A Network-based Kit-of-parts Virtual Building System

A. Scott Howe, architect*

1. A Research Programme

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, instantly gleaning performance data from the constructed object itself, how would an information feedback loop affect current use and future design improvements to future 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 facilitated. As a research programme, it is proposed that a computer tool be developed which can affect such an experimental environment. The computer tool has been conceived in the form of a plug-in to a common Internet browser. Some functions of the plug-in will eventually be made available to a selected number of designers for further research purposes.

2. VBuild Browser

2.1 VBUILD CONCEPT

The browser plug-in (hereafter called VBuild) makes maximum use of two powerful concepts: object-oriented programming and kit-of-parts philosophy. In a way, the two concepts work hand in hand.

2.1.1 Object Oriented Programming

VBuild was programmed using an object-oriented structure in the C++ 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 coding 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 raw coding.

2.1.2 Kit-of-parts Philosophy

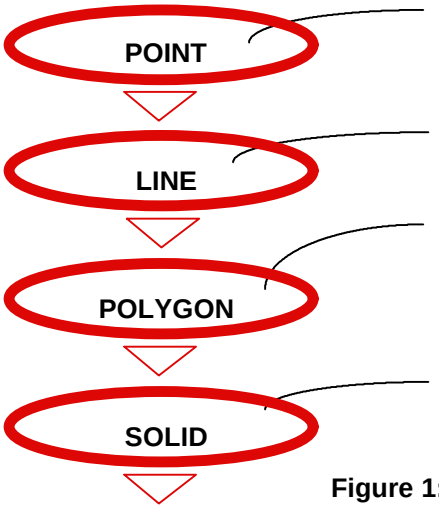
The active use of kit-of-parts philosophy was an important element in the conception of VBuild. A kit-of-parts is a collection of discrete building components that are pre-engineered and designed to be assembled in a variety of ways, much similar to an erector set or toy Lego blocks. When assembled, the entire kit-of-parts can define a finished building or 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 used over and over again, fabrication processes can be worked out in advance for real-time manufacturing at time of need.

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.

2.2 VBUILD CLASSES

The programming 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 would be a discrete component which is manufactured using a combination of different fabrication processes, and follows interface rules for connection to other components (in this paper, “assembly” and “component” may be used interchangeably and refer to the same thing). Construct classes define ways of organizing the assemblies.



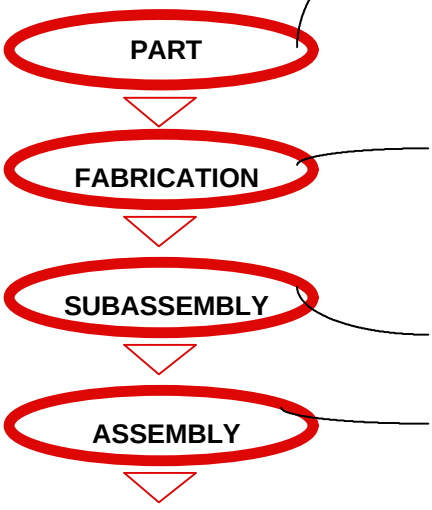
The POINT class represents the most basic form of geometry or 0D geometry. One point object would contain XYZ coordinates as data, and include various methods for manipulating the coordinates.

The LINE class represents 1D geometry and includes both lines and segments. A line will have two point objects as data and include methods for manipulating itself.

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

Figure 1: VBuild Geometry Classes



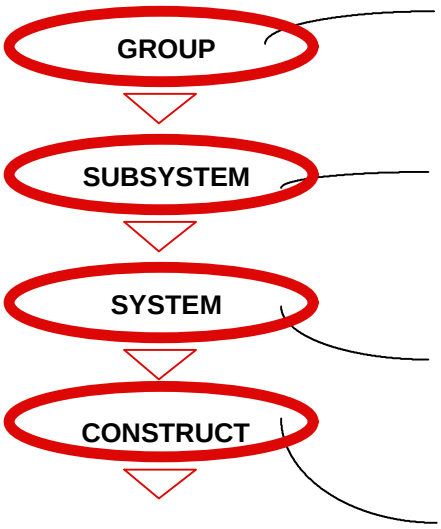
The PART class potentially contains one solid or one polygon or both, as well as a material definition. A part is meant to represent a raw material with limited definition of a potential shape. The solid would be representative of a cast object or folded sheet fabrication, where the polygon would represent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

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

The SUBASSEMBLY 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 represents a kit-of-parts entity. It can have many subassemblies and would 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 would represent real manufactured components assembled on the site.

Figure 2: VBuild Assembly Classes



The GROUP class 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 would manipulate groups.

The SUBSYSTEM class 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 would contain methods which would be for the express purpose of manipulating subsystems.

