An Interactive Cross-representational Data Modelling Approach for Early Stage Design Information

Linan Wang 1, Peter Szalapaj 2
1 School of Architecture, University of Sheffield, 2 School of Architecture, University of Sheffield
1 arponlw@shef.ac.uk, 2 p.szalapaj@sheffield.ac.uk

Abstract. This presents a systematic approach to the organisation of early stage design information, and the associated interactions among various types of representation. Scene-level representations are introduced as the complement of traditional object-level representations in an integrated information system. A cross-representational approach is proposed. The advantage of this approach is in the support of both user-definable representations, and cross-representational manipulations, including organisational transformation, information filtering, information aggregation, and simulations, which are essential for designers to dig into raw data. A prototype program called NetworkBuilder is being developed in order to explore the approach.

Keywords: Scene-level representation, attribute-based network model, declarative approach, cross-representational approach.

Introduction

This research started from an observation that although lots of work has been carried out to help architectural designers collect and refer to early stage design information; a systematic approach (theoretical and computational) has not yet been developed in the domain of Computer Aided Design. Existing research is either problem-specific (for example, the Constraint Manager (Gross & Fleisher, 1984)) or methodology-specific (for example, the framework based on Fengshui (Xu, 2003)). However, architectural design is not a simple collection of solved problems, nor is it dictated by any single design philosophy or methodology. Instead, it is an effort based upon interpretations of designers. A problem-independent design approach that supports information interpretation of designers is potentially beneficial.

Since the focus of this research is on the early stages of design, the accuracy of information is not the main concern. Rather, the ability to support flexible representations is more important. In other words, the representations that we are developing are user-definable and re-interpretable.

In this paper, we are trying to advocate a computational approach, which will allow early stage design information to be integrated through manipulations of representations, and then designers can then further filter information, transform and aggregate representations to interpret information actively and dynamically.
**Background**

There are three theoretical questions that need investigation. First of all, what are the objects that designers operate upon in the information interpretation process? Secondly, how can designers organise the process according to their own views? And finally, how should the design information be organised conceptually? We will explore these three questions in various domains of application.

**What are the objects to be operated on?**

We transform this question to two sub-questions, which are:

- What is the unique method architectural designers have employed in the information interpretation process?
- What are the principle objects in the unique method?

For the first sub-question, our conclusion is that design information is interpreted through dialogue between the external world and the internal world, based on research on sketching in the context of the design process (Galle & Kovacs, 1992; Schon, 1992; Schon & Wiggins, 1992) (Kavakli & Gero, 2001; van der Lugt, 2005; Verstijnen, van Leeuwen, Goldschmidt, Hamel, & Hennessey, 1998). Although there are arguments about the importance of sketching (Bilda, Gero, & Purcell, 2006), we consider sketching as the typical dialogue form adopted by designers. In order to support efficient dialogue, it is necessary to allow users define their own representational forms since different designers will organise information in different ways (Kitchin, 1995).

Traditionally, in CAD systems there is only one level of representation for users to operate on. In drawing systems like AutoCAD, for example, the basic representational elements are primitive geometrical entities like lines, circles, rectangles, etc. They can be created, modified and assembled into more complex ones. In design systems such as Constraint Manager (Gross, Ervin, Anderson, & Fleisher, 1988), the basic representational elements are constraints with higher levels of abstraction. However, this kind of bottom-up approach is not the way we perceive the world. According to cognitive psychology research (Downs & Stea, 1977), people perceive a collection of objects as a scene (or frame of reference, in some literature (Moore & Golledge, 1976)) at a time. Later, we explore lower level details of objects based on scenes according to needs. A ‘scene-level’ of representations is essential to support the top-down approach. In this paper, we are trying to argue that this kind of top-down approach is a necessary complement to the bottom-up approach.

Therefore, we further categorise objects into two levels, which are the ‘object-level’ and the ‘scene-level’. It is important to make it clear that the distinction between the two levels is relative. For example, in a representation that is aimed at revealing the geographical relationships among buildings, the scene-level representation is a map and the object-level representations are references to buildings, which could include photos, shapes, names, and even computed graphs. In a representation of the structure of a building, the scene-level turns out to be the building and object-level is its components.

