CompUTE: A Runtime Infrastructure for Device Composition

Jakob E. Bardram
IT University of Copenhagen
Rued Langgaards Vej 7,
DK-2300 Copenhagen
bardram@itu.dk

Christina Fuglsang
Systematic Software Engineering
Søren Frichs Vej 39,
DK-8000 Aarhus C.
ch.fuglsang@gmail.com

Simon C. Pedersen
Miracle A/S
Borupvang 2C,
DK-2750 Ballerup
simon@clemen.dk

ABSTRACT

In this paper, we present the idea of a *composite device*, which is one device made up of a composition of several separate devices working together in concert. We present the *CompUTE architecture* as a runtime infrastructure for device composition. This architecture is implemented in the *CompUTE .net platform*, which enables device composition between Windows XP devices based on an 'extended desktop' metaphor. We discuss how the CompUTE .net platform enables users to merge multiple computers into one composite device with an enlarged desktop area and several keyboards and mice. Each user is equipped with a unique cursor, which works across all connected devices and she can drag and drop any application window and files between the connected devices. User can also work collaboratively on the composite device. We reports from a usability study involving 24 participants who found the CompUTE .net platform very useful, easy to use, and easy to learn.

Keywords

Composite device, Co-located cooperation, Control re-direction, Application view re-direction, Distributed clipboard

Categories and Subject Descriptors

H.5.2 [**Information Interfaces and Presentation**]: User Interfaces—*graphical user interfaces, windowing systems*

General Terms

Design, Human Factors

1. INTRODUCTION

The fundamental design paradigm underpinning contemporary software and hardware is based on the desktop-centered computer as a personal and isolated device. This device is tailored to the needs of one individual user interacting with one single device to accomplish her tasks. This makes it inherently difficult, and often impossible, for users to share resources, like screen real-estate, applications, files and input controllers among multiple interconnected devices.

A significant body of research has been trying to extend this one-to-one relationship between the computer and the user into a many-to-many relationship where the same computer can be used by many users and the same user can use several computers simultaneously. Examples of the former are single-display groupware [13, 12, 16], which allow co-located users to use the same computer, and examples of the latter are various smart space technologies, which allow users to use several co-located devices [15, 11, 3, 7].

This paper adds to the growing body of research on interaction techniques and technologies for loosening the tight one-to-one coupling between the user and the (personal) computer. The paper presents the *CompUTE*[1] *architecture*, which is a language and platform independent architecture for device composition, and presents the *CompUTE .net platform*, which is an implementation of the CompUTE architecture in the Windows XP operating system. The aim is to support device composition as an integrated component in the devices' underlying operating system, and allow the user to work with legacy applications without any modification.

From a usage point of view, the ComPUTE .net platform supports three basic types of scenarios. First, it supports one user to use several devices in her proximity. It is not uncommon for a user to have more than one personal computer, like a desktop computer and a laptop, as illustrated in figure 1a. In this situation, the ComPUTE infrastructure mimics an 'extended desktop' where several monitors are attached to a multi-output graphics card on one computer. This is a well-known metaphor to many users and is being used in many situations [4]. The extended desktops metaphor in the CompUTE .net client enables the user to; move the mouse freely across both devices' displays; use both devices' mouse and keyboard to interact on all displays; move application windows and sub-windows to all displays, including being able to remotely control these windows and the application they host; move objects and files between connected devices; and use the clipboard across connected devices.

Second, the ComPUTE platform supports co-located users to share available devices. Figure 1b illustrates a typical sce-

[1]CompUTE is an acronym for **Comp**osite, **U**biquitous, **T**ranscending **E**nvironment.

AVI'10, May 25–29, 2010, Rome, Italy.

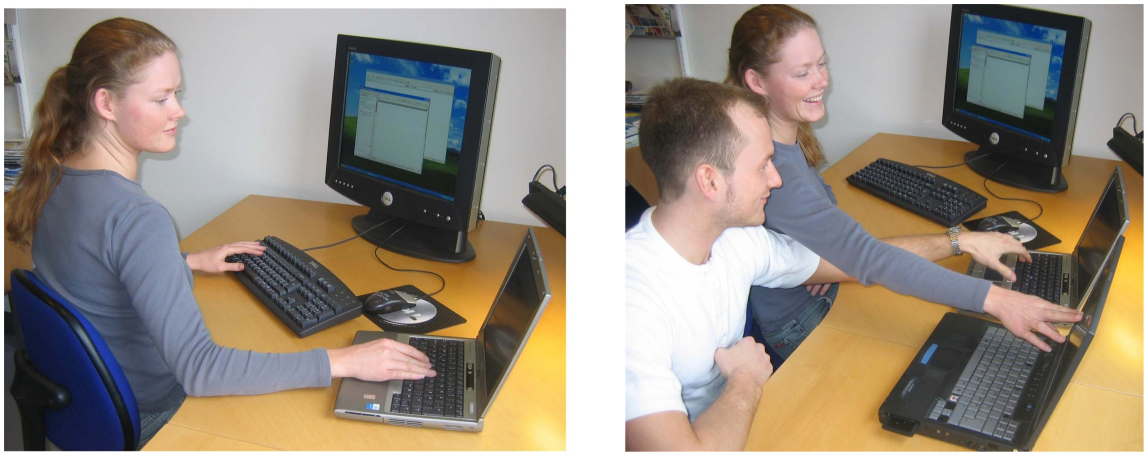

Figure 1: Two typical work situations for a computer user; (a) using several computers together, (b) collaborating with a friend who brings another laptop to the table.

