides are occurring in some corners of the industry, overall evolutionary changes toward automation are taking place universally. Research and development has mainly been from the bottom-up where engineers and construction management have direct influence over the construction process. Top-down approaches from designers and architects are still few and not sufficiently covered.

3.3 Remote monitoring and control

In this section, a brief survey of current off-the-shelf remote and computer-based facility management technologies is discussed. Telephone, Internet, wireless, and cable-based technologies for communicating commands to (and receiving feedback from) architectural devices will be touched upon.

In addition, a simple demonstration involving an experimental Internet facility web page and server-based software engine for monitoring the current condition of a building will be described. The World Wide Web has shown great potential as a medium for distributing current information in real time. Home pages for companies and individuals have become dynamic resumes and portfolios open to the rest of the world. The experimental web page described herein was developed as a home page for a facility rather than an individual or company. The web page contains a VRML model which was conceptually linked to an actual facility. One was able to 'walk-through' the VRML model and click on certain locations to download simulated real time temperatures and other data.

3.3.1 Virtual building for remote monitoring and control

A rich model produced by architects in the design stage could respond to analysis the same way the actual building would. Space planning, structural, accoustical, daylighting, thermal, and energy analysis could be conducted to provide insight into more efficient designs before the actual building is built. Performance data could be compared with optimum mathematical models to produce designs that need less artificial heating and cooling, are naturally lighted, and save energy. Early space planning computer programs such as CRAFT (Armour & Buffa), ALDEP (Seehof & Evans), DOMINO (William Mitchell), and STACKing plan (Charles Reeder) have evolved over the years and become somewhat of an industry standard. Several commercial applications available today have space planning capabilities and can produce models which can be expanded into working drawings. The models are linked with a building's knowledge base and allow for some analysis. An ideal virtual building is getting easier to accomplish with the richer models made possible today.

Construction documents, the conventional products produced by architects, usually lose their usefulness after the building has been constructed. With ever richer computer models produced for representing the actual building, can their usefulness be extended after construction? One aspect of facility management that has become common is the use of computer aided design (CAD) models produced by the architects as working drawings, for the purpose of furniture layout planning, inventory databases, and other facility management applications.

A more advanced facility management technique is using the computer to monitor and control lighting, equipment, appliances, and other devices in the building. Often refered to as “building automation”, this technique has until recently been out of reach of all but the most costly facilities.

3.3.2 Potential media for linking the real and virtual

This section will discuss remote facility management applications using four communications technologies: telephone, wireless, Internet, and cable. Briefly, each technology or “medium” will be described followed by example applications. Some of the topics will be dealt with in more depth than others, allowing expansion of ideas that are perhaps not well represented.

TELEPHONE: The telephone system represents the most widely accessible communications network in the world. A telephone connection is generally set up between two specific points and consists of a full duplex line (where both parties can transmit and receive simultaneously). The system was originally designed to carry sounds within the range the human ear is capable of discerning, but has since been expanded to handle the sending and receiving of data as well.

A very interesting standard which developed in conjunction with the telephone system is the Dual-tone Multi-frequency code (DTMF). DTMF refers to the tones generated by a telephone during the dialing sequence. The DTMF code was originally developed by Bell Labs for the purpose of providing a robust way of broadcasting digits over the network for the purpose of announcing the dialed number to the switchboard. The old pulse technique used by rotary phones proved to be unreliable for long distances and over microwave connections. It was determined that a series of tones lying within the same frenquency range as the human ear can discern would be far more reliable. Each digit could have a single tone, but there would always be a chance that random sound on the same frequency could cause erroneous signals. That is where the dual tone system came in.

The human voice ranges from around 2000 Hz for men, to 3000 Hz for women. Table 1 shows the DTMF frequency grid which lies outside that which the voice usually produces, but within the range that can be discerned by the human ear. As the digits are layed out in a grid, each row and column represents a different frequency. When one of the digits is selected, “5” for example, two tones, one of frequency 770 Hz from the row containing “5”, and another of 1336 Hz from the column containing “5”, are generated.

1	2	3	A	697 Hz
4	5	6	B	770 Hz
7	8	9	C	852 Hz
*	0	#	D	941 Hz
1209 Hz	1336 Hz	1477 Hz	1633Hz	

Table 1: DTMF dual tones

The DTMF code is of particular interest to facility management because it is generated by common devices which occur in most every home. The DTMF signals can be encoded on one end and decoded on the other for a form of digital communication. Computers and modems decidedly do a better job at this, but the availability and portability of telephone handsets, especially cellular phones, allow them to function as a form of remote control device. On the human end, a controller device can be called using the telephone. When the connection is made digits are input in sequences which can communicate with the controller. The controller decodes the DTMF tones into ASCII commands and matches them to preprogrammed sets of instructions. The DTMF grid includes four extra digits which normally do not appear on the common telephone handset, called A, B, C, and D. These digits will prove useful in facility management control over wireless media (as will be explained later), but are not available using telephones.

Common pieces of hardware that can access the telephone network include the telephone (consisting of a speaker, microphone, and a DTMF keyboard), data modem, facsimile, and special dial-in controllers (discussed later.)

WIRELESS: The transmitting and receiving of certain electromagnetic waves through space is known as wireless communication. Though generally refered to radio for communication purposes (two-way, amateur, etc.), wireless can include television, satellite, microwave, and Amplitude Modulation / Frequency Modulation (AM / FM) radio. Where telephone requires wires and an established network infrastructure, wireless in its most basic form only requires a powered transmitter on one end and receiver on the other (with the exception of satellite which needs at least one intermediate repeater -- the satellite -- in order to qualify as such).

Above sound waves, electromagnetic waves used for wireless (as defined in this paper) range from 3 KHz Very Low Frequency (VLF) to 3000 GHz Extra High Frequency (EHF). Frequency is defined in Hertz (Hz) where 1 Hz is defined as one sine wave cycle per second, 2 Hz is two cycles per second, etc. Amplitude is the height of the wave. The transmitter consists of an oscillator which first creates a wave (fc) of the desired frequency. The information to be transmitted (fs), for example a voice which is 2 KHz, is added to the oscillator-created wave in one of two ways: by altering or modulating the frequency from the pure wave form (FM) or by modulating the amplitude of the pure wave form (AM). The modulating can be done by adding (fc + fs) or subtracting (fc - fs) the information wave from the carrier wave.

The question may be asked, why isn't the voice wave transmitted directly instead of piggy-backing it onto a carrier wave? The answer is that the voice wave would be limited to the broadcasting of the frequency of the voice, which would put limits on the power and range of the broadcast. This brings up another question, which is in the case of FM wouldn't the added voice wave change the frequency to another frequency? The answer is yes, but the continuous incrementation from the 3 KHz to 3000 GHz mentioned earlier is divided into channels defined by what is called band width. The band width of a single channel gives leeway on either side of the pure frequency of plus or minus 3 KHz to make a total band width of 6 KHz. It is inside this band or range that the carrier wave plus (or minus) the sound wave must lie. Broadcasting only one side of the band is called Single Side Band (SSB) and using the entire 6 KHz band is called Double Side Band (DSB). Broadcast radio usually uses DSB. By standard, transmitters and receivers are manufactured to produce and receive carrier waves in the middle of the band, so there is no industry-wide overlap or slight shift of channels between manufacturers.

In most cases, the operation of a transmitter requires a license to broadcast. This includes professional and amateur (ham) radio operation. Amateur transmitters are readily available in the 1.9 MHz to 1200 MHz range. Amateur operators often use the shape of the wave itself to traverse long distances. Using the 3 MHz to 24 MHz range, the curves of the waves begin to approximate the curve of the earth. Tuning in just right will allow one to broadcast "around the corner" so to speak. In recent years, amateur radio operators sometimes use their equipment to connect to personal computers in order to send data packets over long distances, usually powered by nothing more than a twelve volt car battery.

Waves can also be bounced off things. Using the density of the atmosphere at different altitudes as a reflecting surface and aiming just right, some waves can be bounced back to the earth for long distance communication. Some amateur operators take pride in bouncing waves off the moon. Another form of this technique is the use of repeaters. Repeaters are actually receiver / transmitters that receive a transmission, strengthen its wave, and transmit it again. Television and radio use this technique to get their broadcasts to far away homes. Everyone has seen broadcast radio towers on tops of mountains or at other places: these are repeaters. Satellites are another form of repeater. Repeaters can work singly or in multiple chains, where a signal is sent from repeater to repeater until it reaches its destination.

One of the disadvantages of wireless communication is that, contrary to closed-linked telephone, transmitters broadcast by “disturbing the ether” much the same way that a thrown rock creates waves in a pond. The waves go in every direction and anyone with a receiver can pick them up. Wireless communication is not private. This problem can be solved in a number of ways. One way is to scramble the message and decode it on the receiving end (which is illegal in some cases). Another way is to use extremely low power to reduce the range of a broadcast (assuming that the receiver is close by). A third way is by the use of extremely powerful directed broadcasts such as microwave.

Being able to receive any wave being broadcast can pose another problem, which is receiving unwanted transmissions. To partially solve this, some receiver transmitters are equipped with, or can be equipped with a filter called a Continuous Tone Coded Sub-audible Squelch (CTCSS) card. A CTCSS card produces a continuous tone which is not within the range a human ear can discern. When information is broadcast, the tone is added to the wave. The receiver is set to the same frequency, so it will only put transmissions containing that tone on the speaker. There are 38 fixed frequencies reserved for the CTCSS tone, so there is still a possibility that another transmitter may be using the same CTCSS tone. Another filter which can be used in conjuction with the CTCSS takes advantage of the DTMF capabilities and is called Dual Tone Signal Squelch (DTSS). This method sends a series of DTMF tones at the beginning of the transmission to notify the receiver (which is set to receive transmissions containing the same tones) of an incoming transmission. Using these two filters in parallel, it is almost impossible to receive unwanted transmissions.

The use of DTMF tones brings us back to the topic at hand, which is facility management. For facility management purposes, wireless can provide an extremely powerful medium for remote device monitoring and control. Using an extremely low energy level, commands can be sent to devices within the building equipped with receivers. Broadcasting FM signals over the wiring system of the building provides a way of communicating with any device which is plugged into the power grid. Wireless connection of computer equipment such as keyboards, ethernet hubs, and mouses are increasing in popularity. These devices send out a wave at an extremely low energy level so the range is only limited to room dimensions in most cases and will not send unwanted signals out of the building. Hardware for device control and monitoring for the purpose of "building automation" is available off-the-shelf. These will be discussed later.

The transmitters and receivers used in wireless communication often have built-in tools that lend themselves toward remote control. Two-way and amateur radios contain a DTMF keyboard, much similar to a telephone. Contrary to telephones, radios have the extra A, B, C, and D digits as well. By sending DTMF tones over radio waves, devices can be controlled with the keyboard simply by decoding the tones into ASCII commands to be sent to a controller. With the right hardware, DTMF tones can be decoded right from the speaker of the radio. Decoders, such as those manufactured by Genave, are available on the market from around $300. Using DTMF tones "on the air" always has risks attached, since the possibility exists that they can be intercepted and duplicated by outsiders. Transmitting DTMF tones in sequence over several frequencies may reduce the risk.

INTERNET: One of the most exciting new technologies is the Internet. The potential for decentralized hard core computer applications is literally without limit.

Whatever can be connected and controlled by the computer can be controlled remotely via the Internet. As was mentioned earlier, in 1995 the author developed an Internet-based kit-of-parts library of VRML components [Howe 1996]. The components were located on a server in Tokyo, Japan, and the building on a server in Michigan. The building model had no components in it, only path names and instance transformation parameters. When the model was viewed over the Internet, each of the parts were collected from Japan in real time and loaded into the local machine at their proper location. In other research, as will be discussed later, a robot was successfully controlled from Denmark and Japan to both construct and disassemble a Michigan-based model kit-of-parts building [Howe 1997a]. The commands were made using a laptop computer and local telephone line, with a local telephone call to a neighborhood Internet service provider in each of the respective countries. Use of the Internet has tremendous potential for facility management applications as well. Later on in this chapter, an experimental web page developed for a facility, which allows WWW visitors to "walk-through" a VRML model of the facility and download current temperature data and other information is described [Howe 1997b].

The Internet started in 1969 as a way to link geographically separated U.S. military computer installations. It was essentially a connection between local networks consisting of computers manufactured by different companies. In order to allow them to communicate with each other, a neutral interface called Internet Protocol (IP) was developed. Over the years the Internet has become a global infrastructure.

Today the Internet is a network of millions of computers located all over the world. Each of the computers have a unique identification number called an IP address, which consists of four clusters of digits separated by periods (for example, Kajima Corporation's gateway server IP address is 126.4.21.90). From left to right, each of the number clusters represent ever tighter domain designations. Each of the individual networks of which the Internet consists has a "gateway" server which allows the network to link to other networks. These series of dedicated servers located in various locations throughout the world form the backbone of the Internet. The individual network is called a Wide Area Network (WAN) and represents a certain domain referenced by one of the number clusters in the IP address. The WAN consists of Local Area Networks (LAN) which are the next smallest domain numbers, followed by the number of the computer itself.

The Internet Protocol allows for another more intuitive address in parallel with the IP address. The secondary address has all of the same information as the IP address, but is represented by names instead of numbers (Kajima Corporation's gateway address in this intuitive form is “kajima.co.jp”). In this way each computer has an address unique from any other in the world. When combined with path name conventions used within the computer, each individual file also has its own address unique from any other document located anywhere in the world. E-mail is facilitated this way, where a specific directory is created on a computer which is a unique name of the user. For example, A. Scott Howe's unique name at Kajima is “ash”. A directory named “ash” is created on the Intelligent Systems Department server “ipc”, to make the e-mail address “ash@ipc.kajima.co.jp”.

In the same path name convention, serial ports also have a unique address. Devices connected to serial ports are made accessible directly to the Internet using the unique address, or indirectly using data dump files deposited on the computer’s hard drive. The entire unique address of a document, file, or device is called its Uniform Resource Locator (URL). This is very significant to remote facility management applications.

The World Wide Web (WWW) is not a network, but is an interlinking of all the individual files located on the computers of the Internet. The WWW consists of another protocol which is ASCII text-based and completely platform independent. This means that every computer which recognizes the ASCII standard can read the data which is passed between each other via the Internet Protocol, because it is all text-based. Data is actually transfered using File Transfer Protocol (FTP) which is another standard adopted generally by the computer industry. The WWW is built upon a special text-based language called HyperText Markup Language (HTML) which allows the the varous documents and files on the Internet to be linked to each other.

Special browser applications installed on the local computer are designed to view these various documents and transparently facilitate FTP functions for sending and receiving copies of the documents. The linking is accomplished by inserting special tags into the document which spell out the entire URL of another document or file. The called document or file is copied transparently via FTP and displayed on the screen. The special tags that allow this to happen are invisible when viewed with the browser application. In this way the World Wide Web consists of many interlinked two-dimensional documents.

The way the browser application calls for documents and files by their URL is called “pulling” the data, since it is all initiated by the program running on the local computer. Another important way of passing documents is by “pushing”. Pushing is facilitated by invoking another program located on another computer and causing that computer to do some work. These remote programs are called Common Gateway Interface (CGI) programs. The CGI's can do anything from generate new documents, to operating robots and specialty equipment. Like the files, documents, computers, and devices, CGI's also have their own URL.

Another recent technology on the Internet and World Wide Web is the Virtual Reality Modeling Language (VRML). Where HTML is a text-based language for two-dimensional documents, VRML is a text-based language for three-dimensional models. Equiped with the right browser, one can walk through, spin, and select parts of VRML models. Just as the HTML creates clickable text that leads to other documents, VRML creates clickable objects which bring other objects. In addition, the clickable VRML objects can call HTML documents, invoke CGI's, and otherwise do most everything HTML is capable of in three dimensions [Pesce 1995].

Having such an advanced common protocol system in place, linked to the millions of computers on the Internet, possibilities for advanced facility management applications become limitless. Any kind of device that can generate some kind of signal or data can be given a URL. Temperature sensors, HVAC systems, audio, video, and many other devices can be connected to the computer via data loggers or other specialty hardware. In this way, the actual devices themselves are accessible by clicking on text or VRML representations. Navigating through the building and clicking on a model lamp can activate or deactivate the lamp. Clicking on a VRML model television can turn the real television on and actually display the moving video image on the screen of the VRML representation (this could be done using a CGI program which controls both the television and the operation of a connected frame grabber.)

Just as individuals and companies have home pages on the Internet, buildings and other artifacts can also have home pages which tell about themselves or their current state. It is conceivable that the day will come when all of our automobiles have their own home pages as well, with special logon screens and live video of the view of the driver. Want to see where your friend is? Look up his car's home page, watch the live video, and find out he is driving down Main Street.

CABLE: The potential for using cable television in remote facility management applications is also quite large. Currently, most cable television systems are installed with only a single cable with one-way transmission from the cable company to the home. Some systems, however, have two cables and are thus capable of both transmission and reception.

There are some recent inventions that take advantage of single cable setups, which hold potential for use with remote facility management. Several devices available now combine single-cable television and telephone lines. The single cable allows reception of signals and the telephone allows transmission of signals. The devices connect to the Internet for browsing and manipulation of the standard HTML documents just as a computer would, but without the computer [Snider 1996]. Another device allows web pages to be piggy-backed onto standard television signals for "more information". When a certain topic is discussed on television, the web page reference would provide sources for further information for those who are interested [Fenton 1996]. While these ideas in themselves would not necessarily become tools for facility management, the remote monitoring and control possible on the Internet would be applicable here as well, without the overhead of the computer.

Upcoming technologies using double cable systems, however, are much more exciting. New cable modems that can be hooked up to the cable system will carry tremendous amounts of data for the transmission of smooth video on the Internet. Using cable modems, the possibility of having a server at each household suddenly becomes a viable option (much more so than standard computer modem connections would allow). A server at each household means that every home could function as a small network and run local CGI's for transmitting video and operating various devices in the home.

3.3.3 Remote facility management applications

In the context of this dissertation, applications of remote facility management technology or “automation” will be confined to the monitoring and control of architectural devices such as HVAC systems, lighting, and temperature sensors, as well as devices and appliances located within the facility. The entire point for using automation technologies is to be able to control devices remotely or automatically according to some predetermined plan. There are various reasons for having such control, which include convenience, providing for security, and saving energy. To facilitate the control, there must be a way to communicate the intentions and to actuate them on the device. Both of these require special hardware at both ends which can range in sophistication depending on the work to be done. The controlling of various devices is known as Direct Digital Control (DDC) technology.

On the controlling end, a simple adaptation of a television hand-held remote control unit coupled with an infrared receiver can be an inexpensive method of communicating commands. More complex methods of control include the use of computers that automatically communicate commands based on external factors including time of day, state of the building, temperature, performance specifications, etc.

On the actuating end, simple devices which open and close power flow, increment values, generate currents, and generally perform some form of work are necessary. These devices are called actuators. Actuators take either binary or analog commands. Since the signals being received are usually from a computer or controller, the signals tend to be fairly low voltage (up to 5V). These signals are not sufficient to power the architectural device itself, but can be used to flip relays that switch power on and off. This form of control is a digital signal, but using software functions that control many of these devices, sophisticated boolean AND's and OR's can be implemented. Analog devices actually use the electric signals to power devices, such as valves, dampers, and motors. Another form of control which is actually digital, but which can simulate analog characteristics is called “pulse width modulation.” Using this technique, timed binary pulses can be continuously derived which correspond to an integer count that the actuator interprets as a range of values. This latter technique is rather inexpensive, but not as precise, and can be used to brighten or dim lights and control thermostats.

In all control situations there must be some sort of feedback in order to verify whether the control command was realized or not, or to discover the current state of the device. In the case of remote control where the persons communicating commands can actually verify through their own senses that the work has been done, and no additional hardware needs to be added to the system. But in the case of computer control or remote control from some other location, additional features must be added to the system to facilitate the feedback. These additional devices are called sensors. Sensors used in the building industry can include devices which measure temperature, pressure, velocity and flow, humidity, air quality, motion, and light levels. Sensors can range in cost, sensitivity, and ruggedness depending on the application. Like the actuators, sensors are either digital or analog. An inexpensive analog temperature sensor is the thermocouple, where two different metals which are twisted together in a wire produce a current when heated. The current must be interpreted with a voltmeter and translated into digital data for the computer to use. Digital sensors use similar innate properties in different materials to detect differences and produce digital signals to the computer.

Having introduced actuators and sensors, the newest generation of “smart” actuators and sensors can be described. In addition to the functions mentioned above, these smart devices have microprocessors and means of communicating sophisticated digital signals for the purpose of interacting with higher-level computers and controllers. In the case of actuators, simple digital signals are received from the computer or controller. The microprocessor interprets the signals and causes the actuator to do its work. Microprocessors built into sensors interpret the input immediately and send off binary signals to the computer or controller. The advantage of having smart actuators and sensors is that the devices can become more standardized in a “plug and play” form. The media of communication between the device and the computer can be anything that allows the digital signals to be carried. This technique can include the power wiring itself, RF wireless, infrared, and all sorts of “hard wired” media such as twisted pair, fiberoptics, and cable [Newman 1994].

The use of smart devices opens up other opportunities. Instead of central control, the microprossesor present in each of the devices allows communication among the devices as well. A sensor device can send a message directly to an actuator device for some desired effect. For example, smart motion detectors can send a message to a smart switch to turn on a light in a dark room when someone enters. Controllers and computers are not eliminated, but become tools for human interface, programming of advanced sequences, storing of event records, gateways for remote control, and computer-aided analysis.

The use of smart actuators and sensors requires smart organization. The building DDC industry has taken technology developed in the computer industry and put it to use networking these smart devices in such a way that each can be controlled separately without being hard-wired. A fieldbus standard has been developed which gives each device its own identity for precision use in control situations. The term "fieldbus" refers to a series of nodes connected along a single line. One signal goes out to all the devices, but only the summoned device answers. Just as computers are given unique addresses and individual files have unique pathnames on the Internet, standards have been developed which give each smart device a unique address in the system. More sophisticated standards even have unique addresses that are worldwide for the very purpose of prividing an interface with the Internet.

Some advanced research on the use of smart devices has been conducted by various organizations around the world. In the United States, the National Association of Home Builders conducted research on a centralized hard-wired automation system called Smart House [Gilmore 1990]. In the Netherlands, the Dutch House of the Future built in the Autotron Theme Park in Rosmalen has become a research project for Dutch businesses, institutions, and universities. In Japan, Tokyo University has been working in partnership with top electronics industry representatives on the ongoing TRON project, which is another hard-wired example [Sakamura and Sprague 1989]. Another example is the research conducted by the Electronics Industries Association (EIA) which gave birth to the Consumer Electronics Bus (CEBus) standard mentioned below. The CEBus work is concerned with remote smart objects which can plug into existing wiring systems as easily as those using a hard-wired installation [Radford 1996]. In Norway, Gjovik College has teamed up with industry to construct a scale model town which is entirely controlled by LonWorks technology (described below). The town includes controls for everything from control of traffic, street lighting, domestic heating, access and various functions in automobiles.

3.3.4 Automation standards

The building industry has been plagued with a lack of common interface and standard for digital control hardware and protocol. In recent years, several distinct standards have evolved into workable solutions. Older applications of the technology have facilitated limited control, but because of the inexpensive hardware, are still a viable solution for remote control automation today. Other newer applications have more sophisticated control built into their smart devices, but are slightly more expensive. The one characteristic of all the standards that will be discussed is that they are all "plug and play" design, which can be easily expanded as the need arises, without special wiring or redesign of the system.

Before discussing commercial applications of the technology, a word on some overall standards is in order. One standard which governs overall networking is the ISO's Open Systems Interconnection Basic Reference Model (OSI). This standard consists of seven layers of independent data communication addressing different aspects of protocol. These layers are: the Application Layer, where the function or purpose for communication is established; the Presentation Layer, where the proper language or protocol for the intended destination is established; the Session Layer, where the priority of the communication is established; the Transport Layer, where the verification strategy is established (to assure the proper delivery of the communication); the Network Layer, where the destination address is established; the Data Link Layer, where the method of conveyance is established; the Physical Layer, where actual data and media (cable, twisted pair, etc.) are established. These layers provide a basis for the design of EIA-232 (RS-232) connections and protocols for sending packets of information between devices.

In the building industry, the American Society of Heating, Refrigeration and Air-conditioning Engineers (ASHRAE) have established a standard protocol for the purpose of enabling various smart devices, computers, and controllers to interoperate. The standard, known as ASHRAE Standard 135-1995, was coined Building Automation and Control NETwork (BACnet). BACnet attempts to address the seven layers from a building industry perspective. The Application Layer describes a collection of network-visible objects which define the function or purpose of the device to the outside without being concerned about the actual implementation of the technology. The Presentation Layer consists of “services” which provide commands for accessing and manipulating the objects. The other layers consist of encoding concepts, networking protocols (which actually can include the communication of any smart device to any other smart device anywhere on the Internet), and the medium on which the information is carried.

3.3.5 Commercially available applications

There are several systems and standards used for remote facility management or “building automation” applications that have developed over the last few years. The standards include X10, CEBus, LonWorks, and a few others. In order of increasing sophistication, three of these standards are described below.

The X10 is essentially a very low cost remote control system. The standard is an older one that predates the BACnet standard, but is quite popular due to its low cost. The system relies on the building's electrical wiring system and has no need of special wiring or installation. The system is entirely plug-in in nature, where switch modules plug into the nearest electrical outlet, and control units plug into any other convenient oulet in the system. Lamps, televisions, heaters, pumps, and other devices plug into the switch modules. Each of the switch modules can be set with their own unique address, up to a total of 256 different addresses. The controller can be manually operated or automatically set to send out certain signals to specific switch modules to either turn them on, off, or dim. When the switch module receives the signal, it closes or opens the electricity flow to the device allowing it to turn on or off.

X10 switcher modules can be used to control any device that takes a current. Some common uses are for lamps and household appliances, but may also include electrically-operated solonoid valves for sprinkler systems and bathroom plumbing, HVAC thermostat control, and security systems. The controller can also be connected to a personal computer for advanced programming. Timed events and graphic point and click control can be facilitated. An infrared (IR) interface unit is also available that can cause X10 signals to trigger IR commands.

X10's disadvantages include the fact that only on, off, and dim control is possible. There is no built-in feedback capacity to automatically determine whether the device was actually turned on or off. The X10 standard includes a product that partially remedies this deficiency, called HomeBase. HomeBase works in conjunction with a personal computer and performs some of the functions of a data logger, with up to 16 digital and 8 analog inputs, and 8 relay outputs. Various temperature sensors, moisture sensors, etcetera can be connected to the I/O interfaces for feedback.

The CEBus is an open architecture set of protocols for making devices communicate through power line wires, low voltage twisted pairs, coax, infrared, RF, and fiber optics. Contrary to the simple on, off, and dim of the X10 units, the CEBus standard allows devices to communicate via data packets that vary in length depending on how much data is included. The minimum packet size is 64 bits, but could expand to hundreds of bits in length. In addition to on, off, and dim, other defined controls include play, rewind, fast forward, pause, skip, and temperature up or down. The CEBus standard uses addresses which are manufactured into the hardware in the factory, which have up to four billion combinations.

The LonWorks standard also allows devices to communicate in a network environment through power line wires, low voltage twisted pairs, coax, infrared, RF, and fiber optics. The LonWorks standard also allows the devices to communicate with data packets varying in length from 64 bits to hundreds of bits. There are 32 predefined commands which include on, off, dim, play, rewind, fast forward, pause, skip, and temperature up or down. LonWorks standard allows for more than 32,000 devices on a single LAN, with addresses preset in the factory or alterable at time of installation.

The LonWorks standard uses the Neuron chip manufactured by Motorola and Toshiba. The system incorporates the ASHRAE BACnet standard and is endorsed by them. With LonWorks, all devices communicate via a common language, “LonTalk”, regardless of the manufacturer.

Comparing the three standards, X-10 is decidedly the least expensive, but does not lend itself to large facilities. Residences, small businesses, churches, and small commercial establishments could take advantage of the affordability and incorporate X-10 technology into their remote facility management systems. X-10 modules might be vulnerable to power surges and other signals penetrating in through the electrical system from the outside (even though filters are available.) On the other hand CEBus and LonWorks have a more robust design for protection against power surges, and are built for commercial use in networks with a greater number of devices. All in all the LonWorks standard seems to have the best backing and may prove to outlive or become incorporated in the others. There are integrated computer applications that combine the control of X-10, CEBus, and LonWorks together into one package. CyberHouse by Savoy Automation interfaces all three standards and is inexpensive.

3.3.6 Demonstration facility web page

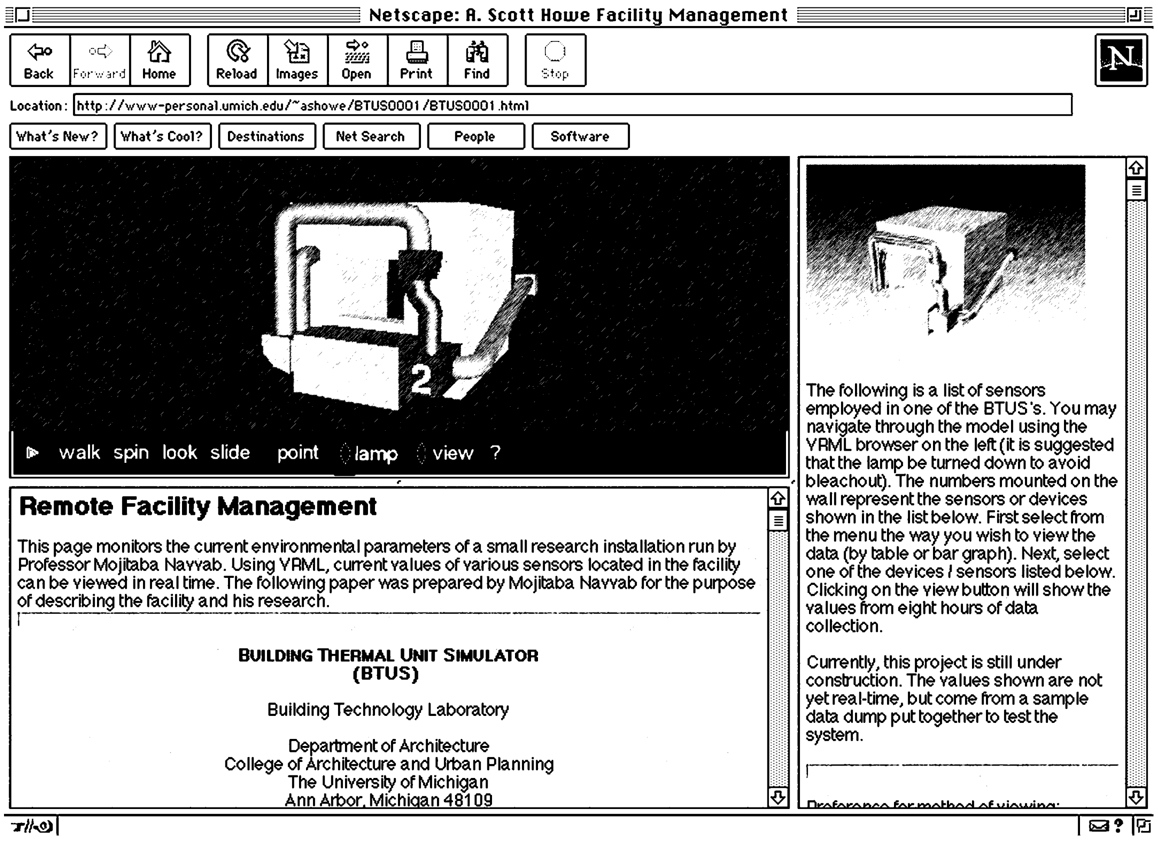
In the course of remote facility management studies, it was determined that an experimental research project should be conducted to demonstrate an example application of the technology. Among the five main media technologies of telephone, Internet, wireless, and cable, it was decided to develop an experimental software application that monitors and displays facility information over the Internet. Conceivably, a building or facility would have its own web page from which building managers and other visitors can observe its current state from any location in the world. This section describes an experimental web page set up for a facility at the University of Michigan.

The experimental software consisted of various independent modules which ran over a networked computer environment. The most visible part of the system was the home page, which is a panel made up of three separate frames. In one frame a VRML virtual building was placed, which had clickable parts for calling CGI’s that conceptually generated real-time graphs from data collected from the actual facility.

The experimental software was used to establish a web page for a small research facility maintained by Mojitaba Navvab at the University of Michigan College of Architecture + Urban Planning. The facility, called Building Thermal Unit Simulator (BTUS) is a pair of small 8' x 8' isulated chambers which have one removable wall which faces south. The chambers are exposed to the outside environment. The purpose of the chambers is to provide a controlled environment for experimenting with various types of windows and wall assemblies. The windows and wall assemblies can be attached to the south facing removable sides of the chambers and compared with each other for thermal insulative qualities. The chambers have their own independent HVAC systems in order to simulate the function of actual buildings.

Each of the BTUS facilities was connected to 30 sensors which produced a measurable current. The sensors include temperature, humidity, air flow velocity, and water flow sensors, as well as binary (on / off) sensors to tell whether fans and pumps are running, and pressure sensors to sense the position of dampers.

The sensors provided raw currents to a data logger, which is a hardware device which translates the currents into ASCII text data and dumped it into a text file on a computer. The scope of the experiment only covered the monitoring of sensors as picked up by the data dumps already provided and did not cover the automated dumping of the data itself. For the purpose of the experiment, a sample data dump from July 1996 was obtained.



Location: http://www-personal.umich.edu/~ashowe/BTUS0001/BTUS0001.html

Remote Facility Management

This page monitors the current environmental parameters of a small research installation run by Professor Mojitaba Navvab. Using VRML, current values of various sensors located in the facility can be viewed in real time. The following paper was prepared by Mojitaba Navvab for the purpose of describing the facility and his research.

BUILDING THERMAL UNIT SIMULATOR
(BTUS)

Building Technology Laboratory

Department of Architecture
College of Architecture and Urban Planning
The University of Michigan
Ann Arbor, Michigan 48109

The following is a list of sensors employed in one of the BTUS's. You may navigate through the model using the VRML browser on the left (it is suggested that the lamp be turned down to avoid bleachout). The numbers mounted on the wall represent the sensors or devices shown in the list below. First select from the menu the way you wish to view the data (by table or bar graph). Next, select one of the devices / sensors listed below. Clicking on the view button will show the values from eight hours of data collection.

Currently, this project is still under construction. The values shown are not yet real-time, but come from a sample data dump put together to test the system.

Preference for method of viewing:

Figure 22: A facility web page

When completed, visitors could view a VRML model of the facility using a WWW browser over the Internet. Clicking on various parts of the model that corresponded with sensor locations in the chambers, temperature values from the data dump could be viewed and compared via graphs compiled in real time.

3.3.7 Summary

Where the Internet and establishment of the information superhighway has layed a foundation for significant advances in the development of vast cyberspace infrastructures, the utilization of economical monitoring and control hardware devices such as those manufactured under the X10, CEBus, and LonWorks standards could link it all back to the real world with never imagined possibilities. As a simple experiment, a web page for a small facility was established which demonstrated data from a real facility garnered using a virtual model.

CHAPTER 4
VBUILD COMPUTER FRAMEWORK

This chapter describes an experimental browser / modeler which will allow a user to collect and assemble virtual kit-of-parts components from "component libraries" located on the Internet (such as manufacturer's databases) and assemble them into a virtual representation of a building. The fully assembled virtual building will provide a basis for ordering and manufacturing actual components and preparing for construction. The browser will allow the designer to affect a limited degree of remote fabrication at real manufacturing facilities, and facilitate eventual interface with built in sensors and actuators. The browser will manipulate and display interactive three-dimensional objects using VRML. Upon assembly, actual components will have sensors built into them to provide data about the real building, which could be viewed during a walkthru of the virtual building by clicking on parts of the model. The virtual building will work as a remote facility management tool for monitoring or controlling various architectural devices attached to the real building (such as electrically driven louvers, HVAC systems, appliances, etc.) The browser (hereafter called VBuild) makes maximum use of the two powerful concepts of object-oriented programming and the kit-of-parts philosophy.

4.1 Object-oriented programming

VBuild was programmed using an object-oriented structure in the Java language. Object-oriented languages utilize data and code in discrete structures called classes. Once instantiated, the class defines types of data called objects. The data itself is protected and can only be manipulated via pre-defined methods. The methods are interfaces with the rest of the program. Once the interfaces are defined, the actual coding for implementing the interfaces can take on any form as long as it supports the interface methods. In this way specific elements in a model can be defined independently of other code according to function or behavior. Methods and attributes specific to that element's behavior can be added as needed to give the objects completely expandable capabilities.

In an object-oriented program, once a class has been debugged it becomes a reliable building block in the entire coding of the program. Object-oriented design becomes a clean way of defining functionality without having to deal with loose ends associated with the complexities of unstructured, non-object coding.

The active use of the kit-of-parts philosophy was an important element in the conception of VBuild. Kit-of-parts components can be thought of as objects in an object-oriented programming environment. With well-defined interfaces which are rigorously followed, the component itself can assume any form. Interfaces can include mounting points, rules for the transfer of loads, specifications for thermal performance, and maximum cost constraints. In short, a kit-of-parts approach lends itself to cheaper and more efficient manufacturing and is a clean way of demonstrating a network-based virtual building system without having to deal with loose ends associated with the complexities of unorganized raw materials.

4.2 VBuild core classes

The classes or objects which have been devised for the browser mostly fall into three main categories: Geometry classes, Assembly classes, and Construct classes. The Geometry classes consist of classes which define geometrical representations of objects, and include 0D, 1D, 2D, and 3D geometry. The Assembly classes define specifications, function, and fabrication processes associated with individual components in a kit-of-parts. An assembly is a discrete component which is manufactured using a combination of different fabrication processes, and follows interface rules for connection to other components (in this work, "assembly" and "component" may be used interchangeably and refer to the same thing). Construct classes define ways of organizing the assemblies.

The POINT class represents the most basic form of geometry or 0D geometry. One point object contains XYZ coordinates as data, and include various methods for manipulating the coordinates. See Figure 23 for the product model diagram of the POINT class. The diagram was produced using a simplified form of NIAM modeling notation (see appendix for a reading guide.)

The LINE class (Figure 24) represents 1D geometry and includes both lines and segments. A line has two point objects as data and includes methods for manipulating itself.

The POLYGON class (Figure 25) represents 2D geometry of enclosed shapes. Eventually it will also represent compound 2D elements such as splines, curves, and open shapes as well, but for now only closed shapes constructed of a series of connected vertices is implemented.

The SOLID class (Figure 26) represents 3D geometry. Eventually it will also represent curved surfaces as well, but for now only faceted solids are implemented.

The PART class (Figure 27) potentially contains one solid or one polygon or both, as well as a material definition. A part is meant to represent a raw material with limited definition of a potential shape. The solid can represent a cast object or folded sheet fabrication, where the polygon would represent a section for extruded objects or the shape of a cutout from sheet stock.

The FABRICATION (Figure 28) class has exactly one part as well as paths for extrusion, and will include methods for creating G-code for actually machine fabricating the part (G-code defines paths for manufacturing tooling). Implementation of these methods is currently underway, and will be improved as new fabrication methods are adopted.

