[Form 3]

[Details of project proposal]

Theme:

SeeThroughTalk - a collaborative image space

1 Abstract

SeeThroughTalk is a highly collaborative working space for remote network. It introduces a new 'overlapping desktop' system to the existing Morphic GUI. The overlapping desktop is a natural extension of the overlapping window for multi-user context. Through the half-transparent desktop, members can see their partners as live objects (avatars, and windows). By utilizing the GUI, SeeThroughTalk enables us to develop shared work collaboratively while not disturbing each individual's concentration. It is useful for programming in a distributed team, or tile scripting by children in remote sites.

2 Project Summary

2.1 Basic Concepts

2.1.1 Overlapping Desktops

For the sake of the well-known overlapping window system, we are able to do multi-tasks smoothly while using only one desktop [1]. Users freely pick up the most interesting window for the present, and after some working, they move to another one. All windows have their own tasks. Even if they are not active, they can do some useful activities in the background. If we have only one window per desktop, these operations are impossible. Sometimes users also utilize multi-window collaborations to do more complex work. For example, after searching referential documents by a web browser, a user can drag & drop the searched URLs to his mail that has been half written.

Actually, it is quite natural to apply this idea to multi-user environment, because in a broader perspective, each user can be interpreted as doing one job in each desktop. Therefore, if we could have a new 'overlapping desktop system' (not window), it would be a great enhancer for team collaboration.

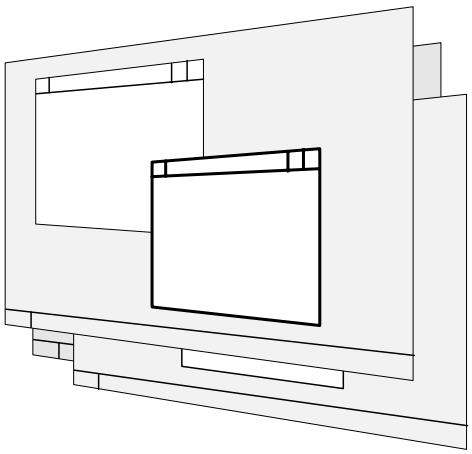


Figure 1: An overlapping desktop system

Apparently, this idea is too naïve. The most serious problem is that we cannot see other persons' activities easily. In order to do this, we explicitly have to switch to other desktops by moving behind the currently working local desktop. Normally, we have a tendency to stick to our local desktop to perform personal jobs, so if there is no support for seeing other desktops in an easier manner, the existence of other partners will be forgotten soon.

Generally, in a collaborative work, it is very important to see what co-workers are doing. By looking through what other people do, we can obtain many chances to give or receive useful feedback. Taking account of these aspects, we have to design a new overlapping GUI, which enables us to participate in creative interactive sessions on demand while keeping concentration on our personal work.

2.1.2 Dissolving Windows

Therefore, we propose an evolved overlapping desktop mechanism for multi-users. The basic idea is to provide half-transparent desktop root windows and overlap them in one area.

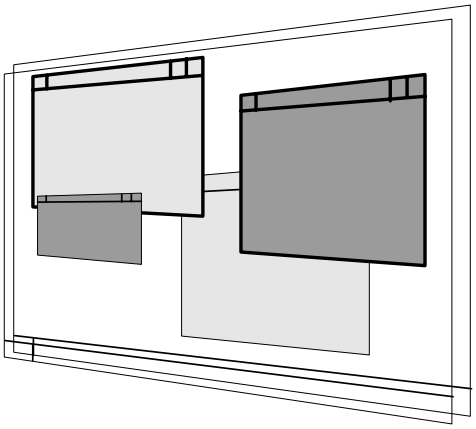


Figure 2: Half-transparent overlapping desktop system

Because the desktops are half-transparent, we can look through the other persons' work uniformly. It is easier to start collaborative sessions since the partners are always in the same place.

However, there are still issues to be considered. If the number of the participants increases, it will rapidly become harder to recognize what windows belong to the local desktop. To avoid the problem, we introduce 'dissolving' windows. In the dissolving window GUI, the windows of the partner become blurred if a user does not collaborate with him for a given period.