![Scene-level and object-level representation](image)

The notion of scene-level representation is not limited to natural perceived scenes. It supplies structures to organise information of objects and maintains relations. A scene could be interpreted as context holders in which designers depict certain attributes of objects, without having to explicitly list all the attributes. For example, with maps, designers actually manipulate the location, size and shape attributes
of objects. This idea can be extended to time-line, colour space, etc., for designers playing with time and colour attributes.

The differentiation of object-level and scene-level representations makes it possible to decouple the two from representations. This decoupling has the potential to be user-definable and re-interpretable, such as the representations demonstrated by MVRDV (MVRDV (Firm) & Maas, 1999).

How to organise dialogue?
Traditionally, design activities were tackled from the procedure-centric point of view. Metaphors and analogies (Erickson, 1996; Newton, 2004; Wade, 1977) (Bertola & Teixeira, 2003) have been widely used as prototype models, upon which the frameworks and computational applications have been developed. The immediate consequence of these approaches is that in order to be faithful to real world activities, conceptual frameworks have to retain a certain level of ambiguity and abstraction. Thus the development of computational implementations is difficult.

On the other hand, after narrowing down the scope and usage of the conceptual frameworks, the computational implementations are bounded with three limitations:
- They can only handle a subset of all the available information.
- They can only handle certain aspects/problems of design.
- Inherently they are prescriptive systems (Bijl, 1989).

We have adopted a declarative approach to avoid the above limitations. Instead of imposing a global conceptual framework that dictates the information interpretation process to designers, we have tried to provide a practical computational approach that supports the constructions of scene-level representations, manipulations on representations and interactions among the representations. We prefer to leave the interpretations of both the representations and the designers’ operations on the representations to designers themselves.

On the implementation side, one of the advantages that the declarative approach brings is that we do not need to invent data structures that correspond to procedures. This approach has reduced the programming difficulties. On the other hand, since it’s impossible for us to implement all kinds of scene-level representations, the system needs to supply an easy way for designers to create new forms. The idea of decoupling the scene-level and object-level eases this task. Designers can only create new scene-level or object-level forms, and the rest will be taken care of by the system.
**How to organise information conceptually?**

The organisation of information becomes a key question for the implementation, as well as for users to mentally organise multiple forms of representation. The same building is represented as its shape on a map, a picture on the time-line, and the scene-level representation on a component view. Without a central concept of the organisation it could be confusing. On the other hand, without the information organisational concept, various representations are just discrete forms so that the interactions among representations will only remain in theory. Based on the IRN (Inter-Representational Network) theory (Haken, 1996) and Conceptual Spaces theory (Gärdenfors, 2000), we propose an ‘attribute-based network’ model to organise information. This model has following properties:

- Nodes of the network are references of objects. Each object has multiple attributes like shape, color, name, as well as sub-objects.
- Nodes of the network are connected through attributes. The links reflect how objects influence each other.
- Nodes of the network are retrievable via queries.
- The network has the potential to be transformed into other information representations for computational tasks.

**The proposed cross-representational data modeling approach**

Based on the previous theoretical investigation, we propose a ‘cross-representational’ approach to model the early stages design information.

- Information is organised conceptually in an attribute-based network.
- Users construct and interact with scene-level representations as media.
- Users can create new forms of representation.
- Users can access and interact with objects within scene-level representations.
- Users can construct, transform, and integrate the scene-level representations.
- Simple simulations are supported.

**The prototype implementation**

We have developed a prototype program called Network Builder to explore the proposed approach.

We chose to program in the Mac OSX operating system, in Objective-C language within the Cocoa Framework. Cocoa was derived from NEXTSTEP and OPENSTEP programming environments developed by NeXT in the late 80s. After Apple acquired NeXT in 1996, it became the core programming environment of Mac OSX operating system (Singh, 2007). It has proved to be an efficient prototyping environment. The most famous example is the first World Wide Web browser (Andreessen, 1993).

One of the strengths of the Cocoa Framework is that it is a strict application of the MVC (Model-View-Controller) principle. It is worth noting that the MVC principle shares some similarities with our understanding of design information-representation-user interactions. For the model part, the Cocoa Framework supplies the Core Data technology on which we implemented the attribute-based network. For the view part, we relied heavily on its Core Animation technology to draw the representations on the screen. For the controller part, we tried to supply an extendable framework that allows users to have the possibility of creating their own representational forms.