nario, where a second person enters the office with his own laptop. By connecting to the ComPUTE infrastructure in the office, he is able to use his own computer to move the cursor over to the desktop computer, grasp a window containing a document, move it to his laptop, and start editing it. The first user is able to move her cursor to his laptop and point to things she wants him to work on. Using her own keyboard, she is able to co-edit the document on his laptop. She may then decide to drag a file from his laptop to her laptop and start editing it by using the keyboard and mouse of the desktop computer.

Third, the ComPUTE platform supports setting up device composition inside a smart space with a pre-configured set of displays e.g. on the walls and in the tables. For example, the infrastructure would help configure three wall displays located adjacent to each other to melt together and work as one large screen. Furthermore, by connecting to the smart room's ComPUTE gateway an user entering the room can connect his own laptop to work in concert with the other displays. This means that a user would be able to move content, like an application window, to a wall-based display for others to see, or be able to use the keyboard of the laptop to edit a document shared on the table display. And if s/he allows for data transfer, others in the room would be able to access files on the laptop.

As discussed in section 2, the ComPUTE platform unifies and – in certain places – extends previous work addressing the challenges of isolated personal computers. For example, PointRight [5], the Synchronized Clipboard [6], and VNC [10]. This previous research provides important pieces and stepping stones in achieving the goal of being able to create composite devices out of personal computers. Nobody has, however, presented a coherent approach to complete device composition which also supports co-located collaboration. Besides making specific improvements to existing work, the main goal – and contribution – of the present work is to present a synthesis and extension of previous pieces of research into a unified platform for device composition.

Section 3 introduce the *CompUTE architecture*, which is a language and platform independent architecture for easy device composition. Section 4 introduces the *CompUTE .net platform*. Section 5 presents an evaluation of the CompUTE .net implementation, which demonstrates that users found device composition very useful, easy to use, and easy to learn. The paper is concluded in section 6.

2. RELATED WORK

A range of research into smart space technology has been addressing architectures for device composition in such an environment. For example, the ConnecTables [15] could be rolled together so that they together form one display. The Gaia middleware for 'Active Spaces' [11] enables applications to move between host devices. From an infrastructure perspective, these smart space technologies are designed to work in a controlled and centralized computing environment running a range of specific services for device composition. In contrast, the CompUTE infrastructure is designed to be more lightweight. This enables easy and ad-hoc device composition outside the walls of such smart spaces, allowing e.g. two users to meet, create a composite device based on their laptops. This goal is similar to the Speakeasy [3] and the Platform Composition [7] systems. The ComPUTE platform, however, seeks to create a very simple user experience by mimicking the well-know metaphor of an 'extended desktop', enabling seamless sharing of applications, files, and data objects. Moreover, the ComPUTE platform supports co-collated simultaneous collaboration; i.e. two users can write collaborative in e.g. a text file.

Looking at the more specific concepts and technologies for realizing the user experience introduced above, the ComPUTE platform builds upon and extends a range of existing research. The re-direction of control flow (mouse and keyboard events) to another device is similar to PointRight [5]. ComPUTE, however, extends control re-direction to support co-located cooperation. PointRight is designed to re-direct a *single* mouse and keyboard across devices. In their case, a mouse attached to computer A takes over control of the cursor on computer B when A moves to B. In contrast, CompUTE allows several cursors – called *remote pointers* – on the same display which can be used together. Hence, both user A and B can work with their mice on the same display.

Research on Single Display Groupware (SDG) has been

trying to support co-located collaboration on a single display. However, due to the inherent singular event loop and focus model i all contemporary operating systems, all examples of SDG are special purpose applications or programming frameworks taking over full control from the operating system. Examples are Kid++ [13], the DiamondSpin TableTop toolkit [12], and the SDG Tootkit [16]. In contrast to these special purpose applications, we have implemented a mechanism – called *event transactions* – as part of the operating system, which comes were close to enabling transparent co-located cooperation on a single display despite the singular event loop.

