Criteria for an object oriented library system of high–level parametric CAD elements

Ir. J. Willem R. Langelaan
Langelaan Architect
Mississauga, Ontario
Canada

ABSTRACT
The subject of this paper is the development of criteria and specifications for an object oriented library system of high–level parametric elements that have an integrated 2D and 3D representation. High–level elements are virtual representations of architectural elements such as windows, doors, etc. High–level parametric elements need few components to be flexible and easily customizable. The generalization of each element by its characteristic parts results in a substantial reduction in the number of polygons that must be processed by the computer during the 3D transformation, graphically clean 3D images and low demand on user intervention. Gestalt theory emphasizes the importance of contour lines for the perception of an element. The “minimalist” symbolic representation will simplify contour lines that enhance perception. The inherent flexibility and functionality of object oriented elements are augmented when the elements are rigorously developed as an object oriented library system, with classes and sub-classes of elements which inherit characteristics of the parent–class. Attribute values of a parent–class give the user global control over all instances of that class and its sub–classes in the model’s database. The concepts which Systems Theory uses for making an abstraction of reality are analogous to the concepts used in object oriented programming. This paper describes how Systems Theory is used as tool to develop high–level parametric elements as a functionally and computationally efficient library system.

1 INTRODUCTION

Conventionally an architectural design is developed, documented and communicated through paper drawings. An efficient alternative to manual drafting is to use vector based application software or solid modeling application software for the production of architectural drawings. A vector based CAD program stores information about building contours and surfaces. It does not store information about building elements. An object oriented solid modeling CAD program stores information about building elements, such as walls, doors, windows, roofs, etc. Additionally the database stores procedures for transforming the elements into a 2-dimensional floor plan or a 3-dimensional projection. These procedures can result in a floor plan with walls that are hatched and have a cutout with a window symbol. The database of a solid modeller gives the architect a new paradigm for working. The architect does not document the design in discrete orthographic projections such as floor plans, elevations and sections. Instead the architect documents the design in a digital model which is a virtual representation of the building.
that is designed. The architect then extracts information from the model. The information can be quantitative such as floor area and number of bricks, or the information can be graphical such as floor plans, sections, elevations, 3D projections, shadow studies, lighting studies, etc.

The model’s elements must be parametric. A parametric element has a set of attributes with values that can be edited by the user. These attributes are descriptions of the element such as width, height, number of window panes, etc. Before an element is stored in the database the designer must add values to the set of attributes to configure the appearance of the element. Initially the design and later the building are subject to change. The model must remain adaptable to changes which may include deletions, additions and changes to attribute values.

A library of high–level parametric elements enhances the functionality of CAD application software. The objects are virtual representations of architectural elements such as windows, doors, etc. The representation can range from minimalistic symbolic abstraction to detailed photo–realistic representation. There are 2 types of parametric elements, procedural and object oriented. When a procedural element is given attribute values and added to the model, a procedure is invoked which places a group of 2-dimensional primitives in the database. If there is a change to the model and the element must be re-configured, the group of primitives must be deleted from the database. The library element must again be added with new attribute values. An object oriented library element includes 2D and 3D procedures. When an object oriented element is configured and added to the model its parameters and procedures are added to the database. When the database is transformed into a floor plan or a 3D projection, the element’s procedures look up the parameters and transform the element accordingly. As a result the attribute values of an object oriented element can always be changed by the user to represent design or building changes.

Experts develop deeply rooted internal (mental) libraries of situations and materials they frequently encounter. Architects generally do not think of a building as lines on a blueprint but as a meaningful and functional assembly of building components such as walls, floors, roofs, columns, windows, doors, stairs. These are mostly elements which separate or connect spaces. It follows that a library of virtual building elements which are analogous to real world building elements would be logical entities for design. Such external representations can improve a person’s performance by effectively augmenting an architect’s memory. High–level objects are closely linked to the thought processes an architect uses in the process of design and therefore allow efficient use of limited cognitive resources like attention and short term memory (Johnson 1997).

Objects for a solid modeller must have an integrated 2D and 3D representation. The 2D representation must conform to generally accepted drafting standards and symbols. The 3D representation must be dimensionally accurate and its general morphology must
be in conformity with the intended reality. The function of the library elements is to be adaptable to design intent and to design changes with minimal manipulation by the designer.

The subject of this paper is the development of criteria and specifications for MasterLibrary (Langelaan 1992–), an object oriented library system of architectural elements which have an integrated 2D and 3D representation. Unless noted otherwise references to “2D” refer to a sub-set of graphical representations which are exclusively intended for representation on the floor plan.

