Off The Page: Object-Oriented Construction Drawings

Michael Kilkelly, Edificium, Inc., USA

Abstract

This paper discusses methods in which inefficiencies in the construction documentation process can be addressed through the application of digital technology. These inefficiencies are directly related to the time consuming nature of the construction documentation process, given that the majority of time is spent reformatting and redrawing previous details and specifications. The concepts of object-oriented programming are used as an organizational framework for construction documentation. Database structures are also used as a key component to information reuse in the documentation process. A prototype system is developed as an alternative to current Computer-Aided Drafting software. This prototype, the *Drawing Assembler,* functions as a graphic search engine for construction details. It links a building component database with a construction detail database through the intersection of dissimilar objects.

1 Introduction

Since the early beginnings of graphic software, the computer has been touted as a means to achieve efficiency and productivity in the architectural office. This has been largely through the use of Computer-Aided Drafting (CAD) software in the production of construction drawings. However, despite the widespread use of CAD in the majority of architectural offices, the computer has had little influence on the nature of construction documents. While the tools to produce the drawings have changed, from pencil and paper to mouse and monitor, the methodology has essentially remained the same. Until recently, CAD software has been used largely as a means to replicate known methods of work. As Malcolm McCullough indicates in *Abstracting Craft*, CAD is a process of task automation, where the computer is used to perform known processes more efficiently as opposed to changing the underlying working process (McCullough 1996). What is necessary is a rethinking of the architect's traditional methodology and an examination of how the inherent capabilities of the computer can be better utilized to achieve a more accurate and consistent transfer of information from designer to builder.

2 Object-Oriented Programming

One area that has received a great deal of attention regarding computer-aided design research is object-oriented programming. Object-oriented programming (OOP) is a set of concepts that have been used extensively in software design and engineering in order to effectively construct and manage large-scale computational systems. While computer programs are certainly not buildings, the concepts of OOP provide a means to organize information in an effective and meaningful manner.

The overriding concept of OOP is the division of complex tasks into small, easily managed pieces, called objects. These objects are computer abstractions that model the components of the system being simulated. In addition to greatly reducing the complexity of a given system, these objects can also be reused in other projects or combined to create more complex software modules. Object-oriented programming also allows for easy modification and extension of individual components without requiring the programmer to re-code the entire component from scratch. (Cox and Novobilski 1991)

2.1 Object-Oriented CAD

Within the last few years, several object-oriented CAD packages have been commercially introduced. Microstation Triforma by Bentley and the recently released Revit platform are two such examples. Typically referred to as “building modelers” these applications use objects to represent building components. The design is three-dimensionally modeled or “built” using these objects. In addition to geometric properties, attributes such as cost or performance criteria can be incorporated into the object. Drawings and other reports are automatically generated and linked to the 3D model, ensuring that all information is accurate and consistent with the model.

Building modelers seek to computationally simulate the constructed reality in its entirety. Plan, section, and elevation drawings are produced as a by-product of this process. These drawings generally indicate what components are to be used and where they are to be located. However, they do not directly indicate how a particular component is to be assembled to its neighboring components. This is largely the domain of the construction detail drawing. Despite the obvious advantage to using a building modeler, the architect is still required to manually locate the necessary details from the firm’s detail library or create the detail from scratch. The intelligence of the software does not extend this far into the construction documentation process. What is needed is a system that recognizes the component nature of construction and provides an immediate means to access the depth of a firm’s construction knowledge.

3 Implementation

In order to directly address the deficiencies related to the construction documentation process, a prototype system, the *Drawing Assembler,* was developed to facilitate information reuse in the documentation process. The *Drawing Assembler* provides access to the extents of a firm’s accumulated detail library by making such information readily available to the user in a contextual manner. It is an approach to construction documentation that functions at the level of the component.

Other research has been conducted into the use of component based construction drawings, specifically that of Harfmann (Harfmann 1993) and Gross (Gross 1996). Both, however, focus primarily on early stages of design and on developing precise geometric representations. Their emphasis is on resolving the relationships between components as they are assembled in either two dimensions (Gross 1996) or three (Harfmann 1993). In contrast, the *Drawing Assembler* operates on the level of the detail. Rather than resolving the geometric relationship of the interfacing components, the *Drawing Assembler* searches and generates details that are specific to the components. It functions more as an interactive search engine for construction details than a drafting or modeling program.

