Continuous User Interfaces for Seamless Task Migration

Pardha S. Pyla, Manas Tungare, Jerome Holman, and Manuel A. Pérez-Quiñones

Department of Computer Science
Virginia Tech
Blacksburg, VA, 24060 — USA.
{ppyla@vt.edu, manas@tungare.name, perez@cs.vt.edu,
jeromeholman@msn.com

Abstract. In this paper, we propose the Task Migration framework that provides a vocabulary and constructs to decompose a task into its components, and to examine issues that arise when it is performed using multiple devices. In a world of mobile devices and multiple computing devices, users are often forced to interrupt their tasks, move their data and information back and forth among the various devices manually, recreate the interaction context, and then resume the task on another device. We refer to this break from the task at hand as a *task disconnect*. Our objective is to study how software can bridge this task disconnect, enabling users to seamlessly transition a task across devices using *continuous user interface*. The framework is intended to help designers of interactive systems understand where breaks in task continuity may occur, and to proactively incorporate features and capabilities to mitigate their impact or avoid such Task Disconnects altogether.

1 Introduction and Motivation

Today, with the advent of mobile devices and the deployment of computing in various form factors, the paradigm of a single user interacting with a single computer on a desk is losing its dominance. Even though the massive storage and computational power of a desktop computer has helped it to continue to be a central part of our daily work, most people interact with more than one device for their everyday tasks. From a recent survey of knowledge workers in a huge software development company and a major university [15], almost all participants reported that they use at least two computational devices for their day-to-day activities. The desktop computer and the notebook computer are still two primary devices that people use to accomplish their daily work. This proliferation of multiple computing devices is burdening the user with overheads for transferring information among different devices. Often, users are forced to copy files back and forth, open and close applications, and repeatedly recreate task context as they move from one device to another. In this paper we provide a theoretical foundation to describe the extraneous actions users need to perform as they switch from one device to another and propose the idea of Continuous User Interfaces (CUIs) to facilitate seamless task migration across multiple devices. We describe the development and evaluation of a proof-of-concept continuous user interface and observations from a preliminary usability study.

1.1 Tasks, Activities, Units and Cost

Before formally defining a *task disconnect*, we first describe the vocabulary and terminology for tasks and the various associated parameters.

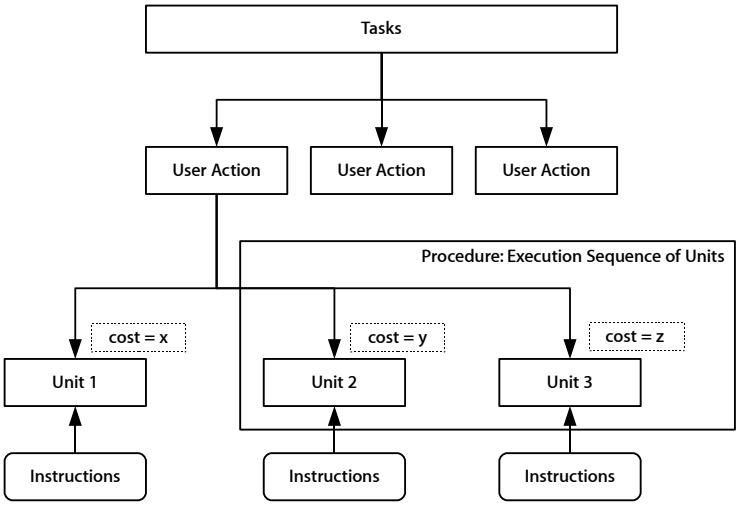


Fig. 1. Tasks, User actions, Units and Instructions

A task can be defined as *a goal to be attained in given conditions* [11]. Leplat expresses these conditions using three points of view: *the states to be covered, the permitted operations, and the procedure* [12]. At a lower level, we define tasks to be user actions. A user action is what the subject puts into operation (cognitive operations, behavior) in order to meet task demands. We also make use of Leplat's definition of *elementary units* to be the *elementary tasks, and elementary states or operations*. Leplat uses these definitions to describe task complexity. However, we use the term *units* to further subdivide user actions to their lowest granularity. For non-trivial tasks (i.e., tasks that involve multiple activities), we define a *procedure* to be an operation execution sequence of multiple units. We also associate with each unit a parameter required for the successful execution: an *instruction*. Instructions are knowledge directions necessary to execute units, that can exist in the user's understanding of the world or it can exist in the aids and artifacts in the task environment.The *cost* of a unit is a multidimensional attribute set that is incurred during the execution of a unit [12]. These dimensions could be cognitive, physical, memory-intensive, resource-intensive or a combination depending on the nature of the unit and the expertise of the user. Another important parameter of a task is time. In the words of Leplat, *every task takes place in time and may be described by the temporal dimensions of its organization* [12]. Out of the few temporal dimensions that Leplat describes, *temporal ruptures* is of particular importance to our work. We adapt and modify Leplat's definition of temporal ruptures to mean interrup-

tions by activities that do not directly contribute to the successful execution of the task at hand.