The concepts which Systems Theory uses for making an abstraction of reality are analogous to the concepts used in object-oriented programming—both simulate the real world with objects which have attributes and attribute values in addition to internal relationships (methods) and external relationships. Systems theory includes semantics to describe relationships of, and similarities between system elements. For this reason systems theory was used as development tool for the library.

Gestalt theory is used as apologia for the visual acceptability of library elements with a morphology that is a simplified and symbolic representation of reality.

2 OBJECT-ORIENTED PROGRAMMING

2.1 Classes, objects, instances

The world can be viewed as an infinite set of discrete objects, each having individual characteristics, attributes, and methods. Each object communicates in its own way to other objects in the shared environment. Objects are classified based on particular sets of distinguishing criteria. A class is the abstract template specifying methods and attributes for instances of a class. A class can be modified and edited, and all objects related to that class are automatically updated. A particular object has particular values assigned to its attributes. An instance is a particular object of a particular class (Prograph 1989). Class syntax can be schematically represented as:

```
CLASS (Name, {Attribute1 : Value1, Attribute2 : Value2, . . . , AttributeN : ValueN} ,
{Method1, Method2, . . . , MethodN})
```

The attribute values can either be a numeric value, a text string or a Boolean value.

2.2 Inheritance

When a subclass of a parent class is created, the subclass inherits all attributes and the full library of methods of the parent class. A subclass is for objects which have a partially different set of attributes or methods than an existing class.
2.3 Encapsulation

Encapsulation is hiding information about an object’s attributes and methods. It is sometimes not relevant for the level of analysis to know how the method works or what all the attributes are.

2.4 Polymorphism

The same message can be sent to different objects. Polymorphism is the response of each object with a method which is appropriate to its class (Pinson, Wiener 1988).

3. SYSTEMS THEORY

Systems theory recognizes sets of discrete objects. Objects are the smallest elements which are considered in an analysis. The complexity of the object therefore depends on the specification level of the analysis. Objects have three aspects: attributes, attribute values and relationships. These relationships can be internal, i.e. among the objects which constitute the system or they can be external, i.e. with the objects which constitute the system’s environment. An object can also be a ‘black box’—a functional assembly of hidden objects and relationships. A black box is the set of objects which remains when a deductive analysis is terminated.

The environment is the set of objects outside the system with which direct external relations exist. It forms the system’s context. The universe is the set of objects outside the system with which no direct relations exist.

A sub-system is formed by a sub-set of objects which form a functional assembly (figure 1—region 01). It is an object set which is the result of an inductive analysis. A partial- or aspect-system is a set of objects which are grouped by a limited set of user defined attributes or relationships. The non-specified attributes are ignored (figure 1—region 02).

A more detailed level of analysis could lead to investigate a sub-aspect system (figure 1—region 03) (In ’t Veld 1975).
4. SEMANTIC ANALOGY

The semantic concepts which Systems Theory uses for making an abstraction of reality have many analogies with the concepts used in object-oriented programming—both consider objects which have attributes, attribute values and relationships to simulate the real world. Topological and morphological aspects of real world objects can be described in the model’s database as geometrical attributes and as relationships. The conceptual similarities can be attributed to the semantics of Systems Theory and object-oriented programming are represented in table 1. (Langelaan 1993)

The equality of the semantic elements “attribute” and “attribute value” facilitate to identify sub-systems and aspect systems.
### Table 1: Semantic Analogies

<table>
<thead>
<tr>
<th>Object-oriented Programming</th>
<th>Systems Theory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Context</td>
<td>Environment</td>
</tr>
<tr>
<td></td>
<td>System</td>
</tr>
<tr>
<td>Class</td>
<td></td>
</tr>
<tr>
<td>Instance</td>
<td>Object</td>
</tr>
<tr>
<td>Attribute</td>
<td>Attribute</td>
</tr>
<tr>
<td>Attribute Value</td>
<td>Attribute Value</td>
</tr>
<tr>
<td>Method</td>
<td>Relationship</td>
</tr>
<tr>
<td></td>
<td>Process</td>
</tr>
<tr>
<td>Encapsulation</td>
<td>Black box</td>
</tr>
<tr>
<td></td>
<td>Sub-system</td>
</tr>
<tr>
<td></td>
<td>Aspect system</td>
</tr>
<tr>
<td>Inheritance</td>
<td></td>
</tr>
<tr>
<td>Polymorphism</td>
<td></td>
</tr>
</tbody>
</table>

5. GESTALT THEORY

Seeing is a psycho-physical activity. The observed object becomes a subjective image. During the period from 1912 onwards (mainly 1920–35) the work of Wertheimer, Koehler, Koffka, and others emphasizes the rôle of “Gestalt” (configuration) in the processes of perception, etc. (Whyte 1951). Perception is both an innate and a cognitive process. Perception is the result of a psycho-physical connection between the observed object and subjective image (Hochberg 1964).