Fundamentally, the *Drawing Assembler* (Fig. 1) is a graphic interface that links two databases. The first database contains a collection of building assemblies and components. The second database contains a series of parametrically defined construction details. A single classification system is used to structure both databases. Several classification systems, such as Masterformat, Uniformat, and CI/Sfb, are already widely used within the construction industry. These systems elementalize the building into assemblies and components and provide a logical hierarchy for organizing this information. Given its prevalence in the United States, the Masterformat system is used to structure the databases within the prototype system.

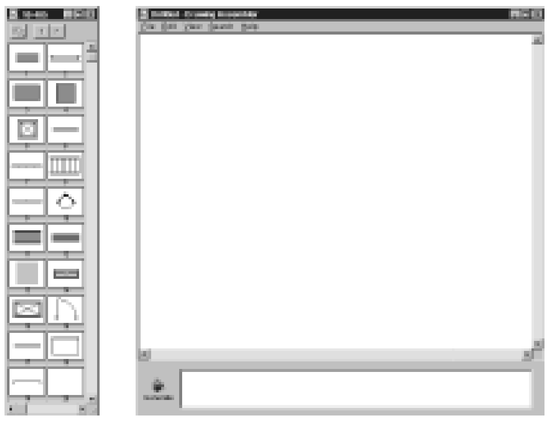

Figure 1. The Drawing Assembler *interface.*

As will be discussed in the next section, objects are entered into the component database through use of the *Object Editor*. As the object is created, the user determines the definition of the object. Similar to the keynote numbering system used in the American Institute of Architects’ *ConDoc* drawing system, the definition consists of two parts: the first part established by the Masterformat designation, followed by a suffix chosen by the architect. A 3’x7’x1-1/2” solid core wooden door, for example, would be defined as “08200_A1”. Similarly, a 6’8” version of the same door would be “08200_A2”. Detail drawings are organized within the detail database using the same system. Details are classified by the components assembled with that drawing. A detail illustrating the connection between 2x4 wood framing at 12” on center and a hollow metal door frame would be defined at “06062_A+08100_B2”.

3.1 *Object Editor*

Objects in the component database are created and modified using the *Object Editor* (Fig. 2). The *Object Editor* establishes the various graphic representations of an object (both plan and section) as well as its particular definition and parameters. Defining objects using the *Object Editor* requires three steps. In the first step, the object is defined according to the Masterformat system, as was described in the previous section. Next, the user enters values for the object's particular parameters. These parameters define the geometric dimensions of the object and are used to generate the resultant detail once a proper match has been made. In the final step, the user selects the graphic symbols that will be used to represent the object in the *Drawing Assembler*. These symbols are the primary means of manipulation in the *Drawing Assembler* but are not necessarily directly associated to the object's parameters. As in traditional drawing techniques, a simplified symbol can be used to abstractly represent a more complex geometric component. It is at the level of the detail that the particular geometric properties of a component will be illustrated in depth. Once a series of objects have been defined and made part of the component database, they can be organized into project specific libraries.

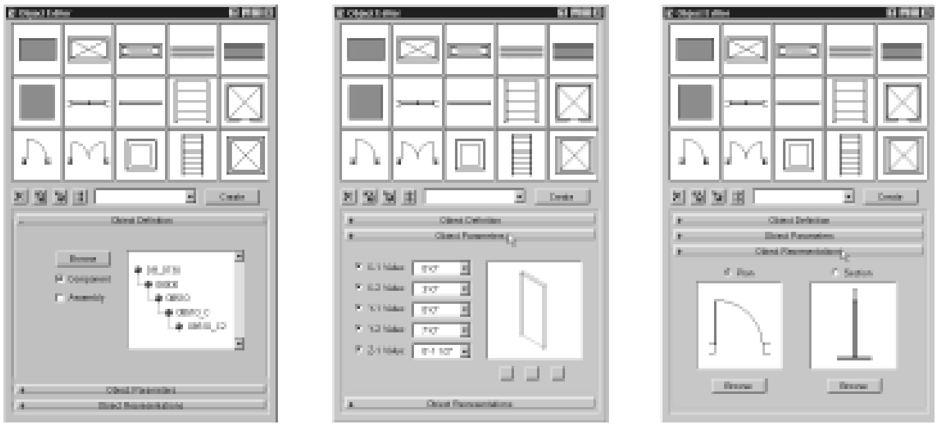

Figure 2. Object Editor *menus.*

3.2 *Drawing Assembler*

The main interface of the *Drawing Assembler* (Fig. 3) is used to assemble a set of components into the building representation. The *Drawing Assembler* uses a "drag and drop" input system. A plan or section is assembled by selecting a particular component from the library and dragging it to the work surface. This component can be positioned as needed. If the component happens to be a variable system, such as wall or floor type, the user is prompted to enter specific values to define this system.