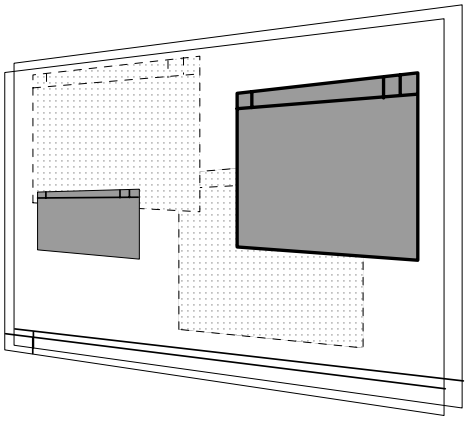


Figure 3: Overlapping desktop system with dissolving windows

Even if the windows are blurred, users can still feel that they are working together, because darker windows are moving in background. The windows never completely disappear. They just dissolve not to disturb local desktop view. If a user finds that a partner stops the work and seems to be in trouble, he can always start a collaborative session with the partner. For that purpose, avatars play an important role to start the collaboration.

2.1.3 Avatars

Avatars in the SeeThroughTalk are likened to the 'taskbar' (or dock) in a normal desktop. Taskbar is generally used to switch active applications in a desktop. It lists all the applications by a row of small icons. When we click the one of the icons, the application window represented by the icon becomes forward and active. In the SeeThroughTalk environment, users are represented as avatars. They are all arranged in a row at the side of the desktop. When we click the one of the avatar, the related user desktop windows become clear again. We can change the focus of the desktops by just clicking avatars.

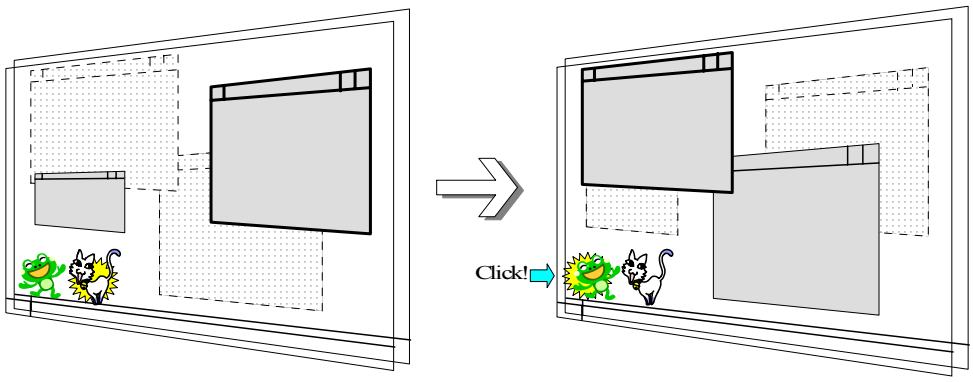


Figure 4: Avatars for changing a presently interesting desktop

After the click, you can observe the partner's work more vividly. When you find something to talk about in the partner's work, all you have to do is to drag & drop the partner's avatar to your one. Alternatively, if you would like to point out something in the partner's window, you can also drag & drop your avatar to the window.

This action will start the communication with others. You can chat with the partner through the avatars. Furthermore, if you drag your avatar to a window of the partner, the window will be sharable. In the conversation, you and the partner can operate on the same window using the cursors by turns or simultaneously. (The participants are not restricted to two persons. This shared window is observable from others, so a new avatar would come around to take part in the discussion).