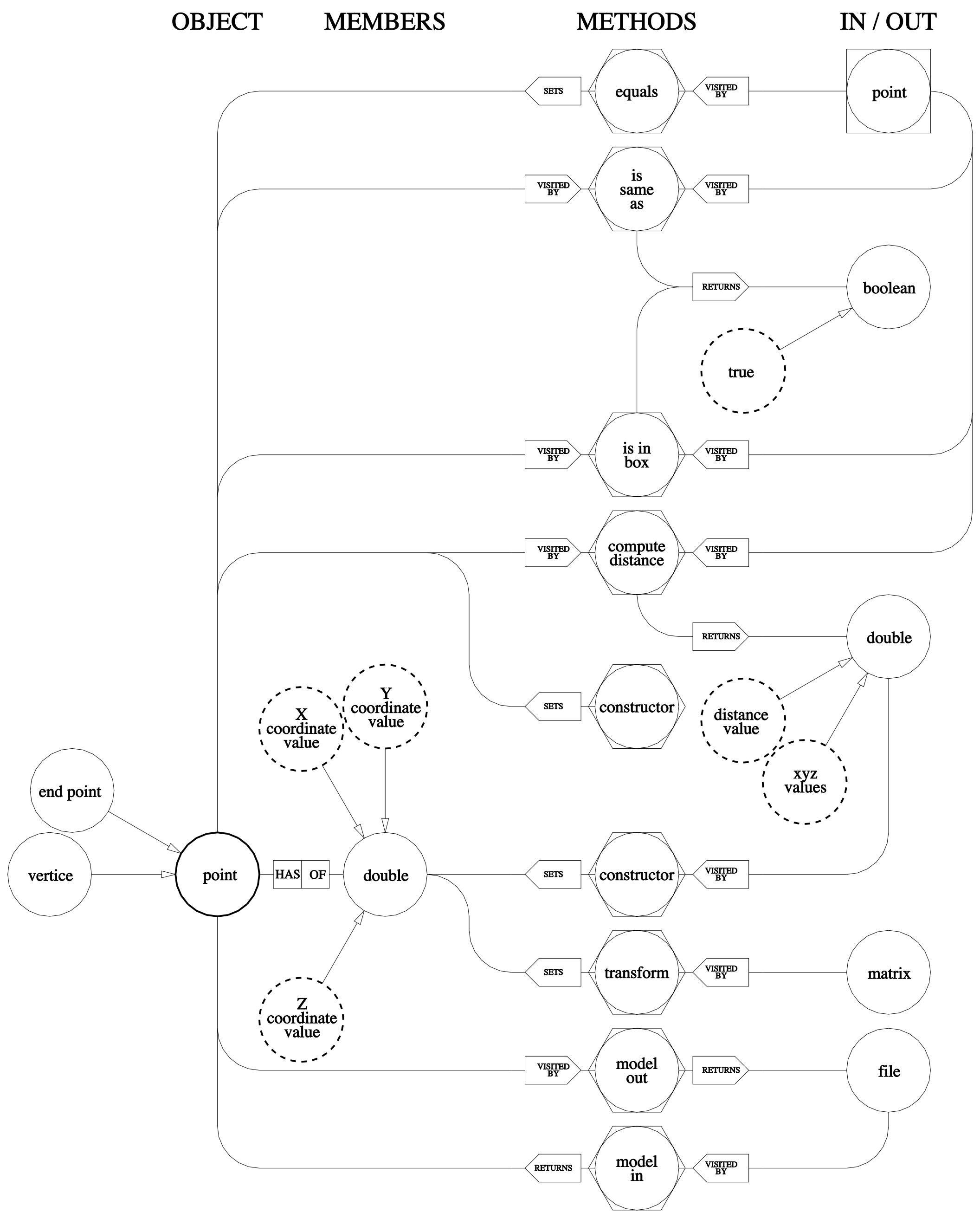


Figure 23: POINT class product model diagram

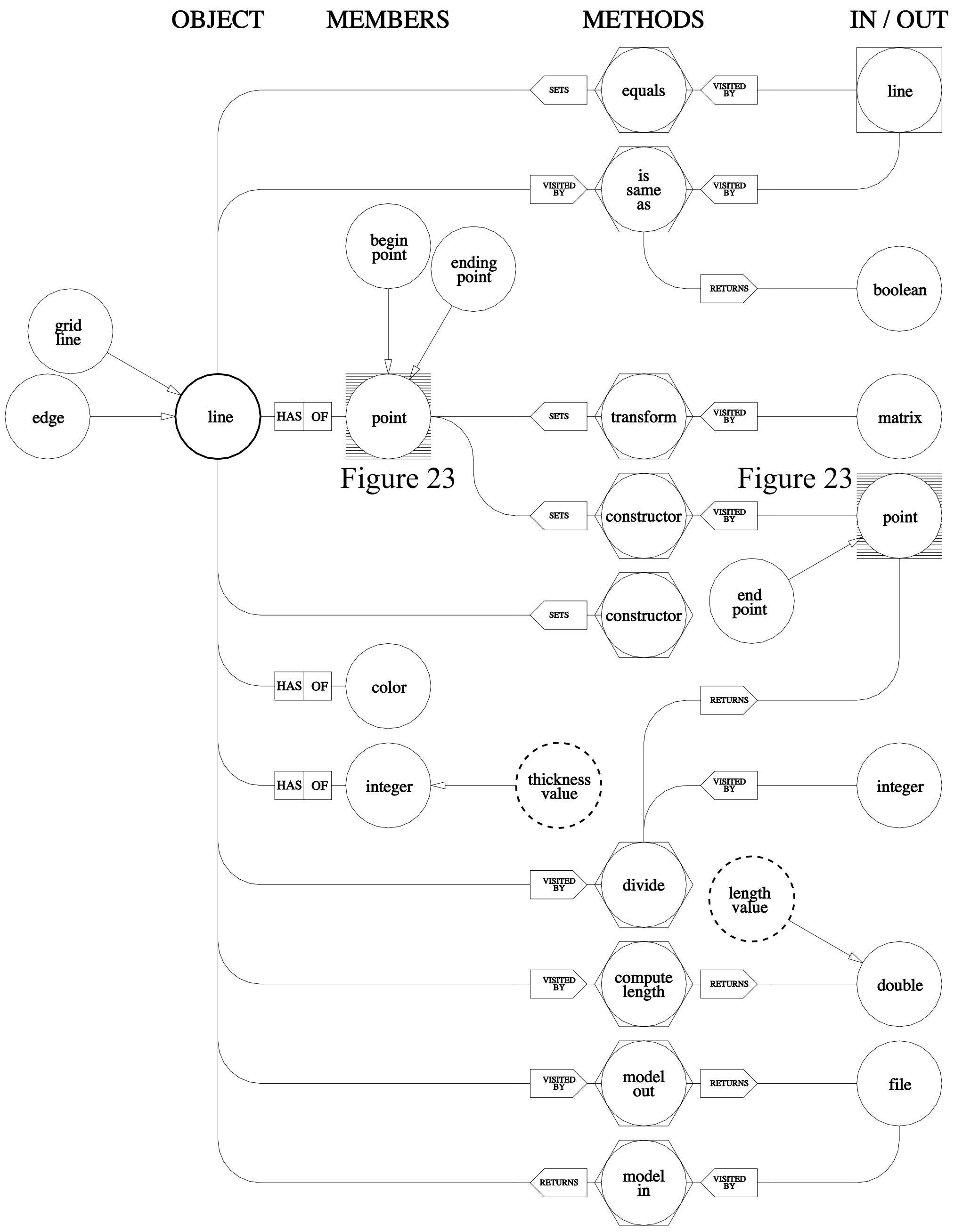


Figure 24: LINE class product model diagram

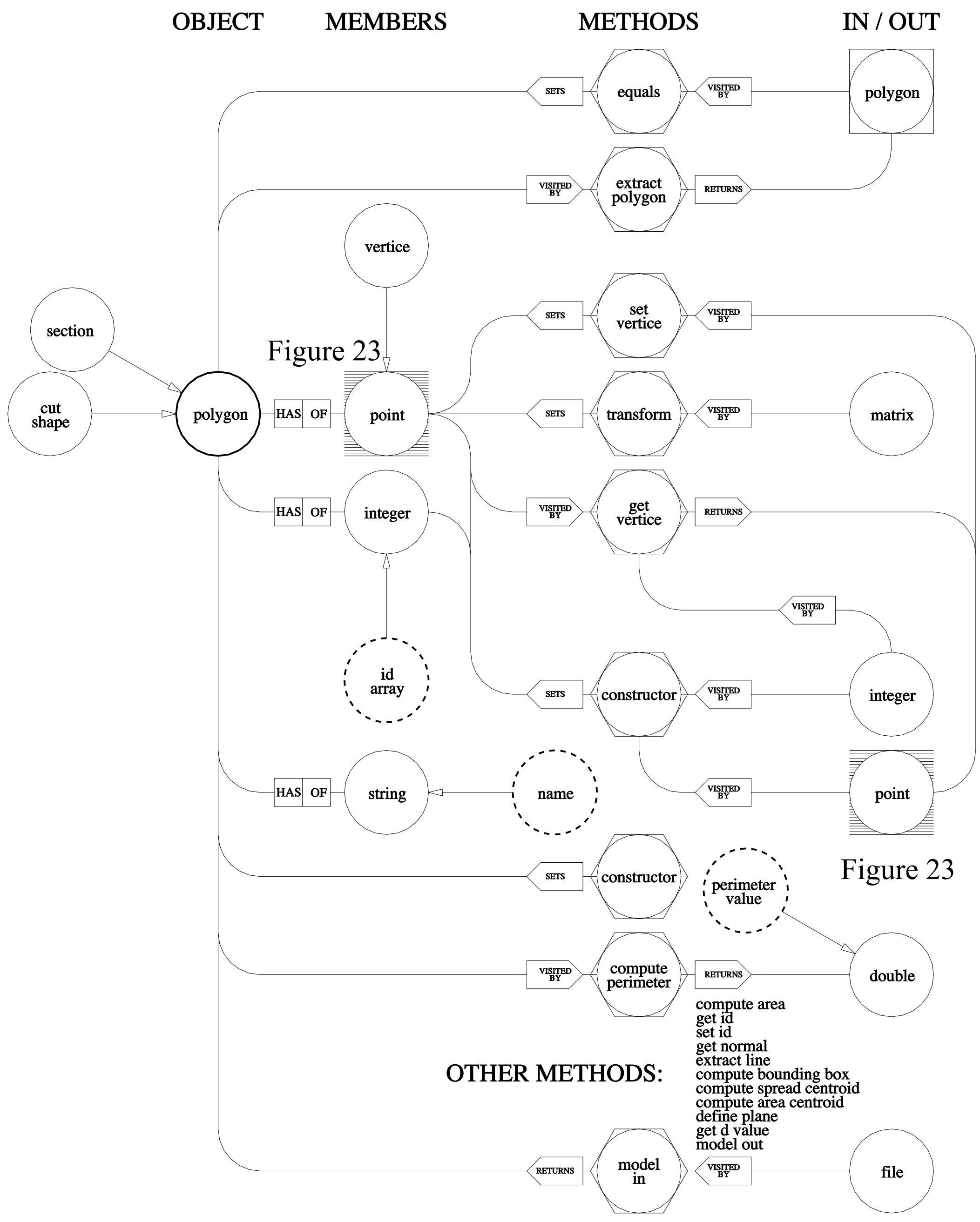


Figure 25: POLYGON class product model diagram

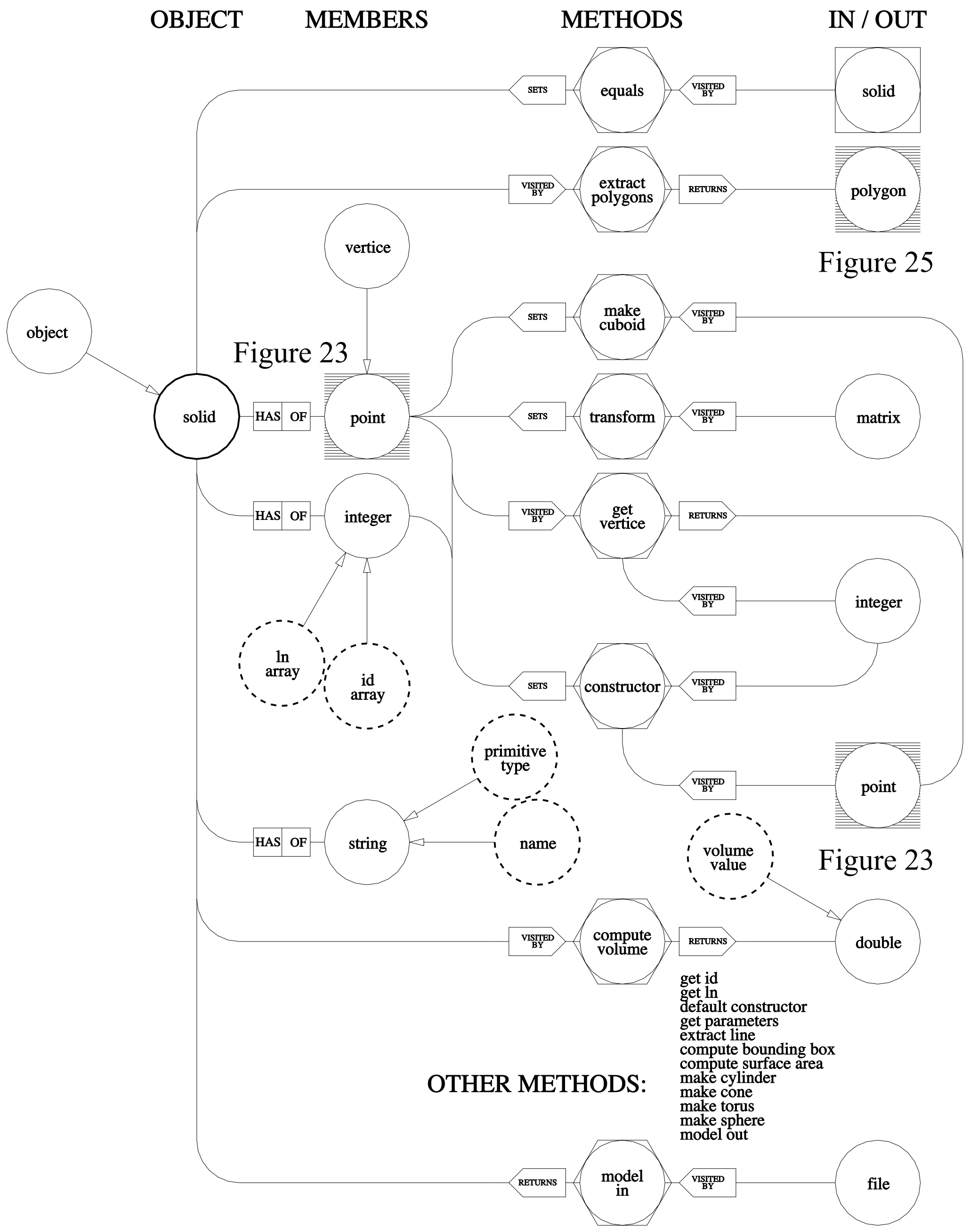


Figure 26: SOLID class product model diagram

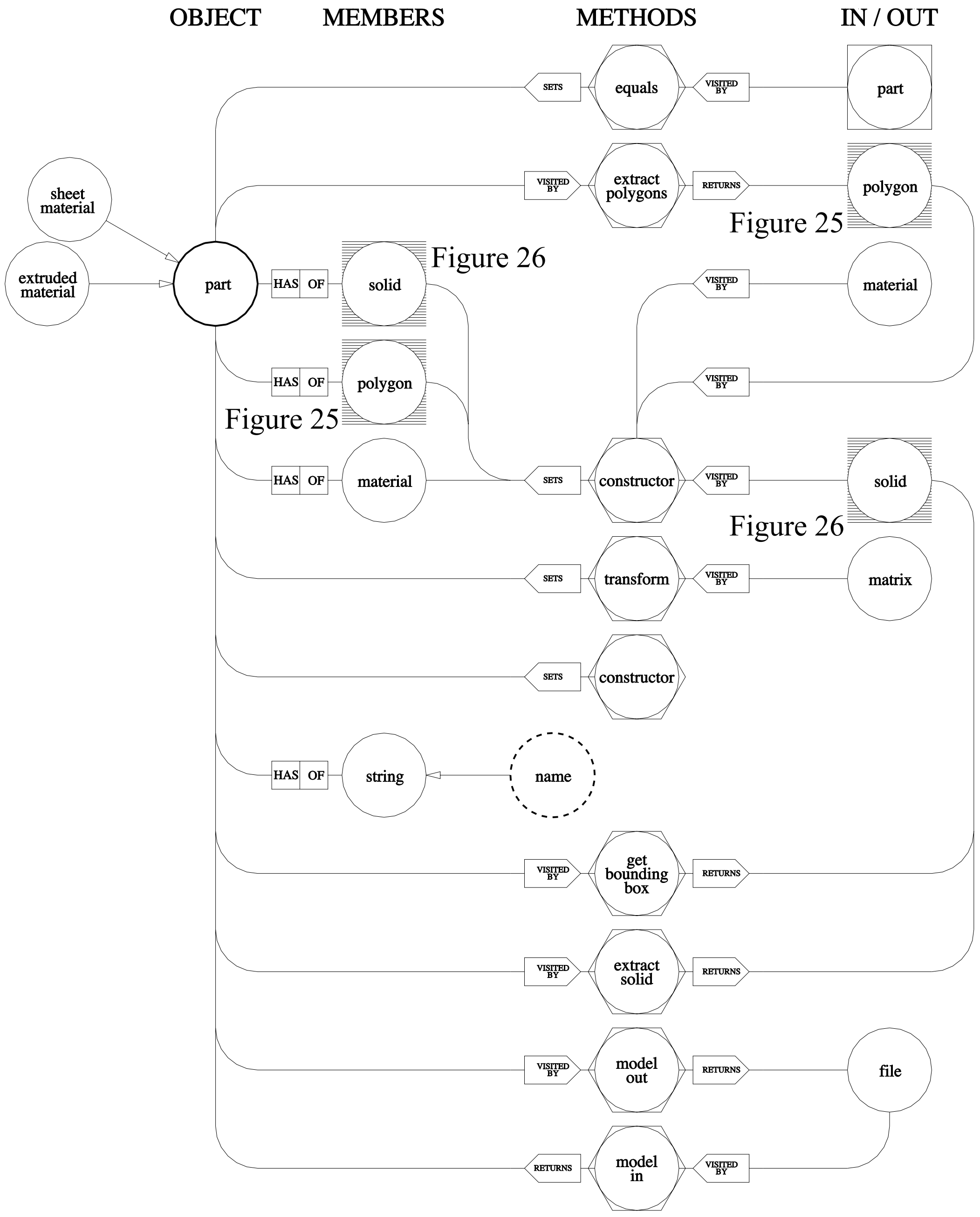


Figure 27: PART class product model diagram

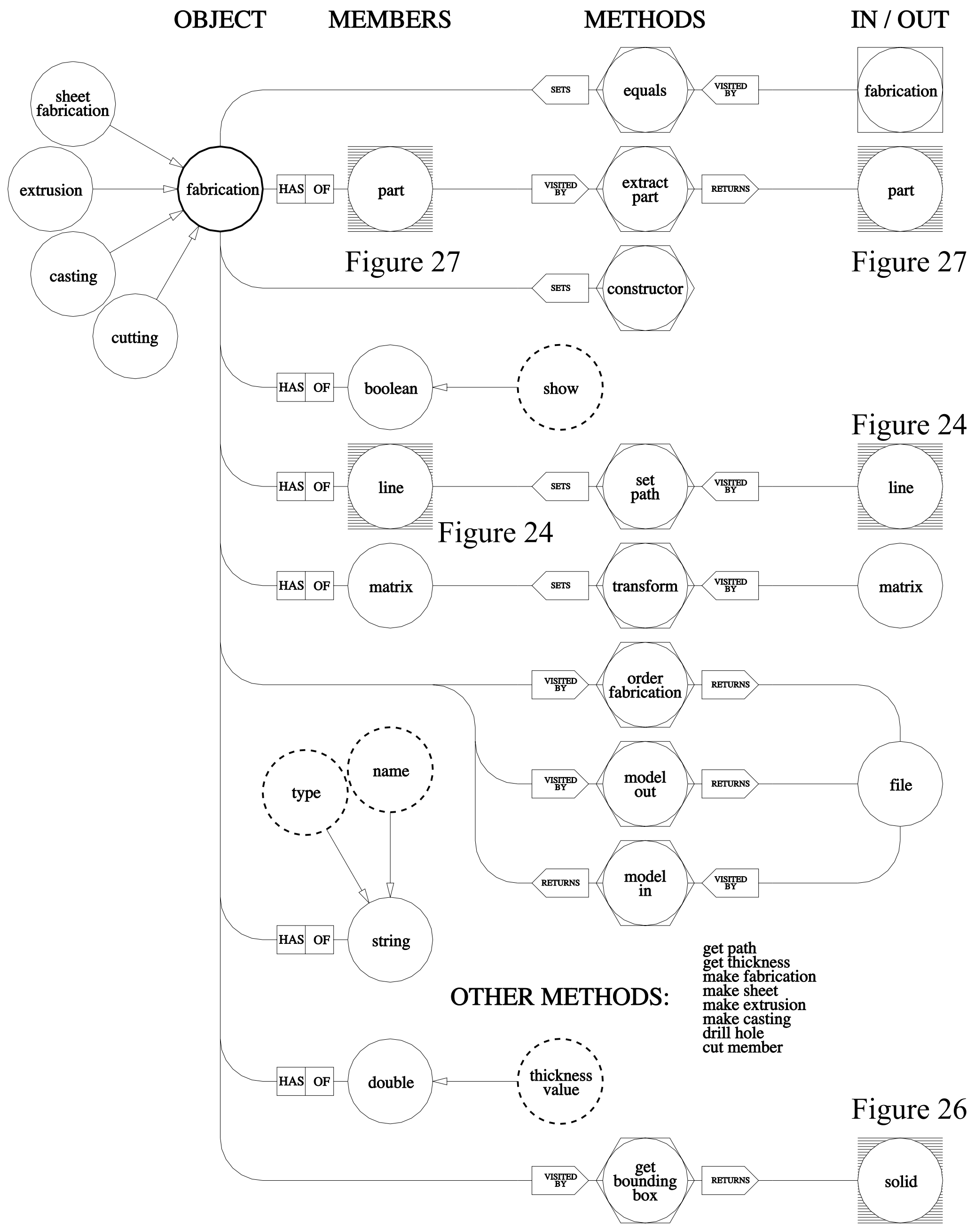


Figure 28: FABRICATION class product model diagram

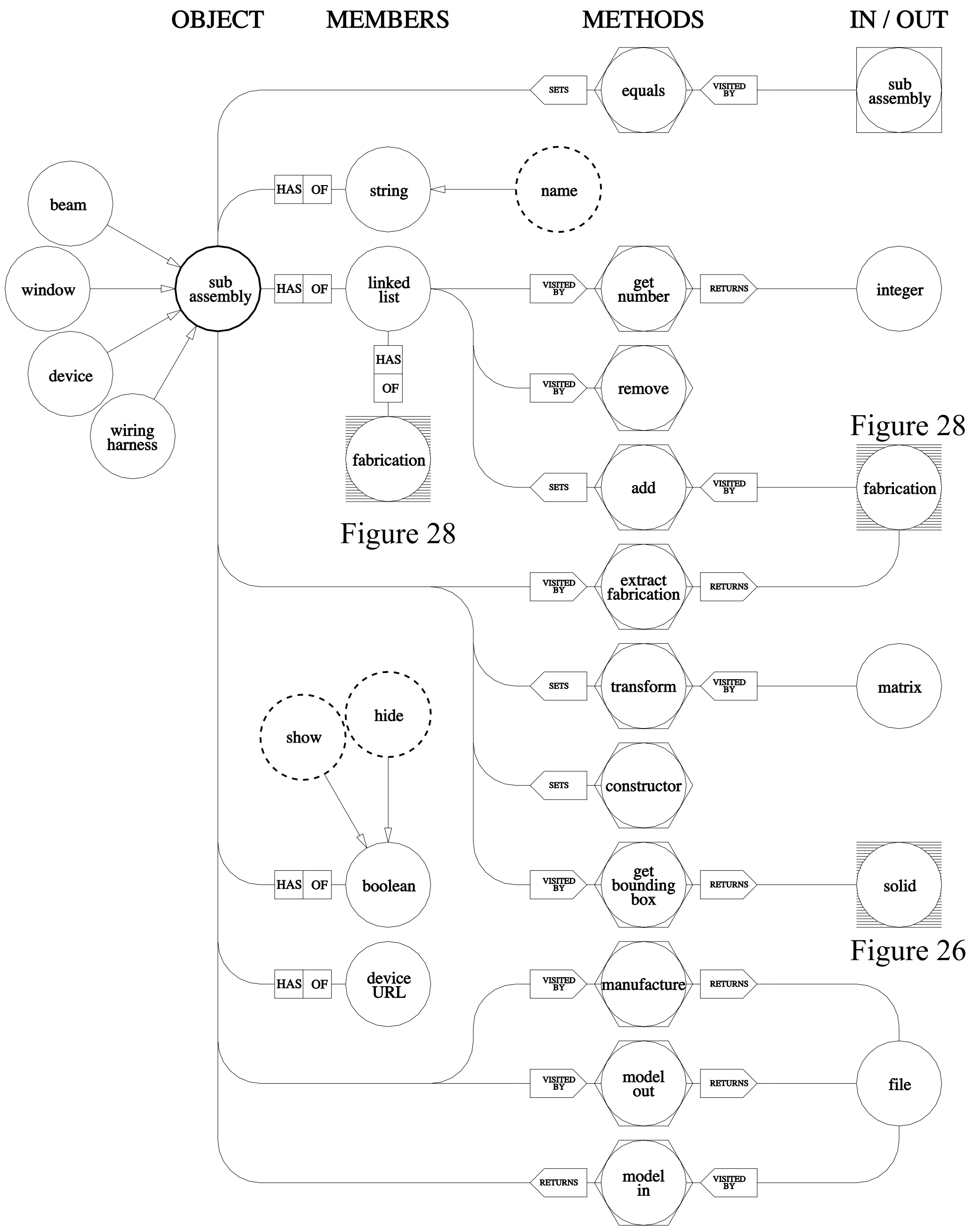


Figure 29: SUBASSEMBLY class product model diagram

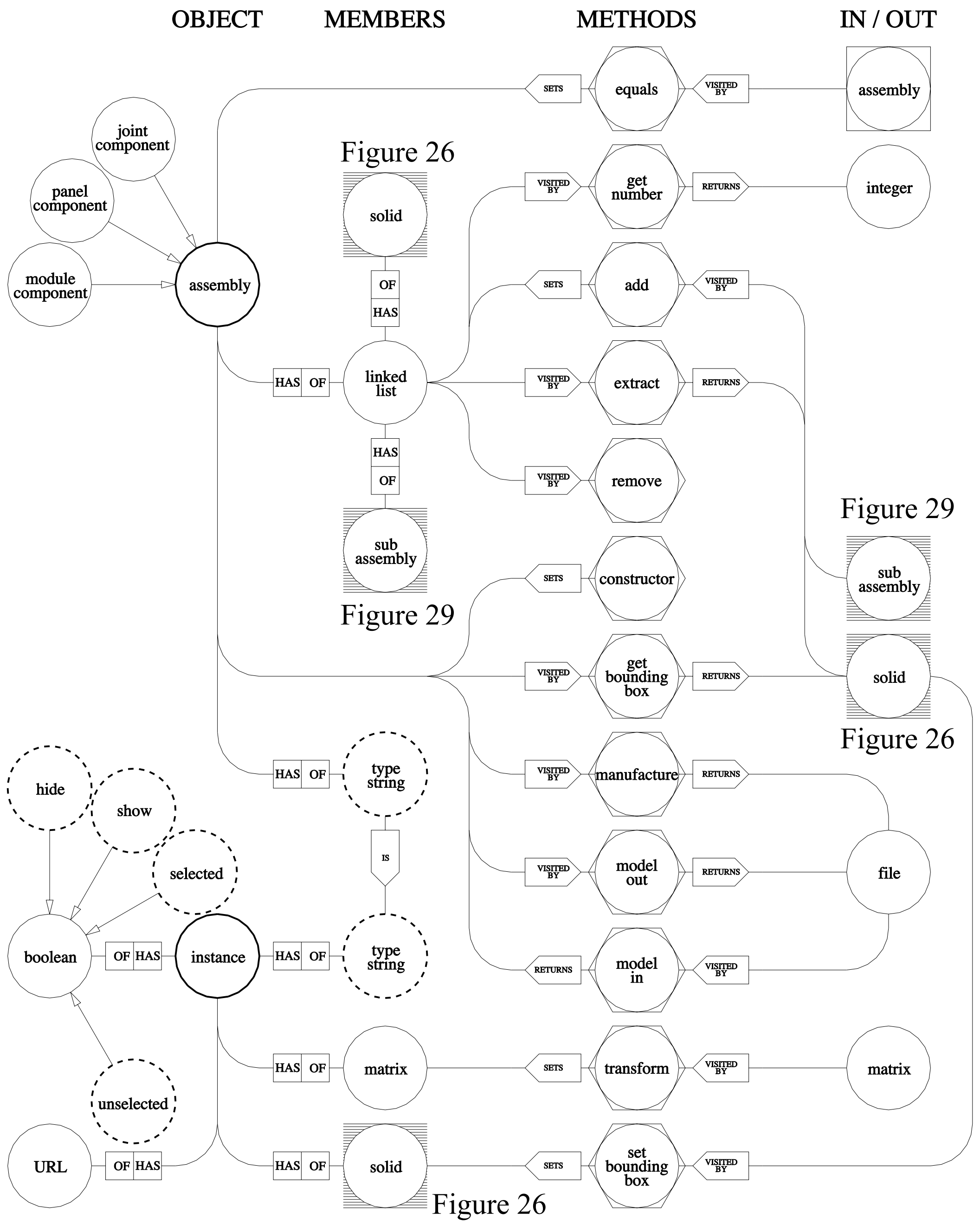


Figure 30: ASSEMBLY class product model diagram

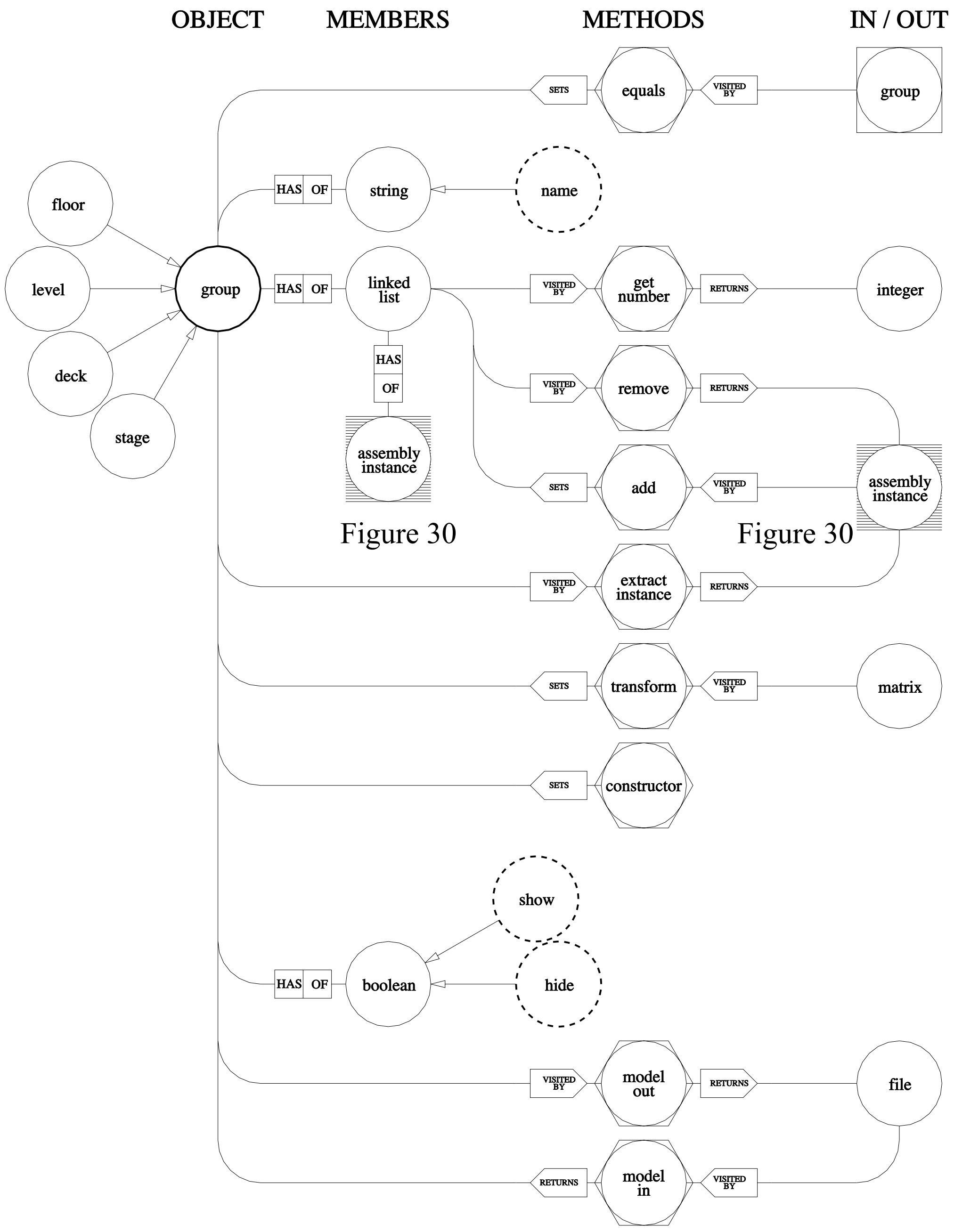


Figure 31: GROUP class product model diagram

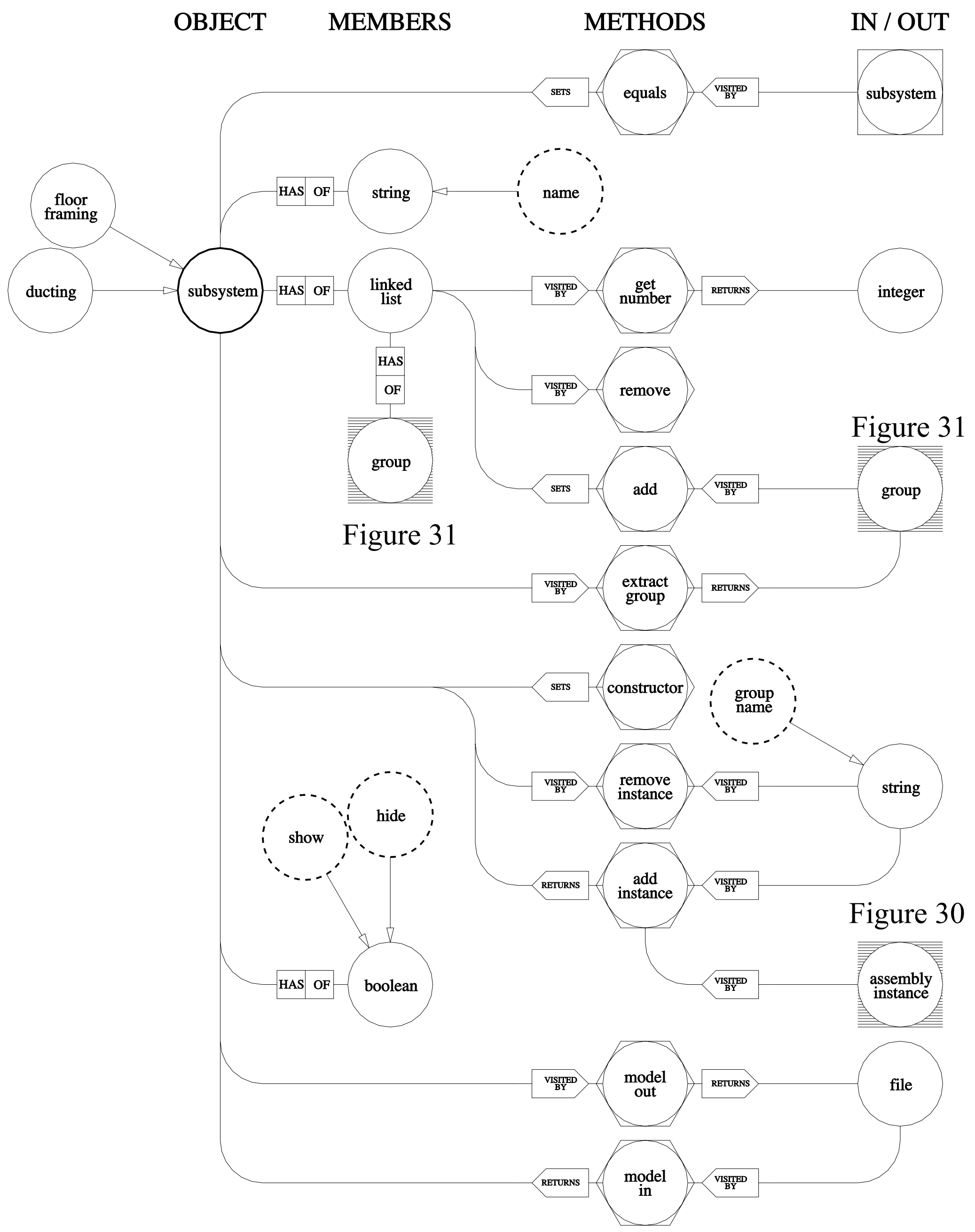


Figure 32: SUBSYSTEM class product model diagram

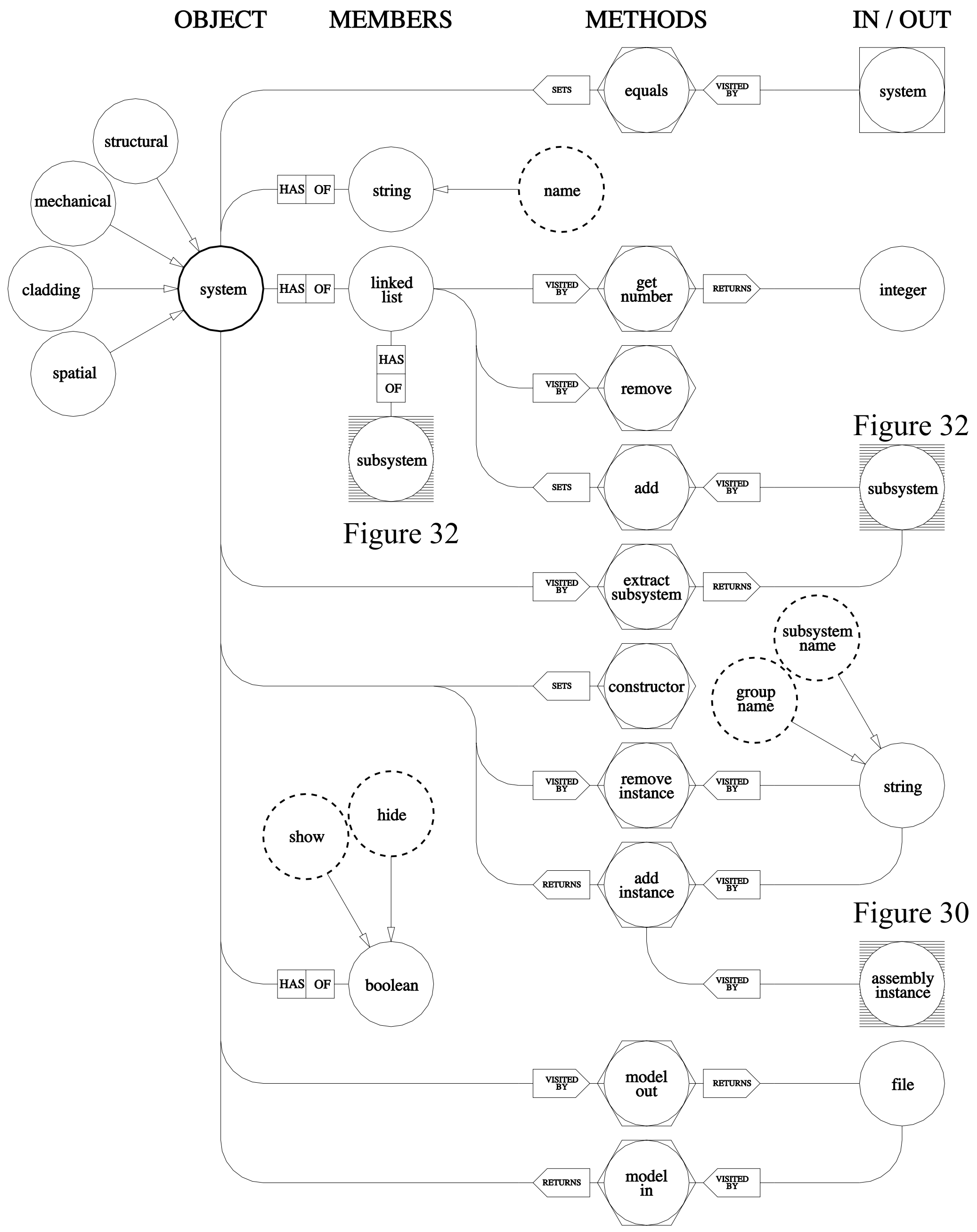


Figure 33: SYSTEM class product model diagram

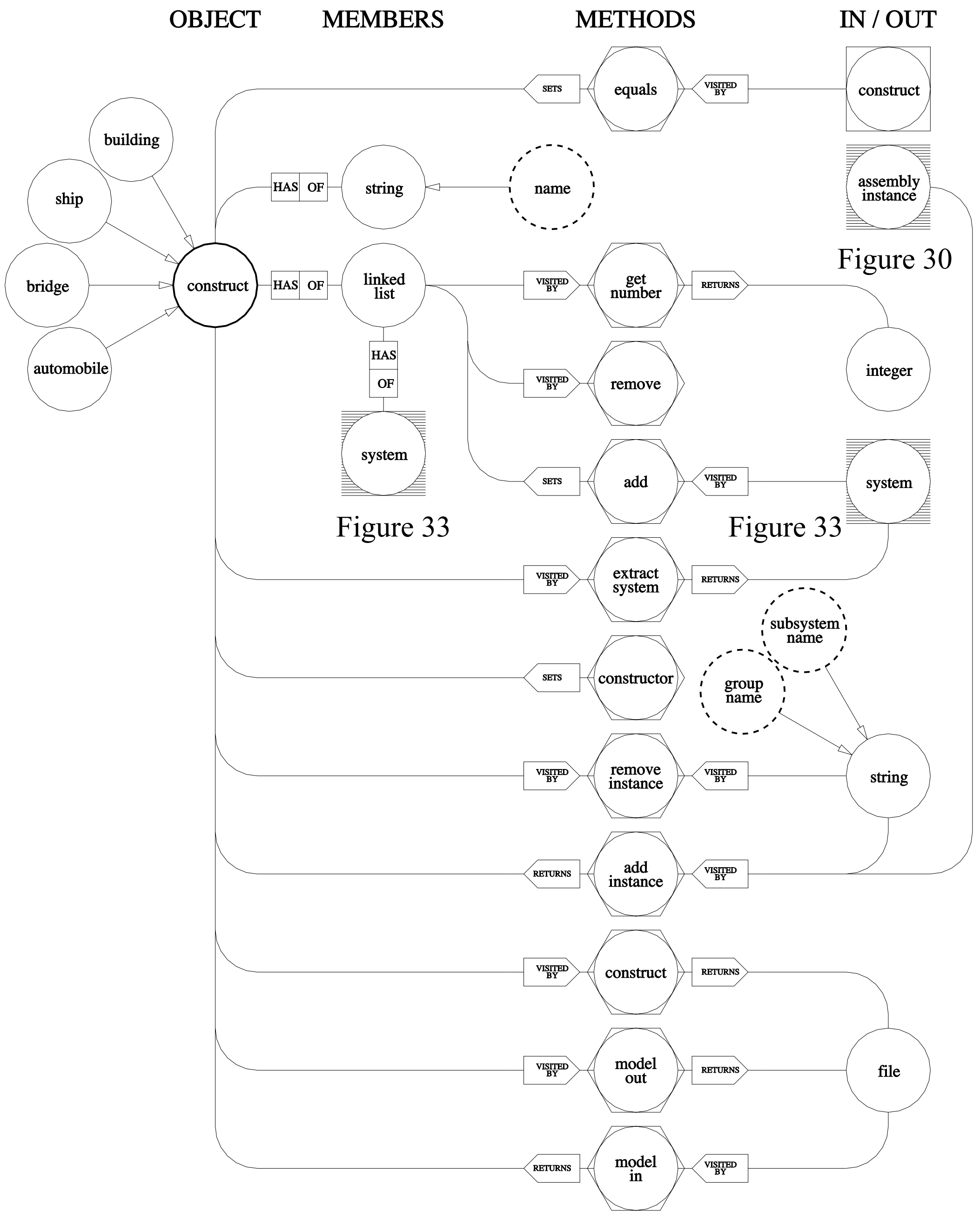


Figure 34: CONSTRUCT class product model diagram

The SUBASSEMBLY (Figure 29) class can have many fabrications, as well as links to sensors and actuators that potentially can be embedded in the assembly-component.

The ASSEMBLY class (Figure 30) represents a kit-of-parts entity. It can have many subassemblies and will be the largest object that would potentially be manufactured off-site. Each assembly of a certain type only exists once and is instantiated as many times as necessary. The instance will represent real manufactured components assembled on the site.

The GROUP class (Figure 31) is a special collection of instances of assemblies which have some common purpose or function, such as all the column assemblies of a certain type that belong to a certain floor of a building. A group can have many instances. Methods in the group class will manipulate groups.

The SUBSYSTEM class (Figure 32) is a collection of groups which have a common purpose or function, such as all the columns of a building. Each floor may have its own group of columns, which added together would constitute the column subsystem. The subsystem will contain methods for manipulating subsystems.

The SYSTEM class (Figure 33) is a collection of subsystems, such as the structural system of a building. The structural system will include column subsystem, floor framing subsystem, and foundation subsystem. System class methods will manipulate, view, or analyze entire systems.

The CONSTRUCT class (Figure 34) is the finished artifact, which includes all the systems. Construct methods will define behavior of the construct, and include mechanisms for viewing.

The VBuild core objects in their current state do not represent an ideal generic building model such as that described by [Turner 1997]. Instead, the classes represent a production-oriented structure ideally containing data required to run numerically controlled (NC) machines over a network environment, and generate geometric approximations of building components for virtual walk-thrus. Rethinking the core objects to conform to a generic building object concept could be the focus of future research (discussed in later chapters).

4.3 Design environment

Virtual representations of kit-of-parts components are held on a server connected to the Internet. These components are the ASSEMBLY data structures in the VBuild core class hierarchy, and are grouped in libraries according to the kit-of-parts system to which they belong. The representations held in the kit-of-parts library consist of files that have the minimum amount of data required to expand or fill-in the VBuild core class data structures. The assemblies are high-level parametric primitives consisting of collections of cuboids, cylinders, cones, frustums, and other solids that can be defined with a limited amount of data. Since the amount of data is small, Internet transfer can occur very rapid. The local program then fills in the rest of the data according to a predetermined formula based on the type of primitive and creates an instance of the component. When the user wants to insert another component, the VBuild application will contact remote virtual librarians which will return requested components according to type.

Each time the same type of assembly is requested by the user, another instance is created from the previously downloaded data. When the user saves the construct, the actual data of each assembly held in RAM is discarded, and only the instance references and their locations are preserved. Each time that model is opened, the real data associated with each instance is once again downloaded in real time from the source. Using filters and cameras, the user is able to view the data in various ways which may include sections and plans. Also, in a fully operational VBuild, the design environment will open up a window on specialized web-based analysis programs and expert systems which can be used to analyze the building or artifact.

In the current version of VBuild, remote virtual librarians are not implemented. Only a simple arrangement which uses the same server upon which the VBuild Java code resides, and reads each component directly, has been implemented. For this reason, the current version only reads the data as is, or in other words as VBuild data. When the virtual librarian is implemented in Java Remote Method Invocation (RMI) code, it will be possible to place libraries on more than one server (such as manufacturer's or designer's servers), and return requested data formats (return VRML representations in addition to VBuild data).

The product of the VBuild design environment is a dynamic virtual building or construct. Being dynamic, the virtual construct is never complete or static, but is always open for addition, partial or full disassembly, or rearrangement according to new needs. Using VBuild, occupants can conceivably build additions onto their residences simply by making changes on the web page.

4.4 Fabrication environment

Once the building is virtually assembled, the assemblies can be manufactured and delivered. Each assembly may consist of many fabrications. Since the fabrications contain the data necessary for their own manufacture, the VBuild program will contact a virtual contractor and request an estimate based on fabrication type. The contractor will select a manufacturer based on price and wait list and return an estimate. The ideal setup would consist of a simple list on the manufacturer's server which would hold current setup and processing rates, with another list on the material supplier's server which would hold current material costs. A queue could also be maintained on the manufacturer's side which would hint at possible wait lists. When the manufacturer finishes processing the fabrication, it would be shipped directly to the designer or another designated address (such as the site). In the case of multiple fabrication types in a single component, a system of bar codes can facilitate the forwarding of one finished fabrication to the next manufacturer, who will build upon the previous work until the entire component is built and sent to the site.

When the component is delivered to the site, VBuild would conceivably affect final assembly. Using robots and automated construction machines, the kit-of-parts components would be fit together and assembled according to the same arrangement as the virtual building which was produced in the design stage.

In the current version of VBuild, the fabricate / build environment has no significant functionality as compared with the design environment, but only displays non-functional displays describing how the program would work in concept. However, exercises which prove the feasibility of the functions were conducted by the author and are discussed in later chapters.

4.5 Use environment

Once the building is assembled in the real world, the virtual building or artifact can become a management tool throughout the life of the actual artifact. Using network-based software and remote control hardware, control and monitoring of various devices located in the building can be affected. An HTTP or remote access server can be paired with a virtual facility manager on a local computer installed in the finished building. The HTTP server could be used to facilitate an Internet connection, as opposed to a remote access server which supports private phone connections. Each subassembly in the VBuild core class data hierarchy has the capacity to link to a remote program which can handle bitwise communication with a LonWorks or X-10 interface connected to the computer's serial port in order to facilitate plug and play monitoring and control of devices on a powerline network or dedicated bus. These devices can plug into a standard 110 volt outlet, and send and receive signals along the power wire to and from other similar devices. In this way, the virtual building functions as a three-dimensional facility home page. The user would be able to walk-through the VRML model of the building and click on various parts of the model to affect monitoring and control. Upon clicking on the link, a Java control panel prepared for that device would be brought up giving access to direct digital control or real-time data published by the device.

As was mentioned earlier, the ideal incarnation of VBuild would allow the user to manipulate the virtual building and then execute the changes on the real building while sitting in front of the computer. Analysis programs that evaluate the state of real-time building data could be executed in order to determine inefficiencies in the design. Since the building is a dynamic system rather than a static artifact, changes could be made which make the design more efficient. VBuild could also be used to disassemble the artifact where the components would be reabsorbed for environmentally-friendly disposal or recycling.

The current version of VBuild includes the functional capacity for remote monitoring and control. As will be discussed in later chapters, links from the virtual artifact to the real one have been created and experiments have been conducted.

4.6 Summary

VBuild is an experimental software which attempts to prove the feasibility of advanced life-cycle management of kit-of-parts building systems, where life-cycle includes the design, real-time fabrication / construction, and real-time remote monitoring and control of the final products. Using a set of core classes, the entire building or artifact is represented virtually in a production-oriented way. In addition to geometrical data, the core class data structures contain methods needed to manufacture individual fabrications. Additional methods include the means to link the core data structures with their physical counterparts in order to affect monitoring and control.

CHAPTER 5

A MODEL BUILDING SYSTEM

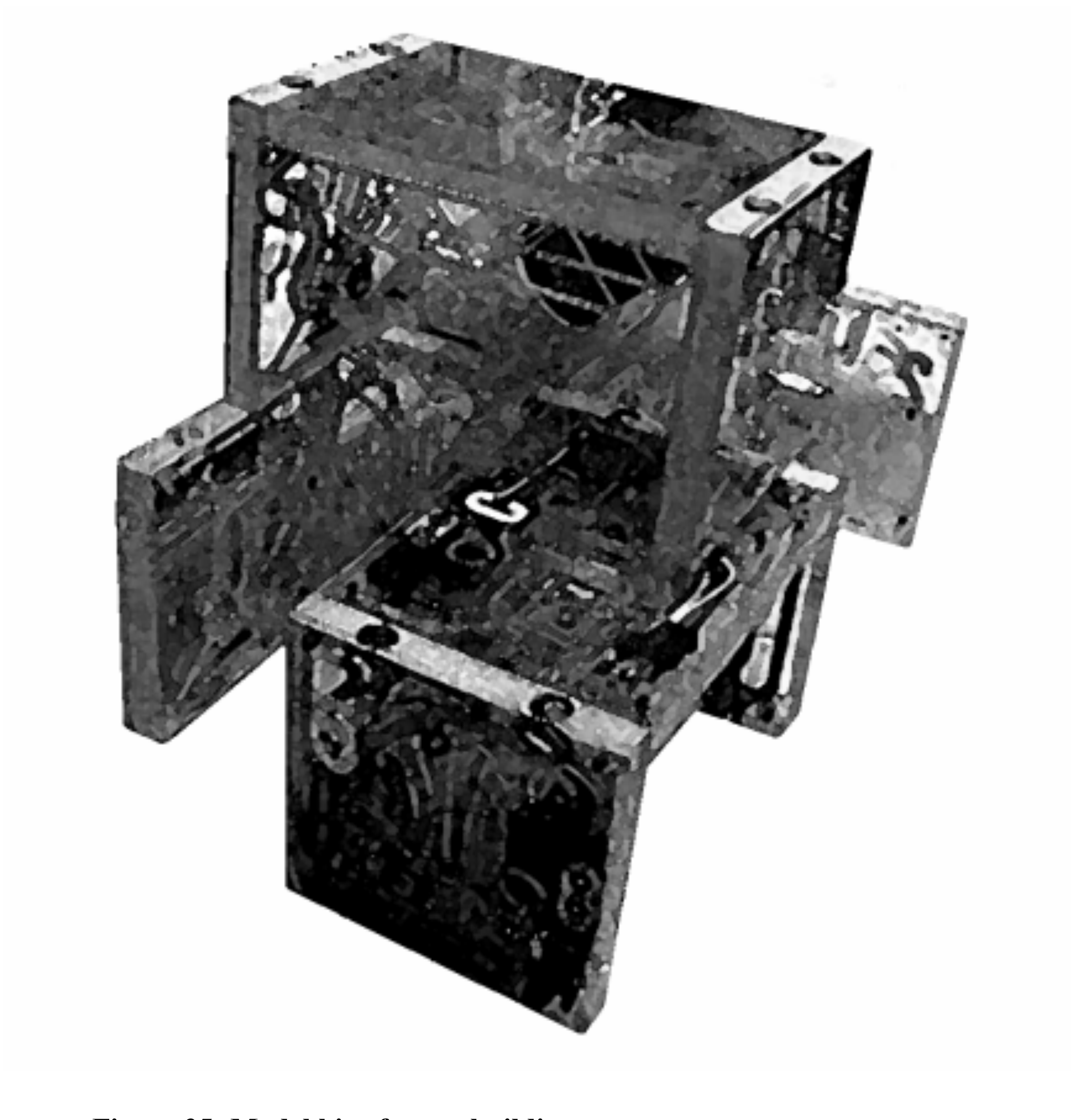


Figure 35: Model kit-of-parts building system

This chapter describes a model kit-of-parts building system which simulates the various stages of the design, build, and use processes, using the life-cycle management concepts suggested by the VBuild research. The model was devised as a testbed to prove the feasibility of each of the VBuild environments. A complete panel-based kit-of-parts was designed, which included only two component types: a wall panel (Figure 39) and a floor panel (Figure 40), with structural and electrical power interface schemes.

In the course of the simulation, full virtual representations of the components were modeled, including geometrical data and fabrication data. Using the virtual model, eight real wall panels and four real floor panels were manufactured for use in the simulations and experimentation. The steps described below were not necessarily executed in this sequence since the attempt of this work was to prove feasibility at each step using the same kit-of-parts.

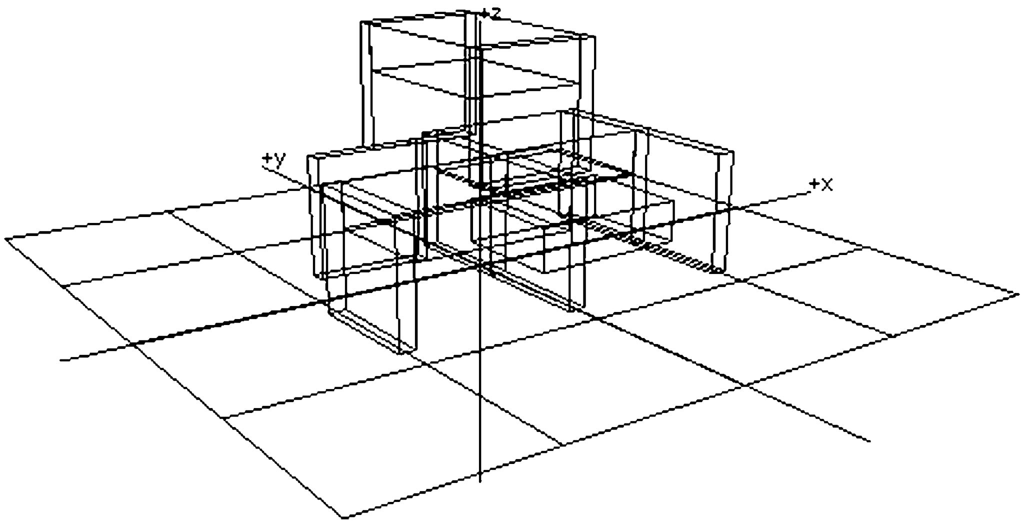
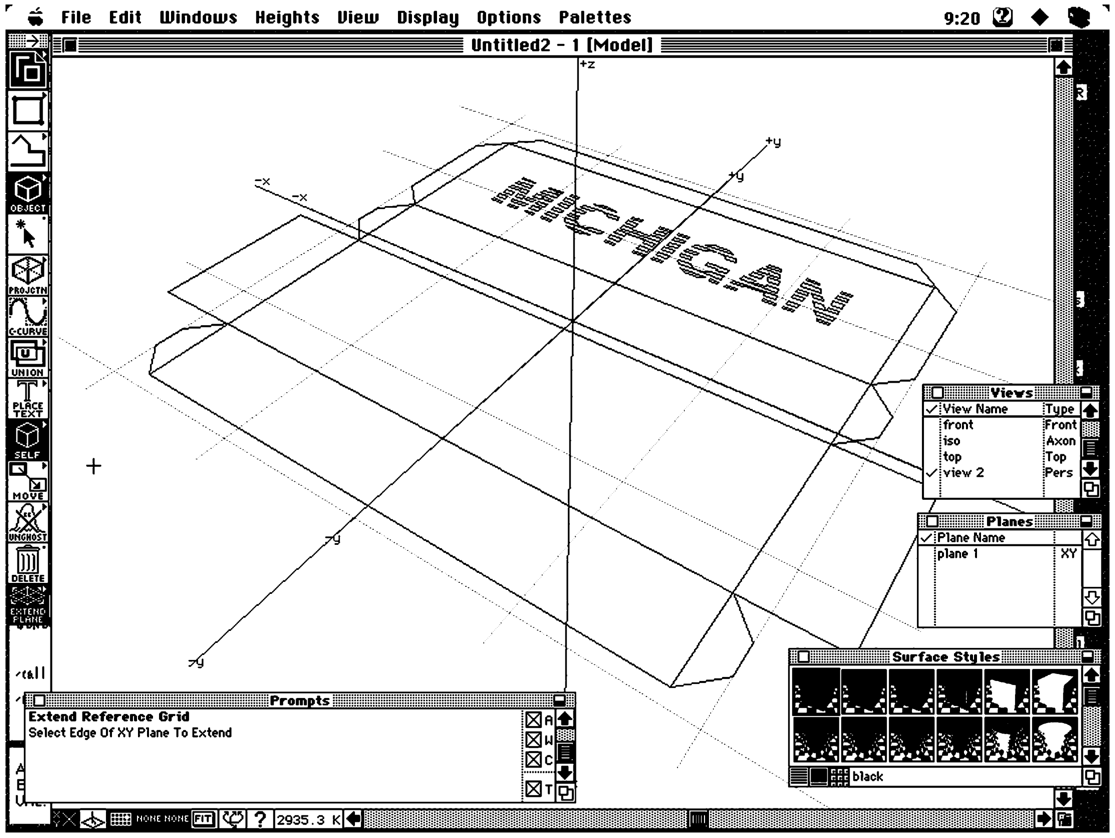


Figure 36: Component placement using VBuild design environment

5.1 Remote design

Once the VBuild representations of the components are loaded onto the remote server, virtual model buildings can be assembled in the computer from any terminal connected to the Internet. On numerous occasions the author used VBuild to assemble single and multi-level virtual buildings (Figure 36), using a variety of computer platforms (including PC and Macintosh). Components were downloaded from their remote locations and manipulated on the VBuild screen to the desired locations. The virtual buildings which were assembled were fully modelled using the core class data structures including CONSTRUCT, SYSTEM, SUBSYSTEM, GROUP, and ASSEMBLY core objects. On the screen the components were represented as bounding boxes of the geometrical data held within the assembly hierarchy.



View Name	Type
front	Front
iso	Axon
top	Top
✓ view 2	Pers

Plane Name	
plane 1	XY

Extend Reference Grid
Select Edge Of XY Plane To Extend

Figure 37: Early attempts: using a CAD system to model fabrications

Once downloaded, each component was added to a list of assemblies which contained the fully developed VBuild data structure held in RAM. Placement of the component on the screen created an instance of the assembly which was loaded into the virtual building model.

5.2 Remote fabrication

Using the model kit-of-parts, a series of experiments were conducted using a real online manufacturer. Initially a simple program conceived as a plug-in for common WWW browsers was developed which allowed a designer to select a DXF file from the local computer environment (Figure 37). The plug-in program would send the chosen file directly to the manufacturer via email (Figure 38).

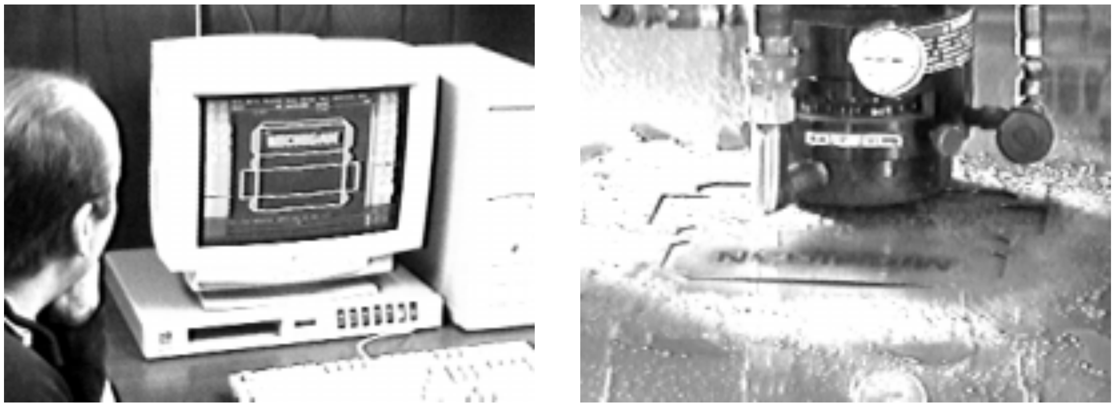


Figure 38: Manufacturer's computer & numerical controlled laser cutter

Before the model kit-of-parts component fabrications were ordered, several online experiments were conducted. The first experiments enlisted the use of design students to design objects for manufacture. The students created objects using a modeling program, created DXF files, and used a web page which communicated with the manufacturer. When they clicked on the link the plug-in was invoked. A dialog would launch which requested the designer to navigate through the local computer environment to choose the

DXF file, then would send the file as text data via email to the manufacturer. As part of the experiments, the student designers and manufacturer were kept unaware of each other, and only interacted through the web page. During the period of experiment, some fabrications for the model wall and floor panels were ordered in this manner by the author.

Eventually the experiments failed when students discovered the location of the manufacturing facility and attempted to order fabrications directly without going through the web page. (The anonymity was lost, and the purpose of the experiment, which was to affect an experimental online fabrication service, was forgotten.) The motivation for the students was to manufacture objects they created. Since the experimental fabrication plug-in was not necessarily reliable, the students became frustrated at times and sought to circumvent the system. Even though the experiments failed, the feasibility of the process was established, and fabrications for the model wall and floor panels were obtained.

5.3 Remote automated construction

In order to understand the implications of using robots for building construction, a simulation using a real robot to construct a model building using the model kit-of-parts was conducted. The simulation provided a testbed for component connection concepts, component / manipulator relationships, and robot control. The simulation tested the following assumptions:

1) Through various circumstances and influences the robot's movements may be imprecise. The design of the building components can be robust enough to accommodate such imprecision by the use of bevels, guides, and other devices.

2) Components can be mountable and demountable for reuse. No permanent constructions or installations are necessary.

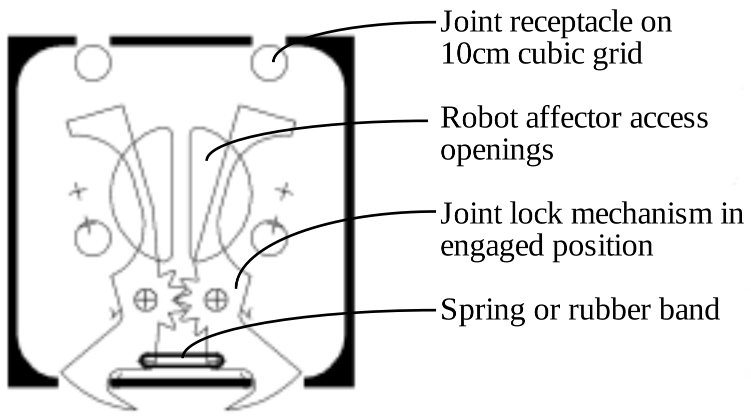
3) Components can have mechanisms or affectors built in which function as self-locking joint connections. The mechanisms can be activated and deactivated by the robot's manipulator to facilitate ease of construction or disassembly.

4) Components can be designed to have maximum flexibility in placement such that a variety of building configurations can be accommodated. Building component placement is only limited by the extents of the robot's work cell.

5) Robots can be completely autonomous in the construction sequence such that human intervention is not required. Construction sequences can be initiated locally or remotely.

5.3.1 Construction simulation

In order to test the assumptions, a model construction site simulation was conducted. The site was modeled on a table within the work cell of an RTX industrial robot. The RTX has six degrees of freedom which facilitate the placement of objects at any specified orientation, and at any specified location within the work cell. The RTX has a gripper-type manipulator with two hinged facing contact plates. The RTX can be controlled manually through teleoperation or autonomously through pre-programmed sequences consisting of an unlimited number of joint commands.


Joint receptacle on 10cm cubic grid
Robot affector access openings
Joint lock mechanism in engaged position
Spring or rubber band

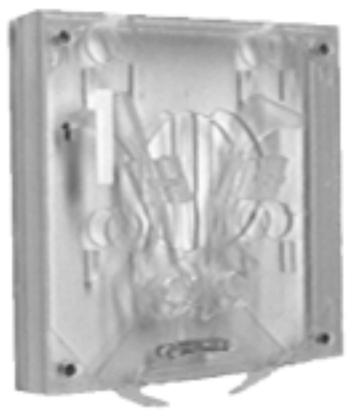


Figure 39: Model wall panel

To simulate the building, the kit-of-parts model assemblies were designed to connect to each other by means of plug-in, self-locking mechanisms which disengage through pressure from the robot's gripper contact plates, and re-engage when the pressure is released (Figure 41). The mechanisms within the assemblies were spring loaded into the locking position and acted as a "seventh" joint when coupled with the robot's manipulator. The components were manufactured from acrylic in order to facilitate ease of re-design and re-manufacture, and to allow the insides to be viewed.

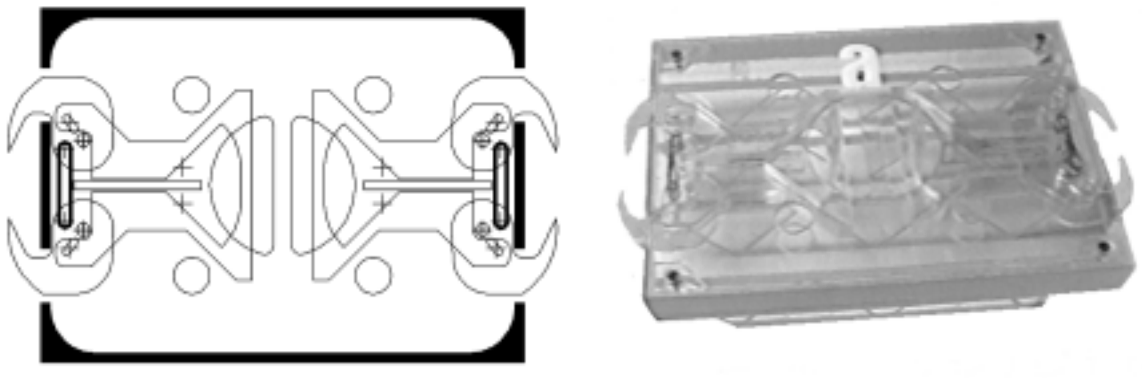


Figure 40: Model floor panel

Foundation components were not produced within the scope of this simulation. Instead, a ground plate was manufactured which had the same plug-in joint receptors as the components. The plug-in joint receptors were located in such a way that components could be plugged-in in a variety of positions and orthogonal orientations.

Control was facilitated by a small computer located adjacent to the robot, which executed either real-time teleoperated commands or pre-programmed sequences of commands. Teleoperation or pre-programmed sequence launching was facilitated by a keyboard attached to the computer locally. Pre-programmed construction sequences were also launchable over the Internet to demonstrate complete autonomy and remote control.

The simulation was conducted by first constructing a small Michigan-based model building by tele-operating the robot. Next, the building was dismantled using teleoperation. Then the building was constructed again from pre-programmed joint control sequences, without human intervention, initiated locally and dismantled in a similar manner. Finally, the fully automated construction and dismantle sequence was initiated from a remote location over the Internet from Japan and Denmark.

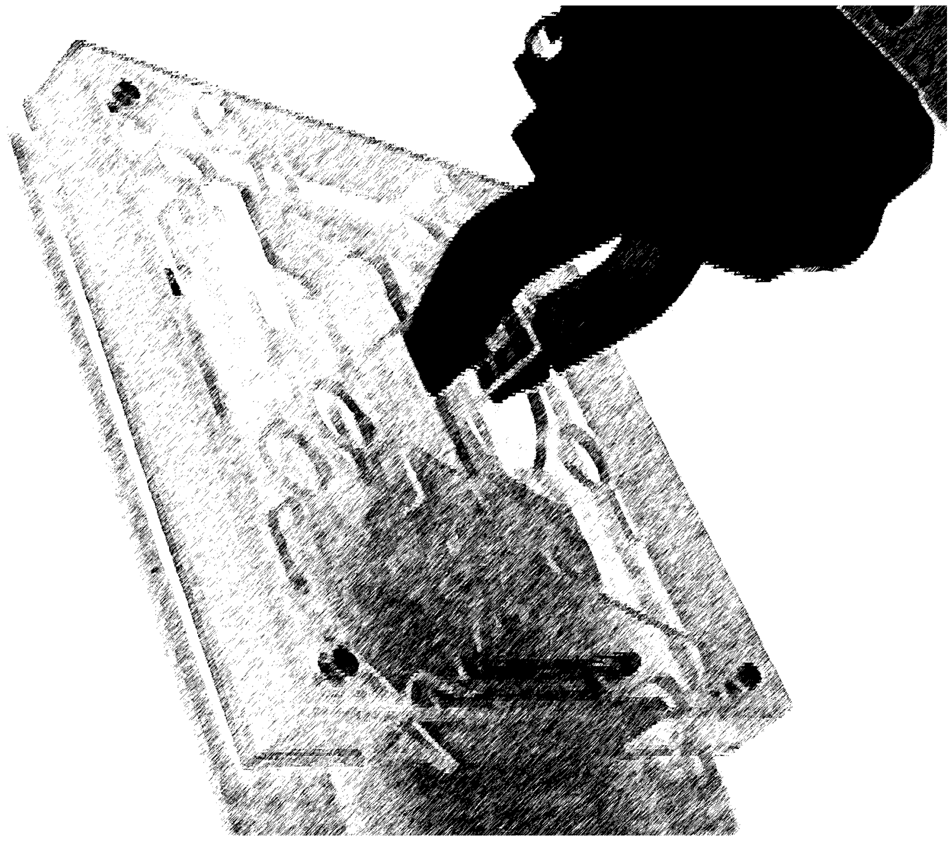


Figure 41: Robot manipulator grasping wall panel

5.3.2 Design principles

The simulation was extremely valuable in that the redesign exercises provided a set of design principles that could apply to scaled-up kit-of-parts and automated construction systems. An essential list of design principles is as follows:

> 1) Components should be designed to compensate for inaccuracies of robot position and orientation; bevels, guides, and snap-together connections are necessary for accurate assembly. All bevels and guides must be oriented in the strong axis of assembly. Natural forces such as gravity also suggest direction which can be used to the advantage of the robotic system, such as allowing a heavy component to settle downward. This principle is coined as the "strong axis principle."
>
> 2) It is advantageous to have a mounting mechanism in the building component itself, which either engages upon installation or is activated and deactivated by the robot's end affector. This principle is coined as the "seventh joint principle."
>
> 3) Construction sequences should be planned as to allow the robotic systems to work freely and have access to the site; parts which will be buried under or hidden behind other parts should be placed first while there is still access. This principle is coined the "assembly sequence principle."
>
> 4) Design of grasp points on the component, as well as design of the nature of the robot's end affector must be done in parallel. The give-and-take of the design will depend on many factors such as ease of manufacture, component appearance, and transportability. Balancing heavy components can become a problem unless the lift points have been carefully placed and designed. This principle is coined as the robot / component "interface principle."

There were other derived principles that were not seen as essential to the design, but were felt to add to optimal construction practices and material handling performance:

> 5) For the purpose of compact transportation and accessibility, the stackable storage nature of components could hold importance in many situations. In the simulation, the original design of having mechanism gripper handles located entirely within the faces of the components allowed for compact storage stackability. Having to redesign with a protruding gripper plate, the stackability

feature was necessarily sacrificed. This principle is coined as the "stackability principle."

6) Another principle relating to the "assembly sequence principle" concerns the path the robot takes from component storage position to install position. It is necessary that both the moving component or the robot do not collide with already installed components or other objects in the environment. In this simulation, paths were defined in Cartesian movements to allow plenty of room, but more efficient motions could be derived to facilitate optimum paths for speed and accuracy. These paths are mainly a construction problem, but careful thought during the design stage could improve manufacturablity. Rule-based behavioral languages for the purpose of robotic path generation have been developed to help in construction management [Stouffs et.al., 1995]. This principle is coined as the "path principle."

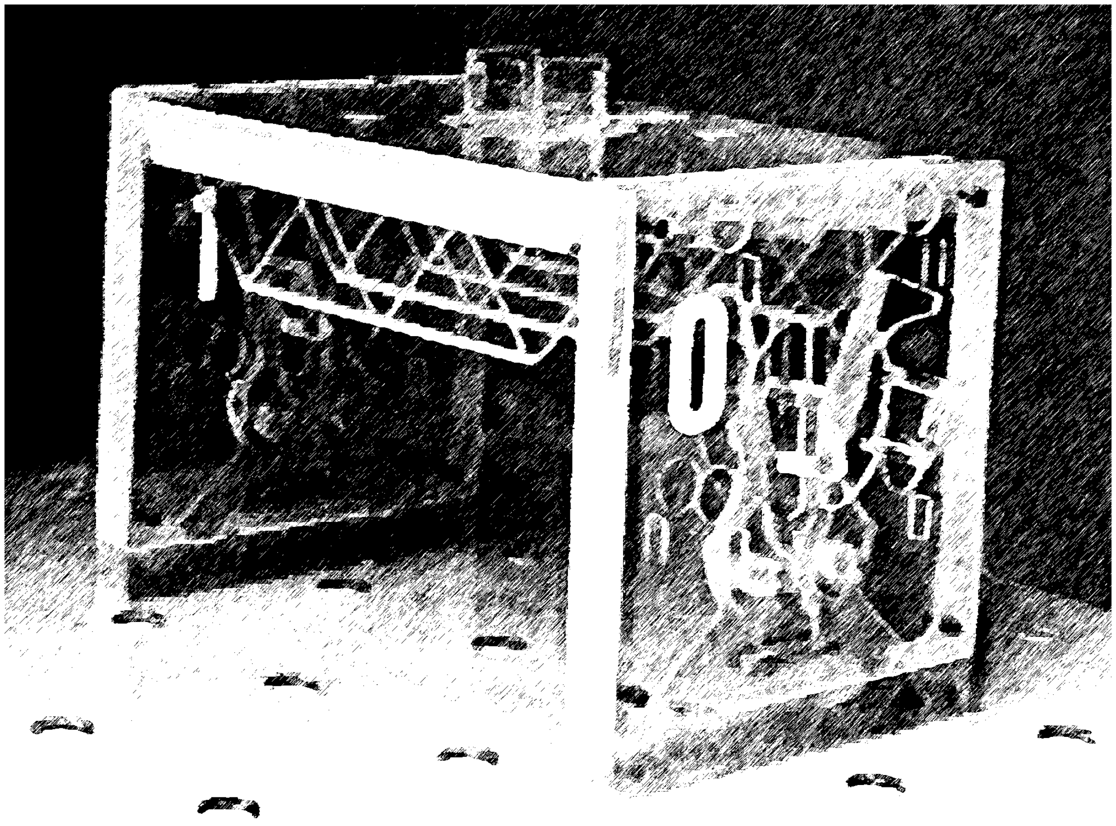


Figure 42: Assembled model building

The simulation consisted of constructing a model building using a kit-of-parts system (Figure 42). The robot used had a gripper-type end affector. How the above six design principles up-scale into the design of an actual building and robotic construction system was thought to be a critical issue. Literal up-scaling would mean having a 60 foot tall industrial robot with a huge gripper-type manipulator which would not be practical or desirable. Case studies discussed in later chapters address this problem and provide some suggestions on how to apply the design principles to actual building projects.

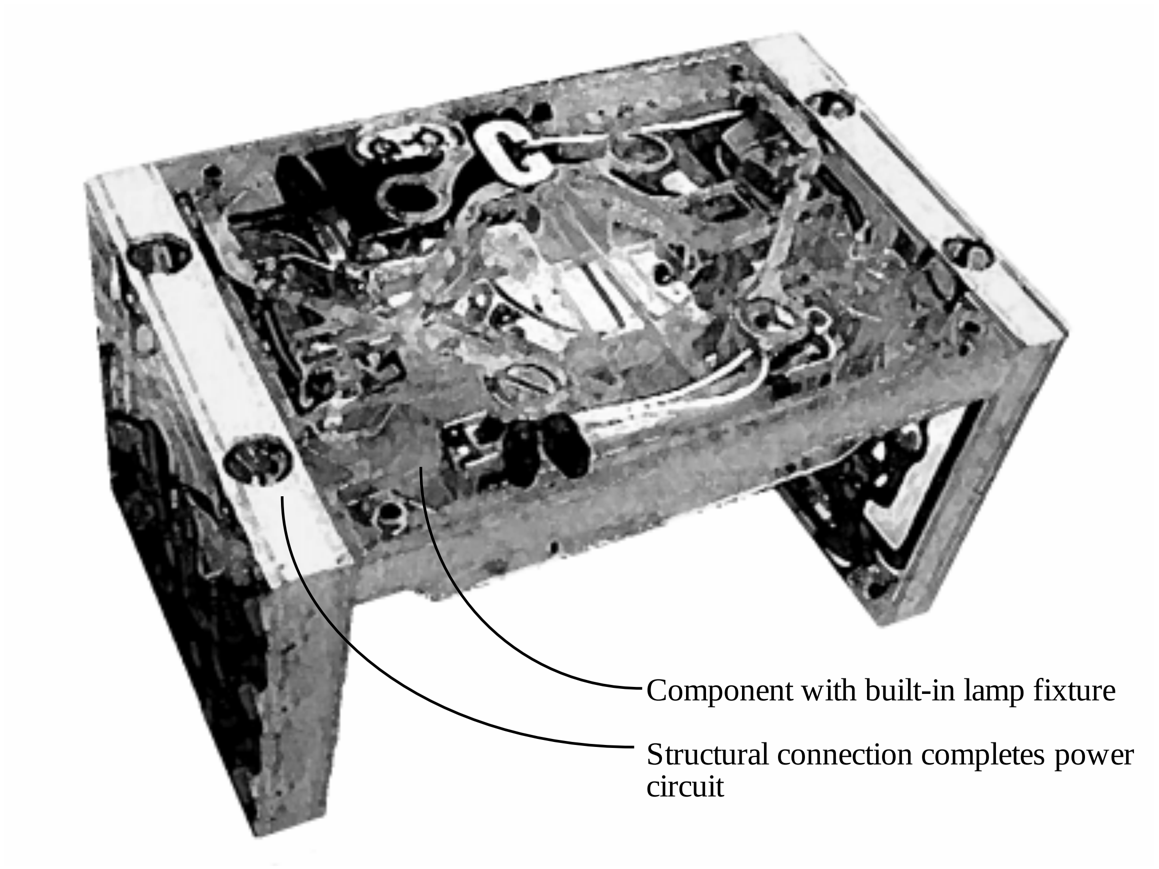

Component with built-in lamp fixture
Structural connection completes power circuit

Figure 43: Components with wiring harnesses and X10 control modules

5.4 Remote control and monitoring

In one of the eight model wall panels, and one of the four model floor panels, X-10 control microprocessors were installed. Internal power wiring and plug-in contact points incorporated into the structural joints were devised, and wiring for one additional wall panel and the floor plate was also installed. The wall panel with the microprocessor was installed with an electrical outlet, and the floor panel was installed with a built-in fluorescent lamp (Figure 43).

On Kajima Corporation's local area network, a computer with a dedicated ethernet connection and static IP address was set up as an onsite server for the model building. Next to the computer, the kit-of-parts was used to construct a model building, and a powerline interface was connected to the computer's serial port. Using a live video camera feeding real-time video to a web page accessed by the VBuild use environment, the current state of the model building was constantly broadcast to VBuild.

Using a Java Script control panel which funneled commands through an HTTP server to an X-10 control CGI, real-time monitoring and control of the model building was affected over the network. From the VBuild use environment, one could view the live video feed of the model, turn on and off the lamp with mouse clicks, and see the result in real-time. Figure 44 and Figure 45 show product model diagrams of the model building.

5.5 Future research possibilities

Future possibilities for research directions for this work is discussed in a later chapter, but within the context of the model kit-of-parts system two additional ideas presented themselves. One is the possibility of deriving a shape grammar for space generation and assembly rules using the model kit-of-parts. The other idea involves the possibility of setting up a separate tree data structure that coincides with the component joints, in order to affect automatic sensing of components.

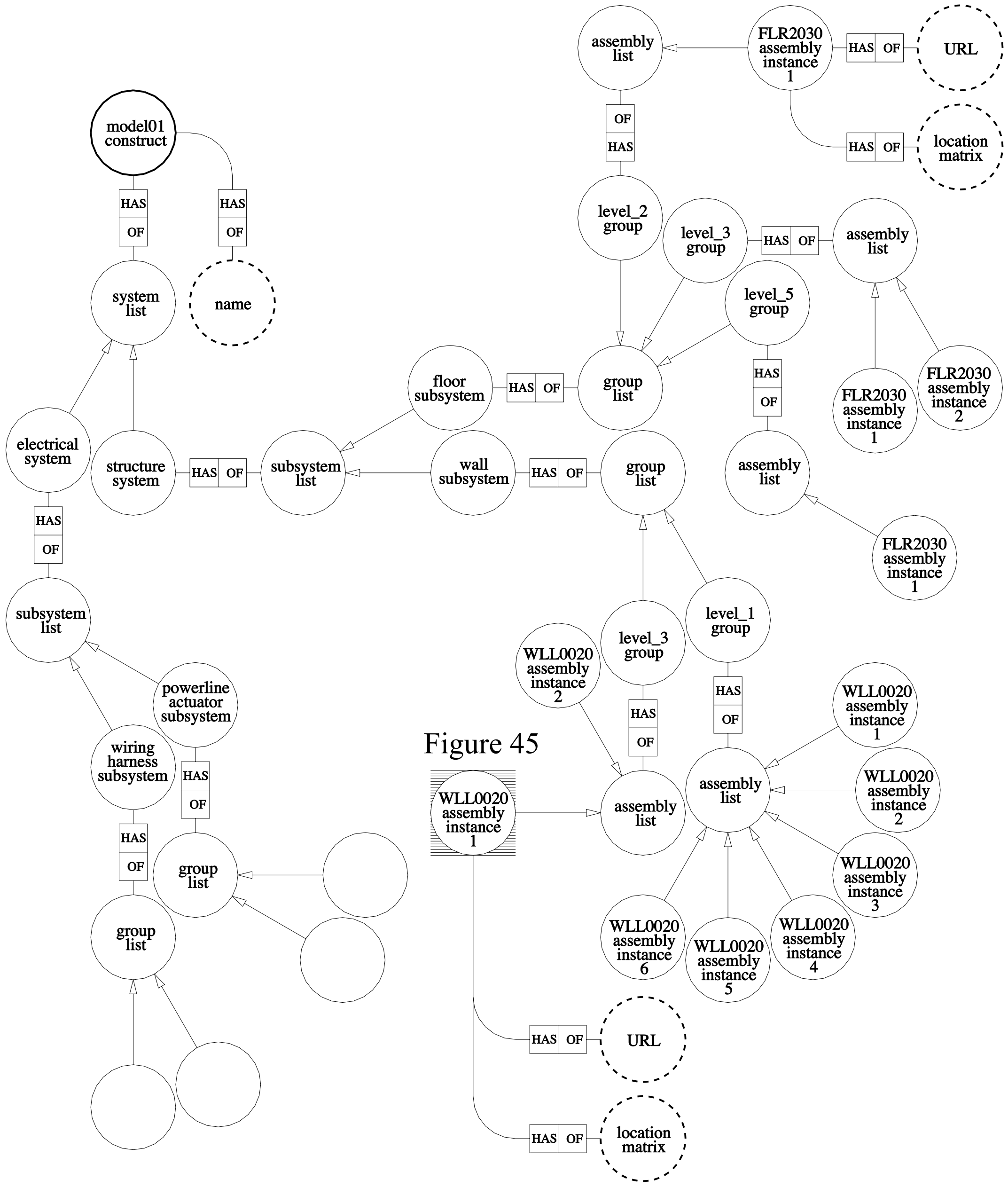

model01 construct
HAS
OF
name
system list
electrical system
structure system
subsystem list
floor subsystem
wall subsystem
group list
level_2 group
level_3 group
level_5 group
assembly list
FLR2030 assembly instance 1
FLR2030 assembly instance 2
URL
location matrix
level_1 group
WLL0020 assembly instance 1
WLL0020 assembly instance 2
WLL0020 assembly instance 3
WLL0020 assembly instance 4
WLL0020 assembly instance 5
WLL0020 assembly instance 6
Figure 45
powerline actuator subsystem
wiring harness subsystem

Figure 44: Product model diagram of model building instance

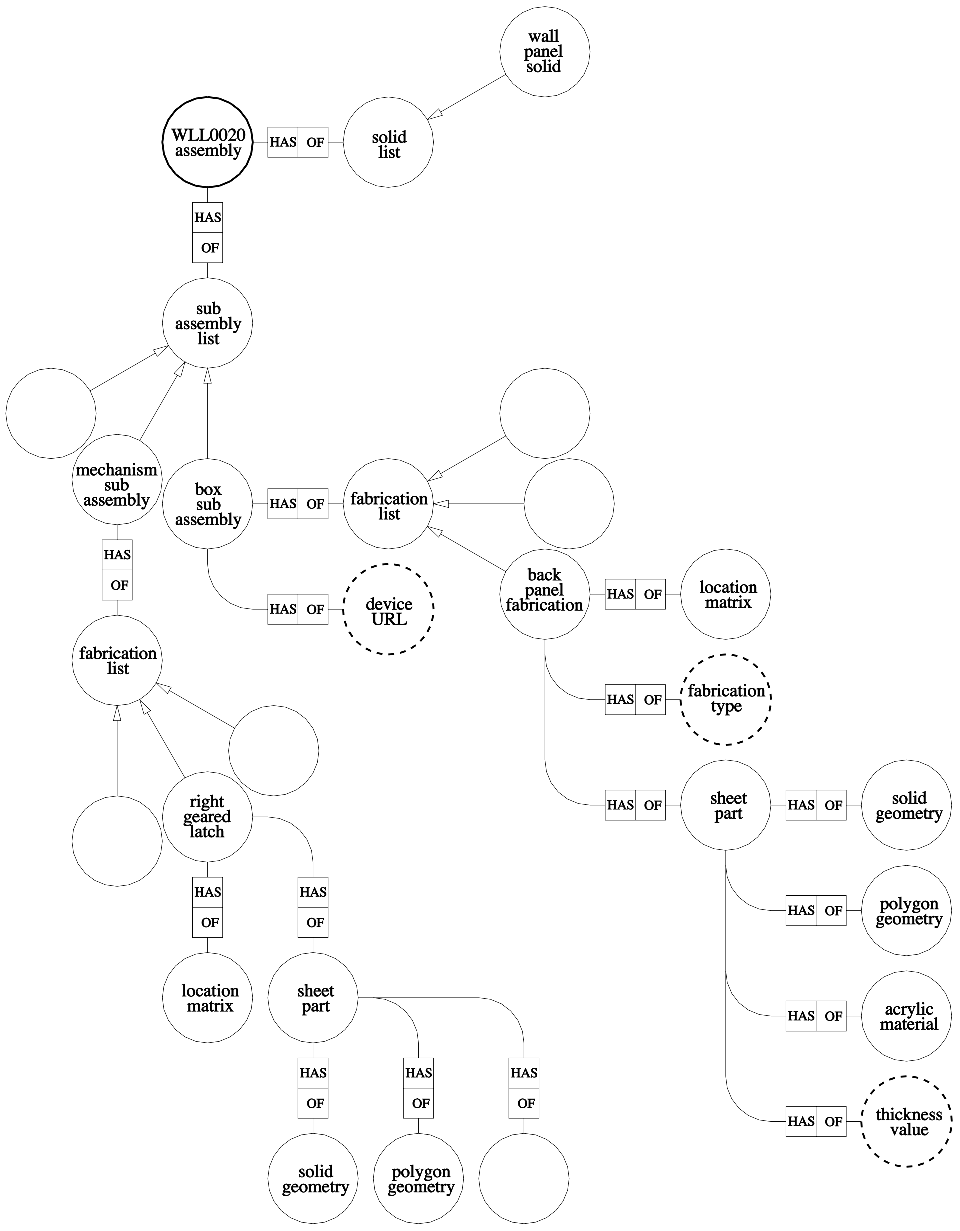


Figure 45: Product model diagram of model wall assembly instance

5.5.1 Shape grammar for space generation

The model kit-of-parts system is based on a three-dimensional cubic grid. Wall panels are two gridwidths wide and two levels high. Floor panels are two gridwidths wide and three gridwidths long. Using blocks of space that coincide with a one by one by three increment, rules for volume generation which prohibits illegal stacking of parts can be derived. Adding a third component to the kit-of-parts library (a virtual space block component), modeling in the design stage could be done using the blocks of space and checked using an expert grammar analyzer. Components could be placed automatically as generated by the locations of the blocks of space.

As for the merit of doing such research, new knowledge may be obtained regarding volume generation in design, within a kit-of-parts context. This concept is discussed in a later chapter in connection with the Emerald Soft Factory design (Life Cycle Architecture System), where shape grammars have been implemented in this way.

5.5.2 Automatic component sensing

Since this model kit-of-parts building system is very simple and contains only two types of components, a data structure which represents relationships can easily be developed. Using a trip line sensor to tell whether a joint has a component plugged in or not, binary 1's and 0's can be loaded into a tree which represents the component relationships in the entire model building. Using a simple system of rules that always match pairs of plugged-in joints, an algorithm can be derived that will build a virtual building automatically from a physical one.

Having the ability to go in the reverse direction from physical to virtual may have many advantages. First, automated assembly of physical components using robots can be checked against a completed virtual model. As soon as a component is plugged-in, the trip sensor could register a 1 in each of the joint pairs which will cause the system to create a new component instance in the virtual construction site. Second, the model being constructed could be viewed as a virtual construction site, and the progress of the construction verified in real time over a network.

The contributions such research could provide would be invaluable. Real-time feedback of component connections in a building could loop back and influence the design of more efficient connections, and help with the analysis of the construction process.

5.6 Summary

For the purpose of demonstrating feasibility of life-cycle management of kit-of-parts building systems, a building was simulated using a model kit-of-parts system. The various stages of design, component fabrication, building construction, and building occupancy were simulated using real processes and technologies. During the simulations there were many successes and failures, and potential for future research was highlighted. In spite of the failures, and due to the successes, much valuable knowledge was gained regarding design, fabrication and construction, and use of kit-of-parts building systems. The value of an overall virtual building structure and life-cycle application such as VBuild was established.

CHAPTER 6

CASE STUDY: LIFE CYCLE ARCHITECTURE SYSTEM

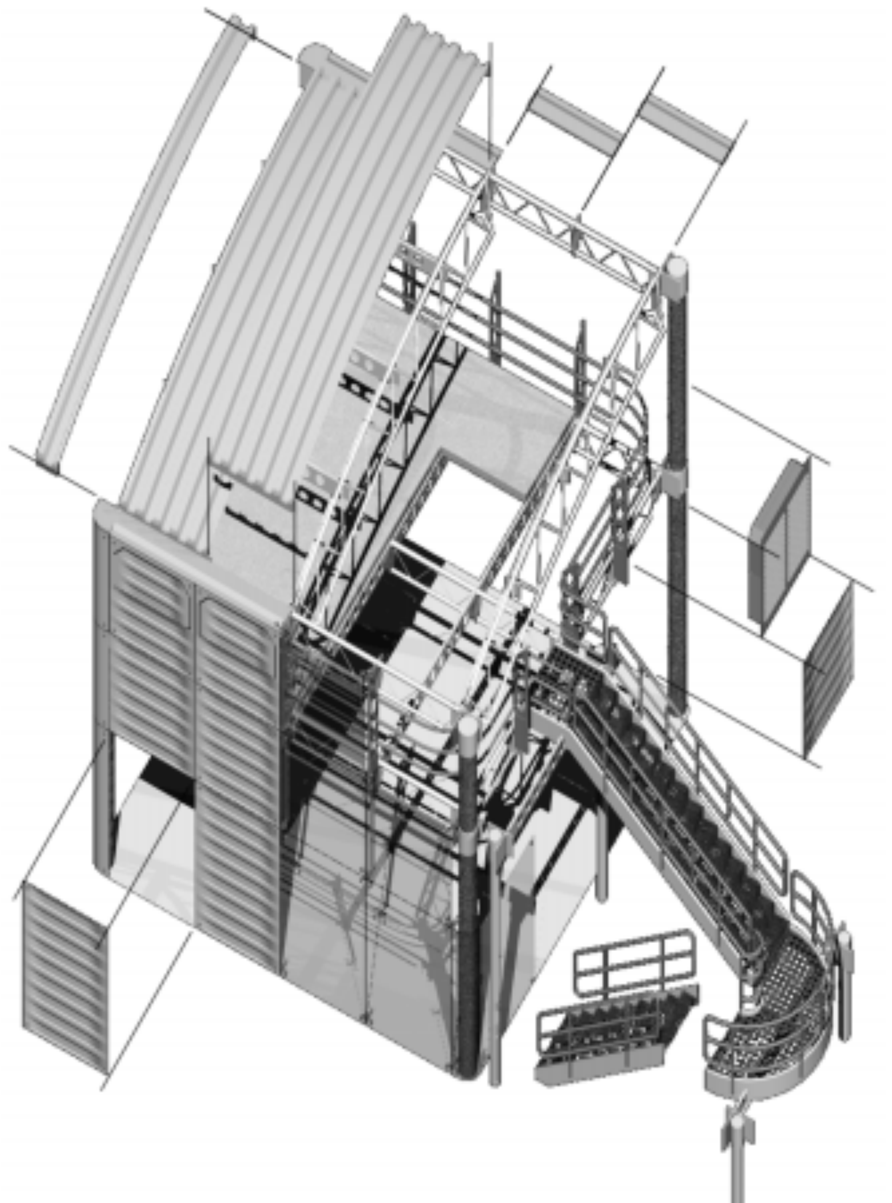


Figure 46: Yukigen kit-of-parts

Three variations of a joint-based kit-of-parts concept demonstrate possible applications of this work to real projects. The three variations represent work completed by the author in sequence, each concept improving or expanding upon the previous, the final being the most refined. In each case the work proceeded with the assumption that a life-cycle management system such as VBuild browser would be used. The projects are LDS Building System (1994), Yukigen project (1995), and Emerald soft factory (1997).

6.1 LDS Building System

The LDS Building System concept was presented to The Church of Jesus Christ of Latter-day Saints (LDS or Mormon) as a potential building solution for their Physical Facilities department. The Mormon church population exceeds ten million members world wide and in the last few years has had an increase of a million or so members per year. The growth explosion has put strains on their building program, which must provide five hundred to a thousand new buildings each year in countries all over the world, under a myriad of building codes. Each of these buildings acts as a community center for a specific geographical area in which a local congregation resides. These congregations are officially called “wards” or “branches”, but generically can be refered to as “local units.” In the context of this project, design studies were conducted for a low-cost, high-profile design, kit-of-parts architectural package for assembly of chapels, office / institutional buildings, and residential projects (Figure 47) on remote sites all over the world.

In economically advanced countries, work is generally commissioned by local architects and bid out according to local laws. However, in remote areas of developing countries the construction of chapels and schools are often undertaken by the users themselves, who are usually unskilled and without modern tools. The LDS building system was conceived as a universal kit-of-parts that could have incarnations in both of these extremes. In its simplest form the building parts can be assembled or unassembled by hand without tools, and can be arranged into any configuration on any site. Since power and communication lines can be integrated into the joint, they automatically connect when the joints are structurally secured. In more sophisticated uses, an automated construction system prepared for the kit-of-parts includes three construction robots which fold up together into a shipping container that can be delivered to remote flat sites and assemble the building system automatically for most configurations.

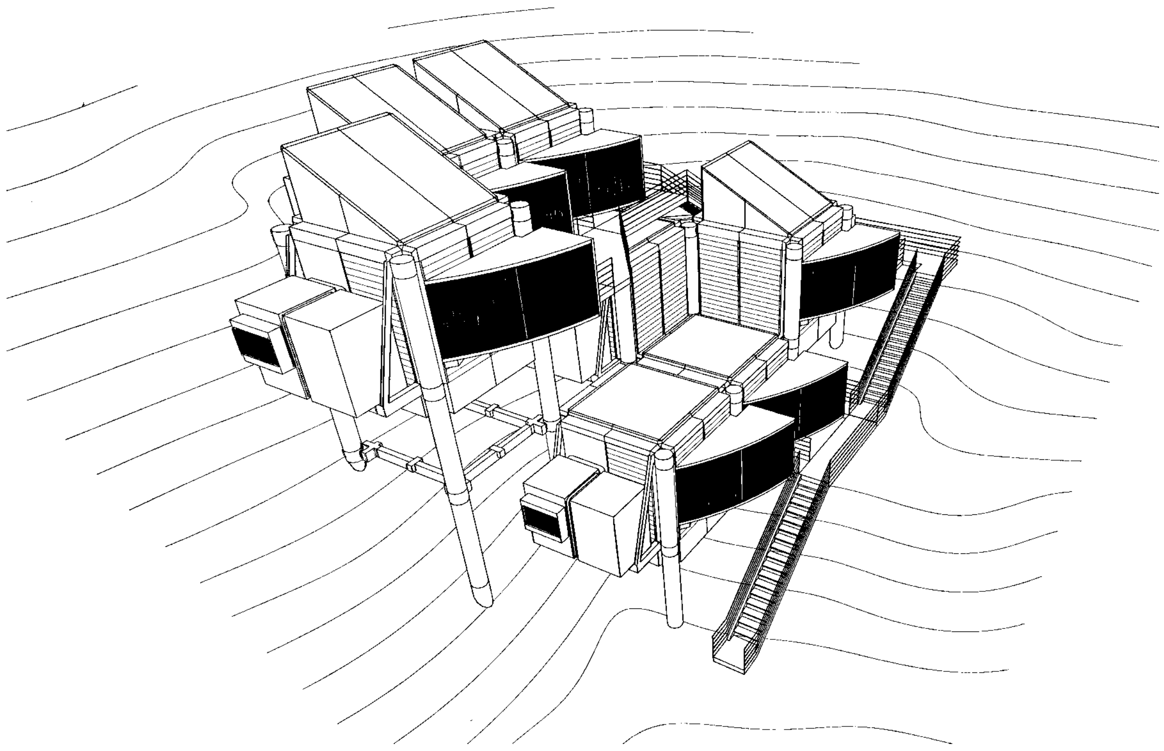


Figure 47: LDS Building System residential complex

6.1.1 Concept

In contrast to conventional prefab systems which generally are applied to a single plan with little or no variations, the kit-of-parts system is conceived as a library of components with maximum flexibility for configuration. The components are easily assembled and meant to be demountable for reuse over and over again. Considering the systematic approach adopted by the toy "LEGO", where no specific final product is suggested but only a library of parts and rules for their assembly provided the designer, the concept of the kit-of-parts building system should become clear.

The kit-of-parts system is joint-based, which means that a rigorous system of standard interfaces between parts is strictly observed, but the actual members themselves can be up to designers. This could facilitate the use of different materials or the creation of new parts that fit into the system. The joint system would be conceived in such a way that the possibility of incorporating power and communication infrastructures into the parts could be facilitated. This means that structural connection would also automatically complete wiring of the building since the "wiring harnesses" would be integrated into each part. The joint system would also have the capability to be "snap together / snap apart" without the use of tools, for the benefit of remote locations where tools and mechanical skills are hard to come by.

It is conceived that a library of standard parts could be available through Church distribution networks which could be ordered as needed by individual local units. Standard plans could be established for certain sized local units, which would be based upon parts from the catalog. Since a component-based system only consists of discreet parts, the number of each part required in a standard plan would be known, and the packageability of the components for shipping to remote sites could be worked out. Nevertheless since the parts are demountable, expansion or contraction of the building could be easily accomplished depending on the needs of each local unit, and required parts ordered singly.

6.1.2 Design grammars

Within the context of the LDS Building System, design grammars and rules were established for space planning and component placement. The design grammar has a two-fold purpose: to prescribe the way the structure should go together to support hierarchies of space, and to exist as guidelines for automated construction system design.

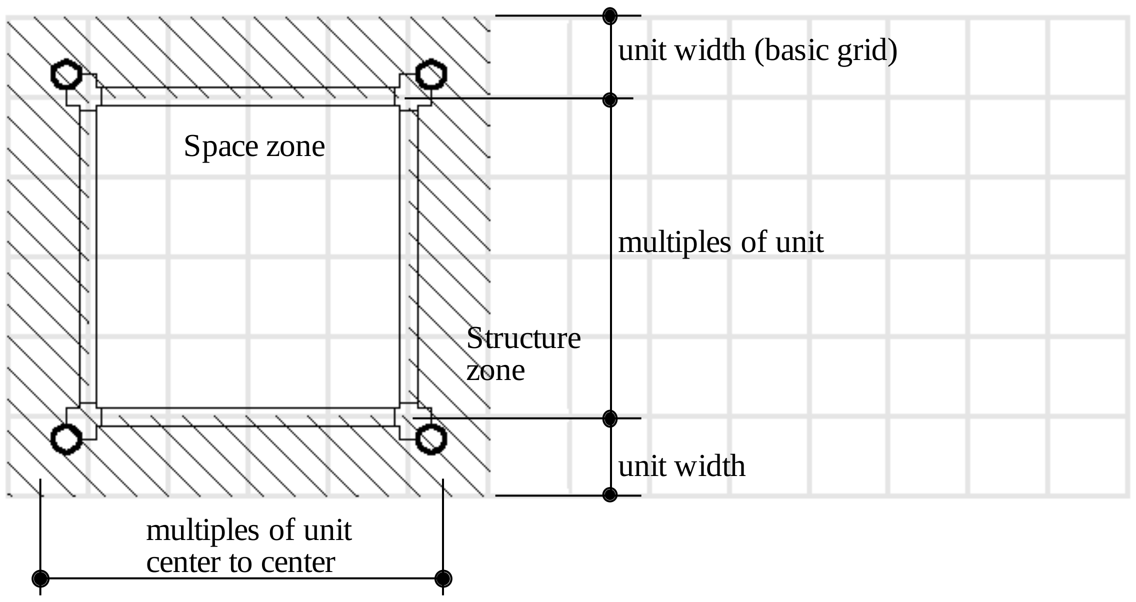


Figure 48: Space zone and structure zone

Architecture is essentially a collection of function-specific spaces. The structure and envelope conceived in the design process is for the purpose of containing those spaces. In this kit-of-parts system, the process for assembly of the structure has been given priority as well, and the concept behind the containment of spaces is therefore influenced. Figure 48 is a diagram of a single structural bay, showing space zones and structure zones. Two overlapping grids are utilized: the basic grid and structural grid. The basic grid is based on economy of material, which is 0.9m in Japan and four feet in United States.