**Data structure**

We used three data schemes to implement the attribute-based network. The first scheme ‘DataAttributeType’ has two data fields: dataType and name. The value dataType field could be 0~6, which stands for 7 possible data types: number, text, time, location, color, image and shape. It allows the extension to more data types. Users can define new attribute types based on the 7 data types by using the ‘Attribute dictionary window’ provided. For example, users can create height and area attributes based on
one data type: number.

The second scheme ‘DataAttribute’ has three data fields: type, object, and rawData. The type field refers to a specific attribute type in the first scheme. The object field refers to the owner object in the third scheme. The rawData field holds the binary representation of the value of the attribute. This design make it possible to support various data types as long as they can be converted to binary format.

The third scheme ‘DataObject’ has four fields: attributes, children, parent and name. The attribute field is a collection of attributes in the second scheme. The children and parent fields pair is designed to reflect the hierarchical structure of objects.

The schemes are implemented using Core Data technology. We do not have to deal with low-level operations like file reading, writing, locking and caching. Three other features are beneficial:

- The first feature is that Core Data supports KVO (Key-Value-Observing). With few more lines of additional code, the application will be notified when values of objects are changed. It’s especially helpful when an object has multiple representations: when its attributes are changed in one representation, the other representations will update automatically.
- The second feature is that it is possible to change the data storage type from local file to network enabled database. The application can be further developed to support multiple users working on the same project at the same time.
- The third feature is related to iPhone, the mobile computing platform. The iPhone uses the Cocoa Framework with the CocoaTouch extension, which supports the multi-touch technology. With little modification, the prototype program can be migrated to the iPhone platform.

Object-level representation

Object-level representations are implemented based on the NSView class supplied in the Cocoa Framework. They can be moved, clicked, dragged and dropped on their containers, the group level representations. To support user-definable representational forms, we provided three basic drawing methods: drawShape, drawText, and drawImage methods corresponding to attribute types. With a few lines of code, users can combine the three drawing methods to create new representational forms. It is also possible to add new drawing methods, such as 3D modeling, for example.

In some types of scene-level representations, the moving, dragging and dropping of objects change the value of certain attributes. For example, on a map, moving an object adjusts the location attributes. We provided a pair of unified methods to access and change the value of attributes programmatically:

- (id) attr:(NSString *)attrName;
- (void)setAttr:(id)val for:(NSString *)attrName;

In addition, we provided an object inspector that allows users to display and edit all the attributes of one objects in a single window.

Scene-level representation

We have developed three basic representational forms: map, timeline, and mind-map.

Maps are probably one the most widely used representational forms in design. They provide contexts for designers to visually manipulate and interpret geographic information. We have integrated the Google Map API into the form, which allows users to setup background map. When users move objects on a map, the geographical information of objects is updated automatically.

The timeline representational form is used to demonstrate the manipulation of non-visible attributes. Users setup the time period, and then move objects on the representation so that temporal attributes of objects are assigned.

The mind-map representational form is introduced to help designers organise information conceptually. It helps designers to focus on the relationships among objects other than physical attributes.

Following the same principles, the prototype application enables users to create their own representational forms. To create new scene-level
representations, users only need to supply the logics that deal with the operations on objects, such as moving, allocating, and link establishing.

**Manipulating scene-level representations**
Simple manipulations on object-level representations are the basis of manipulation of scene-level representations. Representations can be interpreted as channelised informational organisation (Kelly, 1955). To develop interpretations of information, channels need to be re-integrated so that more meanings can be extracted. In this paper, we try to advocate three computational tools to achieve the goal, which are organisational transformation, information filtering, and aggregating multiple representations. Since each object is represented in terms of multiple representations, and users can access the attributes cross-representationally, existing representations can be transformed to other ones with help of Graph Theory.

Furthermore not only attributes of objects can be represented. We have embedded a simple language F-Script interpreter and a command line console inside the system, so that users can access attributes of objects and execute simple computational tasks. For example, we can input a command that filters out certain objects on a map, based on the temporal attributes given on the timeline. The filtering of information could help designers to experiment with selected attributes and visually assess the consequences.

