Adapting Existing Applications to Support New Interaction Technologies: Technical and Usability Issues

Darren Andreychuk, Yaser Ghanam, Frank Maurer
Department of Computer Science
University of Calgary
Calgary, AB, Canada T2N 1N4
{yghanam, djandrey, fmaurer}@ucalgary.ca

ABSTRACT

Engineering interactive systems for use on emerging technologies such as touch-enabled devices and horizontal displays is not straightforward. Firstly, the migration process of a system from an old hardware platform to new multi-touch displays is challenging. Issues pertaining to scaling, orientation, new input mechanisms, novel interaction techniques and different SDKs need to be examined. Secondly, even after we manage to understand and resolve these issues, we need to find effective ways to migrate applications and maintain them.

This paper contributes a thorough analysis of the technical and usability issues that need to be considered when migrating systems to different touch-enabled technologies including vertical and horizontal displays.

Keywords

Adaptation, Multi-touch, Variation

INTRODUCTION

As hardware vendors continue to produce novel technologies such as touch-enabled PCs [6] and digital tabletops [8, 11], efforts are being made by researchers and practitioners to utilize these technologies in improving the usability and usefulness of existing software systems. This issue is becoming more common as new technologies appear in the marketplace and it is increasingly important for practitioners looking to provide support for them. However, taking a system that was originally built for a normal PC and simply deploying it on a new hardware platform such as a digital tabletop is limiting the usefulness of the device. This is because we first need to understand the implications of the change in hardware capabilities and how it might hinder/improve the migrated system. Furthermore, the traditional top-down system design approach does not fit the more dynamic nature of interactive surfaces since new products continually appear at different times in the marketplace.

Software engineers can interface with the new hardware capabilities using the software development kits (SDKs) provided by the vendor – and each vendor currently provides their own SDK (which is incompatible with the SDKs from other vendors).[1] New challenges arise due to the constraints these SDKs have, and also the level of abstraction they operate at. For example, if the APIs provided are too fine-grained, it might be necessary to introduce a level of abstraction that makes reuse of meaningful artifacts possible (e.g. rotatable and translatable object). On the other hand, if the APIs are too abstract, they can limit access to data that is deemed necessary in certain applications (e.g. the angle of the touch point with the surface). In this case, practitioners need to find workarounds and incorporate them in a systematic way. Furthermore, different vendors use different feature sets for input/output mechanisms. This makes deploying a software system on different hardware brands challenging.

The other important aspect is usability. When developing applications for touch-enabled PCs and horizontal displays, the type of input expected from the user and the way the user interacts with the surface are not the same as their counterparts on old-fashioned platforms. That is, what is highly usable on a vertical screen can be of a very poor usability on a horizontal one and vice versa. This implies that it is not trivial to migrate systems that were originally built to target vertical screens to machines that utilize horizontal displays as a front end [10, 13]. While the underlying functionality of the system is probably the same in both environments, interaction and presentation need to be retailored to suit the newer technology. If we were to deploy a customized solution on each platform, whether vertical or horizontal, we will also need to maintain these solutions. And as the number of supported platforms increases, maintaining different solutions has to be efficient to be economical. Therefore, applying the proper software engineering techniques to solve this issue is imperative. The question we answer in this paper is: How can we deploy customized solutions on different touch-enabled technologies including vertical and horizontal displays and maintain these solutions efficiently?

[1] Windows 7 provide options for vendors to support multi-touch with their hardware in a standardized way. How many vendors will use it is not clear at this point in time.

To answer this question, we first define two factors:

1. Technical issues: evident in the SDK definition and abstraction, as well as the various hardware platforms to be supported.
2. Usability issues: often resulting from migrating an application from vertical to horizontal surfaces.

The rest of the paper is organized as follows. First we talk about related work. Next we discuss the specific context of our experience. We present our approach to solve the problem at hand. Next we discuss the implications of the reported experience for practitioners. Finally, we highlight our conclusions.