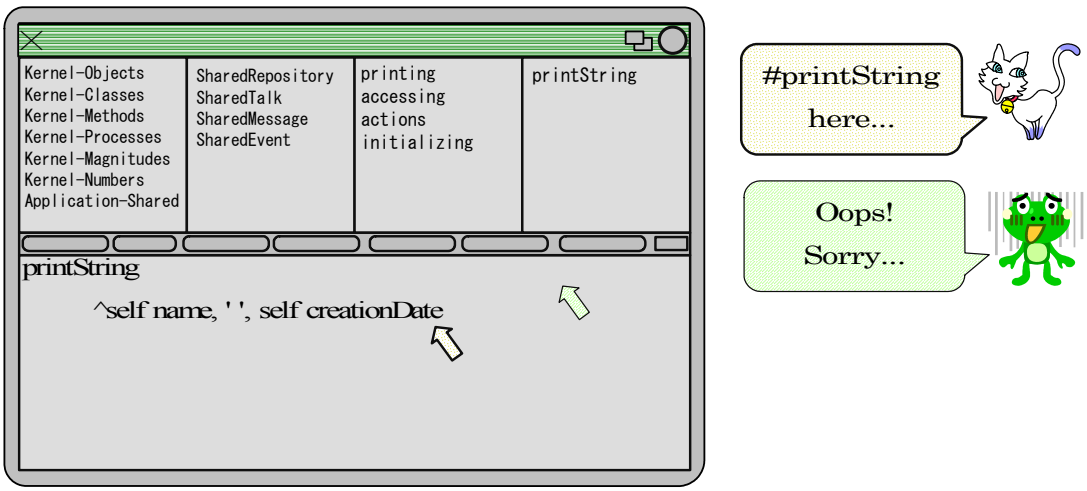


Figure 5: The shared window and chatting through avatars

As the above figure shows, avatars indicate users' feelings or working statuses. This behavior is also familiar in the normal desktop world, where taskbar's icons often show the states of the related applications.

The avatars can express the various states of users. For example, if an avatar is sleeping, it indicates that the user is not working for a while. If another avatar is alerting, your editing area conflicts with the avatar's user's area. This behavior removes the unwanted tedious interruptions that just confirm 'what's going on?' and leverages smooth communication among users.

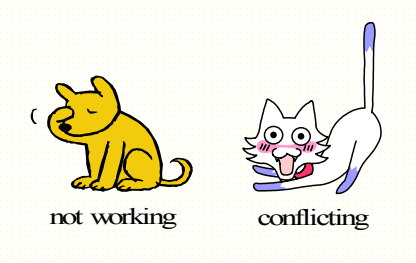


Figure 6: Avatar's various expressions indicate states of the users

2.1.4 Prevailing Images

In order to support dynamic collaborations of users, any changes made to a local image are immediately updated to the other users. Although SeeThroughTalk may use a Prevayler-like mechanism to guarantee sequential updates [2], it does not force the users to lock the editing classes or methods explicitly. This eliminates the many tedious check-in/check-out steps that are thought to be essential to many traditional source code management systems. We believe that in a highly human collaborative environment, most of the conflicts can be resolved by close virtual conversations of good-citizen members. Basically, all images in the team have the same classes and methods at any period, but only their overlapped desktop focuses are different.

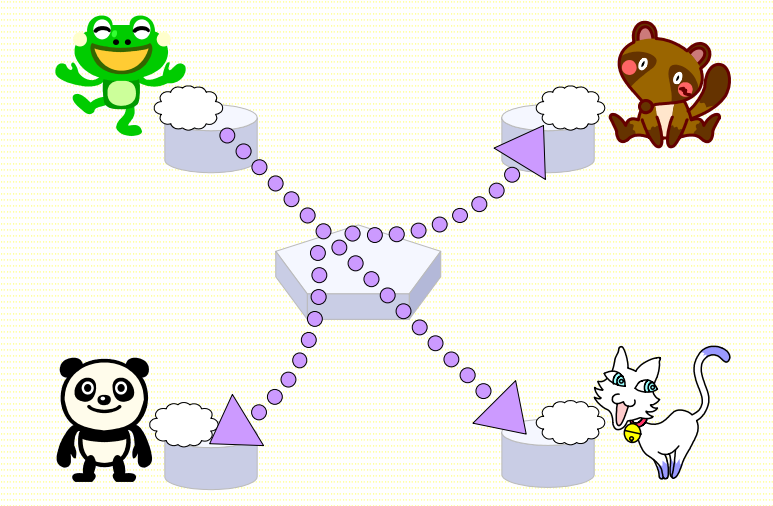


Figure 7: Prevailing images

2.2 Applications

SeeThroughTalk is primarily intended to provide a collaborative image space for Smalltalk. It magnifies the developers' speed of breeding ideas and makes the team programming amusing. Moreover, since it has a generic new GUI policy for distributed environment, there are many application usages.

For example, suppose a 'SeeThrough-SqueakToys' system [3]. If children can see the other children's work on demand, they will help each other to create interesting scripts. In the NetMorph system, students can develop scripts collaboratively if only they are sitting side-by-side [4]. In the 'SeeThrough-SqueakToys', even students in remote sites can participate in the collaborative work.