Redirecting of the application view in the ComPUTE platform is done by re-directing the applications graphical output to another device's display buffer similar to e.g. VNC. However, whereas VNC [10] redirects one computer's whole desktop to another computer, and WinCuts [14] redirect a small cut-out of an application's window to another computer, the ComPUTE platform re-directs an application view, and by some technical engineering makes it re-appear as an application window on the other device's display. The way a user re-directs one application from one device to another is simply by dragging it from one display to the other. The commercial system called MaxiVista[2] allow the user to attach one PC to another and use the input devices across both devices as well as drag windows back and forth. In MaxiVista, however, the 'primary' PC takes full control over the 'secondary' PC, and the latter becomes a passive screen attached to the primary screen. Hence, Maxi Vista does not enable parallel working on both devices.

The ComPUTER platform implements a distributed clipboard similar to the synchronized clipboard [6] allows users to share data object using the familiar cut-copy-paste methods. The CompUTE platform also implements a distributed drag-and-drop mechanism allowing a user to drag a file from one device and drop it on another. This mechanism resembles the Hyperdragging mechanism [9] but is much more straight-forward in its interaction design.

3. THE COMPUTE ARCHITECTURE

The CompUTE software architecture is illustrated in figure 2. The Olympus Gateway is responsible for service registration, discovery, and publication. The CompUTE Client resides on each of the devices participating in the device composition. The main dynamic control flow in the architectures is as follows:

1. In the *discovery and publication phase*, a device discovers the gateway and registers its services at the gateway.

2. In the *bind phase* two or more devices query the gateway for possible services to bind to each other in matching pairs. This includes technical details of how to communicate with each other.

3. In the *usage phase* the devices work peer-to-peer using the requested services, like control re-direction, application view re-direction, and/or data sharing. Note that the gateway is not involved in this actual usage phase.

[2]http://www.maxivista.com/

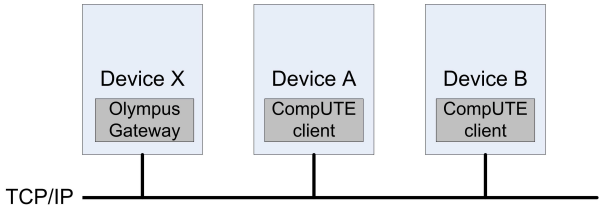

Figure 2: The CompUTE Architecture consisting of a client component running on each device and a gateway for service discovery and binding.

4. In the *unbind phase* the binding between the devices are removed and this change is passed on to the gateway.

The Olympus Gateway can be deployed centrally in an organizational network but can also reside on the different devices, like a laptop. This latter option enables two or more user to engage in ad-hoc collaboration as they meet in new places, like an airport (c.f. [3]). In order to be platform and language independent, the CompUTE architecture is using UPnP for service discovery and publication, and SOAP for remote method invocation.

The two main objects which are used and stored in the Olympus Gateway are 'Device' and 'Binding'. A *Device* represents a physical device (e.g. a laptop) and models general data (e.g. its user friendly name), the services that this device is able to handle (control re-direction, application view re-direction, and/or data exchange), and details about these individual services including technical details on how this device wants to handle the direct peer-to-peer connection in the usage phase.

A *Binding* represents a reference between two devices which are bound together. Bindings are always done in pairs – this turned out to be a simple technical as well as conceptual model for the users to understand. A binding contains information about the specific binding, like which services are connected and used in this binding.

The CompUTE Client resides on each of the devices participating in service composition. The main responsibility of the CompUTE client is to support the communication and control protocol specified above in collaboration with the Olympus Gateway, and to present a user-interface to the user which allows a transparent use of the platform for ad-hoc device composition.

4. THE COMPUTE .NET CLIENT

The CompUTE .net implementation of the CompUTE architecture enables device composition in the Windows XP operating system. This client mimics an 'extended desktop' where several monitors are attached to a multi-output graphics card on one computer. Extended desktops enable the user to:

- Move the mouse cursor across multiple displays.
- Use input devices like the mouse and keyboard to interact on all displays
- Move application windows to all displays, including being able to remotely control these applications.

