Appeared in Knowledge-Based Systems Journal, Elsevier Science, Vol.13, No.6, pp.375-384, November, 2000.

Hands-on Representations in a Two-Dimensional Space for Early Stages of Design

Yasuhiro Yamamoto[1], Kumiyo Nakakoji[1,2,3], Shingo Takada[4]
[1]Graduate School of Information Science, Nara Institute of Science and Technology
8916-5, Takayama-cho, Ikoma, Nara, 630-0101, Japan
{yasuhi-y, kumiyo}@is.aist-nara.ac.jp; http://ccc.aist-nara.ac.jp/
Tel: +81 743-72-5381; Fax:+81 743-72-5383
[2]Software Engineering Lab., SRA Inc.
[3]PRESTO, JST
[4]Faculty of Science and Technology, Keio University
michigan@doi.cs.keio.ac.jp

ABSTRACT
In design, problem analysis is as important as solution synthesis. Strategic knowledge is required not only for constructing a solution but also for framing a problem. While externalized representations play critical roles in design tasks, different types of representations are necessary for different stages of a design task. In early stages of a design task, design support tools need to provide *hands-on representations* with which a designer can easily perform trial-and-error and examine the whole as well as parts of the whole, allowing the designer to represent any levels of preciseness as he/she likes. Sketching and drawing with paper and pencil provide an ideal representation for this process. But what about supporting design domains, such as writing or programming, where no sketching exists? In this paper, we argue that two-dimensional positioning of objects in a design support tool serves for the same purpose as sketching does for architectural design. Two-dimensional positioning allows a designer to produce hands-on representations that "talk back" to him/her without forcing the designer to formalize or verbalize what to be externalized. Two systems, ART for writing and RemBoard for component-based programming, illustrate the framework.

KEYWORDS: computational support for early stages of a design task, representations for strategic knowledge, two-dimensional positioning, design theories, cognitive models

1. INTRODUCTION

In ill-structured design, a problem and a solution co-evolve [1,2]. In architectural design, writing, or programming, for instance, what components need to be constructed (problem analysis) and how they need to be integrated (solution synthesis) depend on each other - *parts* define the *whole* but the roles of *parts* are defined by the *whole*; a design process can be viewed as forming a hermeneutic circle [3].

Throughout a design process, a designer is engaged in a cycle of producing a representation (such as sketches, mockups, notes and solution forms), and reflecting on them [4]. The externalized representations serve as a "situation" that talks back to the designer. During the process, the designer has a conversation with a material asking questions such as:

- what parts are missing;
- how much the designer is "sure" about a newly created part;
- what the role of this newly created part is in terms of the whole design;

- what the role of this newly created part is in terms of other parts; or
- which direction the whole design is moving toward and whether the direction is in accordance with the intention behind the design.

A type of strategic knowledge that our research focuses on is related to low-level design decisions required to address these questions.

Such strategic knowledge is necessary in early stages of a design task when understanding what the problem is plays a large role. The strategic knowledge is required to explore a possible problem space by uncovering implicit requirements, by making trade-off among conflicting goals, and by setting up constraints.

Appropriately applying the strategic knowledge to a design situation requires the “right” kind of external representation. Most existing design-support tools focus on providing representations in a solution domain. CAD systems, for instance, allow designers to produce detailed pretty-printed representations, allowing them to do precise simulation and detailed analysis of a produced artifact. On the other hand, these representations support designers very little in answering questions listed above.

In early stages of a design task, designers produce rough sketches using paper and pencil. Programmers record hand-written notes. Not all of those externalized representations would constitute the final solution. Most of sketches are in fact, drawn as a part of a trial-and-error process or produced as a result of doodling, and abandoned later. Those representations serve for the application of the strategic knowledge in framing a problem by triggering creative ideas.

This paper presents our approach of supporting early stages of a design task by providing representations that are suitable for the application of the strategic knowledge. In early stages of a design task, design support tools need to provide *hands-on representations* with which a designer can easily perform trial-and-error and examine the whole as well as parts of the whole, allowing the designer to represent any levels of preciseness as he/she likes. Acknowledging the widespread use of sketches and drawing, these can be considered as ideal representations for early stages of architectural design. We argue that two-dimensional positioning of objects serves as the same purpose as sketching does for domains where no sketches exist, such as writing or programming. We focused on the positioning of *objects* in the space and what types of positioning, including placing, moving, resizing and merging, emerge during the design process. Such *objects* can be any type of representations including parts of a final product, comments, or design rationale.

In what follows, we first argue for externalized representations for early stages of a design task. We discuss sketches and drawing as one of ideal representations and identify requirements for such representations. Section 3 presents two-dimensional positioning of objects as an alternative representation for domains where no sketches exist. Section 4 presents two prototyped systems, ART for writing and RemBoard for component-based programming, which illustrate the framework. The following sections discuss the two systems and conclude the paper.

2. EXTERNAL REPRESENTATIONS FOR EARLY STAGES OF DESIGN

The design process requires both generating parts and structuring them (solution synthesis) while exploring what to design (problem analysis) [1]. One cannot understand a problem without having started solving it. A partially constructed solution helps uncover problems. In design, problems and solutions co-evolve.

