COSINE: A Sketch-Based Interactive Environment for Cooperative Design

Wei Jiang, Zhengxing Sun, Wentao Zheng
State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093
Department of Computer Science and Technology, Nanjing University, Nanjing 210093
jiangwei@graphics.nju.edu.cn; szx@nju.edu.cn

Abstract

Sketch plays an important role in the brainstorming of ideas. During conceptual design stage, designers use generally 2D graphical tools such as pencil and paper to draw a lot of sketchy documents for conveying ideas and communicating with other designers. Furthermore, conceptual design for complicated system involves a group of geographically distributed participants. However, most of current graphical design tools do not support such kind of collaborative sketching. This paper introduces a novel sketch-based interactive environment (COSINE) for cooperative design. Three key problems are addressed in this research: (1) how to represent and store sketch document for sketch-based cooperative design; (2) how does the system infer the designers’ intention and help them to finish the design process; (3) how to implement the communication between client and server. Solutions to these problems are proposed in this paper. A prototype system for cooperative UML class diagram design based on COSINE is also outlined.

Keywords: Computer Supported Cooperative Work (CSCW), Computer Supported Conceptual Design (CSCD), Sketch-based Interface, COSINE.

1. Introduction

During the early stage of conceptual design, the ideas are unfinished and uncertain. Designers do not need to consider too much about the detail. What they need is to find the possible solutions as soon as possible [4]. During this stage, sketches are produced to communicate with self or others. People who cooperate in groups often need to meet together to do some brainstorming, or to convey, adjust or hit upon (new) ideas. During brainstorming sessions, designers generate ideas and turn them into quick sketches with traditional 2D tools such as pencil and paper.

Sketch plays an important role in the brainstorming of ideas. Sketches help us convey ideas and guide our thinking process both by aiding short-term memory and by helping to make abstract problems more concrete. Most importantly, sketching is a natural input modality of increasing interest [7].

However, conventional CAD tools usually require complete, concrete and precise definitions of the geometry of the geometry information, which is only available at the end of the design process [9]. Therefore, they cannot support the conceptual design process in which a great many of sketch documents are generated.

Complicated conceptual design usually involves geographically distributed participants. Currently, this kind of design is based on conventional synchronized media like audio and video, and some other asynchronous tools such as E-mail [8]. However, they cannot support the participants to simultaneously create, modify, or view a shared design model in the form of sketch with other members of the design team.

Our research investigates sketch as an interface in distributed cooperative design. Designers sketch out their idea in their own location and transfer the design to a cooperative design server through network. Therefore, distributed designers can exchange ideas in the form of sketch and work together in a virtual design environment. We have developed an intelligent sketch-based interactive environment for cooperative design, which is named COSINE (Cooperative Sketch-based Interactive Environment).

While sketch-based interface is used in CSCD (Computer Supported Cooperative Design), some problems arise:

1) How to represent and store the sketch documents, which are produced by the design team?
2) How does the system know the designer’s intention and help them to complete the design process without any interruptions?
3) How to implement the communication between client and server?

For the first problem, we proposed a XML-based sketch document format, which is fit for represent and store sketch data. For the second problem, we involve a sketch recognizer in COSINE, which uses a primitive-based sketch recognition algorithm to recognize the designer’s input and gives useful guidance to the designer. For the third problem, we implement hybrid system architecture and involve event-based model to facilitate the communication between client and server.

The rest of the paper is organized as follows. First, we describe some related works on sketch-based user interface both in single-user applications and in CSCD. Next, we address the COSINE framework and discuss...
three key issues mentioned above in detail. Then we describe an experimental prototype system for cooperative UML class diagram design based on COSINE. Finally, we make a conclusion and identify the future works.

2. Related Work

Sketch-based interface draws a lot of attention in the past years. A number of academic prototypes and commercial products have been developed. Alvarado [1][2] proposed ASSIST which uses sketch-based interface to support mechanism design. Hong and Landay [6] developed SATIN, which is a toolkit to support the creation of pen-based applications. Landay and Myers [7] developed an interactive sketching tool called SILK, which helps designers to sketch out a user interface.