A criterion of Gestalt is the ability to deduce the permanent form from a set of contours. A contour is the boundary between a shape and the background. The observer knows that contour lines indicate the edge of an object.

One of the most important criteria of Gestalt is the fundamental independence from particulate elements—the whole appears to dominate the parts (Lorenz 1951). This ability is also evident in the minimum principle, the capacity to take in the relevant details and leave out the irrelevant ones: we see the easiest to perceive pattern. Less lines make a shape simpler and therefore easier to identify. Hence small details do not contribute to perceiving the whole. A simplified contour facilitates perception of the whole element.
6. A SYSTEMS ANALYSIS

Development of a library of high-level objects as a system can lead to increased functionality and efficiency. The added value is the result of, semantic consistency, morphological consistency, coordinated behaviour of instances, and centralized global control of common attributes.

A CAD library system is a virtual system in which the system elements are virtual digital elements. It is a deterministic system—the output can be predicted from the input. It is an open system—the system elements have relationships, input and output, with elements in the environment.

6.1 System environment

The environment of MasterLibrary is the ArchiCAD application software and the ArchiCAD GDL (Geometric Description Language) parametric programming language (Graphisoft R&D Rt. 1984— ). ArchiCAD is an object oriented solid modeler. GDL syntax is similar to BASIC. Every library part described with GDL has scripts, which are lists of the GDL commands that construct the 3D shape and the 2D symbol. Library parts can also include a description for quantity calculations in ArchiCAD.

6.3 Library system objective

Architectural design levels are characterised by a set of elements, but also by the type of situation in which the element is placed, such as neighbourhood, site, building, room (Bax 1989). When designing a building, architectural spatial and functional decisions are predominantly made at the building and the room design levels.

It is therefore the objective of the library system to provide design support in particular at the building and room design levels.

The task of the library elements is to implement and document change with minimal manipulation by the designer.

6.5 Library sub-systems

The library sub-systems consist of methods and library classes. These are,

- Graphical output sub-system
- Window sub-system
- Door sub-system
- Object sub-system
- Error detection and warning sub-system
6.6 **Library aspect systems**

An aspect system is a set of objects which are grouped by a limited set of attributes or relationships.

- User interface aspect system
- Morphological aspect system
- Topological aspect system

6.7 **Data output and input aspect-systems**

Output and input aspect systems are based on data exchange relationships between library elements and elements in ArchiCAD. The relationships constitute methods to exchange data which result in changes to attribute values. An output aspect system originates in elements of the library system and affects attribute values in elements in the system’s the environment. An input aspect system originates in the elements of the environment and affects attribute values of elements in the library system.

- Data output aspect-systems
  - Output to ArchiCAD element classes
  - Output to ArchiCAD element sub-system

- Data input aspect-systems
  - Scale aspect-system
  - Material colour aspect-system
  - Floor-to-floor height aspect-system
  - Wall curvature aspect-system
  - Relative position aspect-system

7. **DETAILED SYSTEM CRITERIA**

7.1 **Graphical output sub system**

The graphical output sub-system includes methods which transform the instances into a graphical representation. In the transformation the topological attribute values shall not be affected by the morphological attribute values. The topological attribute values take precedence over the morphological attribute values. Some classes must include methods to simplify the morphology.

- 2D transformation sub-system

The 2D symbol shall be parametric and the graphical representation on the floor plan shall be in conformity with the attribute values.
3D transformation sub-system

The 3D symbol shall be parametric and the graphical representation of the orthographic, isometric or perspective projection shall be in conformity with the attribute values.

7.2 User interface aspect system

The following criteria shall apply to the parameter list which is a part of the user interface sub-system,

- Consistent semantics
- Consistent sequence of attribute groups,
  - morphological attributes
  - topological attributes
  - material colour attributes
  - 2D and 3D modifier attributes

7.3 Error detection and warning sub-system

The parent class in a library system shall include methods that evaluate user input values. If a value is detected that would result in a morphological impossibility such as a self-intersecting polygon, a warning shall be printed on the screen.

7.4 Morphological aspect system

The requirement for morphological complexity of library elements is dependent on the scale of the design level and on the function and quality of the image that is generated.