While they are inseparable, types of cognitive activities that designers are engaged in would typically change as stages in a design task proceed. During the early stages of a design task, designers focus more on understanding and identifying problems. As the design proceeds, the designer's focus shifts toward synthesizing solutions. Types of strategic knowledge that the designer applies thereby change as the design stage proceeds.

Different types of representations are required for different types of strategic knowledge. This paper focuses on what external representations that computational tools need to support for the application of strategic knowledge in early stages of a design task.

In this section, we first discuss the importance of external representations in design, and describe a spectrum of different types of external representations used during a design process. Section 2.2 describes representations necessary for early stages of design, and Section 2.3 argues that sketching and drawing are ideal for such representations.

2.1. Different Roles of External Representations in Design

The power of externalization cannot be overemphasized. Bruner [5] comments that externalization "produces a record of our mental efforts, one that is 'outside us' rather than vaguely 'in memory' ... It relieves us in some measure from the always difficult task of 'thinking about our own thoughts' while often accomplishing the same end. It embodies our thoughts and intentions in a form more accessible to reflective efforts" [5; p.23].

Even with this recognition of the importance of externalization, not many design support tools take into account what representations and interactions are appropriate in what design stages. Simply supporting external representations for a solution form will not be enough, especially in early stages of design. We need to carefully examine what representations a designer uses to most efficiently apply the strategic knowledge, and what types of interactions are appropriate for the designer not to disturb their thought processes.

Designers produce various types of representations for different purposes throughout a design process. There is a spectrum of types of representations serving for different purposes. At one end of the spectrum, representations serve for solutions, while representations at the other end serve for problems. Representations that serve for problems are produced mainly in early stages of a design task. Those that serve for solutions are produced as the design proceeds to late stages.

Representations that serve for solutions are produced more toward late stages of a design process. The designer produces such representations with the intent to evolve them into a final product. The purpose of this type of representations is to share and communicate ideas underlying the design. Therefore, such representations need to be communicable and sustainable among design stakeholders being understood in the same (or similar) manner as the original designer intended.

Most existing design support tools are oriented toward supporting representations that serve only for solutions. CAD systems and most word processing and presentation tools (such as Microsoft Word and PowerPoint) are typical examples. They display how an artifact would look when it is finished as a WYSIWYG interface. However, those representations do not support a user deciding what to write through a trial-and-error process. Another example of the lack of appropriate representation for early stages of a design task is bookmark facilities of major Web browsers. A typical Web search task can be viewed as an ill-defined design task; understanding what to look for depends on what intermediate search results the user obtains. A bookmark facility, which organizes a list of URLs in a hierarchical structure, relieves a user of remembering important and relevant URLs. Appropriately organized URLs are useful for a revisit to those sites when necessary. However, they are not useful when a user is exploring a number of Web sites opening many windows looking for certain information. The user’s goal is not to neatly organize those temporary encountered Web sites; a way to represent the user’s understanding about each of those Web sites (e.g., how important it is, how relevant it is, or how it is related to one of the previously found ones) is eagerly needed during such a search task.

2.2. Representations for Early Stages of Design

In early stages of a design task, in contrast, designers produce representations that are not necessarily used in a final design artifact. They use such representations not as a direct contribution to a solution but as a means to apply the strategic knowledge, which is necessary to understand problems.

Such representations may take the form of drawings, textual annotations, notes, coloring, sizing or positioning of objects. Designers heavily use visual cues to reflect in the current situation [6]. One small aspect of a representation, such as the straightness or the thickness of a line, may play an important role in helping them understand the problem.

The “meanings” of these representations may be vague and fluctuate. Designers may use such representations simply as a reminder. It is impossible for a system or for other designers to objectively identify the underlying meaning behind the representation, as it is not created for the purpose. Such representations are the results of externalizing informal, non-linguistic, and non-symbolic application of strategic knowledge. The representations are processed by a designer perceptually rather than cognitively, exploiting human perceptual abilities [7]. Even the designer who produced a representation may not be able to understand why he/she made the representation.

2.3. Sketches and Drawings for Early Stages of Design

One of the most widely-used representations that serve for problems is sketches and drawings using paper and pencil [8]. Sketching “mediates and facilitates thoughts, and design ideas emerge as a result of this interaction” [8, p.128]. The designer can externalize his/her thoughts using a pen, which requires minimum cognitive load on a sheet of paper, which is two-dimensional space. The designer can reflect in while drawing, and reflect on what has been represented [4].

While drawing, a designer keeps making hypotheses and verifying them, gradually uncovering design ideas through a trial-and-error process. For instance, when designing a floor plan, an architectural designer keeps asking questions to him/herself "*what if I put a refrigerator here*;" "*what happens with the door*;" or "*how the sunlight comes in*" while repeatedly draw lines and circles on his/her sketches.

Lawson [9, p.242] suggests that there are two important characteristics for sketching and drawing to be useful in early stages of design:

- It should not show or suggest answers to questions which are not being asked at the time; and
- It should suggest only a level of precision which corresponds to the level of certainty in the designer's mind at the time.