SeeThroughTalk can also provide a dynamic collaborative 'wall of wonder' in Croquet [5]. In Croquet environment, we have a 3D virtual space and many rooms to interact with other people. However, sometimes we also need whiteboard-like facilities to collect or develop our ideas. The current Croquet system has already supported displaying a Morphic 2D project in one of its windows, but since legacy GUI tools were not designed for sharing work, it is often difficult to collaborate by that window. By using SeeThroughTalk mechanism, it is possible to build a truly sharable powerful desktop board in Croquet.

3 Background and Purpose

In Smalltalk, users can write programs in a tremendously interactive way. In that development environment, they can freely browse and manipulate program objects. Users send arbitrary enquiry messages to some part of the codes at any time, and the system immediately answers the result. In general, it is very important for people to get instant feedback in order to develop interesting ideas, so this environment is ideal for many programmers. Most modern programming languages are trying to achieve such responsiveness in their IDE, but they do not reach the level of the original Smalltalk environment. However, the classic Smalltalk environment only targeted individual persons. It does not have direct supports for programming (or scripting) in teams. The Smalltalk virtual image is sometimes described as a 'lonely place'. In order to develop interesting ideas, we also need feedback not only from the system, but also from various people. Making the current Smalltalk programming environment sharable has been demanded for a long time.

Although some attempts to build team development environment have been done in commercial Smalltalks [6][7], these works have been too much influenced by file-based languages and do not support creative interactions with other programming partners (after all, they only deal with packages - bunch of source codes, not people).

Therefore, we would like to propose a true collaborative working space in Smalltalk. It focuses on human interactions and supports 'Ma' (Japanese word) in the collaborative environment, because it encourages dynamic interactions while not preventing individuals' intensive work ('Ma' means comfortable distance between people).

4 Novelty

There have been many team-programming tools designed in both academic and industrial fields. However, their paradigms are mainly based on keeping the integrity of source codes. They do not cope with the real factor essential in the success of team development - the dynamic interaction among members.

SeeThroughTalk extends the popular desktop metaphor to team programming. Team members can see their partners as live objects (avatars, and related windows), never being aware of merging files. SeeThroughTalk is a human-oriented approach and it aims at being real 'team development' environment, where a team can grow their ideas.

Several efforts have been done to support team programming in Squeak. SqCVS was the first effort to share the Squeak source codes by using CVS repository [8]. Collage and SCAN tried to provide the repository by Squeak itself [9]. However, since these approaches were rather file-oriented, they lost the liveness that is essential to the Smalltalk programming. After all, they did not attract the Squeak community well.

Nowadays, Squeakers are much broadly using SqueakMap to get/put archives of source codes [10]. Although this mechanism really serves effectively as a catalog of available works, it does not provide live, dynamic collaborative space.

Nebraska provides a collaborative space for remote users [11], but its communication model is rather a client/server style and co-developed changes only have effects on the server side. In SeeThroughTalk, all images have prevailed and they get any changes instantly, which would accelerate the collaboration.

There is also a notable system called TUKAN [12]. TUKAN adds team awareness interfaces to the existing Smalltalk browsers. In addition, it supports chatting with other developers. However, it is build by VisualWorks with Envy, so its repository model does not have the notion of the image prevalence.

5 Project Plan

We divide 8 months project into 4 phases. The 1st period has 3 phases and the 2nd period has one phase

5.1 The 1st Period (6/2003-11/2003)

5.1.1 Inception Phase: 6/2003 – 7/2003 (2 months)

- Research and technical investigation
- List main technical risks
- Research available technologies
  - Ma Client/Server, Whisker Browser, Prevayler, Croquet etc.
- Develop prototypes to evaluate the right solutions
  - Display policy
  - Replication of the images

5.1.2 Elaboration Phase: 8/2003 – 9/2003 (2 months)

- Develop overlapping desktop mechanism
  - One window, whisker mode
  - Not dissolving - remote windows just appearing on local desktop
- Develop simple dissolving windows
- Develop simple sharable window
- Develop simple avatars
- Develop image replication

