ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.18, Supplement, December 2007, pp.35−44

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

基于草图的面向概念设计的协同图表绘制工具*

蒋 维[1,2], 孙正兴[1,2+]

[1](计算机软件新技术国家重点实验室(南京大学),江苏 南京 210093)

[2](南京大学 计算机科学与技术系,江苏 南京 210093)

A Sketch-Based Cooperative Diagramming Tool for Conceptual Design

JIANG Wei[1,2], SUN Zheng-Xing[1,2+]

[1](State Key Laboratory for Novel Software Technology (Nanjing University), Nanjing 210093, China)

[2](Department of Computer Science and Technology, Nanjing University, Nanjing 210093, China)

+ Corresponding author: Phn: +86-25-83686099, Fax: +86-25-83686099, E-mail: szx@nju.edu.cn, http://cs.nju.edu.cn/szx/

Jiang W, Sun ZX. A sketch-based cooperative diagramming tool for conceptual design. *Journal of Software*, 2007,18(Suppl.):35−44. http://www.jos.org.cn/1000-9825/18/s35.htm

Abstract: Diagrams are used for representing design concepts and constructing function structure in the stage of conceptual design. This paper considers sketches as the language of communication in cooperative conceptual design and presents a tool that enables distributed participants to cooperatively construct diagrams in the form of freehand sketches. The tool is designed and implemented as a multi-agent system, which contains a set of agents to fulfill the requirements, including sketch-based interaction, sketch recognition, communication and coordination. The operating strategies of these agents are also addressed, including communication between agents, sketch recognition algorithm, and semantic consistency maintenance. The proposed approach is implemented in cooperative flowchart drawing.

Key words: computer supported cooperative work; conceptual design; cooperative diagramming; sketch-based user interface

摘 要: 概念设计过程使用了多种图表来表示设计概念以及构建功能结构.将手绘草图作为设计者之间的一种通信语言引入到异地同步协同概念设计过程,提出了基于手绘草图的协同图表绘制多 agent 系统,包含草图交互、草图识别、通信、协调等多个 agent.讨论了各 agent 之间的通信机制以及主要 agent 的实现策略,包括草图识别以及语义一致性维护策略.将系统应用到协同流程图绘制领域,运行结果表明,该工具能够改善人机交互模式,促进多用户之间的交流和协作.

关键词: 计算机支持的协同工作;概念设计;协同图表绘制;基于草图的用户界面

1 Introduction

The objective of conceptual design is to generate possible solutions quickly and to choose an optimal one

* Supported by the National Natural Science Foundation of China under Grant Nos.69903006, 60373065, 60721002 (国家自然科学基金); the Program for New Century Excellent Talents in University of the Ministry of Education of China under Grant No.NCET-04-0460 (国家教育部新世纪优秀人才支持计划); the National High-Tech Research and Development Program of China under Grant No.2007AA01Z334 (国家高技术研究发展计划(863))

Received 2007-04-30; Accepted 2007-11-23

according to functional requirements. Recently, design tasks are becoming more and more complicated and usually involve geographically distributed participants. The design team usually starts with clarified requirement specifications, and establishes the functional structures. Designers search for appropriate working principles and their combination, evaluate the concept according to the technical and economic specifications, and make the final decision[1]. During this stage, the design concept is usually difficult to capture, visualize, and communicate among the teammates. In real-world design sessions, a number of diagrams are used for representing the design concept, constructing the function structure, and communicating ideas among the designers, e.g. function block diagram, flow chart and logic diagram.

Diagrams are graphical representations of a design, concept, or object[2]. People draw diagrams to communicate concepts more clearly with each other. Many computer aided drawing tools have been developed to assist people to produce diagrams. However, most of them are implemented in drag-and-drop drawing style. To the user's disappointment, this kind of interaction mode interferes in user's thinking and discussing processes. One key problem of these tools is that they do not provide users the ability to construct diagrams in the form of freehand sketch, which is a designer's most natural working style in conceptual design stage.

Sketch-based interface draws a lot of attention in the past decades[3−5]. In our previous work, a sketch-based graphical input system, named MagicSketch[6], has been developed. In this study, we consider freehand sketches as the communication language in cooperative conceptual design, especially in remote collaboration circumstances. Designers draw freehand sketches to construct diagrams to communicate with themselves and other designers. We have developed a multi-agent based cooperative diagram construction tool, which will be described in detail in this paper. The system contains a set of agents to fulfill the different requirements, such as sketch-based interaction, sketch recognition, communication and coordination.