Sketching with paper and pencil works best as a representation for early stages of design because a designer has full control over producing a representation. It does not require a designer to make unnecessary commitment in externalizing a representation.

Lawson also suggests that sketching on a sheet of paper imposes a natural constraint on a designer that helps him/her understand what has been drawn:

- It should not draw at a size larger than A4 or A3 to keep everything in mind at once.

Sketches drawn on a large sheet of paper cannot be viewed without head or eye movement. It seems really critical for a designer to be able to see everything at once.

The critical aspects discussed above make sketching as a useful representation that serves for design problems. We call representations that have these aspects *hands-on representations*. Sketching and drawing with paper and pencil best serve as a representation for early design because they provide *hands-on* representations for a designer.

We can support early stages of architectural design by supporting sketching and drawing. But how can we support other design domains where no sketches exist?

This paper argues that two-dimensional positioning of objects serves as a representation for early stages of a design task for design domains such as writing and component-based programming in the same way as sketching does for architectural design. The following section discusses how two-dimensional positioning help writers and programmers as hands-on representations.

3. TWO-DIMENSIONAL POSITIONING AS A REPRESENTATION FOR EARLY DESIGN

In some design domains, such as writing or component-based programming, sketching does not exist because elements necessary to represent ideas either already exist or are electronically constructed in computer systems. For instance, let us think of a situation where a user wants to write a grant proposal based on previously conducted email discussions. Writing a proposal is a design task, and while deciding what the main theme would be in early stages of the writing process, the user needs to reorganize pieces of text contained in the email messages through trial-and-error processes.

We use two-dimensional spatial positioning for those domains that do not have sketches. This section first discusses how to support representations for early design on a computer system, followed by a discussion on how two-dimensional spatial positioning works as a hands-on representation for strategic knowledge in early stages of design.

3.1. Computer Support for Hands-on Representations

Based on the important aspects of sketching with paper and pencil as a hands-on representation (as discussed in 2.3), we have identified a list of requirements for computational tools that support representations for early stages of design:

- to allow multiple interpretations;
- to make it easy to assign a meaning to a representation;
- to make it easy to grasp a meaning from a representation;
- to look at the whole at a glance;
- to remember what has been done; and
- to identify what was being looked at.

These requirements are tightly related to the issue of cognitive overload [10] in human-computer interaction. Cognitive overload happens when a user is forced to remember, decide and perform things that are not the goal of the user in order to achieve the goal. Application of strategic knowledge is a cognition-intensive task and cognitive overload hinders the effective application of the knowledge.

A typical example of cognitive overload in a design support tool is when the designer is forced to select a menu item and to choose a pen-width *before* starting to draw. When a designer is dealing with representations for early stages of a design task, the designer's consciousness and thought need to be focused on what to externalize, and not on how to implement the externalizations. The designer generates a representation to uncover what the intention is; it is not that the designer has an explicit intention a priori and then externalize the intention to reflect on it [4].

Our goal is to design and build a computational environment that provides designers a medium to which designers apply strategic knowledge in early stages of a design task. Such representations would then allow them to *listen to* the back-talk of the situation.

We have studied a concept called *Representational Talkback* [11] for the goal. Representational talkback, based on Schoen's design theory [4], is defined as: "perceptual feedback to the human designer from the externalized design artifact." Representational talkback is an intermediate situation that emerges during a design task. We focus on visual, perceptual representation rather than textual representation because the type of strategic knowledge we are interested in supporting is typically non-symbolic, non-quantifiable, and non-verbalizable. Perceptual external representations "provide information that can be directly perceived and used without being interpreted and formulated explicitly" [7], and external pictures can give people access to knowledge and skills that are unavailable from internal representations [12].

A computational medium can support a designer in the early phase of his/her design task by amplifying the representational talkback. The amplification of representational talkback is concerned with two issues: how to make it easier for designers to externalize what they want to express, and how to make it easier for designers to understand what has been represented.

Our approach toward this problem is the use of two-dimensional spatial positioning of objects. The following section describes the rationale for this approach.

3.2. The Use of Two-Dimensional Space Positioning

This paper presents our approach of using two-dimensional spatial positioning of objects as a representation that serves for design problems. By two-dimensional positioning, we refer to a two-dimensional space, a process of placing and moving objects on the space, interaction with objects on the space, and interaction with the space itself.

Instead of paper and pencil, we argue that we can use computer tools to provide such two-dimensional positioning for writers and programmers. With the two-dimensional space on a computer, there are things that are easy in a real world but difficult in the system. For instance, in reality, it is easy not to place one object on top of another physically, while in a computer, special implementation is necessary to impose such constraints. On the other hand, there are things that are impossible in the real world but possible and easy on a computer. For instance, it is easy to duplicate an object on a system, move an object, or "undo" moves. A system can also handle media that are not possible on a sheet of paper, for instance, displaying a video clip on a two dimensional space.

With the direct manipulation style, it is easy to grasp and move objects to produce different visual properties. Simply looking at the space will help people identify a vast amount of visual properties from the space. Thus, the use of positioning as a representation addresses the concerns mentioned to amplify representational talkback.