This puts a limitation on the size of element components. The current standard for high quality printers and plotters is 600 dpi. This corresponds to a resolution of approximately 0.04mm. Generally screen resolution is 72 dpi which is a dot size of about 0.35mm. The closest proximity of contourlines on the screen is also 0.35mm. Thus the smallest contour that can be represented on the screen has a width of 0.35 x 3 = 1.05mm. A screen representation of 1mm corresponds to a real world dimension at the following scales which are typical for the room and building design levels,

<table>
<thead>
<tr>
<th>scale</th>
<th>minimum size</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 : 20</td>
<td>20mm</td>
</tr>
<tr>
<td>1 : 50</td>
<td>50mm</td>
</tr>
<tr>
<td>1 : 100</td>
<td>100mm</td>
</tr>
<tr>
<td>1 : 200</td>
<td>200mm</td>
</tr>
</tbody>
</table>

Table 2: Drawing scale and resolution
Certainly modeling elements which are smaller than 20mm have limited applicability. It is prudent to avoid components which are less than 50mm.

At the building and at the room design levels functional and architectural spatial quality are investigated and developed. Only those element components are relevant which affect spatial and compositional qualities.

7.5 **Topological aspect system**

Topological attribute values must form a steady state system. Morphological changes shall not affect topological attribute values.

7.6 **Data output aspect systems**

Within the ArchiCAD environment, two output aspect systems can be developed for the library system. One outputs data to the encapsulated ArchiCAD method for the Boolean operation to subtract an opening from a wall. This makes it possible to over-cut or under-cut an opening in a wall. This is necessary when the element total size attributes are interpreted as nominal attribute values.

The other outputs graphical data to another storey. Sometimes drafting practice requires that an element that is drawn on the floor plan of one storey must also be shown on a storey above or below. Furthermore the 2D representation may differ on storeys above or below.

7.7 **Data input aspect systems**

Within the ArchiCAD environment, input aspect systems can be developed for the library system.

7.8 **Scale aspect-system**

When a subset of the ArchiCAD database is transformed into a graphical representation of a floor plan, most vectors are scaled in conformity with the selected output scale. Typically text characters retain their specified point size. Text is scale independent. It has a constant “paper size”. It is a general accepted drafting practice that some symbols have constant paper size. A sub-system, a sub-class, must be created of such library elements whose output dimensions shall remain constant and are unaffected by the scaling factor that ArchiCAD applies to the transformation of the database.

Another universally accepted drawing practice is to draft the amount of graphical detail that is appropriate for the scale of the drawing. Generally the amount of graphical detail is proportional to the scale of the drawing. This means that some architectural elements are represented on the floor plan with a symbol that is scale dependent. A sub-system, must
be created of such library elements whose output symbol is affected by the scaling factor that ArchiCAD applies to the transformation of the database.

7.9 **Material colour aspect-system**

All library elements with a 3D representation shall have one or more attribute fields to specify material colour references for the components of the element.

7.10 **Floor-to-floor height aspect-system**

Some library elements must be configured in relation to the floor-to-floor height. These elements must include the option that the storey height value is received as input value from the ArchiCAD database. Such elements shall dynamically adapt to subsequent changes of the storey height.

7.11 **Wall curvature aspect-system**

In the ArchiCAD environment instances of the door class and the window class can only exist within a wall. The curvature of a wall shall affect the morphology of the mullions in 3D projections and on the 2D floor plan. The wall shall input into the door and window instance whether the wall is straight or curved and the radius of the curvature.

7.12 **Relative position aspect-system**

Elements that are placed together and that are intended to form a contiguous object shall join seamlessly. ArchiCAD does not yet support a data exchange between library elements. Relevant elements shall include an attribute with a Boolean attribute value (yes/no) and an encapsulated method that will enable seamless joining. In the case of seamlessly joining door and window frames, the encapsulated method shall include a means to reduce the mullion width to half width to maintain the specified frame width for the compound element.

7.13 **Design development and construction documents**

The virtual building model can be used for design development and also for extracting graphical information that can be used for construction documents. Construction drawings are mostly line drawings showing floor plans and orthographic projections such as vertical building sections and building elevations.

In the aspect systems that were discussed above, element attributes were exchanging data with elements in the environment. Other aspect systems can be found whose attribute values can lend a coordinated “global” behaviour to the 3D or the 2D representation of many library elements. Such coordinating attributes shall be removed from the library
elements and made into library system attributes and must be represented in a parent class.