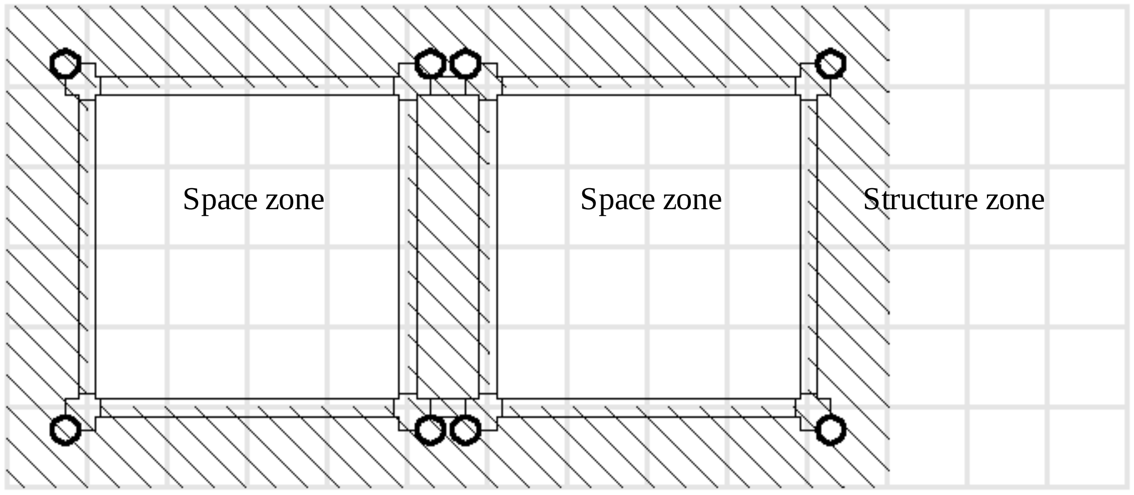


Figure 49: Overlapping structure zones

The structural grid is derived from the basic grid. The structural grid consists of zones which are multiples of the basic grid in width, and define space zones which are also multiples of the basic grid. When two bays are put together, the adjacent structure zones overlap, but the actual structure may not necessarily do so.

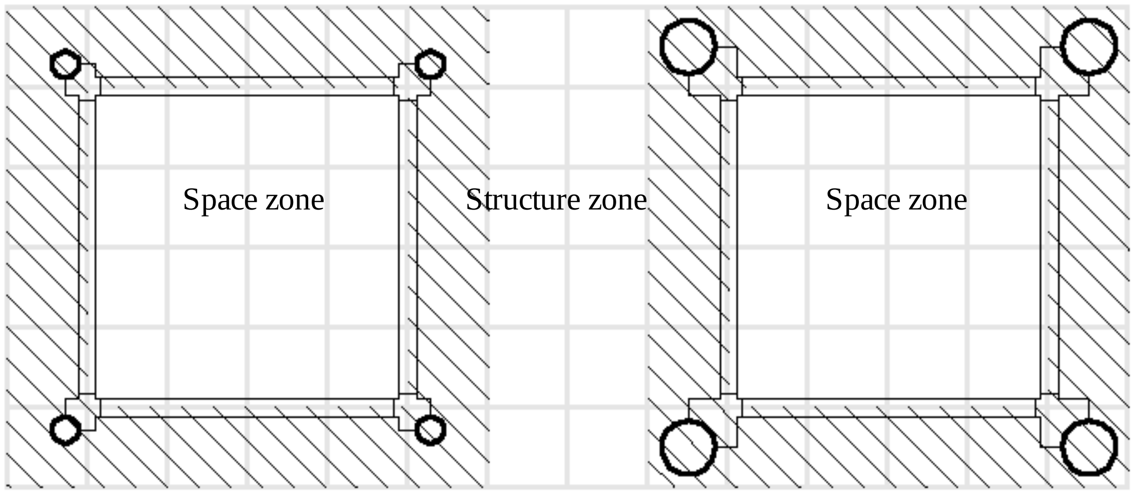


Figure 50: Different-sized columns

In Figure 49 the structures of two bays are shown completely independent of each other. By keeping the structures independent, expansion joints, passive seismic connections, etcetera can be facilitated.

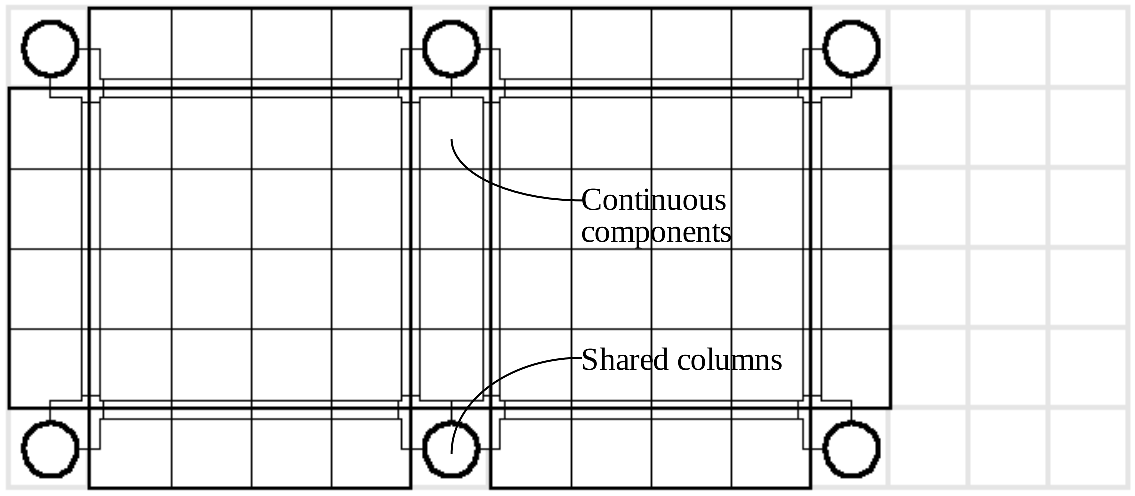


Figure 51: Common columns and continuous spaces

The corners of the space zones are protected and supported by eccentric joints that allow the centerlines of girders and trusses to lie exactly on the boundary between the two zones. Columns are placed offset from this centerline depending on the diameter of the column. Using this rule, within the structure zone the size of columns may change without effecting the space zone or size of local structure members (Figure 50).

The design grammar allows each bay to be completely independent or share structure. Two bays may share a column within the structure zone (Figure 51) or each have their own independent column.

The design grammar also assumes the existence of four basic types of space: user space, service space, circulation space, and exterior space. A simple application of the four spaces can be seen in Figure 52.

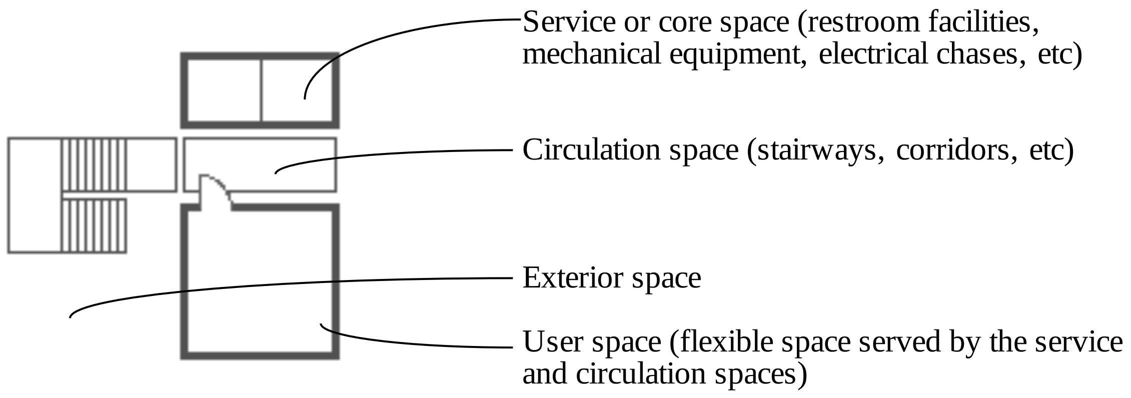

Service or core space (restroom facilities, mechanical equipment, electrical chases, etc)
Circulation space (stairways, corridors, etc)
Exterior space
User space (flexible space served by the service and circulation spaces)

Figure 52: Four types of spaces

The systematization of the four types of space according to the design grammar allows for dynamic space planning in three dimensions. Where a structural bay will always enclose space, the space may be exterior as well as one of the three interior types.

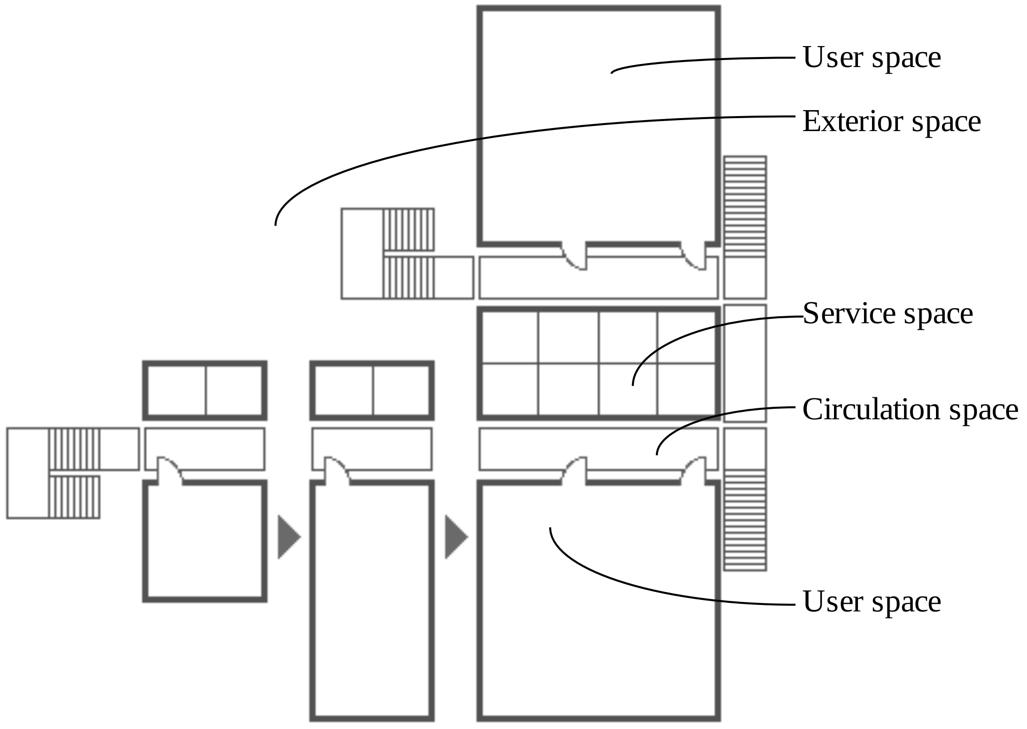


Figure 53: Dynamic space planning

This can create dynamic open volumes when there is mixed stacking of interior and exterior space. The design rules also encourage the establishment of hierarchical spaces serviced by a circulation spine (Figure 53).

6.1.3 The kit-of-parts system

Within the context of the design grammar, a hybrid joint-based and module-based kit-of-parts system was conceived. The joint-based cladding and structural systems were designed to plug into the same standard joint, which was designed for the worse possible case within the context of the design rules (also influenced by the structural strength of each member). The joints were design to take a wiring harness which established connection when structurally secured.

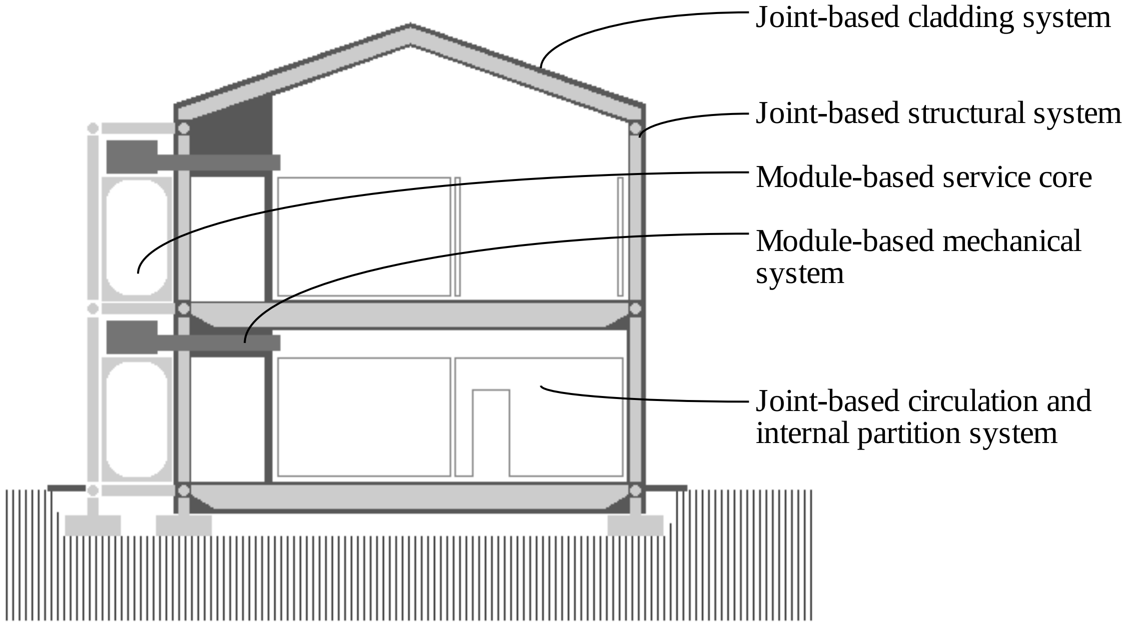

Joint-based cladding system
Joint-based structural system
Module-based service core
Module-based mechanical system
Joint-based circulation and internal partition system

Figure 54: Hybrid joint-based and module-based kit-of-parts

Core modules are pre-manufactured special purpose rooms which are fully self-contained for the specified function. The modules are weatherproof and zip together and to the main structure with rubber gaskets. All plumbing, communications, computer equipment and such are located in the modules. The modules can be fully stocked with necessary equipment at the time of manufacture, shipped to the site, and plugged into place. Plumbing and power connections would have standard interfaces to ease mounting and demounting. Standard simple modules could be arranged to form more complex spaces such as restrooms or kitchens.

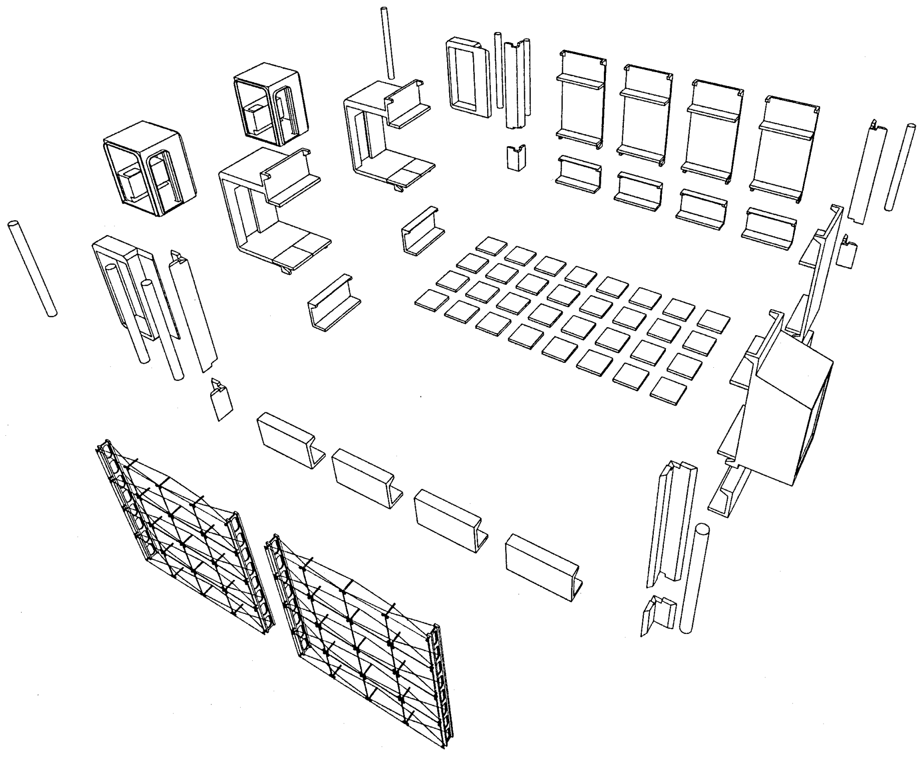


Figure 55: Joint-based kit-of-parts

In the context of the Church, an example of a core module could be a library module which would be fully stocked at the time of manufacture with all the materials needed for Sunday School teaching aids. Another example would be a kitchen module fully stocked with pots, pans, and other utensils. The clerk's office would be another example, fully equipped with computer, telephone, and paper stock.

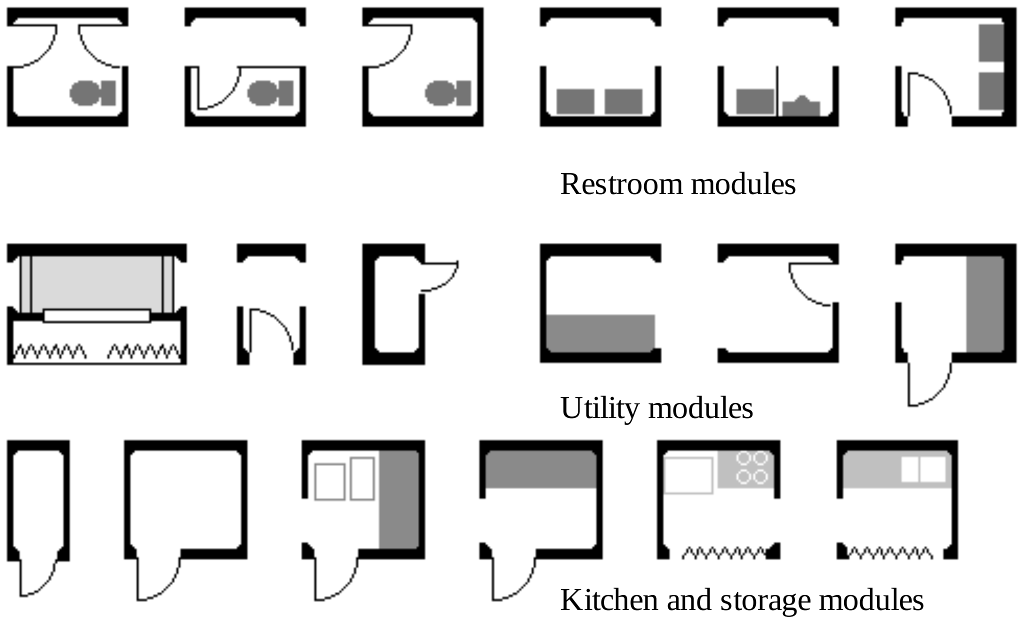


Figure 56: Zip-together core modules

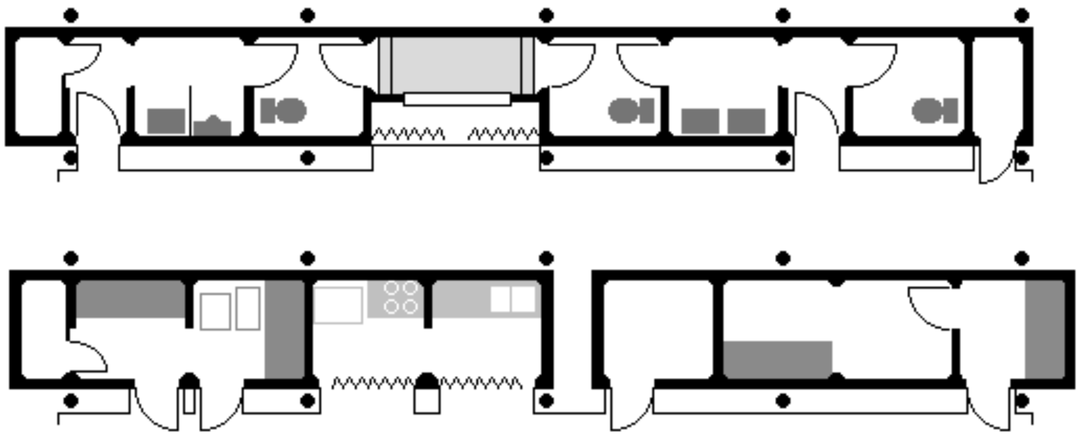


Figure 57: Example service space cores

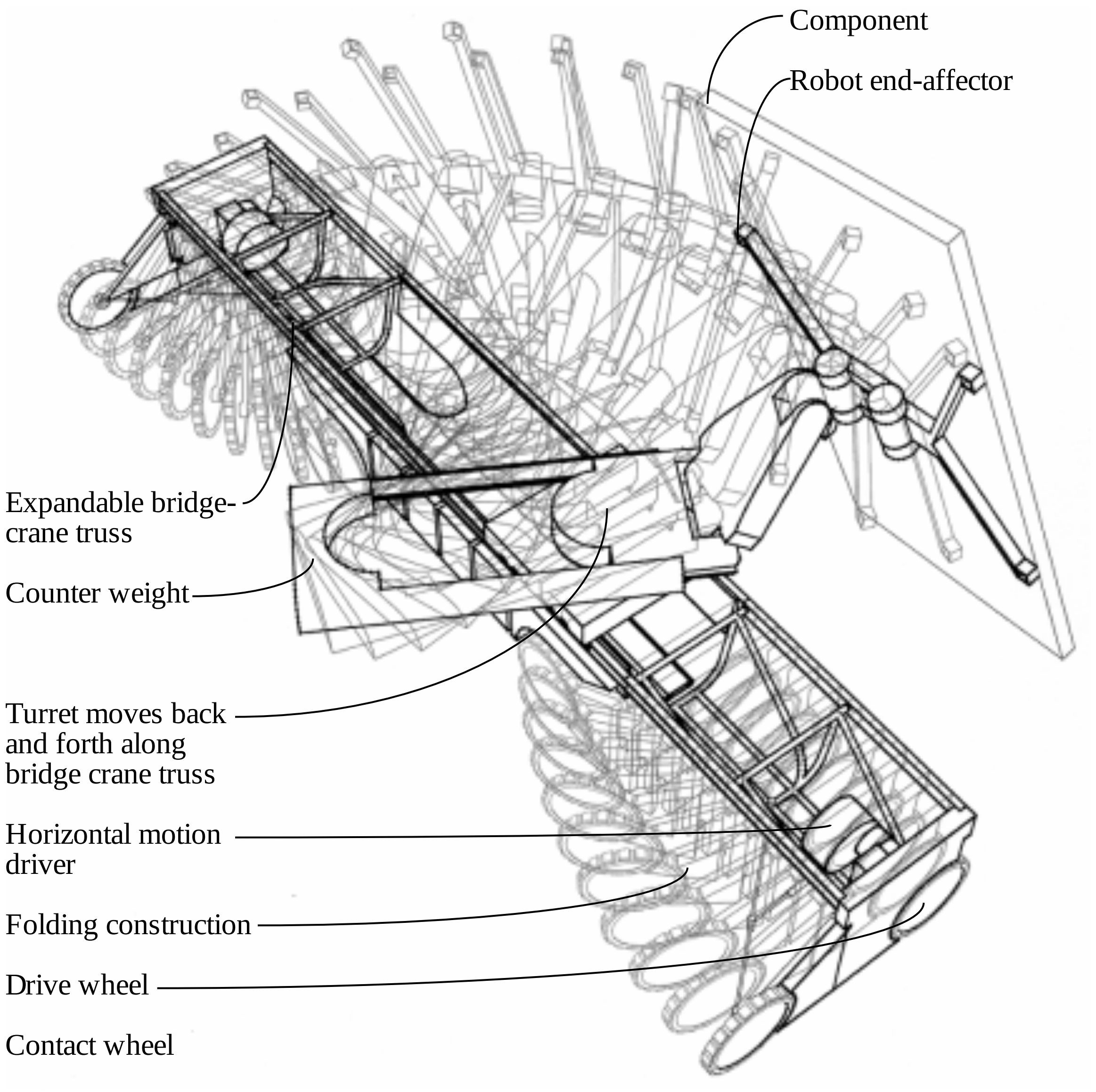


Figure 58: Bridge-crane robot

6.1.4 Automated construction

In the current research programme, the structures of each bay may be kept independent for another reason: automated construction. A set of three construction robots have been devised and control programming developed for the purpose of constructing a single bay up to 8.1 meters wide (center to center of structure zone). Using the automated construction concept, the building would go up a single bay at a time to an indefinite height (limited by the pre-engineered specifications of the members), by building each floor at the bottom and jacking up the entire building a floor at a time using a version of Kajima's AMURAD technology [Sekiguchi, Honma, Mizutani, and Takagi 1997].

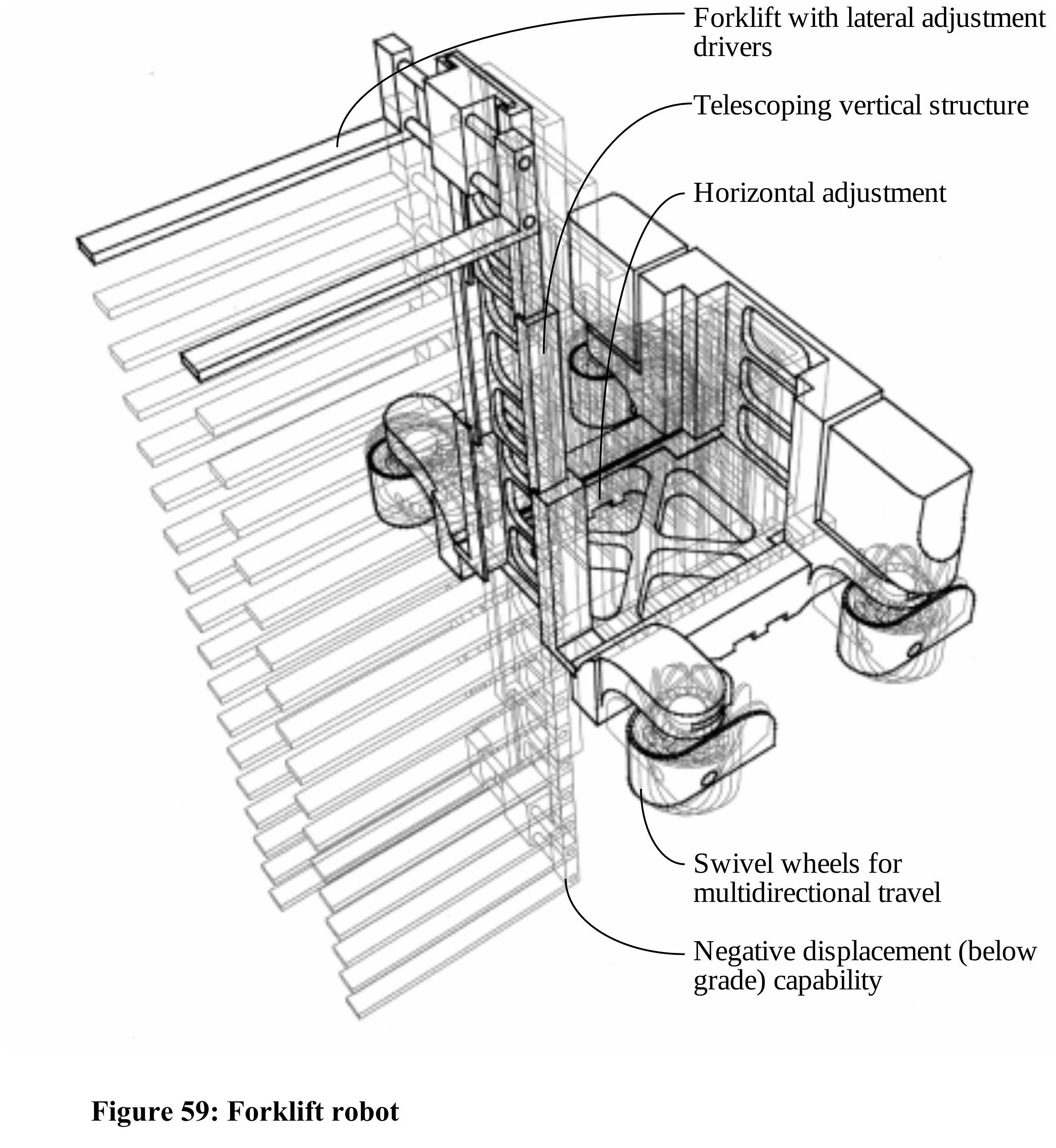

Forklift with lateral adjustment drivers
Telescoping vertical structure
Horizontal adjustment
Swivel wheels for multidirectional travel
Negative displacement (below grade) capability

Figure 59: Forklift robot

Using the same design grammar around which the kit-of-parts system was designed, the structural bay defined by space zones and structure zones becomes the work cell of the robotic system. The shape grammars specify maximum and minimum bay widths but not bay depth. For this reason a robotic system is required which can expand or contract the width of its work cell according to the bay width, but extend its depth an unspecified distance.

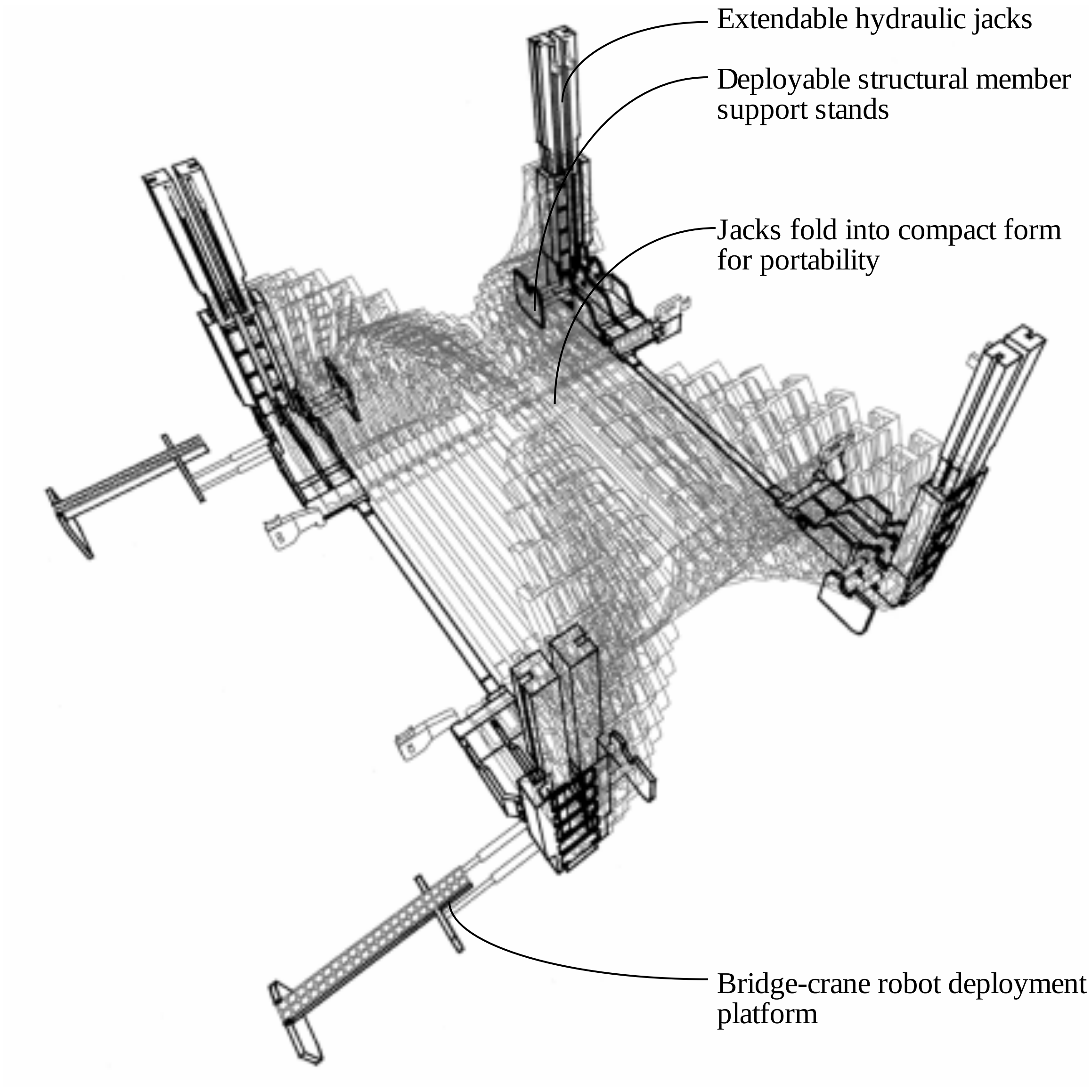

Extendable hydraulic jacks
Deployable structural member support stands
Jacks fold into compact form for portability
Bridge-crane robot deployment platform

Figure 60: Hydraulic jack robot

In the context of this example, a system of three robots was designed. One robot was a mobile autonomous forklift for carrying component pallets and materials (Figure 59). Another robot was a bridge-crane-like robot (Figure 58) with a special six-jointed robot attached for component installation. The third robot was a set of four hydraulic jacks for lifting the already constructed portion of the building (Figure 60). The three robots fold together into each other and are carried in the storage position by the forklift robot.

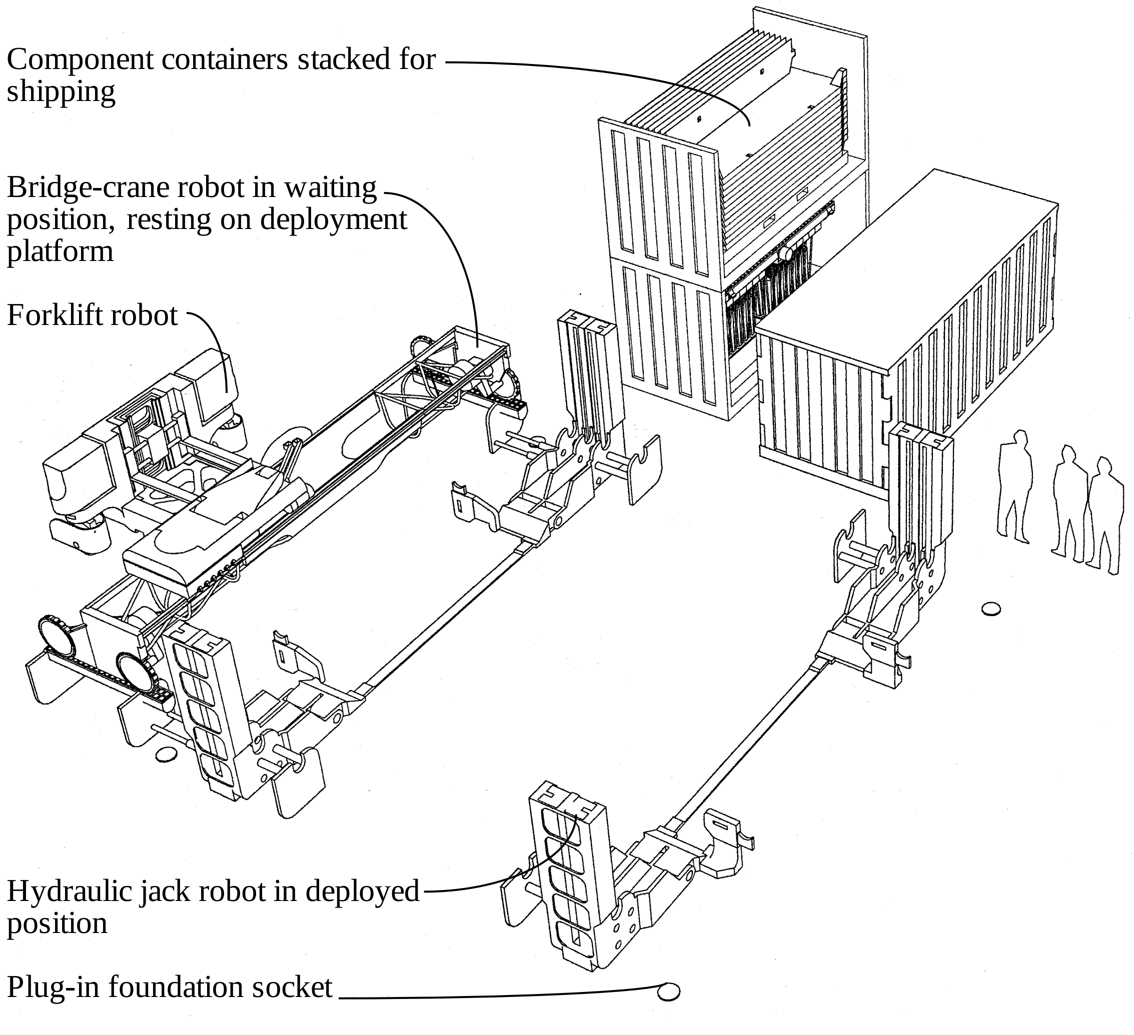

Component containers stacked for shipping
Bridge-crane robot in waiting position, resting on deployment platform
Forklift robot
Hydraulic jack robot in deployed position
Plug-in foundation socket

Figure 61: Robotic construction system deployment

In the construction sequence, the forklift would deploy the other two robots over a proposed bay location (Figure 61). The bridge crane robot would be waiting on a launch platform, the jacks would be ready to support the first-assembled structure, and the forklift would retrieve the first parts pallet. Next the bridge crane would begin taking girders from the forklift stack and laying them in a direction parallel with the bay depth. It would use the girders as rails to move up and down the depth of the bay, forklift following (Figure 62).

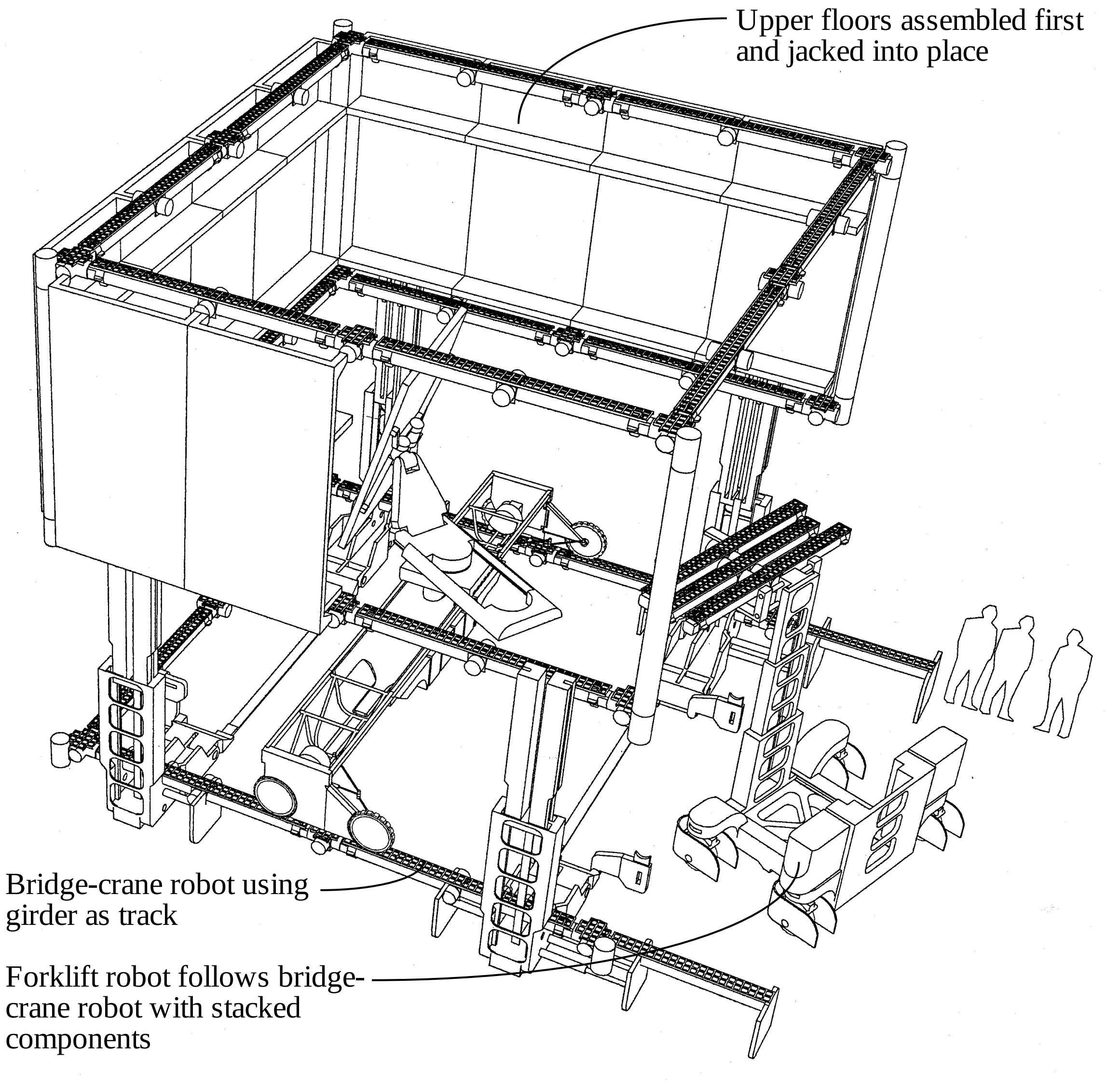

Upper floors assembled first and jacked into place
Bridge-crane robot using girder as track
Forklift robot follows bridge-crane robot with stacked components

Figure 62: Automated construction system

In this manner an unspecified depth of a bay may be assembled, with the entire bay width fully accessible by the bridge crane robot. When a floor is complete, the bridge crane would move out of the way onto the launch platform and the jacks would lift the floor overhead and allow the assembly of the next floor. This sequence would continue until the proper number of floors for the bay was constructed, whereupon the set of three robots would pack up and move over to the next proposed bay. This example automated building system has been tested and simulated on a computer using a kinematics and robotics software.

6.1.5 Chapel example

Using the kit-of-parts, a series of chapels were considered as standard designs for the kit-of-parts library. The chapels are for use in the “block” schedule of The Church of Jesus Christ of Latter-day Saints. The block schedule is three back-to-back one-hour time frames which enclose certain activities. The three slots can be mixed around according to the local unit’s needs. During one of the time slots, all the members of the unit are in attendance in an assembly hall called the chapel. During this time no other meeting rooms are needed. During the other two hours however, several classrooms are needed (see Table 2 and Table 3).

Four standard plans were developed during the design exercises. Since the number of configurations is virtually limitless, only one example is described in this work. The following plans are based on a ward of 120 members. Nevertheless since these are conceived based on parts from a component library, any ward building or stake center of any size could be configured on any reasonable site (sloped or flat).

Based on these numbers, chapel, offices, and classrooms have been laid out in various configurations. The configurations represent various shaped sites. The Plan A chapel was designed for a very small site in medium to high density areas. The total floor area is 300m^2, with 130m^2 on the first floor and 170m^2 on the second (the second floor overhangs the first, which space could be used for parking on tight sites). The seating capacity of the chapel is 114, which could be expanded to 135. There are only two offices, a clerk and bishop's office for the use of only one unit. Library, kitchen, and most classrooms are located on the first floor, with restrooms and chapel located on the second floor (see Figure 63 and Figure 64).

Class	Persons	Classrooms
Gospel Doctrine	43	1
Gospel Essentials	13	1
Family Relations / Family History	10	1
Youth 12-13	10	1
Youth 14-15	9	1
Youth 16-18	9	1
Primary Sharing	8	1
Sun Beams		
Stars		
C.T.R.	3	1
Valiants	4	1
Merry Miss	4	1
Blazers	3	1
Nursery	4	1
Teacher Development	(5)	1
Total	120	13

Table 2: Block schedule usage: Sunday School hour

Class	Persons	Classrooms
Priesthood	43	
High Priests	7	1
Elders / Prospective Elders	22	1
Priests	5	1
Teachers	4	1
Deacons	5	1
Relief Society	37	1
Young Women	14	
Laurels	4	1
Mia Maids	5	1
Bee Hives	5	1
Primary Sharing	14	1
C.T.R.		
Valiants		
Merry Miss		
Blazers		
Sun Beams	4	1
Stars	4	1
Nursery	4	1
Total	120	13

Table 3: Block schedule usage: Priesthood / Relief Society hour

Since the chapel would not be needed during Sunday School and Priesthood / Relief Society hours, it can be divided into classrooms during those times. Including the pulpit area, thirteen teaching stations can be accommodated. However, since the "Primary" group uses two rooms during the Priesthood / Relief Society hour, one teaching station cames up short, requiring the combining of two age groups (Table 4).

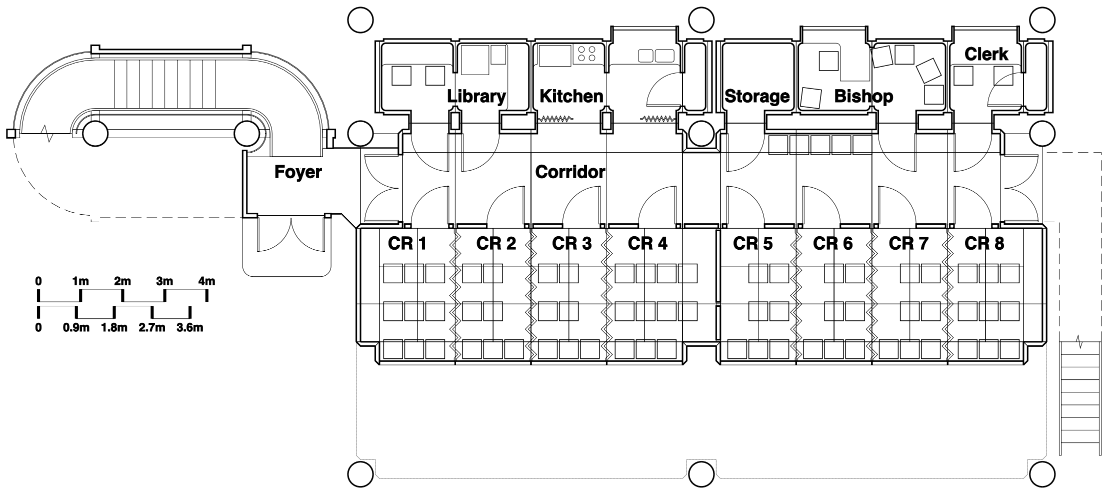


Figure 63: Plan A first floor plan

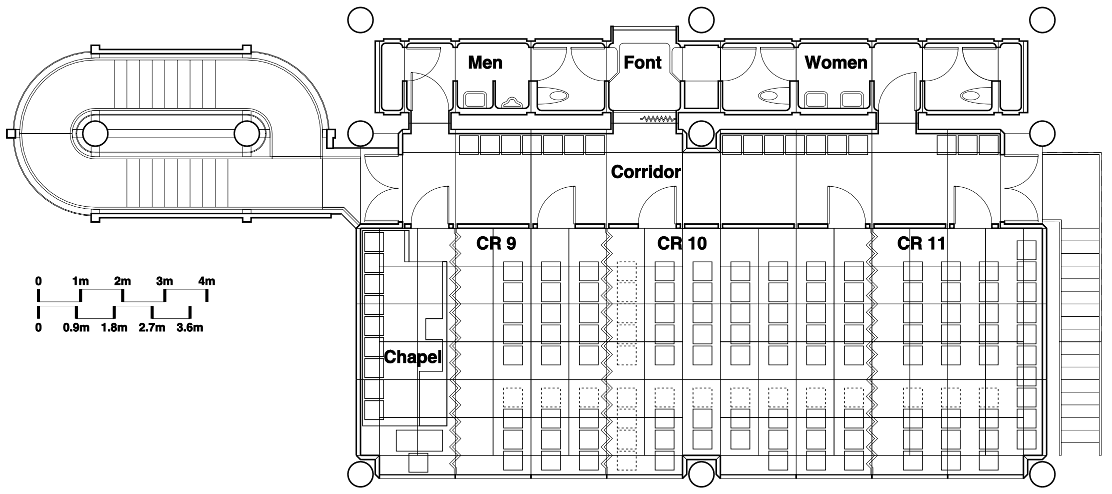


Figure 64: Plan A second floor plan

Sunday School Hour			Priesthood / Relief Society Hour		
Class	Capacity	Rm #	Class	Capacity	Rm #
Gospel Doctrine	43	CR 10	High Priests	24	CR 9
Gospel Essentials	34	CR 11	Elders	43	CR 10
Family History	24	CR 9	Priests	5	Bishop
			Teachers	9	CR 8
Youth 16-18	9	Pulpit	Deacons	9	Pulpit
Youth 14-15	9	CR 8			
Youth 12-13	11	CR 4	Relief Society	34	CR 11
			Laurels / M. Maids	11	CR 4
			Bee Hives	7	CR 3
Primary Sharing	9	CR 1	Sun Beams	7	CR 5
Sun Beams			Stars	7	CR 6
Stars			Primary Sharing	16	CR 1, CR 2
C.T.R.	7	CR 2	C.T.R.		
Valiants	7	CR 3	Valiants		
Merry Miss	7	CR 6	Merry Miss		
Blazers	7	CR 5	Blazers		
Nursery	7	CR 7	Nursery	7	CR 7

Table 4: Plan A classroom usage

6.1.6 Residential example

Using the same LDS Building System kit-of-parts, a residential complex was designed to prove that the system could accommodate sloped sites. Though the robotic building system could not be used on non-flat sites, assembly could be done using conventional methods. The complex consisted of seven units layed out in a stepping fashion on the side of a hill (see Figure 65 and Figure 66).

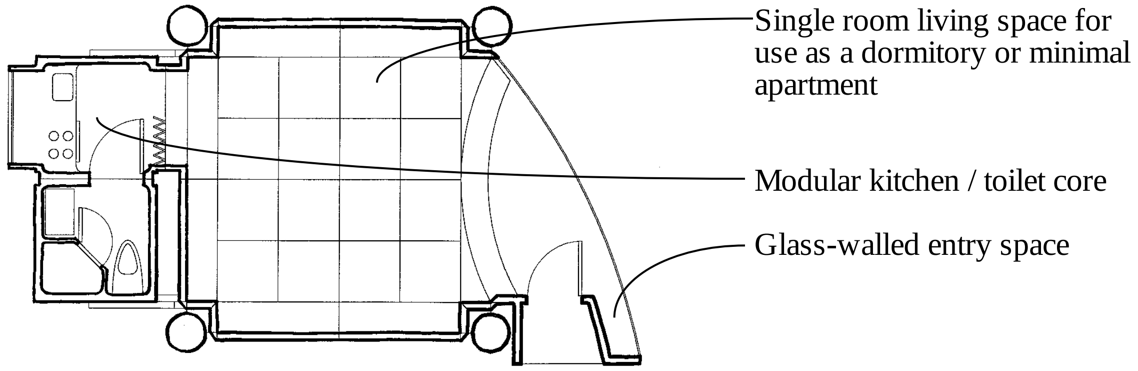

Single room living space for use as a dormitory or minimal apartment
Modular kitchen / toilet core
Glass-walled entry space

Figure 65: Residential unit floor plan

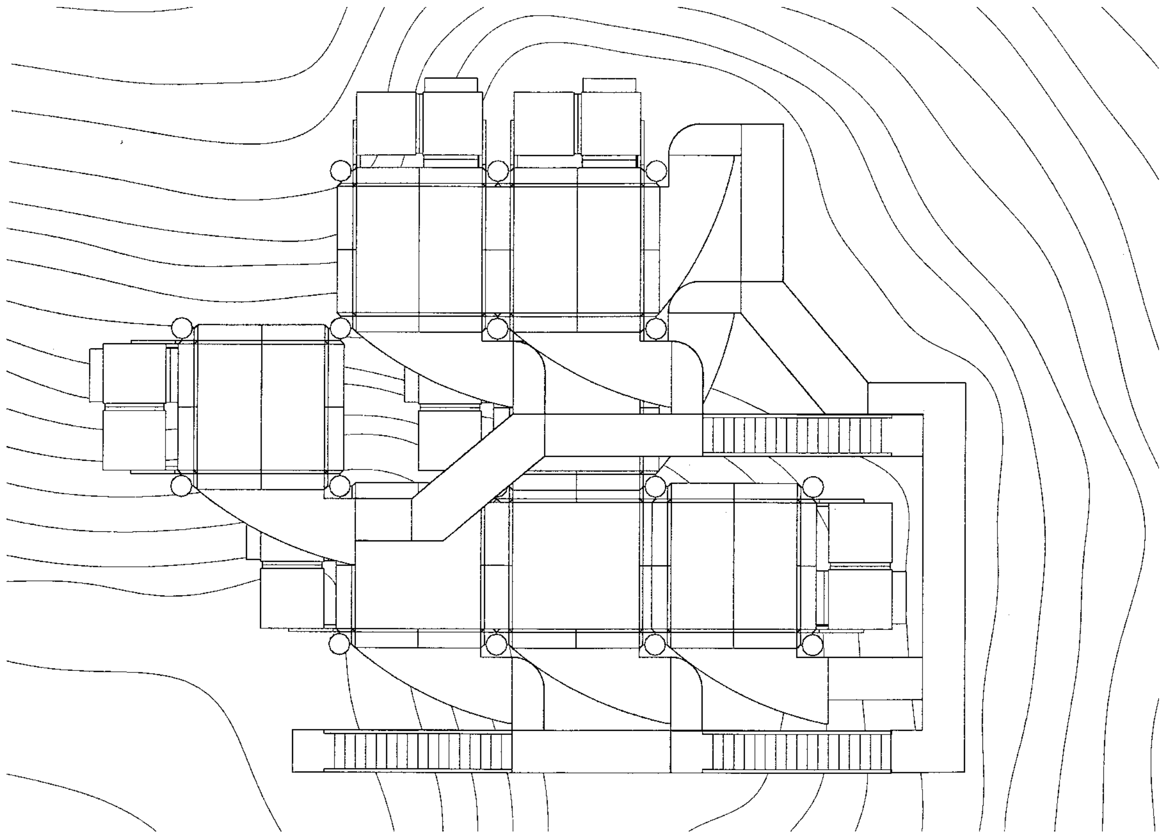


Figure 66: Apartment complex site plan

Since the robotic system was not needed, independent structural bays were not necessary. Instead, a heavy timber column was used that was held in common by adjacent bays. Using the hybrid joint and module-based kit-of-parts, wood panelled components and kitchen / toilet modules were arranged in a manner which used the sloped site to its fullest (Figure 67). In the process of the design, special sloped roof components and glass walled entryways were designed according to the design grammar and added to the kit-of-parts library.

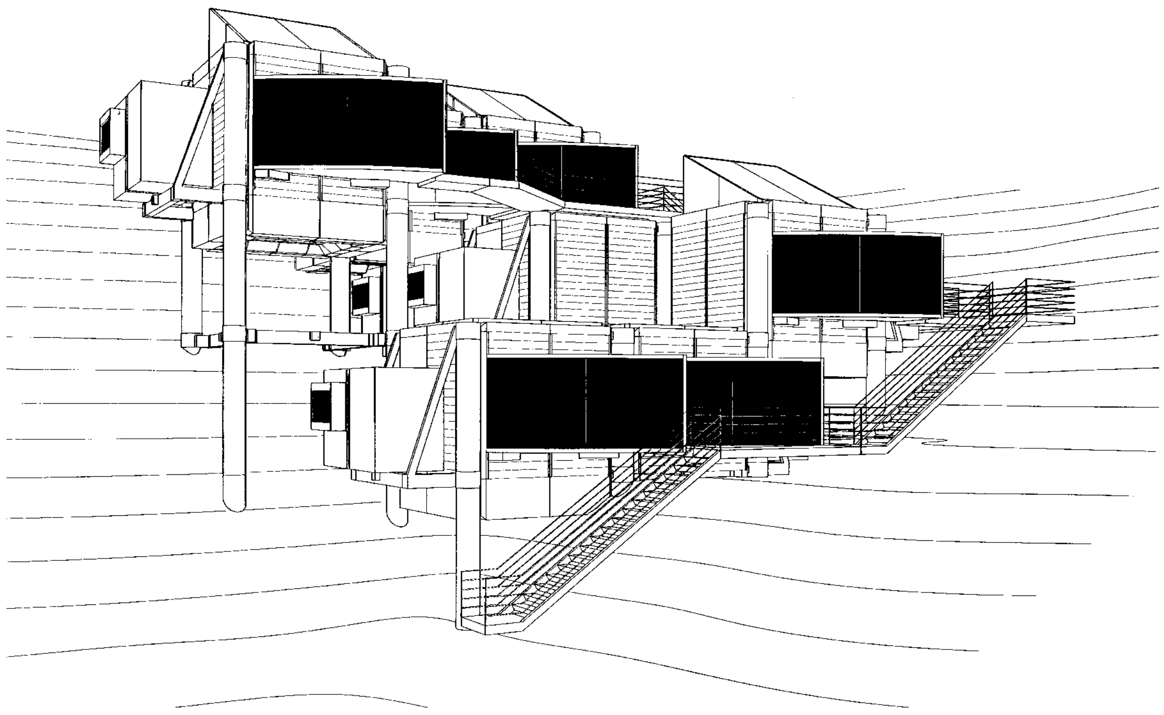


Figure 67: Perspective of apartment complex

The exercise was successful in that the same kit-of-parts used in the chapel design created a totally different building configuration. The chapel designed for a flat site could have been constructed using the automated construction system, while the apartment complex which used the same kit-of-parts would be assembled conventionally. With the establishment of a design grammar, the design of the kit-of-parts system and the construction machines occured in parallel, resulting in a flexible system constructable using robots and automated construction techniques. Since the design grammar was established, new components can be easily designed to address new needs within the framework of the system.

6.2 Yukigen "limited life" building system

The Yukigen project started as an expansion of the LDS Building System, in an attempt to create a marketable kit-of-parts system by Kajima Corporation (as of this writing the project is currently underway). Figure 68 shows some of the features of the proposed system. Since most of the descriptions in the previous LDS Building System section apply here, only a few additional points need be made.

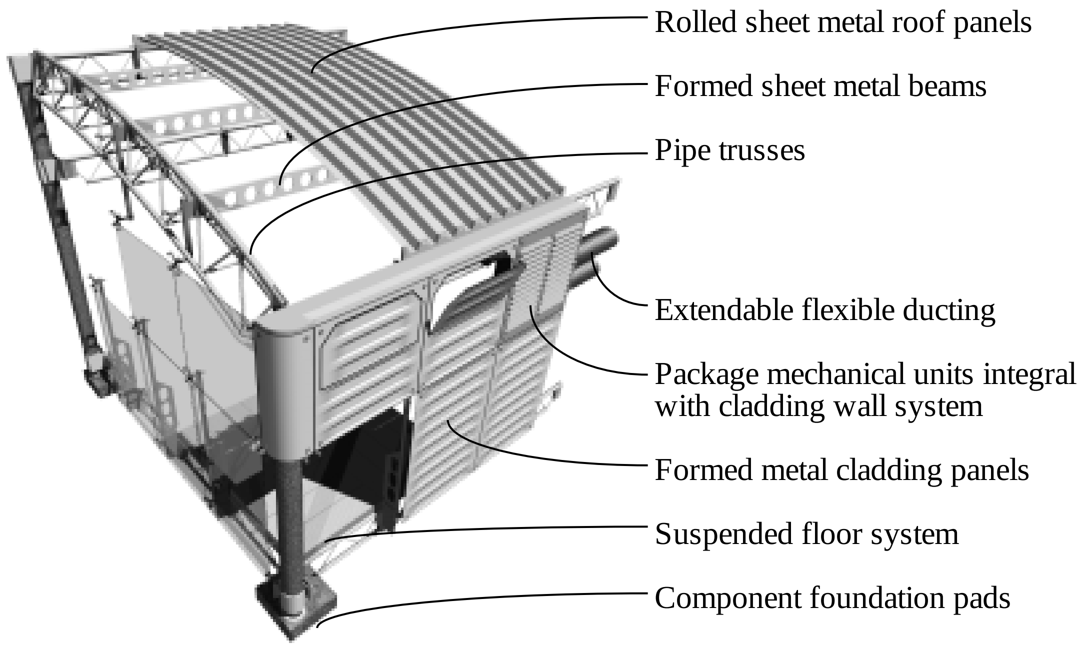

Rolled sheet metal roof panels
Formed sheet metal beams
Pipe trusses
Extendable flexible ducting
Package mechanical units integral with cladding wall system
Formed metal cladding panels
Suspended floor system
Component foundation pads

Figure 68: Yukigen system

6.2.1 Yukigen concept

Yukigen is Japanese for “limited life.” Using the core of the LDS Building System, Yukigen was conceived as an answer to short term architectural needs. In cases where a landowner needs to construct a usable building for temporary purposes, such as for expositions, road-side shops, concerts, temporary additions, or until the permanent building gets underway, a quickly constructed, easily disassembled system was deemed necessary. The kit-of-parts system would be inexpensive and easy to assemble, but would have a high design profile (Figure 69).

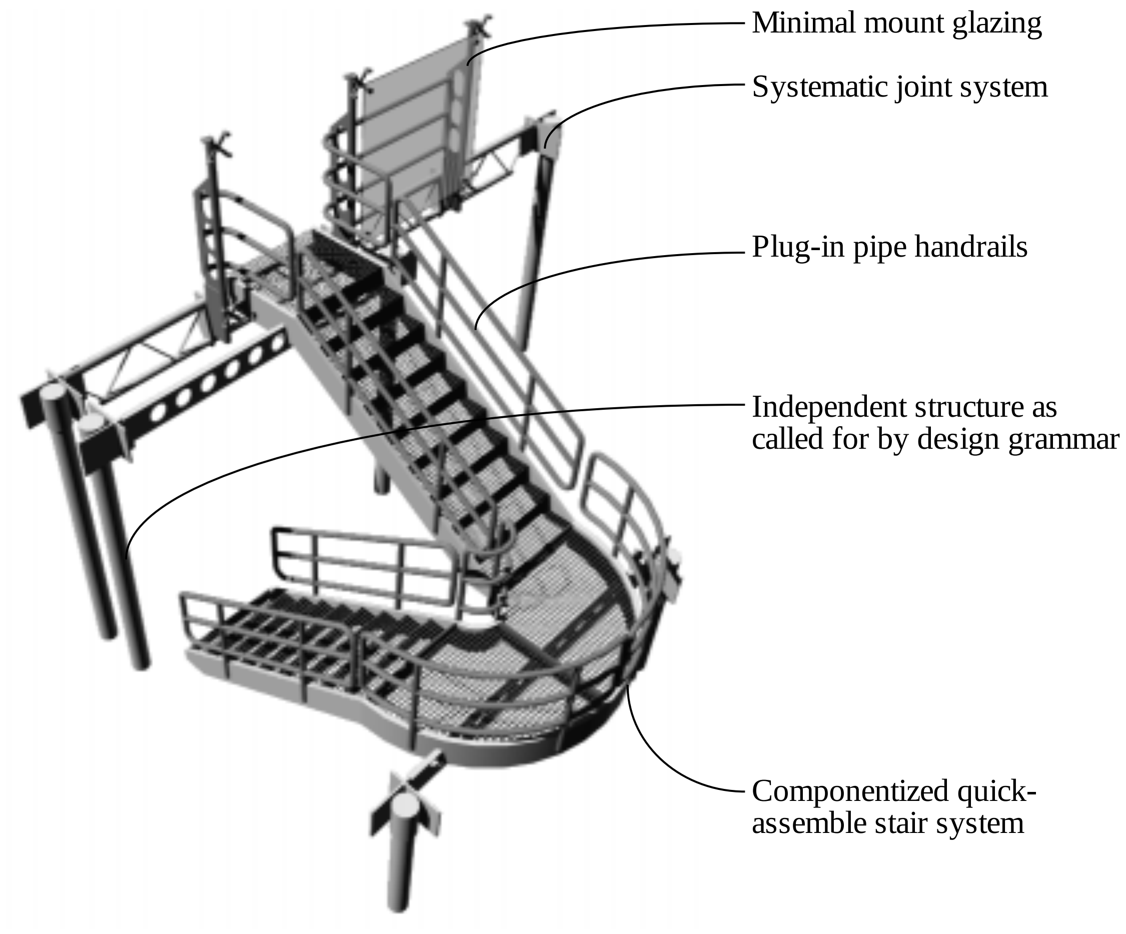

Minimal mount glazing
Systematic joint system
Plug-in pipe handrails
Independent structure as called for by design grammar
Componentized quick-assemble stair system

Figure 69: Quick-assemble, high-profile design components

It was the common belief among the design team members that “Yukigen” could also mean “Mukigen”, which means “unlimited life” architecture. The reason for this was that components that are easily demountable could also be easily recycled and replaced as needed, giving the building a longer life. Under such a concept, a building life of three months or three thousand years, though improbable, would not be unthinkable.

During the design process, it was thought that the building system could be used by a layperson as well as Kajima’s architects. It was determined that putting together a simple database and user interface for retrieving components might be advantageous. There were several preliminary attempts to produce simple web-based catalogs. The importance of developing a life-cycle management paradigm for kit-of-parts systems became apparent. Having the ability to virtually assemble components in the design stage, keep track of them during construction, and monitor their performance for the purposes of maintenance and replacement seemed advantageous. Uses for a management system such as VBuild became clear.

6.3 IMS GNOSIS soft factory “Emerald”

The Emerald factory falls under the IMS Gnosis project (Knowledge Systematization Configuration Systems for Design Manufacturing) which seeks to promote a transition to a new paradigm which will permit sustainable development and changes in the quality of manufacturing systems. Based on an awareness of the limitations of natural resource, the new paradigm will permit a reduction in the necessary material input for manufactuing and product usage as well as in output of discarded items. To achieve this, Gnosis insists that design and manufacturing processes consider the complete lifecycle of products, from design and manufacturing through reuse, recycling, or disposal.

The Gnosis project also embraces a new concept in manufacturing called the Post Mass Production Paradigm (PMPP). Where the current paradigm calls for the production of high volumes of identical products, PMPP attempts to add more flexibility into the manufacturing environment. This is in response to the need for manufacturers to have shorter retooling and setup times in order to have the ability to produce a greater variation of products. Using kit-of-parts concepts and grammar-based design techniques, it is thought that many of the PMPP goals may be achieved.

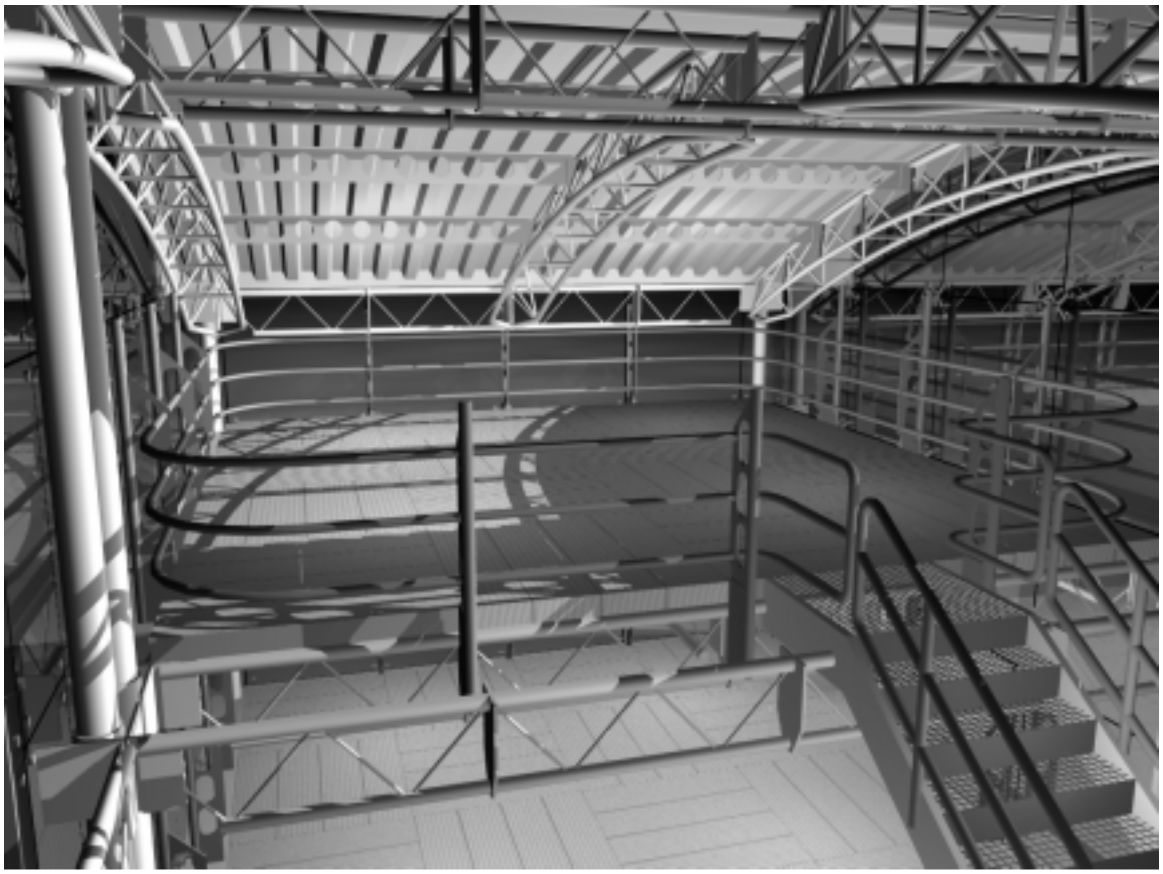


Figure 70: Emerald factory interior

The Emerald factory is conceived as a demonstration project designed according to shape grammars and constructed from a kit-of-parts. The kit-of-parts is based on units or blocks of space which generate their own enveloping structure. The enveloping structure can be real or imaginary depending on the adjacent space. In the Emerald project, an expansion of the LDS Building System / Yukigen kit-of-parts was made in attempt to define the role of life-cycle management in more depth. Attention was given to design grammars, such as shape grammars and rule-based assembly, and an analysis of the design process was undertaken in order to extrapolate principles for use in the development of a designer expert system for kit-of-parts systems.

6.3.1 Shape grammars

Buildings and other artificially created structures have three-dimensional mass and volume, and consist of sophisticated combinations of different shapes. If it were possible to extract the shapes in such a way as to define a finite number of basic primitives, a shape "alphabet" of sorts could be derived which could potentially be used to describe the structure. If rules governing the various combinations of the shape primitives could be established, an underlying shape language which describes not only the original building, but other similar (and possibly non-similar) structures could be derived. In current and past shape grammar studies, tools have been developed which encapsulate design styles for the analysis of existing structures and the generation of new designs [Liou 1992]. Most shape grammar studies have either addressed the analysis of existing buildings, or have dealt with new designs using traditional forms of construction [Flemming 1987]. Very few attempts have been made to derive shape grammars which actually celebrate the construction itself or which actually cater to newer technologies.

It has been claimed that shape grammars often take a "kit-of-parts" reductionist approach as opposed to the top-down holistic view [Chase 1997]. Yet in considering some advanced building and structural systems which use prefabrication and modular technologies, a reductionist approach may be appropriate at some level in the design process. In the current research, shape grammars have been considered which are innately tailored to physical kit-of-parts building components and their different combinations. The shape grammars define rules for the design of buildings which lend themselves toward automated construction systems and robotic erection [Howe 1997a]. In this respect certain trade-offs have been made: complete flexibility as opposed to efficient manufacturing; free utilization of raw materials as opposed to controlled wastage and recycling; cluttered, hazardous construction sites as opposed to immaculate, safe, low labor intensive working environments; long, drawn-out erection schedules as opposed to overnight assembling. Since the same trade-offs have successfully been made in the manufacturing industries, this research advocates the possibility that shape grammars be developed which specifically serve the *Hi-tech* corner of the architectural design community as well. In short, though not necessarily as abstract as the main stream shape grammar studies, this research encourages the idea that kit-of-parts shape grammars may contribute to the development of PMPP and the overall goals of an IMS Gnosis type project.

Starting with the kit-of-parts based on previous work in the LDS Building System and Yukigen projects, a brief description of the shape grammar is as follows:

1. Emerald factory kit-of-parts tree structure

   a) Zoning parts (units or blocks of space set on a base grid of 0.90 meters)

   - space zone: includes a set of discrete spatial components. BAYyyxx is standard form with p being the size parameter

     BAYyyxx : {p = xx,yy |p = 18,36,54,72}
   - structure zone: indicated or derived from the corresponding space zone

   b) Material parts (kit-of-parts system which encloses or is generated by the zoning parts) AAAxxxx (standard form) : {p = xxxx |p is parameter}

   - space zone:

     FLRyyxx : {p = xx,yy |p = 09,18}

ROFyyxx : {p = xx,yy |p = 18,36,54,72}

...

- structure zone:

COLxxxx : {p = xxx |p = 0200, 0350, 0800}

EVCxxxx : {p = xxx |p = 0090, 0180, 0270, 0360}

EVCyyxx : {p = xx,yy |p = 04,09,18}

EVSyyxx : {p = xx,yy |p = 04,09,18}

JNTxxxx : {p = xxxx |p = 0200, 0350, 0800}

TRSxxxx : {p = xxxx |p = 0018, 0036, 0054, 0072}

WLCxxxx : {p = xxx |p = 0090, 0180, 0270}

WLCyyxx : {p = xx,yy |p = 04,09,18}

...