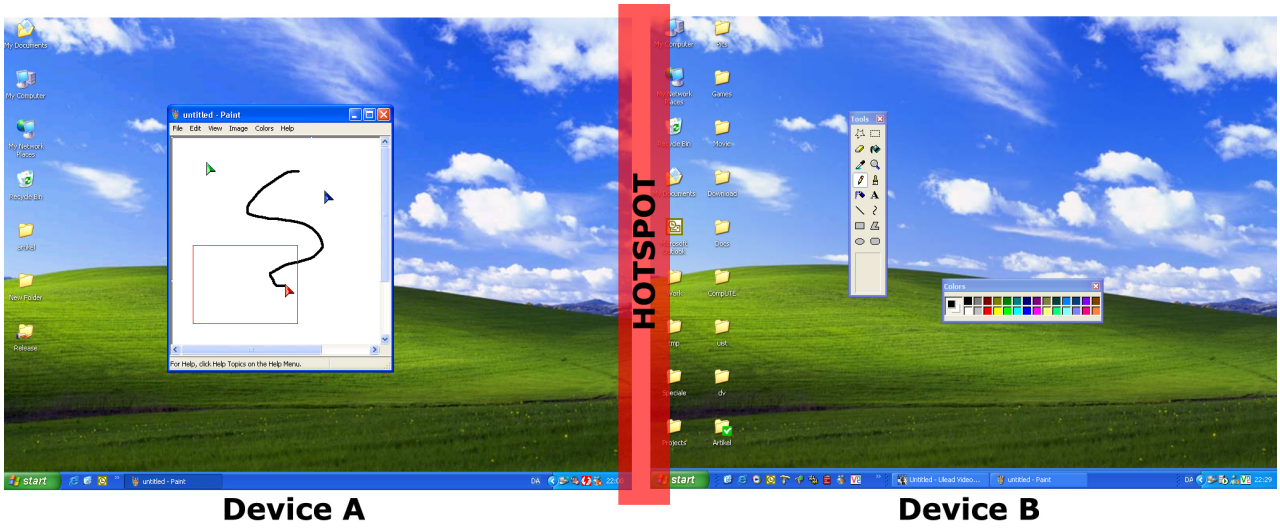

Figure 3: The Extended Desktop Metaphor in the CompUTE platform. Notice the three telepointers supporting *control re-direction* and how MS Paint is distributed across the two displays, supporting *application view re-direction*.

- Move objects and files between displays.
- Support a shared clipboard across connected devices.

An extended desktop in the CompUTE client is illustrated in figure 3. Devices can be bound together according to their physical location in relation to each other – a device can be bound to other devices located north, east, south, or west of it. Bindings are made in pairs according to this physical layout – device A is, for example, at its east border bound to the west border of device B. When a binding is created the user specifies this orientation of the binding (see also section 4.1). Hence, when e.g. the mouse cursor is moved to the east side of device A's workspace, it is re-directed to device B's workspace and appears on the west side.

To support automatic re-direction of control events (e.g. the mouse cursor) and application view re-direction when an application window is dragged to the border of the display, the CompUTE .net client operates with *hotspots* at each side of the display. A hotspot represents an interactive contact point between the displays of two connected devices (see figure 3). A hotspot is activated when the mouse is moved to an edge and is used when redirecting control (mouse and keyboard), application windows, and objects/files between devices.

4.1 Configuration of a Composite Device

When the CompUTE .net client is installed, it runs as a Windows XP service and its only visible representation is a small icon in the System Tray. When the CompUTE client service is started it automatically starts searching for an available gateway. When a gateway is discovered, this is notified to the user as shown in figure 4. Information about this device, including service information, is stored in the gateway.

When double clicking on the CompUTE icon the user is presented with the CompUTE control panel shown in figure 5. The most important tabs are the 'Devices', 'Bindings', and 'Hotspots' tabs. Once the client is connected to

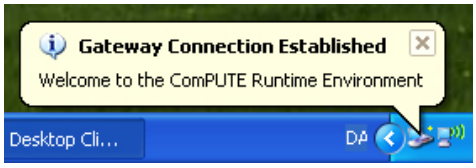

Figure 4: The icon with the welcome message shown when a CompUTE .net client discovers the Olympus Gateway.

the gateway, the 'Devices' tab lists known devices and their available services.

Implemented services in the CompUTE .net client are:

- The *Mouse* service for control re-direction of mouse events (see section 4.2).
- The *Keyboard* service for control re-direction of keyboard events (see section 4.2).
- The *View* service for application view re-direction, i.e. to support Windows XP windows to be moved between connected devices (see section 4.3).
- The *Clipboard* service for distributed cut-copy-paste (see section 4.4).
- The *File* service for file sharing, including distributed drag-and-drop (see section 4.4).

A binding can be made by selecting a device in the list and pressing the 'Add Binding' menu. As shown in figure 6, the user first selects which services to bind to and then choose the location (i.e. hotspot) of the device in relation to the current computer.

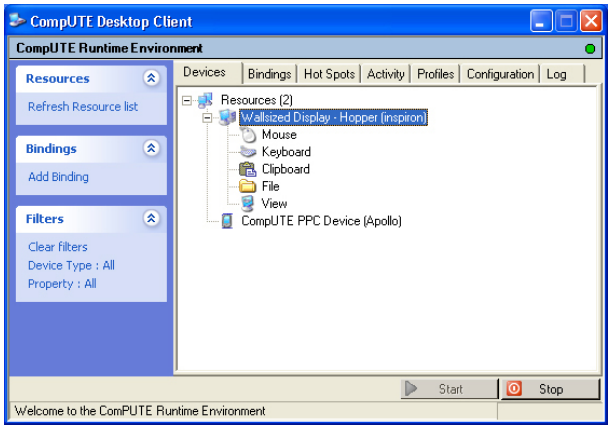

Figure 5: The CompUTE .net Control Panel

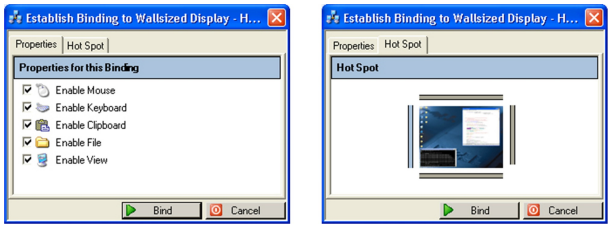

Figure 6: Creating a binding. a – Selecting the services ('properties') of this binding. b – Selecting the hotspot, i.e. the physical orientation of the binding.