Most of current researches on sketch-based user interface focus on sketch recognition and most of them are designed for single-user use.

To date, a few works have involved freehand sketch in cooperative design. NetSketch [8] is a sketch-based application, which supports cooperative conceptual design. However, there is a limitation of the graphical shapes that can be created in sketch. NetSketch uses a peer-to-peer network topology and cannot guarantee the consistency among all users. Zhe et al. [12] describe a sketch-based interface for collaborative 3D conceptual design. Using the Web-based system, designers can rapidly and easily create, and edit a 3D design geometric model. Their work focuses on 3D geometric modeling and simulation.

Different research works focus on their own requirements. Research in this field is still emerging and there are no mature models and standards at hand. Most of current researches just support a few simple freehand shapes. Therefore, only a few design works can be done by sketch. Actually, the cooperative design tasks in real-world require broader capabilities than are provided by most of current available applications.

3. Cooperative Sketch-based Interactive Environment - COSINE

This section presents an overview of the architecture and some key issues of COSINE.

3.1. COSINE Architecture

COSINE is based on Client/Server model. There are mainly three kinds of architectural alternatives for constructing CSCW system: centralized approach, replicated approach and hybrid approach. We use a hybrid approach which benefits from both centralized and replicated approach. Each participant retains a replicated version of the whole design document. Therefore the network traffic is reduced and the designer can work effectively without be interrupted by network communication. The participant processes use a central control process in COSINE server side for synchronization and for mediating users’ potentially conflicting requests.

Figure 1 shows the COSINE architecture. During the cooperative design session, designers connect to the server and start a new drawing session. Then they begin to draw sketches on their own user interfaces as they do on physical paper. The COSINE server controls the design group to work smoothly and effectively.

As we can see in the figure, COSINE client is composed of five modules: UI (user interface), sketch recognizer, command generator, data manager and display controller. UI contains a draw panel for the designer to draw and view sketches. Sketch recognizer recognizes the designer’s input. It is a fast and simple approach to identify multi-stroke geometric shapes which will be described next. After the inputting sketch is recognized, the command generator generates a proper command according to the recognition result and sends it to the COSINE server side. The data manager module retains the latest local design data (maybe different from the latest sketch data in the server side) and provides the basic functions on the sketch data (such as add stroke). The server responds the user’s command and returns the result to the client side. Then data manager updates local sketch data. Finally, the display controller displays the latest modification according to the results. The display controller also supports different view for different designer, which is very helpful for cooperative design work since different designers may in charge of different parts.

![Figure 1. COSINE Architecture](image-url)
side is first dispatched to conflict detector. The conflict detector first parses the command and finds out the objects to be operated and the designer’s intention. Then it looks up current designer’s privilege and the state of the objects and decides whether the command is legal. If the command is an illegal one, which means it may lead confliction; the conflict manager module will process the command and return results to result generator module. In the other way, if the client user’s command is a legal one, the command processor will execute it (add new objects or make some modifications use data manager module). The data manager in server side is similar to the one in the client side. However, it retains the latest design data. Finally, the result generator generates the result and sends the result to proper client side(s).

COSINE is implemented in Java, a network programming language, which is platform-independent and fit for distributed cooperative work. Moreover, we can easily use Java Applet to implement a Web-based CSCD system.

3.2. Sketch Document

When using sketch-based interface in CSCD systems as we mentioned before, a fundamental problem is how to represent sketch data. Recent years, ink-based document format draws lots of attention in both academics and industries. Several standards were proposed, such as JOT [13], UNIPEN [5] and InkML [3]. Although these standards are very capable for storing ink-based data, they cannot be applied into CSCD system directly. Conceptual cooperative design cares about both the low-level raw sketch data (such as points in the graphical shapes) and the high-level semantics of sketch (such as what the sketch represents and its components). As a result, we propose a new sketch-based document format called Sketch Document, which will be described below.