2. Emerald factory kit-of-parts shape rules

a) viewpoint

(V, ϕ) -> (V, {(0,0): $^{\text{LABEL}}$})

(V, {(0,0): $^{\text{LABEL}}$ }) -> (V, {(x,0): $^{\text{LABEL}}$ })

(V, {(0,0): $^{\text{LABEL}}$ }) -> (V, {(0,y): $^{\text{LABEL}}$ })

...

b) rotation

BAYyyxx-000 -> BAYyyxx

c) addition

BAYy1x1-rr1 -> BAYy1x1-rr1 + BAYy2x2-rr2

d) subtraction

BAYy1x1-rr1 + BAYy2x2-rr2 -> BAYy1x1-rr1 xor BAYy2x2-rr2

These notations simply encapsulate the design grammar described earlier under the LDS Building System, and attempt to provide a means of manipulation for Genetic Programming (GP) experts. In the context of the Emerald project, the author designed the kit-of-parts libraries and overall shape grammar concepts. The author's co-researchers, Katsuhiko Watanabe, Hiroyuki Nakagawa, and Hiromasa Akagi, have developed GP algorithms which use the kit-of-parts and attempt to derive design solutions through emergence (discussed later).

6.3.2 Spatial kit-of-parts

A space is a three-dimensional volume with real or implied boundaries. In architectural design, the most important task is to provide spaces for the support of human activity. It is imperative that the space be sufficiently sized to contain the specified function or activity without detracting from it. The second great task of the architectural design process is to devise a way of containing the spaces. The designer must conceive of boundaries for defining the volumes in such a way as to hint at the nature of the proposed activity. These physical or implied envelopes ideally incorporate mechanisms for enhancing the atmosphere of the volumes, which can also add to or detract from the proposed activity.

In the Emerald kit-of-parts research, a concept was developed which takes the spaces themselves and establishes a spatial kit-of-parts. The spatial components are derived by shape grammars which ideally could define any imaginable volume primitive. In considering the void itself as a physical component instead of concentrating on the substance which defines the void boundaries, a number of interesting possibilities present themselves. The void as a solid represents the way a designer thinks when performing volume studies, and specific or ambiguous blocks of function can be assembled with ease within legal buildable envelopes. Translating the volume boundaries into a buildable structure is typically a demanding task. In the current research, the spatial kit-of-parts components themselves generate the structure around themselves according to the shape grammars. Assembly rules fill in the solid components, and where overlapping spatial primitives occur they are either deleted at the common boundaries or are changed into interfacing members.

6.3.3 Types and functions

Though the Emerald work attempts to maximize the use of kit-of-parts technology and reductionist shape grammar to help define the Post Mass Production Paradigm, the application of genetic programming to obtain emergence in design required a certain holistic approach as well. Since the spaces generate their own structural envelopes, a mechanism was needed that would act as a set of rules for the grouping of spatial components themselves.

It was decided to implement a heirarchy of grammars that would start with simple graphic imput for overall conceptual diagramming, wherein the genetic algorithm could have a starting point for manipulating the spatial components. As a foundation for this, work by [van Pelt and Westfall 1991] involving building types was explored as a means for assigning attributes or parameters to the spatial components.

Pelt and Westfall explained, "A building type is a generalized, unbuildable idea of a building containing within it all the possible examples of actual buildings of that type that have been and can be built". The work describes six types that all buildings can be abstracted into, based upon their function: "tholos" as a place of veneration, "templum" as a temple or place of celebration, "theatrum" or theater, "regia" as a place of governing, "domus" as a place of dwelling, and "taberna" as a shop or place of production. Though each of the six types generally represent buildings in themselves, they may also represent spaces within a building. In other words, all buildings may actually be composites of several different types.

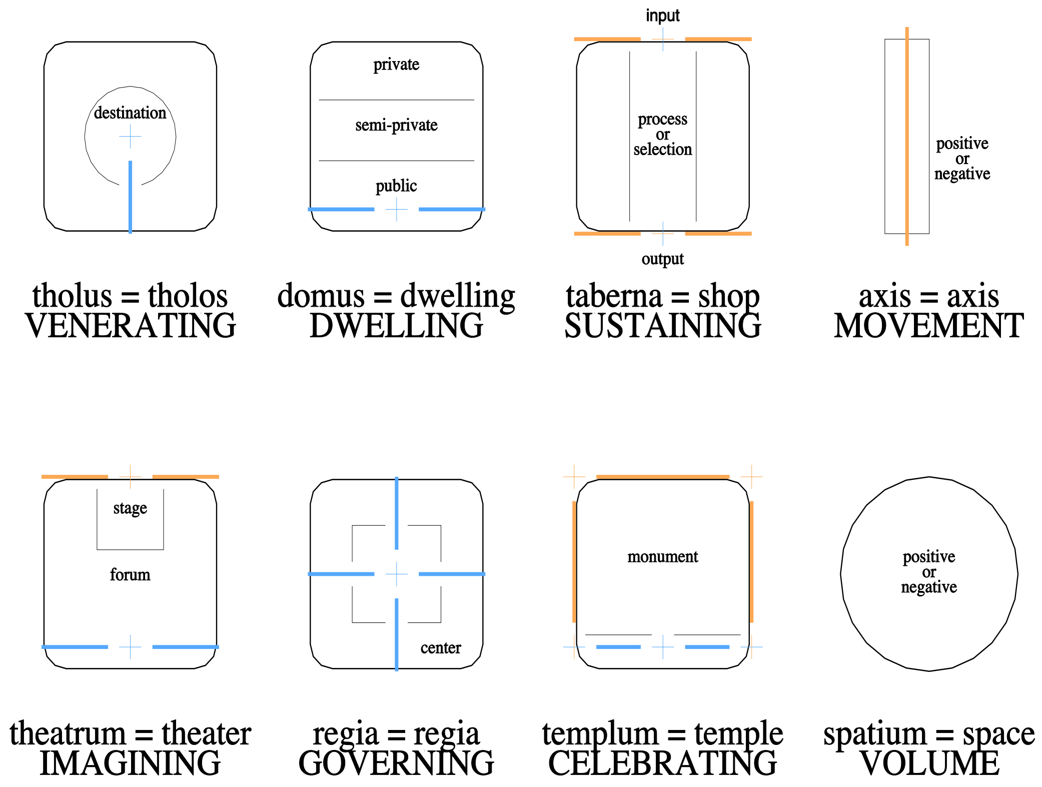


Figure 71: Type icons

For the purposes of developing the Emerald kit-of-parts life-cycle management system, two more types have been added: "axis" as a place for movement, and "spatium" as a generic volume when the specific type is yet unknown. Since a space may be a volume with physical or implied boundaries, all eight types (shown in Figure 71) could be considered either positive or negative for the purpose of defining physical building volume or virtual volumes such as plazas, corridors, and streets. These eight types have been prepared as a macro kit-of-parts for overall concept diagramming. Using the eight types, it might be possible to devise recipes for minimal inclusion and adjacencies that a designer can use as is, or adapt according to need.

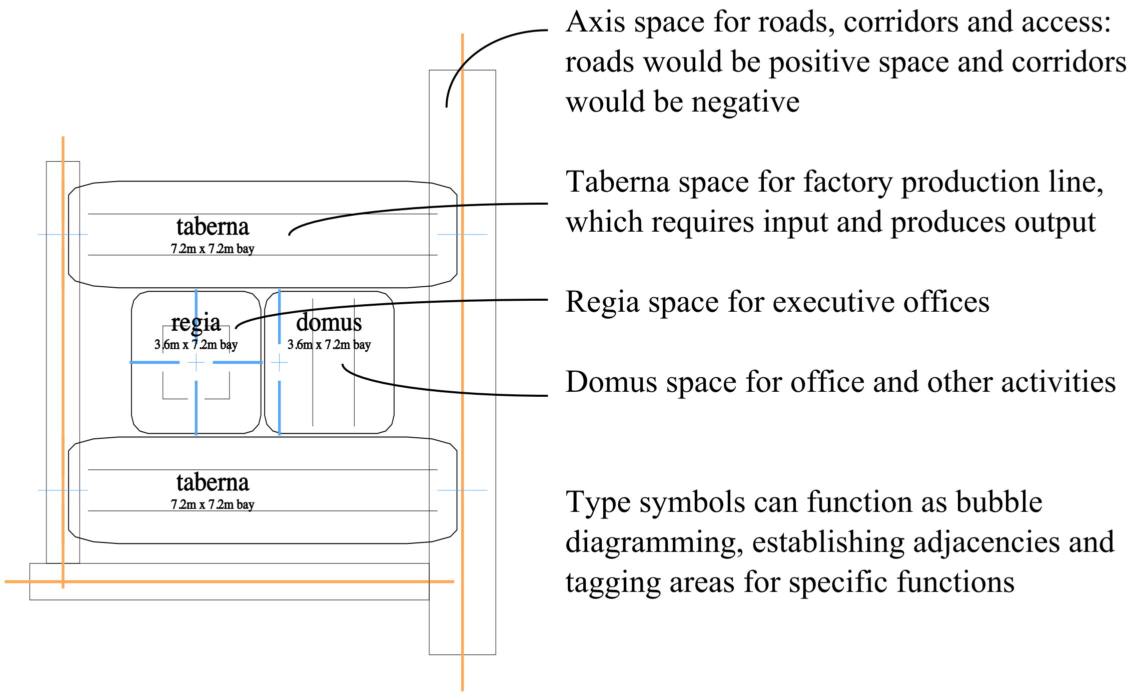

Axis space for roads, corridors and access: roads would be positive space and corridors would be negative
Taberna space for factory production line, which requires input and produces output
Regia space for executive offices
Domus space for office and other activities
Type symbols can function as bubble diagramming, establishing adjacencies and tagging areas for specific functions
taberna
7.2m x 7.2m bay
regia
3.6m x 7.2m bay
domus
3.6m x 7.2m bay
taberna
7.2m x 7.2m bay

Figure 72: Type template for factory with multiple production lines

Since the Emerald factory problem was chosen as an example architectural problem, three sample recipes for factoies have been explored: 1) a "bare minimum" template consisting of a simple taberna component with input and output, and an imbedded domus or regia component for a small office, 2) a template with one production line and separate office / administration, and 3) a template with multiple, separate production lines (Figure 72). These templates were devised for the purpose of this research only and should not be considered as ideal recipes. It was assumed that more appropriate templates can be devised by appropriately experienced designers, or in real time using knowledge bases and expert systems, which is beyond the scope of this research. However, within the context of the Emerald project it was assumed that templates could help establish a basis upon which an expert designer for kit-of-parts systems could be developed.

6.3.4 A design scenario

In a design scenario, the type icons can be a preliminary kit-of-parts. In the beginning the site is valueless as indicated by rows of zeros in Figure 73. The zero values shown here for clarity stand for probability of building density, and are actually invisible to the designer. Initially there is no value until the designer indicates where building volumes should go. Figure 74 shows the site with density probability values assigned using spatium and axis space icons. The designer would indicate positive space by assigning the icon a blue color, and negative by assigning red. In cases where two blue spatium or axis icons overlap, the probability is strengthened. In the same way, two negative overlapping icons decrease the value.

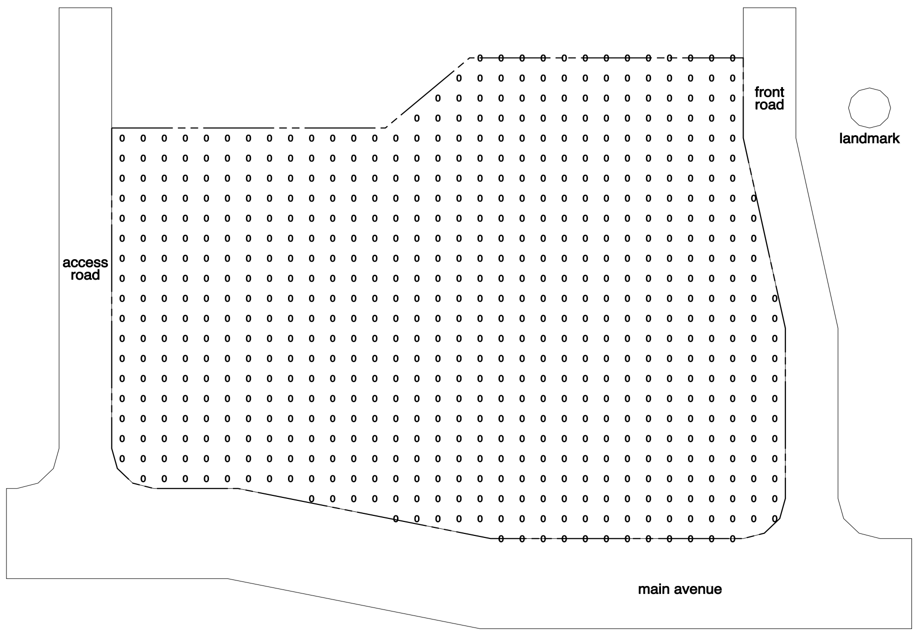


Figure 73: Initial site: valueless

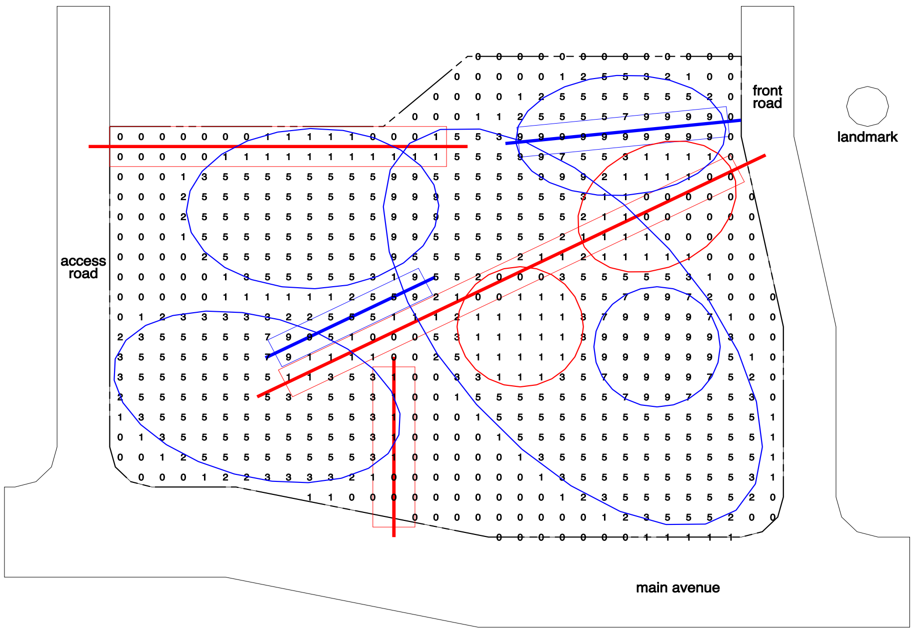


Figure 74: Density probability values assigned

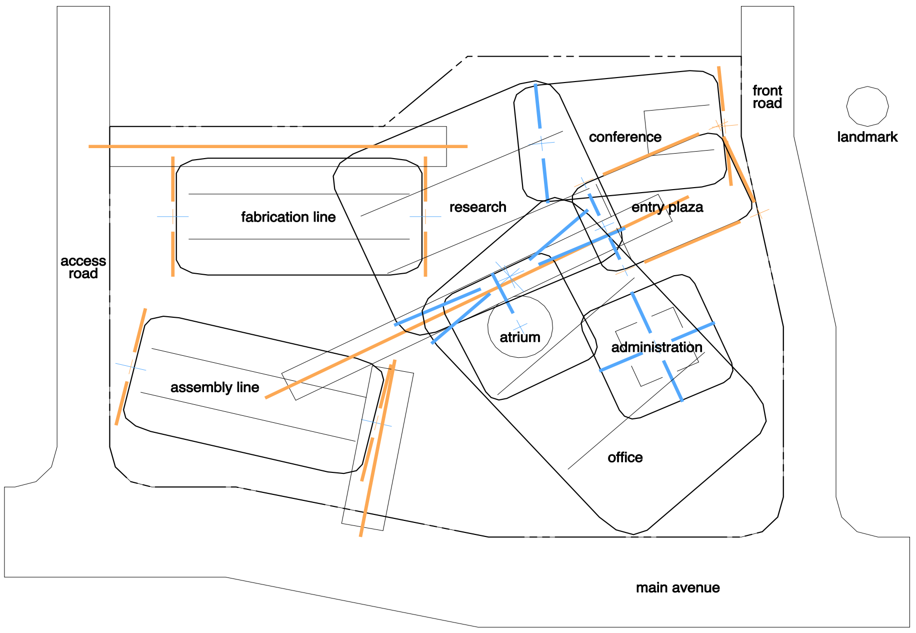


Figure 75: Using type icons to assign density probability

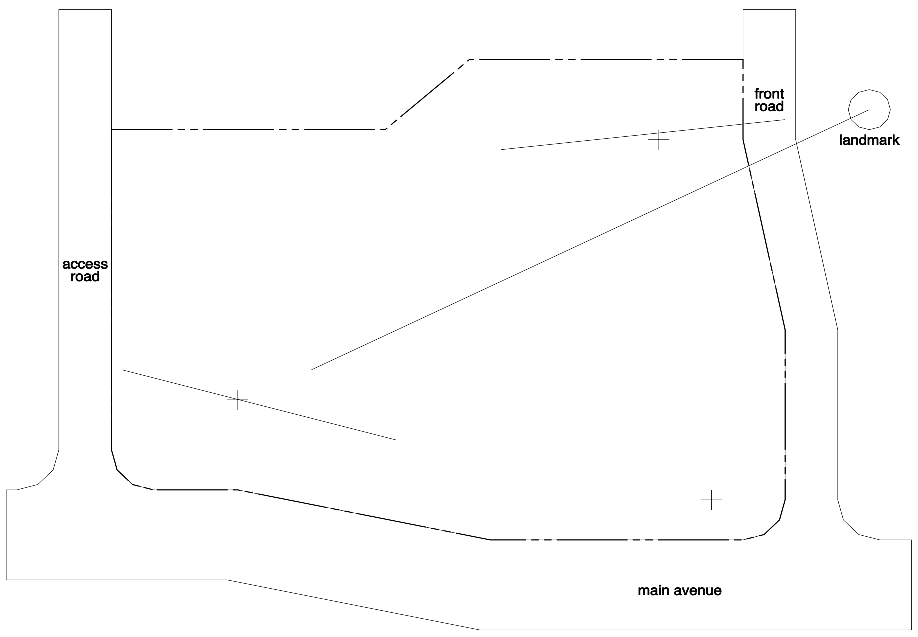


Figure 76: Inputting orientation

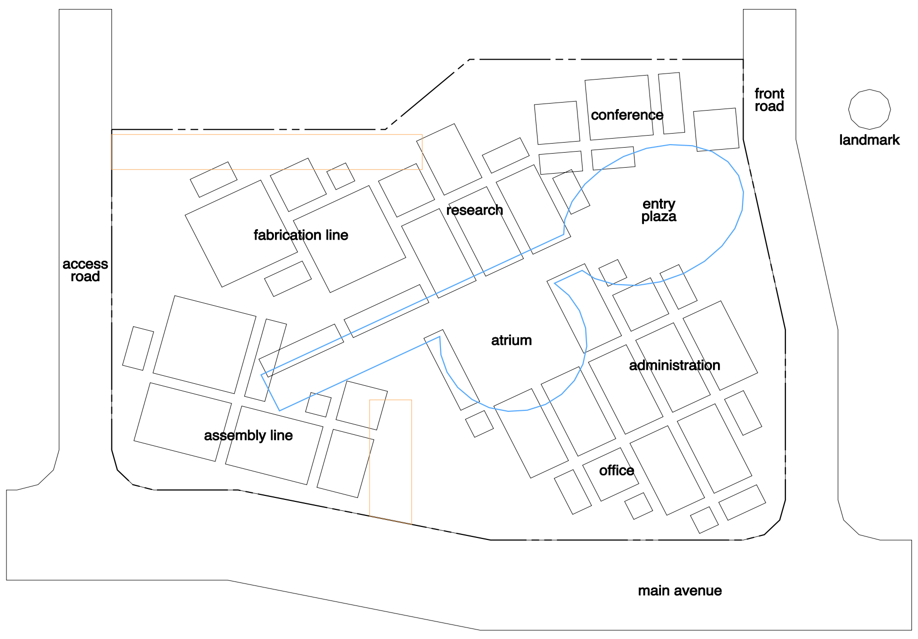


Figure 77: Spatial component layout

At this stage, rather than use the spatium icon, the designer could use the templates for starters or begin by placing the other type icons according to intuition and experience. Figure 75 shows intuitive placement of icons, taking into account the front road and landmark.

This example factory starts with a negative axis aligned with the landmark, with supporting research, office, and conference facilities branching off. Within the office area, a negative atrium space opens off the main axis, and higher density administration is set apart a ways. Two factory production lines, fabrication and assembly, are placed away from the main public area and have access roads to accomodate the input and output.

In the next step, the designer indicates orientation. Figure 76 shows a series of lines and plusses. The designer first inputs a straight line to show structure orientation, then a plus to indicate where the grid should begin. In this example, three different orientations have been inputted. As a result, the GP would generate locations for spatial component layout according to the inputted orientations. In Figure 77 notice that the fabrication line in the upper left-hand quarter of the site has spatial components oriented the same as the research and office portion of the complex, which breaks up the overall space. Also, though the assembly line in the lower left-hand quarter may be oriented correctly, it is also broken up. Perhaps this is intentional, but there may be problems since the taberna type icons in their nature indicate large spans. With such a broken-up condition, the designer would probably want to go back and reset the orientations and adjust the encroaching axis icon.

The author's co-researchers have created two programs. A Prolog language GP engine derives generations of variations of design solutions through emergence and outputs a three dimensional model consisting of spatial components from the kit-of-parts library. The other program is in the Java language and is called the Rule-based Assembler (RBA). RBA assembles structure and cladding components from the kit-of-parts library onto the spatial blocks in the GP output model: the spaces generate their own structure. RBA also resolves adjacencies, deleting members that occur in continuous adjacent spaces and replacing overlapping members with interfacing components.

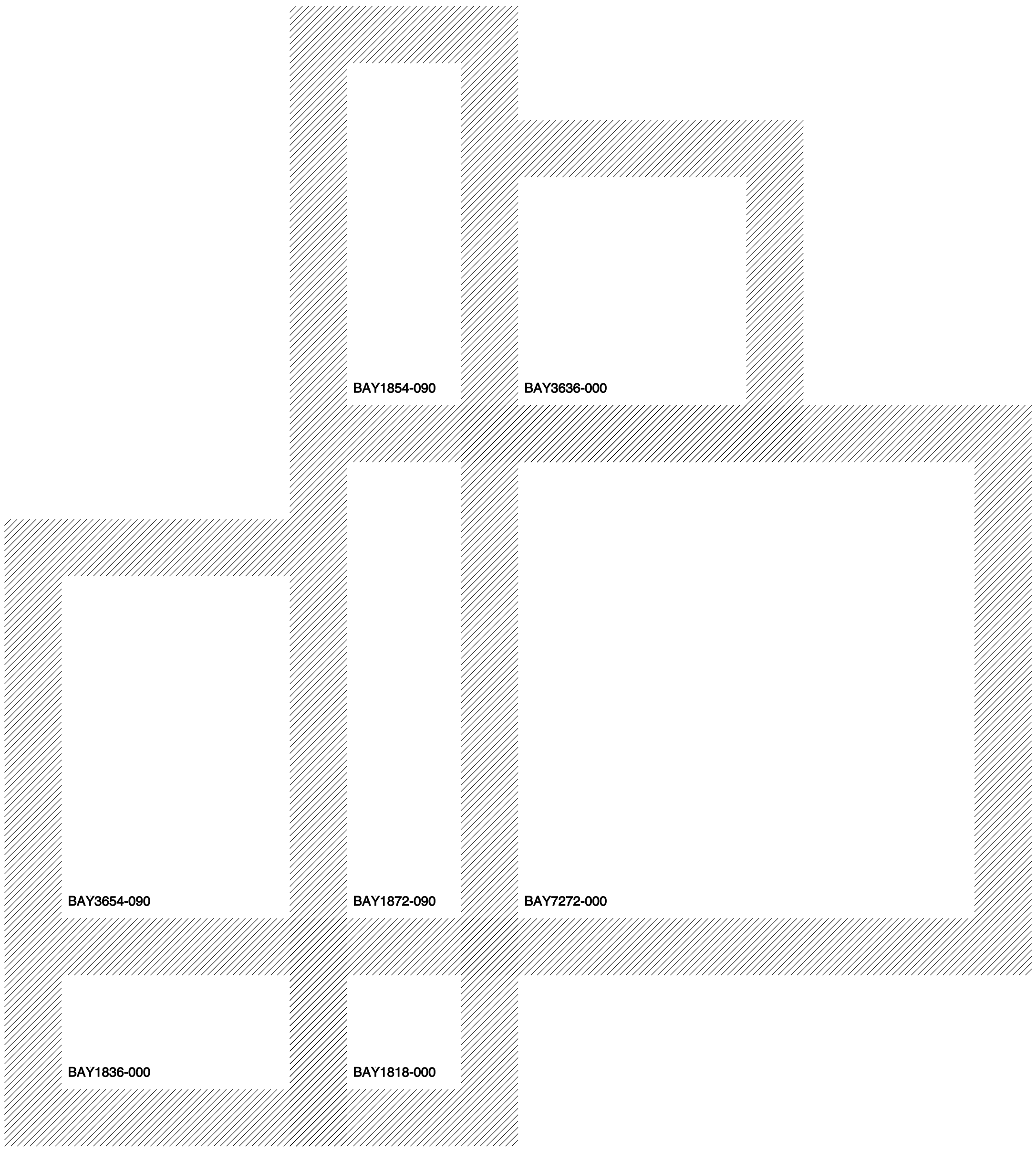


Figure 78: Spatial components in space zone / structure zone layout

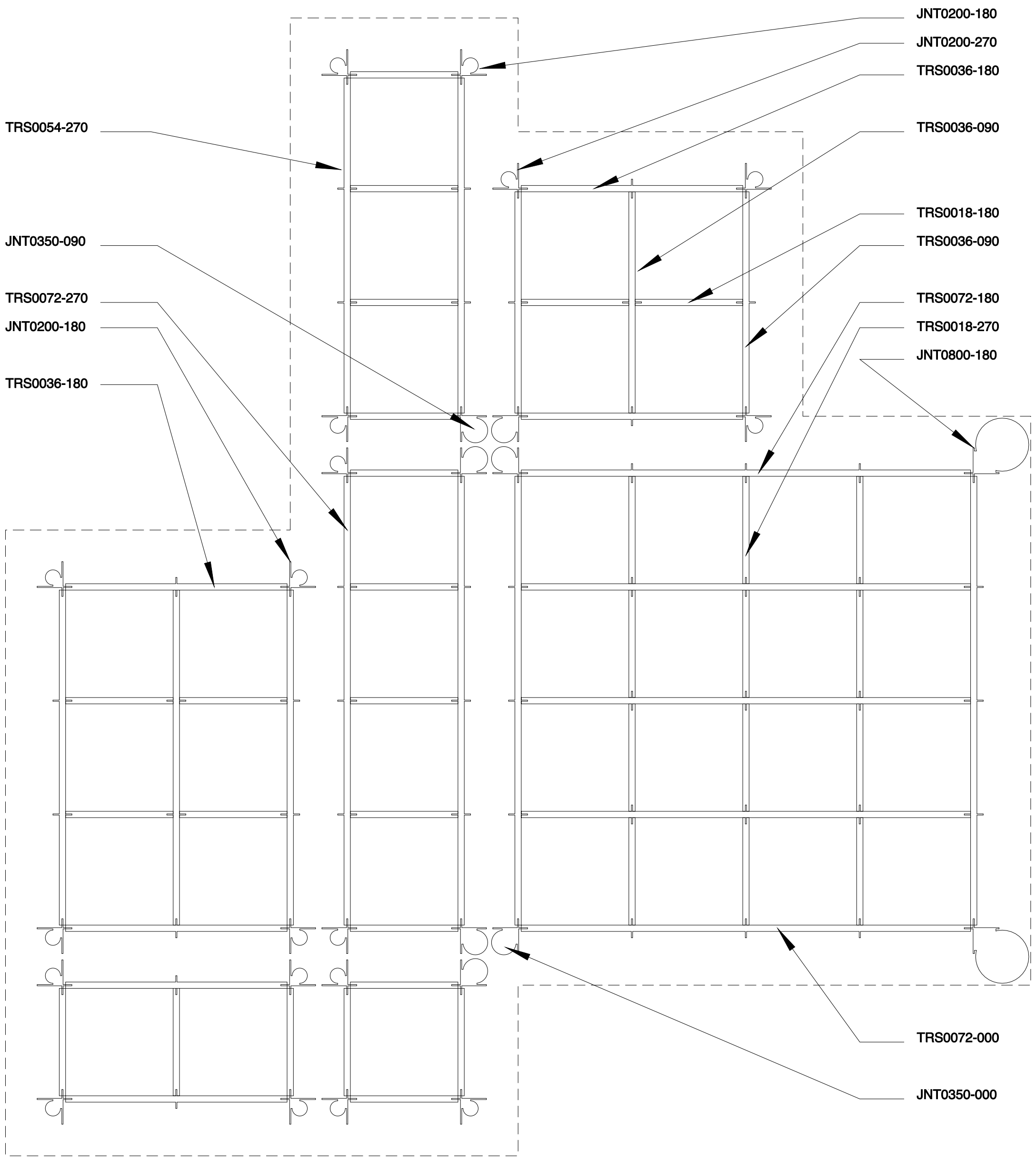


Figure 79: Framing component placement

In Figure 78, spatial components are shown in a space zone / structure zone context. Spatial component names are shown in each space, placed similar to how the GP engine would derive an optimum placement through emergence. Figure 79 shows how RBA would place structural components.

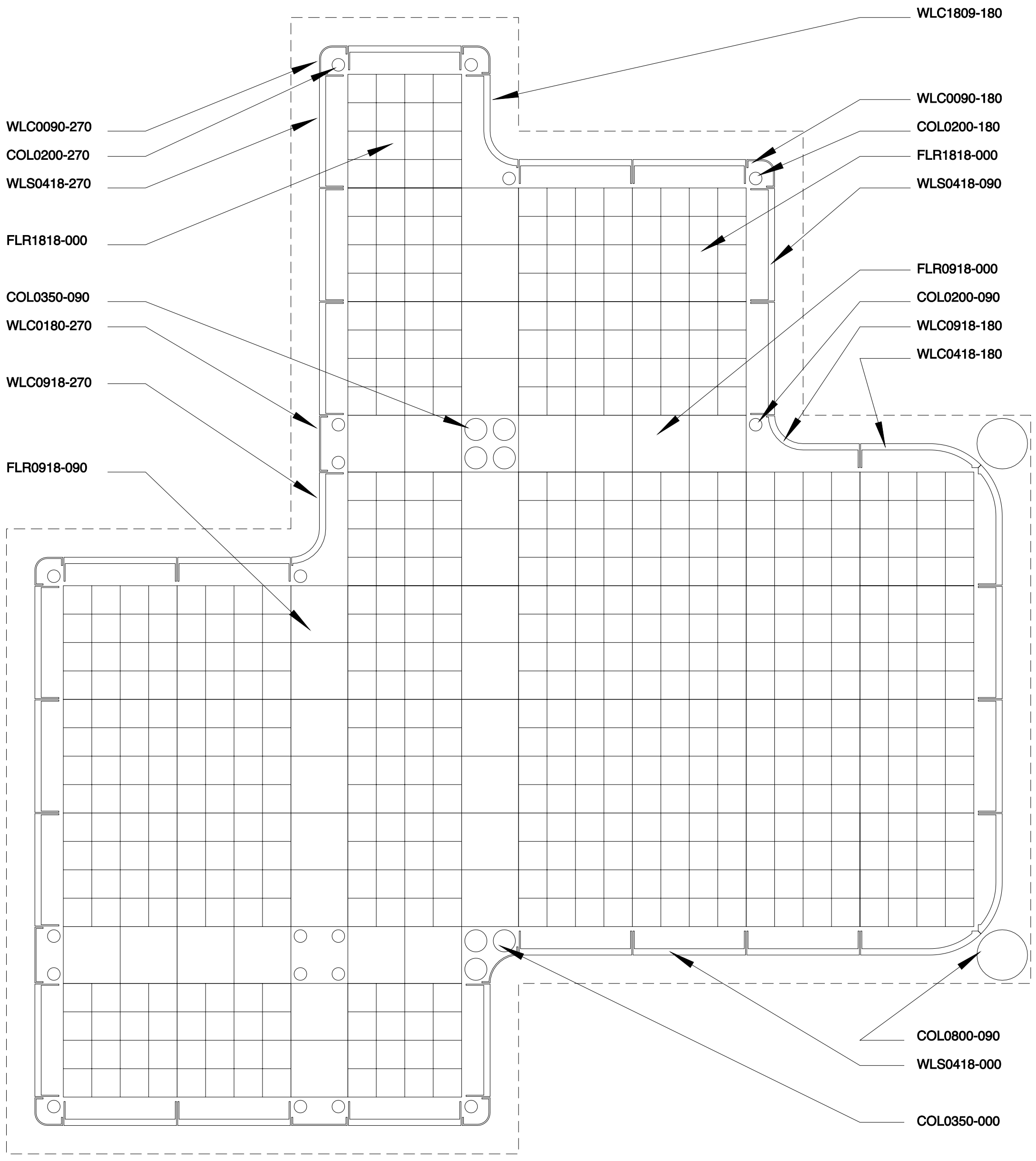


Figure 80: Floor plan component placement

RBA can be given rules to replace long-span spaces with pre-designed deep trusses and large diameter columns, and give short spans smaller members. In a similar manner, RBA places floor plan and roof plan components as in Figure 80 and Figure 81, respectively. Currently there are plans to post the GP and RBA programs on the Internet and link them to VBuild as a design expert under the expert menu.

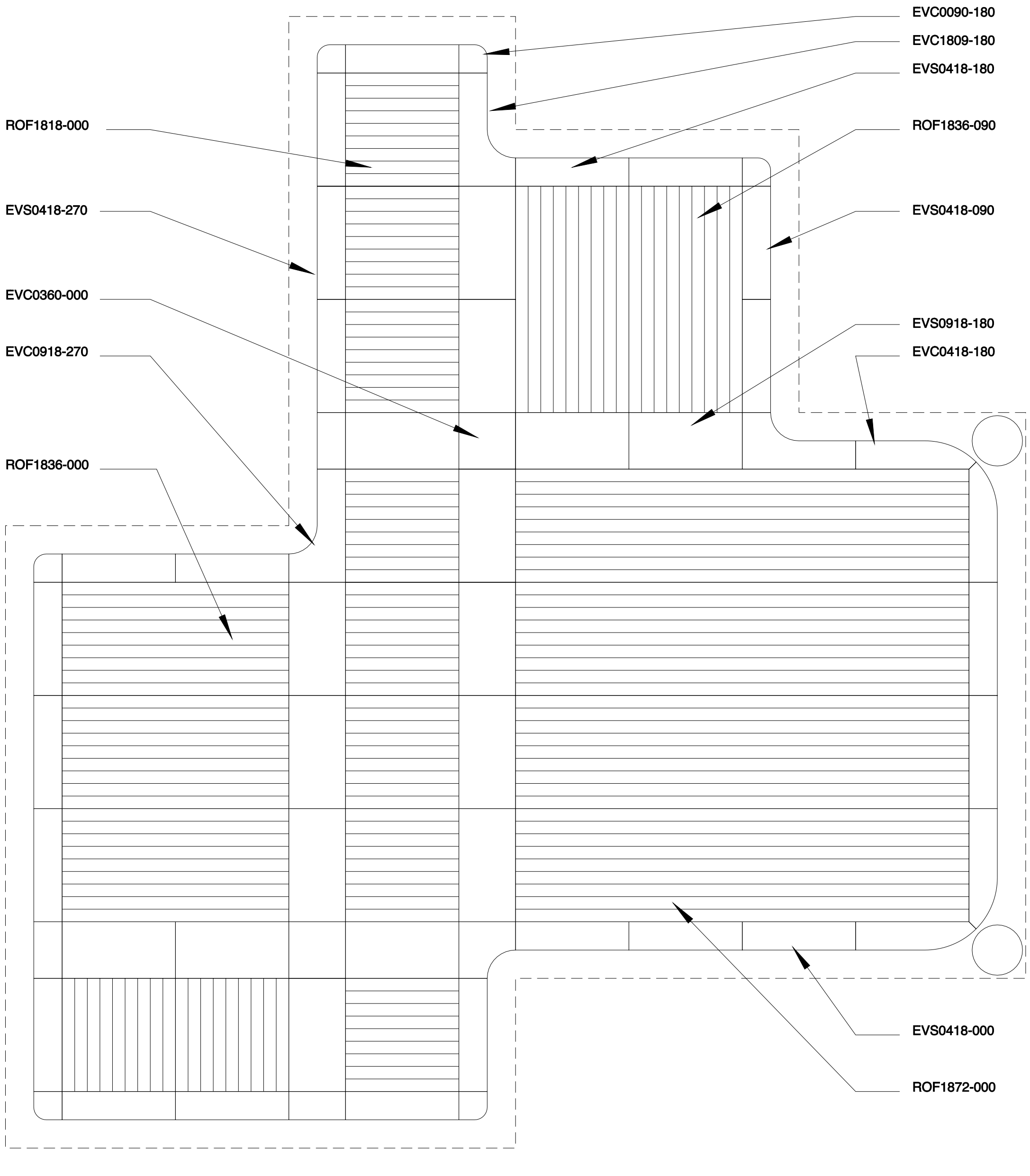


Figure 81: Roof plan component placement

6.4 Summary

A hybrid joint and module-based kit-of-parts system was designed as a possible solution for the building facility needs of a growing international organization. A design grammar was established which could guide structural and spatial design, and also dictate automated construction system design. The grammar allowed the building structural bays to coincide with a robotic building system work cell.

In extended research, the need for a virtual kit-of-parts library that could be accessed by a designer via a network was established. Considering that the spatial volume of the building could be divided up into blocks of space, spatial components were added to the kit-of-parts library. Tools were developed which could be the beginning of an expert design system which uses the kit-of-parts and generates design solutions through emergence.

Except for the lack of remote monitoring and control, the Life-cycle Architecture System project could be an ideal representative of the life-cycle management paradigm. A representation of the kit-of-parts in a virtual form such as the VBuild core objects would enhance the management capabilities of the system and bring the different stages together in a single environment. Though the Life-cycle System has not been entirely conceived with the paradigm in mind, the research and design investigations conducted in conjunction with the project has been instrumental in the development of the paradigm.

CHAPTER 7
CASE STUDY: PLUG-IN CONDOMINIUM PROJECT

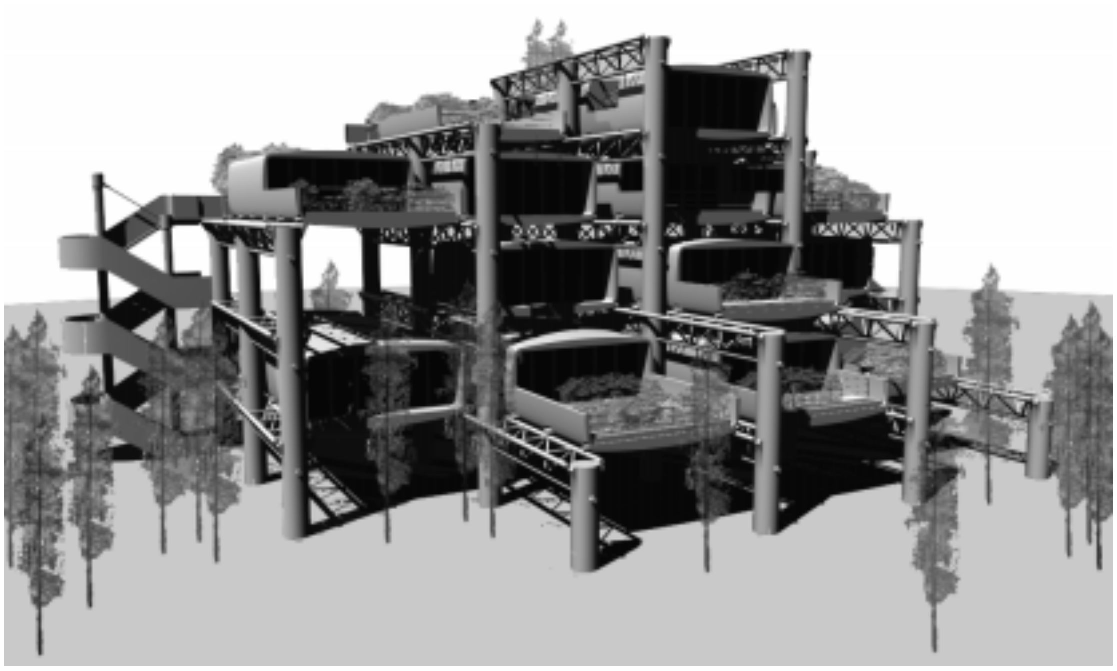


Figure 82: Plug-in condominiums front view

The Plug-in Condominium project falls under the IMS IF7 project (Innovative and Intelligent Field Factory) which aims at the realization of innovative and intelligent production systems which will execute effectively the assembly of large scale structures such as buildings, ships, or bridges [Miyamoto, et.al. 1998]. Various efforts for mechanization or automatization of large scale structures have been carried out and diverse satisfactory results have been achieved. The IF7 research, based on these results, introduces the new concept of Field Factory. The Plug-in Condominium project is conceived to be a demonstration project which showcases the Field Factory concept. Using kit-of-parts construction, Constructivist and Metabolist spatial arrangement, and Support / Infill (SI) philosophy, the plug-in apartment project attempts to apply a life cycle management concept as proposed by this work. Though the project was not modeled using VBuild, the author's design proceeded in parallel with the development of Vbuild, and it was assumed that a fully developed program could be used in the life cycle management of the project.

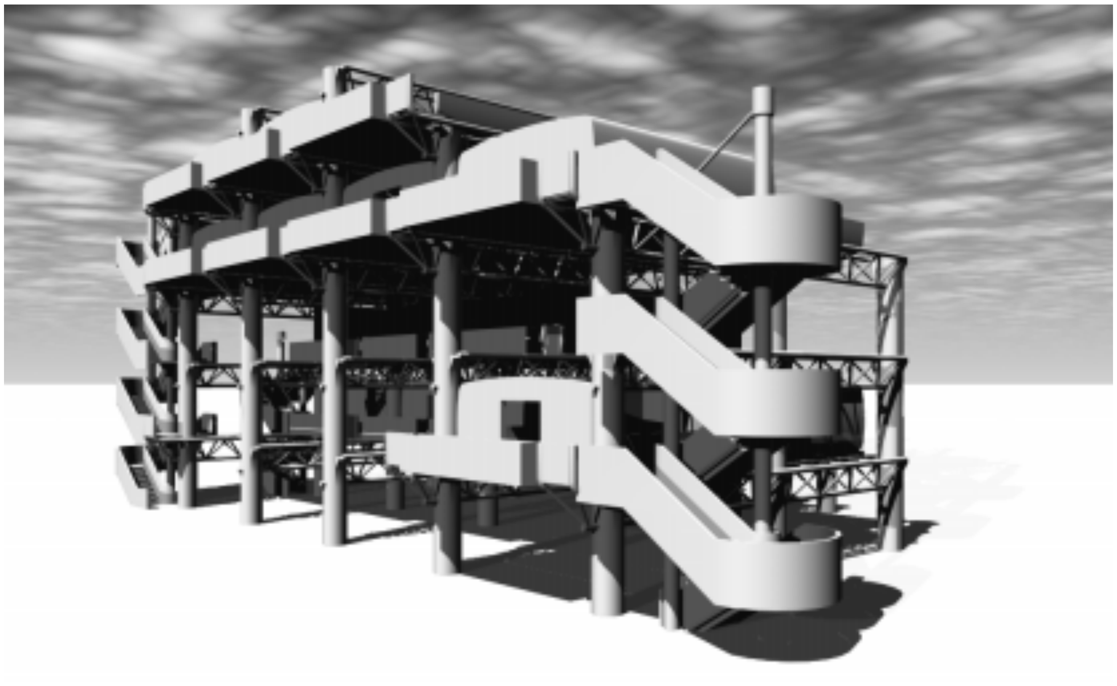


Figure 83: Plug-in condominiums rear view

7.1 Design process

The Plug-in Apartment project consists of two major systems: the skeletal support superstructure and the plug-in infill module system. The two systems interact with each other using a standard virtual module which defines infill module size, support points, utility connections, and circulation infrastructure. Support systems must be designed to bear the standard virtual module, and the infill components must be enclosed in the module and use the standard interfaces. The standard virtual module is established to allow different manufacturers to develop their own concepts based on the module, in order that various systems can be mixed and matched as needed for individual needs. In the IF7 project, three major Japanese construction firms devised three different support systems which fit the virtual standard module. The three support systems are completely different in appearance, construction, and erection process. In this chapter, only the system proposed by Kajima Corporation is described, which was designed by the author using principles, concepts, and technologies developed or obtained during the course of this work for the purpose of life cycle management using a web-based tool such as Vbuild.

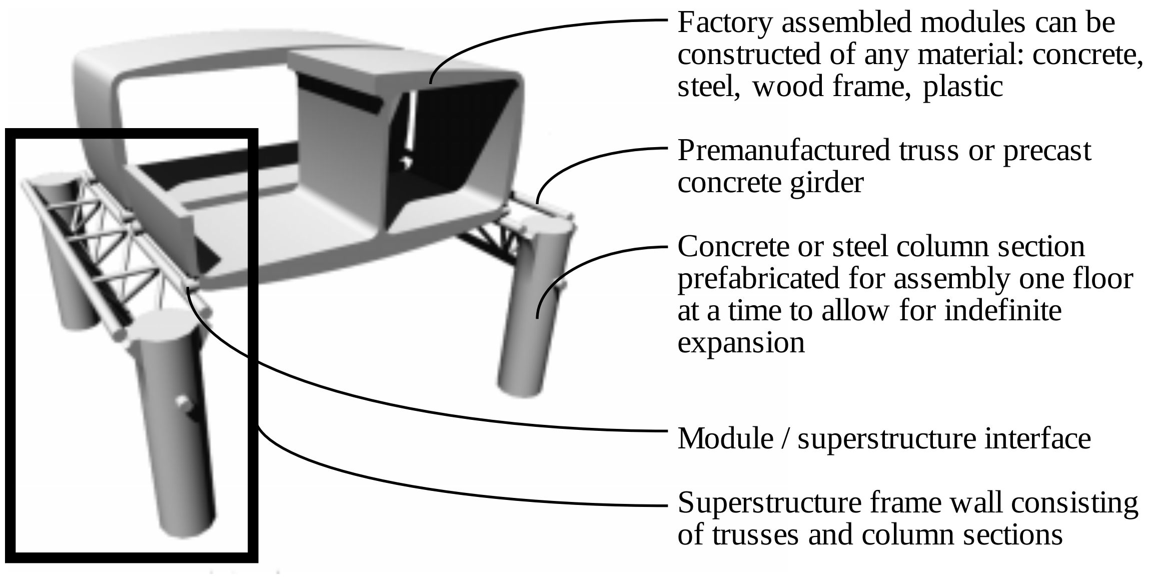

Factory assembled modules can be constructed of any material: concrete, steel, wood frame, plastic
Premanufactured truss or precast concrete girder
Concrete or steel column section prefabricated for assembly one floor at a time to allow for indefinite expansion
Module / superstructure interface
Superstructure frame wall consisting of trusses and column sections

Figure 84: Support and infill system

7.1.1 Support design

Initially a foundation would be constructed, designed to bear the maximum sized structure possible for the site. The foundation pads are empty sockets waiting to receive column components. Using a joint-based kit-of-parts system, the support structure would be expandable or demountable to allow for specific needs. Not all of the foundation sockets need to be used at first, only enough to meet initial needs. Large numbers of column and truss components may be owned by lease agencies and rented out when needed in order to facilitate flexible stock management.

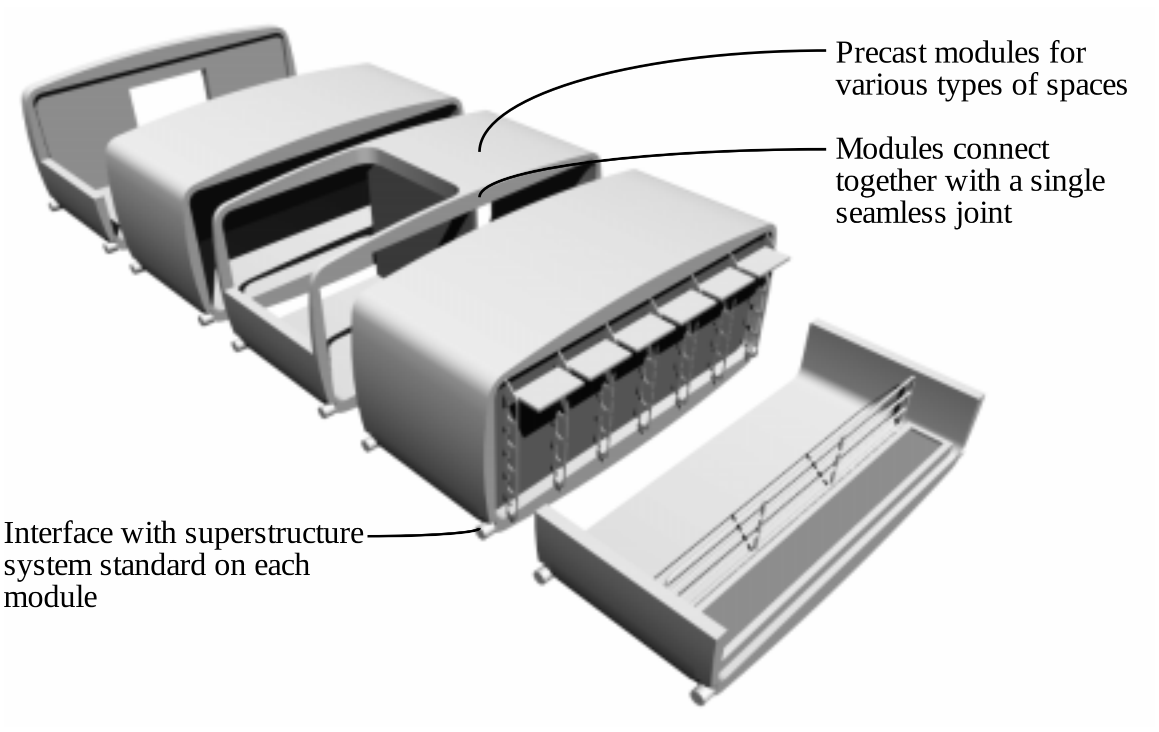

Precast modules for various types of spaces
Modules connect together with a single seamless joint
Interface with superstructure system standard on each module

Figure 85: Concrete modules

In a design environment such as that proposed in VBuild, pre-engineered column and truss components would be assembled virtually by an architect or site manager according to short term projected needs. This would be based on a formula which gives a projected superstructure size per number of dwelling units proposed. Since the building is just one possible configuration of a dynamic system (as opposed to being a static object), the size and scope of the project can change by adding or deleting components from the virtual building.

In this way the support structure becomes an infrastructure much similar to roads and utilities, waiting for infill construction to begin. Contrary to road systems, the support superstructure is fully demountable and recyclable.

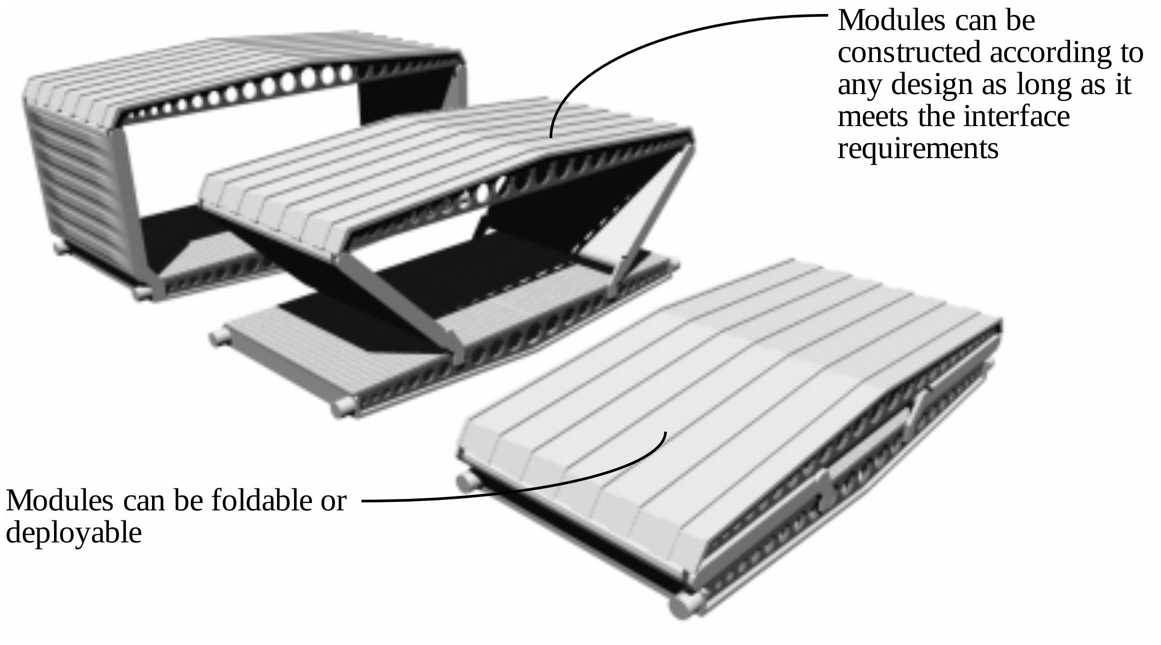

Modules can be constructed according to any design as long as it meets the interface requirements
Modules can be foldable or deployable

Figure 86: Steel modules

7.1.2 Infill design

It may be said that design of the infill occurs in parallel to the design of the support structure, but in fact the two processes are unrelated. Infill design concerns the custom design of single family residences, whether it is to be constructed within the context of the support superstructure or not. This can be accomplished because the infill is designed around the standard virtual module, and a residence would use multiples of the module. In Kajima's version, a module-based kit-of-parts system is implemented, where a single prefabricated module would fill the shape of the standard virtual module.

In infill design, it is conceived that the modules are empty shells having various configurations, with further interfaces that allow for the possibilities of interior plug-in furniture and partition systems. In VBuild the modules would fall into a hybrid structure / cladding system, where further interior components would fall into the various systems (mechanical, etc.) for which they are designed.

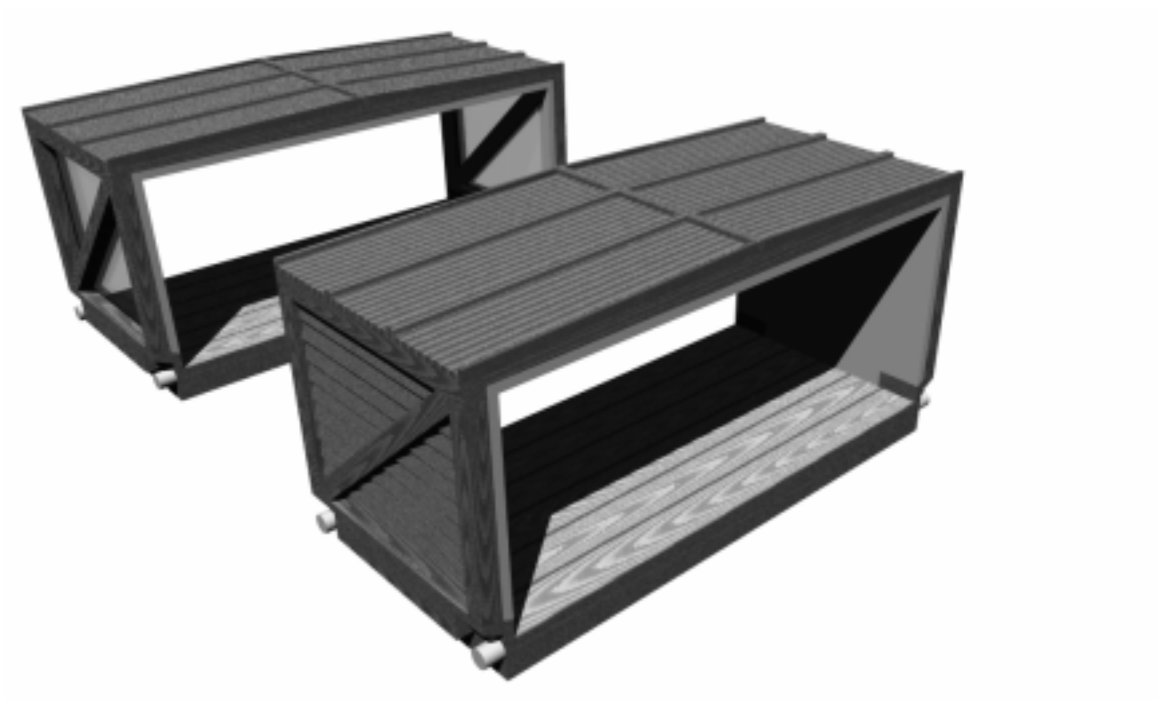


Figure 87: Wood frame modules

It is conceived that the infill design task could be done not only by architects, but by the potential home owners and users themselves, through the use of expert design aids. Since the various modules and interior infill systems are pre-engineered and pre-designed, rules for their assembly can be worked out in advanced and assembly by lay persons can be carried out according to shape grammars derived for that purpose.

7.2 Fabrication and construction process

The fabrication and construction of the support and infill portions would also occur independently. Since the support structure could be considered an infrastructure, its construction would happen in advance of any infill installation, and would have a longer life than the infill.

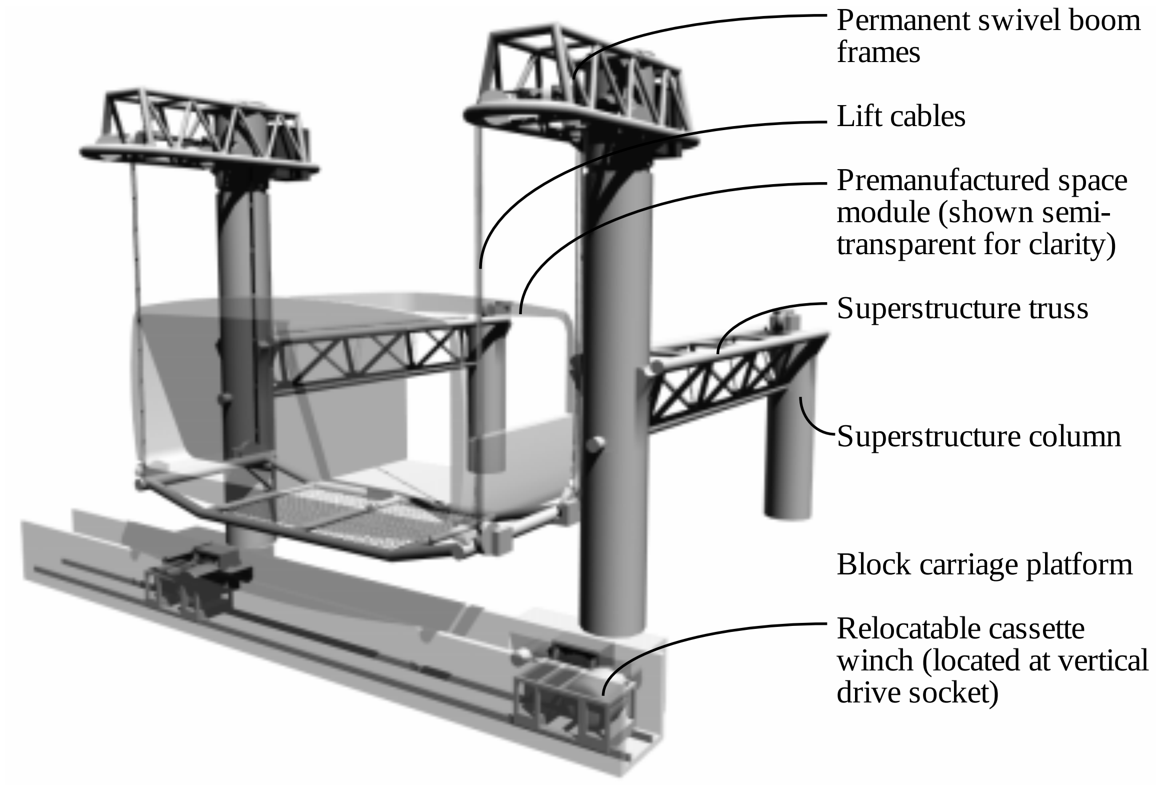

Permanent swivel boom frames
Lift cables
Premanufactured space module (shown semi-transparent for clarity)
Superstructure truss
Superstructure column
Block carriage platform
Relocatable cassette winch (located at vertical drive socket)

Figure 88: Vertical positioning of module

7.2.1 Support construction

Fabrication and construction of the support superstructure would begin by excavation and construction of the foundation system. In the current concept, the foundation "sockets" would be prepared in the conventional manner. Using VBuild, the architect would choose the "build" environment and order the fabrication of column and truss components. In the current concept, material handling would not be implemented between the factory and the site.

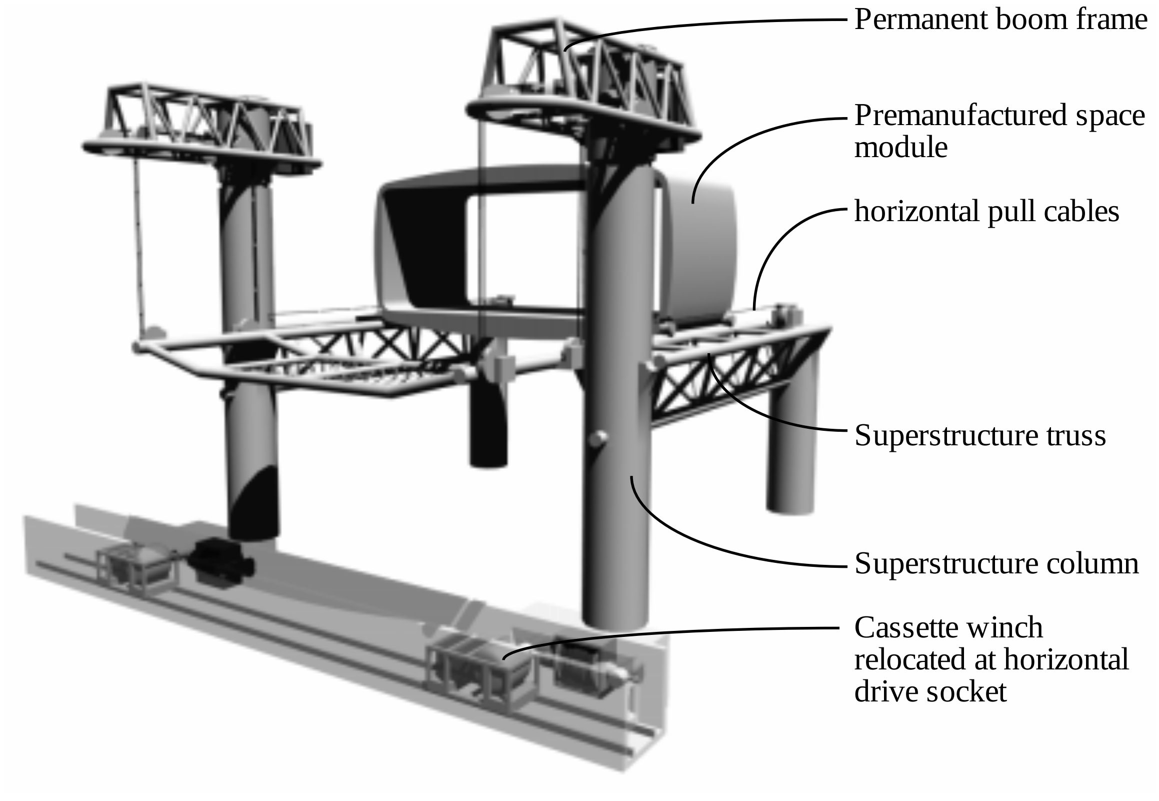


Figure 89: Horizontal positioning of module

Instead, building components and construction machines would be delivered in the conventional manner. The contractor would bring support frame construction robots to the site and deploy them over the waiting foundation sockets, initiating calibration sequences in order to match virtual foundation locations with real ones. Column and truss components would be delivered to the site in their proper sequence, and assembled according to the virtual building created during the design phase.

In the current version of VBuild, construction is conceptually initiated via a button on the control panel. As has been discussed earlier, such a simplistic control over the construction of an entire building would not be feasible unless the entire process was completely automated, robust, and reliably safe. In this project portions of the process are not automated, so control would need to be initiated on site. Once the assembly of the support structure has been completed, the construction machines would need to be removed from the site manually. Any subsequent support structure expansion (or reduction in size) would also require delivery and removal of the machines.

7.2.2 Infill fabrication

Once the future home owner has finished with the design, an environment such as VBuild may be used to order the fabrication of the modules. Two scenarios present themselves at this point. In the first scenario, modules are assembled in a factory environment via automated means, with interior partitions and furniture modules installed in place. In the second scenario, the module is fabricated as in the first scenario, but the interior partitions are stacked or otherwise stored in the unfinished module at time of delivery to the site, or left out all together to allow the homeowner to work out the arrangements later.

Again, since delivery to the site is a manual process, simply pushing a button on a VBuild control panel would not be appropriate. Once the modules have been delivered to the site however, assembly robots have been devised which would plug the modules into the support structure using fully automated means.

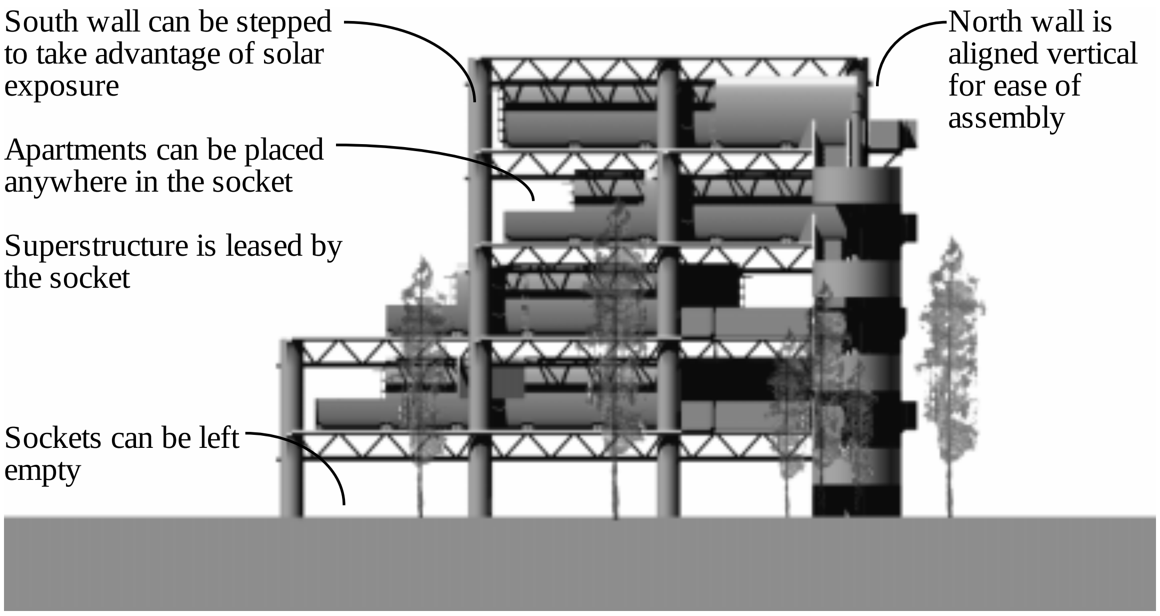

South wall can be stepped to take advantage of solar exposure
Apartments can be placed anywhere in the socket
Superstructure is leased by the socket
Sockets can be left empty
North wall is aligned vertical for ease of assembly

Figure 90: Elevation view of condominiums

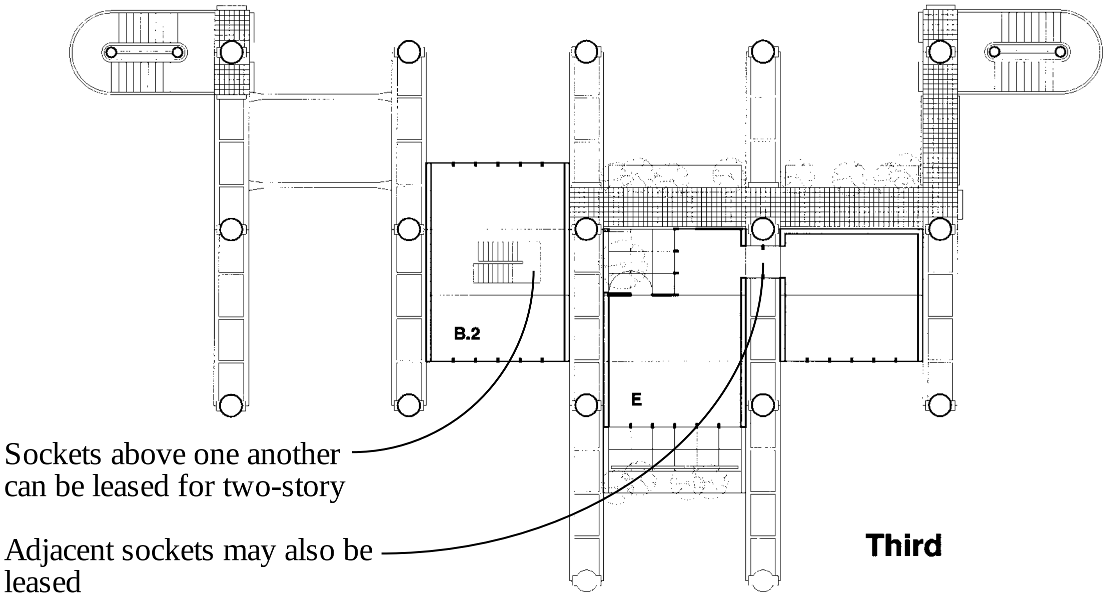

B.2
E
Third
Sockets above one another can be leased for two-story
Adjacent sockets may also be leased

Figure 91: Condominium floor plan

Several concepts for automated placement of the modules have been explored. In each case, the machinery is permanently located on the site to allow addition and removal of modules at any time. One concept calls for the construction of a trench along the length of the building, wherein two portable cassette winches move back and forth as needed. Initially the winches are placed at the two columns framing the bay in which the module is to be installed.