A two-dimensional positioning can have a variety of visual properties. Such properties

Table 1: Examples of Properties in Positioning

Unitary properties	*size of the object, color of the object, form of the object, …*
Local-relational properties	*above, below, on top of, next to, close-to, far-from, …*
Global-relational properties	*in the top-left corner of , far away from others, …*

include unitary properties, local-relational properties, and global-relational properties. Unitary properties illustrate inherent visual properties of the object, while local-relational properties refer to those that are identified in terms of other objects. Global-relational properties refer to those that are identified in terms of the whole space. Table 1 illustrates examples of such properties.

We use positioning of objects that are representations for solutions. In the domain of writing, for example, we provide a way to position a set of text "chunks" that can be freely mapped on a two-dimensional space (see Section 4.1). While positioning, designers can use those properties to externalize a variety of situations. For instance, if a designer (in this case, a writer) thinks that a paragraph-A is better than paragraph-B, then the writer can place paragraph-A to the left of paragraph-B to represent that paragraph-A is favorable to paragraph-B. The designer can use the distance between the two objects to reflect the degree of "better-ness." If paragraph-A is located very far from paragraph-B, then the representation would remind the writer afterward that paragraph-A was much better than paragraph-B.

Thus, the use of two-dimensional spatial positioning provides designers with hands-on representations allowing them to represent the current state of mind without verbalizing or formalizing the state. The exact same representation (positioning) may mean very different things to different designers or in different situations.

4. TWO EXAMPLES

This section illustrates how our approach can be implemented in two types of design domains: writing and component-based programming.

4.1. ART

The ART system [13] (Figure 1) supports document construction as a design task allowing users to position segmented text as "elements" in a two-dimensional space. An element is any unit that writers choose to think of as one, such as a phrase, a sentence, a paragraph, or a longer piece of text.

- **System Overview**

The ART system consists of the following three components: *ElementsMap, ElementEditor*, and *DocumentViewer*.

ElementsMap (Figure 1 top right) is a two-dimensional space that graphically displays elements that comprise the document. Each element is represented as an icon. An icon does not show the entire content of the element, but only the first ten percent of the element's text; therefore, the size of the element box corresponds to the size of the actual element (unless the user resizes the box). A user can freely change the position of elements by pointing and dragging icons on *ElementsMap*.

Elements can be created and edited with *ElementsMap* and *ElementEditor* (Figure 1 bottom). The *ElementEditor* component is a text editor for the contents of an element providing editing functions such as cut, copy, paste, and "spin off," which divides one element into two. Selection of an element in *ElementsMap* allows a user to modify the content of the element in the *ElementEditor*. When nothing is selected in *ElementsMap*, a user can edit text in *ElementEditor* and create a new element by positioning the newly

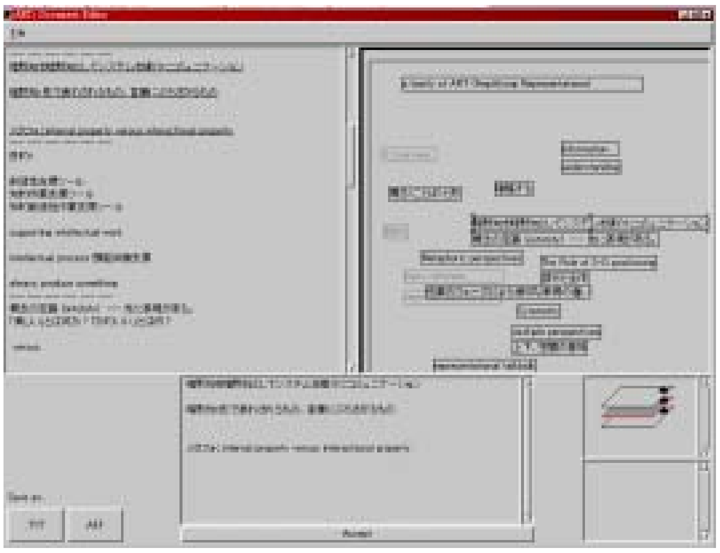


Figure 1: The ART System

created icon in *ElementsMap*. Two or more elements can be merged by selecting multiple elements on *ElementsMap*.

An interesting aspect of the ART system is that the system has a partial understanding of the “meaning” of the positioning of elements in *ElementsMap.* An element’s vertical position in the *ElementsMap* are interpreted as corresponding to its position in the document sequence, and the *DocumentViewer* (Figure 1 top left) component displays the actual content of the document by sequentially scanning the elements displayed in the *ElementsMap* from top to bottom. Thus, a user can freely change the order of elements in the whole document by changing the vertical relationship of elements in *ElementsMap*. Positioning changes and content changes made in *ElementsMap* and *ElementEditor* are automatically reflected in all of the three components.

- **Representations in ART**

The ART system provides “views” to look at both parts and the whole of the document simultaneously. *ElementsMap* provides an overview of the whole in terms of the structure of parts, while *ElementEditor* provides details of a part. *DocumentViewer* displays the context of the part with details of neighborhood elements. The three views are integrated and changes made in one component are dynamically reflected on the other components.