At first, we will introduce some important concepts for representing sketch data.

Sample Point is a point captured by input device. It is parameterized as \((x, y, p, t)\), \(x\) and \(y\) refers to the coordinates of point, \(p\) is for pen pressure, and \(t\) is for timestamp.

Stroke is a set of sample points from pen-down to pen-up.

Primitive is a predefined geometric shape (line segment, arc segment and ellipse) that is extracted from stroke by stroke segmentation.

Sketch Component is a domain-related visual object (such as a resistance in a circuit diagram, or a class in a UML diagram). It contains certain semantics, which is meaningful to human participants.

Sketch Document is the whole design document and consists of a set of sketch components. For example, a sketch document can be UML class diagram in the form of sketch, which consists of a set of classes and arrowhead lines.

Figure 2 shows the data structures mentioned above and the relationships between them.

The sketch document is XML-based. Sketch document is organized in a layered structure. From the bottom up, the layers are: raw sketch data, semi-raw sketch data, recognized sketch components, sketch documents semantics. Raw sketch data refers to sample points in the stroke captured by the input equipment. Semi-raw sketch data refers to the primitive shapes extracted from the stroke. Recognized sketch component refers to finished sketch component. Finally, sketch semantics refers to some high-level information of the sketch document, such as sketch category, author name and create date. Each of the component-elements and stroke-elements in the XML file contains an attribute called “author”, which refers to the creator of that element and is used for cooperative control.

Figure 3-1. A simple sketch sample
3.3 Sketch Recognition

COSINE contains a sketch recognizer module. We have developed some sketch recognition algorithms in our previous work [10][11]. We use a primitive-based algorithm in COSINE, where the inputting strokes are first decomposed into basic geometric primitives (ellipses, line and arc segments) and then assembled into a graphical structure that encodes both the intrinsic attributes of the primitives and their relationships. Sketching recognition is accordingly formulated as a template-matching problem.

Figure 4 gives a sketch of the recognizer. First of all, the primitives of sketch are recognized in the primitive recognition stage. Then the feature vector of a sketch is extracted in order to construct vector-model for the sketch. In our research, features are refined into two categories: edge feature and spatial relationship feature. The edge feature represents statistical primitive information of a sketch, such as the number of line segments. The relationships between two primitives (such as parallelism, intersection and tangency) are also considered in our approach. We combine the edge feature and spatial relation feature together into a multi-dimensional feature vector. During the template matching stage, the recognizer calculates the similarity between a given sketch and each standard shapes in a domain-specific shape library which results in an objects set, based on the vector-based model. We employ Euclidean distance for matching. Finally, the recognizer returns the recognition result to user.

3.4. Event-based model for C/S communication

In this section, we will discuss the communication between COSINE client side and server side.

As we mentioned before, each client has a copy of sketch document data. After one designer makes a modification to the sketch document, he only sends the corresponding data (including the action type such as delete and graphical object to be modified such as one single sketch component).

We involve event-based model in COSINE. Each of the commands generated by command generator module in client side is regard to be a COSINE Event. There are two types of COSINE Events. One is for control, such as login, logout. The other is for sketch data processing such as add a simple stroke to the sketch document, delete a stroke in the document and so on.

We use a unified XML-based format for both of the two types of events. All of the XML-formatted events have a root element named “CosineEvent”, an “Author” element to indicate designer ID, a “Type” element to indicate event type, and “Arguments” element to list parameters corresponding to current event. Figure 5 and 6 show two COSINE Event XML format.

Figure 5 is for login action. It encapsulates the password, the user IP address and network port in the XML-formatted file.
Figure 6 is for add stroke function. The figure shows that the client side sends only the new added stroke data and the sketch component ID to be modified to the server.