Researchers in computer supported cooperative work (CSCW) have investigated collaborative drawing tools for many years. Some of the early work have proposed electronic whiteboard system, which just provides group members a shared palette to record and display the original graphical notations in the form of sketches. In other words, they cannot produce formal diagrams. Recently, sketch-based interaction is used more and more in the Ubiquitous Computing environment. Most of the current systems are implemented for co-located cooperative work while the participants are in the same place, such as Roomware[7] and MCSKetcher[8]. To date, there also exist a few sketch-based tools which support distributed cooperative design[9−11]. However, concurrency control and consistency maintenance mechanisms are not universally implemented in these systems.

The rest of the paper is organized as follows. In section 2, we will describe the system architecture. In section 3, we will address the sketch recognition algorithm. Semantic consistency maintenance mechanisms are discussed in section 4. Section 5 shows the implementation of the tool in cooperative flowchart drawing and an experimental session. Finally, we make a conclusion in section 6.

2 System Overview

The system is designed as a multi-agent system which contains a group of agents to perform the complex task. There are mainly two agents in our proposed system: diagramming agent and coordination agent. Diagramming agent is running in each user's local machine and represents the user in a cooperative session. Coordination agent, usually running in a separated server, acts as a mediator of diagramming agents. During a cooperative session, each diagramming agent communicates with the coordination agent in order to construct diagrams cooperatively (as is shown in Fig.1).

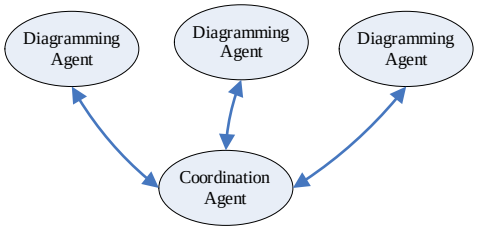

Fig.1 Two types of agents

The diagramming agent contains four sub-agents (as is shown in Fig.2): Interaction agent, sketch recognition agent, communication agent, and data management agent.

Interaction agent senses and identifies current user's input, and generates a proper event according to the user's input type. Then it calls communication agent to send the event to the coordination agent. Interaction agent usually calls sketch recognition agent to recognize sketches drawn by the user. Sketch recognition agent uses some kind of recognition algorithm to identify multi-stroke geometric shapes. The data management agent stores the latest local data and provides a few basic sketch operations. After the coordination agent responds the event, the data management agent will update the local data. Finally, the interaction agent displays the latest modification to the user.

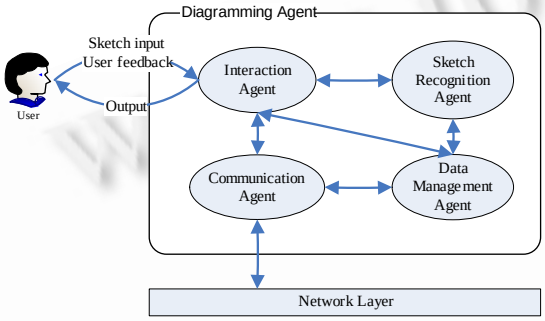

Fig.2 Diagramming agent architecture

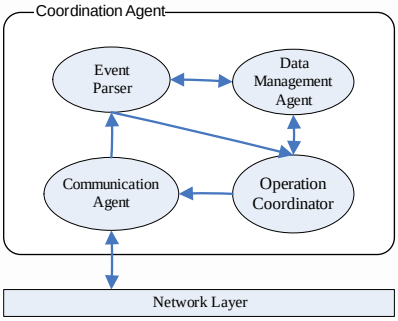

Fig.3 Coordination agent architecture

The coordination agent is also composed of four parts (as is shown in Fig.3): Communication agent, event parser, data management agent, and operation coordinator.

The communication sub-agent receives events from diagramming agent, and then it dispatches the events to the event parser. The operation coordinator, core of the coordination agent, will coordinate the cooperative operations after the received events are parsed. The main coordination mechanism of operation coordinator is semantic consistency maintenance, which will be discussed in detail in section 4. The data management agent is similar to the one in the diagramming agent, but it holds the global data.