1.2 Task Disconnects

With the increasing proliferation of mobile and ubiquitous computing devices, people are forced to adapt their daily workflows to try and keep their work products and other data available to themselves at all times. E.g., Jones et al. [10] found that people often emailed to themselves the URLs of the websites they visited or their favorites list when they needed to access it from a different location. In scenarios such as this, where a user attempts to work with two devices, the need to transfer task and information back and forth between them is a burden. Users are forced to use workarounds like USB key drives, remote desktop software, e-mail, network file storage, and other means. These attempts to orchestrate a migration of data back and forth between the two devices create a gap in task execution. When a task is to be migrated from one device to another, the process involves more than just breaking it into two chunks along a selected boundary. It also involves the insertion of extra units into the task procedure that are entirely absent when the same task is executed on a single device. It is the inclusion of these extra units that hinders the seamlessness of task migration. Depending upon the exact nature of the task and the two devices in question, this process may involve simply adding more instructions to the procedure (low additional cost), or may involve an entirely new set of user actions (high additional cost).

A task disconnect is *a temporal task rupture arising due to extraneous user actions required when performing a task using multiple devices. Such extraneous user actions are required to accomplish the task across multiple devices, but do not directly aid in the completion of the task at hand*. This raises issues such as how to help the user make a switch from one task condition to another in such a way that the need for a user's attentional resources, cognitive workload, reaction time, and memory demands are minimized.

2 Related Work

Bellotti and Bly [2] observed information workers to be mobile within the confines of their office; this local mobility existed mainly to enable the use of shared resources and for communication with other staff members. While mobile, users were seen to use secondary computational devices in addition to traditional devices such as desktops. A few strands of research have tried to address the problem of migrating tasks or applications over multiple devices. However, most of these studies have focused primarily on the technological aspects of this problem: Chu et al. [6] take the approach of migrating an entire application to support seamless task roaming, but with considerable latency (thus interrupting the user's tasks sequence.) They do not discuss the implications on the user's tasks and goals. Bandelloni et al. [1] describe user interaction with an application while moving from one device to another, in three levels of migration: total, partial and mixed. Chhatpar and Pérez-Quiñones [5] call this migration *dialogue mobility* and propose a requirement for the application data and logic to be separate from

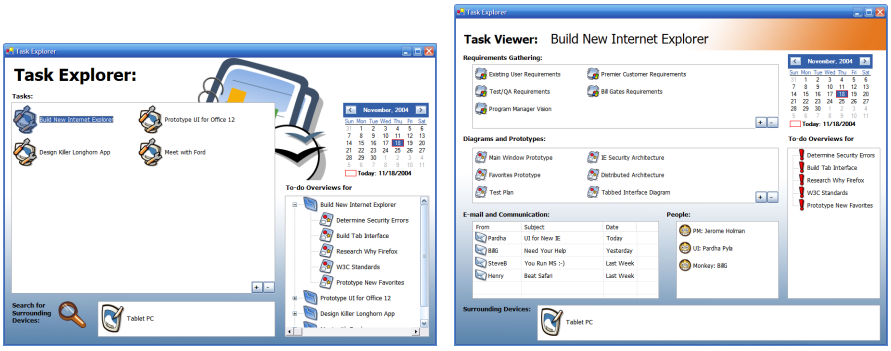
the user interface. Neither one of these projects take the task perspective we propose in this paper.

Florins et al. [9] describe rules and transformations that attempt to provide graceful degradation of user interfaces while an application is migrated from one device to another; even though their work is based on the same principle of continuity, their focus is on user interface generation and not on task migration. ARIS [4] is a window management framework that relocates running applications from one display to another; TERESA [14] helps design and develop model-based nomadic applications. Toolkits and tools such as TERESA have utility in rapidly deploying applications that can be migrated over multiple devices, but do not address the task semantics that users wrestle with while trying to interact with a multi-device interface. Denis and Karsenty [7] discuss a conceptual framework for the inter-usability of multiple devices. They provide an analysis of different cognitive processes in inter-device transitions and postulate two dimensions required for seamless interaction: knowledge continuity and task continuity. We base our work and the definition of Continuous User Interfaces on this requirement of seamlessness. We take this task-centered approach to solving the problem and we provide a definition, description, parameters, requirements, and a prototype to demonstrate a seamless interaction over multiple devices without task disconnects.