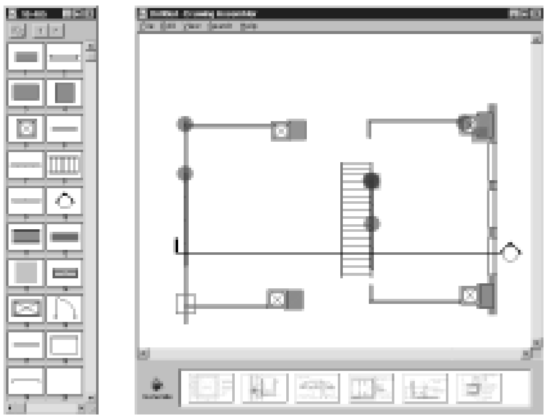

Figure 3. Generation Sequence - Details are generated based on the intersection of dissimilar objects.

When the building has been assembled to a reasonable level of completion, construction details can be determined by activating the "Generate" button. Once activated, the system isolates all of the intersections between dissimilar components. Each component's definition is identified and added to the intersecting component's definition, forming a compound definition for the particular assembly. The system then uses this compound definition to search the detail database. A double hung window, for example, intersects with a masonry wall component. The window is defined in the component database as "08XXX-D1". The masonry wall is defined as "04200-A". The system then searches the detail database for any details defined as "04200_A+08XXX_D1". Similar to Internet search engines, the *Drawing Assembler* will locate details that most closely match the compound definition. In instances where there is more than one matching detail, the user is presented with the selection from which to choose (Fig. 4). In this sense, the *Drawing Assembler* can act as either a strict documentation tool, searching for a specific detail, or as a more suggestive tool, providing the architect with a range of potential solutions.

Once a detail has been selected, the system reads the parameters for each component and generates the detail based on these values. The finished detail is then placed in the detail window located along the lower edge of the interface and a marker is generated at the intersection of the components to indicate the existence of such a detail. Similarly, if no details match the compound definition, a marker is generated to indicate that there is no detail for this specific intersection (Fig. 5). At this point, the user can either change the definition of the objects and search again or create a specific detail to address that particular condition. Throughout the course of this search and generate process, the *Drawing Assembler* maintains a list of the details generated and checks if instances of a particular detail have already been created in order to

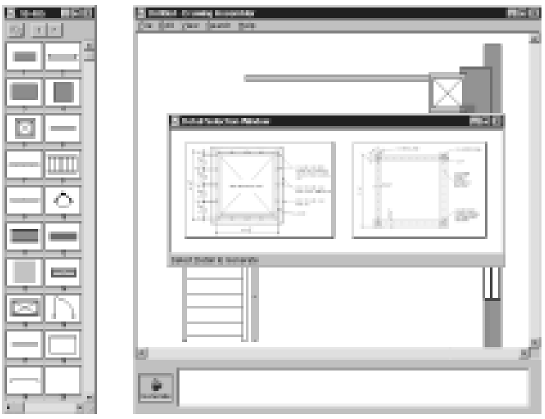

Figure 4. Generation Sequence – Detail selection window prompts user to select appropriate detail.

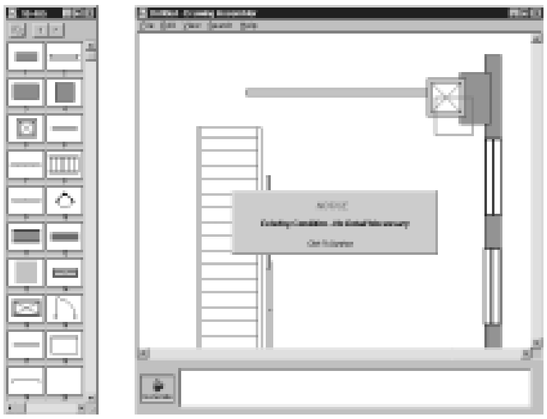
Figure 5. Generation Sequence – Alert window indicates that no matches were found in the detail database.

prevent redundant information. If a detail has previously been generated, the subsequent similar intersections are indicated with markers referencing the initial generation.