The communication mode between diagramming agent and coordination agent is described below.

At first, we define a layered sketch-based document format, including the following data structures[12]. A *Sample Point* represents a point captured by the input device. A *Stroke* is consisted of a set of sample points captured from pen-down to pen-up events. A *Primitive* is a predefined geometric shape (such as line segment, arc segment and ellipse), which is extracted from strokes by segmentation. A *Sketch Component* is a domain-related visual object (such as a class symbol in a UML diagram), which contains certain semantics. *Sketch Document* represents the whole document, which contains a number of sketch components.

As we mentioned before, each diagramming agent retains a copy of the whole sketch data. Therefore, the local agent only needs to send the corresponding data (including the action type and graphical objects to be modified) to

the coordination agent after the user makes a modification to the sketch document. The system encapsulates the modification arguments into an event, which is formatted into an XML string and sent to the coordination agent.

The event sent by a diagramming agent is parsed into a sketch document edit command (SDEC) batch in the coordination agent, which will be sent to the related diagramming agents as the response. SDEC is a pre-defined operation that the system can operate on sketch data. Currently, the system supports four levels of basic operations as follows: (1) operations on sketch documents (create, delete, store or obtain a sketch document); (2) operations on sketch components (add, recognize, lock/unlock or delete a sketch component); (3) operations on primitive shapes (add or delete a primitive); (4) operations on strokes (add or delete a stroke).

Several operations (SDEC) can be combined into a SDEC batch and executed at one time. SDEC batch is useful in cooperative work while participants submit the operations at times. Moreover, one simple event may result in more than one SDEC. For example, while a user draws a new stroke on a blank location that does not belong to any existing sketch components, the system will generate a SDEC batch, which contains two SDECs (one is for adding a new sketch component, the other is for adding the new stroke to the sketch component).

3 Sketch Recognition Agent And Recognition Algorithm

Freehand sketches are ambiguous and informal which need to be transformed into definite and formal graphical results (formal diagrams). One challenge of sketch-based diagramming tools is the implement of a sketch recognizer in order to parses the users' input into domain-specific and semantic symbols.

In this study, we consider the main objective of sketch recognition is to recognize sketch components. Multiple users usually concern more than one sketch component at the same time. Therefore, there will be more than one active sketch components simultaneously. While one user adds a new stroke to the sketch document, the system need to determine which sketch component the stroke should be added into. We assume that the strokes in a sketch component are close to each other in spatial relation, and present an automated stroke grouping algorithm.

Each sketch component has an attribute named *bounding box*, which is the minimal rectangle containing the component and represents the location and the size of the component. New strokes are added to an existing sketch component, if the sample points are totally or partially (determined by a fixed threshold) contained in the bounding box of the sketch component. Then the bounding box will be recalculated. If the newly added stroke does not belong to any existing sketch components, the system will create a new sketch component, which contains the current stroke.

We have proposed a set of sketch recognition algorithms for different requirements in the previous studies, especially for complicated sketch recognition (based on Spatial Relationship Graph[13], Sketch Parameterization[14], Dynamic Programming[15] and Dynamic user modeling[16]) and user adaptation (based on Support Vector Machine[17], Hidden Markov Model[18] and Bayesian Networks[19]). We use a similarity-based recognition algorithm in this study. The main idea is that the system calculates the similarities between the input sketch component and each standard shape in the library, and returns the shape category with the highest similarity. The system can also return a fixed number of candidate shapes (with the similarities from high to low) to the user for feedback.

4 Coordination Agent and Semantic Consistency Maintenance

4.1 Sketch semantics

The main motivation of coordination agent is semantic consistency maintenance. We divide sketch semantics into three levels:

1) Graphical shape semantics: Contains domain-independent geometric information, which is extracted by sketch preprocessing and recognition.

2) Object semantics: Maps a sketch component into a specific object in a given domain. It is domain, user, and sketch context dependent.

3) Relationship semantics: Represents semantics relationships between objects (object semantics has already obtained). It usually contains spatial and connection relationships between objects.

The object semantics and relationship semantics are more complex and domain-related, which can be changed by operations on sketch components and should be coordinated in a cooperative diagramming session. We have proposed concurrency control mechanisms on sketch component to fulfill the object semantics maintenance and relationship maintenance strategies to satisfy the relationship semantics requirements.