The essential part of the system is the use of *ElementsMap*. In user studies of ART, we found that subjects used a variety of visual properties of two-dimensional positioning as a representation. Some put elements that need further attention in the bottom right corner of the *ElementsMap*. Some subjects made a set of completed elements as the same size and carefully aligned them. One user had two elements overlapping each other with a verbal protocol saying that she felt that they should be related to each other but could not describe how they were related (therefore they were overlapped and not aligned). Another user made some elements particularly larger than others so that it would “call for attention” later in the task.

Interestingly, no subjects complained about the constraint ART imposes on the vertical relationships of elements in *ElementsMap*; the contents of the elements are always concatenated in the order from top to bottom. Subjects used different distances between two vertically positioned elements to represent different types of relations of the two elements. Some subjects placed two elements that were almost completely horizontally aligned but with a slight height difference so that they “looked” horizontally aligned but are not from the system’s point of view.

4.2. RemBoard

RemBoard [14] (Figure 2 (a)) is a tool for remembering classes, methods, notes, or other *things* which may become necessary while programming in Smalltalk. Component-based programming allows programmers to reuse classes and methods from a large class library. In Smalltalk programming, for instance, programmers can exploit a library of more than 900 reusable classes, increasing program quality and productivity.

Finding “necessary” reusable classes from the large library, however, is challenging, making Smalltalk programming hard to learn for novice programmers. It is not easy to understand the whole class hierarchy, and having a hierarchy browser and keyword matching retrieval mechanisms is not enough for novices because it is difficult for them to understand what retrieved classes and methods “really mean” [15]. There is a need for a mechanism that allows Smalltalk programmers to remember intermediate search results - as one cannot decide which classes and objects to use unless one fully understands their detailed behavior. RemBoard provides a representational medium for such programmers to remember what has been retrieved.

- **System Overview**

RemBoard is a system component that is added on top of the VisualWorks (Smalltalk) environment. RemBoard is a free two-dimensional space where programmers can place “objects.” Operations can be performed directly on objects displayed in RemBoard.

Users can put classes on RemBoard by using a window to directly input the class name (Figure 2 (d)) or by copying from a Smalltalk tool, such as an editor or a tool showing search results. Recorded classes are shown as icons on the two-dimensional space with the class name as its label. Users can also place their own annotations on the two-dimensional space. Users can move, delete, or duplicate displayed objects on RemBoard.

The system also allows users to directly perform operations on iconized classes that are necessary for a programming task: for example, to open an editor (Figure 2 (e)) on a class, or to show attached comments by the original programmer (Figure 2 (f)).

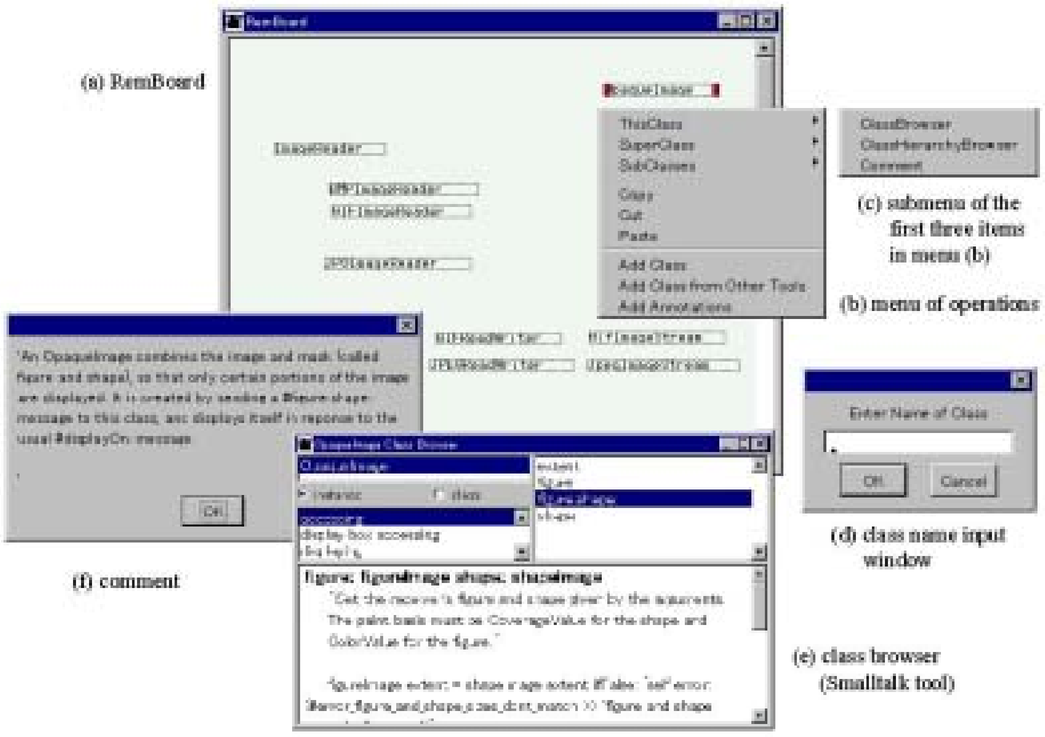
- **Representations in RemBoard**

RemBoard uses a two-dimensional space in addition to conventional textual annotations to allow programmers to express relationships among the recorded objects. It is up to the user to associate meanings to a particular type of spatial positioning, and how to view the positioning.