Interruptions are events that break the user's attention on a particular task to cater to another task that is in need of attention, and are the focus of a whole area of study by themselves [13]. Task disconnects can also be envisioned as analogous to interruptions, but occurring over multiple devices, locations, contexts, and most importantly, over a much longer time interval. Interruptions happen often in normal conversations [3, 8]. However, in the field of linguistics [8, 3], not all interruptions are disruptive or need repair. Even in cases where the interruptions are disruptive, the costs associated with repair are low, because humans have an inherent ability to repair, recover and proceed with most of the conversations using their ingrained social and cultural aids.

3 Study Design

We targeted a specific application domain with sufficient complexity to allow us to observe clearly the different parameters responsible for task disconnects. Software development is a domain that involves work on several tasks of different affinities for devices. Software engineers perform requirements gathering and early design at client locations, away from their own offices, with laptop computers. They bring the artifacts of this stage to their office and continue to create a complete design using their desktop computers. The software that is designed must finally run on clients' machines. We chose this application domain because of the need to use several tools such as text editors, drawing packages, scheduling programs, etc. when accomplishing a task, and because the nature of the task requires the use of multiple devices. We built a prototype to support the preliminary design phase of software engineering where developers collect client requirements and generate initial design prototypes, diagrams, and models.



(a) Task Explorer (b) Task Viewer

Fig. 2. Task Explorer and Viewer

3.1 Prototype

We built a prototype that incorporated knowledge continuity and task continuity to provide a seamless multi-device user experience. The Task Explorer (Figure 2(a)) allowed users to create and demarcate a task, track the activities they performed in a to-do list tool (included), and provided constant visual feedback on the status of the connected devices in range. Opening a task in the Task Explorer launched the Task Viewer (Figure 2(b)), a place to view documents and files related to a task. In our prototype application domain, these are requirements documents, diagrams and prototypes, e-mail addresses, and people related to the project in a unified view. Each task is uniquely color-coded to establish a visual identity with the task. Opening a document such as a requirements specification launched that file in an external editor.

This was implemented as an application on a tablet interface. The interface leveraged spatial organization, shape, color, partitioning of data and function, recovery of state of data and recovery of activity context on its user interface. Tasks were migrated from the desktop computer to the tablet interface either automatically (because the task was open) or manually (by dragging and dropping). For each task, we displayed the task parameters in the same color gradient on the tablet and the desktop. The last drawing that was being accessed on the desktop computer was automatically loaded to maintain task continuity and activity context. If the drawing was cleared and replaced by another, the new diagram was synchronized automatically with the desktop. This obviated the need to open and save documents, making the interface more like paper. As artifacts were being generated, they were populated into the task tree on the right side of the screen. The task tree on the tablet brought together artifacts related to the task on the desktop computer, e.g. requirements documents, people, to-do list, and email messages.

3.2 Evaluation

Interviews and user surveys were conducted to gather insights into the example task of prototyping and the existence of disconnects when using multiple devices to proto-

type. Six professional software developers were asked open-ended questions targeting the technologies and devices they used to prototype and any insights into disconnects arising due to the mediation by these technologies. In addition, we received N=32 responses to a survey that targeted software developers, graduate students with software development experience, and researchers in HCI who were familiar with computing and did prototyping tasks.

The prototype was evaluated with a group of graduate students with a software engineering background. A total of six participants participated in the evaluation. Three of the six participants constituted a control group where they were given tasks that required switching between a tablet and a desktop computer. The other three participants comprised our test group and were asked to perform the same tasks using our prototype. Each participant was assigned a total of seven tasks. Each task required drawing simple low-fidelity user interface prototypes using our custom drawing tool, updating requirements specifications using a text editor, or a combination of these two tasks. Participants were provided a background scenario explaining the context of a software development project for a fictitious client and the need to transfer documents between the tablet and the desktop. The participants were asked to use a tablet when meeting the client. Their interaction with the client was scripted in the scenario provided. The participants were asked to think aloud while they worked and the evaluator prompted the users when they stopped talking during a task.