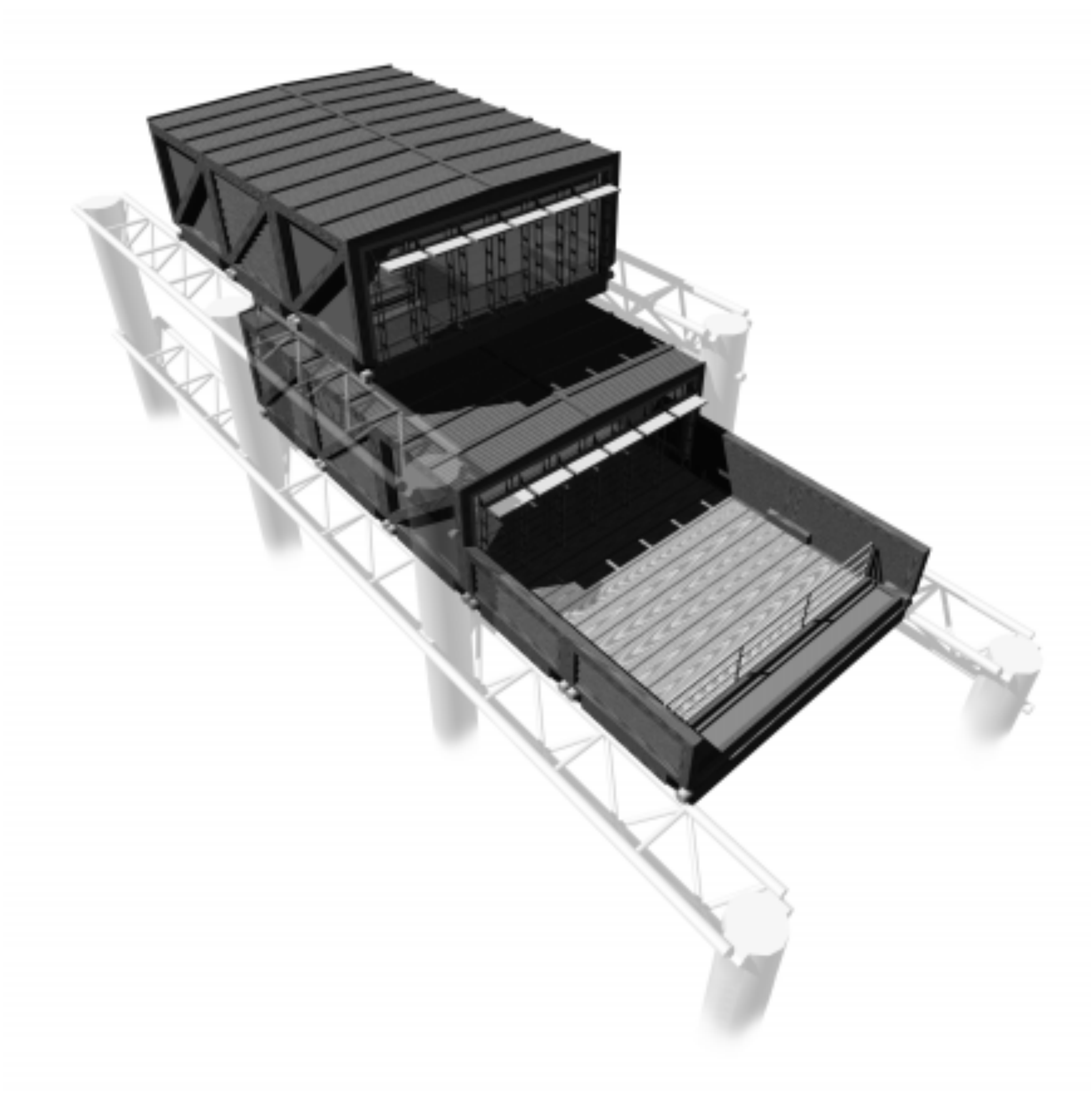


Figure 92: The virtual residence

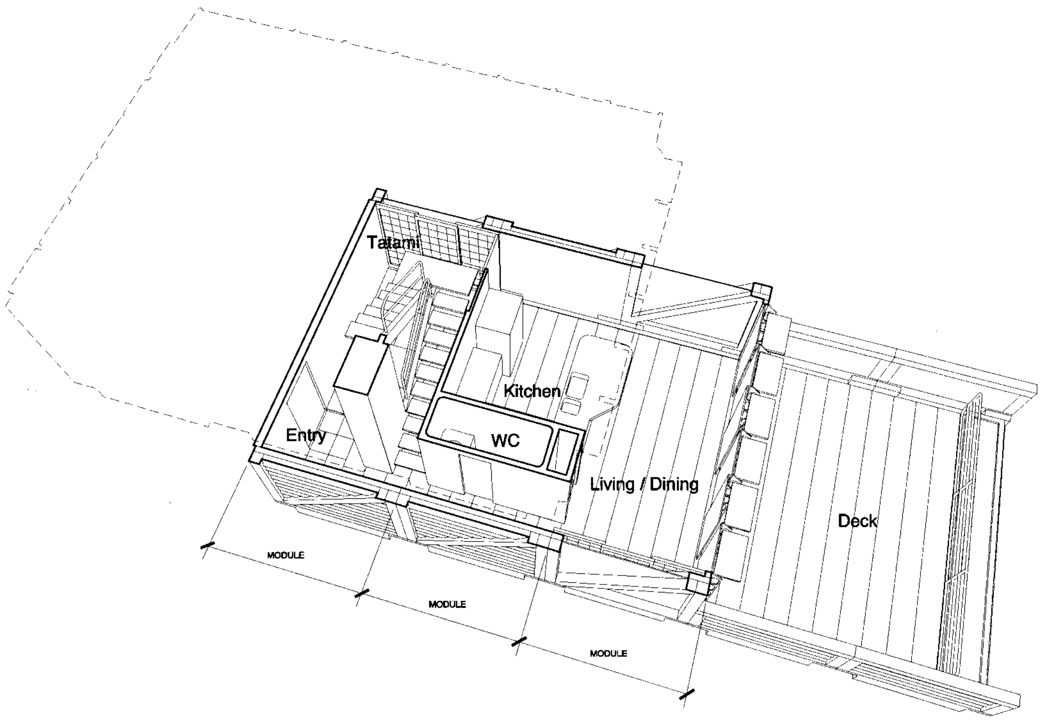


Figure 93: Residence first floor

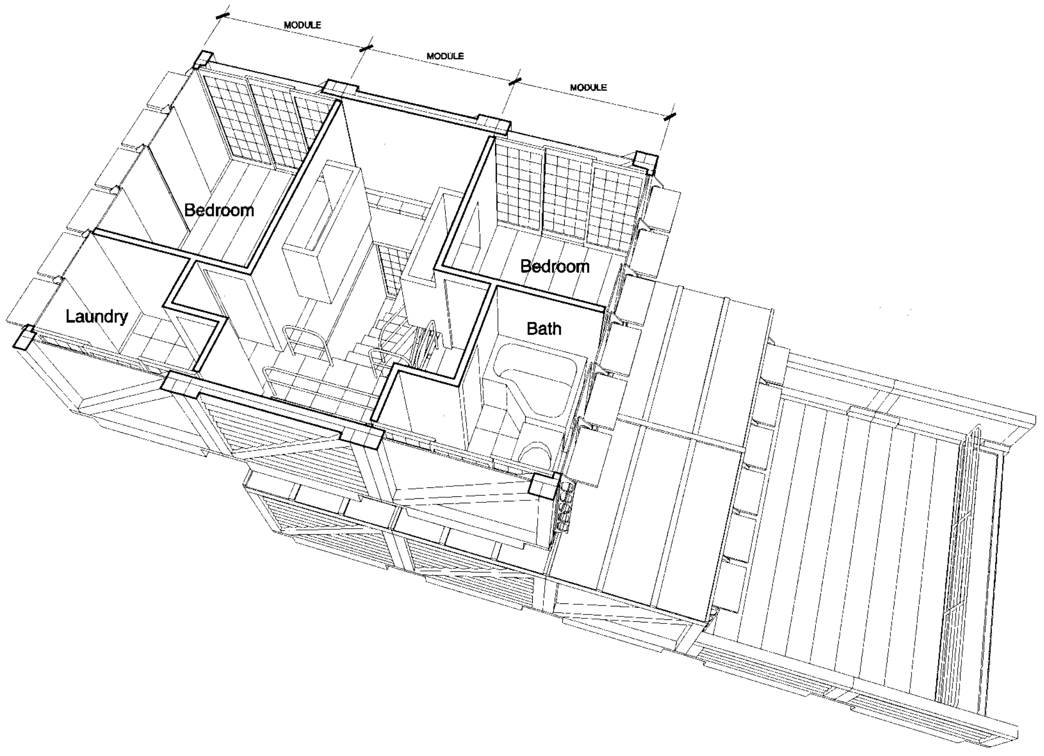


Figure 94: Residence second floor

In these locations, vertically oriented windlass mechanisms plug into the cassette winches, and cables strung through overhead booms are employed to lift a staging platform upon which the modules rest. When the module reaches the desired floor level, the winches are moved over to plug into horizontally oriented windlass mechanisms, and the module is moved sideways into place. This process would be repeated until all the modules have been installed.

7.3 Facility management and use

It is conceived that each dwelling would have its own independent server connected to the Internet. A facility-wide server would hold the entire virtual building with simple bounding-box representations of each apartment. Using the World Wide Web, VBuild components would be downloaded from their source in VRML format, and assembled in real time on a visitor's browser each time the building is viewed over the Internet.

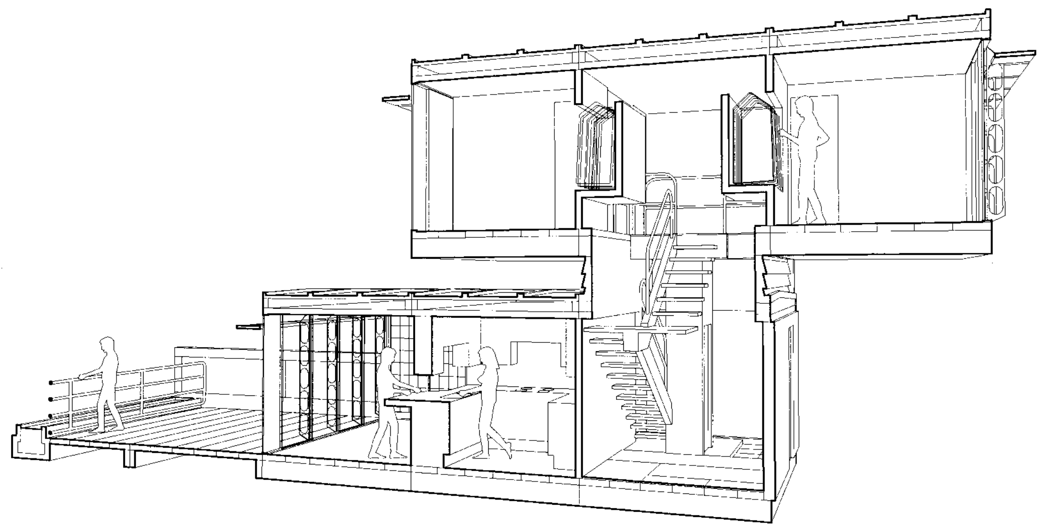


Figure 95: Residence section

Clicking on each of the dwellings would follow the link to the virtual building held within the server of that dwelling, and would show a VRML model of it on the visitor's browser, in the form of a facility home web page. The homeowner who has access privileges would be able to use VBuild or another browser to monitor the condition of the apartment, or control various appliances in the apartment from remote locations. A visitor would be able to view limited information, and perhaps have access to some sort of mail, videoconferencing or chat service to communicate with the occupants.

From the use environment of VBuild, the homeowner could go back to the design environment and affect expansion, or move the residence to another location in anticipation of a move. In this way a homeowner could move between any support superstructures that have been designed to interface with the standard virtual module.

7.4 Summary

The Plug-in Apartment complex is a demonstration project that represents a possible answer to the growing interest for standard machine / building interface design, to speed the erection of buildings and provide low-cost building solutions. Using a hybrid joint and module-based kit-of-parts system designed around a standard virtual module, a double industry which encourages the development of support and infill systems independent of each other can be achieved. Using kit-of-parts concepts, automated construction processes can more easily be incorporated into the building process. Using a network-based application such as VBuild, occupants of the apartments themselves conceivably have some control over the design and build process, where an environment is available for them to arrange and rearrange their dwelling according to individual changing needs.

CHAPTER 8

CASE STUDY III: APPLICATION TO OTHER FIELDS

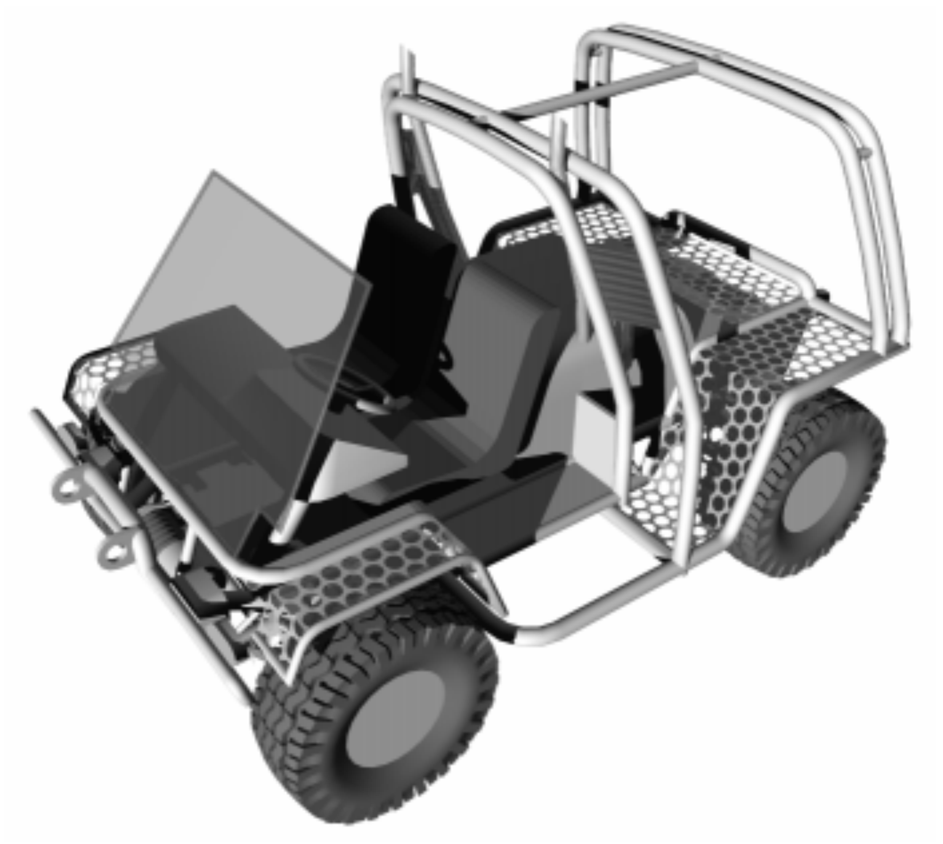


Figure 96: Qamel specialty utility vehicle

The kit-of-parts life-cycle paradigm may apply to other areas besides architecture. During the process of this investigation, two fields besides architecture were studied: naval architecture and automotive design. Though these fields lie outside of the author's specialty, during the course of the study it was determined that the life-cycle kit-of-parts approach may be applicable in certain cases. The investigations began with the concept of design, and how the process occurs in various fields.

8.1 Design process

DEFINITION: Design can be thought of as the conscious creative process or activity by which a person conceives or conceptualizes the nature of an artifact which does not yet exist, for the purpose and possibility of artificially bringing that artifact into existence. Whether or not the artifact is then actually produced may be irrelevant, as long as the development of the idea of its existence proceeds with that possibility in mind. Design may therefore be concerned with forming a virtual likeness or model of a proposed artifact, in order to communicate its nature to others who would help with its actual creation, or to simulate its appearance and / or performance. For comparison, architectural design, naval architecture, aerospace design, and automotive design are discussed.

8.1.1 Architectural design process

The artifacts traditionally created in the field of architecture are buildings and other structures which are used to define and enclose spaces for human activity. Buildings and can be permanent or temporary, and are usually fixed in one location on dry land. Buildings are generally one-of-a-kind and unique, due partly because of unique requirements arising from the human activity to be contained within them, partly because of the unique locations where they are to be built, and partly because of the unique artistic expression of the architects or designers who conceived them.

The above definition for design applies to architecture with a few extensions. Architects usually think of design as an artistic activity performed for the purpose of generating aesthetic and functional solutions to problems defined by functional space adjacency requirements and building laws. Some aspects of the derived solutions are occasionally qualified by calculation, but as a whole the design activity is artistic and intuitive in nature. An architect generally starts the design process by establishing an overall binding concept considered appropriate for the functions and human activities to be contained within the building. The concept is an embodiment of an idea or theme which acts as a structure upon which the designer can base decisions. One example of a concept for a proposed terminal building at an airport could be "airplane", where icons associated with the aircraft industry could be borrowed and incorporated into the aesthetics of the building. The architect may choose to take the "airplane" concept all the way to the point of making the terminal building appear to be an airplane itself (or suggest an airplane by its shape), and can base decisions on anything from lighting fixtures to seating on the concept. An example of this is Renzo Piano's linear-shaped Osaka air terminal building whose entire length has the distinct shape of a airplane wing.

In architecture, the term "structural design" or "environmental design" also have connotations of artistic creativity more than numerical calculation. The physical appearance of the structure, conceived intuitively from an underlying knowledge of basic structural principles (such as triangulation and load path vectors), can be aesthetically designed based on the design concept. This artistic activity may be the first thing an architect thinks of when seeing the term "structural design". To actually prove whether the structure will stand up, architects perform a "structural analysis" of the building themselves or hire a structural engineer to do it. In architecture it is the structural analysis which uses intense numerical calculation, rather than structural design.

Buildings are first designed spatially, with the function or human activity in mind, always considering how spaces will be perceived and used by the occupants. Structural systems, mechanical systems, and other engineered aspects of buildings are generally considered secondary behind the design of the spaces.

8.1.2 Naval architecture and marine engineering design process

In naval architecture and marine engineering, the proposed artifacts are self-propelled vessels and structures which float on liquid media. Different from architecture, the vessels are not conceived primarily for the purpose of containing human activity but for transporting humans or cargo from one location to another. Floating structures usually perform some sort of function not necessarily directly related to human activity, such as drilling for oil. Marine vessels and structures are not generally tied down to one specific location but can be relocated according to expediency (especially vessels which are self-propelled, whose purpose involves mobility). Like architecture, large vessels and structures may be one-of-a-kind and unique due to the scale of the vessel construction undertaking and prohibitive costs. Even in the case where large ships are ordered in numbers, rarely are any two actually built exactly the same. Small vessels on the other hand can be mass-produced similar to automobiles.

In the author's observations, the above definition of design also holds in naval architecture and marine engineering fields, but the perception of what design entails differs somewhat from that of architecture. Naval architects consider design to be a numerical calculation process which literally starts with numbers and rarely touches upon aesthetic considerations. The design process seeks to create a vessel which performs according to some specification. For example, a naval architect may start with a certain tonnage of cargo, the vessel's proposed route, and a proposed voyage duration. From these three numbers, required cargo space, complement and crew, engine size, and ship length can be estimated. The design process proceeds in a spiral of estimation and calculation which gets tighter and tighter until the numbers converge (so to speak) and are finalized.

Naval architects consider each system or aspect in parallel. The systems include power, cargo capacity, weight distribution, buoyancy, and cost among others. The design of the superstructure and deckhouses where the human complement will be housed is often considered secondary, and sometimes is hired out to building architects.

8.1.3 Aerospace design process

In aerospace design, the artifact to be created is a craft which functions in the atmosphere or the vacuum of space. Since the author’s own experience is limited to participation in a spacecraft design project, this discussion will only cover that subdiscipline. Spacecraft are rarely designed to carry human occupants. Spacecraft are usually one-of-a-kind and unique, built for a single mission or purpose. Since spacecraft are required to operate in extreme environments with severe design constraints, the costs are quite high and prohibitive. As with ships and maritime structures, spacecraft are not fixed in a specific location but must be mobile in order to fulfill their functions. Spacecraft can generally be thought of as highly function-specific instrument packages surrounded by control, propulsion, and communications systems.

The above definition of design holds well in the aerospace engineering discipline. As with naval architecture, aerospace engineers think of the term "design" as referring to number crunching and calculation. This especially being true since the spacecraft after being built has no need for aesthetic design at all since it may never be seen by a human again.

According to the author’s observations, spacecraft design is greatly influenced by sizes and capacities of available launch vehicles. Regardless of the mission, the payload capacity of the launch vehicle determines stowed dimensions and total overall mass of the spacecraft. The total overall mass is the most important factor in the spacecraft design problem. The available mass is budgeted out to various subsystems such as propulsion, power, communications, bus structure, and mission-specific scientific instrumentation.

As in naval architecture, a gradually constricting spiral of mass allotment / reallotment, estimating, and calculation continues until all the right numbers are finally available.

8.1.4 Automotive design process

The artifacts traditionally created in automotive design are mobile vehicles which generally have a human operator and serve to transport people or goods from one location to another. Some specialty vehicles differ slightly from this description, where specific work may be performed without transporting something. Vehicles generally are powered and steered through wheels or tracks, but there are some examples that hover or are legged. Vehicles are rarely one-of-a-kind, but are usually mass produced in large numbers.

The above definition of design also applies here. Automotive design is similar in many ways to architecture in that the term "design" does not necessarily equate with numerical calculation. Automotive design often begins with an attempt at defining the needs of a specific type of potential buyer, and then producing a vehicle that will meet those needs. The design process starts with styling, which is entirely an aesthetic exercise. The designers will deal with "emotive" (emotional + motive) concepts while producing flashy drawings of vehicle exteriors and interiors designed to appeal to the potential buyer. Once a concept is decided upon, engineering design begins to run in parallel with the styling process. Since the vehicle will eventually be mass produced, the engineers not only attempt to quantify the concepts developed in the design department, but also design processes for their manufacture. The term "engineering design" comes back to the numerical analysis common in other engineering disciplines. Some major systems or steps in the automobile design process include styling, interior design, steering, suspension, power train, engine, and electrical system among others.

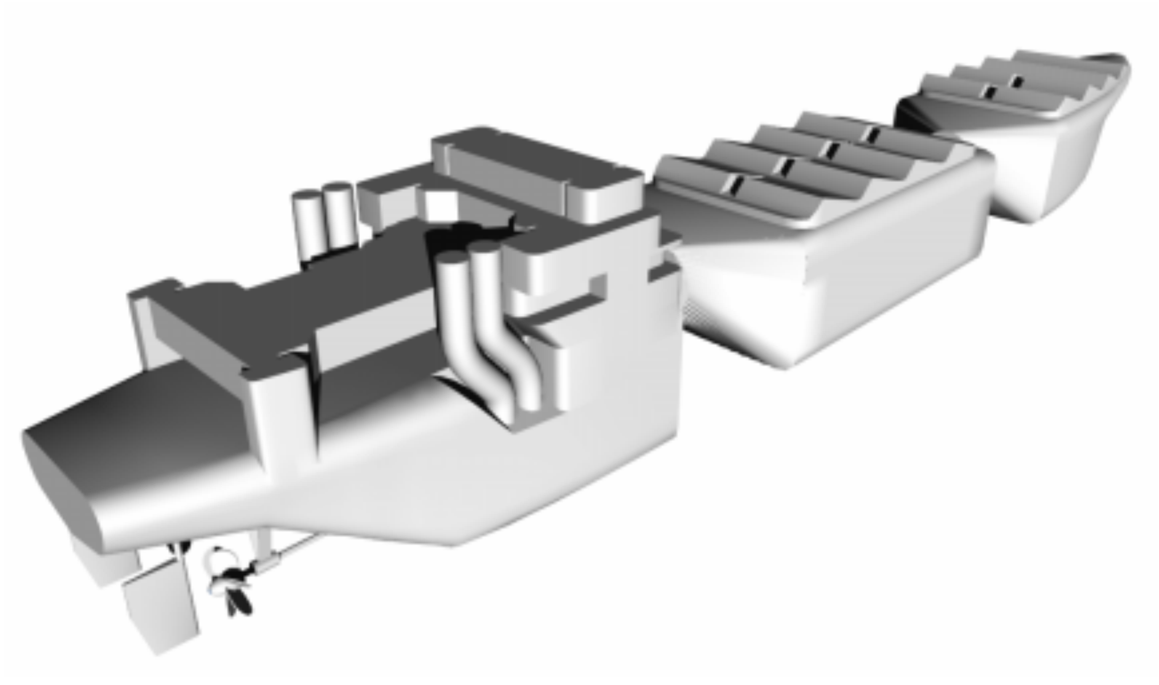


Figure 97: Segmented ship design

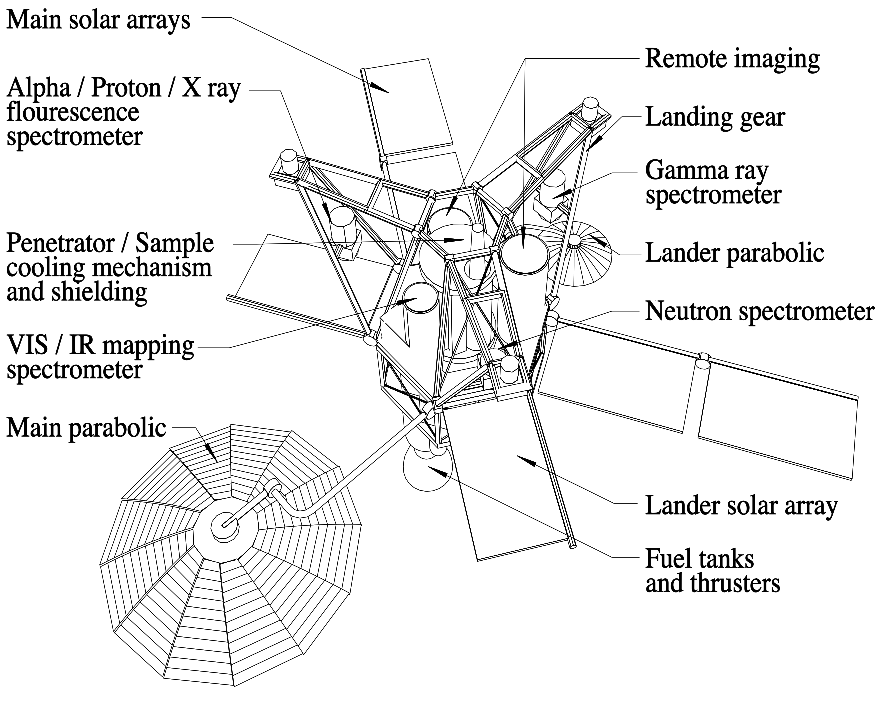


Figure 98: Raven spacecraft design

8.2 Project case studies

During the course of this investigation, the author participated in several design exercises involving projects from other fields. The disciplines studied were naval architecture, aerospace design, and automotive design.

Although several design projects were participated in, only two examples are described which were thought to be relevent to the topic of the kit-of-parts life-cycle paradigm. These are the Ukitecture system and the Qamel specialty utility vehicle. Other projects not discussed in detail include the Segment Ship design (Figure 97), which is a modular plug-together ship system, and the Raven spacecraft (Figure 98).

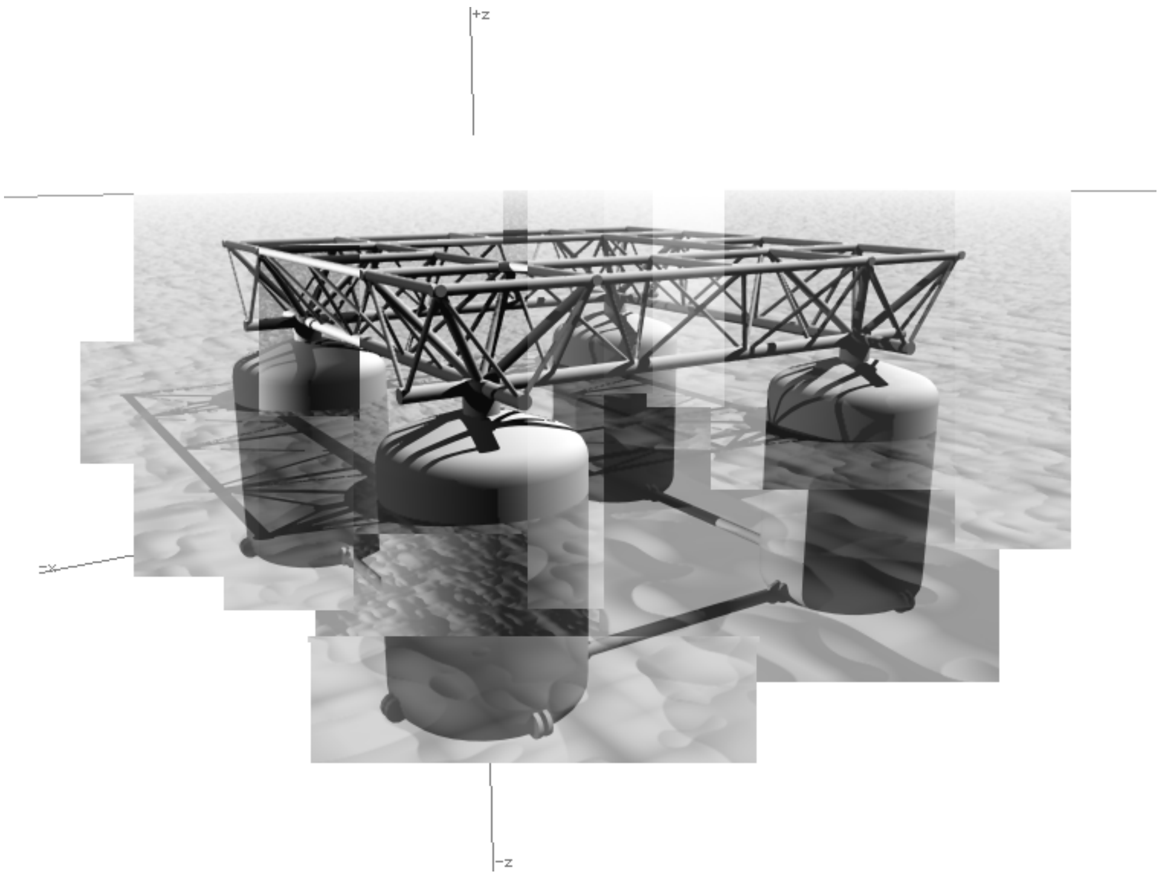


Figure 99: Ukitecture artificial platform

8.2.1 Ukitecture system

The word "Ukitecture" comes from words in two different languages. The first part, "uki-" comes from the Japanese verb meaning to float or be bouyed up. The second part, "-tecture" comes from the English use of the word "architecture". The Ukitecture system is essentially a multi-purpose, floating artificial foundation. In the same respect that pile foundations work on boggy soil, the Ukitecture structure would provide a firm foundation on entirely liquid surfaces such as oceans or lakes. The following is a description of the Ukitecture system, followed by an example architectural project utilizing the system.

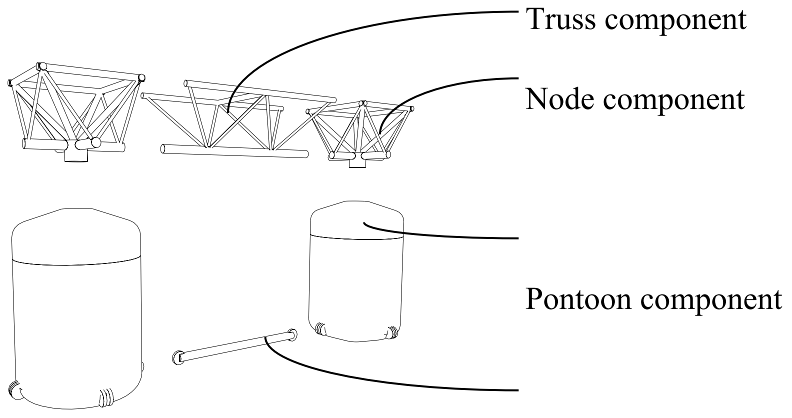


Figure 100: Ukitecture platform components

The Ukitecture system consists of only four simple components that can be assembled into a floating platform of unspecified length and width. The four components are pontoon, node, truss, and brace.

A likely first-time construction scenario would be to assemble three or four pontoons with their connecting nodes, trusses, and braces, and tow the structure to its final location of anchorage. The minimum would be three pontoons to avoid overturning. Additional pontoons can also be assembled in groups of three or four, towed to the site, and connected to the earlier ones as needed. Repairs and maintenance can also be accomplished without major overhauls, by inserting temporary support barges and removing faulty pontoons, trusses, or nodes.

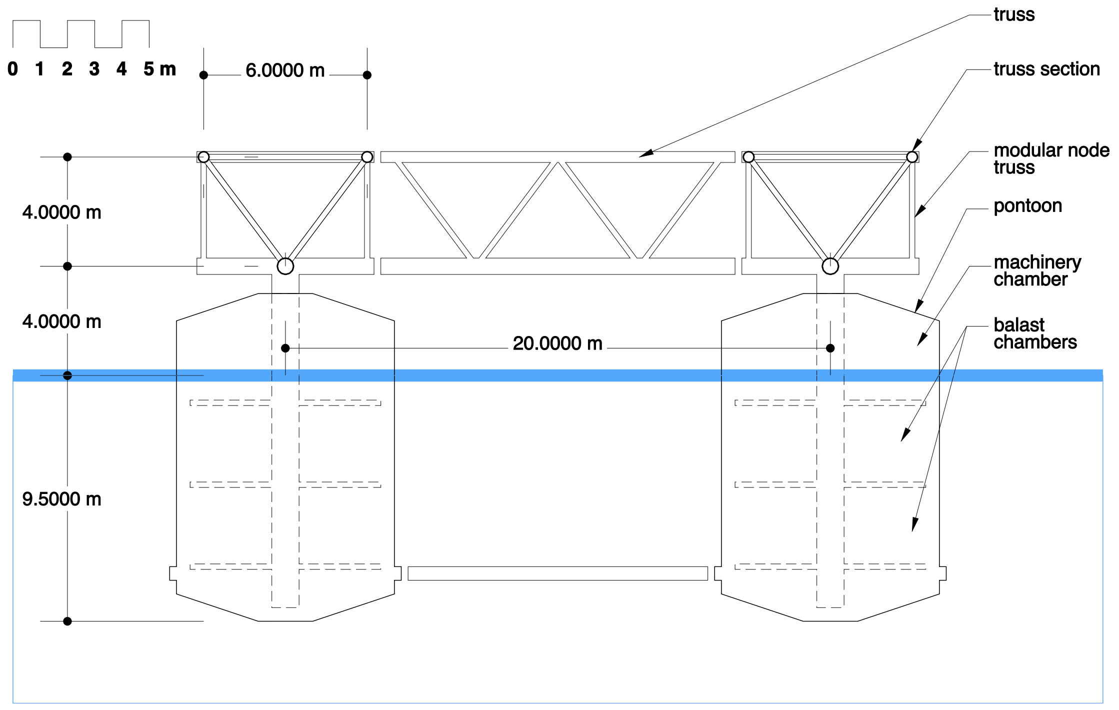


Figure 101: Ukitecture platform section

Each pontoon component would have a machine room on the upper most level, and ballast tanks on the lower levels. The machine room can be fitted with electric generators, bilge pumps, water purification equipment, sewage treatment equipment, water heaters, and other necessary equipment. In addition to plug-in / plug-out structural connections, a generic utility "harness" would be integrated into each component. The utility harness would include power cables, communication wiring, water piping, sewer piping, and other necessary conduit. The harness would be fitted with reversible pumps, transformers, and standard junction points for interface with the construction system intended to be built on the platform. In each pontoon, the harness would have terminations which could be utilized by equipment installed in the machine room, or capped as required. Through a system of valves and bi-directional pumps, access to anywhere else on the platform can be facilitated through adjustment.

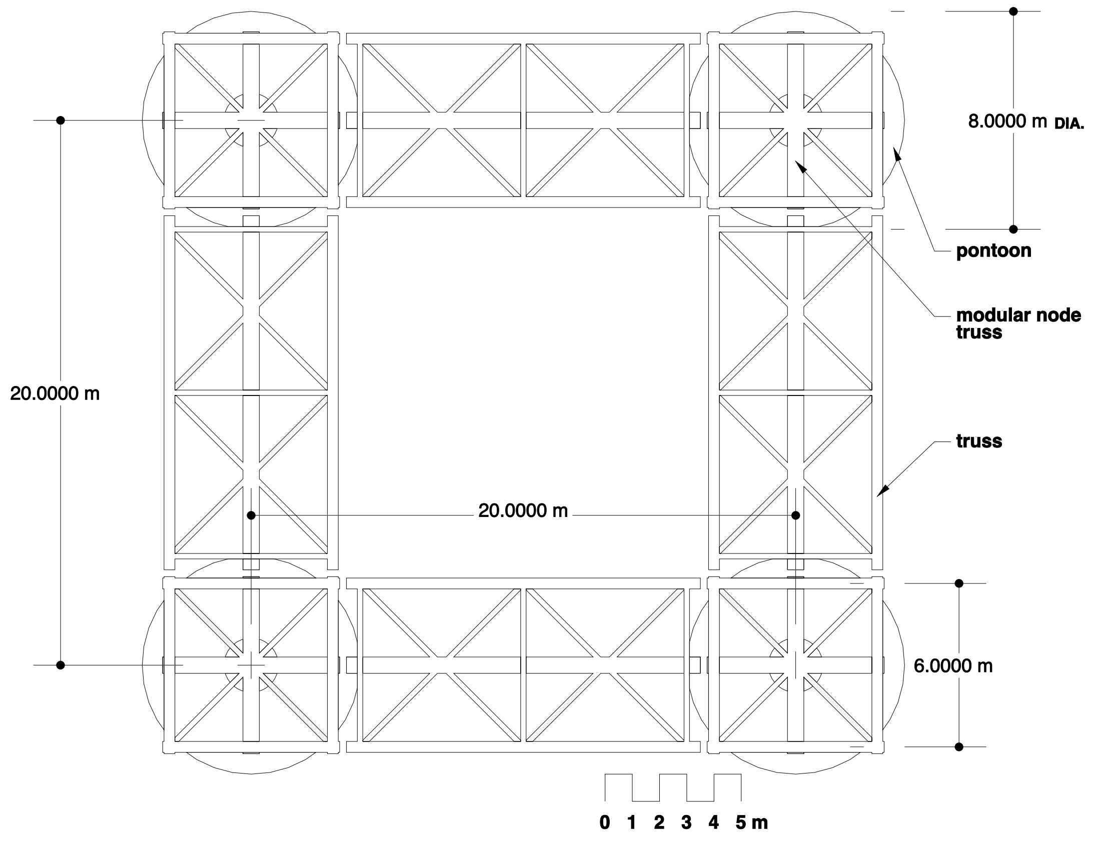


Figure 102: Ukitectutre platform plan

The Ukitecture system is designed on a worse-case scenario for a single pontoon with maximum loading. In the system, the pontoons are placed at 20 meters (78'-9") on center. Half the span of a truss leaves 10 meters on each side of the pontoon, or a total area of 400m^2 (6,241ft^2). This area given certain restrictions to guide the design of the construction system intended to be built on the platform. 400m^2 area was broken up into two zones: a build zone which would include partitions and heavier structure, and a deck zone without partitions. The build zone was situated directly above the truss, coinciding with the 6 meter width of the truss, making a cross-shape.

The remainder of the area supported by the pontoon would be the deck zone. In both cases, the live load was estimated at 100lb/ft^2, and structure load at 50lb/ft^2. The build zone would have additional 30lb/ft^2 structure load. The total load in the build zone was thus estimated to be 673,200lbs, and the deck zone was estimated to be 331,200lbs, for a total of 1,004,400lbs (455 tonnes) on one pontoon. The maximum steel weight of the system above a single pontoon is estimated to be 30 tonnes.

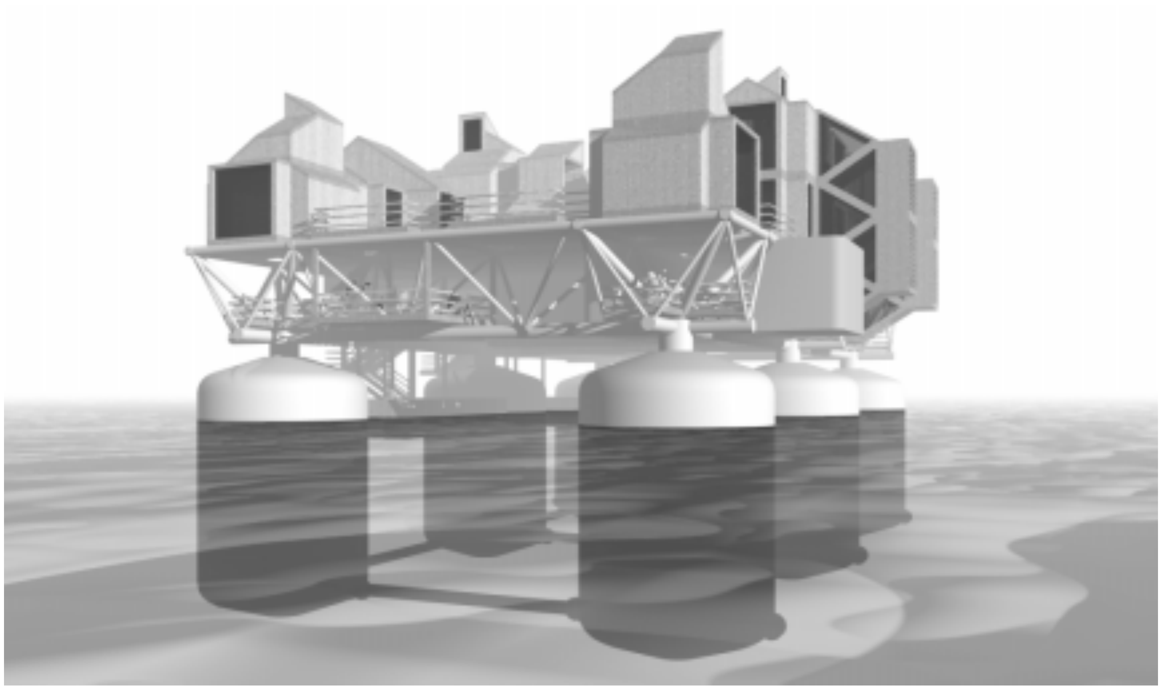


Figure 103: Floating restaurant facility

RESTAURANT FACILITY: Using the Ukitecture system, a plug-in building system was designed for the top of the platform. A restaurant facility is proposed, which would function as a stop on the Sea Bass shuttle boat system in Yokohama Harbor, Japan.

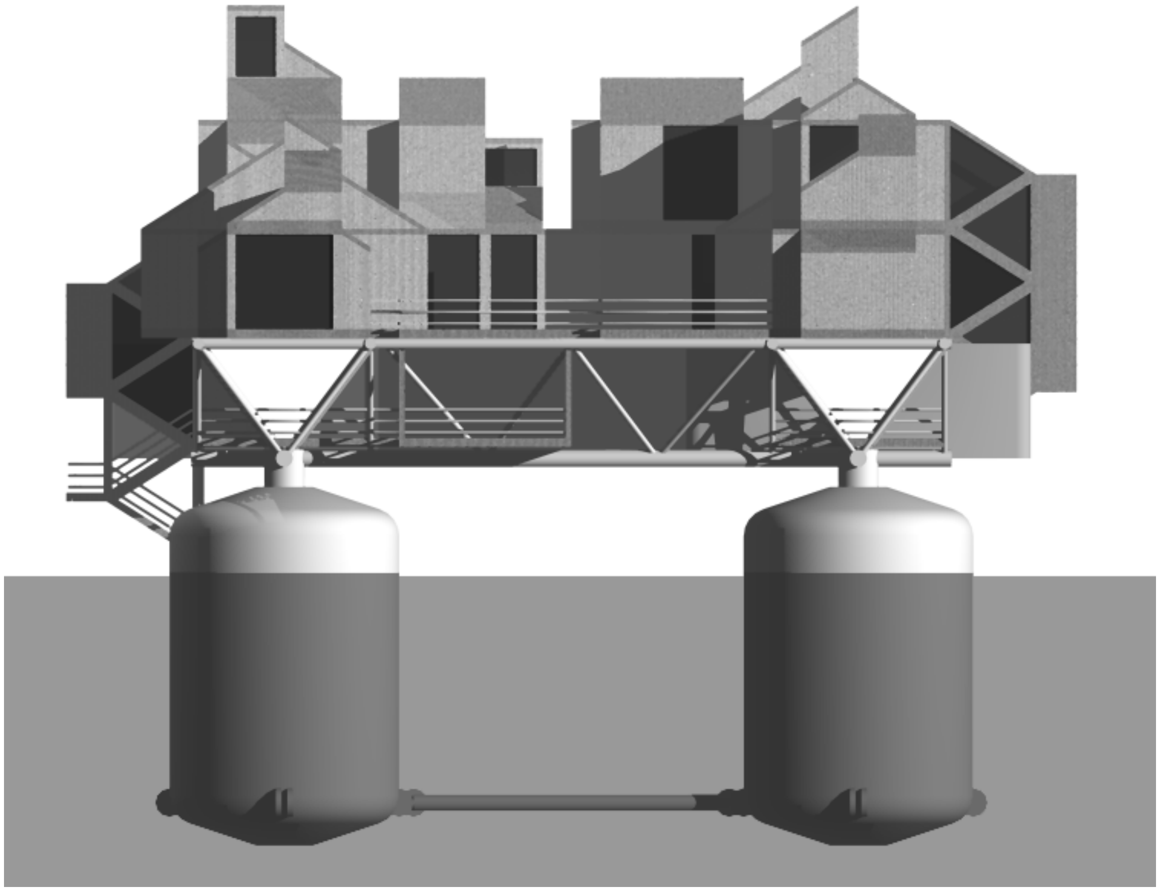


Figure 104: Floating restuarant facility elevation

A series of components were designed as a kit-of-parts building system that can bolt onto the Ukitecture platform, and interface with its utility harness junction points. The components are mainly heavy timber construction partially supported by a steel moment frame. For the design of the kit-of-parts system, a shape grammar was devised which provided a set of rules for space planning, connection, and interface. The shape grammar follows the rules set up by the Ukitecture system, including the use of build zones and deck zones. For space planning, five main components were conceived: two node components (A, B), and three space components (C, D, E). In addition, five roof elements (F, G, H, J, K), two stairway components (L, M), one landing component (N), four deck components (P, Q, S, T), two catwalk components (U, V), and four special wall elements (W, X, Y, Z) were conceived.

A second plug-in modular building system was also conceived which interfaces with both the first kit-of-parts and the Ukitecture system. This system has six special removable modules (sizes up to 6m x 7m) containing complex plumbing or electrical fixtures. These modules are easily removable and accessible for maintenance purposes.

The space components are reversible (R) and have possible variations, which are actually sub-components which plug into the main component. A systematic notation was devised for the space components where a decimal point followed by 0-5 would denote element type (0 = no wall, 1 = solid wall, 2 = window wall, 3 = window wall with a door, 4 = floor opening, 5 = solid floor), and a second digit 2-3 would denote span units. The node components (A, B) are roughly 3m x 4m, and the space components are (C) 3m x 3m, (D) 3m x 4m, and (E) 4m x 5m. Node component (A) has no variations. Node component (B) has four possible variations: one (.02.02), one (.22.22), one (.22.32), and one (.32.32). Space component (C) has eight possible variations: one (.02), one (.12), two (.22)'s, two (.32)'s, one (.40), and one (.50). Space component (D) has seven possible variations: one (.02), one (.22), one (.32), one (.03), one (.13), one (.23), and one (.33). Space component (E) has three variations: one (.13), one (.23), and one (.33). A typical notation for a space component with its attached sub-components could be "C.12.22" or "ER.13", etc.

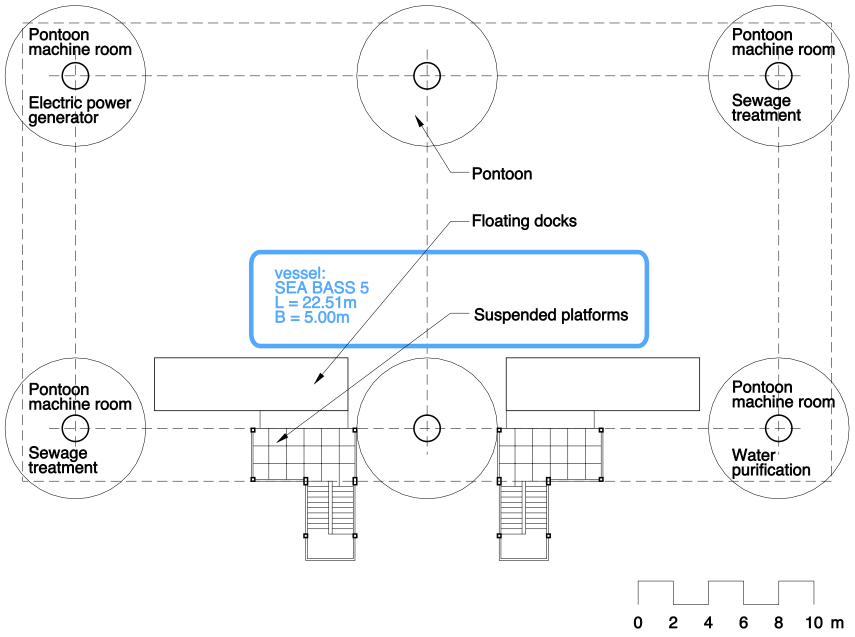


Figure 105: Restaurant facility water level plan

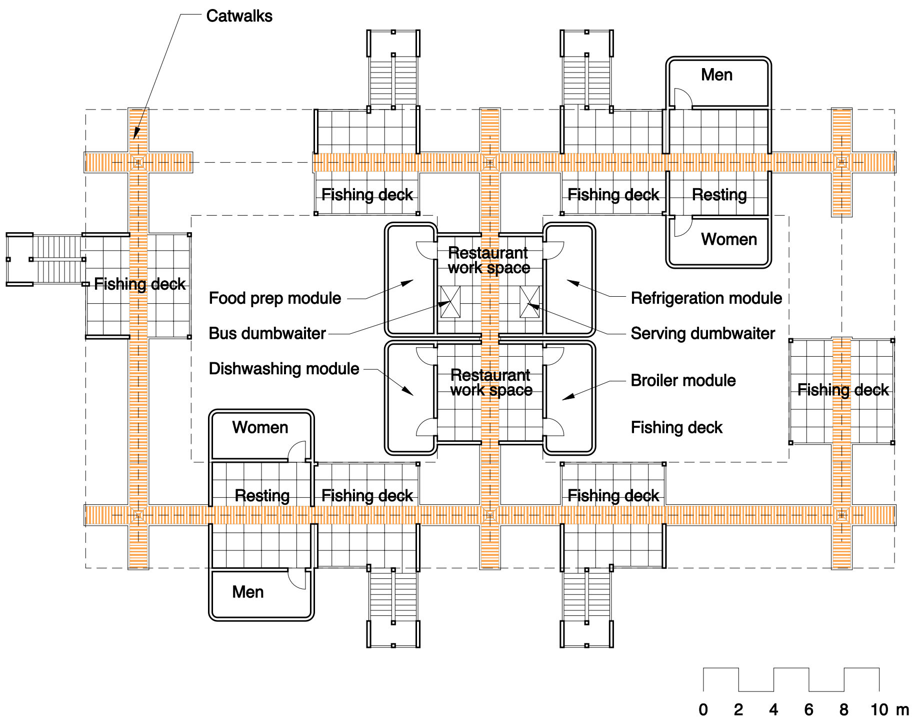


Figure 106: Restaurant facility first floor plan

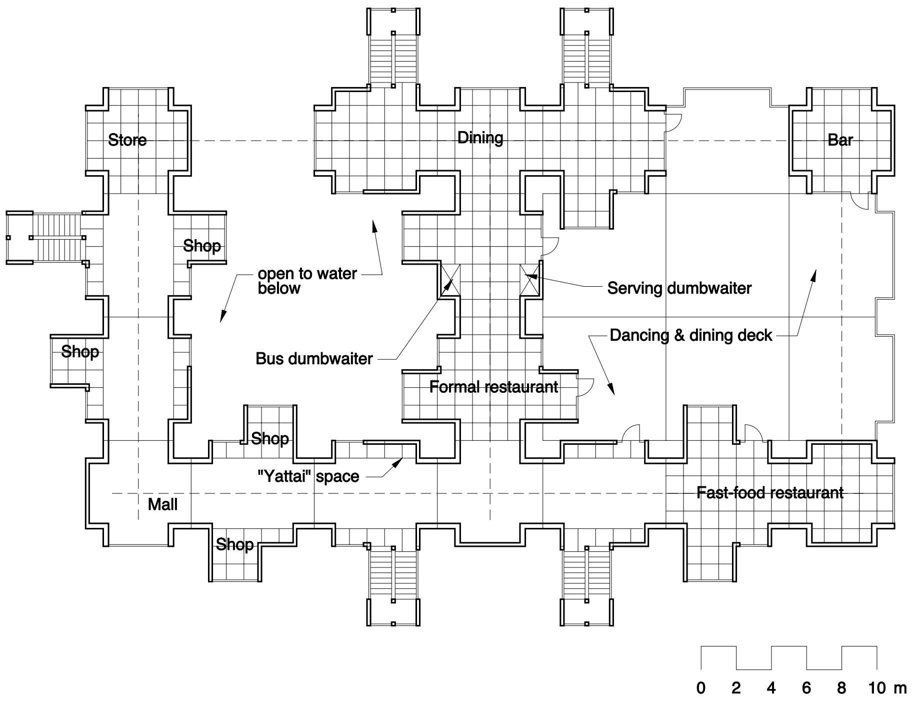


Figure 107: Restaurant facility second floor plan

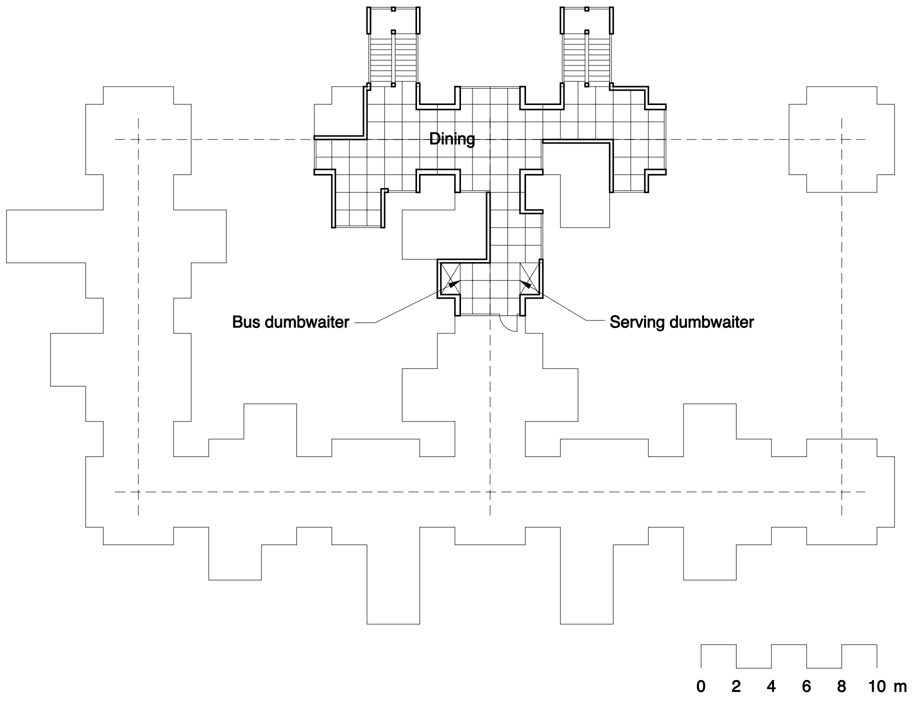


Figure 108: Restaurant facility third floor plan

Of the other components, only the catwalks (U, V) have variations on handrail configurations 6-7, and continue with the same notation as the space components (6 = no rail, 7 = rail). The stairway, landing, two special wall, and one deck component can also be reversed. All in all, this brings the total number of different components (including reversed) in the kit-of-parts system library to only 40 elements.

	SPACE NAME	WATER LEVEL	FIRST FLOOR	SECOND FLOOR	THIRD FLOOR	EXTERIOR SPACES	INTERIOR SPACES
Tenant space	Restaurant						474 m^2
	Kitchen		144 m^2				
	Dining			208 m^2	122 m^2		
	Convenience store			32 m^2			32 m^2
	Shops			44 m^2			44 m^2
	Fast-food restaurant			82 m^2			82 m^2
	Bar			32 m^2			32 m^2
	Yattai space			19 m^2			19 m^2
Restrooms	Restrooms		72 m^2				72 m^2
	Resting lobbies		60 m^2				60 m^2
Circulation	Mall			215 m^2			215 m^2
	Stairways	27 m^2	67.5 m^2	67.5 m^2	27 m^2	27 m^2	162 m^2
	Catwalks		150 m^2			150 m^2	
	Other		75 m^2			75 m^2	
Decks	Dancing & dining deck			297 m^2		297 m^2	
	Fishing decks		105 m^2			105 m^2	
Mechanical & service	Electric generator	48 m^2					48 m^2
	Water purification	48 m^2					48 m^2
	Sewage treatment	96 m^2					96 m^2
	Other	48 m^2					48 m^2
Docks	Floating docks	66 m^2				66 m^2	
	Platforms	36 m^2				36 m^2	
	TOTALS	342 m^2	673.5 m^2	781.5 m^2	149 m^2	756 m^2	797 m^2

Figure 109: Restaurant facility areas

In example, a conceptual restaurant facility is proposed. The facility will include a 474m^2 formal restaurant, an 82m^2 fast-food restaurant, a 32m^2 convenience store, a 32m^2 bar, 44m^2 of small shop space, and 19m^2 of "yattai" space (yattai = a small portable food cart similar to a hot dog or popcorn stand). The facility will also have 297m^2 of deck space for outdoor dining and dancing, and special decks for fishing enthusiasts.

The restaurant facility uses two full bays of the Ukitecture system, which translates into six pontoons, six nodes, seven trusses and seven braces. Four of the pontoons will be used for specialized mechanical equipment, namely electrical power generation, water purification, and sewage treatment.

The Ukitecture system and floating restaurant facility demonstrates the possibilities for the use of kit-of-parts systems in naval architecture. In many ways, the Ukitecture system is similar to the Plug-in Condominium project discussed in an earlier chapter, where the Ukitecture platform becomes a common infrastructure similar to roads in a community. Tenants could design and arrange their own shops using the architectural kit-of-parts according to the shape grammar. Through design, build, and use stages, the kit-of-parts system could be managed along its entire life-cycle, including for renovations and expansions.

8.2.2 Qamel project

The Michigan Integrated Design Initiative recently conducted a multi-disciplinary design project at the University of Michigan. The project generated concepts for a multi-purpose commercial off-road vehicle. The vehicle was named "Qamel", partially in reverence to the strong, swift, load-carrying animal.

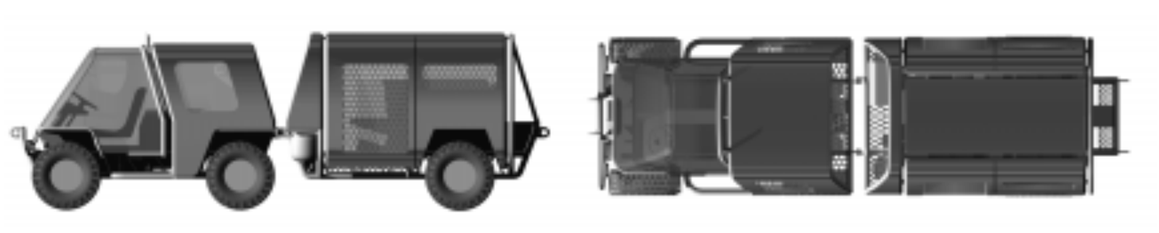


Figure 110: Qamel vehicle top and side views

The goal of the design team was to generate concepts for a modular commercial off-road vehicle which could be outfitted to function as an off-road "motorhome". The problem was to design a vehicle for the United States Park Service or similar organizations that could function as a remote ranger station or scientific research outpost. Requirements included the ability to travel on-road, carry eight persons off-road, and contain living and storage space for four occupants.

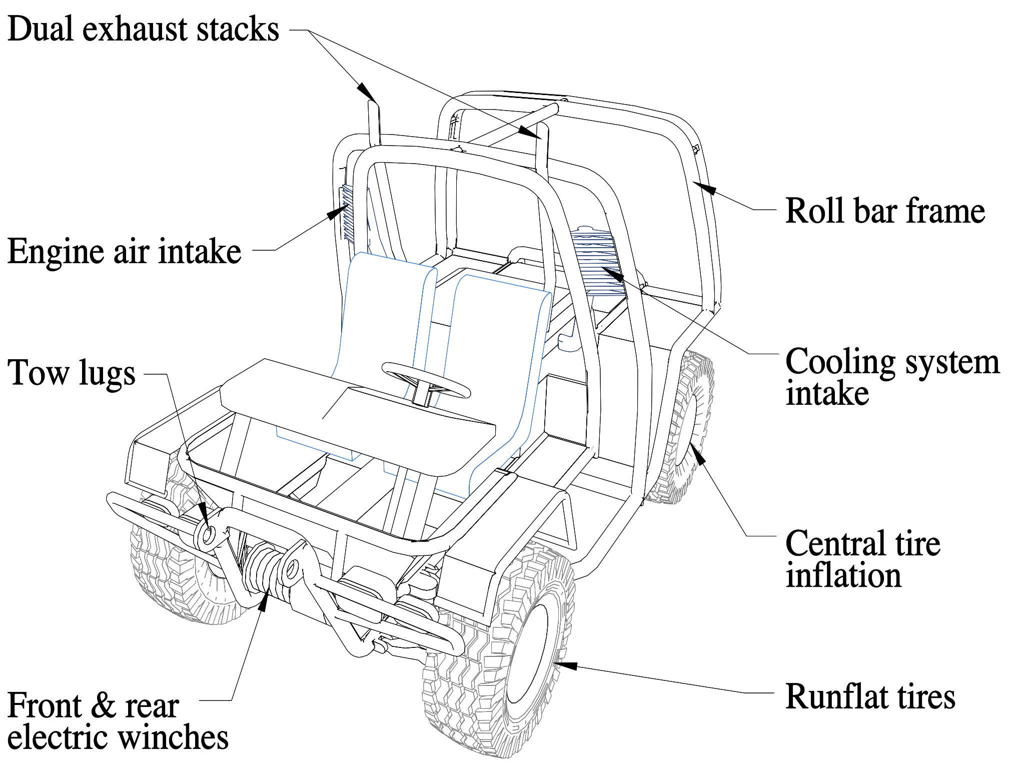


Figure 111: Qamel tractor unit

It was assumed that the vehicle would be used by forest rangers, geologists, or forestry industry personnel for specific projects in remote locations, such as the construction of a hiking trail, observance of a herd of animals, surveying of forests, mineral testing, etc. The length of stay of the outpost was assumed to be one or two weeks. Requirements for the living module included fold-down cots, food preparation area, chemical toilet, and shower. Support equipment for the living module included refrigerator, electric generator, water filtering equipment, field shower (for obtaining high pressure water flow from lakes and streams), and wireless communication equipment.

It was conceived that a standard tractor unit would be coupled with modular special function units specifically designed for various uses. Those uses may include portable living space, mobile hospital, and mobile communications center.

The features which contribute to the Qamel's off-road ability include four-wheel drive, front and rear electric winches, geared hubs, central tire inflation, runflat tires, independent suspension, sealed engine compartment, stack exhaust, and standard snorkel. The Qamel's rear axle is fixed, with steerable front wheels. When mounted, specialty modules would not be considered trailers in the usual sense but would have specially designed coupling mechanisms which actually convert the vehicle into a single hinged body with one degree of freedom in pitch. Upon connection, the module's wheels would be steerable, hydraulically powered to respond to the tractor units steering controls. The module wheels would normally rotate freely, but through the power of an electric motor would provide additional push for all-wheel drive under certain low speed conditions.

The tractor module was conceived to be a removable power plant. When the specialty module, whatever its function, is deployed, the tractor module would provide the rangers with mobility in their work. Using the specialty module as a home base, or as one independently functioning unit among many, the tractor would be free to tow other modules or transport the rangers to more remote locations. For example, in an emergency rescue situation such as a wilderness plane crash or multi-vehicle pileup, mobile hospital modules could be towed to the site and dropped off where needed. Immediately upon uncoupling, the tractor could return to fetch another module or reposition previously placed ones.

On the road, the tractor would legally carry two passengers in its cockpit. Off-road, additional detachable thin seats over the rear wheel wells would accommodate another two passengers on the outside of the vehicle. These exterior seats could facilitate drive-by tree marking, game observation, beach rescue, and other activities more easily conducted outside the envelope of the vehicle. The expanded metal decking utilized over the wheel wells and rear frame would also facilitate equipment lashing and the carrying of cargo. All body panels, including the cockpit enclosure, are snap-on to facilitate multi-purpose usage. In a beach rescue environment, a fold-down plate windshield may be appropriate, where a winter snow environment would call for a fully enclosed cockpit with environmental controls.

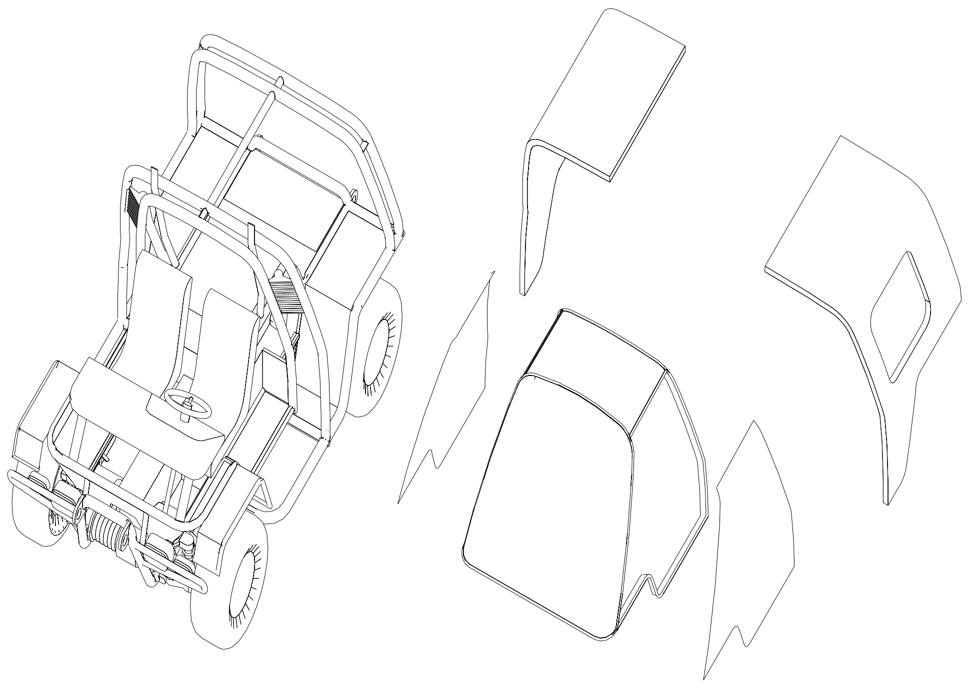


Figure 112: Snap-on body panels

The original concept for the structural system was conceived to be extremely lightweight tubular stainless steel construction. The frames joints are welded, with formed bent tubing. Front and rear tow lugs are welded directly to the frame, with the rear lugs providing a base for the module connector mechanisms to mount. The suspension system is a double wishbone mounted to the top and bottom chords of the tubular frame truss. The wishbone system in the rear mirrors that of the front, with spring and shock absorber mounted to the lower wishbone. The Qamel's approach angle is 67^o, departure angle 75^o, and ramp breakover angle 43^o. In addition, specialty module departure angle is 38^o, and the tractor / module joint has 26^o of play above horizontal and 24^o of play below horizontal. Turning radius with the tractor only is 6 meters, and 7.5 meters when specialty modules are attached.

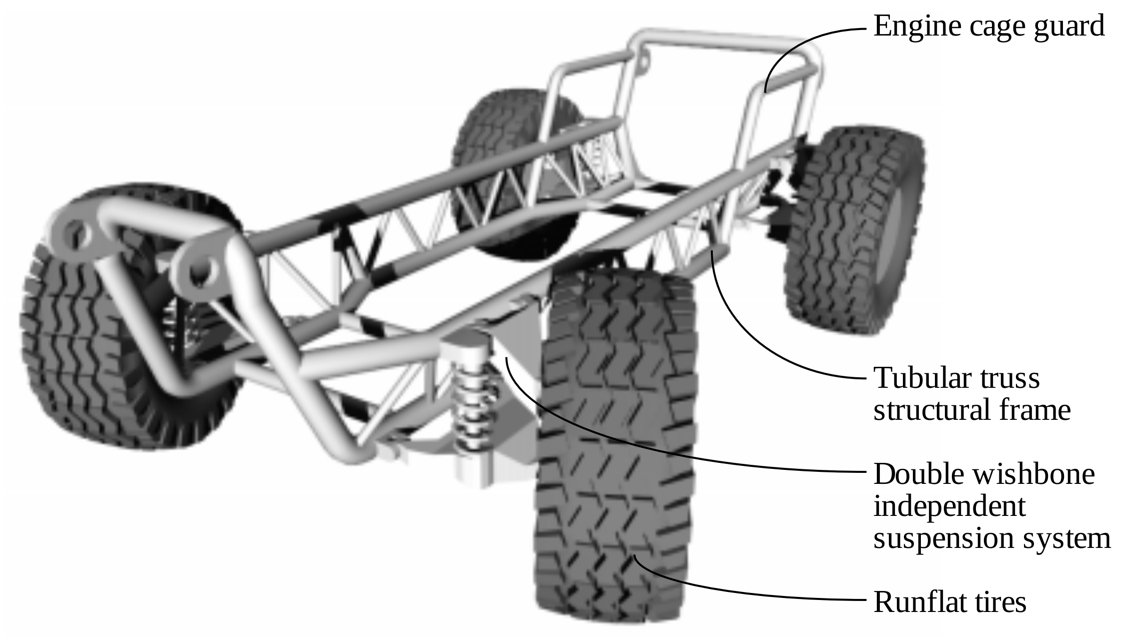

Engine cage guard
Tubular truss structural frame
Double wishbone independent suspension system
Runflat tires

Figure 113: Lightweight structure

The engine is located slightly rear of center. The 170hp diesel fuel injection engine has a fully sealed moisture proof electrical system. In conjunction with the engine air and cooling air snorkel system and stack exhaust system, the Qamel is designed to be able to ford depths of 1.6 meters (5 feet). Qamel has three fuel tanks for a total diesel fuel carrying capacity of 170 liters. The power disk brake system is located inboard for protection and to increase the ground clearance. Qamel is equipped with two electric winches, both in the front and rear.

In the context of this project, a living module was designed. In order to meet the design requirements of having the ability to transport eight persons in an off-road situation, the module of necessity was required to accommodate passengers in transit. The living module was designed to transport up to four passengers in the sitting position, or two passengers laying down.

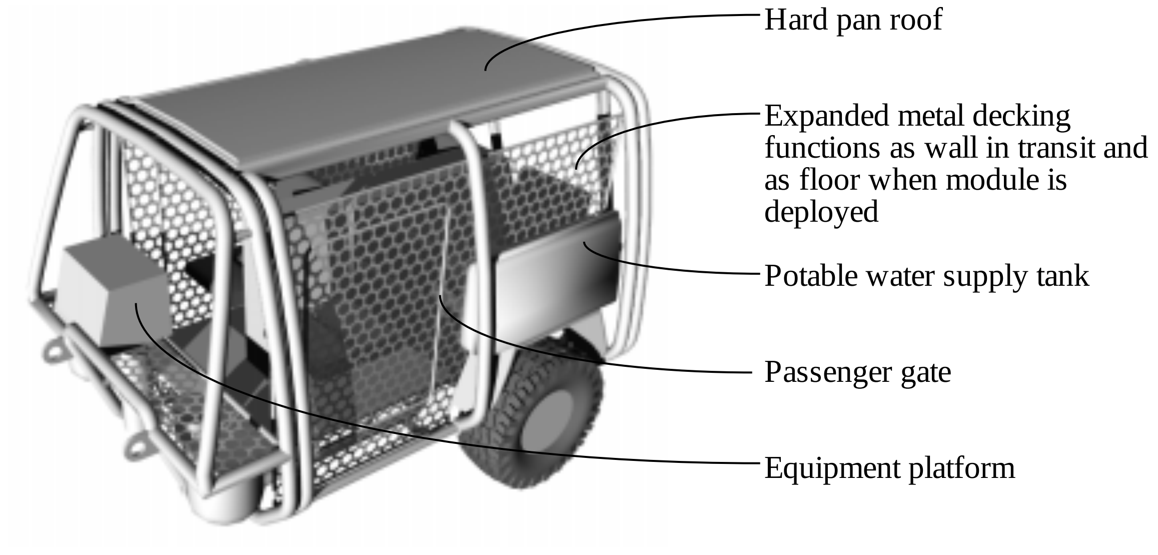

Hard pan roof
Expanded metal decking functions as wall in transit and as floor when module is deployed
Potable water supply tank
Passenger gate
Equipment platform

Figure 114: Specialty module in storage position

The frame is of the same tubular stainless steel construction as the tractor unit, and in its most basic form is conceptually skeletal. As with the tractor, a snap-on paneling system consisting of everything from canvas to thickly insulated panels would allow a flexible use in different climates. In the stowed position, the walls follow the same general profile as the tractor. An expanded metal inner wall would enclose the passengers to a certain extent, but real enclosure would be afforded by the snap on panels which are fastened to the outer tubular frame. In the space between the tubular frame and the expanded metal decking, equipment and supplies could be lashed to the decking. Permanent potable water tanks are installed in that zone.

Passenger seats fold down and double as cots, and also fold up out of the way. A single passenger gate on each side, piercing the expanded metal decking, afford exit for the passengers when the module is in the stowed position. A passenger sitting in the rear seat (which is mounted on top of the wheel well) would fold up the front seat and swing it out of the way to open up an aisle to the passenger gate. Snap-on panels affixed to the frame would allow the passenger gate to function normally.