4.2 Concurrency control on sketch component

Sketch-Based cooperative diagramming tool is a particular class of cooperative graphical editing systems. There are mainly two types of concurrency control strategies in cooperative graphical editing systems. One is optimistic control, which means multi-users can perform write-operation on a shared object simultaneously. Multi-Versioning approach is often used[21], in which a server is used to maintain the global data and the last updating user is asked to merge the data. The merge action is usually time consuming and may cause conflicts. The other is pessimistic control, which means only one user can perform write-operations on a shared graphical object at one time and the other uses can only “read” that object, such as in CoDiagram[22] and GroupDraw[23].

There are two challenges in implementing concurrency control in sketch-based cooperative diagramming systems. First, the most important reason of using sketch-based interaction (or pen-based interaction) is that this kind of interaction mode is natural and quick. Therefore, people will be frustrated if they need to wait a long time to obtain the right to modify an object, which is locked by a careless user. Second, freehand sketches are usually ambiguous, informal, and incomplete. Most of the existing concurrency control approaches in cooperative graphical editing systems are based on precise and complete shapes.

We find that a user usually continues to work on the sketch component last updated by him in a short time, until the component is complete, in the area of diagramming. We have proposed an automated lock mechanism based on domain knowledge. The coordination agent determines whether a sketch component is complete according to some predefined domain constraints and locks (unlocks) the object automatically.

In our approach, domain knowledge is used to define a set of domain constraints to determine whether a sketch component is complete. We have defined two types of constraints.

1) Inner-Component Geometry Constraints

They define the basic primitive shapes and the inner-relationships of a single sketch component in the given domain. They are determined by stroke grouping and sketch recognition in runtime.

2) Outer-Component Relationship Constraints

Spatial and semantic relationships between sketch components in the related domain are defined. In most 2D diagramming domain, they may be the connective constraints between components and determined by the connection lines.

The completeness of a sketch component is defined by the following three states.

Uncertain, indicates that the sketch component is unrecognized and we do not know the category of the component.

Unfinished, means the sketch component is recognized but do not completely satisfy the second type constraints.

Finished, shows that the sketch component satisfies all of the constraints.

The changing of states of a sketch component is shown in Fig.4 and will be described below.

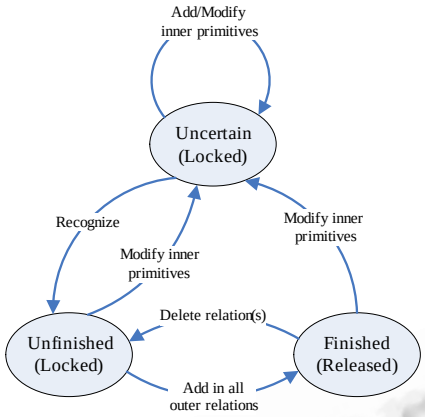

Fig.4 States changing of a sketch component

While a sketch component is newly created (by adding a single stroke to a blank area in the canvas), the state of the component is set to uncertain. The coordination agent locks the component automatically and assigns the creator as the owner of the component. Then the user continues to modify the component or add outer relations to the component. Once the sketch component is recognized, the state is changed to unfinished and is still locked.

The coordination agent automatically checks whether the unfinished components match the second type constraints. Once an unfinished component matches all of the constraints, the state will be set to be finished and the lock is released.

If the inner primitives of an unfinished or a finished component are modified successfully, the component will become an uncertain one and is locked. A finished component can be changed to an unfinished one and be locked if the outer relations are modified. The last updating user will be assigned the ownership of the lock.

The traditional acquiring lock mechanism is also supported in the system, which can be triggered by gestures when a user wants to release an uncertain or an unfinished sketch component locked by him.

4.3 Relationship maintenance

The main ideas are checking before an operation is executed and modifying after the operation is executed. Once the coordination agent parses an event, it will check whether the operation leads to relationships inconsistency (if so, the agent will deny it and notify the event sender). After the operation is executed, the coordination agent will update the global relationships and make some modifications if necessary.

The relationship maintenance is also domain related, and we need to define a set of rules for operations in a given domain.

5 Application in Cooperative Flowchart Drawing

We have implemented a prototype system for cooperative flowchart drawing, in order to evaluate our proposed strategies. Flowcharts are graphical representations of a process. Steps in a process are represented in certain symbolic shapes, and the flow of the process is indicated with lines connecting the symbols.