5.1.3 Construction Phase: 10/2003 - 11/2003 (2 months)

- Develop a full-fledged overlapping desktops
  - Multi windows, using normal Smalltalk browsers
- Develop dissolving windows
- Develop sharable window
- Develop avatars
- Develop error-prone image replication
- Deploy a beta release
- Documentation
  - Release note of the beta release

5.2 The 2nd Period (1/2004-2/2004)

5.2.1 Deployment Phase: 1/2004 - 2/2004 (2 months)

- Improve the stability
  - Error Handling
  - System Testing
- Performance tuning
- Final code cleanings
- Deploy the first formal release as SAR
- Documentation
  - Tutorial, Design Description
  - Project Report

5.3 Task Allotment

Four members are assigned in this project. The main developer is Masashi Umezawa. Other three members are employed as part-time workers.

Members' roles are as follows:

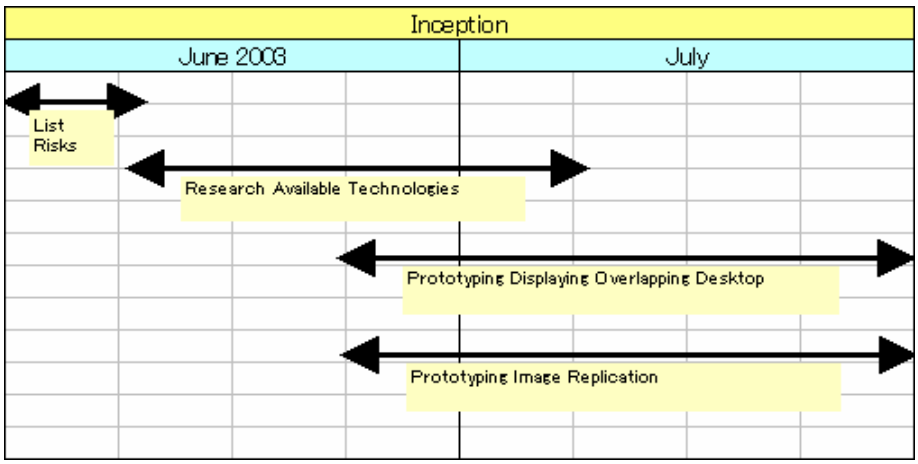
- Manager and Chief Developer
  - Masashi Umezawa
- Support Developers (part-time)