RELATED WORK

Migrating applications that were originally developed for vertical displays over to horizontal ones is becoming common. These applications span over a wide range of domains such as education [7], meetings [5], arts [12], programming [4], games [1] and many others. DigiTile [10] and AgilePlanner [13], for instance, are applications that started out on a vertical surface. They were then migrated to horizontal surfaces and a number of issues were reported such as the tabletop size, orientation and the user group size. Challenges like multiple collaborators working at the same time and standard GUI components being unsuitable for the new environment were observed and addressed. Other applications also exist that intend to support a hybrid of vertical and horizontal displays such as WeSpace [5]. Besacier [2] suggested a generic technique to support the use of legacy applications with innovative interaction systems by rewriting the user interface toolkit. Other efforts such as [9] tackled the same issue from a different perspective by using the accessibility API to adapt the user interface to new interaction techniques. Our work is different in that we are interested in all aspects of variability across different interactive systems, and we try to achieve high efficiency in the migration process as well as deployment and maintenance.

EXPERIENCE CONTEXT

System Overview

The application we will discuss throughout this paper is called eHome. It is a software system to monitor and control smart homes. Generally, the interface of the application consists of a floor plan representing the smart environment to be controlled, a number of items that can be dragged and dropped on the floor plan, and a set of graphical user interface (GUI) controls. Interacting with eHome occurs in User Mode where dwellers can view and modify the current status of lamp devices, track items in containers using RFID technology and obtain climate information in the house, and Design Mode where dwellers can register new lamp devices, containers and sensors in the system.

Initial Development

An industrial partner we have been working with for the past two years requested all of the abovementioned features. The initial request was to deploy eHome on an HP TouchSmart PC [6] which has a single-touch vertical display. However, actual development of eHome was done on normal PCs with different screen dimensions and no touch capabilities. When we deployed eHome on the HP machine (which happened frequently because we had a testing HP PC onsite), we often needed to adjust certain scaling factors to fit the HP wide screen. We also realized that some decisions that had been made during development on the normal PCs needed to be revisited. Examples are:

- The size and design of some GUI elements made it challenging to interact with eHome using a finger touch because the latter is much thicker and less accurate than a mouse pointer.
- One event in eHome was triggered by a right-click which, on a touch-screen, did not make sense.

New Technologies

As we went along, we wanted to deploy eHome on a large-scale SMART DViT Table [3] with an older version of the SMART SDK. A later request from our partner was to deploy eHome on a digital tabletop they had recently purchased. Specifically, it was the New SMART Table [11] which supported multi-touch input and had a newer version of the SMART SDK. Later on, we obtained a Microsoft Surface [8] and we decided to include it within the hardware platforms that we should support. As more platforms were supported, more decisions were revisited and the software design underwent drastic yet incremental changes. These changes were mainly driven by the two factors we mentioned in Section 1: technical issues and usability issues. Examples of such issues include:

- Three different SDKs that dealt with touch point input, one for each hardware platform.
- Conventional GUI elements like menus and tabs assumed a single orientation (vertical).

Sources of Variability in eHome

The technical and usability issues were not the only sources of variability in eHome. In fact, the first source of variability was business-driven. Smart homes vary widely with regards to what smart devices exist in the home, and what kind of monitoring and controlling is requested by a given customer. This variation in requirements often results in delivering a different

application for each smart home. However, in spite of the differences between these applications, they share a lot of underlying functionality and business logic.

In the discussion to follow, each section talks about one variability aspect. For each aspect, we analyze the issues we encountered and their implications on our system, and then we describe our approach to contain them.

Variability within Vertical Displays

By vertical displays, we refer to the normal PCs that were used by developers to develop eHome as well as the HP TouchSmart PC on which eHome was initially deployed. The differences between these two groups were issues related to the mouse-versus-touch input. Table 1 describes these issues and their implications.

Table 1 – Issues leading to variability between a normal PC and an HP TouchSmart PC

Issue	Implication
Right-click events do not make sense on a touch screen.	An alternate way (provided by the HP machine) to capture the right-click event on the touch screen was 'press-&-hold'.
The tip of the mouse cursor is tiny and accurate compared to the tip of a finger.	All GUI objects have to be larger to accommodate the finger touch more precisely.
When applying a touch on the vertical surface, the body of the finger covers some content on the screen (Figure 2a).	A vertical slider that was used to control the intensity of a light was changed into a horizontal slider (Figure 2b).