5.1 Domain constraints definition

We have defined the following concurrency control constraints.

First, we define the first type constraints. There are five basic shapes in flowcharts, which are shown in Fig.5.

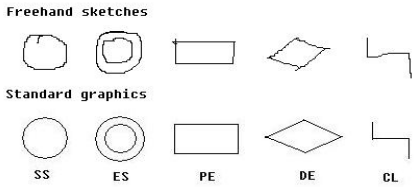

Fig.5 Basic graphical components in flowcharts

The detail of the inner geometry rules (IGR) is as follows:

IGR-1: Start State (SS)

It is composed of one single circle, which represents a start state of a flow chart.

IGR-2: End State (ES)

It is composed of two homocentric circles, which represents an end state of a flow chart.

IGR-3: Process Element (PE)

It is represented as a single rectangle, which indicates a process in a flow chart.

IGR-4: Decision Element (DE)

It is a conditional, choose element represented as a diamond.

IGR-5: Connection Line (CL)

A line coming from one element and ending at another element, which represents that control passes between the two components.

The second type constraints are defined as connection rules (CR):

CR-1: Start state connection rules

Each start state must be connected to one element.

CR-1-1: {SS: Element 1};

CR-2: End state connection rules

Each end state must be connected to at least one element.

CR-2-1: {ES: Element1 … Element n}; n>=1

CR-3: Process element connection rules

Each process element must be connected to two basic components. The components cannot be two start states or two end states. A process element only needs to satisfy one of the following sub-rules.

CR-3-1: {PE: Element 1, Element 2}

CR-3-2: {PE: SS1, Element 1}

CR-3-3: {PE: Element 1, ES1}

CR-4: Decision element connection rules

Each decision element must be connected to three components (at least one element). A decision element only needs to satisfy one of the following sub-rules.

CR-4-1: {DE: Element 1, Element 2, and Element 3}

CR-4-2: {DE: SS1, Element 1 and Element 2}

CR-4-3: {DE: Element 1, Element 2, and ES 1}

CR-4-4: {DE: SS1, Element 1 and ES1}

According to relationship maintenance requirements, we have considered the following operations and rules.

1) Add a relation

The coordination agent checks whether the relation is a legal relation according to the outer-component relationship constraints: the two elements are matched and the max connection number is not reached.

2) Delete a relation

The coordination agent checks whether a finished component will become an uncompleted one after the relation is removed.

3) Modify a component

The coordination agent checks whether the component will be modified to be another type of element.

4) Move a component

The relationships should be maintained after the component is moved to a new position. For example, A and B are both elements and connected. If A is moved to a new position, A and B are still connected.

5) Delete a component

The coordination agent checks whether the action will change a finished component into an uncompleted one. It will also update current relationships automatically. For example, A, B, and C are all process elements. A connects to B, and B connects to C. If B is deleted, A and C will be connected automatically.

5.2 Experiments and results

We have performed experiments with the prototype system. The database and coordination agent were running in a separated computer (a Dell desktop with a CPU of P4 3.0GHz, 512MB of DDR RAM). Figure 6 shows a cooperative interaction session involved by two users (User A and User B). To simplify the demonstration, we combine the two user's screenshots into a single image. User A's UI is on the left and User B's UI is on the right.

The interaction agent can give some awareness cues to the users to show the lock states of all sketch components on the UI. Each of the sketch components is bounded by a rectangle in a certain color. A red bound indicates that the component is locked by some other else; a yellow bound indicates that the user himself locks the component; and a grey bound indicates the component is not locked.