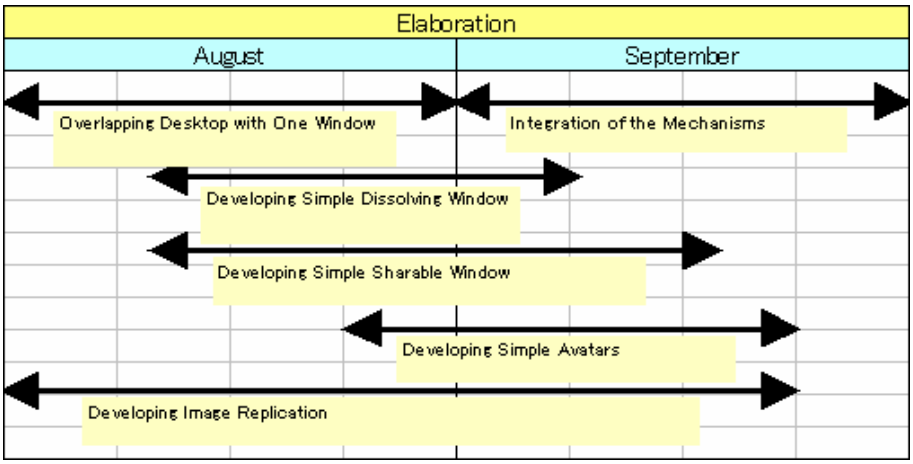
- Kazumi Okamoto, Yoji Kanno

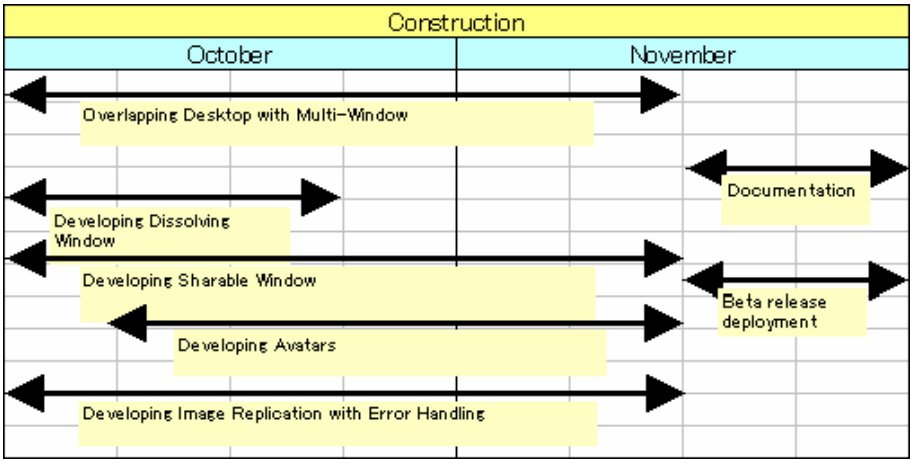
- Beta Tester (part-time)
  - Tetsuo Ogino (Kyoto University student)

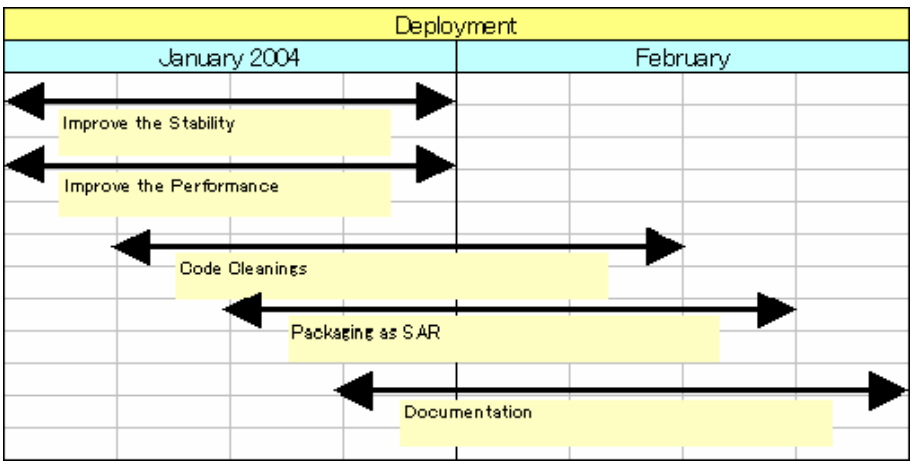
5.4 Schedule

These tables show the summarized schedule:


Inception
June 2003
July
List Risks
Research Available Technologies
Prototyping Displaying Overlapping Desktop
Prototyping Image Replication


Elaboration
August
September
Overlapping Desktop with One Window
Integration of the Mechanisms
Developing Simple Dissolving Window
Developing Simple Sharable Window
Developing Simple Avatars
Developing Image Replication


Construction
October
November
Overlapping Desktop with Multi-Window
Documentation
Developing Dissolving Window
Developing Sharable Window
Beta release deployment
Developing Avatars
Developing Image Replication with Error Handling


Deployment
January 2004
February
Improve the Stability
Improve the Performance
Code Cleanings
Packaging as SAR
Documentation

6 Project Expenses, Budget Estimation

6.1 The 1st Period (6/2003-11/2003)

- Personnel costs: 6,060,000 yen
  - (7500 yen/h * 741 hours) + (2500 yen/h * 78 hours)*2 + (1500 yen/h * 75 hours) = 6,060,000
- Management costs: 1,240,000 yen
- Other costs: (Machines, Supplies, Traveling Expenses, Investigations): 470,000 yen
- Total: 7,770,000 yen

6.2 The 2nd Period (1/2004-2/2004)

- Personnel costs: 1,770,000 yen
  - (7500 yen/h * 218 hours) + (2500 yen/h * 21 hours)*2 + (1500 yen/h * 20 hours) = 1,770,000
- Management costs: 360,000 yen
- Other costs: (Machines, Supplies, Traveling Expenses, Investigations): 100,000 yen
- Total: 2,230,000 yen

- The whole project total: 10,000 thousand yen

7 Impact of the Project on Completion, Commercialization Plan

SeeThroughTalk proposes a new human-oriented approach in a team programming development. Those who are currently using traditional source management tools in open-source projects will find more power in SeeThroughTalk. There are also merits in end users. Wherever they live, they can enjoy great dynamic lectures from remote masters. SeeThroughTalk lecture would be very effective because you can see the whole desktop operations of teachers and put some questions at any time.