4.2 Control Re-directing

'Control re-direction' denotes the action of re-directing events from an input controller attached to one device to another device. With respect to the Windows mouse we have two design options:

1. Adhere to the 'extended desktop' metaphor and let the physical mouse on device A control the operating system mouse on device B. Consequently, device A takes control of device B and only the user sitting at device A can use device B. This supports *single-user* situations.

2. Represent the mouse from device A on the display of device B with a new special mouse cursor. In this way several users can simultaneously use their own mouse, each with a unique mouse cursor. This supports *co-located collaborative* situations.

PointRight [5] implements option 1, which is only usable in the single user scenario. Because we also want to support co-located collaboration, the CompUTE .net client implements option two. Hence, the ComPUTE client supports multiple cursors on a device's display, each representing mouse events from this and other devices bound to this device. Figure 7 shows three remote pointers; one from the host device and two from two connected devices. Each mouse pointer is drawn with a distinctive color. The host device's system mouse pointer is replaced with a colored pointer once a remote device moves a pointer to the device. When the last remote pointer leaves the devices again, the normal system mouse pointer is restored.

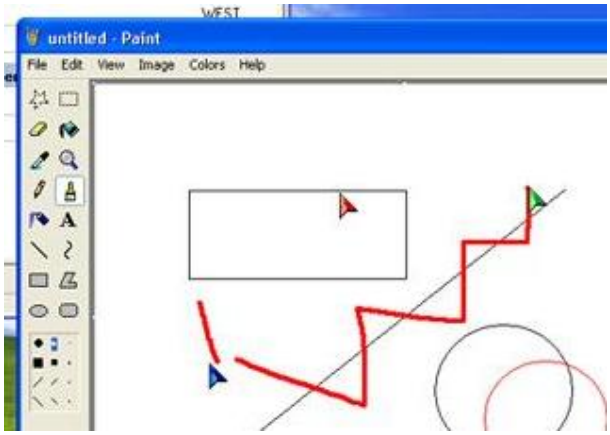

Figure 7: Remote pointers – This picture shows three remote remote pointers each with a distinctive color (red, green, blue). All three of them can manipulate the drawing with no modification of MS Paint.

The goal is to support multiple users using the same display with their remote pointers. However, because of the underlying singular focus model in Windows XP, only *one* window can have focus at the time and all input events goes to this window. Furthermore, only *one* event queue exists, which means that input events from multiple input devices are serialized into one line of events, irrespectively of their origin. These limitation are so deeply rooted in the Windows XP operating system. A core design goal of the CompUTE platform was, however, to support applications natively in the default operating system and we hence have to accept these limitations.

In order to support co-located collaboration despite these limitations, we have implemented an *implicit floor control mechanism*. In this floor control mechanism we distinguish between passive and active input events. Passive input events, like movement of the cursor, are controlled solely by the property listener that controls the representation of a specific remote pointer. Active input events, like mouse clicks and keyboard typing, are put into the operating system's event loop. In order to support dragging, we have created an *event transaction* mechanism in the event handling. A transaction is defined as a set of input events which have to be executed in one coherent transaction, like a database transaction. In the CompUTE client, a mouse click transaction runs the floor control mechanisms; a mouse event transaction is created when a 'Mouse Down' event is triggered and *only* events from this mouse are put into the event queue until a 'Mouse Up' event occurs, which ends the transaction. In this way, mouse events from different mice are not mixed up randomly in the event queue while a user is e.g. dragging a file or a window across the screen. However, due to the single event loop in all operating systems, this floor control mechanism consequently disables all active mouse events from other mice during the transaction. The consequence is, for example, that when user A is dragging a file, user B cannot click with his mouse. He can only move it around. This floor control mechanism is however localized to a single machine, so it is possible for the user sitting at

device A to send active mouse events to device B, while the user sitting at device B simultaneously sends active events to device A.

Furthermore, singular focus implies that all input events go to the application in focus. If two users are connected to a device both users' keyboard events would go to the application in focus. In this way two users can write collaboratively in a shared window like a text editor. But they cannot write collaboratively in two different windows on the same device.