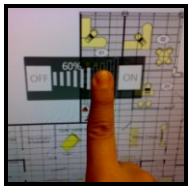
(a)  (b) 

Figure 2 – (a) part of the vertical slider is blocked by the body of the finger. (b) the horizontal slider solves this issue.

As mentioned earlier, the development for normal PCs and HP TouchSmart PCs was the initial stage in the evolution of eHome. At this stage the Presentation layer included all the view-related elements, whereas the UI Controller managed the communication between the Presentation layer and the Data Object Model. The Hardware Controller was responsible for communication between the actual hardware devices with the Model or the UI Controller. External Resources included the hardware devices, XML configuration files, and web services.

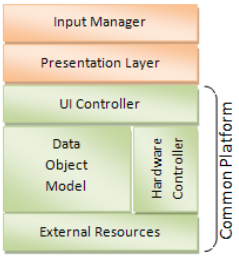


Figure 3 – eHome architecture after considering variability at the Presentation layer.

At first when we only considered the first issue (right-click vs. press-&-hold) as a source of variability, a conceptual layer was added to reflect this variability as shown in Figure 3 (previously, input was managed within the Presentation layer). The common platform included everything but the Input Manager where variability occurred. One variation point (source of variation) was defined as "input mechanism" with the two variants (instances) "mouse" and "touch." Later, when the other two issues were to be managed, variability penetrated down to the Presentation layer as shown in Figure 3. That is, the variability profile we had so far could be described as:

- ***InputMechanism = {mouse, touch}***
- ***InputMechanism = {mouse, single touch, multi-touch}***
- ***Layout = {Normal PC, TouchSmart PC, Digital Table}***

Variability between Vertical & Horizontal Displays

To migrate eHome from a vertical surface to a horizontal one, we initially deployed eHome on a horizontal display without any modification to understand the differences. After a number of usability observations, and going back and forth between the vertical and horizontal settings, we realized a raft of issues. Table 2 lists these issues and their implications on the migration process. In this paper, we do not argue that these implications improved usability as this is yet to be appraised. The point, however, is that usability issues introduced new sources of variability. We realized new variability occurring at the same two layers of the architecture. Not only did we have to go back and modify the variability we had previously defined in the Input Manager, but we also needed to explicate more variability in the Presentation layer. All the other layers were left intact.

Table 2 – Issues leading to variability between vertical and horizontal displays

Issue	Implication
Horizontal displays are, typically, physically larger than vertical ones.	A new scaling adjustment factor is defined for UI objects to make them bigger, and hence easier to interact with, on larger displays.
Horizontal displays deal with multiple concurrent touch points not only single touch points or mouse clicks.	This new input mechanism needs to be incorporated into the Input Manager layer as a new variant.
Conventional GUI elements like buttons, menus and tabs were oriented in a top-down fashion, which for a horizontal surface did not seem natural because people sit on different sides of the table.	The conventional GUI elements were replaced by panels available on each of the four sides of the tabletop, in Figure 4. Instead of one Exit button on the top left corner of the screen, an Exit button was added on each corner of the tabletop. The "change mode" button (user/designer) was removed. Instead, the change of mode on the digital tabletop happens automatically.
Feedback to the user was provided using a status bar at the bottom of the screen, which was not suitable for a multi-oriented surface (i.e. horizontal display).	Alternative ways to provide feedback were used. For example, when a certain operation executes successfully, the corresponding icon on the surface glows.
When using a slider control, vertical and horizontal sliders seemed counterintuitive if there were people sitting around the table (e.g. moving a vertical slider up means a person on the other end sees it moving down).	A circular slider was used with clearly flagged ON/OFF positions, as shown in Figure 5. Regardless of where you sit around the table, if the handle of the slider is moving towards the ON button, then the intensity is increasing and vice versa.
Some features were not readily easy to use for everybody around the table because the UI controls were closer to a certain part of the screen.	For deleting an object, instead of a single trash can on the bottom right corner of the screen, the user has the option to drag it to any of the trash cans distributed on the corners of the screen.
Readability of text on the horizontal display was limited because of the presumed top-down orientation.	The horizontal interface includes far less text than the vertical one. Descriptive icons and UI controls, animations, as well as visual cues like pulsation or glowing are used to replace text.
Horizontal displays with multi-touch capabilities provided new interactions not possible on PC displays.	On horizontal displays, it was made possible to zoom in and out of the floor plan using two finger touches.
On a big scale tabletop, drag-and-drop became difficult due to the physical limitations on the reach of an arm.	Gestures were made available as additional (not substitutional) ways of executing certain tasks. (e.g. scratch-out to delete object)