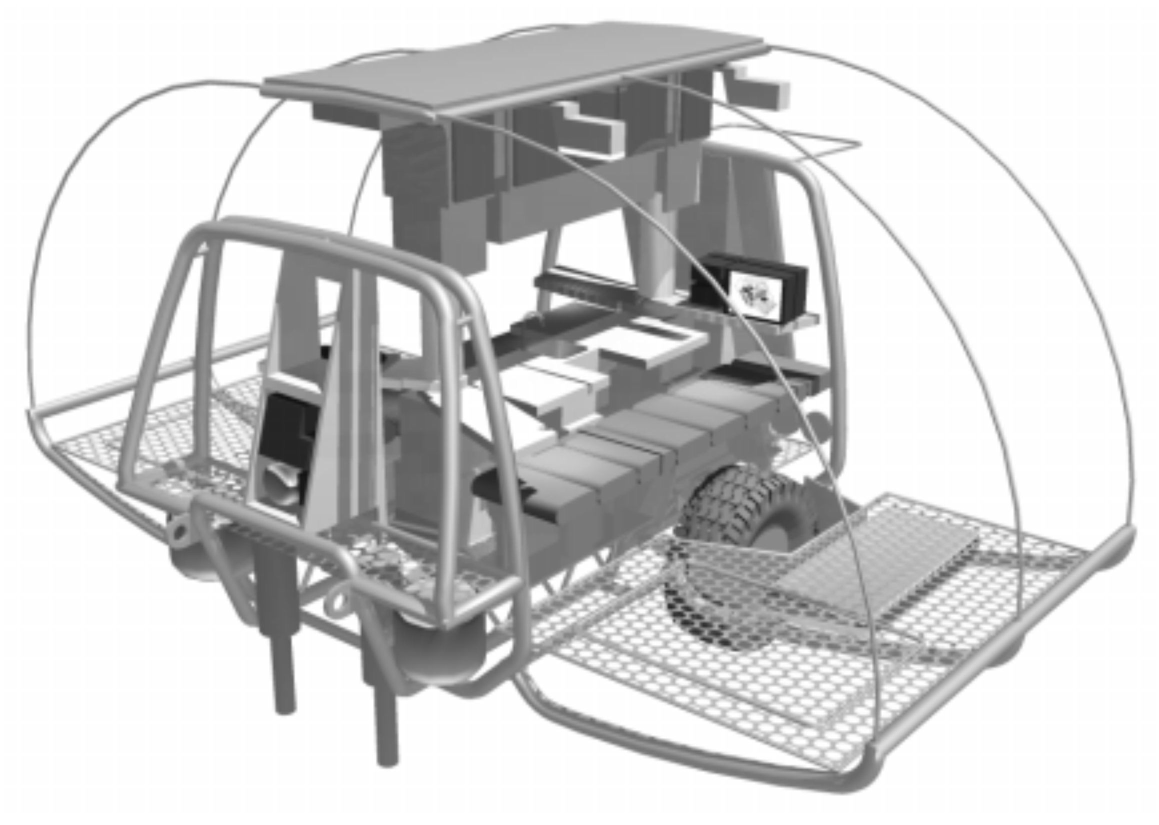


Figure 115: Specialty module deployed

When the living module is deployed, it is detached from the tractor and the front hydraulic jackstands are locked into support position. The side walls which are hinged at the bottom swing out and become horizontal platforms. The expanded metal decking that formed the walls when the unit was in the stowed position becomes the floor decking for the living space. Passenger gates are locked into place, since they too have become floor. Canvas walls, attached to the outer extent of the floor deck and the edge of the hard pan roof, sags out from the tucked-in storage position as the frame is opened. The hard pan roof then extends upward to locking position to provide full-height standing head room for the occupants. After the frame has been locked into place in the deploy position, elastic jointed aluminum "tent poles" are inserted into the outer edges of the floor deck and hard pan roof to support the sagging canvas into a taut membrane. Fold-out adjustable jackstands provide additional support at each of the corners of the floor deck. As with the skeletal frame and snap-on body panels, the bare expanded metal decking can be fitted with zip-in indoor / outdoor carpeting or other appropriate flooring.

Entrance to the module is by zippered doors built into the end panels of the extended canvas membrane. Between the passenger seats running through the center of the module from front to rear, is a food-prep core. The food-prep core is packed with fold-out tables, a refrigerator, fold-out stove, vinyl sink, and cooking utensil storage cabinets. In the direct rear of the module, a fold-out shower platform opens directly to the rear and is accessible from the outside. A chemical toilet is located on the rear of the food-prep core and is actually inside the shower stall when it is fully deployed. Extending a special umbilical consisting of a pair of hoses and an electric wire, the field shower can be put into operation. One of the hoses, affixed with a push pump powered by the electrical wire, is inserted into a lake or stream, and the other hose is placed a certain distance away for channeling away the brown water for leaching (only brown water and not sewage).

In this example, a living module was designed. Other special function modules would generally have the same overall frame, with slight modifications to the interior. A hospital module may have permanent medical equipment storage where the passenger seats are, with fold-down stretchers at the outer edge of the floor deck. A communications module would have parabolas and antennas stowed on one side, which would fold out when the unit is opened up, using the floor deck as a mounting platform and the other side as an enclosed operator booth. A mobile office would have smaller antennas for connecting to the phone system and Internet, and would have desks and shelving for computers and peripherals. Emergency fire fighting headquarters may have lockers for equipment and cots or lounges for tired personnel to take needed breaks.

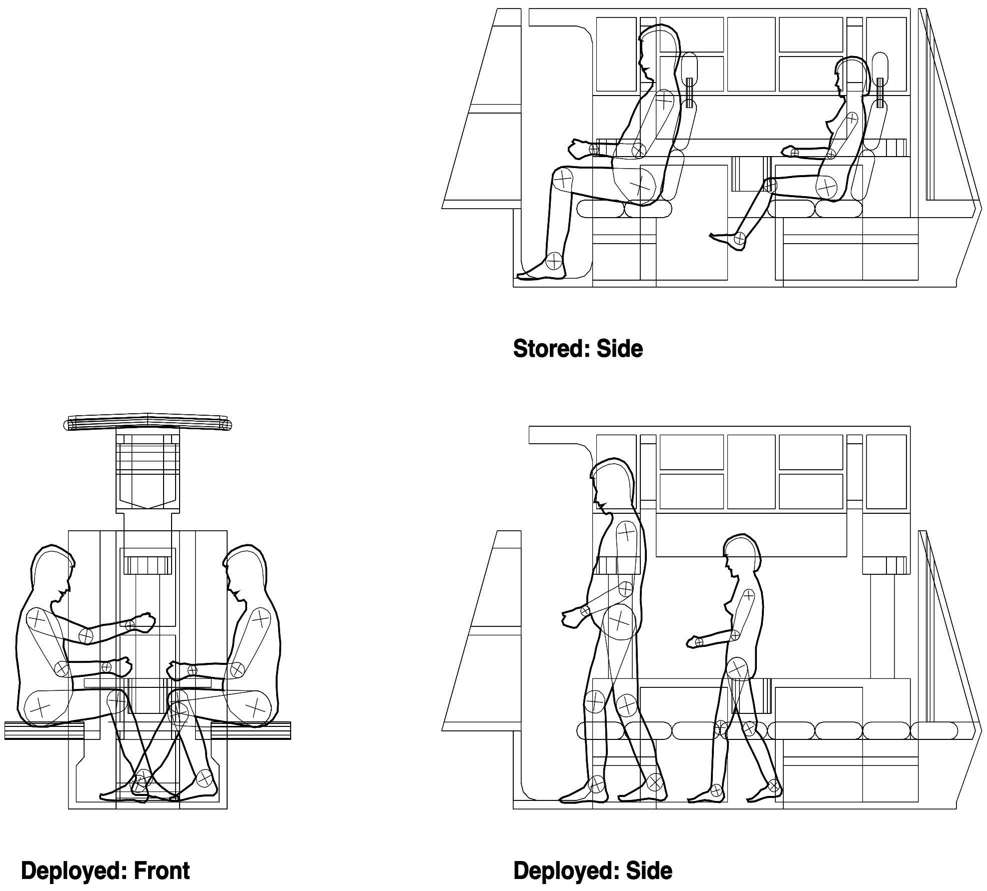


Figure 116: Specialty module core

The Qamel specialty utility vehicle project shows possible applications of the kit-of-parts life-cycle paradigm to automotive design. In the design stage, potential buyers can apply preferential body panel systems while the vehicle only exists virtually. Specialty modules can also be designed by the potential owners. Variations of suspension systems, engines, and other systems could be interchangeable if a vehicle shape grammar were established and the components designed according to the grammar. If a sustained or dial-in network connection could be established, vehicles could also have their own web page in order to affect remote control or monitoring of the various systems.

As a side note, the Qamel vehicle concept was awarded the Audi International Design Award in 1997 and was displayed in various trade shows including the Hanover Trade Fair in Hanover, Germany.

8.3 Summary

In various fields, design processes can include aesthetic investigations as well as quantified analysis. Architecture and industrial design tends to center more on aesthetic design, whereas engineering fields are more number-crunching in nature. However, the basic definition of design is the same in all fields in spite of varied interpretation of terminology and process. The kit-of-parts concepts and grammar-based design practices advocated in this work can be applied to other disciplines besides architecture. A network-based life-cycle management paradigm may be appropriate for naval architecture, automotive design, and other disciplines as well.

CHAPTER 9
RESULTS AND CONCLUSION

With the advent of new technologies in computer networking and information management, a rich “cyberspace” community spanning the globe has evolved. An explosion of computer application and tool development has brought vast amounts of information closer to hand directly from sources in real-time. The sources are usually digital or computer-based, such as carefully prepared databases or feedback generated in real-time from user-input parameters. However, the flowering of cyberspace has very little to do with the real world. When it does, the real world subject is usually disjoint from the cyber counterpart, requiring manual input to keep the information about the current state of the subject up-to-date. This work investigates methods for linking real-world physical objects and virtual representational counterparts for the purpose of two-way real-time generation of information. In one direction, information about a virtual object can be used to initiate the manufacture or alteration of its real-world counterpart. In the other direction, the current state of the real-world object can be quantified and used to generate and accurate virtual representation. Using this two-way linking concept, this research focused on the establishment of a new paradigm for life-cycle management of real-world physical objects through manipulation of their virtual digital counterparts. A summary of this work, results, and conclusions follows.

9.1 Summary and results

The main thrust of this research was design using computers and networks as the primary tool, where a virtual representation of a designed object becomes an instrument for facilitating manufacture of the real object and observing or altering its condition during use. In order to do this, an object-oriented approach was taken in both the computer and real world environments. Using object-oriented programming and kit-of-parts construction concepts, feasibility for linking the virtual and real objects in the ways described was established.

The study began with a discussion on the use of the kit-of-parts as a design concept. Historical examples of design investigations using kit-of-parts concepts were cited, including *Structuralism*, *Metabolism*, and *Hi-tech* movements in architecture, which celebrate the systematization of space and building components. Design investigations conducted by the author were also described as a foundation of previous research upon which this work rests. The kit-of-parts concept discussion attempts to establish the object-oriented approach, which the concept advocates, as a means for cleanly demonstrating and developing the new paradigm.

After establishing the need for a kit-of-parts approach, various pertinent technologies needed to implement a demonstration of feasibility were reviewed. Since the development of a rich model for virtual representation is important, a literature review and discussion on computer visualization was conducted. In conjunction with the study, an experiment of three-dimensional object representation in a network environment was executed. In order to facilitate the manufacture of an object from a remote location, an extensive study of the state-of-the-art of automated construction was conducted. Including projects the author has personally worked on in the past, many exemplars were cited. A literature review and discussion of direct digital control technology was also conducted. A study of available hardware that could implement remote monitoring and control of devices was made, and an experimental facility web page that can display sensor data dumps from a real facility was designed.

With all of the tools in place and relevant technologies touched upon, a set of core objects rich enough to represent an object and provide data for manufacturing, monitoring and control was developed first in C++ and then in Java object-oriented programming languages. A proof-of-concept computer application was programmed using Java. The program, called VBuild, incorporated tools for the three environments of design, build, and use representing the life cycle of an object or artifact. The program collected virtual representations of the kit-of-parts components from over the Internet, allowing the designer to assemble them together into a virtual building or artifact. The virtual artifact could then conceptually be used to order the manufacture of actual components and execute automated construction sequences for the final assembly. Finally, the VBuild program would facilitate the remote monitoring and control of the final object while it is in use.

To demonstrate the feasibility of the technology and concepts embraced by the experimental VBuild program, a model kit-of-parts was conceived. Though the various activities occured separately and independently, the three stages of design, build, and use were successfully demonstrated. Virtual representations of the kit-of-parts were assembled virtually in the design stage, components were ordered from real manufacturers online, assembly and disassembly of the final model was conducted using robots and automated means, and remote control and monitoring facilitated the observation and alteration of the final state of the model while it was in use. The demonstration proved the feasibility of an integrated life-cycle management concept and helped develop a foundation upon which the paradigm could stand.

To further illustrate the value of a life-cycle paradigm, three case studies were discussed. The chapter covering the Life-cycle Architecture System steps out of the model simulation phase and demonstrates an application of the life cycle principles to a series of real projects. With the exception of remote control and monitoring, the life-cycle of the kit-of-parts system is addressed. Virtual representations, the development of design and shape grammars, and expert-aided design processes are discussed in conjunction with the projects.

Another case study project, the Plug-in Condominium project discusses application of life-cycle management in all three phases of the building’s life, including owner participation in the design of residences, automated construction of the residential complex, and the establishment of facility web pages for each dwelling in order to affect remote facility management.

Finally, possible applications of the life-cycle paradigm to other disciplines was discussed. Specific applications in naval architecture and automotive design were cited with the Ukitecture floating structureand Qamel specialty utility vehicle projects.

9.2 Conclusion

This work represents a new paradigm for the design, construction / manufacture, and use of buildings and other artifacts. There has been considerable effort to develop theories and technologies relating to computer-based visualization, manufacturing, automated construction, and facility management in regards to the building industry. A major portion of that research was directed at conventional construction and design methods. In this work, it has been established that the use of kit-of-parts concepts eliminates many factors that have been hindering the research in these various fields. In addition, feasibility has been established that unification of these various technologies can be accomplished in an orderly and systematic way using kit-of-parts design and construction techniques. The research advocates the development of dynamic systems which have the potential to govern themselves through shape grammars, rule-based behaviour, automated assembly / disassembly, and remote monitoring / control rather than static one-time artifacts.

The study contributes to the understanding of design as it relates to component-based building systems, and provides new directions for automated construction and shape grammar research. This research is believed to be significant because of the following:

1. a new approach to modeling and representation of artifacts based on network decentralization and object-oriented concepts is proposed.
2. the proposed representation binds the virtual object to the actual physical artifact.
3. the feasibility of life-cycle building management based on the real-time gathering of information along the entire life span is established.
4. the development of kit-of-parts design grammars can contribute to the efficient design of both building and construction equipment.
5. simulations and example projects demonstrate the feasibility of using kit-of-parts design for efficient life-cycle management.

Architects have often shied away from kit-of-parts approaches, following the argument that pre-designed systems place too many restrictions on creative processes. Though certain movements such as the *Metabolists* and *Hi-tech* advocates have initiated significant design investigations relating to kit-of-parts construction techniques, design research on the topic has never been mainstream. Prefabrication, and the special subset kit-of-parts construction, was among other reasons born out of a need to produce inexpensive shelter in a short amount of time. A natural human tendency to achieve greater efficiency by preparing certain items and performing processes in advance gave rise to the manufacture and assembly of building components before they reach the site. Research and development of kit-of-parts systems has occured in spurts, usually coinciding with some historical event or movement in social structures. History has shown that attempts at establishing mass-production manufacturing and kit-of-parts concepts as mainstream construction methods have usually failed. However, many methods that were once considered to be cutting-edge prefabrication have become so common that they have fallen out of the current kit-of-parts definition.

In this work, isolated steps in the life-cycle of a building were simulated. Though the research establishes feasibility, the many issues and problems that would accompany a real-life scenario involving the life-cycle of a building are much too complex to address within the scope of this work. Therefore, this exercise does not establish practicality in its current state, but only suggests possibilities and establishes a foundation upon which future research can build.

9.3 Future research

Future research should concentrate on specific areas within the design / build / use life-cycle, with a narrower focus on individual topics. Through this work a paradigm has been established. Further research should attempt to work within the framework of the paradigm in order to develop theories and solutions that support it. Though this work concentrates on building design, construction, and facility management, the principles and technology can be applied to many other fields as well, such as the automotive industry, naval architecture, machine design, and consumer products industries. A list of possible research topics could be as follows, grouped according to life-cycle stage.

9.3.1 Design and visualization topics

Future research topics relating to design and visualization could include a further development of the core VBuild objects, or investigations into methods for translating back and forth between VBuild objects and other data representations. Rewriting the class coding in a more elegant manner using more recent versions of Java or some other language would be important. Since the VBuild representation is primarily production-oriented with geometric representation and fabrication methods, a next logical step would be to develop a single generic building object from which the core classes can be derived. A generic building model such as the one proposed by [Turner 1997] would also support relationships and dependencies, which the current VBuild core classes do not. The development of a generic object could provide an elegant way of modeling real-world artifacts, with fully incorporated handles for real-world linking, monitoring, and control. Object life span, behavior scripting, and task loops included as attributes can be combined with rule-based grammars for a rich virtual representation of buildings, vehicles, machines, and other real-world objects.

Other future research topics could include the development of simple web-based experts or analysis programs for kit-of-parts systems. One example would be the full linkage of the Genetic Programming (GP) engine and Rule-based Assembler (RBA) developed in conjunction with the Emerald project in the Life-cycle Architecture System. Analysis programs could include acoustical, environmental, and structural analysis programs.

Another future topic could be the further development of shape grammars and other design grammars specifically tailored for use with kit-of-parts systems. This research would include the mastery of GP and other rule-based languages.

9.3.2 Fabrication and construction topics

Future research topics relating to fabrication and construction could include the establishment of a web-based remote fabrication service. The service would start out one fabrication type at a time, incorporating the necessary data translators into VBuild in order to have the correct representations for manufacturing. Fabrication service research would include automated material handling, packaging, shipping and transport. Grammars which address fabrication size and type and match standard packaging strategies could be developed.

When a significant number of fabrication types have been addressed by a single web-based service, the automation of assembly and subassembly processes can be another topic of research. Perhaps grammars for subassembly can be derived which would be flexible enough to automatically manufacture a large number of variations.

Automated material handing from a factory to the site, regardless of distance, can be another topic. This would be the transport of assemblies and component to the site, where on-site automated construction equipment would be waiting.

Further investigations into the design of on-site automated construction systems would be another important topic. Again, the development of a generic object would be advantageous if it included three handles representing the main parts of robotic systems: controller for outside input, driver for manipulation in 3D space, and encoder for monitoring the current state and position. Using the same programming object for manipulator and manipulated may prove to be a powerful tool for life-cycle management of kit-of-parts systems. Adding rule-based grammars for behavior and task definition, pre-defined task and motion libraries can be developed which adds another dimension to the kit-of-parts concept.

9.3.3 Remote monitoring and control topics

Future research topics relating to remote monitoring and control could include further investigations into sensing and actuator hardware, and data logging techniques. Further simulations using X10 or LonWorks control networks would require the construction of mock-ups and on-site installations. Other research would need to include the study of networking and media, such as satellite and cable, especially for the purpose of sustained remote Internet connections.

Other research could include design studies componentizing various mechanical equipment and household appliances. Using a generic modeling object with its built-in handles for monitoring and control, development of equipment can include tight relationships with virtual counterparts right from the start.

APPENDICES

APPENDIX A
VBuild Applet, Server, and File Setup

The VBuild program currently in operation is structured to be expandable, using hyper-text markup language (HTML), Java, and Java Script. The directories, files, and programs are all under one root directory "VBuild."

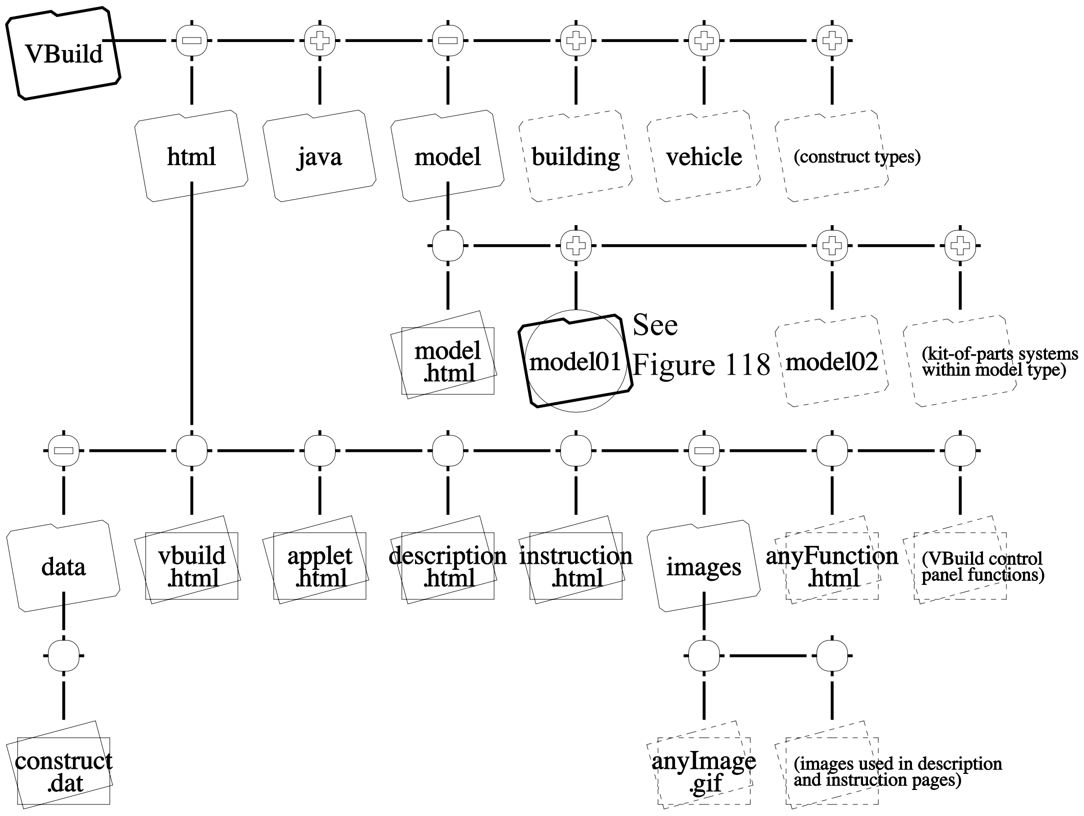


Figure 117: VBuild directory organization

At the first level within the VBuild main directory are the html and java directories. The html directory contains all the HTML pages used within the main VBuild program, as will be explained later. All the Java programming classes are present in the java directory, including both core objects and many other classes required for the user interface. This is where the VBuild applet is located, invoked from the applet.html document. Also in the first level are construct folders. Each type of construct will have its own directory. Currently only the “model” directory exists, but other future folders can easily be added as kit-of-parts systems are developed for other construct types. Future construct types can include buildings, vehicles, vessels, bridges, furniture, etc. Each construct directory will have an HTML file that holds a list of all available kit-of-parts systems for that particular construct. Each system will have its own kit-of-parts library located in a folder named after the system.

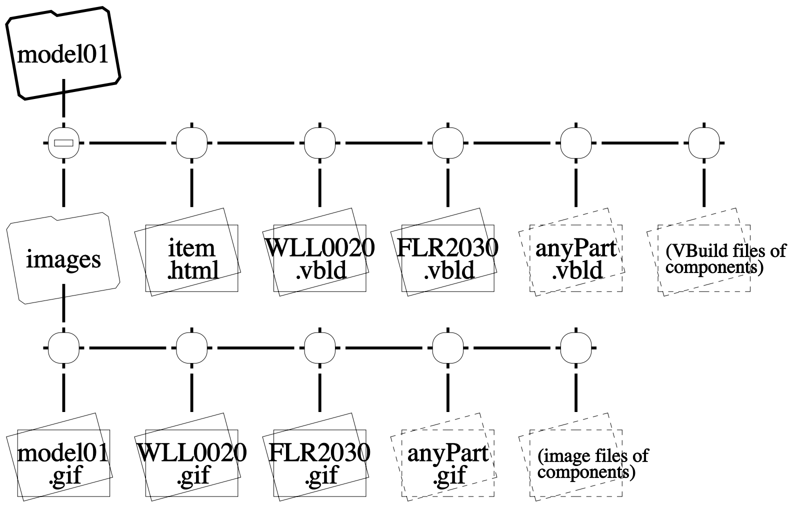


Figure 118: Folder and file placement in construct directories

As of this writing only one kit-of-parts system is available, “model01” which is discussed in Chapter 5. Inside each of the kit-of-parts library folders is an HTML file that lists all the available components in the library. Also included are images of an example construct or artifact assembled using the kit-of-parts, and images of each of the components. However, the main items of interest in the kit-of-parts library folder are the VBuild files of each of the components.

In the html directory are the HTML pages which run the VBuild program. As will be described in Appendix B, the VBuild web page consists of three frames: list frame, work frame, and instruction frame.

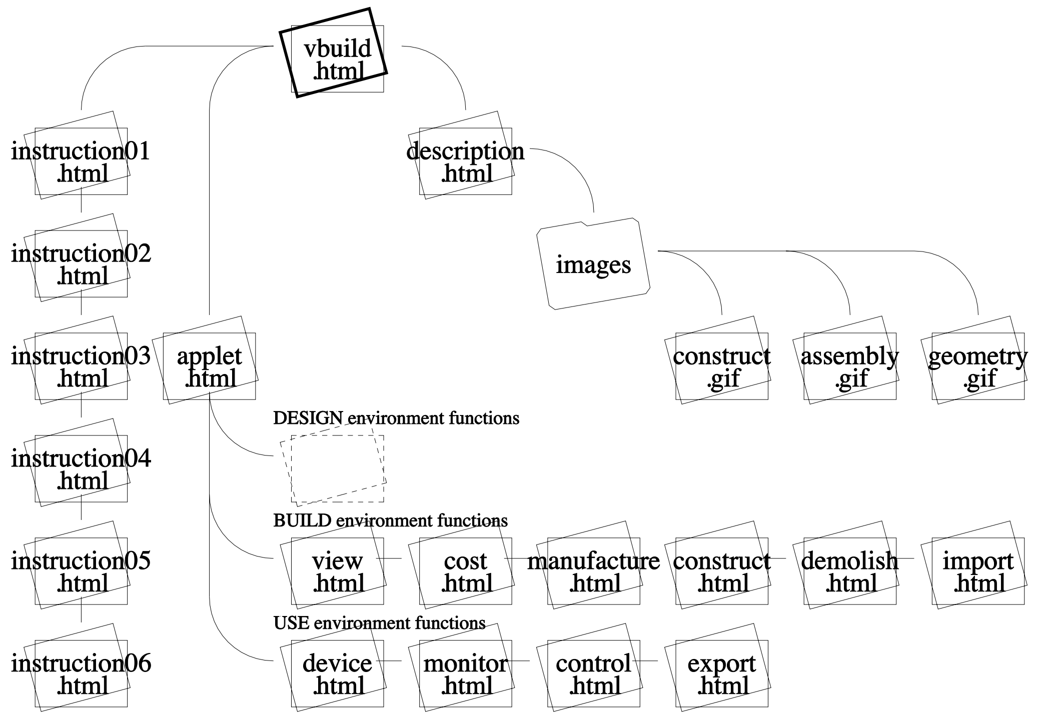


Figure 119: VBuild web page structure

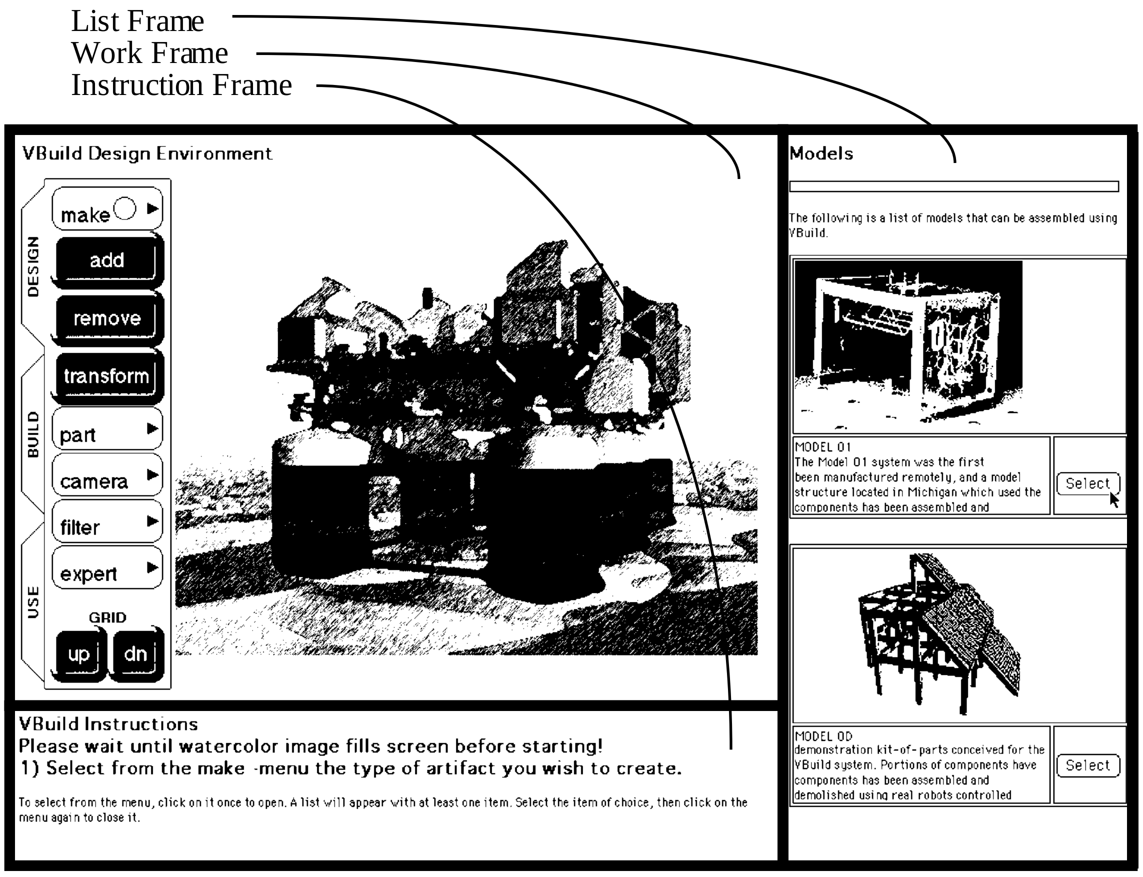
The vbuild.html file defines each of the three frames. Initially the applet.html file is loaded into the work frame, the instruction01.html file into the instruction frame, and the description.html file into the list frame. Throughout the entire use time of Vbuild, the applet.html file stays loaded since it contains the VBuild applet and main Java Script methods. The other frames, however, are reloaded with other HTML files from time to time as the need requires. For example, the instruction01.html file is replaced with other instruction files when the user changes environments. Buttons in the VBuild applet control panel load HTML files into the list frame in order to allow the user to choose from lists or run other Java Script functions on the side.

Once kit-of-parts libraries have been modeled, expanding VBuild to recognize them is a simple matter. If the library is of a new construct that didn't exist before, one would edit the construct.dat file that is located in the data folder, in the html directory. The construct.dat file is a list of names of construct types that will appear in the "make" menu of the VBuild control panel. The name is accompanied by a universal resource locator (URL) that points to where the construct directory is located. In addition to editing the construct.dat file, one would also prepare a construct HTML file which lists the kit-of-part system, or add the new system to one that already exists, such as the model.html file in the model construct directory. Next, an item.html file would be prepared which lists all the available components, along with images and VBuild files of each of the components. These would be inserted in a folder that has the same name as the new system, which would be placed in the appropriate construct directory.

Currently the entire VBuild directory is on a single server. Due to Java language security features, an applet will not read files that do not originate from the same machine that hosts the applet code. This means that if component VBuild files were located on other servers (such as one maintained by a manufacturer or designer), the VBuild program would not read them. The way to solve this would be to use Java's Remote Method Invocation (RMI) technology, and place a librarian object on the same server as the component files. This way VBuild can call up the remote librarian and have it serve up its own files.

APPENDIX B
VBuild User's Guide

The current version of VBuild has many functions that work as designed, but some controls represent previous research or are still in development so they only display conceptualizations. The conceptual functions will be pointed out in turn during the course of this guide.



The following is a list of models that can be assembled using VBuild.

MODEL 01
The Model 01 system was the first
been manufactured remotely, and a model
structure located in Michigan which used the
components has been assembled and

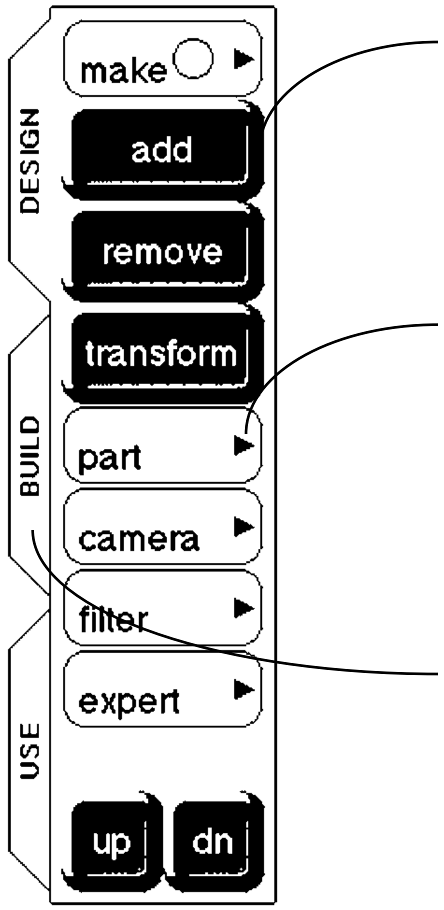
MODEL 0D
demonstration kit-of-parts conceived for the
VBuild system. Portions of components have
components has been assembled and
demolished using real robots controlled

VBuild Instructions
Please wait until watercolor image fills screen before starting!
1) Select from the make -menu the type of artifact you wish to create.

To select from the menu, click on it once to open. A list will appear with at least one item. Select the item of choice, then click on the menu again to close it.

Figure 120: VBuild starting screen

When first brought up inside the browser, the VBuild computer environment will look like Figure 120. The Java applet may take a while to download, so please wait until a watercolor image appears (there are three possible watercolors which do not necessarily coincide with the one shown in the figure). The window is divided into three frames: Work Frame, Instruction Frame, and List Frame.



BUTTON: Moving the mouse over the button will highlite it. Clicking once on the button will call its function.

MENU: Moving the mouse over the menu will highlite it. Clicking once within the area of the menu will open it, clicking again will close it.

If the list items in the menu have submenus, a triangle indicator will appear to the right of the item. Submenu items are selectable, which means that a circle indicator will appear at the left of the item when selected. Clicking on the item itself will select / deselect it, and clicking on the triangle indicator will open / close the submenu.

TAB: Clicking on the tab will move between environments.

EXCEPTIONS: If an item in the “make” menu has not been properly chosen, none of the other functions will work and tabs will not respond.

Figure 121: VBuild control panel

The Work Frame contains the applet which draws a control panel and views of the geometric model. The Instruction Frame contains directions on how to use VBuild, based on where the user is currently located in the program. The List Frame will hold lists, forms, auxiliary control panels, and added instructions depending on the function chosen. In the Work Frame, the main control panel is on the left of the drawing screen.

There are several parameters that need to be set from the beginning in order for VBuild to work properly. For this reason, none of the functions will work unless an artifact type has been chosen in the "make" menu.

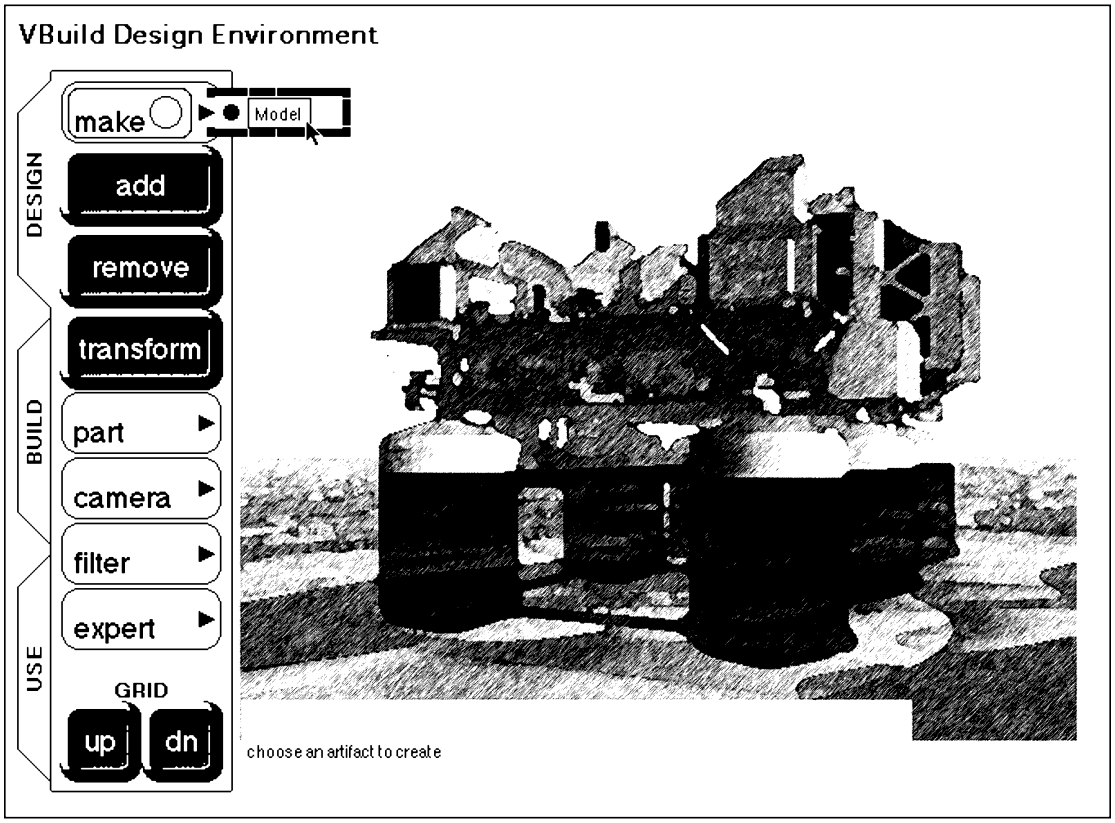
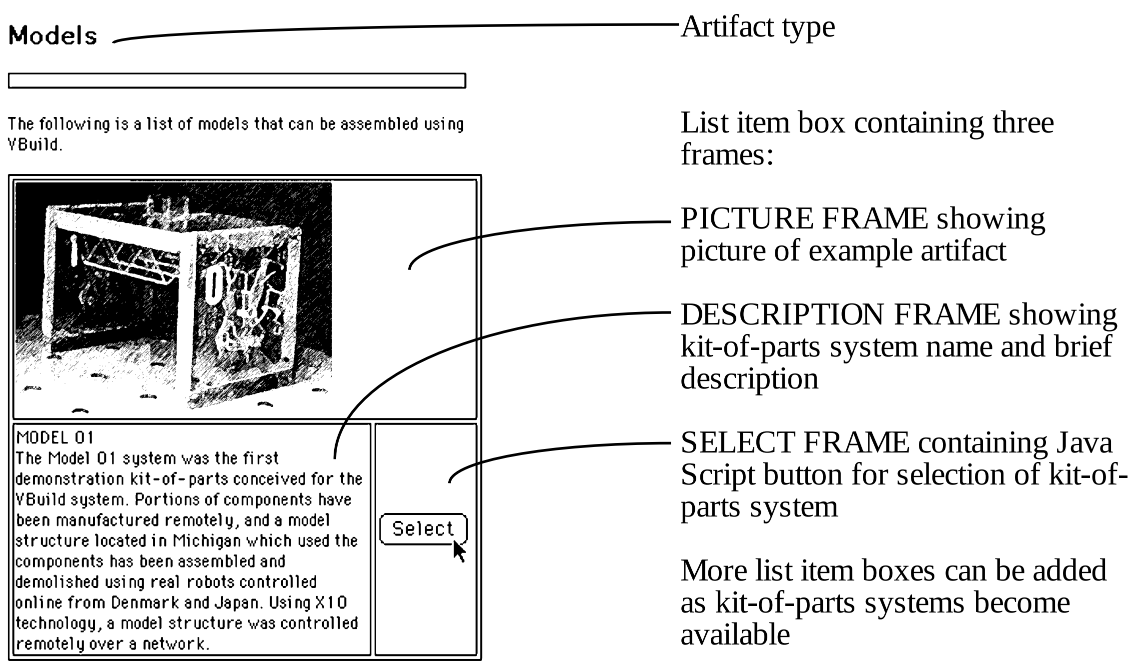


Figure 122: Construct selection menu

First, click on the "make" menu. The menu will open which contains one or more items which are artifact types. The items are read in real time from over the Internet, so new items can be added by an administrator as new kit-of-parts systems become available. Select an item. Note that the item is not really selected yet, until the "make" menu is closed by clicking on it again. If another button or menu is clicked on first, the "make" menu will close but the selection will be lost and VBuild will not work properly.



Models

The following is a list of models that can be assembled using VBuild.

MODEL 01
The Model 01 system was the first demonstration kit-of-parts conceived for the VBuild system. Portions of components have been manufactured remotely, and a model structure located in Michigan which used the components has been assembled and demolished using real robots controlled online from Denmark and Japan. Using X10 technology, a model structure was controlled remotely over a network.

Artifact type

List item box containing three frames:

PICTURE FRAME showing picture of example artifact

DESCRIPTION FRAME showing kit-of-parts system name and brief description

SELECT FRAME containing Java Script button for selection of kit-of-parts system

More list item boxes can be added as kit-of-parts systems become available

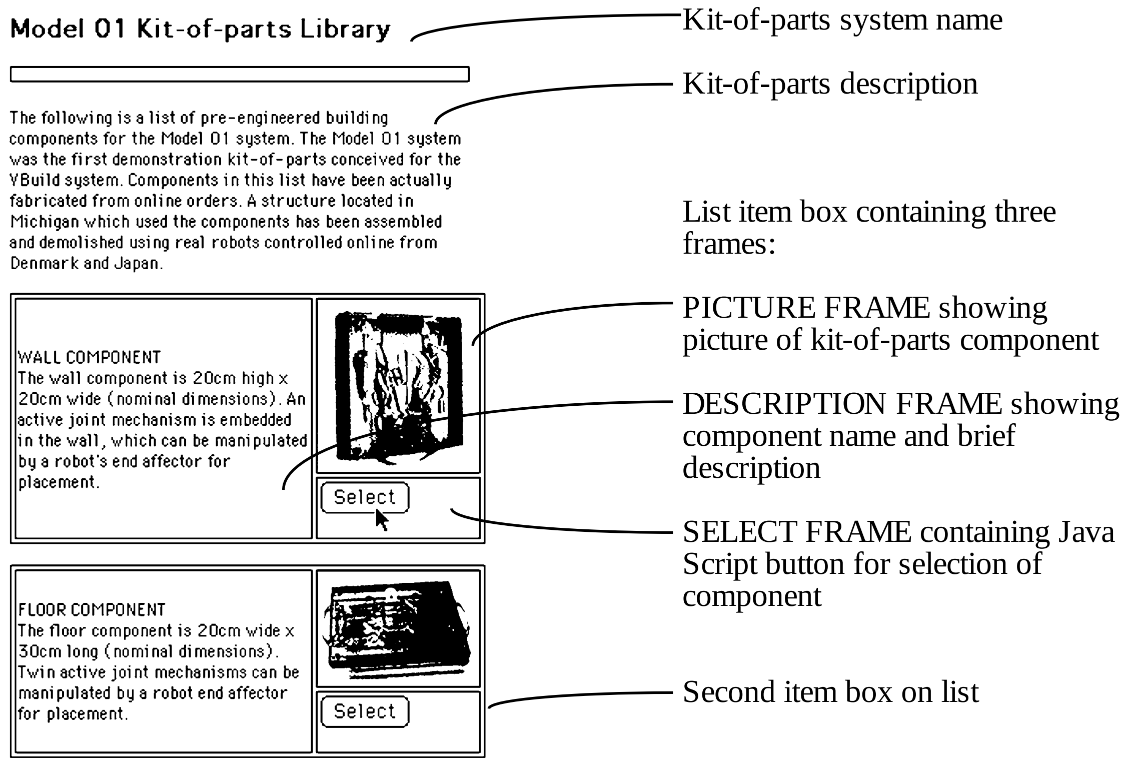
Figure 123: Kit-of-parts system selection list

Another useful utility is the status bar at the bottom of the screen. This area will be written over with text each time a button or menu is selected, giving added useful information and instruction.

Once the artifact type has been selected, the watercolor image will disappear and the screen will become blank. In the List Frame to the right, a list of available systems will appear. There will be at least one box containing a picture of an artifact made with the kit-of-parts, a brief description of the kit-of-parts system, and a select button.

There will be one such box for as many kit-of-parts systems as are available for that particular type artifact. Each of the systems are self-contained and are not necessarily compatible with other systems on the list. Selecting a system is a one-way endeavor, since clicking on the button will send setup information such as grid size and pitch, URL's for locating components over the Internet, and other parameters to the VBuild applet.

Clicking on the select button will bring up a list of available components that the kit-of-parts library contains. This list will appear in the List Frame, and each individual component will be enclosed in its own box consisting of picture, description, and select button.



Model 01 Kit-of-parts Library

The following is a list of pre-engineered building components for the Model 01 system. The Model 01 system was the first demonstration kit-of-parts conceived for the VBuild system. Components in this list have been actually fabricated from online orders. A structure located in Michigan which used the components has been assembled and demolished using real robots controlled online from Denmark and Japan.

WALL COMPONENT
The wall component is 20cm high x 20cm wide (nominal dimensions). An active joint mechanism is embedded in the wall, which can be manipulated by a robot's end affector for placement.

FLOOR COMPONENT
The floor component is 20cm wide x 30cm long (nominal dimensions). Twin active joint mechanisms can be manipulated by a robot end affector for placement.

Select
Select
Kit-of-parts system name
Kit-of-parts description
List item box containing three frames:
PICTURE FRAME showing picture of kit-of-parts component
DESCRIPTION FRAME showing component name and brief description
SELECT FRAME containing Java Script button for selection of component
Second item box on list

Figure 124: Component selection list

Design Environment

Select one of the components in the list. Clicking on the button will send certain parameters to the applet, such as which system the components belongs to (structure, mechanical, etc.), subsystem, and component type.

Next, press the "add" button. At this point VBuild will search in its own database to see if the component type has already been downloaded or not. If it hasn't it will go out on the Internet and download it. Next VBuild will calculate a bounding box that encloses all the fabrications in the component and display it on the screen. The first time may take a few seconds, but subsequent times it only needs to look in its own database.

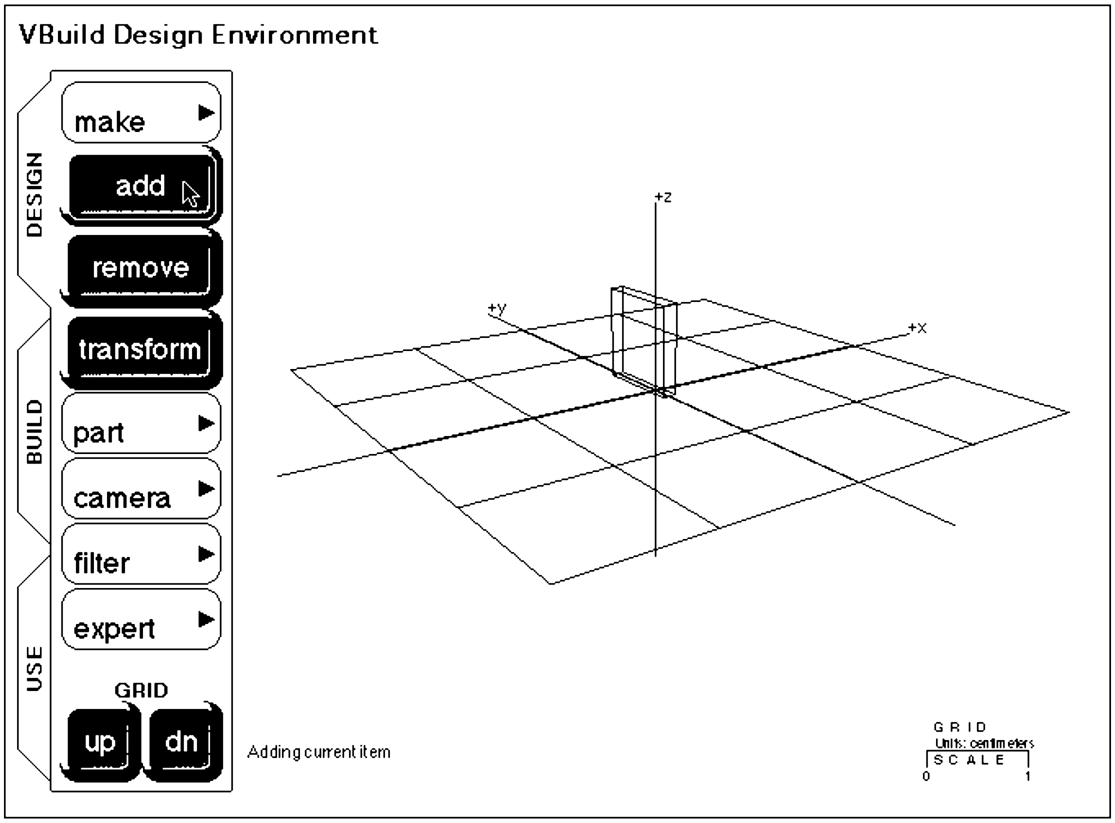


Figure 125: Downloading components real-time from the Internet

Notice the graphic scale in the lower right-hand corner of the screen. The scale is dynamic in that when a view changes, the scale will accurately redraw itself. In a perspective view this can be deceiving, since it is unclear how the scale applies. Behind the scenes, the scale is calculated from the picture plane which is not necessarily obvious to the viewer. On the bottom is the scale. This is according to the units the kit-of-parts system is defined in, which was set when the kit-of-parts system was selected. On the top is the grid pitch, which only shows if it is different from the scale. Of course the pitch or scale will not draw if it is too large to fit in the scale area, and only "GRID" or "SCALE" will show until the view is changed.

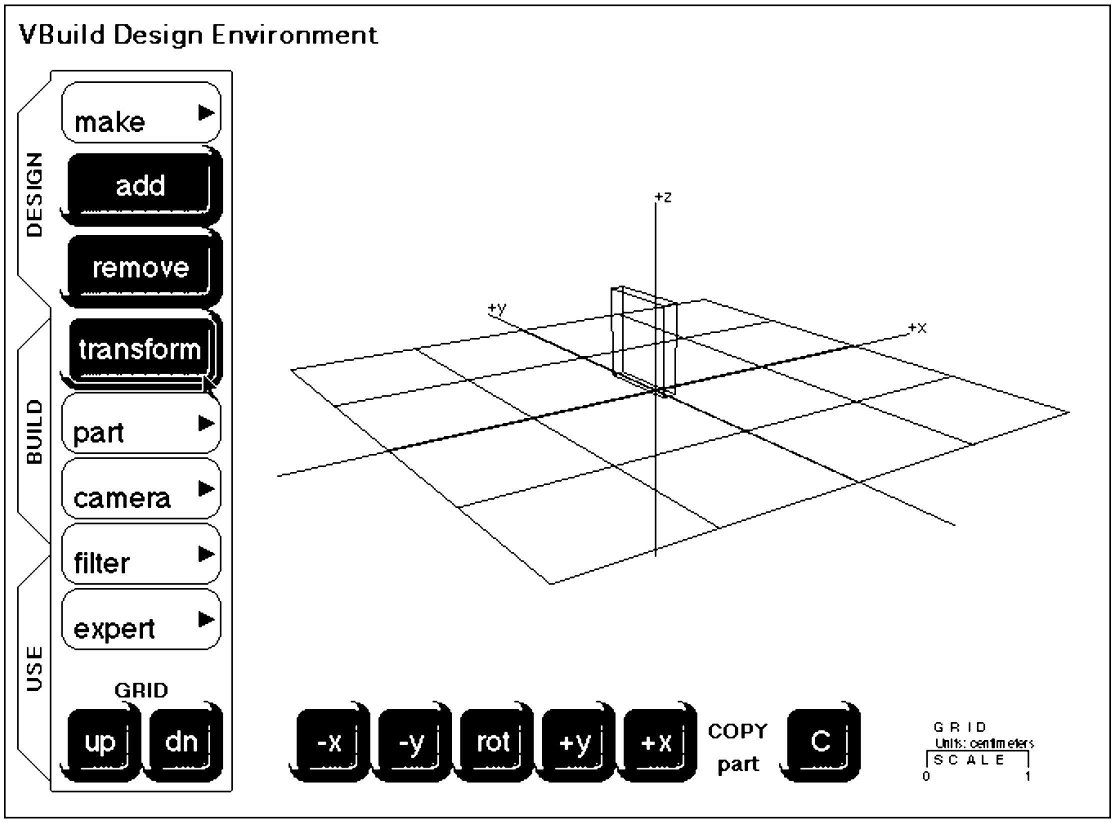


Figure 126: Component transform control panel

When a component is first displayed, it will be a red color which means that it is currently selected. Pressing the "remove" button will delete all selected items from the model. Once a component has been added, pressing the "transform" button will bring up a control panel at the bottom of the screen. Each of the buttons represents a direction in the axis system, except for the rotate button and copy button. Pushing the copy button, the transform function toggles between move and copy / move. Pressing any of the directional buttons will move the component along the grid an increment equal to the grid pitch as is predefined according to the kit-of-parts system that was selected. The rotate button will rotate the component counterclockwise a predefined number of degrees as the kit-of-parts system rules dictate.

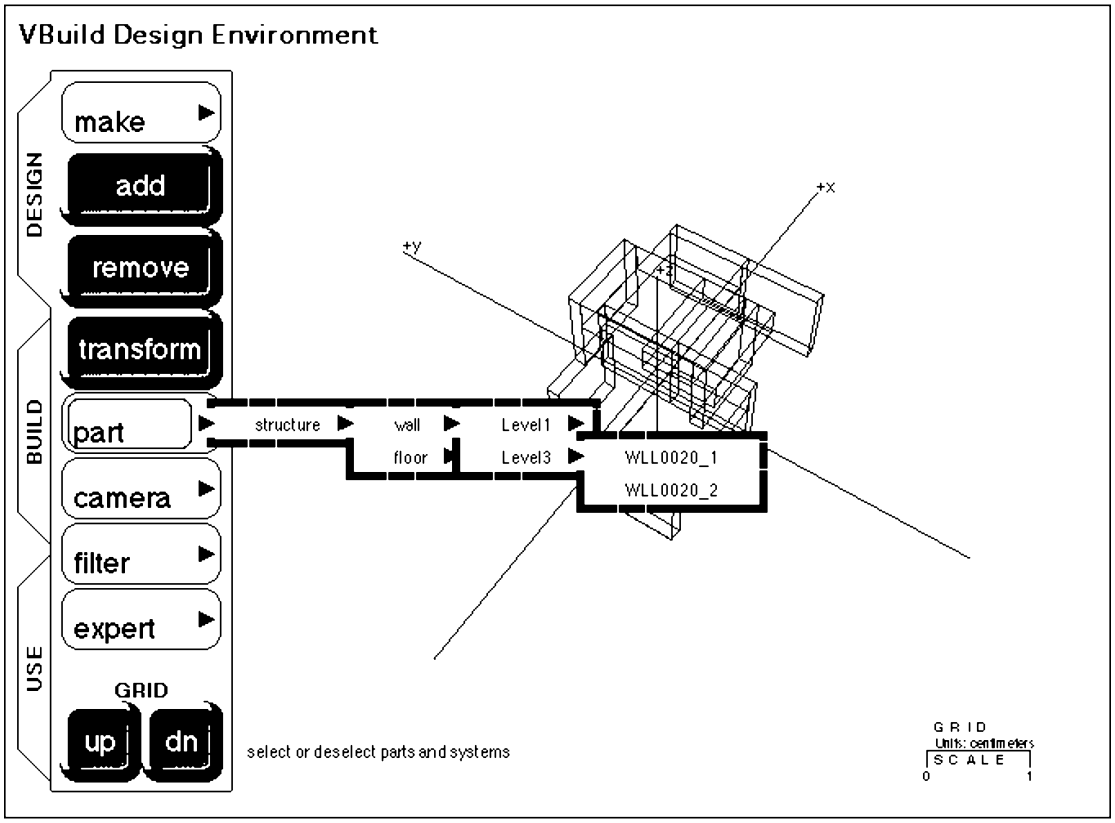
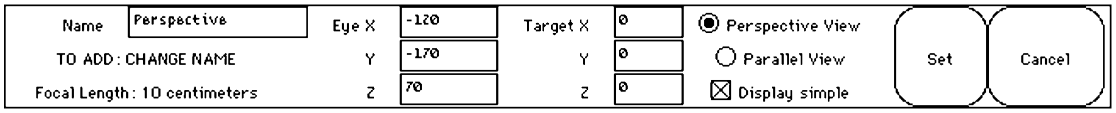


Figure 127: Virtual building hierarchy

When the components are first placed, the grid is at the zero height level. Pressing the "up" or "down" button will move the grid to a new level and allow placement of components on other floors or datum level. In this way, the selection of the component automatically defines the system, subsystem, and component type, and the level upon which the component lies automatically defines the group to which the component belongs. This becomes clear when the "part" menu is opened, and the hierarchical structure of the virtual building or artifact shows in the the nested menu structure. Using the "part" menu, components can be selected or deselected.



Name: Perspective
TO ADD: CHANGE NAME
Focal Length: 10 centimeters

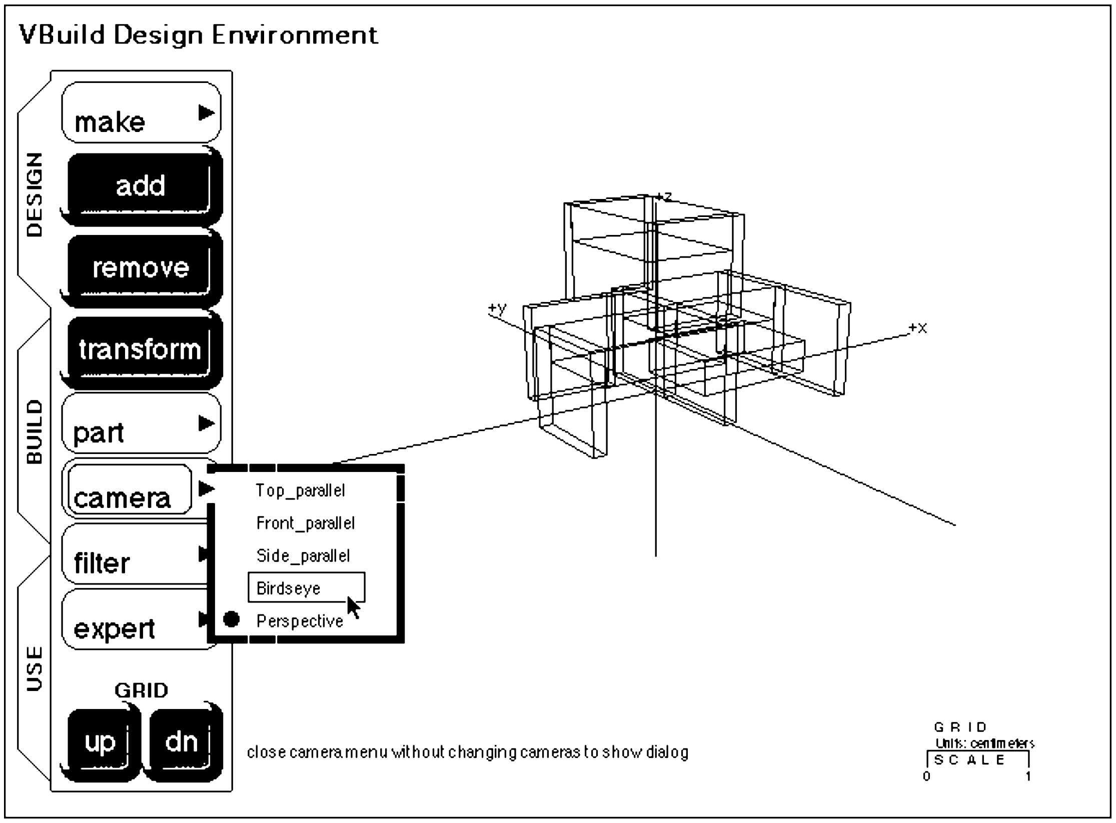
Eye X: -120
Y: -170
Z: 70

Target X: 0
Y: 0
Z: 0

[✓ Perspective View] Parallel View
[✓ Display simple]

Figure 128: Camera dialog

Clicking on the "camera" menu, views can be changed. On the outset, five default views are defined: three parallel views and two perspective views. To change views, simply select one of the unselected cameras and close the "camera" menu. When the menu closes the screen will redraw in the new view.



close camera menu without changing cameras to show dialog

Figure 129: Changing views in the camera menu

Closing the "camera" menu without changing the camera will bring up the camera dialog. The camera default settings can be changed this way, as well as new cameras defined by changing the name and pressing the "set" button.

The "filter" menu is similar to the "part" menu except that all items in the menu are selected by default, and deselecting systems, subsystems, groups, and components will temporarily hide them from display until they are selected again.

Currently under development is the "expert" menu. In the future, clicking on the menu would bring up a list of expert helpers that can assist with the analysis and design of the selected kit-or-parts system. VBuild will read a list of experts in real time from over the Internet, and perform analysis online.

Fabricate / Build Environment

Once the artifact is complete virtually, clicking on the "BUILD" tab will switch over to the fabricate / build environment for the manufacture of the artifact. Performing this simple act, the same virtual model produced in the design stage becomes a tool for actually producing the artifact in the real world.

Selecting a component and then pressing the "view" button will bring up a conceptual fabrication viewer. This function is currently under development but will eventually allow the user to scroll through all the fabrications which make up the component.

Another conceptual function is the cost estimation function. Pressing "cost", a form will appear in the List Frame. This tool will be used to make estimates on individual fabrications, components, or the entire artifact. The function will requrie the use of online agents which check current manufacturing prices in real time.

Since the paradigm presented in this work deals with pre-designed / pre-engineered components, each fabrication which makes up the component would be worked out in advance. The fabrications are integral as data types that are found in the core VBuild classes.

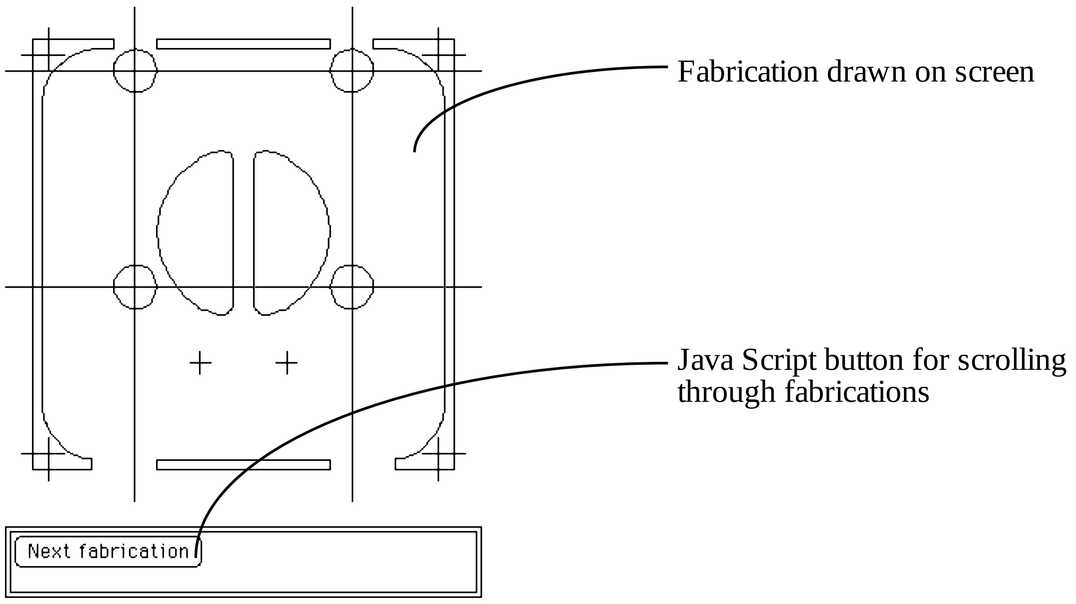
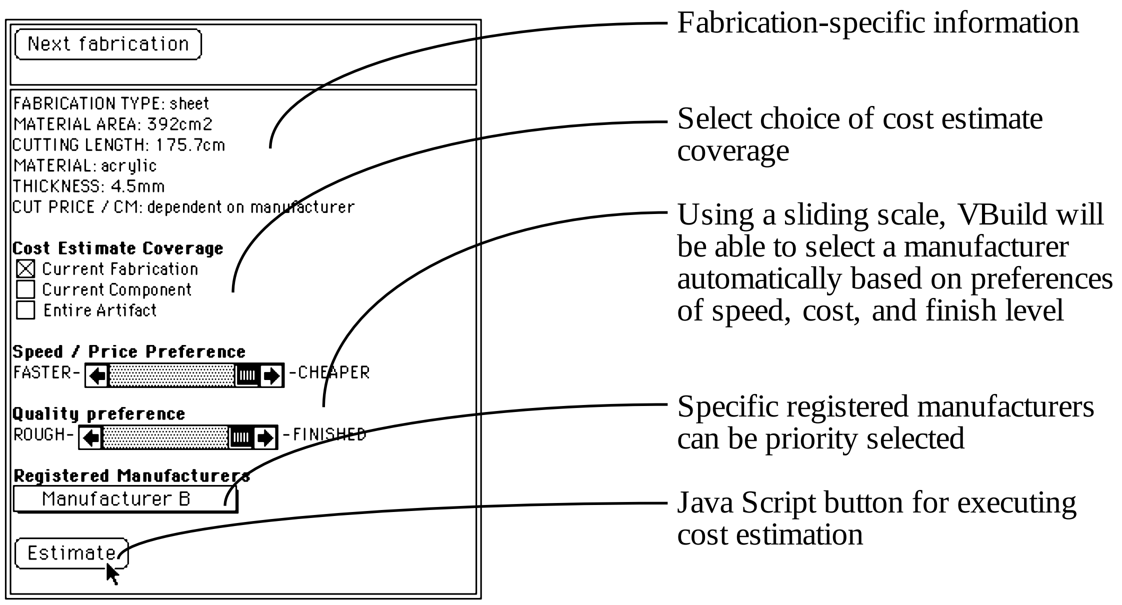

Fabrication drawn on screen
Java Script button for scrolling through fabrications
Next fabrication

Figure 130: Fabrication viewer


Next fabrication
Fabrication-specific information
Select choice of cost estimate coverage
Using a sliding scale, VBuild will be able to select a manufacturer automatically based on preferences of speed, cost, and finish level
Specific registered manufacturers can be priority selected
Java Script button for executing cost estimation
Estimate

FABRICATION TYPE: sheet
MATERIAL AREA: 392cm2
CUTTING LENGTH: 175.7cm
MATERIAL: acrylic
THICKNESS: 4.5mm
CUT PRICE / CM: dependent on manufacturer

Cost Estimate Coverage
[✓ Current Fabrication]
Current Component
Entire Artifact

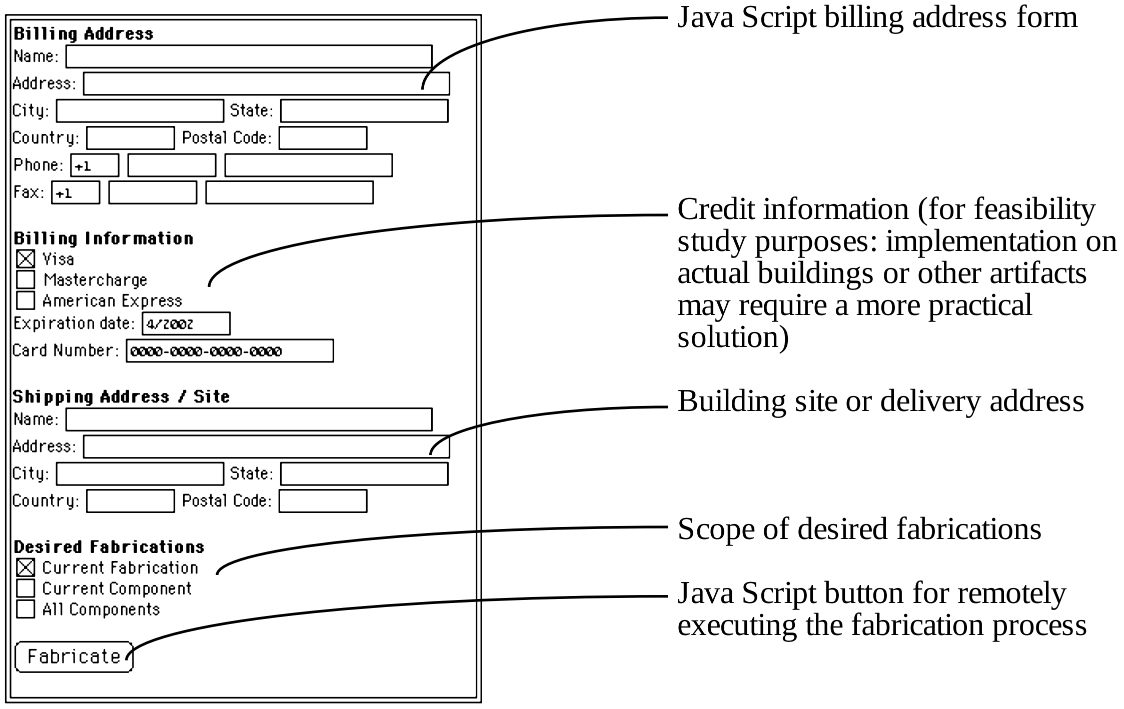
Speed / Price Preference
FASTER- -CHEAPER

Quality preference
ROUGH- -FINISHED

Registered Manufacturers
Manufacturer B

Figure 131: Cost estimation tool

Once the cost has been investigated, fabrication will be ordered using the "fabricate" button, which brings up a form for filling out billing information and delivery address. In a real world situation, it may not be feasible to order full-scale manufacturing or construction services with a credit card, but this research attempts to establish feasibility rather than discuss appropriateness. This function is not conceptual but represents investigations conducted in 1996-1997 where fabrications were ordered online from a real manufacturer.


Billing Address
Name:
Address:
City: State:
Country: Postal Code:
Phone: +1
Fax: +1

Billing Information
☒ Visa
☐ Mastercharge
☐ American Express
Expiration date: 4/2002
Card Number: 0000-0000-0000-0000

Shipping Address / Site
Name:
Address:
City: State:
Country: Postal Code:

Desired Fabrications
☒ Current Fabrication
☐ Current Component
☐ All Components

Fabricate
Java Script billing address form
Credit information (for feasibility study purposes: implementation on actual buildings or other artifacts may require a more practical solution)
Building site or delivery address
Scope of desired fabrications
Java Script button for remotely executing the fabrication process

Figure 132: Ordering fabrications

Pressing on the "construct" and "demolish" buttons in the current version brings up video clips of automated construction simulations conducted in 1996. Currently there is no projection as to when these functions could link to real world robotic construction systems, however in earlier chapters of this study can be found discussions on feasibility and actual working tools.

Pressing on the "import" button will bring up another conceptual function. This tool will be used for reading saved VBuild files or for creating new VBuild objects out of data produced in other applications.

Use Environment

Once the artifact has been manufactured / constructed, clicking on the "USE" tab will change environments to the facility management stage. This also allows the designer to seamlessly pass the design model, which was a pattern for producing the actual artifact, to the user, and produce tools for remote control and monitoring of it over the Internet.