In the user studies, we have found that subjects used positioning of objects on RemBoard to remember things such as:

(1) classes that were being made or modified;
(2) classes that were found potentially useful for the task;
(3) classes that were thought "related" to the above two types of classes; and
(4) relationships (inheritance, association, being reused, etc) among the above classes.

We have observed that subjects used the positioning in both local and global ways. For



'An OpaqueImage combines the image and mask (called figure and shape), so that only certain portions of the image are displayed. It is created by sending a #figure:shape: message to this class, and displays itself in response to the usual #displayOn: message.

```
figure: figureImage shape: shapeImage
    "Set the receiver's figure and shape given by the arguments.
    The paint basis must be CoverageValue for the shape and
    ColorValue for the figure."

    figureImage extent = shapeImage extent ifFalse: [self error:
#error_figure_and_shape_sizes_dont_match >> 'figure and shape
```

Figure 2: RemBoard and Its Associated Tools

example, one class was positioned below another class to show inheritance (the relationship between `ImageReader` and `BMPImageReader` in Figure 2 (a)). This is a local relationship between just two classes. In another example, a class was placed far from other classes to show that it was not directly related to the other classes but was deemed important to remember (`OpaqueImage` in Figure 2 (a)). This is a global way of representing the relationship of the object in terms of the whole. Interestingly, these two viewpoints were taking place simultaneously within a single two-dimensional space without causing any confusion for the subject. The "meaning" of the positioning depends on what part of the space the programmer is looking at.

5. DISCUSSION

We use positioning of solution-related objects in a two-dimensional space as a representation that serves for problems. The approach has been applied to two design domains: in writing and in component-based programming. Although ART and RemBoard both use a two-dimensional space for designers in externalizing the design situation, we have identified important considerations for the approach due to the difference between the two design domains. This section first compares ART and RemBoard, briefly mentions three-dimensional positioning, and presents design principles of our approach.

5.1. Comparison of the Two Systems

- **Positioning and What It Represents**

We have stressed the role of positioning of design objects in a two-dimensional space as a representation that serves for problems. Designers should be able to use positioning as a representation of their state of mind in ways they like: including close-to, away-from, overlapping, or very large.

However, we may need to consider other purposes that the space may be able to serve. In fact, the usage of space can be considered on a spectrum between two options. One option is to not impose any other meanings at all and use the space as a simple "means"; the other option is to overload some pre-determined semantics on positioning and to use what is being represented with the space as a (partial) representation for the final product.

Because we designed ART so that the top-to-bottom order of elements in *ElementsMap* represents the flow of text in the final document, the two-dimensional space of *ElementsMap* serves not only for problems but also for the final product (document). Positioning of elements in ART can be viewed as direct manipulation of design artifacts. On the other hand, we have not assigned any semantics to positioning in RemBoard thereby the positioning of classes in RemBoard may or may not be related to how the final program will be designed.

This difference comes from the existence of "natural mapping" in each domain. A mapping is natural when *"the properties of the representation match the properties of things being represented"* [10, p.72]. In writing, a document flows from top to bottom on a computer display. There is a natural mapping between the order of sentences and the top-to-bottom positioning on a display. In component-based programming, no such natural mapping has been identified between programming constructs and the use of RemBoard in terms of two-dimensional positioning.

- **Automatic Generation of a Representation that Talks Back to Designers**

In Nakakoji et al. [11], we used a scroll-bar representation in Windows as a good example of representational talkback. The length of a scroll-bar-handle represents a portion of the amount of what is visible to the entire size of the information space. Thus, one can quickly "perceive" how large a displayed document is in terms of the visible space. This works because again there is a natural mapping between the size of the document and the length of the handle; the larger the document, the shorter the handle.

In supporting designers in early phases of a design task, it is critical to identify the "right" balance between what should be automatically done by the system and what should not be done but left with users. We present two examples to illustrate this point.

First, the ART system automatically creates an icon for an element in *ElementsMap* where the size of the element represents ten percent of the element's text. Thus, the size of the icon roughly shows the size of the content of the element. Subjects of the user studies found this functionality useful by saying that "the large icon in the right corner is the element that I have not worked on yet." However, the same subject often changed the size of other elements using the size as a representation – well refined elements were ordered in *ElementsMap* with the same size.

Let us take the labeling of elements as another example. In ART, the system automatically creates an icon by using the initial ten percent of the text content of the element. In RemBoard, the system uses names of classes as labels for the elements. The automatic labeling provided in both systems were welcomed by the subjects in the user studies. If a user is asked to name an element whenever the user creates a new one in *ElementsMap* in ART, it would have disturbed the user's cognitive process. On the other hand, some subjects of the ART study inserted a line or two at the beginning of some of the elements so that those lines appear in *ElementsMap* serving as labels.

What we have found from these episodes is that users appreciate the system's automatic generation of representation as long as the mapping can be considered to be "natural." At the same time, even natural mappings should be modifiable by designers if they want to.

5.2. Three-Dimensional Positioning

So far, we have argued that two-dimensional spatial positioning is a powerful hands-on representation for designers in early stages of a design task. Would it be even more powerful if we use a three-dimensional space?