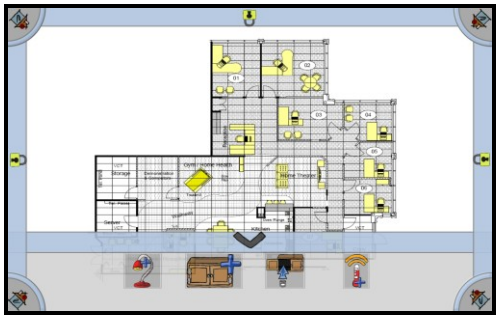


Figure 4 – eHome on a horizontal display has redundant GUI elements to support multiple orientations.

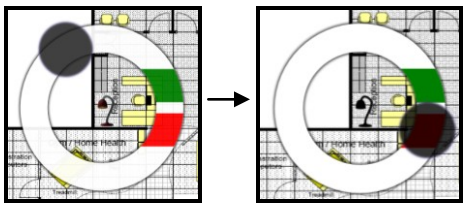


Figure 5 – Circular slider to control light intensity

Variability within Horizontal Displays

In this section we will discuss variability among different horizontal displays. By horizontal displays, we namely refer to three hardware platforms: SMART DViT Table, SMART Table, and Microsoft Surface. We dealt with three different SDKs, two of which were different versions from the same vendor.

The first tabletop eHome was deployed on was the SMART DViT table. We utilized the dual-touch capability of this table by adding a feature that allowed the user to place two touch points on the floor plan in order to zoom in and out. This kind of interaction required the hardware platform to support at least two simultaneous touches.

A specialized controller was introduced in the UI Controller layer to manage all communication between eHome and the touch handlers in the SMART SDK, as shown in Figure 6 – A. By this separation, it was easier to plug this feature in and out. The new controller was

responsible for managing three events, namely: TouchDown, TouchUp and TouchMove. In case the touch events were part of a zooming interaction, the specialized controller will handle the zooming. Otherwise, the touch events were rerouted to mouse events we had previously defined in the UI Controller for the previous platforms in order to maximize code reuse and avoid code redundancy.

The second step was deploying eHome on a SMART Table which uses FTIR technology that supports forty concurrent touches. A new specialized hardware controller was also created to manage communication between eHome and the touch handlers in the new SMART SDK. At this stage, we had two different controllers one for each table. These controllers, however, shared common aspects such as the main triggering events and the zooming interaction. These common aspects were abstracted in a new layer we called "Multi-Touch Library" as shown in Figure 6 – B. The new layer was abstracted in a way so that it was completely agnostic to the target hardware platform – all specificities were kept in the specialized controllers.

Later, this abstraction served well in accommodating the new digital tabletop – Microsoft Surface. That is, it only took about one day worth of work to deploy eHome on the MS Surface, because all we needed to do was create a new specialized controller to communicate with the Surface SDK, while all other aspects were managed by the Multi-Touch Library. Figure 6 – C shows the final organization.

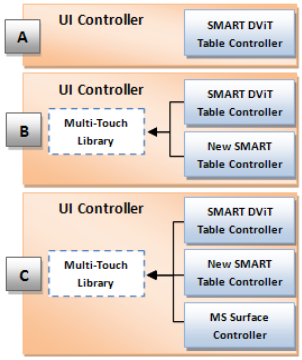


Figure 6 – Handling variability due to SDK differences

As was done before, variability was evolved to include a new layer, namely the UI Controller layer. This variation point was added to the variability profile:

Multi-Touch SDK = {SMART DViT Table, New SMART Table, MS Surface}

IMPLICATIONS FOR PRACTITIONERS

In this section, we discuss the practical implications of our approach, and outline some of the lessons learnt.