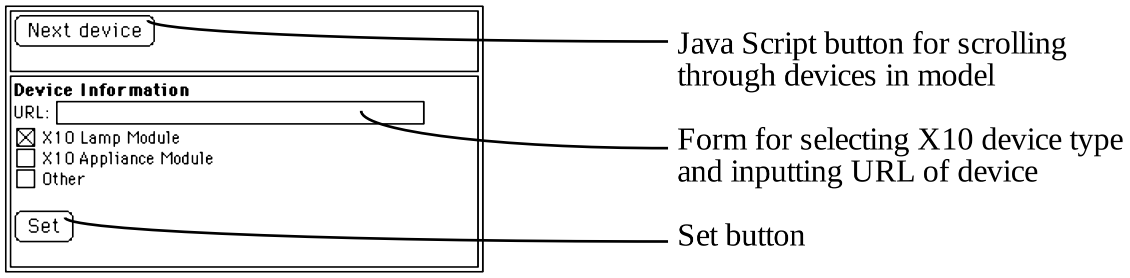
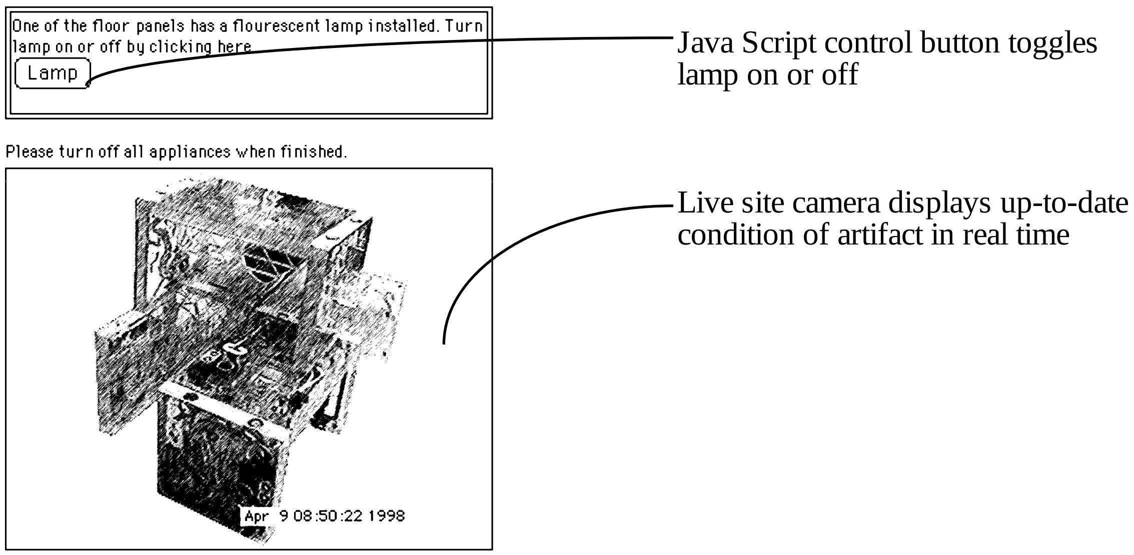

Next device
Device Information
URL:
☒ X10 Lamp Module
☐ X10 Appliance Module
☐ Other
Set
Java Script button for scrolling through devices in model
Form for selecting X10 device type and inputting URL of device
Set button

Figure 133: Control panel for setting devices

Pressing the "device" button, the user can scroll through components in the artifact and assign URL's to specific devices which have been manufactured as part of the component. Currently this is also a conceptual function under development. These URL's are included as data members in the core VBuild classes, and can be used to generate anchors in publicly accessible VRML models of the artifact. In this research, X10 pseudo-standard control modules have been used, but it is expected that other DDC hardware such as LonWorks can be used as well.

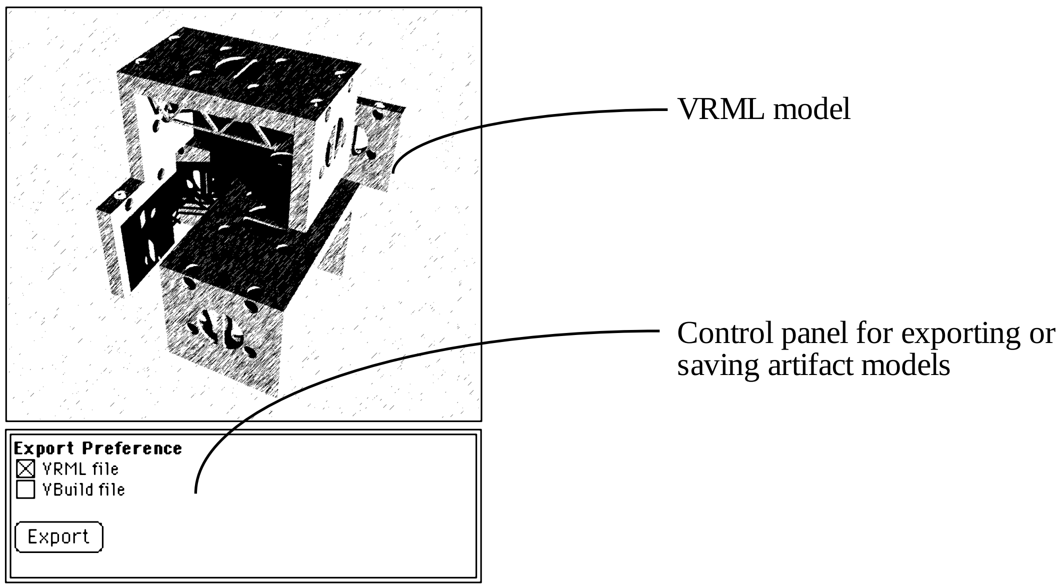

One of the floor panels has a flourescent lamp installed. Turn lamp on or off by clicking here

Please turn off all appliances when finished.

Apr 9 08:50:22 1998

Lamp
Java Script control button toggles lamp on or off
Live site camera displays up-to-date condition of artifact in real time

Figure 134: Remote monitoring and control


Export Preference

[✓ VRML file]

VBuild file

Figure 135: Exporting models

Pressing the "monitor" or "control" buttons will bring up a setup for direct digital online control of the devices in the VBuild model. At the time of this writing, this function is actually working behind Kajima Corporation's firewall. Another version of VBuild posted outside the firewall has a simulated version which uses image files.

Finally another conceptual function, pressing the "export" button will eventually provide the ability to export VRML models that can be used as a tool for remote control and monitoring. Within Kajima's firewall, clicking on parts of this VRML model will turn on and off the associated devices built into the components.

APPENDIX C
VBuild Core Class Methods

VBuild core classes were written first in C++ and then Java 1.0. The VBuild program includes many other classes which support the graphic user interface and environment, but only the core methods will be listed here. This list only includes those methods which were functional at the time of this writing. The classes are group in the three main categories of geometry, assembly, and construct in ascending hierarchy.

GEOMETRY CLASSES

Geometry classes are the most basic data types, which represent the physical geometry of the various fabrications and objects in the model. The four geometry classes are NetPoint, NetLine, NetPolygon, and NetSolid.

class NetPoint

The NetPoint class represents zero-dimensional geometry. Points are used as vertices in solids and end points of lines. The primary members are the three xyz values that locate the point in space, and additional methods manipulate those values. Within VBuild, a positive z value is up, with the x and y forming a horizontal plane.

public double x_coordinate

The x value in three-dimensional space.

public double y_coordinate

The y value in three-dimensional space.

public double z_coordinate

The z value in three-dimensional space.

NetPoint()

The default constructor method for NetPoint which creates a new point at the origin.

NetPoint(double,double,double)

Another constructor method which creates a new point at the xyz values specified.

public void equals(NetPoint)

A method which functions the same way as the = operator. This method will set the xyz values to be the same as the point specified. This method departs from Java 1.0 convention which uses the "equals" function to return a boolean value if the two are the same. Instead, this method is more equivalent to Java's "clone" convention.

public java.lang.String toString()

This method formats the xyz values of the point in an order for placement in a string data type for output purposes.

public java.lang.String toString(int)

This method is the same as the previous one, except that the xyz values are rounded and truncated at the decimal place specified.

public void pt_transform(NetMatrix)

This method resets the xyz values as recalculated by the transformation matrix.

public double compute_distance(NetPoint)

This method returns the distance from the specified point.

public boolean is_same_as(NetPoint,double)

This method returns a true if the point specified is the same within the specified tolerance, and a false if it lies outside the tolerance.

public boolean is_in_box(NetPoint,NetPoint,double)

This method returns a true if the xyz values describe a point that lies within a box defined by the specified points within the specified tolerance, and a false if it lies outside the box.

class NetLine

The NetLine class represents one-dimensional geometry. Lines are used in grids and other similar objects. The primary members are two NetPoints which form the beginning and ending points of the line. Other members include values for line thickness and color, and methods for manipulating the members.

public NetPoint begin

The beginning point of the line.

public NetPoint end

The ending point of the line.

public int line_thickness

The value which represents the thickness of the line for output purposes.

public NetColor line_color

A member which represents the color of the line.

NetLine()

The default constructor which creates a new line one unit long in the positive x direction.

NetLine(NetPoint,NetPoint)

A constructor which creates a new line using the specified points and beginning point and ending point.

public void equals(NetLine)

A method which resets the beginning and ending points to be the same as the specified line.

public java.lang.String toString()

This method formats the members of the line in an order for placement in a string data type for output purposes.

public java.lang.String toString(int)

This method is the same as the previous one, except that all values are rounded and truncated at the decimal place specified.

public void line_transform(NetMatrix)

This method relocates the beginning and ending points to new locations as recalculated by the specified transformation matrix.

public void set_thickness(int)

This method sets the line's thickness member.

public int get_thickness()

This method returns the line's thickness value.

public void set_color(NetColor)

This method sets the line's color.

public NetColor get_color()

This method returns the line's color member.

public double compute_length()

This method returns the line's length.

public boolean is_same_as(NetLine,double)

This method returns a true if the line specified is the same within the specified tolerance, and a false if it lies outside the tolerance.

public NetPoint compute_divide(int)[]

This method divides a line by the specified number and returns an array of points that coincide with the segment end points.

class NetPolygon

The NetPolygon class represents two-dimensional geometry. Polygons are used to represent tool paths in sheet fabrications, extruded sections, etc. The primary members are an array of NetPoints which form the vertices of the polygon, and an array of integers called the id vector which contain information about the polygon. Since these members are private they do not appear in the member list. Other members include names and methods for manipulating the members. The NetPolygon is actually a polygon set, where several individual polygons can become holes or be detached from larger polygons. Using the right-hand rule, polygons defined with vertices in counter-clockwise order are surfaces, where clockwise polygons are holes. In the id vector, the values are as follows:

id[0] = total number of points or vertices

id[1] = total number of individual polygons

id[2] = number of points in first polygon

id[3] = number of points in second polygon (if present)

id[4] = etc.

public java.lang.String polygonName

This member is a string type member that functions as the polygon's name.

NetPolygon()

This method is the default constructor which creates a new triangular polygon at the origin with legs one unit long in the positive x and y directions.

NetPolygon(NetPoint [],int [])

This method creates a new polygon set from the specified array of points and integer array which will be loaded up as the id vector.

public void equals(NetPolygon)

This method removes the original vertice and id arrays and loads copies of the specified polygon’s members in their stead to make the two polygons the same.

public java.lang.String toString()

This method formats the members of the polygon in an order for placement in a string data type for output purposes.

public java.lang.String toString(int)

This method is the same as the previous one, except that all values are rounded and truncated at the decimal place specified.

public void pol_transform(NetMatrix)

This method relocates the vertice points to new locations as recalculated by the specified transformation matrix.

public double compute_perimeter()

This method returns the total perimeter of all the polygons in the polygon set.

public double compute_area()

This method returns a value that represents the area (minus subtracted holes) of all of the polygons in the polygon set.

public NetPoint get_vertice(int)

This method returns a point from the vertice array according to the specified integer. The integer specifies the order in the array.

public void set_vertice(int,NetPoint)

This method replaces the point in the vertice array, located in the position specified, with another point.

public int get_id(int)

This method returns the value in the id vector whose position corresponds to the specified integer.

public void set_id(int,int)

This method replaces the value in the id vector, located in the position specified, with another value.

public NetVector get_normal()

This method returns a vector object which represents the normal of the polygon set.

public double get_d_value()

This method returns the "d" value for the polygon set which is required for the plane definition equation $ax+by+cz+d=0$.

public NetLine extract_line()[]

This method returns an array of NetLines that are equivalent to the edges of the polygons in the polygon set.

public NetPolygon extract_polygon()[]

This method returns an array of single polygons extracted from the polygon set.

public NetPoint computeBoundingBox()[]

This method returns two points which lie on the same plane as the polygon set, that represent opposite corners of a bounding box that encloses the set.

public NetPoint compute_spread_centroid()

This method returns a point that is in the middle of the widest spread of vertices in the xyz directions.

public NetPoint compute_area_centroid()

This method returns the point that represents the centroid of the net areas in the polygon set.

public double plane_def()[]

This method returns an array that contains all four abcd values for the plane definition equation ax+by+cz+d=0.

class NetSolid

The NetSolid class represents three-dimensional geometry. Solids are used to represent objects. The primary members are an array of NetPoints which form the vertices of the solid, and two arrays of integers called the id vector and ln vector which contain information about the solid. Since these members are private they do not appear in the member list. Other members include names and methods for manipulating the members. The NetSolid is actually a polyhedron set, where several individual polyhedra can become cavities or be detached from larger polygons. In the id vector, the values are as follows:

id[0] = number of edges or values in the ln vector id[1] = total number of points or vertices

id[2] = total number of polygon sets

id[3] = total number of polyhedra

id[4] = size of the id vector array

    id[5] = number of edges in polyhedra 1

    id[6] = number of polygon sets in polyhedra 1

        id[7] = number of points in polygon set 1

        id[8] = number of polygons in polygon set 1

            id[9] = number of points in first polygon

            id[10] = etc.

    id[n] = number of edges in polyhedra 2

    id[n+1] = etc.

In the solid, each of the edges would be defined by the points that make up their end points. The ln vector holds the positions in the array of the vertice points that define each of the sides. For example a cube has six sides and eight vertices in array positions 0-7. The ln vector would contain the values 0 1 3 2 for the first face, 2 3 5 4 for the second face, 4 5 7 6 for the third face, 6 7 1 0 for the fourth face, 0 2 4 6 for the top face, and 1 7 5 3 for the bottom face.

public java.lang.String sol_name

This member is a string type member that functions as the solid’s name.

public java.lang.String primitive

This member is a string type member that functions as the primitive name for the solid. At this writing, the primitive types include SOL = general solid, CUB = cuboid, CYL = cylinder, CON = cone, SPH = sphere, and TOR = torus.

NetSolid()

This method is the default constructor which creates a new cuboid solid at the origin one unit cubed in the positive xyz directions.

NetSolid(NetPoint [],int [],int [])

This method creates a new solid which sets the vertice array, id vector, and ln vector with the specified arrays.

public void equals(NetSolid)

This method removes the original vertice, id, and ln arrays and loads copies of the specified solid's members in their stead to make the two solids the same.

public java.lang.String toString()

This method formats the members of the solid in an order for placement in a string data type for output purposes.

public java.lang.String toString(int)

This method is the same as the previous one, except that all values are rounded and truncated at the decimal place specified.

public void solid_transform(NetMatrix)

This method relocates the vertice points to new locations as recalculated by the specified transformation matrix.

public NetPoint get_vertice(int)

This method returns the point located in the vertice vector at the specified position.

public int get_id(int)

This method returns the value located in the id vector at the specified position.

public int get_ln(int)

This method returns the value located in the ln vector at the specified position.

public double get_parameters()[]

This method returns an array holding built-in primitive parameters. The array is [0] = x dimension, [1] = y dimension, [2] = z dimension, [3] = primary radius, [4] = secondary radius.

public NetSolid getBoundingBox()

This method returns a cuboid solid which is the bounding box for the polyhedron set.

public void createFromStream(java.io.DataInput)

This method is a special VBuild method which creates a solid from a stream input from a VBuild component file.

public double compute_volume()

This method returns the volume of the Polyhedron set (minus cavities).

public double compute_surface_area()

This method returns the total surface area of all the polyhedra in the set.

public NetLine extract_line(double)[]

This method returns an array of NetLines that are equivalent to the edges of the solids in the polyhedron set.

public NetPolygon extract_polygon()[]

This method returns an array of NetPolygons that are equivalent to the faces of the solids in the polyhedron set.

public void make_cube(double,double,double)

This method makes a CUB cuboid primitive having the xyz dimensions specified. The method automatically fills in all the vertice, id, and ln vector information.

public void make_cube(NetPoint,NetPoint)

This method is similar to the previous one except that it uses two opposing points to define the cuboid.

public void make_cylinder(double,double,int)

This method makes a CYL cylinder primitive having a radius and height as specified, with a specified number of facets.

public void make_cone(double,double,double,int)

This method makes a CON cone primitive having a top and bottom radius and height as specified, with a specified number of facets. The top radius may be zero.

ASSEMBLY CLASSES

Assembly classes are the data types that define the individual components in a kit-of-parts library. They include classes that represent fabrications and methods that drive net-based manufacturing processes. The four assembly classes are NetPart, NetFabrication, NetSubassembly, and NetAssembly. An additional companion to the NetAssembly class is the NetInstance class.

class NetPart

The NetPart represents raw materials. Primary members include a polygon and a solid. Depending on the type of part, either the solid or the polygon are used but not both at the same time. The solid would represent a solid object, and the polygon would represent either a section for extrusion or a cutout from sheet stock. Since the polygon and solid members are private they are not included in the list.

public NetMaterial material

This member type provides material attributes for the part.

public java.lang.String part_name

This member is the part name.

NetPart()

This is the default constructor which initializes the member polygon and solid.

NetPart(NetSolid,NetPolygon,NetMaterial)

This constructor creates a new part using the specified solid, polygon, and material object.

public void equals(NetPart)

This method sets the various members to be the same as those of the specified part.

public java.lang.String toString()

This method formats the members of the part in an order for placement in a string data type for output purposes.

public java.lang.String toString(int)

This method is the same as the previous one, except that all values are rounded and truncated at the decimal place specified.

public void part_transform(NetMatrix)

This method relocates the part members to new locations as recalculated by the specified transformation matrix.

public NetSolid getBoundingBox(java.lang.String)

This method returns a cuboid solid that represents the bounding box of the part.

public NetSolid extract_solid()

This method returns the member solid.

public NetPolygon extract_polygon()

This method returns the member polygon.

class NetFabrication

The NetFabrication represents formed, shaped, drilled, or cut parts. The fabrication uses either the solid or the polygon from the part. In the case of fabrications cut from sheet stock, the polygon would represent the cut lines or tool path. In the case of extrusions, The polygon would represent the section of the extrusion which would be extruded along a path. Primary members include a NetPart and a path, which is currently a NetLine element but could eventually be more complex. The part and path, along with another member which represents thickness, are private so they are not included in the list. Other members include a transformation matrix and methods for manipulating the fabrication. In the assembly classes, the final product is the NetAssembly (described later). The 3D space origin of the fabrication is located at a point convenient for the numerical control machine which will produce it, then the matrix is used to locate the fabrication in relation to the NetAssembly model's origin.

public NetMatrix fab_matrix

This member is the transformation matrix which is used to recalculate the location of the fabrication within the model.

public java.lang.String fab_type

This member is the type of fabrication. Except for the generic fabrication type FABRICATION (which is simply a solid), each fabrication type must be specifically designated in order to use the proper method to prepare the data for creation of the fabrication. As of this writing, the other fabrication types available in VBuild are EXTRUDE and SHEET. Other fabrication types can be added as needed.

public java.lang.String fab_name

This member is the fabrication's name.

public boolean show

This member is used to filter which fabrications are to be displayed. If true the fabrication is displayed.

NetFabrication()

This default constructor creates a new fabrication.

public void equals(NetFabrication)

This method sets the various members to be the same as those of the specified fabrication.

public java.lang.String toString()

This method formats the members of the fabrication in an order for placement in a string data type for output purposes.

public java.lang.String toString(int)

This method is the same as the previous one, except that all values are rounded and truncated at the decimal place specified.

public void fab_transform(NetMatrix)

This method multiplies the matrix member by the specified transformation matrix. In this way the location of the fabrication within the model may be changed, but the fabrication itself keeps its optimal location for numerical control machine production.

public void set_matrix(NetMatrix)

This method sets the matrix member to be the same as the specified transformation matrix.

public void set_path(NetLine)

This method sets the private path member to be the same as the specified line.

public NetLine get_path()

This method returns the path in the form of a line element.

public double getThickness()

This method returns the private thickness member of the fabrication.

public NetSolid getBoundingBox()

This method returns a cuboid that represents the bounding box of the fabrication.

public NetPart extract_part()

This method returns the part member of the fabrication.

public void make_fabrication(NetPart)

This method loads the specified part into the fabrication.

public void make_extrusion(NetPart,NetLine)

This method creates an extrusion-type fabrication using the specified part and line element.

public void make_cut_sheet(NetPart,double)

This method creates a sheet-type fabrication using the specified part and a value that will be loaded into the thickness member.

public int order_fabrication()

This method contacts a remote contractor CGI over the network and commissions fabrications to be made. The CGI then orders the fabrications from real manufacturers. At this writing, this method is not yet in full working order but represents the coding developed in the remote fabrication experiments (discussed in an earlier chapter).

class NetSubassembly

The NetSubassembly represents an object built from one or more fabrications. An example of a subassembly could be a door (including door and frame), or an HVAC fan unit. The subassembly's primary member is a list object that can contain an unlimited number of fabrications. The list member is private so it is not included here. Other members include subassembly name, boolean show attribute, and Uniform Resource Locator (URL). The URL is used in remote control and monitoring when the subassembly is a device, sensor, or actuator. In this case, the URL would represent the actual location of the device on the network in order to affect virtual / real linkage.

public java.lang.String subassemblyName

This member is the subassembly's name.

public java.net.URL subassemblyURL

This member is the URL which represents the location on the network of the real subassembly counterpart.

public boolean show

This member is used to filter which subassemblies are to be displayed. If true the subassembly is displayed.

NetSubassembly()

This is the default constructor which initializes all the members.

public void equals(NetSubassembly)

This method sets the various members to be the same as those of the specified subassembly, including the contents of the fabrication list.

public java.lang.String toString()

This method formats the members of the subassembly in an order for placement in a string data type for output purposes.

public java.lang.String toString(int)

This method is the same as the previous one, except that all values are rounded and truncated at the decimal place specified.

public void subassembly_transform(NetMatrix)

This method relocates each of the fabrications in the list to new locations as recalculated by the specified transformation matrix.

public int get_number()

This method returns the number of fabrications in the fabrication list.

public NetSolid getBoundingBox()

This method returns a cuboid solid which represents the bounding box of the subassembly.

public void createFromStream(java.io.DataInput)

This method creates a subassembly from a VBuild file located on the network.

public void add_fabrication(NetFabrication)

This method adds the specified fabrication to the fabrication list, and automatically names it according to fabrication type and list size. For example, if there are already three fabrications in the list and a new SHEET-type fabrication is to be added, this method would rename the fabrication "SHEET_Fabrication_4" before adding it to the list.

public void remove_fabrication(java.lang.String)

This method removes the fabrication with the specified name, and automatically goes down the list and renames the remaining members from the beginning according to the convention mentioned above.

public NetFabrication extract_fabrication(java.lang.String)

This method returns the fabrication with the specified name.

class NetAssembly

The NetAssembly represents the component in the kit-of-parts library. An example of an assembly could be a column or wall panel complete with connection hardware, or a prefabricated module. The assembly's primary member is a list object that can contain an unlimited number of subassemblies. The list member is private so it is not included here. Other members include assembly type and solid list, and methods for manipulating the assembly.

The solid list is a linked list that can contain an indefinite number of solids for visual representational purposes. Independent of the subassemblies and fabrications, the solids can be used to simplify the display.

public java.lang.String assemblyType

This member is the assembly type.

NetAssembly()

This default constructor creates a new assembly with an empty subassembly list.

public void equals(NetAssembly)

This method sets the various members to be the same as those of the specified assembly, including the contents of the subassembly list.

public java.lang.String toString()

This method formats the members of the assembly in an order for placement in a string data type for output purposes.

public java.lang.String toString(int)

This method is the same as the previous one, except that all values are rounded and truncated at the decimal place specified.

public void assembly_transform(NetMatrix)

This method relocates each of the subassemblies in the list to new locations as recalculated by the specified transformation matrix.

public int get_number()

This method returns the number of subassemblies in the subassembly list.

public void set_type(java.lang.String)

This method resets the assembly type to the specified string.

public NetSolid getBoundingBox()

This method returns a cuboid solid which represents the bounding box of the assembly.

public void add_subassembly(NetSubassembly)

This method adds the specified subassembly to the subassembly list, and automatically names it according to the list size. For example, if there are already four subassemblies in the list and a new one is to be added, this method would rename the subassembly "Subassembly_5" before adding it to the list.

public void remove_subassembly(java.lang.String)

This method removes the subassembly with the specified name, and automatically goes down the list and renames the remaining members from the beginning according to the convention mentioned above.

public NetSubassembly extract_subassembly(java.lang.String)

This method returns the subassembly with the specified name.

public void createFromStream(java.io.DataInput)

This method creates an assembly from a VBuild file located on the network.

public void add_solid(NetSolid)

This method adds specified solid to the solid list.

public LinkList extractSolids()

This method returns a copy of the solid list.

class NetInstance

The NetInstance class is closely related to the NetAssembly class. A NetInstance is an instance of an assembly, whose type is the same as the assembly's type. In addition to having the same type as the assembly, the instance carries a transformation matrix, URL of the assembly location, unique name, and bounding box for the assembly.

public java.lang.String instanceName

This member is the unique name that will be associated with the instance.

public java.lang.String instanceType

This member is the same type as that of the assembly the instance represents.

public java.net.URL instanceURL

This member is the URL which shows the location of the assembly instance on the network.

public boolean show

This member is used to filter which assembly instances are to be displayed. If true the instance is displayed.

public boolean selected

This member is used to filter which assembly instances are to be selected. If true the instance is selected.

NetInstance()

This method is a default constructor which creates a new instance unassociated with any assembly.

NetInstance(NetMatrix,java.lang.String,double,double,double)

This method is a constructor which creates a new instance at the location of the specified transformation matrix, associated with the assembly having a type that is the same as the specified string object, having a bounding box with the specified xyz dimensions.

public void equals(NetInstance)

This method sets the various members to be the same as those of the specified instance.

public java.lang.String toString()

This method formats the members of the instance in an order for placement in a string data type for output purposes.

public java.lang.String toString(int)

This method is the same as the previous one, except that all values are rounded and truncated at the decimal place specified.

public void instance_transform(NetMatrix)

This method multiplies the matrix member by the specified transformation matrix. In this way the location of the assembly instance within the model may be changed, but the assembly itself keeps its own origin.

public NetMatrix get_matrix()

This method returns the transformation matrix.

public NetSolid getBoundingBox()

This method returns a cuboid solid which represents the bounding box of the assembly associated with the instance.

public void setBoundingBox(double,double,double,double,double,double)

This method sets the bounding box to the specified +x, -x, +y, -y, +z, and -z values.

public void setBoundingBox(NetSolid)

This method sets the bounding box to be the same as the specified solid.

CONSTRUCT CLASSES

Construct classes are the data types that define the building or artifact. They include classes that represent systems and methods that drive net-based automated construction. The four construct classes are NetGroup, NetSubsystem, NetSystem, and NetConstruct.

class NetGroup

The NetGroup class represents a special grouping within the artifact. An example in a building would be a floor of the building, where the group would include all components associated with the floor. Primary members include a list that can contain an unlimited number of NetInstances. Other members include the boolean show member and methods for manipulating the group.

public java.lang.String groupName

This member is the name of the group, such as “floor_2”.

public boolean show

This member is used to filter which groups are to be displayed. If true the group is displayed.

NetGroup()

This method is a default constructor which creates a new empty group.

public void equals(NetGroup)

This method sets the various members to be the same as those of the specified group, including the contents of the instance list.

public java.lang.String toString()

This method formats the members of the group in an order for placement in a string data type for output purposes.

public java.lang.String toString(int)

This method is the same as the previous one, except that all values are rounded and truncated at the decimal place specified.

public void group_transform(NetMatrix)

This method relocates each of the instances in the list to new locations as recalculated by the specified transformation matrix.

public int get_number()

This method returns the size of the instance list.

public void setAllSelected()

This method scrolls through the instance list and sets the boolean selected member of each instance to be true.

public void add_instance(NetInstance)

This method adds the specified instance to the instance list, and automatically names it according to the assembly type and list size. For example, if there are already two instances in the list and a new one of type WAL0200 is to be added, this method would rename the instance "WAL0200_3" before adding it to the list.

public void remove_instance(java.lang.String)

This method removes the instance with the specified name, and automatically goes down the list and renames the remaining members from the beginning according to the convention mentioned above.

public NetInstance extract_instance(java.lang.String)

This method returns the instance with the specified name.

public LinkList extractInstances()

This method returns a linked list containing all instances that have their show member set at true.

public LinkList extractALLinstances()

This method returns a copy of the instance list, containing all members regardless of whether the show attribute is set at true or not.

class NetSubsystem

The NetSubsystem class represents a subclass of a system. An example in a building would be floor framing of the building, which is a subsystem of the structure system. Primary members include a list that can contain an unlimited number of NetGroups. Other members include the boolean show member and methods for manipulating the subsystem.

public java.lang.String subName

This member is the subsystem name.

public boolean show

This member is used to filter which subsystems are to be displayed. If true the subsystem is displayed.

NetSubsystem()

This method is a default constructor which creates a new subsystem.

public void equals(NetSubsystem)

This method sets the various members to be the same as those of the specified subsystem, including the contents of the group list.

public java.lang.String toString()

This method formats the members of the subsystem in an order for placement in a string data type for output purposes.

public java.lang.String toString(int)

This method is the same as the previous one, except that all values are rounded and truncated at the decimal place specified.

public int get_number()

This method returns the size of the group list.

public void setAllSelected()

This method scrolls through the group list and calls the setAllSelected method of each group.

public void add_group(NetGroup)

This method adds specified group to group list.

public void remove_group(java.lang.String)

This method removes specified group from group list.

public NetGroup extract_group(java.lang.String)

This method returns specified group.

public LinkList extractInstances()

This method scrolls through the group list and calls the extractInstances method from each one, then returns a linked list containing the contents of each of the lists returned by the method.

public LinkList extractGroups()

This method returns a copy of the group list.

public void addInstance(NetInstance,java.lang.String,int)

This method adds the specified instance to the group having a name that matches the specified string object combined with the specified integer. For a building example, if an instance is to be added to the fifth floor, this method would scroll through the list and find the group with the name "floor_5" and call that group's addInstance method.

public void removeInstance(java.lang.String)

This method scrolls through the group list and calls the removeInstance method of each group, using the specified instance name.

class NetSystem

The NetSystem class represents a systems which make up a building or artifact. An example in a building would be the structural or mechanical systems. Primary members include a list that can contain an unlimited number of NetSubsystems. Other members include the boolean show member and methods for manipulating the system.

public java.lang.String systemType

This member is the system type, such as STRUCTURE or MECHANICAL.

public boolean show

This member is used to filter which systems are to be displayed. If true the system is displayed.

NetSystem()

This method is a default constructor which creates a new empty system.

public void equals(NetSystem)

This method sets the various members to be the same as those of the specified system, including the contents of the subsystem list.

public java.lang.String toString()

This method formats the members of the system in an order for placement in a string data type for output purposes.

public java.lang.String toString(int)

This method is the same as the previous one, except that all values are rounded and truncated at the decimal place specified.

public void set_type(java.lang.String)

This method sets the system type member to be the same as the specified string object.

public int get_number()

This method returns the subsystem list size.

public void setAllSelected()

This method scrolls through the subsystem list and calls the setAllSelected method of each subsystem.

public void add_subsystem(NetSubsystem)

This method adds specified subsystem to subsystem list.

public void remove_subsystem(java.lang.String)

This method removes specified subsystem from subsystem list.

public NetSubsystem extract_subsystem(java.lang.String)

This method returns specified subsystem.

public LinkList extractInstances()

This method scrolls through the subsystem list and calls the extractInstances method from each one, then returns a linked list containing the contents of each of the lists returned by the method.

public LinkList extractSubsystems()

This method returns a copy of the subsystem list.

public void addInstance(NetInstance,java.lang.String,int,java.lang.String)

This method adds the specified instance to the group having a name that matches the specified string object combined with the specified integer, within the specified subsystem.

public void removeInstance(java.lang.String)

This method scrolls through the subsystem list and calls the removeInstance method of each subsystem, using the specified instance name.

class NetConstruct

The NetConstruct class represents the building or artifact. Primary members include a list that can contain an unlimited number of NetSystems. Other members include methods for manipulating the construct or artifact. Though not implemented in this version, additional members may include methods for running robots and construction machines for assembling the actual building.

public java.lang.String constructName

This member is the construct or artifact's name.

NetConstruct()

This method is a default constructor which creates a new empty construct or virtual artifact.

public void equals(NetConstruct)

This method sets the various members to be the same as those of the specified construct, including the contents of the system list.

public java.lang.String toString()

This method formats the members of the construct in an order for placement in a string data type for output purposes.

public java.lang.String toString(int)

This method is the same as the previous one, except that all values are rounded and truncated at the decimal place specified.

public int get_number()

This method returns the number of systems in the system list.

public void add_system(NetSystem)

This method adds the specified system to the system list.

public void remove_system(java.lang.String)

This method removes the specified system from the system list.

public NetSystem extract_system(java.lang.String)

This method returns the specified system.

public LinkList extractInstances()

This method scrolls through the system list and calls the extractInstances method from each one, then returns a linked list containing the contents of each of the lists returned by the method.

public LinkList extractSystems()

This method returns a copy of the system list.

public void addInstance(NetInstance,String,int,String,String)

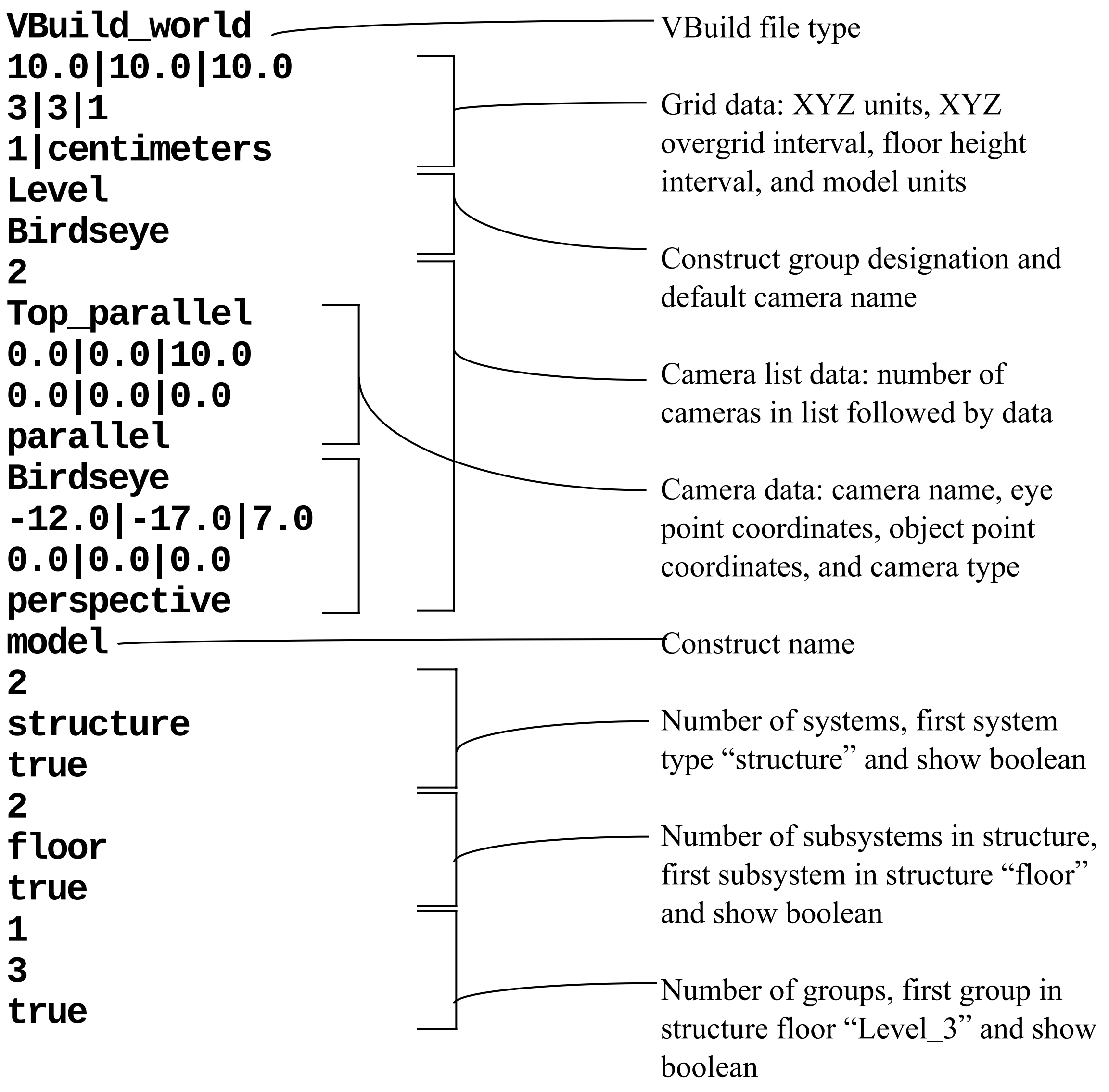
This method adds the specified instance to the group having a name that matches the specified string object combined with the specified integer, within the specified subsystem, within the specified system.

public void removeInstance(java.lang.String)

This method scrolls through the system list and calls the removeInstance method of each system, using the specified instance name.

APPENDIX D
VBuild File Formats

There are two kinds of VBuild files: world files and assembly files. The sample files here have been simplified with a limited number of members for clarity. The world file contains one construct and other information such as a list of cameras.



```
VBuild_world
10.0|10.0|10.0
3|3|1
1|centimeters
Level
Birdseye
2
Top_parallel
0.0|0.0|10.0
0.0|0.0|0.0
parallel
Birdseye
-12.0|-17.0|7.0
0.0|0.0|0.0
perspective
model
2
structure
true
2
floor
true
1
3
true
```

VBuild file type

Grid data: XYZ units, XYZ overgrid interval, floor height interval, and model units

Construct group designation and default camera name

Camera list data: number of cameras in list followed by data

Camera data: camera name, eye point coordinates, object point coordinates, and camera type

Construct name

Number of systems, first system type “structure” and show boolean

Number of subsystems in structure, first subsystem in structure “floor” and show boolean

Number of groups, first group in structure floor “Level_3” and show boolean

```
1
FLR2030
http://www.url.com
true
0.0|0.0|20.0
1.0|1.0|1.0
0.0|0.0|90.0
30.0|20.0|3.0
wall
true
1
1
true
2
WLL0020
http://www.url.com
true
0.0|0.0|0.0
1.0|1.0|1.0
0.0|0.0|90.0
3.0|20.0|20.0
WLL0020
http://www.url.com
true
30.0|0.0|0.0
1.0|1.0|1.0
0.0|0.0|90.0
3.0|20.0|20.0
electrical
...


...
3.0|20.0|20.0
```

Number of assembly instances in structure floor Level_3

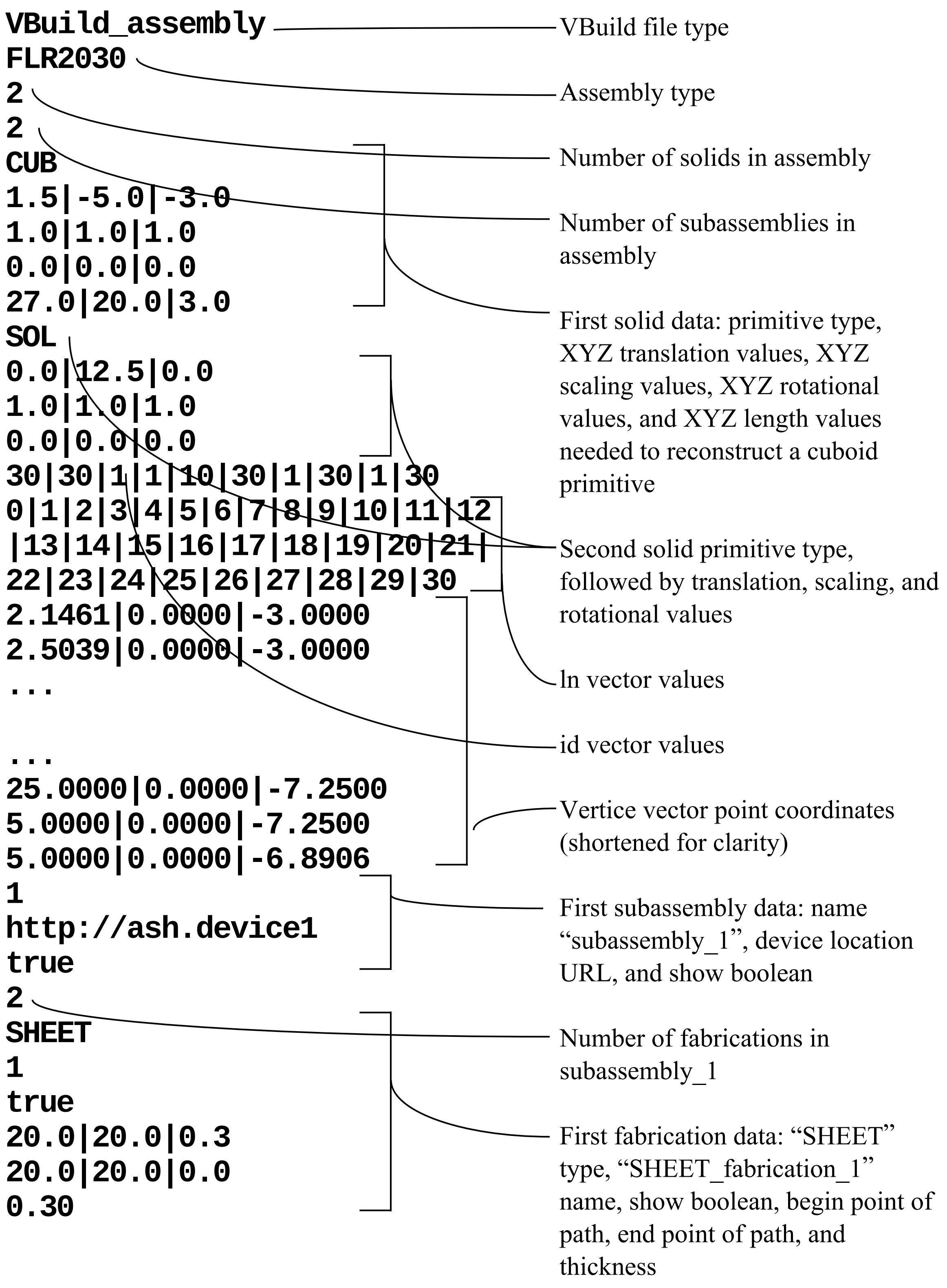
Assembly instance data: assembly type, location URL, show boolean, XYZ translation values, XYZ scaling values, XYZ rotation values, and bounding box XYZ length values

Second structure subassembly "wall" with member groups and assemblies

Second system type in construct "electrical", etcetera

If there are no more systems, the file ends with the bounding box XYZ values of the last assembly

The assembly files contain geometrical data for displaying the component, and also fabrication data needed for manufacturing itself.



```
VBuild_assembly
FLR2030
2
2
CUB
1.5|-5.0|-3.0
1.0|1.0|1.0
0.0|0.0|0.0
27.0|20.0|3.0
SOL
0.0|12.5|0.0
1.0|1.0|1.0
0.0|0.0|0.0
30|30|1|1|10|30|1|30|1|30
0|1|2|3|4|5|6|7|8|9|10|11|12
|13|14|15|16|17|18|19|20|21|
22|23|24|25|26|27|28|29|30
2.1461|0.0000|-3.0000
2.5039|0.0000|-3.0000
...

...
25.0000|0.0000|-7.2500
5.0000|0.0000|-7.2500
5.0000|0.0000|-6.8906
1
http://ash.device1
true
2
SHEET
1
true
20.0|20.0|0.3
20.0|20.0|0.0
0.30
```

VBuild file type

Assembly type

Number of solids in assembly

Number of subassemblies in assembly

First solid data: primitive type, XYZ translation values, XYZ scaling values, XYZ rotational values, and XYZ length values needed to reconstruct a cuboid primitive

Second solid primitive type, followed by translation, scaling, and rotational values

ln vector values

id vector values

Vertice vector point coordinates (shortened for clarity)

First subassembly data: name "subassembly_1", device location URL, and show boolean

Number of fabrications in subassembly_1

First fabrication data: "SHEET" type, "SHEET_fabrication_1" name, show boolean, begin point of path, end point of path, and thickness

```
acrylic
cut_sheet
polygon
4|1|4
0.0000|0.0000|0.0000
20.0000|0.0000|0.0000
20.0000|20.0000|0.0000
0.0000|20.0000|0.0000
SHEET
2
true
20.0|20.0|0.3
20.0|20.0|0.0
0.30
acrylic
cut_sheet
polygon
4|1|4
0.0000|0.0000|0.0000
20.0000|0.0000|0.0000
20.0000|20.0000|0.0000
0.0000|20.0000|0.0000
2
http://ash.device2
true
...

...
30.0000|20.0000|0.0000
```

SHEET_fabrication_1 part material

Part name “cut_sheet”

Part polygon name (part solid is not used so it does not appear in the file

Polygon id vector values

Polygon vertice point coordinates

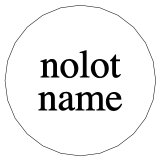
Second fabrication in subassembly_1

Second subassembly named “subassembly_2”, etcetera

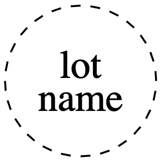
If there are no more subassemblies, the file ends with the last vertice point coordinates of the last part’s polygon

APPENDIX E
Guide for Reading Simplified NIAM Diagrams

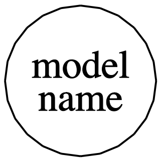
The following is a guide for reading the simplified Nijssen Information Analysis Method (NIAM) binary semantic models.



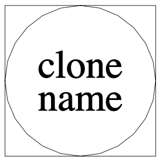
NOLOT: Non-lexical objects represent a set of non-representable entities having common properties. The NOLOT can be a class of objects such as those used in VBuild: point, fabrication, assembly, etcetera. The symbol for a NOLOT is a solid circle containing the NOLOT name.



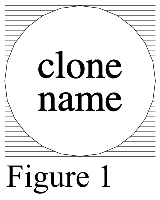
LOT: Lexical objects represent a set of values of an entity, such as names and properties. The symbol for a LOT is a dashed circle containing the LOT name.



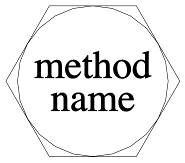
MODEL: The main NOLOT of the NIAM diagram is the MODEL. The remainder of the diagram usually supports its definition. The symbol for a MODEL is a heavy circle containing the MODEL name. This is a NIAM extension proposed by [Turner 1991] and adopted in this work.



CLONE: A CLONE is an object that appears elsewhere on the current NIAM model. The symbol for a clone is a square enclosing either a LOT or a NOLOT. This is another Turner extension, slightly revised.



CLONE: A second form of the CLONE is an object which appears in another NIAM model. The rendered square may enclose either a LOT or a NOLOT, and will have a reference to the figure number which more fully describes the object. This is another Turner extension, slightly revised.

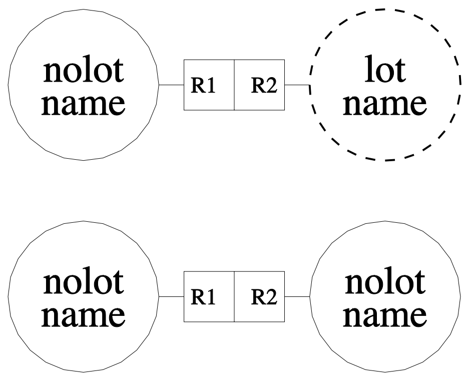


METHOD: A METHOD is an object which is a member method associated with the main NOLOT of the NIAM diagram. The symbol is a hexagon which encloses a solid circle containing the name of the method. This is an extension of NIAM, and may not support the binary semantic model definition.

Figure 136: NIAM objects

The guide is simplified from work done by [Turner 1991] with a few extensions. NIAM modeling is essentially a way of representing objects and their relationships in a graphical way. An attractive feature of NIAM modeling is the way the diagrams can be read almost like natural language [Mark 1987]. The diagrams contained in this work use a simplified version of NIAM for the purposes of graphically describing VBuild objects and object instances, and are not complete binary semantic models.

As used within the context of this work, objects are programming class structures and data types that represent tangible or abstract entities. Objects can also represent members within the class structures and data types, such as methods. Objects are described in Figure 136.



BRIDGE: A BRIDGE is a role between a LOT and a NOLOT. In this example, the NOLOT plays role R1 with the LOT, and the LOT plays role R2 with the NOLOT.

IDEA: A role between two NOLOTs is an IDEA. In this example, the left NOLOT plays role R1 with the right NOLOT, and vise versa with role R2.

Figure 137: NIAM roles

Roles are relationships or associations between two objects. The symbol for BRIDGE or IDEA is a box located on the axis between the two objects. The box is split into two and contain roles which occur between the objects. The binary relationship is read in either direction, with the role contained in the closest half of the BRIDGE or

IDEA box inserted into a sentence similar to natural language. In Figure 137, the diagram is read “LOT plays role R2 with NOLOT” and “NOLOT plays role R1 with LOT”.

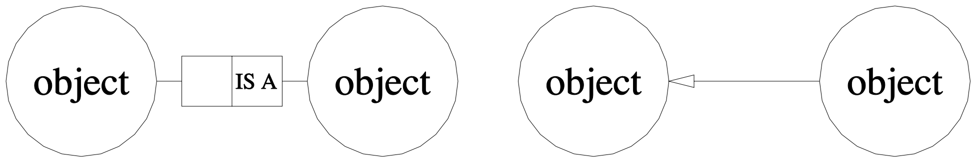


Figure 138: NIAM "is a" role

The “is a” role is a common relationship between NOLOTs. Therefore, a special symbol in the form of a directed line segment is provided. This sets up a system of object subtypes and supertypes. In Figure 139, the object supertype is “person”. Subtypes of person can be “woman” or “man”. In this example the man has the name “Jack”.

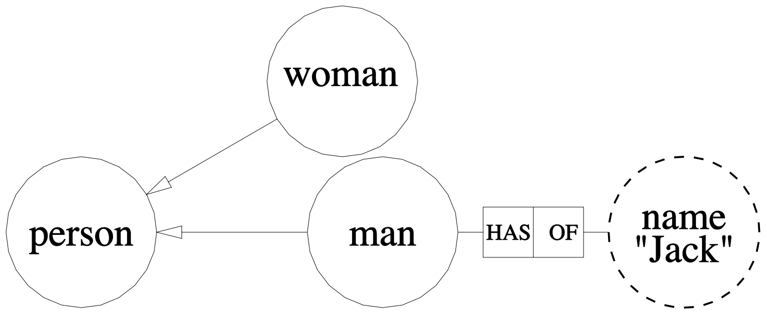


Figure 139: NIAM subtyping

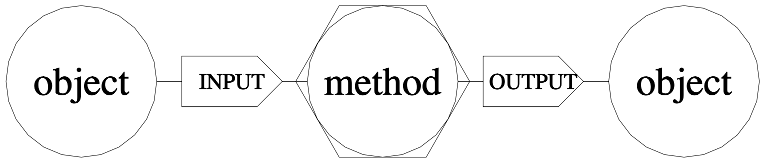


Figure 140: Methods and actions

Another form of relationship developed within the context of this work is the ACTION relationship. The ACTION relationship defines the role between a NOLOT and a METHOD. In most cases, the ACTION role describes a manipulation of NOLOT members within a model, and the result. In Figure 140, the object on the left is used as input to produce or alter the state of the object on the right. Either the left or right objects may be members within the model.

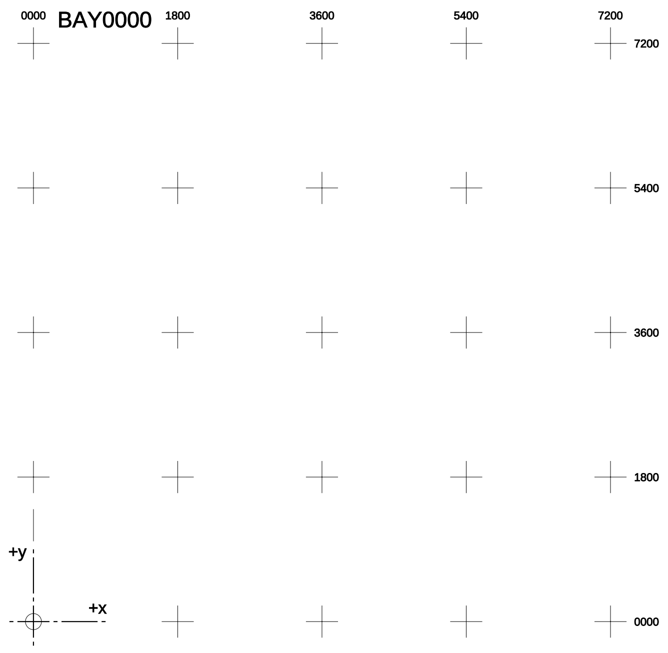
The METHOD object and ACTION role are tentative types and may or may not depart from the binary semantic model. The reason for this was the need to represent method members in a purely object-oriented fashion as fellow objects. Methods which return a certain object or data type can themselves be mapped into that data type as an object, but with unsatisfactory representation. In such a case the relationship within the model would not map well using a BRIDGE or an IDEA, therefore the ACTION symbol was devised.

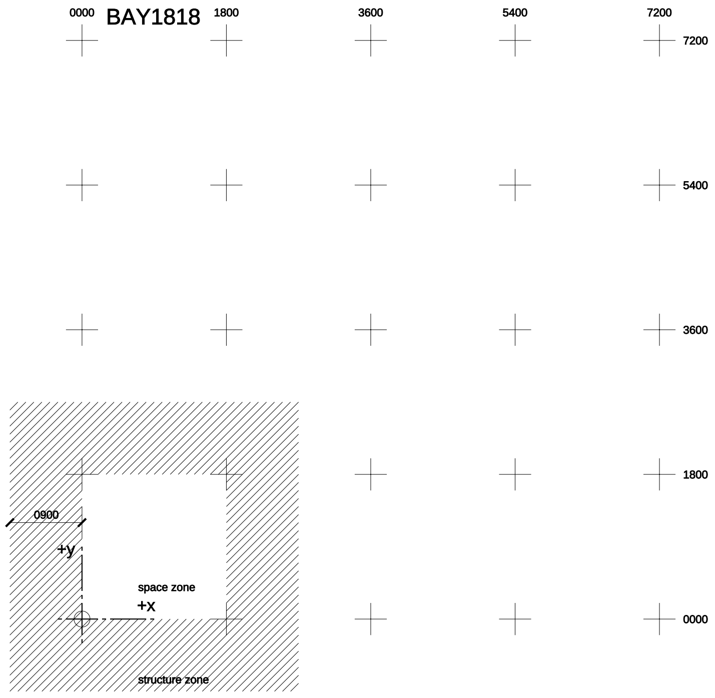
APPENDIX F
Life-cycle Architecture System Kit-of-parts

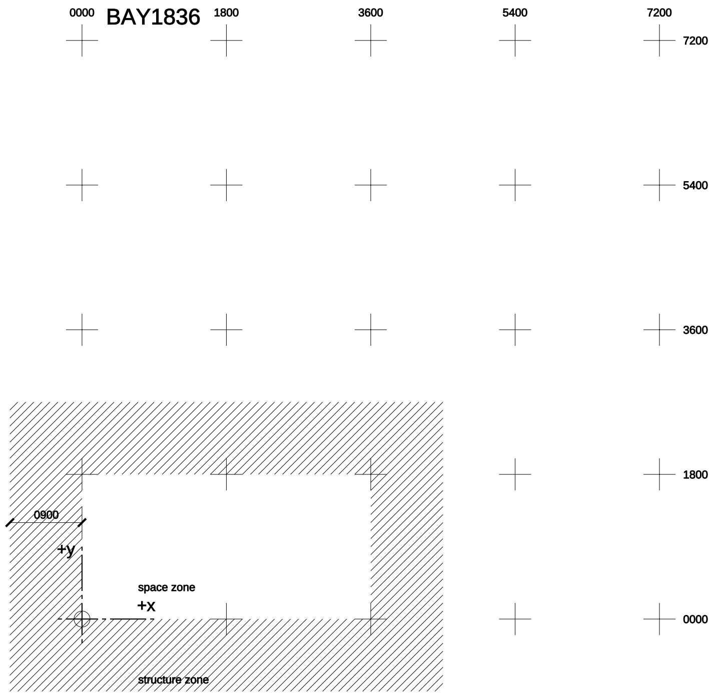
The following is a kit-of-parts library modeled for the IMS Gnosis soft factory Emerald project. The library is not complete, but contains only enough components to provide an enclosure. Doors, windows, and other cladding components, as well as mechanical and other system components may be derived from these components or should be added to the library at a later time. The components are in alphabetical order.

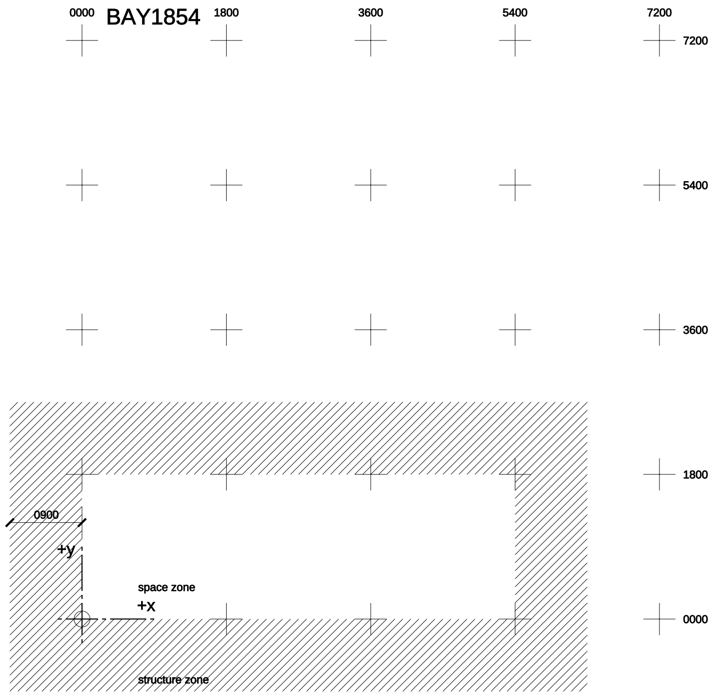
Spatial components

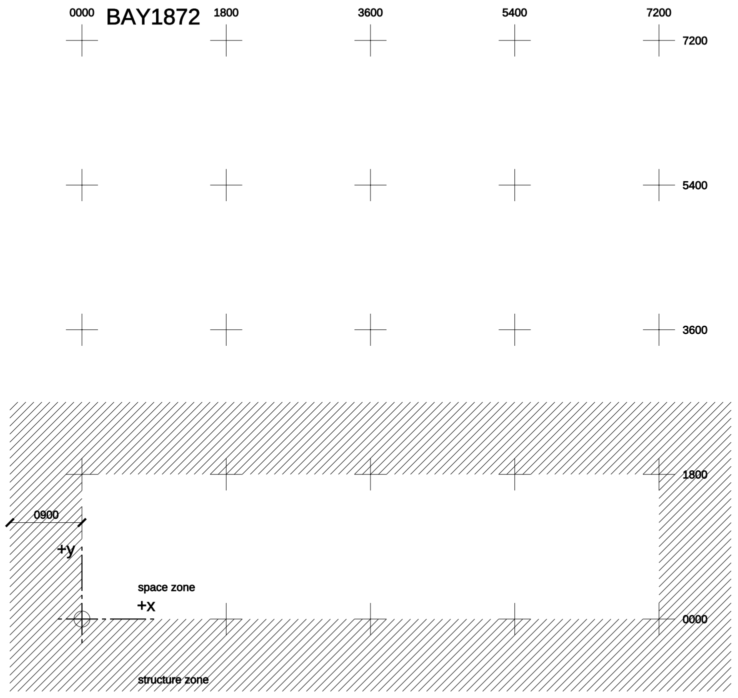
Each Spatial component has a unique name derived from the bay size itself. The naming convention used is BAY0000, where the zeros would be replaced by X and Y nominal dimensions of the space. Spatial components are shown layed out on the following graphical grid, in the space zone / structure zone fashion.

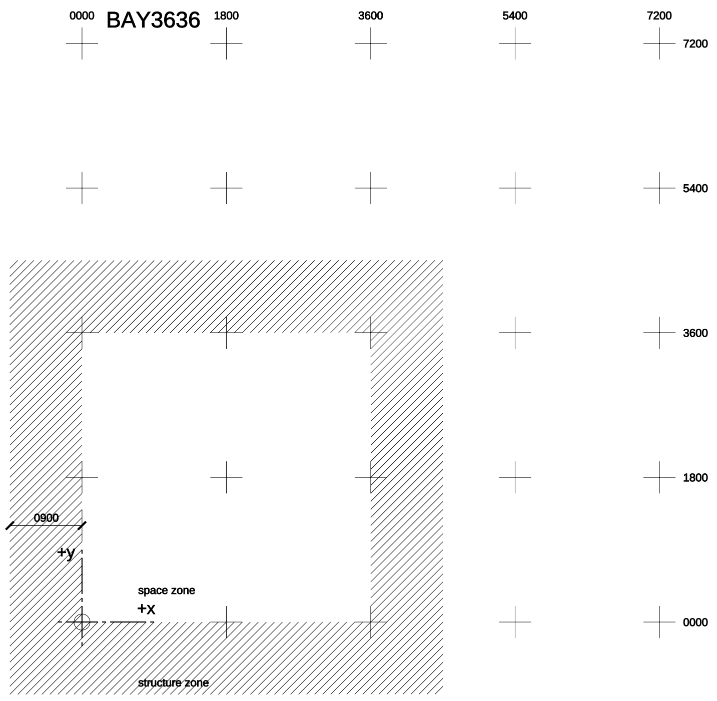


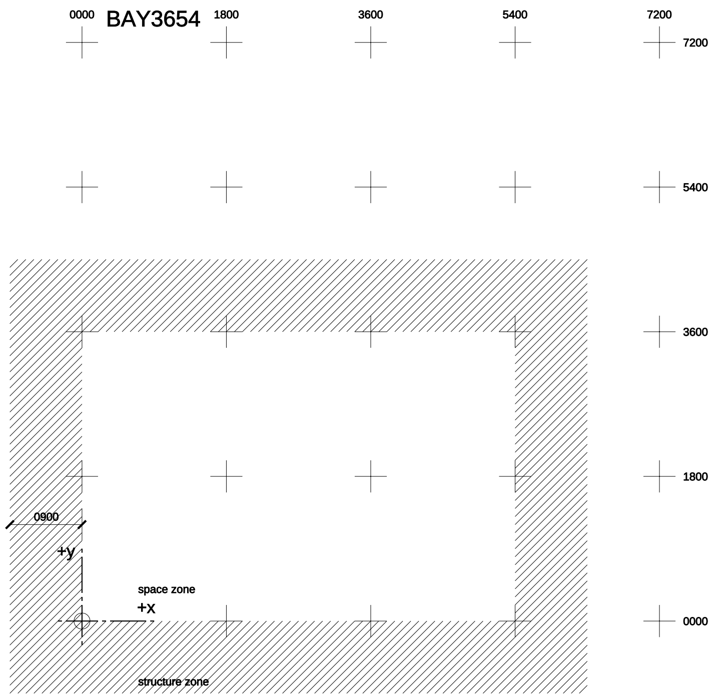

BAY1818
0000
1800
3600
5400
7200
7200
5400
3600
1800
0000
0900
+y
+x
space zone
structure zone

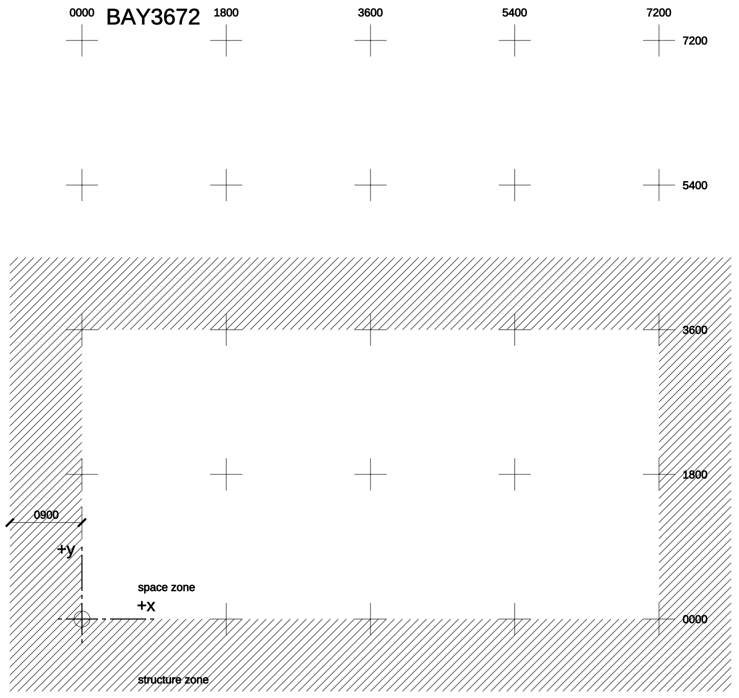

BAY1836
0000
1800
3600
5400
7200
7200
5400
3600
1800
0000
0900
+y
+x
space zone
structure zone

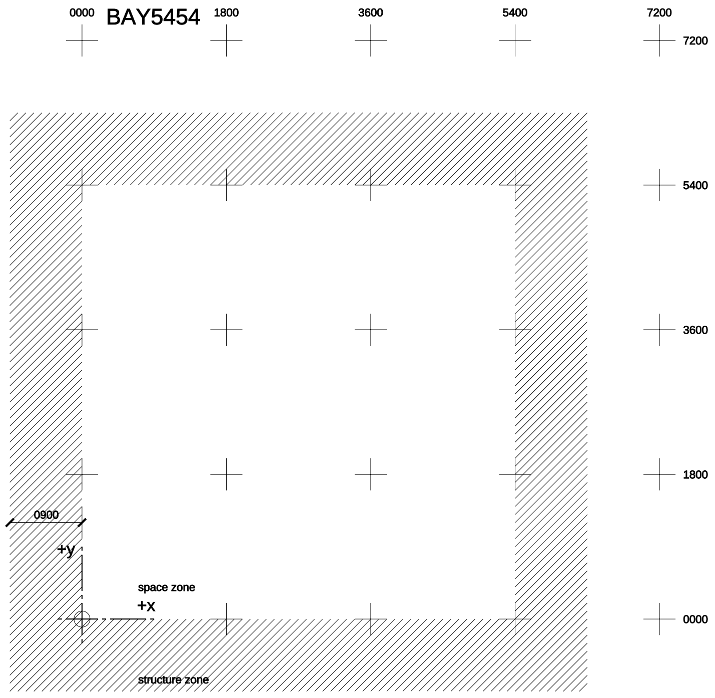

0000
BAY1854
1800
3600
5400
7200
7200
5400
3600
1800
0000
0900
+y
space zone
+x
structure zone

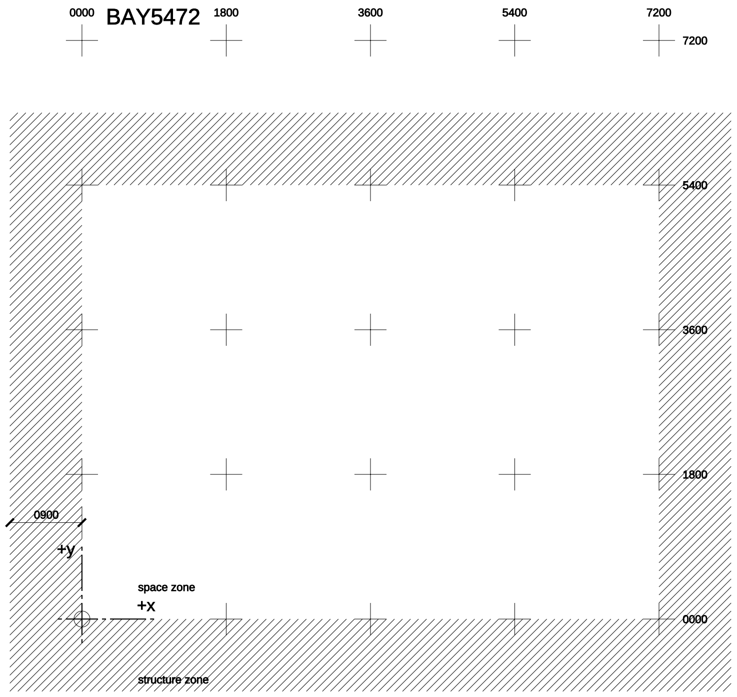

0000
BAY1872
1800
3600
5400
7200
7200
5400
3600
1800
0000
0900
+y
space zone
+x
structure zone

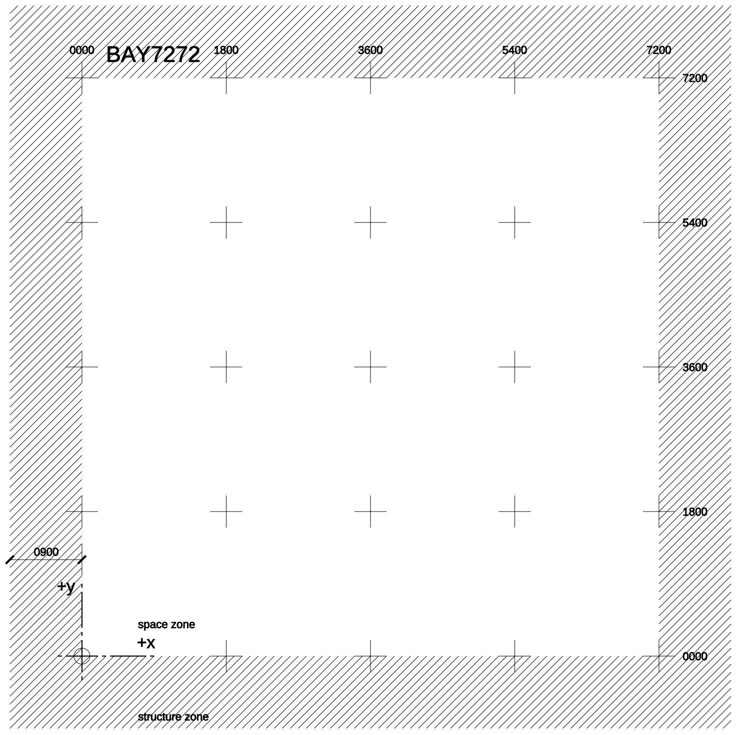

BAY3636
0000
1800
3600
5400
7200
7200
5400
3600
1800
0000
0900
+y
+x
space zone
structure zone


BAY3654
0000
1800
3600
5400
7200
7200
5400
3600
1800
0000
0900
+y
+x
space zone
structure zone


0000
BAY3672
1800
3600
5400
7200
7200
5400
3600
1800
0900
+y
space zone
+x
0000
structure zone

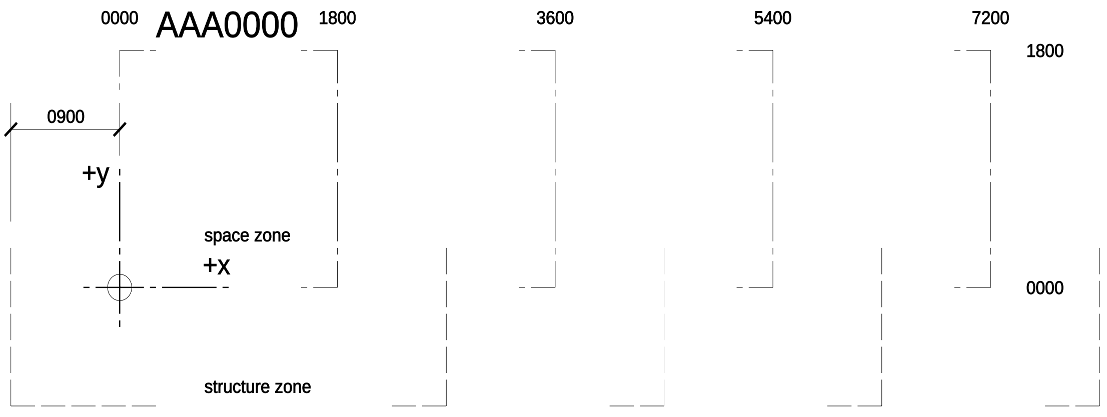

0000
BAY5454
1800
3600
5400
7200
7200
5400
3600
1800
0900
+y
space zone
+x
0000
structure zone


BAY5472
0000
1800
3600
5400
7200
7200
5400
3600
1800
0000
0900
+y
+x
space zone
structure zone

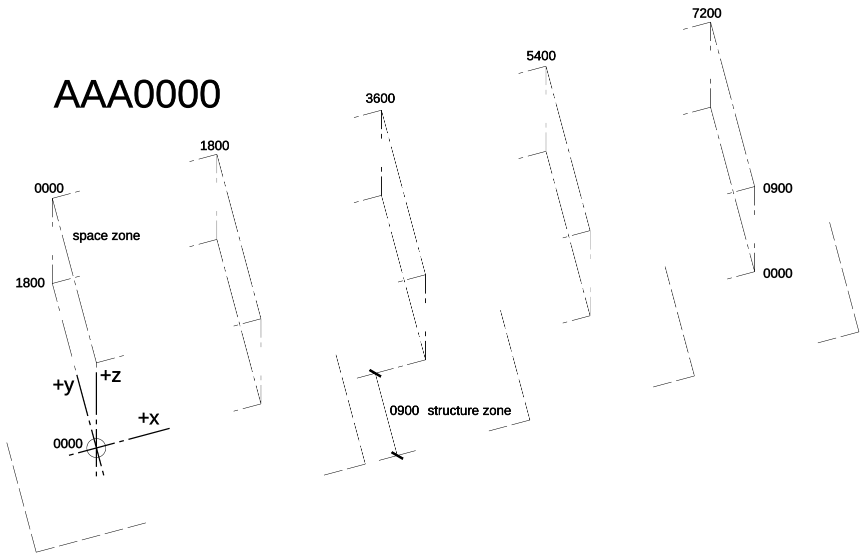

BAY7272
0000
1800
3600
5400
7200
7200
5400
3600
1800
0000
0900
+y
+x
space zone
structure zone

Structure and cladding components

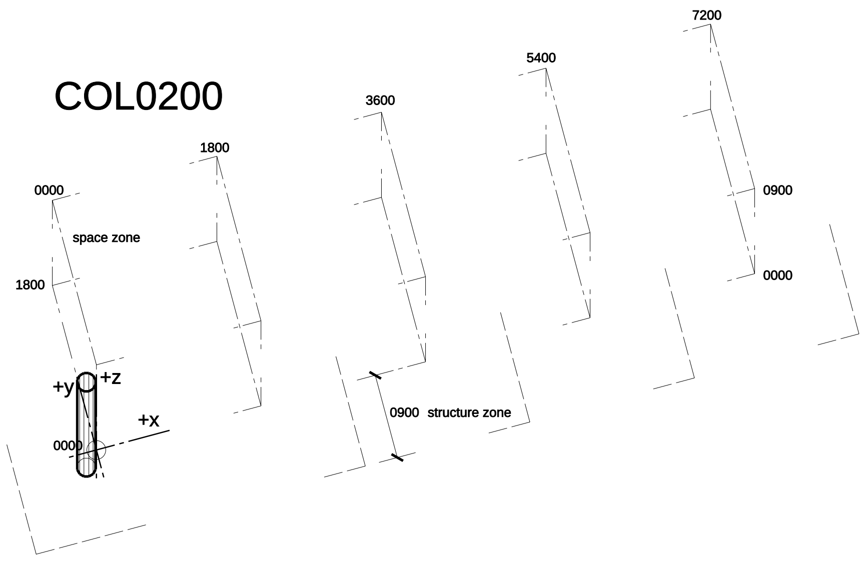
Each component has a unique name derived from the component size itself. The naming convention used is AAA0000, where the zeros would be replaced by X and Y nominal dimensions of the space and the AAA would be replaced by a letter combination that hints of the nature of the component's use. Some of the components are scalable in the Z or vertical direction for the purposes of simplifying the model. Flat components are shown layed out on the following graphical grid.