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

The CONSTRUCT class is the finished artifact, which includes all the systems. Construct methods would define behavior of the construct, and include mechanisms for viewing it in various ways.

Figure 3: VBuild Construct Classes

In addition to the three main groups of classes, VBuild also has attribute classes and utility classes for the viewing and general manipulation of the various data types. In each step of the hierarchy, methods specific to that data type are implemented. More methods can be added later as the need arises.

2.3 PROPOSED VBUILD CONFIGURATION

VBuild will actually consist of several programs functioning in unison in various locations on the Internet. The browser Plug-in portion is merely the local tool which helps the user view and manipulate the data. Locally the user will create a construct which can be saved as a file locally. Assemblies / components will be returned on request from remote kit-of-parts virtual librarians. Fabrication of real parts will be coordinated by a virtual contractor, and remote control and monitoring will be facilitated by a virtual facility manager. Except for the plug-in, each of the virtual servers will consist of simple Common Gateway Interface (CGI) programs, which are small programs linked to web pages that perform simple automated tasks. The process is delineated in the three steps of design, manufacture, and facility management.

2.3.1 Design

When the user wants to insert another component, the VBuild plug-in will contact remote virtual librarians which will return requested components according to type. The assemblies / components for the most part 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 quickly. The local plug-in 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. 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 / construct is deleted 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, the user will be able to view the data in various ways which may include sections and plans.

2.3.2 Manufacture

Once the building is virtually assembled, the assemblies / components can be manufactured and delivered. Each assembly / component can consist of many fabrications. Since the fabrications contain the data necessary for their own manufacture, the VBuild plug-in 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 have 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 will 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 will 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.

2.3.3 Facility Management

Once the building is actually assembled, a Hypertext Transfer Protocol (HTTP) or remote access server can be paired with a virtual facility manager CGI on a local computer installed in the finished building. An HTTP server would be used to facilitate an Internet connection, as opposed to a remote access server which supports private phone connections. Each subassembly has the capacity to link to a CGI program which can handle bitwise communication with a LonWorks, CEBus, or X-10 interface device connected to the computer's serial port. LonWorks, CEBus, and X-10 hardware are brand name pseudo-standards which facilitate plug and play monitoring and control of other 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. The user would be able to fly-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, special CGI programs would bring up a Java control panel especially prepared for that device.