3.3 Tasks

The first task required the participant to make changes to an existing requirements document based on a fictitious client's new insights into the project at the client's location (i.e. using a tablet). The second task required the participant to prepare a low-fidelity prototype for the new requirements specification on the desktop. The third task asked the participant to visit the client to demo the prototype that was created on the desktop at the participant's office. The fourth task required the participant to work on the desktop and to add more description to some requirements based on the client's feedback. The participants were asked to assume they were at home for the fifth task (i.e. they were to use a tablet.) When they thought of a design feature, they were to create a new prototype with that insight to demo to the client the next day. The sixth task asked the participant to visit the client and demo the new prototype and get feedback. Based on the feedback, they were required to change the prototype and the requirement specification. The last task was set at the participant's office where they were asked to update their desktop files with the latest prototype and requirements specifications.

These tasks were designed with the goal of making the participants transfer information between the two devices as they progressed through the tasks. In the test group, this transfer was automatic because the participants used our prototype. In the control group, the participants had to move the files themselves using their choice of a USB pen drive, email, or other server-based technologies. The control group participants were provided the tablet and the desktop that were both connected to the Internet. They were given one task at a time with the associated scenario to provide the context of the interaction. At the end of the session, all participants were asked to fill out a questionnaire.

4 Results and Discussion

In this study, we found several aspects of multi-device interaction that current systems do not adequately support. Specifically, several users reported dropping the use of multiple devices in favor of a single computer, to avoid the costs of task migration. They also reported that migrating files across devices taxes their short-term memory, is often frustrating, and likely to result in errors. We examine each of these in turn, based on our observations and responses from our study participants.

4.1 Use of a Single Computer

Another interesting observation that one participant made was: *"this [migrating data] almost makes me use the tablet alone for all the tasks and forget about my desktop if I had the choice"*. When asked if she would do that even if the task at hand required more processing power (such as that available in a desktop), she responded affirmatively. Several survey respondents in another study [15] also confirmed that they chose to use only a single computer for convenience rather than have to copy/move files among several computers.

This illustrates that the high costs of user actions associated with a task switch from one device to another prompt users to forgo computational power in favor of eliminating the switch entirely.

4.2 Consistency (or the Lack Thereof) of File Locations

One common complaint from participants was that they needed to remember file locations and the state of a file on each device. As one participant put it, *"this version control is getting irritating"*. Remembering such extraneous information increases the short-term memory costs of this activity tremendously. Given that short-term memory is unreliable, it is difficult for the user to remember which device had the latest version of the data if temporal ruptures in a task take place over a longer period of time. This is another observation that directly supports our hypothesis that transferring activity context is important.

The experimental group, who performed the same tasks using our prototype, were instantly able to locate documents and the information they needed. When switching from one device to the other, they reported being able to restart their task immediately and to be productive because the environment was already automatically loaded with the appropriate task context. Because information was re-displayed using less screen real estate, users were immediately able to focus on their work while keeping related information in their peripheral vision. The only limitation of the system was that users spent time moving and resizing the requirements window to enable them to easily see both and work between them.

The act of copying files manually involves two steps: copying the data over, and placing it in the correct location on disk. Most current data copying tools and media (e.g. USB drives, email to self, etc.) assist in performing the first task, but not in the second. Thus, to perform the second step, users are forced to rely on their short-term memory, increasing cognitive workload and scope for error. Automatic system support for the

second step therefore was viewed as a distinct advantage of our prototype. The related issues of version control and conflict management were also automatically handled.

4.3 Fear, Frustration

In the questionnaire, all three control group participants reported a fear of making errors due to the overheads associated with the migration of information across devices. They also reported difficulty in keeping track of document version information. One participant commented that if a scenario such as the one in the evaluation were to occur in real life, the costs would be higher because of longer temporal ruptures. One of the participants forgot to copy the files from the desktop to the tablet before visiting the client. When she realized this, she remarked, *"Wow! In real life, this would mean I'd have to go back to my office to get my updated files or redo the prototype that I did in the last task."*.

On the questionnaire, members of the experimental group reported that they were less likely to make errors accomplishing the tasks. Also, because file state and application state were transferred automatically, the experimental group only had to worry about finding the appropriate location in the UI to begin work again. There were comments by some users that it would be nice to have a better view of all the files related to a project, but creating a new file system view was not the purpose of our prototype. Overall, participants of the experimental group responded that the application was more satisfying and easier to use (per Likert scale ratings.)