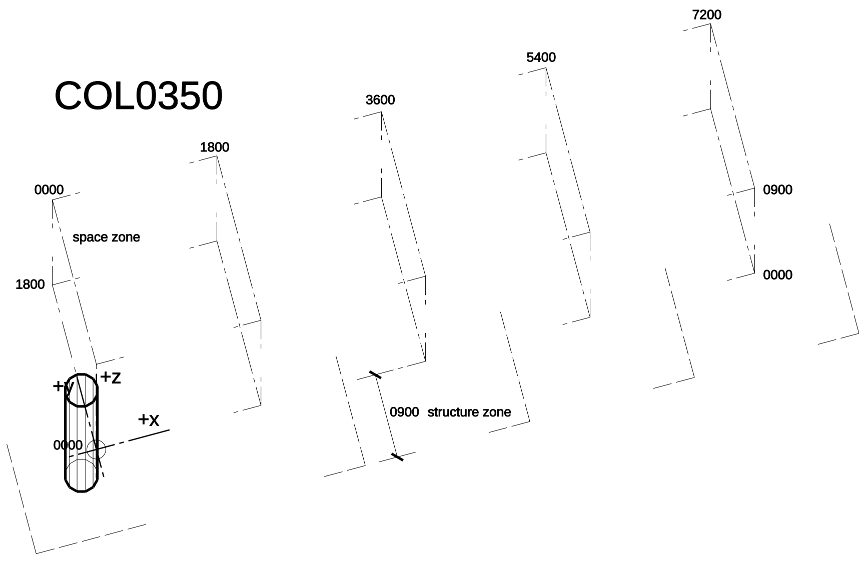
Three-dimensional components are shown layed out on the following graphical grid.



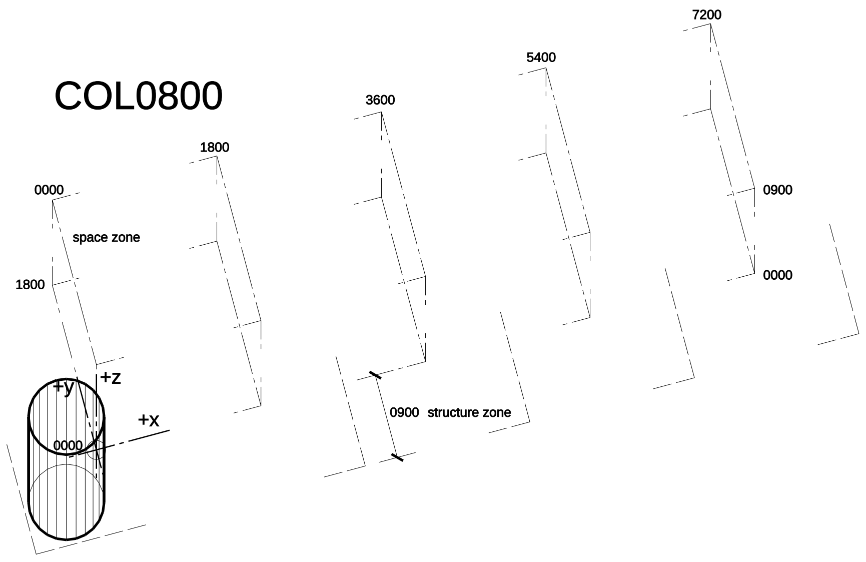
COL0200

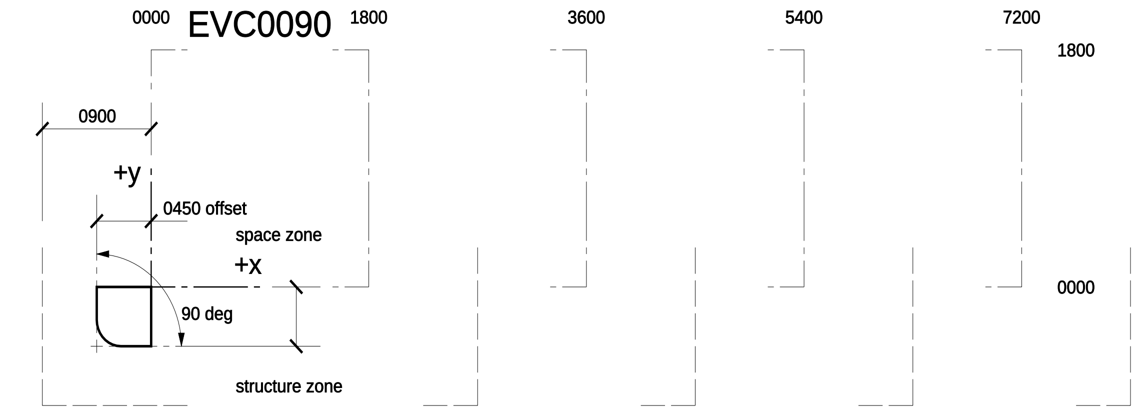


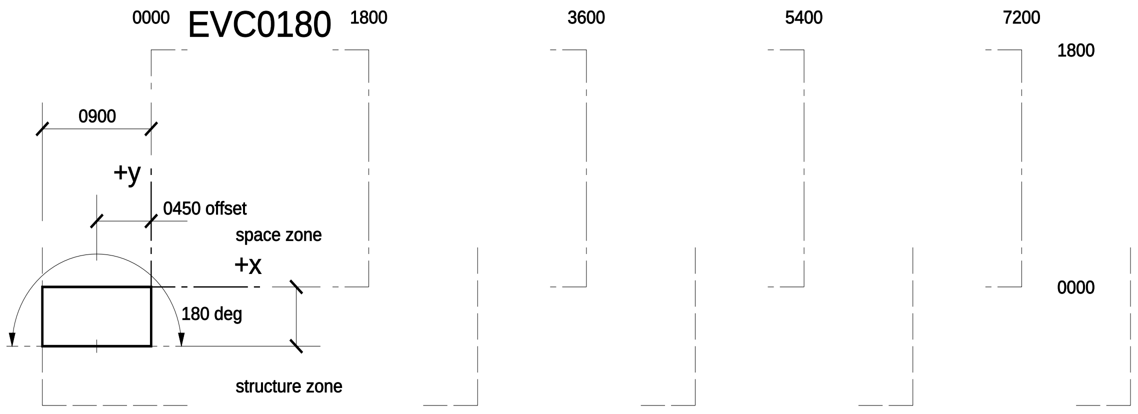
COL0350

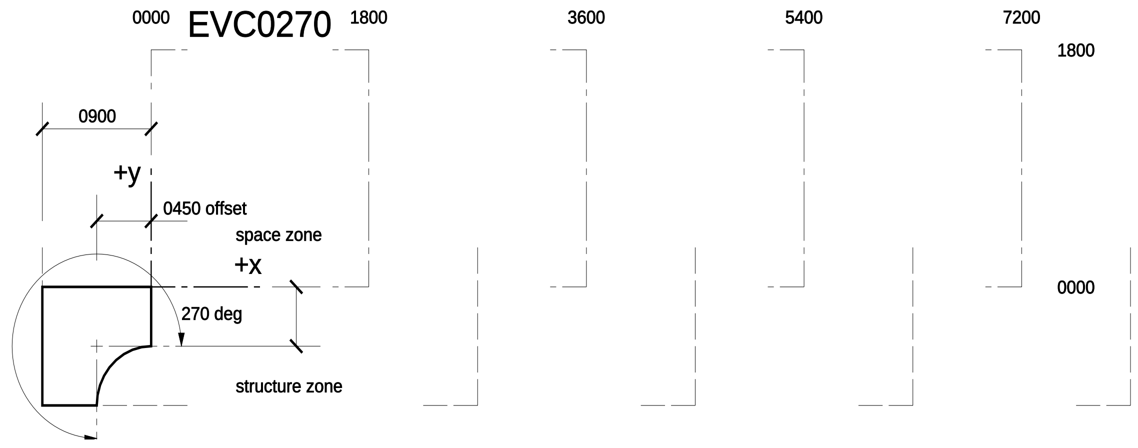


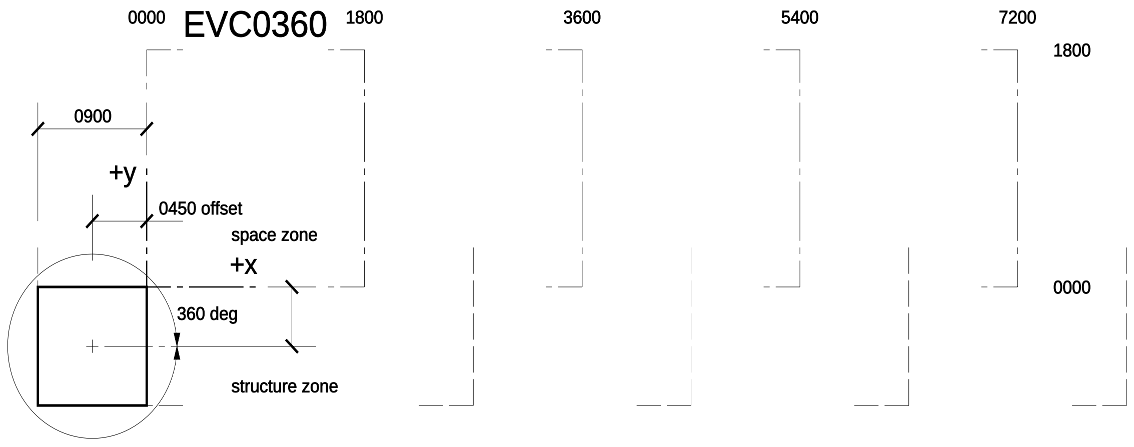
COL0800

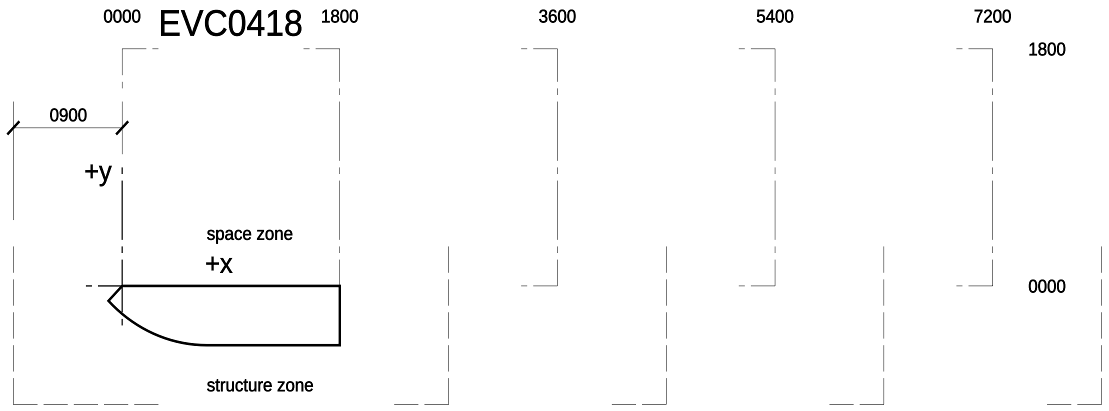


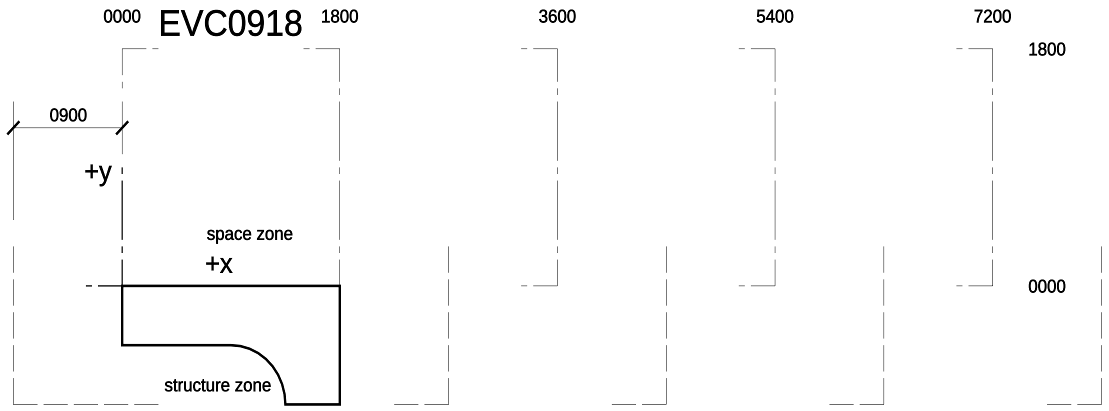

0000
EVC0090
1800
3600
5400
7200
1800
0900
+y
0450 offset
space zone
+x
0000
90 deg
structure zone

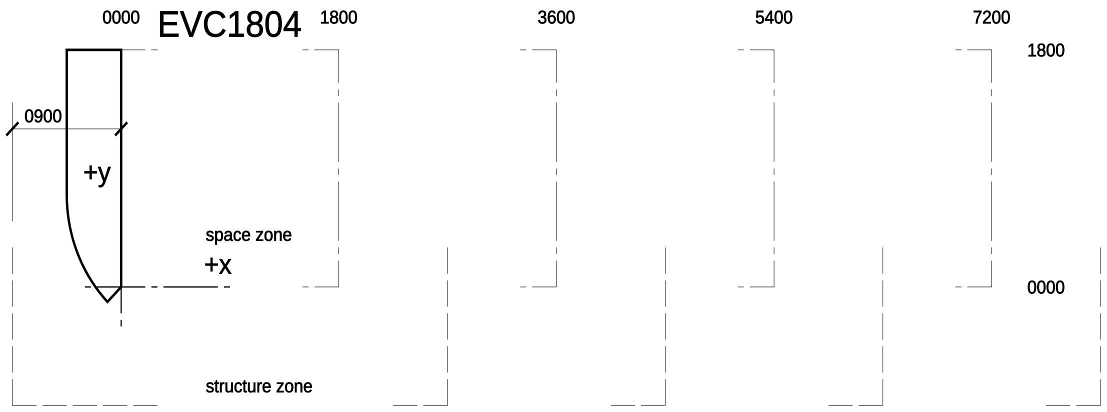

0000
EVC0180
1800
3600
5400
7200
1800
0900
+y
0450 offset
space zone
+x
0000
180 deg
structure zone

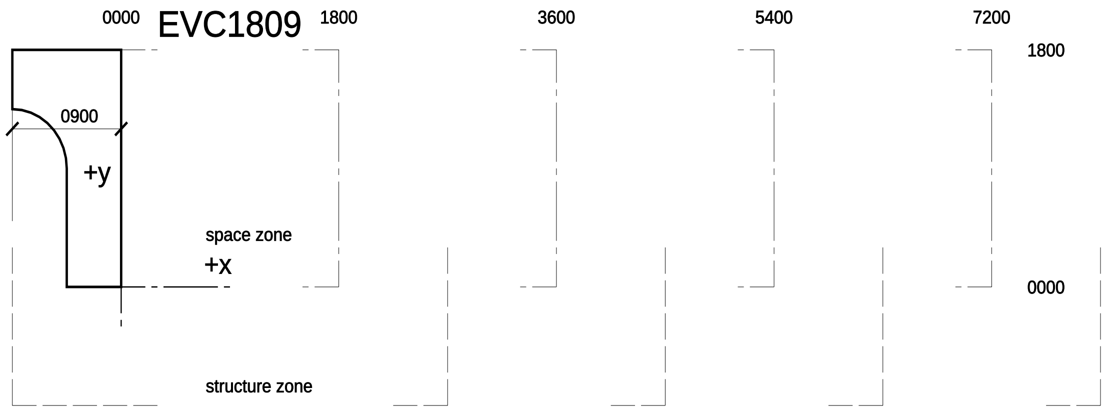

0000
EVC0270
1800
3600
5400
7200
1800
0900
+y
0450 offset
space zone
+x
0000
270 deg
structure zone

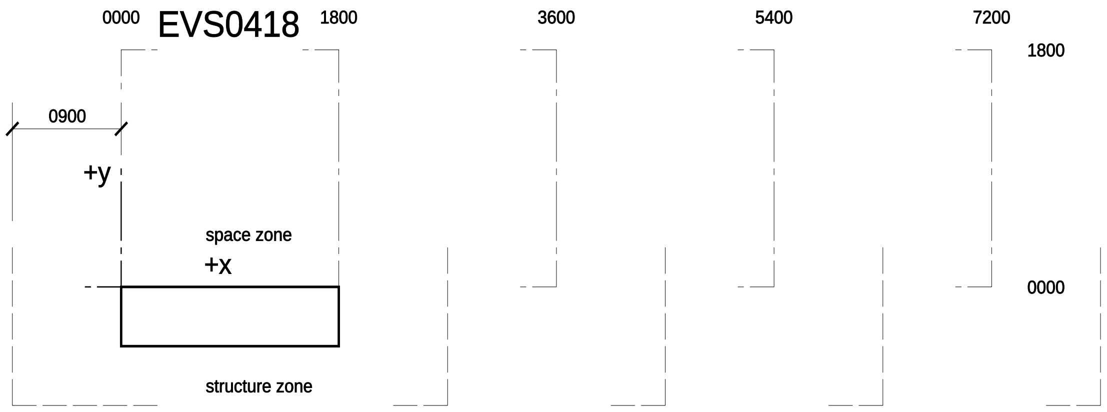

0000
EVC0360
1800
3600
5400
7200
1800
0900
+y
0450 offset
space zone
+x
0000
360 deg
structure zone

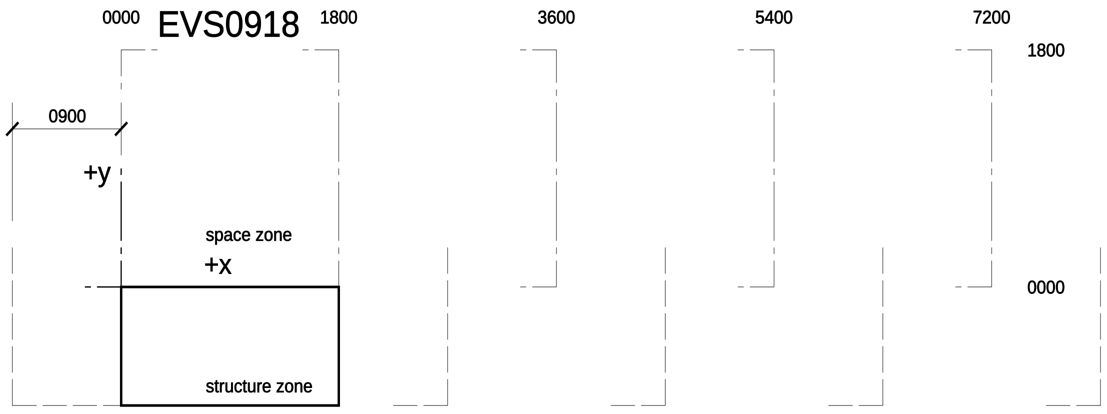

0000
EVC0418
1800
3600
5400
7200
1800
0900
+y
space zone
+x
0000
structure zone

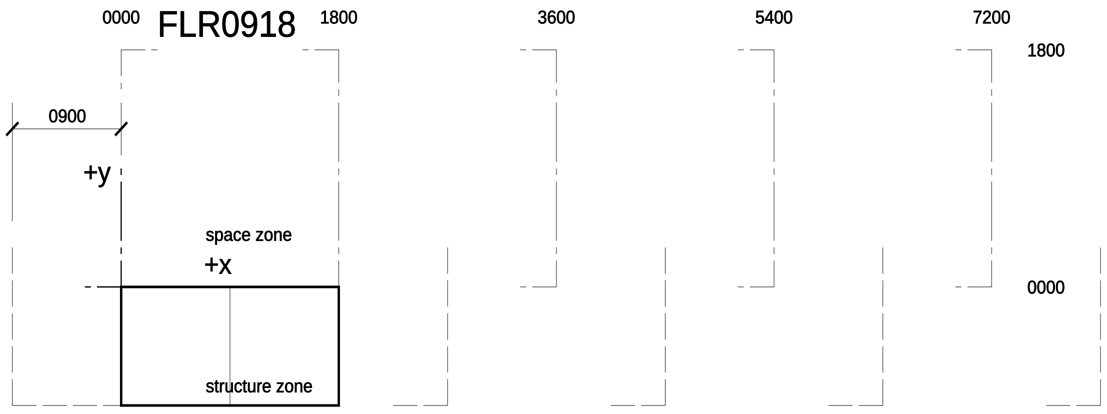

0000
EVC0918
1800
3600
5400
7200
1800
0900
+y
space zone
+x
0000
structure zone

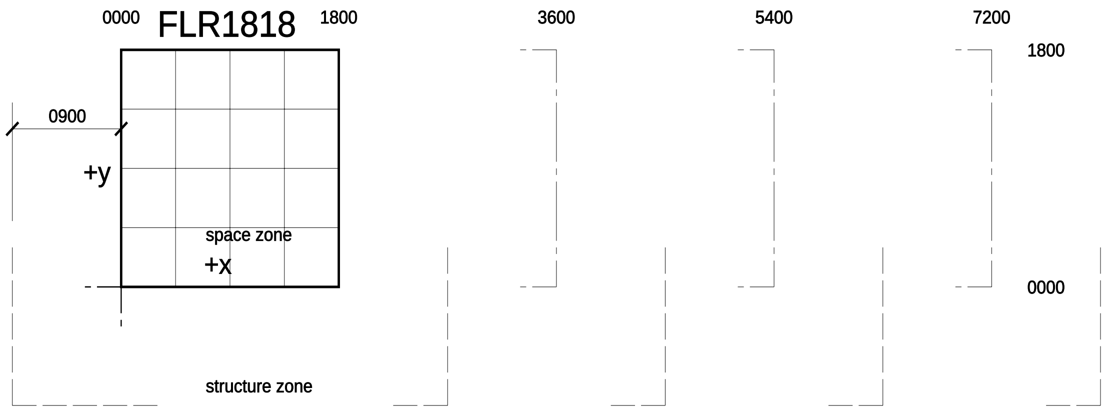

0000
EVC1804
1800
3600
5400
7200
1800
0900
+y
space zone
+x
0000
structure zone


0000
EVC1809
1800
3600
5400
7200
1800
0900
+y
space zone
+x
0000
structure zone

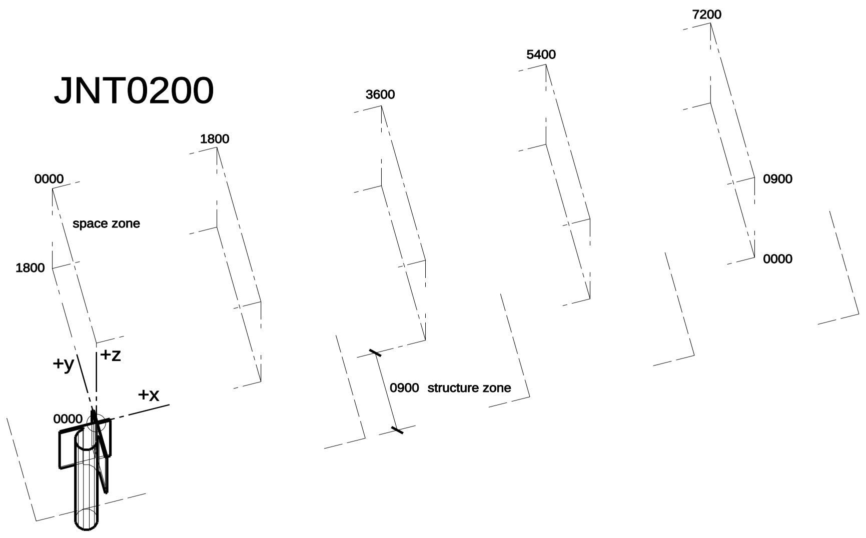

0000
EVS0418
1800
3600
5400
7200
1800
0900
+y
space zone
+x
0000
structure zone


0000
EVS0918
1800
3600
5400
7200
1800
0900
+y
space zone
+x
0000
structure zone

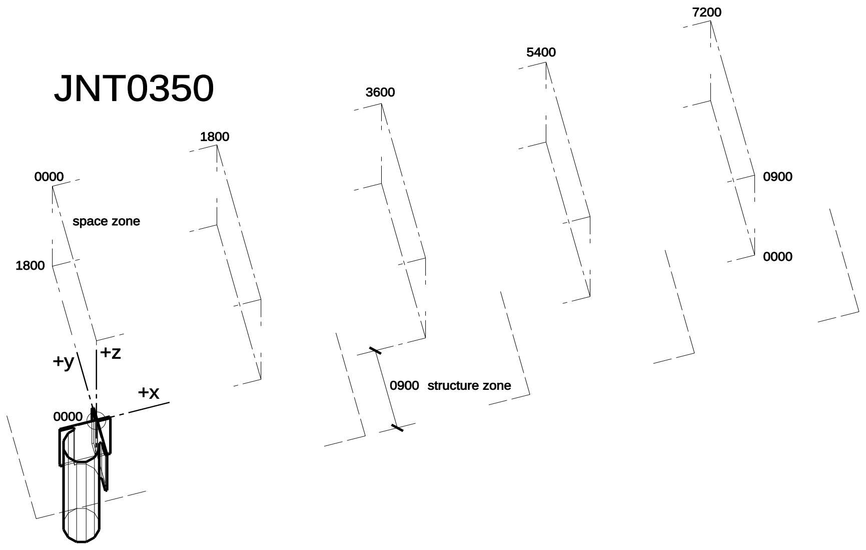

0000
FLR0918
1800
3600
5400
7200
1800
0900
+y
space zone
+x
0000
structure zone


0000
FLR1818
1800
3600
5400
7200
1800
0900
+y
space zone
+x
0000
structure zone

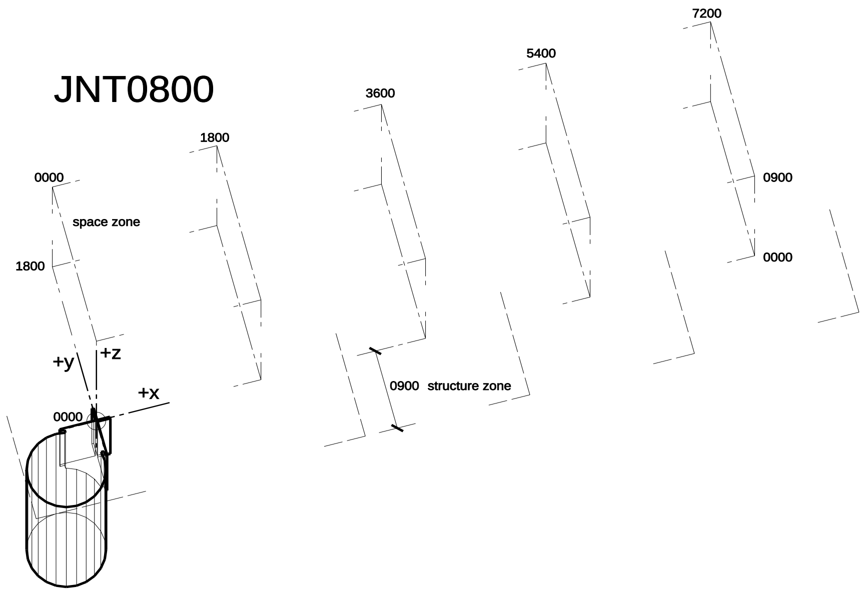
JNT0200

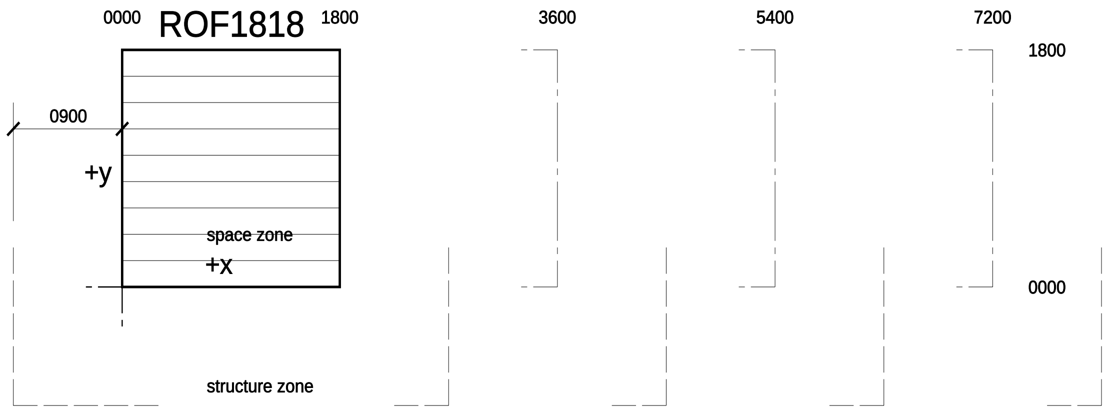


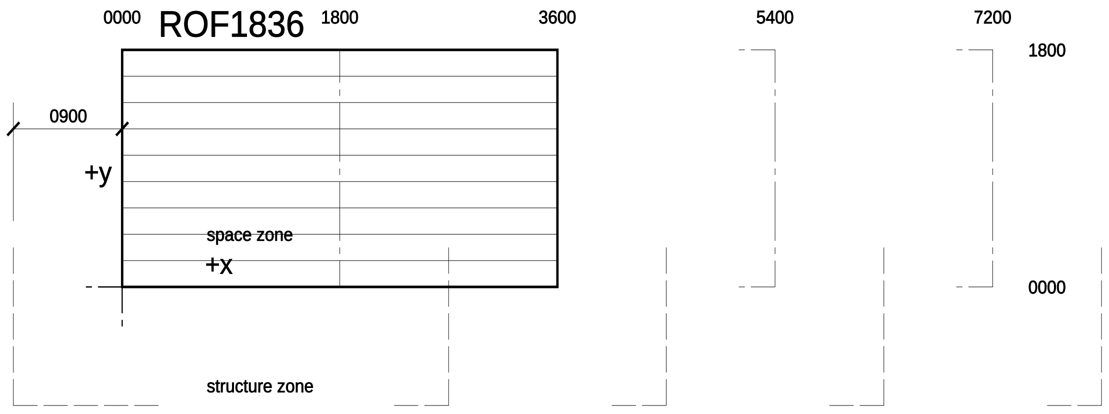
JNT0350

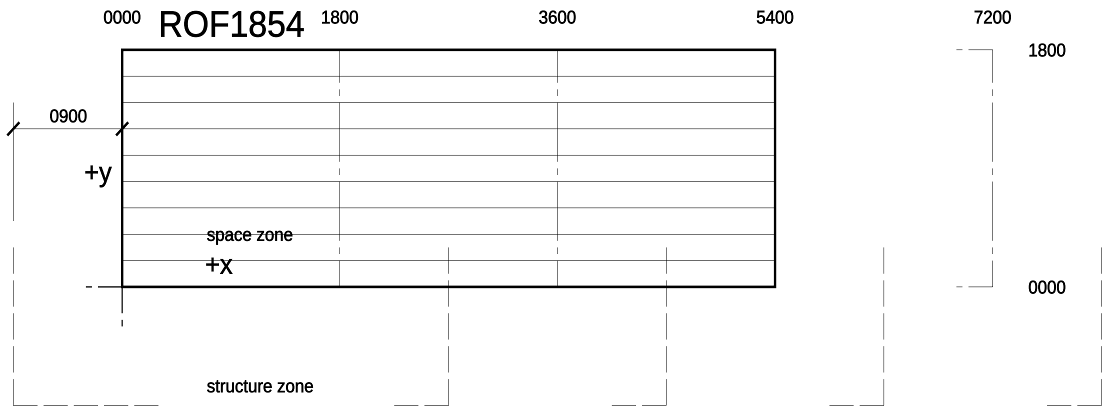


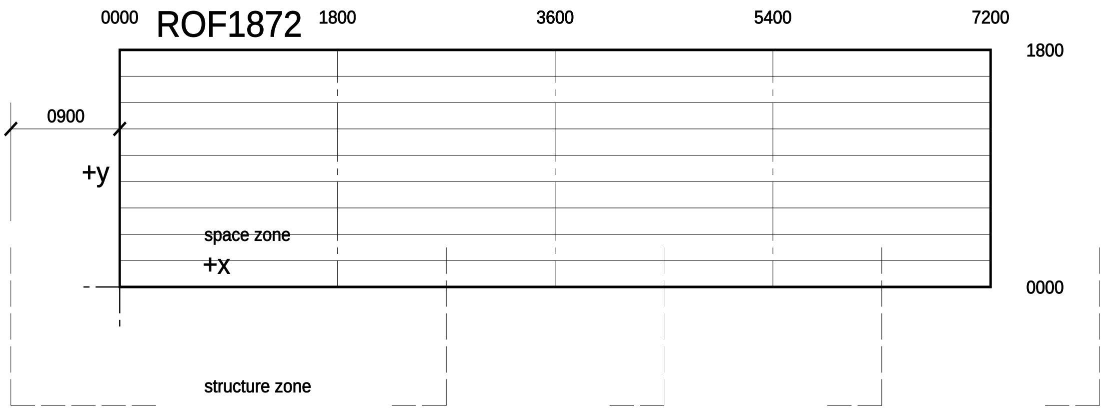
JNT0800

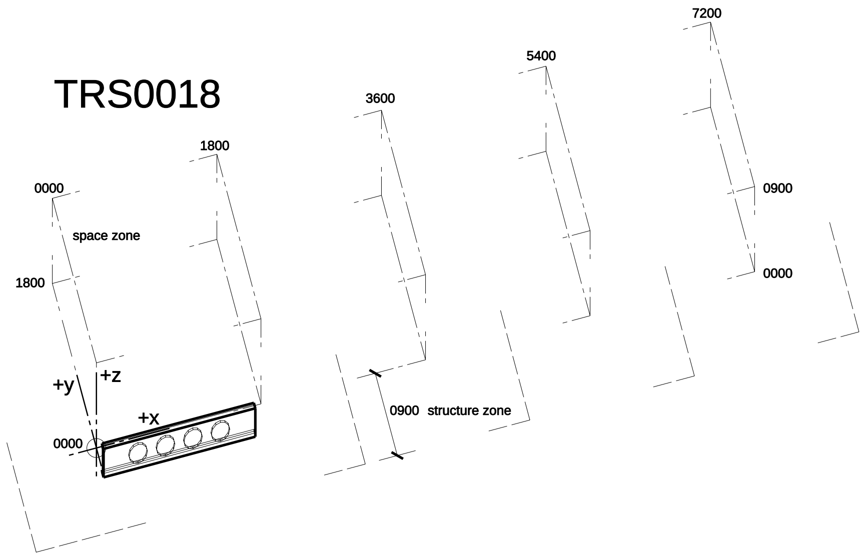


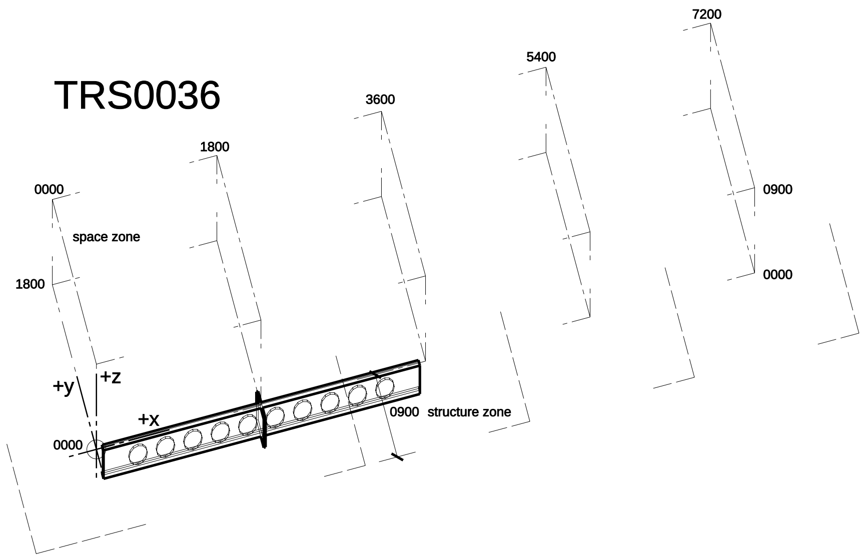

0000
ROF1818
1800
3600
5400
7200
1800
0900
+y
space zone
+x
0000
structure zone


0000
ROF1836
1800
3600
5400
7200
1800
0900
+y
space zone
+x
0000
structure zone

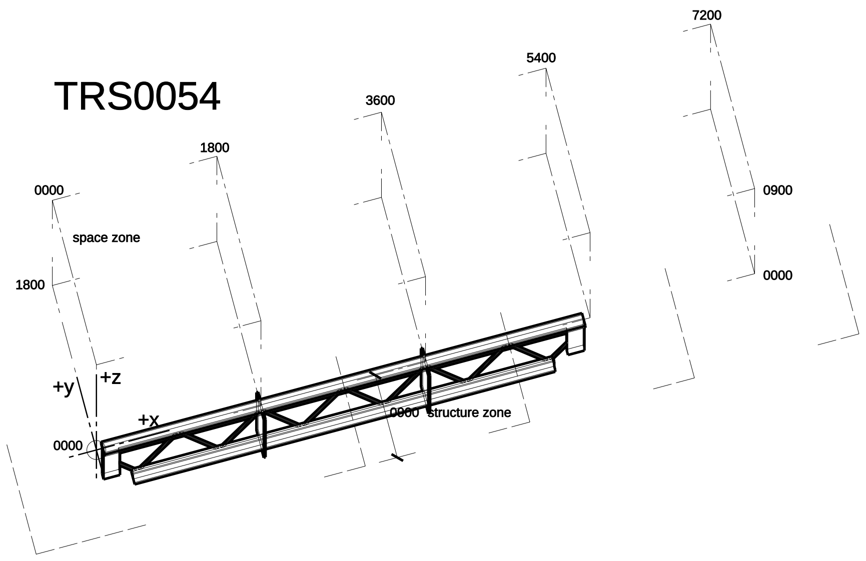

0000
ROF1854
1800
3600
5400
7200
1800
0900
+y
space zone
+x
0000
structure zone


0000
ROF1872
1800
3600
5400
7200
1800
0900
+y
space zone
+x
0000
structure zone

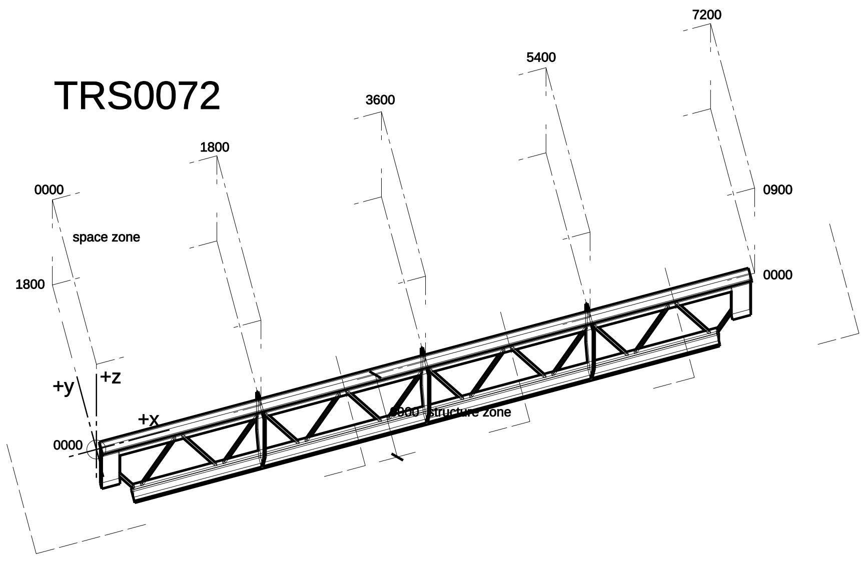

TRS0018
7200
5400
3600
1800
0000
0900
space zone
1800
0000
+y
+z
+x
0000
0900
structure zone


TRS0036
7200
5400
3600
1800
0000
0900
space zone
1800
0000
+y
+z
+x
0000
0900
structure zone

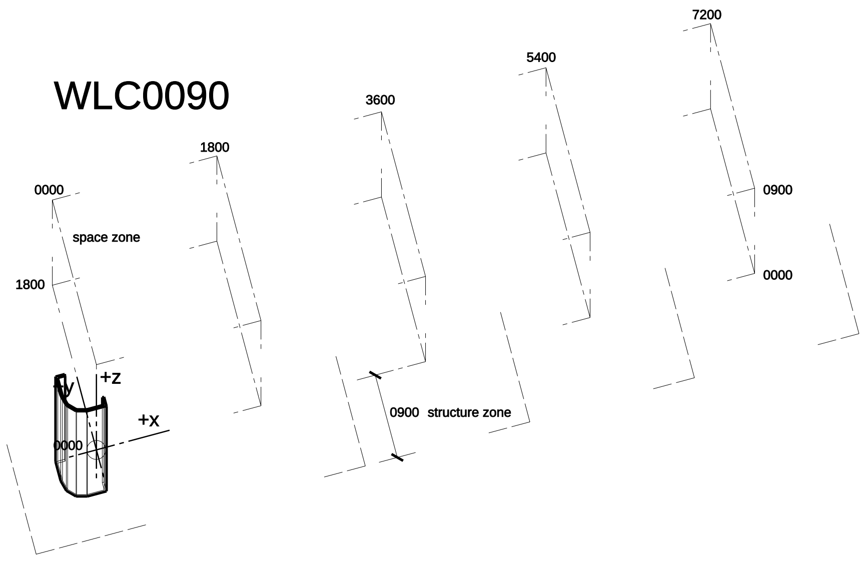
TRS0054



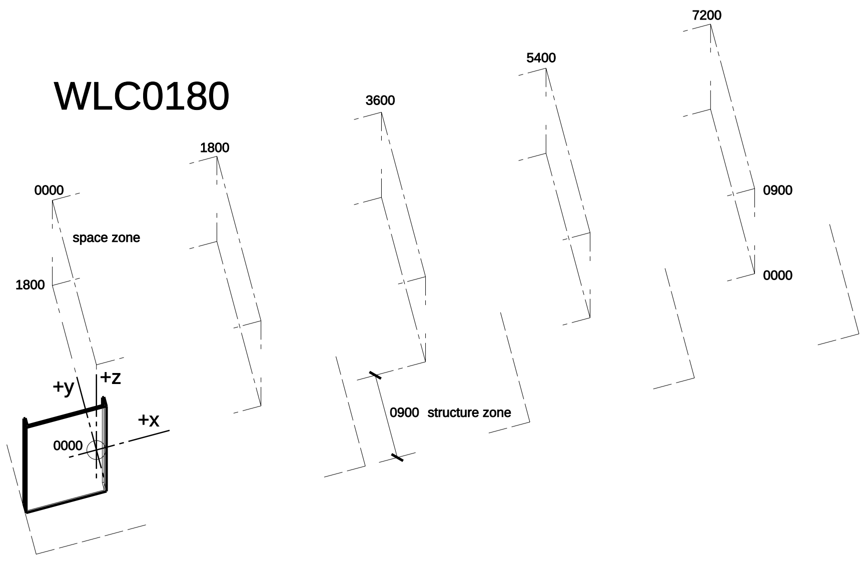
TRS0072



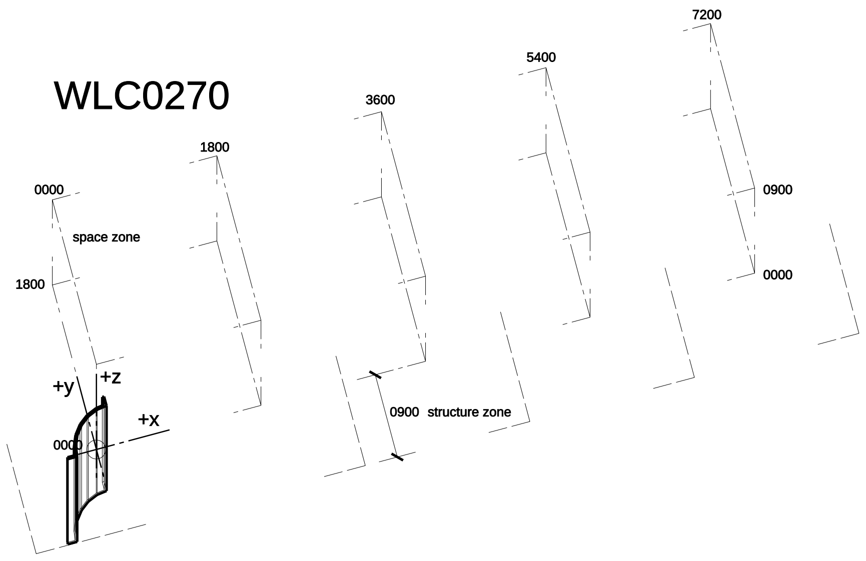
WLC0090



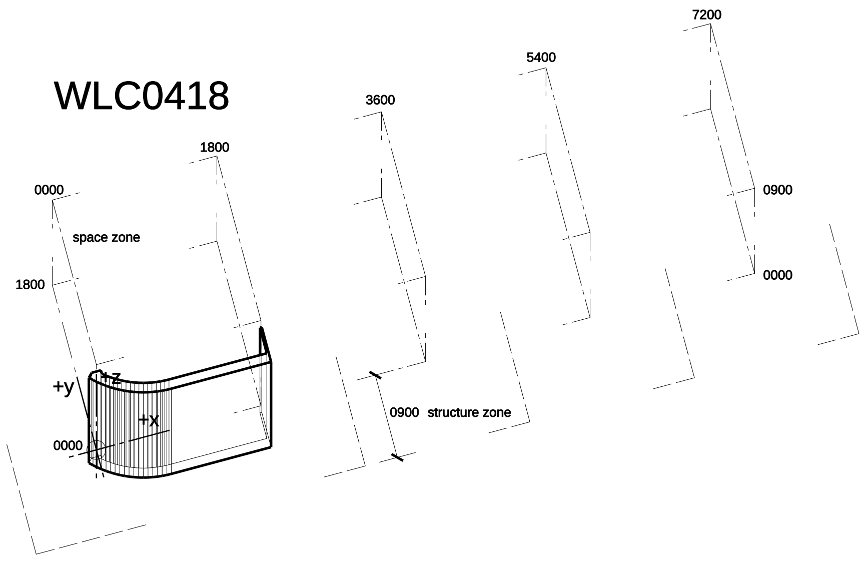
WLC0180



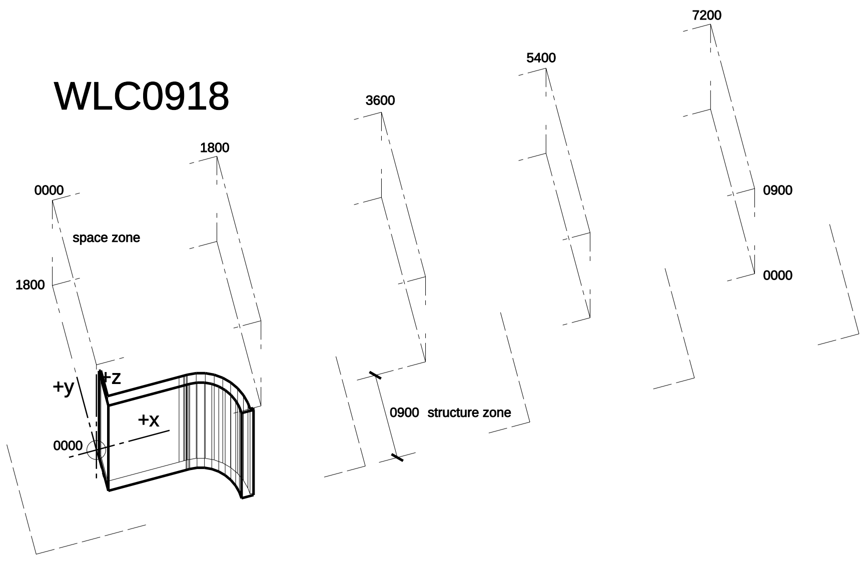
WLC0270



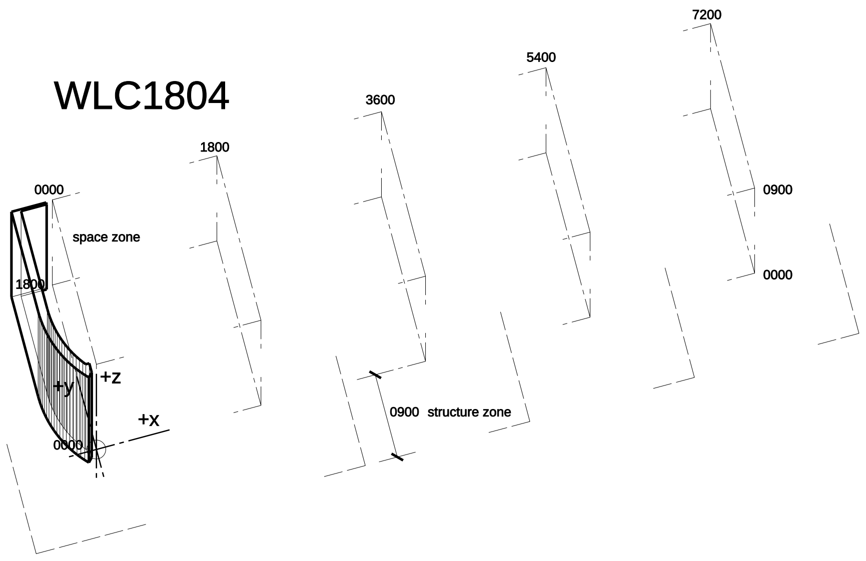
WLC0418



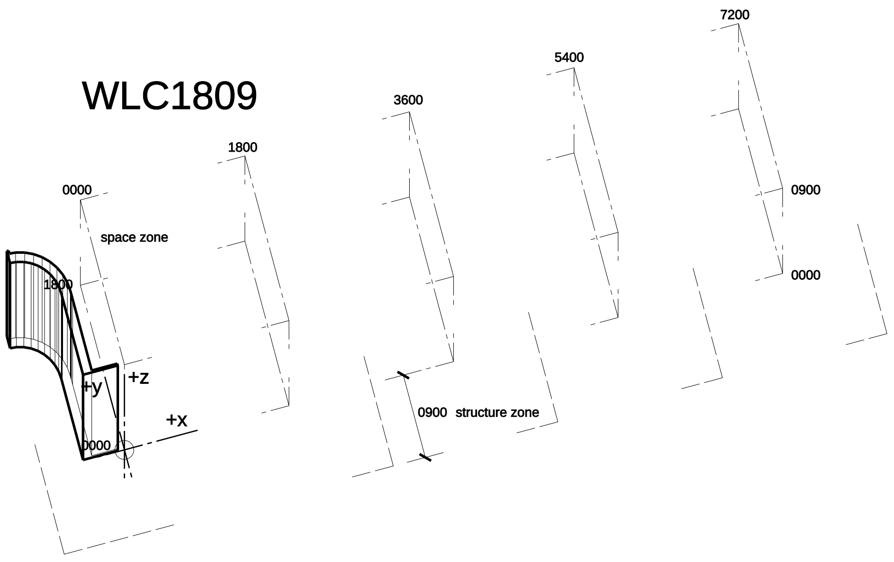
WLC0918

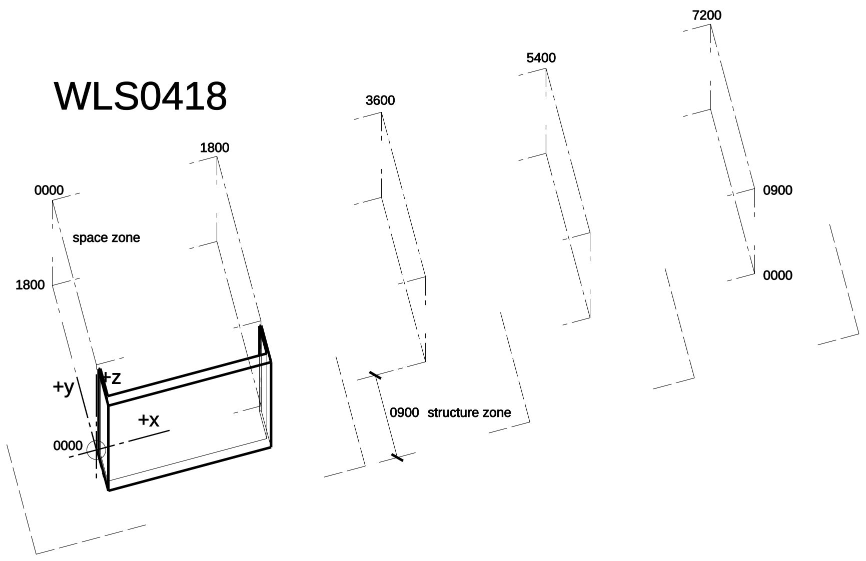


WLC1804



WLC1809




WLS0418
7200
5400
3600
1800
0000
space zone
1800
0900
0000
+z
+y
+x
0000
0900 structure zone

BIBLIOGRAPHY

BIBLIOGRAPHY

Ambasz, Emilio, *Italy: The New Domestic Landscape, Achievements and Problems of Italian Design*, The Museum of Modern Art, New York, NY, April 1972.

Apple Computer, Inc., *Inside Macintosh: Devices*, Addison-Wesley Publishing Company, Menlo Park, California, 1994.

Banham, Reyner, *Megastructure: Urban Futures of the Recent Past*, Thames and Hudson Ltd., London, England, 1976.

Bederson, Benjamin B., Stead, Larry, and Hollan, James D., "Pad++: Advances in Multiscale Interfaces", *SIGCHI '94* short paper, 1994.

Bernhold, Leonhard E., Abraham, Dulcy M., and Reinhart, Davis B., "FMS Approach to Construction Automation", *Journal of Aerospace Engineering*, Volume 3, No.2, April 1990, pp.108-121.

Bernhold, Leonhard E., and Livingston, Eric E., "A prototype for intelligent computer integrated wood truss fabrication", *Robotics and Autonomous Systems*, volume 6, Elsevier, North-Holland, 1990, pp.337-349.

Bini, Dante, "Binisystems" http://www.webville.com/oak/bini

Boutell, Thomas, *CGI Programming in C & Perl*, Addison-Wesley Publishing Company, Menlo Park, California, 1996.

Bridgewater, Colin, "Principles of Design for Automation applied to construction tasks", *Automation in Construction*, Volume 2, Elsevier, Amsterdam, 1993, pp.57-64.

Brooks, Alan J. and Grech, Chris, The Building Envelope, Butterworth & Co., UK, 1990.

Bruce, Alfred and Sandbank, Harold, *A History of Prefabrication*, John B. Pierce Foundation, New York, NY, July 1943.

Candela, Felix et.al., *Arquitectura Transformable*, Escuela Technica Superior de Arquitectura de Sevilla (ETSAS), 1993.

Chase, Scott C., "Logic based design modeling with shape algebras", *Automation in Construction*, Vol. 6, Elsevier, Amsterdam, 1997, pp. 311-322.

Cornell, Gary, and Horstmann, Cay S., *Core Java*, second edition, Sun Microsystems, Inc., 1997. Japanese edition.

Davies, Colin, *High Tech Architecture*, Thames and Hudson, London, Great Britain, 1988, pp. 42-55, 68-85.

December, John, *Presenting Java*, Sams.net Publishing, Indianapolis, Indiana, 1995.

Diamant, R. M. E., *Industrialised Building 2*, Lliffe Books Ltd., London, England, 1965.

Diamant, R. M. E., *Industrialised Building 3*, Lliffe Books Ltd., London, England, 1968.

Fenton, Brian C., "Rabbit ears meet the mouse", *Popular Mechanics*, October 1996, pp86-89.

Flemming, U., “More Than the Sum of Parts: The Grammar of Queen Anne House”, *Environment and Planning*, B 14, 1987.

Fukai, Dennis, “PCIS: a piece-based construction information system on the world wide web”, *Automation in Construction*, Vol. 6, Elsevier, Amsterdam, 1997, pp. 287-298.

Furnas, George W., "Generalized Fisheye Views", *Human Factors in Computing Systems CHI '86* Conference Proceedings, Boston, 13-17 April 1986, pp16-23.

Gadre, Sharad H., “Enterprise & Information Model”, *Database Programming & Design*, premier issue, 1987.

Gilmore, V. Elaine, "U.S., Japan, Europe: The World's Smartest Houses", *Popular Science*, September 1990, pp. 56-65, 102.

Gloag, John and Wornum, Grey, *House Out of Factory*, George Allen & Unwin Ltd., London, England, 1946.

Hamer, Jeffrey M., *Facility Management Systems*, Van Nostrand Reinhold Company, New York, 1988, pp79-84.

Herzog, Thomas, *Pneumatic Structures: A Handbook of Inflatable Architecture*, Oxford University Press, New York, NY, 1976.

Housing Systems Proposals for Operation Breakthrough, U.S. Department of Housing and Urban Development, Washington, DC, December 1970.

Howe, A. Scott, and Yasutoshi, Hirata, "Architecture, Urban Planning Example 1: Design Using 3D CAD", *IBM CATIA: State of the Art 3D CAD / CAM, Technical Papers '92 / '93*, Tokyo, Japan, 1993, pp. 6-9, 36-40.

Howe, A. Scott, "A Genesis System", *Special Research Report of the Sensitivity Engineering Product Development Research Group: A Sensitively Designed City*, Japanese Ministry of International Trade and Industry , Tokyo, 1994, pp.73-92. Report in Japanese.

Howe, A. Scott, "Designing for Automated Construction", presented at the *International Symposium on Automation and Robotics in Construction (ISARC14)*, Pittsburgh, Pennsylvania, 9 June 1997a.

Howe, A. Scott, "Intelligent Living" and "Intelligent Office", *Kawasaki Safety Intelligent Plaza*, University of Utah Master of Architecture Thesis, Salt Lake City, Utah: University of Utah 1989.

Howe, A. Scott, "Internet-based Architectural Visualization", presented at the *ACSA European Conference*, Copenhagen, Denmark, 27 May 1996.

Howe, A. Scott, "Internet-based Remote Facility Management" *The Third International Convention on Urban Planning, Housing, and Design (ICUPHD '97)* held in Singapore, 22-24 Sept. 1997b.

Howe, A. Scott, and Yasutoshi, Hirata, "A Feasibility Study for a New Architectural Design Approach Using 3D Solid Modeling CAD Systems", *Fourth International Conference on Computing in Civil and Building Engineering*, Tokyo, Japan, July 1991.

Ibanez-Guzman, Javier, "Modeling of on-site work cells for the simulation of automated and semi-automated construction", *Construction Management and Economics*, 1995, pp.427-434.

Intelligent Manufacturing System (IMS) Collaborative, "Automated Materials Delivery System for High-rise Buildings" in product pamphlet, IMS Collaborative, Tokyo, 1995. Pamphlet in Japanese.

Japanese Ministry of Construction, "Development of Automated Systems for the Construction of Reinforced Concrete Structures", *Ministry of Construction Technology Development Project Collection: New Construction Technology Development for the Construction Industry*, volume 2, Japan Ministry of Construction, Tokyo, 1994, pp.14-95. Report in Japanese.

Jencks, Charles, *Architecture 2000: Predictions and Methods*, Praeger Publishers, New York, New York, 1971, pp. 94-95.

Kajima Corporation, "AMURAD Grow-up System" in project report, Kajima Corporation, Tokyo, 1994. Report in Japanese.

Kajima Corporation, "Push-up System (Electric Powered Screw Jack Method) Plan" in project report, Kajima Corporation, Tokyo, 1995. Report in Japanese.

Kangari, Roozbeh, and Halpin, Daniel W., "Potential Robotics Utilization in Construction" , NSF research report under grant no. CEE-8319498, National Science Foundation, Washington, D.C., 1983.

Kelly, Burnhan, *The Prefabrication of Houses*, Technology Press & John Wiley and Sons, New York, NY, 1951.

Kelly, Phyllis M. and Hamilton, Richard W., "Housing Mass Produced", *1952 Housing Conference*, Albert Farwell Bemis Foundation, Massachusetts, 1952.

Kernighan, Brian W., and Ritchie, Dennis M., *The C Programming Language*, Bell Telephone Laboratories, Incorporated, United States, 1978.

Kultermann, Udo, *Kenzo Tange: Architecture and Urban design 1946-1969*, Praeger Publishers, New York, NY, 1970.

Kurita, H., Tezuka, T., and Takada, H., "Robot Oriented Modular Construction System - part II: Design and Logistics", *Automation and Robotics in Construction X*, Proceedings of the 10th International Symposium on Automation and Robotics in Construction (ISARC), Houston, Texas, 24-26 May 1993, Elsevier, Amsterdam, 1993, pp.309-316.

Kurokawa, Kisho, *From Metabolism to Symbiosis*, Academy Group, London, England, 1992.

Kurokawa, Kisho, *Metabolism in Architecture*, Westview Press, Inc., Boulder, Colorado, 1977.

Lalani, Suleiman, and Jamsa, Kris, *Java Programmer's Library*, Jamsa Press, Las Vegas, Nevada, 1996.

Lampugnani, Vittorio Magnago, *Renzo Piano: Progetti e architetture 1987-1994*, Electa, Milano, Italy, 1995 (in italian).

Liou, Shuenn-Ren, "A Computer-based Framework for Analyzing and Deriving the Morphological Structure of Architecture", Ph.D. dissertation, The University of Michigan, 1992.

Mackinley, J., "An organic user interface for searching citation links", *Human Factors in Computing Systems CHI '95* Conference Proceedings, ACM, 1995, pp67-73.

Makoto, Akinaga, "Four Firms Start Construction in the Search for a New Architectural Manufacturing Paradigm", *Nikkei Architecture*, 7 June 1993, pp.160-165. Article in Japanese.

Mark, Dave, Cartwright Reed, *Macintosh C Programming Primer Volume I*, Addison-Wesley Publishing Company, Menlo Park, California, 1992.

Mark, Dave, *Macintosh C Programming Primer Volume II*, Addison-Wesley Publishing Company, Menlo Park, California, 1990.

Mark, Leo, “The Binary Relationship Model -- 10th Anniversary”, 6th ER Conference, New York, New York, November 1987.

Marks, Robert W., *The Dymaxion World of Buckminster Fuller*, Reinhold Publishing Corporation, New York, New York, 1960, pp88-91.

Marumoto, Yukihiro, "Animation of Automated Building Construction Development and Feasibility" in technical paper, Obayashi Corporation, Tokyo, date unknown, pp.335-342. Paper in Japanese.

Mitchell, William, *The Logic of Architecture: Design, Computation, and Cognition*, MIT Press, Cambridge, Massachusetts, 1990.

Miyamoto, Ishii, Shibata, Dobashi, Howe, Yoshida, Takada, Ueno, Kunugi, Yagi, Nakada, Hatakeyama, Kikawada, Yomo, and Koga, “A study on assembly process for large scale structures”, Intelligent Manufacturing Systems IF7 report, IMS Collaborative, Tokyo, 1998.

Mukherjea, Sougata, and Foley, James D., "Visualizing the World Wide Web with the Navigational View Builder", *Proceedings of the 3rd International World-Wide Web Conference*, Darmstadt, Germany, 10-14 April 1995, Computer Networks and ISDN Systems; Vol. 27, No. 6, pp. 1075-1087.

Murray, Irena Zantovska, *Moshe Safdie: Buildings and Projects 1967-1992*, McGill-Queen's University Press, Montreal, Quebec, 1996.

Nakamura, Toshio, "Norman Foster 1964-1987", *A + U Architecture and Urbanism*, May 1988 Extra Edition.

Nakamura, Toshio, "Richard Rogers 1978-1988", *A + U Architecture and Urbanism*, December 1988 Extra Edition.

Newman, H. Michael, *Direct Digital Control of Building Systems: Theory and Practice*, John Wiley & Sons, Inc., New York, New York, 1994.

Ohara, Takashi, “An Automated Construction System which Pushes up Completed Floors”, *Nikkei Architecture*, 25 August 1997, pp. 154-157. Article in Japanese.

Paraskevopoulos, Stephen C. A., Borkin, Harold, et.al., *Research on Potential of Advanced technology for Housing*, Architectural Research Laboratory, The University of Michigan, Ann Arbor, Michigan, 1968.

Paraskevopoulos, Stephen C. A., Borkin, Harold, et.al., *Structural Potential of Foam Plastics for Housing in Underdeveloped Areas*, Architectural Research Laboratory, The University of Michigan, Ann Arbor, Michigan, May 1966.

Pawley, Martin, *Future Systems: The Story of Tomorrow*, Phaidon Press Limited, London, England, 1993.

Pesce, Mark, *VRML: Browsing & Building Cyberspace*, New Riders Publishing, Indianapolis, Indiana, 1995.

Radford, Denny, "Spread-spectrum data leap through ac power wiring", *IEEE Spectrum*, Vol33 No11, November 1996, pp48-53.

Ross, Michael Franklin, AIA, *Beyond Metabolism: The New Japanese Architecture*, McGraw-Hill Book Company, New York, New York, 1978, pp. 36-38.

Russell, Frank, *Richard Rogers + Architects*, St. Martin's Press, New York, NY, 1985.

Safdie, Moshe, *Beyond Habitat by 20 Years*, Tundra Books, Montreal, Quebec, 1970.

Sakamura, Ken and Sprague, Richard, "The TRON Project", *Byte*, Vol. 14, No. 4, April 1989, pp. 292-301.

Sakao, T., Kondoh, S., Umeda, Y., and Tomiyama, T., “The Development of a Cellular Automatic Warehouse”, *Proceedings of the 1996 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 96): Robotic Intelligent Interacting with Dynamic Worlds*, Vol. 1, IEEE 96 CH 35908, IEEE, Osaka, Japan, November 4-8, 1996, pp. 324-331.

Satir, Gregory, and Brown, Doug, *C++ The Core Language*, O' Reilly & Associates, Inc., Sebastopol, California, 1995.

Schierhorn, Carolyn, "Robotics in Masonry", *Masonry Construction*, Vol. 7, No. 7, July 1994, pp.288-294.

Sekiguchi, T., Honma, K., Mizutani, R., and Takagi, H., “The Development and Application of an Automatic Building Construction System Using Push-up Machines”, presented at the *International Symposium on Automation and Robotics in Construction (ISARC14)*, Pittsburgh, Pennsylvania, June 1997.

Shaio, Sami, "chart.java" code, Sun Microsystems, Inc. March 1995.

Sheppard, Richard, *Prefabrication in Building*, The Architectural Press, London, England, 1946.

Shimizu Corporation, "Shimizu Manufacturing system by Advanced Robotics Technology (SMART)" in R&D product pamphlet, Shimizu Corporation, Tokyo, 1993.

Skibniewski, Miroslaw J., and Wooldridge, Stephen C., "Robotic materials handling for automated building construction technology", *Automation in Construction*, Vol. 1, Elsevier, Amsterdam, 1992, pp. 251-266.

Smith, Edward F., "Virtual Buildings: Knowledge Based CAD Models for Design, Analysis, Evaluation and Construction", *Computer Solutions*, Summer 1992, pp. 30-32.

Smith, Joseph Jr., *Doctrine and Covenants*, 1833 (republished by The Church of Jesus Christ of Latter-day Saints, Salt Lake City, 1988.)

Snider, Mike, "Now you can tune your TV to Internet", *USA Today*, 19 September 1996, p. 4D.

Stouffs, R., Krishnamurti, R., and Oppenheim, I., "A behavioral language for motion planning in building construction", *Automation in Construction*, Vol. 3, Elsevier, Amsterdam, 1995, pp. 305-320.

Swaback, Vernon D., *Production Dwellings*, The Frank Lloyd Wright Foundation, Spring Green, Wisconsin, 1970.

Taisei Corporation, "Announcing the Age of Construction Automation, Now!", *Nihon Keizai Shinbun* newspaper article, 2 November 1990. Article in Japanese.

Taisei Corporation, "Robot for Painting the Exterior Walls" in product pamphlet, Taisei Corporation, Tokyo, 1991. Pamphlet in Japanese.

The Bible.

The Editorial Committee of The Second Archtectural Convention of Japan, *Structure Space Mankind Expo '70*, The Architectural Association of Japan, Osaka, Japan, May 1970.

Turner, James A., "Guide to Reading NIAM Diagrams", Architecture and Planning Research Laboratory, University of Michigan, Ann Arbor, Michigan, 22 May 1991.

Turner, James A., et.al., "AEC Building Systems Model", ISO TC/184/SC4/WG1, Document 3.2.2.4, Architecture and Planning Research Laboratory, University of Michigan, Ann Arbor, Michigan, 2 August 1990.

Turner, James A., et.al., GEDIT solid modeler, The Architecture and Planning Research Laboratory, University of Michigan, Ann Arbor, Michigan.

Turner, James A., “Some Thoughts on the Existence of a Generic Building Object”, unpublished paper, Architecture and Planning Research Laboratory, The University of Michigan, Ann Arbor, Michigan, 1997.

Vacca, John R., *VRML: Bringing Virtual Reality to the Internet*, Academic Press, London, 1996.

van Heuvel, Wim J., *Structuralism in Dutch Architecture*, Uitgeverij Publishers, Rotterdam, The Netherlands, 1992.

van Pelt, Robert Jan, and Westfall, Carroll William, *Architectural Principles in the Age of Historicism*, Yale University Press, New Haven and London, 1991.

Lampugnani, Vittorio Magnago, *Renzo Piano: Progetti e architetture 1987-1994*, Electa, Milano, Italy, 1995 (in italian).

Weiss, Mark Allen, *Algorithms, Data Structures, and Problem Solving with C++*, Addison-Wesley Publishing Company, Menlo Park, California, 1996.