Maps, timelines and mind-map representations are all 2D or 1D representations. It is natural to aggregate one 2D and one 1D representation can be combined into a 3D space. The three tools can be further used in a combinational style. Along with extendable scene-level representational forms, and user-definable object-level representations, this system supplies a rich set of interpretational tools.

**Simulation**
The attribute-based network model and the embedded language make it possible for designers to visualise the relationships among objects dynamically. The change in one attribute of an object, will affect the surrounding objects in the real world, and the effects will be propagated in a non-linear style. These connections are invisible but important for architectural design.

The Network Builder provides a framework for simple simulations. Designers need to first tell the system how two objects (source object and destination object) are influenced in the F-Script language, for example:

```plaintext
delta_price:= (source attr:`price_delta') * (0.6/distance).
price:= (dest attr:`price') + delta_price.
dest setAttr:price for:`price'.
dest setAttr:delta_price for:`price_delta'.
```

The first three lines of code relate the change of a property's value to its neighbour. The variable 'distance' is calculated by the system and given to designers. The last line sets the value change of this property for the next round of the propagation.

After having the connections setup, the next step is to visualise the changes. One way to visualise numbers is to convert numbers to colours, like:

```plaintext
red:=(dest attr: `price')/max_price.
colour:=NSColor colorWithCalibratedRed:red
green:0.0 blue:0.0 alpha:1.0.
dest setAttr:colour for:`colour'.
```

The routine 'NSColor colorWithCalibratedRed:green:blue:alpha:' is supplied by the Cocoa Framework to create new colour from red, green, blue and alpha components. The above three lines of codes convert the numeric attribute value to a red colour which the depth of the red component has a linear relationship with the value. When running the simulation, the colors of objects are changing accordingly. Designers can visually access the consequence of an attribute's change.
A hypothetical case study

We have conducted a hypothetical case study to test the prototype program. It is aimed at redesigning the Moor, a commercial district in Sheffield city centre, UK. Since it is an urban planning project that involves multiple stakeholders, we firstly modeled the district from multiple perspectives, including visitors, business managers and designers. Having these representations on hands, we were able to dig out more information by means of cross-representational manipulations and simulation.

Multiple representations can be developed from the above perspectives, including: visiting route maps, business logistics and shopping flow (Figure 4), analytical diagrams (Figure 5), time-line, resting squares map, etc.

Cross-representational manipulations and simulation

Analytical diagrams, visiting route maps, and business logistics and shopping flows are all map based. They can be combined into an interactive pseudo-3D representation as shown in Figure 6.

To explore the time-geographical configuration of the cultural landmarks in the area, we integrated the map and timeline into a temporal-geographical pseudo-3D interactive model. Designers can rotate, zoom, pan the model, and click objects in the model to obtain detail information.

Figure 7, 8 are the steps towards a market activity simulation in the main shopping area. Based on the resting squares map, and the visiting routes, the people density distribution was computed out (Figure 7). The grayscale represents the density. The darker the higher density is of the 20 x 20 metres square. The store distribution was also integrated in
Figure 7. Having the people density and stores setup, we started the agent-based simulation (Figure 8). Each autonomous individual has his own shopping schedule that is generated based on the two typical shopping task lists. The bars besides the shopping points denote the total sales. It could be further extended to include the tax rate, which is useful for government to make related decisions.

**Conclusion**

Emphasis has been placed in this paper upon the need to accommodate user-definable, reinterpretable representations and support interactions among the representations. This approach addresses key representational issues including how to organise information across representations and the corresponding interactions.

A rich set of representations has been developed in various domains to help designers extract meanings from raw data. We need to re-integrate channelised information to reach full views. The ‘scene-level’ representation introduced here complements the ‘object-level’ representation, since it supplies the context within which designers interact with objects.

We have developed the prototype application under the influence of the description system idea demonstrated in the MOLE project (1984-1992) (Tweed, 1999), that it could accommodate unforeseen changes, and most importantly, users would be able to define and re-interpret new representational forms.

**References**


MVRDV (Firm) and Maas, W.: 1999, Metacity datatown, MVRDV/010 Publishers, Rotterdam.


Xu, J.: 2003, A framework for site analysis with emphasis on FengShui and contemporary environmental design principles