4.3 Application View Re-directing

Application view re-direction in the CompUTE .net client allows any application window and child window to be positioned on arbitrary displays participating in the composition – see figure 8. Application re-direction is coupled to the control re-direction mechanism. This enables the user to drag a window from one screen to another by moving it to the edge (i.e. a hotspot), thereby initiating a re-direction to the device bound to this hotspot. In contrast to e.g. Gaia [11] application view re-direction directly supports the 'extended desktop' metaphor and is done by direct manipulation. Furthermore, because users often arrange their use of an application by having the main document, like a large CAD drawing, on one main display and related toolboxes and input boxes on adjacent displays [4], the CompUTE .net client also supports users to distribute toolbars or other sub-windows of an application across the displays in the composite device, as illustrated in figure 3.

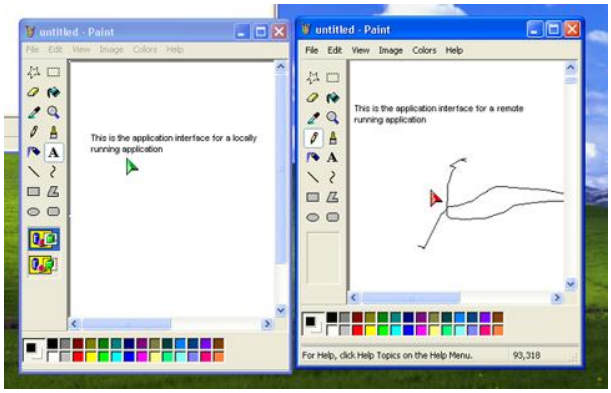

Figure 8: Application View Re-direction – In this picture one instance of the MS Paint application is running locally and the other instance is re-directed from another computer. There is no visible or interactive differences between these two instances and both can be manipulated by the mouse and keyboard from both devices.

Note that the application is not migrated between the connected devices. The application is still running on the executing computer, and the graphical output of the application window is redirected to the remote computer. This enables the user to freely move a window from an arbitrary legacy application to a connected machine without interrupting the executing application. The executing application is unaware of the re-direction and continues to execute on the hosting device, but is not visible there any more. When an application window is moved from one display to another, the actual migration happens when the mouse pointer hits the edge of the display (i.e. the hotspot). This implies that in the current implementation, a window cannot be shown across multiple displays.

The CompUTE .net client uses an *event-based* updating of the remote window. This is in contrast to VNC [10] and WinCut [14], which is based on a periodic update (every second) of graphic content a window. Instead of these pull strategies, the CompUTE .net client uses a push strategy where the application window on device A is pushing graphic content to device B when a change occurs. This event-based updating of graphics has a clear performance advantages over VNC and WinCut.

4.4 Data Sharing and Exchange

In order to support transparent sharing of data between combined devices, the CompUTE .net client supports two services:

- Flexible exchange of files, including drag-and-drop of files between connected devices.
- Distributed clipboard enabling the user to cut-copy-paste across connected devices.

When the user drags a file to an edge (i.e., a hotspot) on device A, then a file transfer process (using the FTP protocol) is started to the connected device B. We cannot, however, support continuous dragging on device B since the file transfer might take some time and the mouse cursor would hence be 'frozen' meanwhile. The file is merely placed in a specified folder on device B – default is the desktop folder. When the file has been transferred, the user can continue to drag the file to its final location.

The *distributed clipboard* is implemented using a 'Clipboard Controller' and a 'Clipboard Listener'. The clipboard controller captures content from the local clipboard and sends it to clipboard listeners on all connected devices. The clipboard listener receives clipboard content and inserts it into the local clipboard. Consequently, the ComPUTE infrastructure maintains an illusion that the composite device has one shared clipboard. In the current implementation, the types of object are limited to text strings and graphical bitmaps, but can be extended to support all Windows objects.

5. EVALUATION

Our experimental evaluation of the ComPUTE .net platform focused on the following usability aspects: (i) *Usefulness* – can users see the benefit from having such a possibility embedded in their operating system; (ii) *Ease of Use* – how easy is the ComPUTE to use; (iii) *Learnability* – how easy is the ComPUTE platform to learn to use; (iv) *Configurability* – how easy is it to create, maintain, and finalize composite devices that are aligned with the need of the user in a certain situation.

24 (7 female and 17 male) subjects participated in the experiment. A survey questionnaire done before the experiment revealed that these users used computers several times every day; they had, on average, more than two computers on their desk (2.33); they frequently transferred data between these computers and used them simultaneously for various tasks; and they often collaborated with others by sitting next to them, each using their own computer. All is all, these users were heavy computer users and hence good

candidates for using technologies like the CompUTE platform.

5.1 Methods

In order to evaluate co-located collaboration, the subjects participated in the experiment in pairs. The experiment contained three phases: (i) a short introduction to the CompUTE platform, (ii) solving four distinct assignments using the CompUTE platform, and (iii) filling in a questionaire and participating in an interview. The solving of the assignments and the interview were video-recorded and later reviewed for qualitative and formative information to help us improve the design of the system.