With the increase of CPU power and memory space, more and more tools provide three-dimensional modeling spaces for a designer. However, interactivity with the space is not necessarily superior to the two-dimensional space. Three-dimensional space is suitable for visualization: for instance, a complex information space with a time axes, or display realistic objects on a computer display. On the other hand, there has not yet been suitable interfaces for a user to interact with objects in the three-dimensional space; it imposes a lot of cognitive load on a user manipulating objects in a three-dimensional space affecting smooth thought processes. For instance, this issue is in some sense reflected in the fact that typical three-dimensional modeling tools offer not only three-dimensional spatial representation but also mechanisms to simultaneously look at an object from three orthogonal viewpoints.

Two-dimensional positioning is suitable for selecting, positioning, moving objects as well as easily identifying how objects are positioned and interacting with the objects. While three-dimensional space is a powerful mechanism to view complex information spaces, it is currently not as easy to interact with as the two-dimensional space.

Interactivity with a three-dimensional space has the following issues:

- It is not intuitive for a user to understand the whole three-dimensional space where objects are positioned.
- It is not easy to select an object if the object is behind another object; the user needs either to change the viewpoint, to rotate the objects, or to move the overlapping object to select the object of interest.
- It is not intuitive to move, reposition, or resize an object using a mouse and keyboard in a three-dimensional space.
- It is not intuitive for a user to place an object in the three-dimensional space; we cannot place objects in the air unless we go to the outer space where there is no gravity. People get used to putting objects on top of other objects in terms of a two-dimensional space because everything they interact within this real world is being affected by gravity.

5.3. Design Principles for Creating Tools for Early Phases of Design

Our goal is to support designers in early phases of a design task by allowing them to externalize their strategic knowledge so that the representations would talk back to the designers helping them understand what the problem is. We argue for designing a computational environment that amplifies representational talkback as a way to support the aspect of design.

We use positioning of design objects in a two-dimensional space as a representation for strategic knowledge. In doing so, we have identified the following design principles:

- to be able to easily create objects in a two dimensional space at any level of granularity as designers like. The presentations (or labels) of objects must be automatically done by the system but designers should be able to overwrite them;
- to be able to easily identify objects in the two-dimensional space;
- to be able to search for objects in terms of the whole design and in terms of other objects;
- to be able to examine details of an object of interest;
- to be able to operate on objects displayed in the two dimensional space in a direct manipulation style;
- to use a mapping between domain constructs and physical properties of two-dimensional space to automatically process displayed objects if and only if the mapping is “natural,” for example, first is at the top and last is at the bottom. Designers must be allowed to overwrite these mappings when necessary.

6. RELATED WORK

This section discusses related work from two perspectives: research that focuses on representations serving for problems rather than solutions, and research that uses two-dimensional positioning as a representational medium.

6.1. Representations for Strategic Knowledge

A line of research in design rationale [16] has focused on representation for strategic knowledge. Design rationale is typically a textual description of what alternatives should be taken and arguments that support or negate each alternative. Although such design rationale mechanisms provide powerful cognitive representations for designers to understand the history of design evolution and how to proceed with the design task, they aim at a larger scale in terms of time. Most design rationale system allows users to record (externalize) rationale after the design session finishes. It is also limited to textual representation.

Our focus is more on on-time help for reflection. We use perceptual representations that help designers. We view our approach to be complementary to the design rationale research rather than as a replacement.

Tools that allow free-hand drawing, such as the CocktailNapkin system [17], share the same goal with our approach. While our approach uses two-dimensional positioning as a representation for a "designer's state-of-mind," such tools use free-hand drawing as a representation. A sketch-based interface can be viewed as amplifying representational talkback. Users can externalize various situations without having to verbalize or formulate sentences to express such situations. The meaning associated with the representation is "obvious" to the user who made the sketches - the representation talks back to the user.

6.2. The Use of Two-Dimensional Space

Various research on using space for representation has been done. Shipman et al. [18] found that people use the visual and spatial characteristics of graphical layouts to express relationships between icons and other visual symbols. Fentem et al. [19] argues that spatial positioning serves as a shared language among a group of people working together. Other work has focused on inferring the user's underlying intent of a positioning based on methods such as statistical analysis [20] and genetic algorithms [21].

We focus on the use of a representation produced by a user using space. The representation can be considered as an intermediate status of some task. The representation helps the user in their task, while using it does not disturb their cognitive processes, i.e. it does not detract from what they want to do.

Some research offers a two-dimensional space to represent a user's intention but the meaning of axes are pre-assigned by the system. The SearchSpace system [22], for instance, uses a two-dimensional space to represent a query for document search. The vertical axis of the space is used to represent the degree of importance of positioned keywords and the horizontal axis is used to represent the degree of spelling ambiguity of positioned keywords. A user can position multiple keywords in the space with positioning as the representation of the properties of the keywords.

7. CONCLUSION

This paper presented our approach to support early phases of a design task by providing hands-on representations that better allow designers to externalize his/her thoughts and ideas without forcing him/her to verbalize or formalize them; therefore interaction with the medium does not interfere with the designer's cognitive processes. Our focus is not on representations that serve for final artifacts but on ones that serve for problems. We use two-dimensional spatial positioning of design objects as a perceptual representation that allows designers to express their state of mind.