By using SeeThroughTalk, people all over the world can participate in exciting sessions more easily and develop great ideas. This communication augmentation tool is desirable in the upcoming broadband internet age.

For the commercialization, there are great possibilities to make SeeThroughTalk a popular product like Envy. However, for the first step, we think it is very important to popularize this new paradigm. Therefore, SeeThroughTalk itself is completely open-source and free. The derived tools (like SeeThroughTalk in other languages) could be commercial products.

8 Past Work and Accomplishments of the Members

8.1 Masashi Umezawa

8.1.1 Academic Background:

Kyoto University, American Literature, 1994

8.1.2 Business Career:

- Osaka Gas Information System Research Institute (OGIS-RI), 1994-2001
- Mamezou Inc., 2001-2003

- VisualWorks (Smalltalk) product support (3 years)
- VisiBroker (CORBA ORB) product support (2 years)
- Object-Oriented software development/consultation (9 years)
- Trainer of Smalltalk, Java and distributed computing (4 years)

8.1.3 Experiences:

- Developed device control system in Java (using CORBA)
- Developed CORBA performance evaluation toolkit (C++)
- Localization of DistributedSmalltalk
- Localization of Objectivity/Smalltalk (OODB)
- Developed Rose/Smalltalk (OO-CASE) Japanese version
- Developed EJB framework for order system
- Developed courseware of distributed computing and advanced Java
- Defined specification of ORB for car area network.
- Developed prototype IDE for car ORB (Squeak)
- Developed car-navigation system prototype (Squeak)
- Developed OS prototype for home appliances (Squeak)

8.1.4 Technical Skills:

- Programming Language: Smalltalk, Java, C++, XML
- Distributed Computing: CORBA, SOAP
- DB: Objectivity (OODB), Oracle
- Modeling: UML, Booch
- Development Process: XP, RUP

8.1.5 Free Software:

- Squeak
- Dandelion (Smalltalk analysis/output framework)
- SmallInterfaces Squeak Port (Interface support for Smalltalk)
- SoapOpera (SOAP basic implementation and Lightweight ORB wrapper for SOAP)
- SIXX (Portable Smalltalk XML serializer/deserializer)
- NetMorph (an intuitive mobile object system - 2002 IPA Exploratory Software Project)

9 References

[1] Alan Kay, "The Early History of Smalltalk", in Bergin, Jr., T.J., and R.G. Gibson. History of Programming Languages - II, ACM Press, New York NY, and Addison-Wesley, Reading MA 1996, pp. 511-578

[2] Klaus Wuestefeld et al., "Prevayler", http://www.prevayler.org/, 2001

[3] Alan Kay, "Etoys and Simstories in Squeak", http://www.squeakland.org/author/etoys.html

[4] M. Umezawa, K. Abe, S. Nishihara, and T. Kurihara, "NetMorph - an Intuitive mobile object system", http://swikis.ddo.jp/NetMorph, 2002

[5] David Reed, David Smith, and Andreas Raab, "Croquet", http://www.opencroquet.org/, 2002

[6] Joseph Pelrine, Alan Knight, and Adrian Cho, "Mastering Envy Developer", Cambridge University Press, 2001

[7] Cincom Systems, Inc., "Team development in VisualWorks", http://www.cincom.com/pdf/store-twp-1020-01.pdf, 2000

[8] Martin Kobetic et al., "SqCVS", http://cvstproj.sourceforge.net/, 2000

[9] Hans-Martin Mosner, "SCAN", http://minnow.cc.gatech.edu/squeak/745, 2001

[10] Göran Hultgren, "SqueakMap", http://minnow.cc.gatech.edu/squeak/2726, 2002
[11] Lex Spoon and Bob Arning, "Nebraska", http://minnow.cc.gatech.edu/squeak/1356, 2000
[12] Till Schümme, "TUKAN", http://www.ipsi.fhg.de/concert/activities/internal/tukan.html, 2002