This means that as the task procedure lengthens (in light of the extraneous actions required for task switching), and associated increase in costs, there is a corresponding rise in the likelihood of user error. In continuous user interfaces, such costs are reduced because of in-built system support for task migration.

4.4 Use of Mobile Computers as Secondary Displays

For the second task, where the participants were required to create prototypes based on the requirements specification document, all the three participants in the control group preferred using the tablet as an information display. They opened the specification document on the tablet and referred to it as they sketched the prototype on the desktop. When asked about this, they said that having the information on a secondary display was good as it did not make them switch between different windows on one device. This might mean that CUIs should leverage the capabilities of the various devices even when they are co-located.

5 Discussion and Summary

We explored the issues that arise when users use multiple devices to execute a single task. We found that current systems lend inadequate support for several user actions that must be performed during a task migration between/among devices. Among the problems reported were: dropping the use of multiple devices in favor of a single computer; increased short-term memory costs while migrating files across devices; higher frustration; and a higher likelihood of errors.

We proposed and designed a prototype Continuous User Interface that ensured a seamless task migration for users attempting to perform a requirements specification and gathering task, using a tablet computer and a desktop computer. This system provided support for automatic migration of task context (e.g. the applications that were in use; pages and objects such as diagrams that were selected and active; etc.) between the two devices. In an evaluation conducted, participants reported that it helped mitigate the disruptive effects of task disconnects to a high degree. An interesting observation was that users expected to be able to annex existing collocated devices when performing their tasks (i.e., using the tablet computer at their desk along with their primary desktop computer.) They also reported that the automatic availability of necessary data on mobile computers directly contributed to higher perceived reliability and lower likelihood of error.

References

1. R. Bandelloni and F. Paternò. Flexible Interface Migration. In *IUI '04: Proceedings of the 9th international conference on Intelligent user interface*, pages 148–155, New York, NY, USA, 2004. ACM Press.
2. V. Bellotti and S. Bly. Walking away from the desktop computer: distributed collaboration and mobility in a product design team. In *CSCW '96: Proceedings of the 1996 ACM conference on Computer supported cooperative work*, pages 209–218, New York, NY, USA, 1996. ACM Press.
3. A. Bennett. Interruptions and the interpretation of conversation. *Discourse Processes*, 4(2):171–188, 1981.
4. J. Biehl and B. Bailey. Aris: An interface for application relocation in an interactive space. In *Proc. 2004 conference on graphics interface*, pages 107–116, 2004.
5. C. Chhatpar and M. Pérez-Quiñones. Dialogue mobility across devices. In *ACM Southeast Conference (ACMSE)*, Savannah, Georgia, 2003.
6. H.-h. Chu, H. Song, C. Wong, S. Kurakake, and M. Katagiri. Roam, a seamless application framework. *Journal of Systems and Software*, 69(3):209–226, 2004.
7. C. Denis and L. Karsenty. Inter-usability of multi-device systems - a conceptual framework. In A. Seffah and H. Javahery, editors, *Multiple User Interfaces: Cross-Platform Applications and Context-Aware Interfaces*, pages 373–384. John Wiley and Sons, 2004.
8. K. Drummond. A backward glance at interruptions. *Western Journal of Speech Communication*, 53(2):150–166, 1989.
9. M. Florins and J. Vanderdonckt. Graceful degradation of user interfaces as a design method for multiplatform systems. In *IUI '04: Proceedings of the 9th international conference on Intelligent user interface*, pages 140–147, New York, NY, USA, 2004. ACM Press.
10. W. Jones, H. Bruce, and S. Dumais. Keeping found things found on the web. In *CIKM '01: Proceedings of the tenth international conference on Information and knowledge management*, pages 119–126, New York, NY, USA, 2001. ACM Press.
11. A. Leontiev. *Le Developpement du Psychisme*. Editions Sociales, Paris, France, 1972.
12. J. Leplat. *Tasks, Errors and Mental Models*, chapter Task complexity in work situations, pages 105–115. Taylor & Francis, Inc., Philadelphia, 1988.
13. D. C. McFarlane. *Interruption of people in human-computer interaction*. Doctoral dissertation, The George Washington University, 1998.
14. G. Mori, F. Paternò, and C. Santoro. Tool support for designing nomadic applications. In *IUI '03: Proceedings of the 8th international conference on Intelligent user interfaces*, pages 141–148, New York, NY, USA, 2003. ACM Press.

15. M. Tungare and M. Pérez-Quiñones. It's not what you have, but how you use it: Compromises in mobile device use. Technical report, Computing Research Repository (CoRR), 2008.