While passive materials and artifacts cannot speak for themselves, computational materials can. Although this fundamental difference provides great leverage in improving the way designers work and learn, it can also be a pitfall by imposing representations that may not necessarily be "right" for the task. What is important is to give designers representational media that allow them to externalize what they want to express in ways they like. Our approach is a step forward to let designers deal with implicit/tacit knowledge on a computer system. Meanings can be extracted from a representation only by the designer; the system remains as a medium – but a useful one.

ACKNOWLEDGEMENTS

We would like to thank Linda Candy for valuable suggestions on the previous version of this paper. We thank Mark Gross, Brent Reeves, and Atsushi Aoki for their comments on the framework of our approach. This research is partially supported by NEDO (New Energy and Industrial Technology Development Organization), MITI, Japan.

REFERENCES

[1] Simon, H A, The Sciences of the Artificial (Third ed.) (The MIT Press, Cambridge, MA. 1996).

[2] Rittel, H, and Webber, M M, Planning Problems are Wicked Problems, in: N. Cross, ed., Developments in Design Methodology (John Wiley & Sons, NY, 1984) 135-144.

[3] Snodgrass, A and Coyne, R, Is Designing Hermeneutical? (Dept of Architectural and Design Science, University of Sydney, 1990).

[4] Schoen, D A, The Reflective Practitioner: How Professionals Think in Action (Basic Books, NY, 1983).

[5] Bruner, J, The Culture of Education (Harvard University Press, Cambridge, MA, 1996).

[6] Arnheim, R, Visual Thinking (University of California Press, CA, 1969).

[7] Zhang, J, The Nature of External Representations in Problem Solving, in: Cognitive Science, 21(2), (1997) 179-217.

[8] Lawson, B, Design in Mind (Architectural Press, MA, 1994).

[9] Lawson, B, Designing with Drawings, in: How Designers Think: The Design Process Demystified (Architectural Press, MA, Chapter 14, 1997) 241-259.

[10] Norman, D A, Things That Make Us Smart (Addison-Wesley Pub. Co., MA 1993).

[11] Nakakoji, K, Yamamoto, Y, Suzuki, T, Takada, S, and Gross, M, Beyond Critiquing: Using Representational Talkback to Elicit Design Intention, in: Knowledge-Based Systems Journal, 11(7-8), (1998) 457-468.

[12] Reisberg, D, External Representations and the Advantages of Externalizing One's Thoughts, in: Proc. of the 9th Annual Conf. of the Cognitive Science Society, (Cognitive Science Society, 1987).

[13] Yamamoto, Y, Takada, S, and Nakakoji, K, Representational Talkback: An Approach to Support Writing as Design, in: Proc. of 3rd Asia Pacific Computer Human Interaction Conf. (Kanagawa, Japan, IEEE Computer Society, 1998) 125-131.

[14] Takada, S, Nakakoji, K, and Torii, K, Using 2D Space for Understanding What Search Options We Have Taken in Exploring a Class Library, Technical Report ccc-98-9. (Cognitive Science Lab, NAIST, 1998).

[15] Takada, S, Otsuka, Y, Nakakoji, K, and Torii, K, Strategies for Seeking Reusable Components in Smalltalk, in: Proc. of the 5th International Conf. on Software Reuse (ICSR5) (Victoria, CA, IEEE Computer Society, 1998) 66-74.

[16] Moran, T P, and Carroll, J M, eds., Design Rationale: Concepts, Techniques, and Use (Lawrence Erlbaum Associates, Inc, NJ, 1996).

[17] Do, E Y-L, and Gross, M D, Inferring Design Intentions from Sketches: An Investigation of Freehand Drawing Conventions in Design, in: Proc. of the 2nd Conf. on Computer Aided Architectural Design Research in Asia (CAADRIA'97) (Taipei, Taiwan. Hu's Publishing, 1997) 217-227.

[18] Shipman, F M, Marshall, C C, and Moran, T P, Finding and Using Implicit Structure in Human-Organized Spatial Layouts of Information, Human Factors in Computing Systems (CHI '95) (Denver, CO, 1995) 346-353.

[19] Fentem, A, Dumas, C, and McDonnell, J, Evolving Spatial Representations to Support Innovation and the Communication of Strategic Knowledge, in: Knowledge-Based Systems Journal, 11(7-8) (1998) 417-428.

[20] Sugimoto, M, Hori, K, and Ohsuga, S, A System for Visualizing Viewpoints and its Application to Intelligent Activity Support, IEEE Trans. on Systems, Man, and Cybernetics, 28C(1) (1998) 124-136.

[21] Igarashi, T, Matsuoka, S, and Masui, T, Adaptive Recognition of Implicit Structures in Human-Organized Layouts, in: Proc. of the 11th IEEE Symp. on Visual Languages (1995) 258-266.

[22] Tsutsumi, F, and Shinohara, Y, Search Space: Document Retrieval by 2-D Positioning Keyword Query, Computer Software, 15(4) (1998) 2-15 (in Japanese).