7.14 3D representation aspect systems

The sections and elevations are derived from 3D transformations of the model’s database. It is a drafting convention that in sections and elevations all glazed areas in doors and windows are shown as opaque surfaces and all doors and windows are shown as closed. When evaluating design decisions it may be preferable to see what is beyond open doors and transparent windows. Rather than individually adjusting every door and window on every floor, it is preferable to globally change the status of glass, doors and windows. The parent class shall have attributes for,

- the opacity of glass
- the opening angle of doors
- the opening angle of windows.

Because the library system is developed as object oriented system, all sub-classes inherit the attribute values that are specified in the parent class.

7.15 2D representation aspect systems

Architectural floor plans are subject to a discrete set of drafting conventions for the representation of elements. These images are often symbolic and not representational. It is therefore not possible to derive them from the 3D representation. The floor plan is derived from a 2D transformation of the model’s database. Hence the model’s database requires additional data fields for the elements 2D graphical attributes and methods. Office standards for the production of drawings are intended to give a uniform and coordinated look to presentation drawings and to construction drawings, including floor plans. The function of a presentation drawing is not the same as for a construction drawing. Often different standards are used for a presentation drawing and for a construction drawing. However in the new paradigm these drawings are extracted from different generations of the same database. Rather than individually adjusting the 2D attributes of many elements on every floor, it is preferable to globally change the status of 2D attributes.

7.16 Text aspect system

A number of elements include text fields. The default font shall be a parent class attribute.

7.17 Scale thresholds aspect system

The 2D representation of a number of elements is transformed by scale. This is not a linear but a discontinuous transformation. The value of the scale thresholds shall be a
parent class attribute. The user can set the value based on office standard or based on the resolution of the output device.

7.18 **Door angle aspect system**

Door angles may not be shown as 90° degrees in plan. The value of the door angle shall be a parent class attribute.

7.19 **Closet door angle aspect system**

Closet door angles may not be shown as 90° degrees in plan nor be shown with the same angle as doors. The value of the closet door angle shall be a parent class attribute.

7.20 **Door arc line type aspect system**

The colour, line weight and line type shall be global attributes.

7.21 **Localized symbol aspect system**

Different countries may use different 2D symbol sets for like elements. An array of localized symbol sets must be a parent class attribute.

8 **CONCLUSIONS**

8.1 **Synergy**

Library elements shall be designed with only those parts which are most characteristic of the element’s typology and which are perceptible in a broad range of scales. The number of components exponentially increases the permutations of the element. Hence a few components suffice for a highly flexible and easily customizable element. The generalization of each element by its characteristic parts results in a substantial reduction in the number of polygons which must be processed by the computer during the 3D transformation. The synergistic effect is low demand on user intervention, fast rendering and graphically clean 3D images.

8.2 **Efficient use of cognitive resources**

When developing high-level CAD objects one is tempted to create virtual objects which closely resemble reality. For reasons of design and computational efficiency, library elements must be developed that have a simplified symbolic morphology. To evaluate the spatial function and quality of a design, the architect studies the proportion and relative location of the building elements and the resulting delimitation of space. The rich visual information that is present in a photo realistic representation obscures the perception of proportion and space. Gestalt theory emphasizes the importance of contour
lines for the perception of an object. The specified “minimalist” symbolic representation results in simple contour lines which enhances perception and does not force the architect to filter out extraneous information. Too much information distracts attention and fills memory with extraneous information (Johnson 1997).

8.3 Increased efficiency and functionality

By developing a library as System, additional efficiencies can be achieved for the architect. This is most notable when attributes are removed from element classes or subclasses and are placed in a parent class. This gives the architect an efficient means to globally change attribute values of instances that inherit the attribute.

High–level objects derive efficiency and functionality from the complexity of their pre–assembled parts and from encapsulated methods which regulate the geometry and interaction of the constituent parts. When modeling a building it is more efficient to use a single gesture to place a logical architectural entity such as a stair than to make multiple gestures for placing components such as risers, runs and stringers. The polymorphic aspect of object oriented elements has resulted in consistent semantics in the user interface.

8.4 Future developments

In the future, library classes will be developed that have a dynamic hierarchical attribute structure. Initially the default class attributes pertain to a generic morphology. As the design progresses and design decisions become more detailed, additional attribute groups which represent a higher morphological complexity can be invoked by the designer.

Another future development may be methods for adjacency detection. This could have the result that, for example, instances of elements which are adjacent and which are functionally related will appear as single object or, an instance may always snap to an appropriate location relative to another element.

9 REFERENCES