The assignments included (i) making a composite device between two computers; (ii) start MS Paint on device A and move its toolbars and color palette to device B, and then draw a picture using different tools and colors; (iii) use MS Word to create a document together by doing shared editing, and by copy-pasting different material into the document from both devices; (iv) use MS Powerpoint on device A and MS Visio on device B and making a presentation with drawings; (v) go to a meeting room and bind a laptop to the computer running the projector, and then move the presentation to the projector and make the presentation, using the mouse on the laptop to control it.

The questionaire was designed by combining the lessons from the Questionaire for User Interface Satisfaction [1], the 'Perceived Usefulness' method [2], and the questionaire used by Rekimoto [8]. For each of the usability aspects listed above, we created a number of questions which were mixed in between each other. In total, the questionaire consisted of 47 multiple-choice questions on a 5-point Likert scale.

5.2 Results

The overall quantitative results regarding the stated usability aspects is shown in figure 9. The results show, in general, that the subjects found the system very useful (4.29 out of 5); they found the system very easy to use (4.36); and they found is easy to learn as well (4.01). However, overall it seems like the subjects found the configuration of a CompUTE composite device less easy (3.24 out of 5).

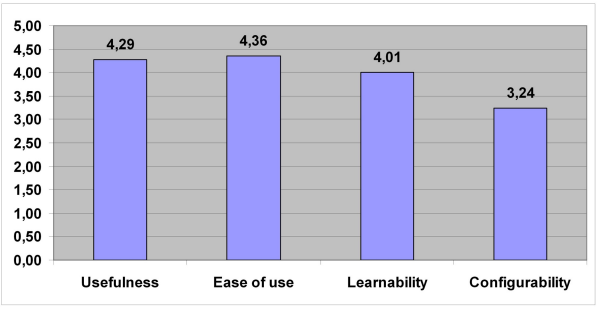

Figure 9: The overall results of the experimental evaluation of the CompUTE .net client.

Figure 10 more specifically shows the results for each of the three core services provided by the ComPUTE platform. In general all services were perceived useful, was easy to use, and easy to learn. When analyzing the reason for the marginally lower grading of the ease of use for control redirection, the users stated the lack of simultaneous use of several mice on the same screen, i.e. the inherent limitation of the single focus and single event loop in the Windows operating system. Similarly, the marginally lower score on learnability for view re-direction were due to a small delay (0.5 seconds), which occurred between a window was dragged to a hotspot and when it appeared on the remote display. In the follow-up interviews most subjects stated that they quickly got used to this delay. More importantly, however, there were no perceived performance bottlenecks in the system. Once the view was re-directed, control and view re-direction happened so efficiently that no users experienced any delay or breakdown in the 'extended desktop' metaphor.

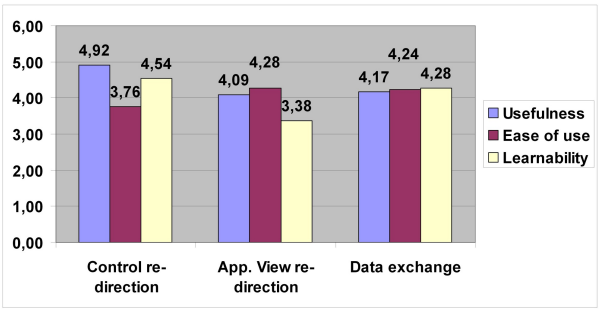

Figure 10: The detailed results of the experimentatal evaluation with regards to control re-direction, application view re-direction, and data exchange.

The qualitative results drawn from our analysis of the video recording of the use of the system and our subsequent interviews revealed that users were able to use the CompUTE system transparently. Once a composite device was created, the users fluidly used their mouse and keyboard on several devices, they transparently moved around application windows, and they moved files between computers and used the distributed clipboard without paying any attention to the fact, that they were on different physical devices. Most of the time, users did not even notice that they were not using the native mouse and keyboard on a device, even though the telepointers had a different look compared to the system mouse pointer (see figure 3). One common pattern identified in our qualitative analysis was, however, some trouble and overhead associated with configuration and device bindings. This can also be seen from the quantitative data in figure 9.

6. CONCLUSION

This paper presented the CompUTE software architecture and the ComPUTE .net implementation. In ComPUTE, the Olympus gateway handles device discovery, represents and manages devices and their services, and handles the binding between devices. On each device the CompUTE client manages discovery and device binding via the gateway. Once the bindings between two or more devices have been established, the clients on each device cooperate peer-to-peer.