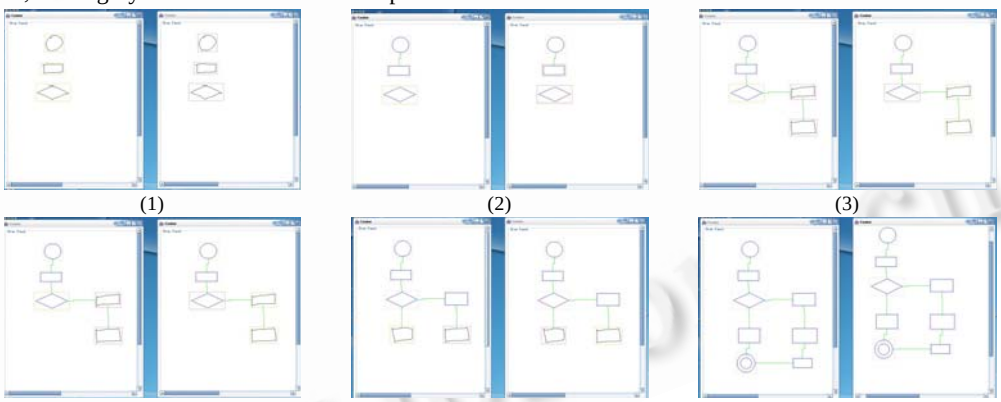

Fig.6 Screenshots of a cooperative design session (From (1) to (6))

At the beginning, the two users started a new session and began to draw sketches on their own UI. In Fig.6(1), User A drew the first three components. User B could see A's input via the coordination agent. All of the components were uncertain and locked by User A. The bounds on A's UI were yellow, and they were red on B's UI. User B could not modify any of the three components. Fig.6(2) shows that the three components were recognized as start state, process element, and decision element. The three components became unfinished ones. User A drew a connection line between the start state and the process element. The start state matched both the two types of constraints (IGR-1 and CR-1-1) and became a finished component. Then the coordination agent unlocked it automatically. The bound's color was grey now. However, the process element and the decision element did not match the connection rules and remain locked by user A. As the cooperation session was going on, in Fig.6(3), User

B drew two process elements and two connection lines. User B locked the two newly added elements. The decision element still did not match the connection rules and was still locked. In Fig.6(4), User A drew a process element and connected to the decision element. Now, the decision element matched CR-4-1 and was unlocked. One of the process elements drawn by User B was recognized and was unlocked. In Fig.6(5), the two users drew an end state and a process element. All current shapes were finished and unlocked. User A deleted a process element (on the right) which connected to other two process elements. The coordination agent connected the two elements automatically according to relationship maintenance rules. Finally, the two users finished the flowchart drawing (Fig.6(6)).

As we can see from the above description, the proposed tool can assist remote users to construct diagrams effectively.

6 Conclusion and Future Directions

In this study, we consider freehand sketches as the communicating language in cooperative conceptual design. Designers draw freehand sketches to construct diagrams to communicate with themselves and other designers. The importance and requirements of sketch-based cooperative diagramming tools are analyzed, and a multi-agent system based tool for cooperative diagram construction is proposed. The system contains a few agents to fulfill the different requirements, including user interaction, sketch recognition, communication and coordination. The features of the system can be identified as follows:

(1) Designers can use freehand sketches and gesture commands to create and edit diagrams rapidly and easily.

(2) The multi-agent system architecture can easily support a group of users to communicate and edit sketches in a common virtual drawing space.

(3) The sketch recognition agent is implemented to infer users' intention and transform freeform sketch documents directly into formalized diagrams.

(4) The coordination agent can coordinates users' cooperative operations and maintain semantic consistency of sketch documents.

We have also implemented the tool in cooperative flowchart drawing application. The primary experiment produced satisfactory results.

In the future, we decide to refine the architecture of current multi-agent system to improve flexibility (can be easily configured to fulfill new domain requirements). Context-awareness is an important issue in CSCW. We have only involved color clues (show lock status of sketch components) in this study. However, we should consider more user action clues, such as user presence clues (user's log in and log off actions), some important operation clues (notification of removal action on an existing component), etc. These are our future directions in this study.

References:

[1] Wang L, Shen W, Xie H, Neelamkavil J, Pardasani A. Collaborative conceptual design: State of the art and future trends. Computer-Aided Design, 2002,34(13):981−996.

[2] Campbell JD. Interaction in collaborative computer supported diagram development. Computers in Human Behavior, 2004,20(2):289−310.

[3] Alvarado C. A Natural Sketching environment: Bringing the computer into early stages of mechanical design [MS. Thesis]. Cambridge: Massachusetts Institute of Technology, 2000.

[4] Hong JI, Landay JA. SATIN: A toolkit for informal ink-based applications. In: Proc. of the 13th Annual ACM Symp. on User Interface Software and Technology. San Diego, 2000. 63−72.