In addition to component objects, the *Drawing Assembler* also makes use of notational objects. The section object is used in the same manner as other component or assembly objects and is dragged and positioned on the work surface. Clicking the section object replaces the plan view with a section at that particular location. As was stated earlier, objects created in the *Object Editor* are defined with both plan and section views as well as parameters for each. The section object recognizes the objects it intersects and creates a view placing the section representations relative to their location in plan. However, when objects are entered into plan view, the *Drawing Assembler* does not request input regarding their position in the Z-axis. When the section object is first used, it is necessary to position the objects relative to the Z-axis. Again, the intention is to focus on the intersection of objects, rather than their precise location in three-dimensional space.

Understandably, a section often contains specific information that is not visible in plan. The section object works in a similar fashion to the plan view, allowing the user to drag and drop objects into place and then search and generate details. In order to relate the section to the plan, a plan object is created in the section, creating a linkage between the two representational views. In either case, it is possible to create multiple representations by creating either plan or section objects accordingly. This provides a consistent referencing structure among the various location drawings. Unlike the notion of a comprehensive three-dimensional building model, this approach makes use of the efficiency afforded by abstract two-dimensional representations while maintaining an integrated framework for building description and documentation.

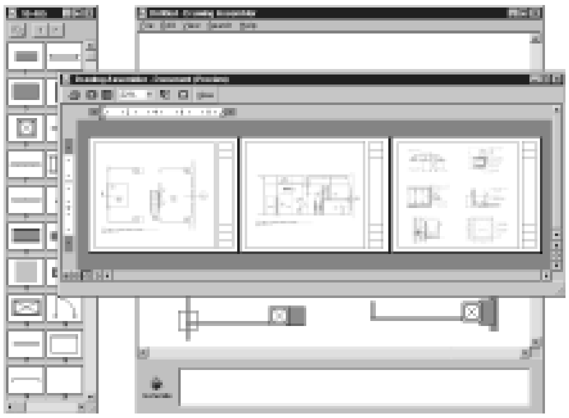
Figure 6. Print Sequence – Details and location drawings are generated and keyed appropriately.

Once a building has been assembled and the details generated, the *Drawing Assembler* organizes the information on drawing sheets. The user first specifies a particular organizational structure (by drawing type, element, or scale) and the system then places the details on the sheets according to this structure. At the same time, location drawings (plan, section, and elevation) are generated and appropriately keyed to the location of the details on the sheets, relieving the user of the tedious task of keying each detail's location. As a final step, the drawings are printed as a set (Fig. 6) and a file encapsulating the drawing information generated and saved.

4 Conclusion

While CAD software is often promoted as a means to achieve efficiency and productivity in the architectural office, it merely replicates known methods of work. Similarly, the current generation of building modelers, while certainly an improvement over traditional CAD applications, do not provide the depth of information required in a set of construction documents. The *Drawing Assembler* automates the process of construction documentation by providing the architect with instant access to an existing knowledge base of construction details. To further test the applicability of the concept, an AutoCAD version of the prototype is currently under development. Research into accessing remote databases through the Internet is also underway.

The intention of this project has been to develop a tool that supports the objectives of the architect through the strategic use of digital technology. Current methods of working do not fully exploit the potential of the technology. The *Drawing Assembler* challenges the traditional methodology of the architect through investigating alternative methods of working while rethinking the appropriate use of information technology in the practice of architecture.

5 References

Cox, Brad J. and Novobilski, Andrew J. (1991). *Object-Oriented Programming: An Evolutionary Approach*. Reading, MA: Addison-Wesley.

Gross, Mark D. (1996). Why Can't CAD be more like Lego? CKB, a program for building construction kits. *Automation in Construction 5,* 286 - 299.

Harfmann, Anton. (1993) Component-Based, Three-Dimensional "Working Drawings". In *ACADIA '93,* 141.

McCullough, Malcolm. (1996) *Abstracting Craft*. Cambridge, MA: The MIT Press.