4. A Prototype System based on COSINE

In order to evaluate our proposed framework, we have implemented a simple prototype system based on COSINE that supports cooperative UML class diagram design.

UML diagrams are widely used in software design. Software developers draw UML diagrams to aid in the brainstorming of ideas, visualizing programming organizations, and understanding of requirements. Let us consider a scenario in customized software design. Software developers and customers need to discuss about the system requirements in the early stage of the development. They need to meet together and draw some UML diagrams. However, in many cases, the software developers and the customers are at physically separate locations. The benefits of sketch cannot be fully exploited with the use of pencil and paper, and these kinds of brainstorming sessions seem impracticable.

Figure 7 shows a simple UML class diagram designing process involved by two designers (User A and User B). Each designer draw sketch in his own UI, and can see the other’s input via COSINE server. The left figures demonstrate the designers’ drawing processes. The right ones show the corresponding results.

In Figure 7-1, User A began to draw sketches in his own UI. First he drew a class. After he had finished the class element, the recognizer recognized the sketch as a class. The client side sent an “Add Component” event to the server. The server accepted the addition to the main sketch document and sent the modification information to both of the two clients. Then both of the two users could see the new added standard class element (which is shown in Figure 7-2).

As the design session was going on, in Figure 7-3, User B started to draw an interface element. In Figure 7-5, User A drew an implementation line between the existed class and interface elements. In Figure 7-7 and 7-9, User B drew a sub-class extended from the first class element. Finally, the two users finished the class diagram design (Figure 7-10).

Figure 7. An example of a design process (From 7-1 to 7-10)

Figure 8. Two different designers’ views of the same design

As we mentioned in section 3, the client side contains a display controller module, with which each designer can have their own view of the same document. This will make the designers feel more comfortable when they can change the view of work in their own way. Figure 8 gives an example. In Figure 8, the two designers both have their...
own views of the document, which are different from each other. However, actually, they have the same sketch document as the one in the server.

As we can see from the class diagram design process, COSINE is a valuable tool for cooperative design in practice. Now, by using COSINE, distributed designers can work together effectively. They can obtain the benefits of freehand sketching in the cooperative design, which are seldom supported by current design tools. Although the case we used is very simple, we can easily extend the design tool to support more complicated cases by extending the shape library in the recognizer module.

Compared with other cooperative design tools such as NetSketch, which support sketch interaction, COSINE’s hybrid architecture and event-based communication mode are more suitable for sketch-based cooperative design and can easily support multi-designer’s requirement. The hierarchical sketch document model can ease the difficulty of representing and storing designers’ ideas in the form of sketch. Furthermore, the recognition algorithm used in COSINE is more complicated. Therefore, COSINE supports a larger shape library and satisfies cases with more complex requirements.

5. Conclusion and future work

Sketching is a nature way for visual thinking and communication. A great many of sketch documents are generated in conceptual design area. However, such kind of interaction seems not fit for cooperative design. Since the participants are distributed in location and they cannot meet together and use physical pencil and paper to discuss the design.

This paper introduces sketch-based interface in CSCD. We considered some fundamental problems that will arise in sketch-based cooperative design and developed a sketch-based cooperative interactive environment: COSINE.

In this paper, we proposed a novel sketch document format based-on XML that is capable for storing, modifying sketch-based document. We used a sketch recognizer to recognize the input sketch so that the system can infer designers’ intention and help them to complete the design work. We proposed an event-action based model to facilitate the communication between the COSINE client and server. We also implemented a prototype system based-on COSINE for cooperative UML diagram design which produced satisfactory results.

However, some problems still have not been solved yet, such as conflicts detection in the design process. We also need to refine the human-computer interaction mode to provide designer a more natural way of design. These require our further work.

Acknowledgement

This paper is supported by the grants from the National Natural Science Foundation of China [Project No. 69903006 and 60373065] and the Program for New Century Excellent Talents in University (Project No.NCET-04-0460).

References