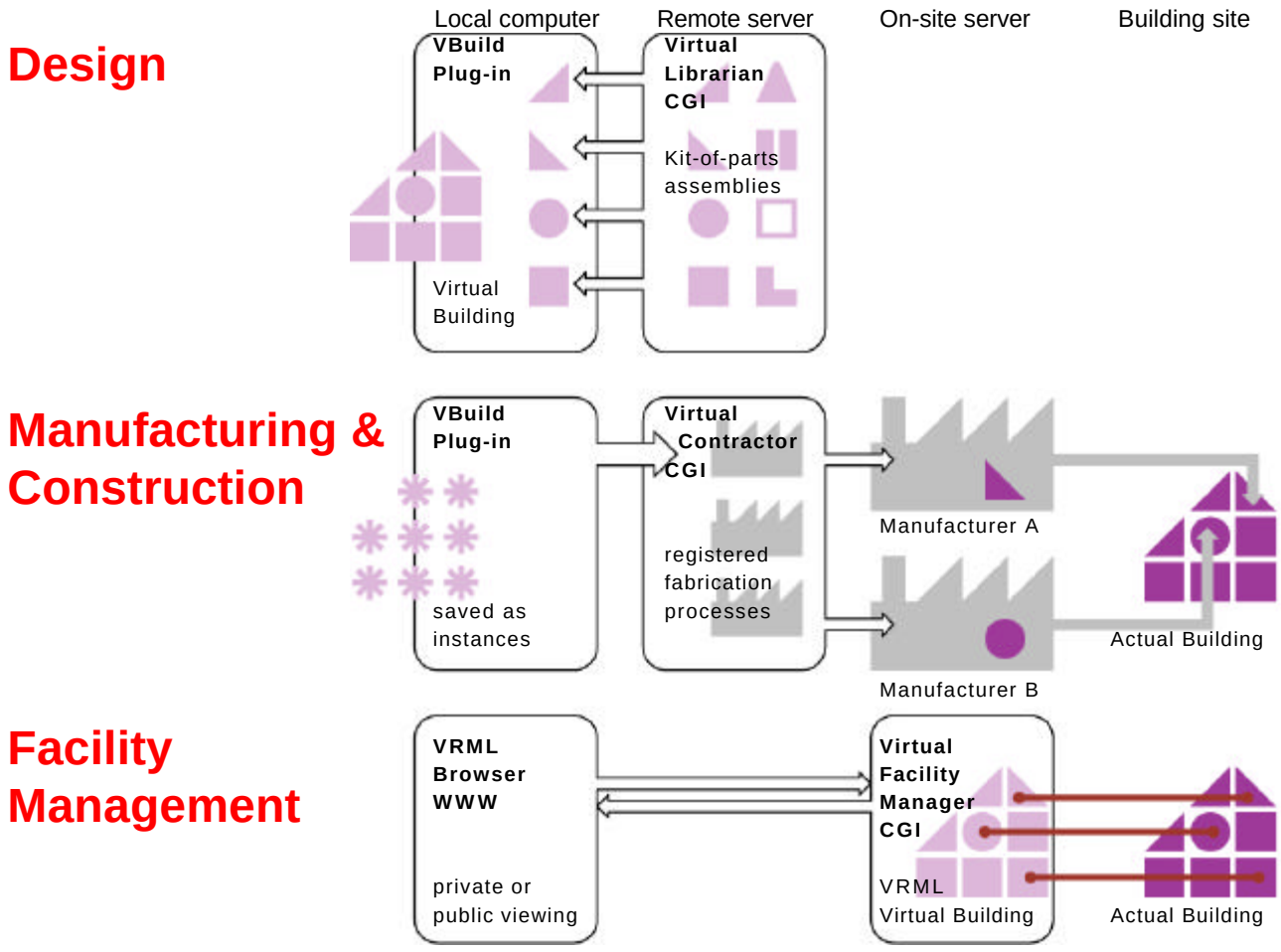


Figure 4: VBuild Proposed Implementation

Figure 4 describes the processes associated with the three phases of design, fabrication, and facility management. In the design stage a designer on a local computer would install the VBuild plug-in in a World Wide Web (WWW) Internet browser such as Netscape. VBuild would allow the designer to collect kit-of-parts assemblies and assemble them together in an intuitive way much the same way a child would construct an object out of Lego blocks. The assembled virtual building could then actually be manufactured component by component through the brokerage of the virtual contractor which

would have various manufacturer's fabrication processes on register. Once the actual building is built, a VRML version of the virtual building would be loaded into a computer installed in the actual building, and would have access to CGI's which provide an interface to the LonWorks, CEBus, or X-10 hardware (symbolized in Figure 4 by lines connecting the virtual building and actual building). Full Internet access to the VRML virtual building could be facilitated for public access, or limited phone connection only could be facilitated for remote private use.

3. Conclusion

For simplicity's sake kit-of-parts philosophy is utilized in this work. When individual designers begin to use the system they may want to be able to define their own components rather than use those already designed by someone else. Along with the ability to harness various fabricatioin processes for the purpose of facilitating the manufacture of a pre-defined kit-of-parts, real time design and manufacture could be a next step. Regardless of whether the design consists of extemporaneously thought-out elements or pre-designed components, the ability to create a virtual artifact and link it back to the real one should for the purpose of design, manufacture, and facility management prove to be a powerful tool.

The eventual goal would be to have a real building that adapts to the users needs through user-initiated instructions affected by the manipulation of its virtual building counterpart. Initial construction and renovative construction could eventually utilize robotic systems which are either brought in from elsewhere or are actually incorporated into the components of the building itself. Redesign and renovation would be facilitated by changing the virtual building to meet the new needs, and executing the automated construction and assembly features to bring about the changes in the actual building.

NOTES & REFERENCES

Mark Pesce (1995) *VRML: Browsing & Building Cyberspace*, New Riders Publishing, Indianapolis, Indiana.

Gregory Satir, Doug Brown (1995) *C++ The Core Language*, O' Reilly & Associates, Inc., Sebastopol, California.

Mark Allen Weiss (1996) *Algorithms, Data Structures, and Problem Solving with C++*, Addison-Wesley Publishing Company, Menlo Park, California.

James Turner, et.al., *GEDIT solid modeler*, The Architecture and Planning Research Laboratory, University of Michigan, Ann Arbor, Michigan.

Kisho Kurokawa (1977) *Metabolism in Architecture*, Westview Press, Inc., Boulder, Colorado.

Colin Davies (1988) *High Tech Architecture*, Thames and Hudson, London, Great Britain, pp42-55, pp68-85.

Martin Pawley (1993) *Future Systems: The Story of Tomorrow*, Phaidon Press Limited, London, England.

James Turner, et.al. (August 1990) "AEC Building Systems Model", *ISO TC/184/SC4/WG1, Document 3.2.2.4*, The University of Michigan, Architecture and Planning Research Laboratory, Ann Arbor, Michigan.

James Turner (May 1991) "Guide to Reading NIAM Diagrams", The University of Michigan, Architecture and Planning Research Laboratory, Ann Arbor, Michigan.