The ComPUTE .net implementation supports transparent device composition in the Windows XP operating system. The user interface follows the 'Extended Desktop' metaphor,

thereby allowing two devices to be bound together and start working together like an extended desktop across two monitors attached to a single PC. To achieve this, the CompUTE .net client supports (i) control re-direction (mouse and keyboard) and uses remote pointers for representing redirected mouse cursors on a remote device; (ii) application view re-direction where an application's window and subwindows can be moved transparently across the displays of the connected devices; and (iii) distributed data exchange via a distributed clipboard and a file drag-and-drop between connected devices. The CompUTE platform goes beyond the 'Extended Desktop' metaphor by allowing multiple colocated users to work together collaboratively using a device each.

An evaluation of the CompUTE .net platform demonstrated that the platform was very useful, very easy to use, and easy to learn – but configuration of a device composition was less easy to make. These findings points to the major theme in our future work on the CompUTE platform, namely to provider more adequate support for ad-hoc configuration of device composition.

In conclusion, we find that CompUTE presents a platform which in technical as well as in usability terms unifies device composition. Technically because the proposed software architecture and .net implementation takes previous pieces of device composition research, and demonstrate that these pieces can be put together in a coherent platform which is scalable, robust, and extensible with respect to different platforms and implementations. Usabilitywise because the CompUTE .net implementation presented a unified user interface design for device composition which enabled a transparent use of the CompUTE platform for device composition.

7. REFERENCES

[1] J. P. Chin, V. A. Diehl, and K. L. Norman. Development of an instrument measuring user satisfaction of the human-computer interface. In *CHI '88: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 213–218, New York, NY, USA, 1988. ACM Press.

[2] F. D. Davis. Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology. *MIS Quarterly*, 13(4):319–340, Sept. 1989.

[3] W. K. Edwards, M. W. Newman, J. Z. Sedivy, and T. F. Smith. Experiences with recombinant computing: Exploring ad hoc interoperability in evolving digital networks. *ACM Trans. Comput.-Hum. Interact.*, 16(1):1–44, 2009.

[4] J. Grudin. Partitioning digital worlds: focal and peripheral awareness in multiple monitor use. In *CHI '01: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 458–465. ACM Press, 2001.

[5] B. Johanson, G. Hutchins, T. Winograd, and M. Stone. PointRight: experience with flexible input redirection in interactive workspaces. In *UIST '02: Proceedings of the 15th annual ACM symposium on User interface software and technology*, pages 227–234. ACM Press, 2002.

[6] R. C. Miller and B. A. Myers. Synchronizing clipboards of multiple computers. In *UIST '99: Proceedings of the 12th annual ACM symposium on User interface software and technology*, pages 65–66. ACM Press, 1999.

[7] T. Pering, R. Want, B. Rosario, S. Sud, and K. Lyons. Enabling pervasive collaboration with platform composition. *Pervasive Computing*, pages 184–201, 2009.

[8] J. Rekimoto. Pick-and-drop: a direct manipulation technique for multiple computer environments. In *UIST '97: Proceedings of the 10th annual ACM symposium on User interface software and technology*, pages 31–39. ACM Press, 1997.

[9] J. Rekimoto and M. Saitoh. Augmented surfaces: a spatially continuous work space for hybrid computing environments. In *CHI '99: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 378–385, New York, NY, USA, 1999. ACM Press.

[10] T. Richardson, Q. Stafford-Fraser, K. R. Wood, and A. Hopper. Virtual Network Computing. *IEEE Internet Computing*, 2(1):33–38, 1998.

[11] M. Roman, C. Hess, R. Cerqueira, A. Ranganathan, R. Campbell, and K. Nahrstedt. A middleware infrastructure for active spaces. *Pervasive Computing, IEEE*, 1(4):74 – 83, oct-dec 2002.

[12] C. Shen, F. D. Vernier, C. Forlines, and M. Ringel. Diamondspin: an extensible toolkit for around-the-table interaction. In *CHI '04: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 167–174, New York, NY, USA, 2004. ACM.

[13] J. Stewart, B. B. Bederson, and A. Druin. Single display groupware: a model for co-present collaboration. In *CHI '99: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 286–293, New York, NY, USA, 1999. ACM.

[14] D. S. Tan, B. Meyers, and M. Czerwinski. WinCuts: manipulating arbitrary window regions for more effective use of screen space. In *CHI '04: Extended abstracts of the 2004 conference on Human factors and computing systems*, pages 1525–1528. ACM Press, 2004.

[15] P. Tandler, T. Prante, C. Müller-Tomfelde, N. Streitz, and R. Steinmetz. Connectables: dynamic coupling of displays for the flexible creation of shared workspaces. In *UIST '01: Proceedings of the 14th annual ACM symposium on User interface software and technology*, pages 11–20, New York, NY, USA, 2001. ACM.

[16] E. Tse, J. Histon, S. D. Scott, and S. Greenberg. Avoiding interference: how people use spatial separation and partitioning in sdg workspaces. In *CSCW '04: Proceedings of the 2004 ACM conference on Computer supported cooperative work*, pages 252–261, New York, NY, USA, 2004. ACM.