[5] Landay JA, Myers BA. Sketching interfaces: Toward more human interface design. IEEE Computer, 2001,34(3):56−64.

[6] Sun Z, Liu J. Informal user interfaces for graphical computing. LNCS 3784, Springer-Verlag, 2005. 675−682.

[7] Prante T, Streitz N, Tandler P. Roomware: Computers disappear and interaction evolves. Computer, 2004,37(12):47−54.

[8] Zurita G, Baloian N, Baytelman F. A face-to-face system for supporting mobile collaborative design using sketches and pen-based gestures. In: Proc. of the 10th Conf. on CSCW in Design. Nanjing, 2006. 159−164.

[9] Ma C, Wang H, Dai G, Chen Y. Research on collaborative interaction based on gesture and sketch in conceptual design. Journal of Computer Research and Development, 2005,42(5):856−861(in Chinese with English abstract).

[10] Zhe F, Oliveira MM, Chi M, Kaufman A. A sketch-based interface for collaborative design. In: Proc. of the EUROGRAPHICS Workshop on Sketch-Based Interfaces and Modeling. Grenoble, 2004. 143−151.

[11] Zheng W, Sun Z. Sketch-Based CSCW system for UML design. In: Proc. of the 11th Joint Int'l Computer Conf. (JICC2005). Chongqing, 2005. 714−719.

[12] Jiang W, Sun Z, Zheng W. COSINE: A sketch-based interactive environment for cooperative design. In: Proc. of the 10th Int'l Conf. on Computer Supported Cooperative Work in Design. Nanjing, 2006. 348−353.

[13] Xu X, Sun Z, Peng P, Jin X, Liu W. An online composite graphics recognition approach based on matching of spatial relation graphs. IJDAR, 2004,7:44−55.

[14] Sun Z, Wang W, Zhang L, Liu J. Sketch parameterization using curve approximation. LNCS 3926, Springer-Verlag, 2006. 334−345.

[15] Sun Z, Yuan B, Yin J. Online composite sketchy shape recognition using dynamic programming. LNCS 3926, Springer-Verlag, 2006. 255−266.

[16] Sun Z, Li B, Wang Q, Feng G. Dynamic user modeling for sketch-based user interface. LNCS 3942, Springer-Verlag, 2006. 1268−1273.

[17] Sun Z, Liu W, Peng B, Zhang B, Sun J. User adaptation for online sketchy shape recognition. LNCS 3088, Springer-Verlag, 2004. 303−314.

[18] Sun Z, Jiang W, Sun J. Adaptive online multi-stroke sketch recognition based on hidden markov model. LNAI 3784, Springer-Verlag, 2005. 948−957.

[19] Sun Z, Zhang L, Zhang B. Online composite sketchy shape recognition based on bayesian networks. LNCS 4222, Springer-Verlag, 2006. 506−515.

[20] Sun Z, Xu X, Sun J, Jin X. A Sketch-Based graphic input tool for conceptual design. Journal of Computer-Aided Design & Computer Graphics, 2003,15(9):1145−1152 (in Chinese with English abstract).

[21] Sun C, Chen D. Consistency maintenance in real time collaborative graphics editing systems. ACM Trans. on Computer-Human Interaction, 2002,9(1):1−41.

[22] Campbell JD. Coordination for multi-person visual program development. Journal of Visual Languages and Computing, 2006,17(1):46−77.

[23] Greenberg S, Roseman M, Webster D, Bohnet R. Issues and experiences designing and implementing two group drawing tools. In: Proc. of the Hawaii Int'l Conf. on Systems Science. Kuwaii, 1992. 138−150.

附中文参考文献:

[9] 马翠霞,王宏安,戴国忠,陈由迪.基于手势和草图的概念设计协同交互的研究.计算机研究与发展,2005,42(5):856−861.

[20] 孙正兴,徐晓刚,孙建勇,金翔宇.支持方案设计的手绘图形输入工具.计算机辅助设计与图形学学报,2003,15(9):1145−1152.

JIANG Wei was born in 1981. He is a Ph.D. candidate at the Department of Computer Science and Technology, Nanjing University. His current research interests include pen-based user interface and computer supported cooperative work.

SUN Zheng-Xing was born in 1964. He is a professor at the Nanjing University. His current research interests include computer graphics, multimedia computing and intelligent user interface, etc.