A. Scott Howe, Hirata Yasutoshi (July 1991) "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.

Edward F. Smith (Summer 1992)"Virtual Buildings: Knowledge Based CAD Models for Design, Analysis, Evaluation and Construction", *Computer Solutions*, pp30-32.

A. Scott Howe, Hirata Yasutoshi (1993) "Architecture, Urban Planning Example 1: Design Using 3D CAD", *IBM CATIA: State of the Art 3D CAD / CAM, Technical Papers '92 / '93*, Tokyo, Japan pp6-9, 36-40.

A. Scott Howe (May 1996) "Internet-based Architectural Visualization", presented at the *ACSA European Conference*, Copenhagen, Denmark.

Roozbeh Kangari, and Daniel W. Halpin (1983) "Potential Robotics Utilization in Construction" , NSF research report under grant no. CEE-8319498, National Science Foundation, Washington, D.C.

Leonhard E. Bernhold, Dulcy M. Abraham, and Davis B. Reinhart (April 1990) "FMS Approach to Construction Automation" in *Journal of Aerospace Engineering*, Volume 3, No.2, pp.108-121.

Miroslaw J. Skibniewski, and Stephen C. Wooldridge (1992) "Robotic materials handling for automated building construction technology" in: *Automation in Construction volume 1*, Amsterdam, Elsevier, pp.251-266.

Akinaga Makoto (7 June 1993) "Four Firms Start Construction in the Search for a New Architectural Manufacturing Paradigm" in: *Nikkei Architecture*, pp.160-165. Article in Japanese.

Colin Bridgewater (1993) "Principles of Design for Automation applied to construction tasks" in: *Automation in Construction volume 2*, Amsterdam, Elsevier, pp.57-64.

A. Scott Howe (June 1997) "Designing for Automated Construction", presented at the *14th International Symposium of Automation and Robotics in Construction (ISARC14)*, Pittsburgh, Pennsylvania.

H. Michael Newman (1994) *Direct Digital Control of Building Systems: Theory and Practice*, John Wiley & Sons, Inc., New York, New York.

Denny Radford of Intellon Corp. (November 1996) "Spread-spectrum data leap through ac power wiring", *IEEE Spectrum*, Vol33 No11, pp48-53.

---

* Kajima Corporation Intelligent Systems Department
EMAIL: ash@ipc.kajima.co.jp
Home Page URL: http://www-personal.umich.edu/~ashowe

ネットワークをベースにした部品化バーチャル建築システム

A. Scott Howe, architect*

最近、デザインプロセスの中に強力なコンピュータデザインツールを使用しているデザイナーが増えています。その動きと並行して、世界中のいろいろな情報源からリアルタイムに情報を入手できるインターネットやワールドワイドウェブが益々発達してきました。デザインコンセプトの元になるものも情報であると考えられるため、適当な質問は、「情報の源に直接つながるデザインプロセスはどう変わるか」となるでしょう。情報をベースした建築現場の自動化やＯＡ化生産プロセスも増えたため、もう一つの質問は、「生産プロセスに直接的なアクセスが可能であれば、デザインプロセスはどう変わるか」となります。最後に、「作品（建築物）の性能データにリアルタイムアクセスできたら、作品の現在の使い方や将来の同じような作品のデザインはどう変わるか」という三つ目の質問もあります。この三つの質問は近いうちに解決できないかもしれませんが、デザイン、生産、メンテをそれぞれについての、リアルタイムなリンクをシミュレートできるような実験的な環境は現在作れるでしょう。

本論文では、ユーザがインターネット上に設置してあるバーチャル部品ライブラリ（たとえばメーカーのデータベース）から部品を集める作業を可能とする、実験的なブラウザーの詳細を説明します。ブラウザーは、集めた部品を使ってバーチャルビルディングを組み立て、実際の部品生産や建設を準備できるツールとなります。ブラウザーを通して、書類を電子的にプリンターに送ると同様に、集めた部品のリアルタイムに近い生産を実現することができますし、実際の建物ができた時点でも、実物の部品に設置したセンサーなどにインターフェースを持つことができます。ブラウザーは、ＶＲＭＬを使ってインターラクティブ３次元モデルを表示します。画面でバーチャルビルディングをウォークスルーしながら、バーチャル部品をクリックすると、建物の性能データを集めるために実物の部品に設置したセンサーの情報を、リアルタイムで見れます。このようにして、デザイン段階で作ったバーチャルビルディングは実物の建物にある装置（空調機など）を監視したりコントロールしたりできるリモートＦＭツールになります。本論文を書いた時点で、ブラウザーのいくつかの機能については、バーチャルビルディングを描くコアクラスオブジェクトを中心に、実験的に実証しています。

* 鹿島 情報システム部
電子メール: ash@ipc.kajima.co.jp
ホームページ: http://www-personal.umich.edu/~ashowe