Whether the intent is to migrate an existing application from a normal PC to a touch-enabled device or from a vertical display to a horizontal display, one should be able to accommodate the different requirements and capabilities of the increasing number of hardware platforms in the market. There are two alternatives to deal with this variety. One is to branch different versions of the application and maintain them separately. However, differences in the display orientation or in the hosting hardware platform occur at specific layers in the architecture and can be managed in a more effective way. This leads us to the second alternative which is embracing this variability in a single product, and then instantiate products as needed following the approach suggested in this paper.

To generalize this approach to other systems, one should consider a raft of issues, namely:

Reuse code and other artifacts. In the case of eHome, about 60% of the code (production and testing) is reused amongst all platforms. This figure could even be higher for systems that have a thinner presentation layer than the one in eHome. Maximizing reuse is desirable because it lessens the time and effort to produce new products and maintain existing ones. For instance, if the underlying technology for a certain feature (e.g. item tracking) changes, we need to make the proper modification in the common platform only once. If a vendor produced a new digital tabletop, all the work we need to do is at the UI Controller layer. The common platform can be used without changes.

Realize the power of combinations. One more advantage of the systematic treatment of variability is the ability to combine different variants to come up with diverse products. For example, suppose we want to support the new HP TouchSmart PC that enables *two* simultaneous touches. We can come up with a new combination of variants to add the zooming behavior.

Minimize speculation. Do it bottom-up. When developing applications for digital tabletops, we are dealing with a new and fast-changing technology, which makes the risk of wrong predictions very high. Therefore, instead of investing much time in domain analysis & design upfront, one can dedicate initial efforts on actual development of single products. Here, we distinguish two cases:

Case 1. If the new platform to be supported is the first tabletop platform (i.e. migration from vertical to horizontal), then some *refactoring* will likely be necessary to make a clear separation between what could and could not be migrated. In our case, we could migrate everything but parts of the UI. It is advisable to maximize reuse even in the UI layers. Nevertheless, when migrating to a horizontal display, many of the usability assumptions need to be revisited, which requires a different mindset in the development process that makes it alright to redo things for the sake of better usability rather than reuse existing UIs.

Case 2. If the new platform is another digital tabletop platform, then minimum work should be done on the UI side. More work, however, is needed to *abstract* common behaviors, interactions, and scaling issues as needed. This iterative abstraction will make it easier to support new platforms in the future. Sometimes, our increased knowledge of what stumbling blocks to expect, and the learning we gained when building previous applications make it more tempting to build a new application for the new platform. However, the disadvantage of doing so is that then we will need to maintain a different base code for each application which is not practical or/and economical.

In both cases, a safety net of regression tests should be provided to observe the effects of refactoring and abstraction. In our case, eHome had an automated testing coverage as high as 90% of the model code and a suite of UI regression tests to be conducted manually.

CONCLUSION

Highly interactive technologies such as digital tabletops are imposing new standards of user interface design and interaction techniques. They also come with new technical constraints that make adapting existing software systems a challenging process.

This paper contributes a thorough analysis of the technical and usability issues that need to be considered when migrating systems to different technologies. We show that an iterative bottom-up adaptive approach is possible. The sources of variability were found to be mainly due to technical limitations arising from the different technologies behind the displays, and usability issues mainly due to the migration from vertical to horizontal displays. We believe this analysis is of significant interest to practitioners who deal with new interaction systems and migration issues.

Currently we are working on introducing a new test-based configuration layer to enable semi-auto-generation of products based on test scenarios.

REFERENCES

1. Al Mahmud, A., Mubin, O., Renny Octavia, J., Shahid, S., LeeChin Yeo, Markopoulos, P., Martens, J.-B., Aliakseyeu, D., "Affective Tabletop Game: A New Gaming Experience for Children," *2nd Annual IEEE International Workshop on Horizontal Interactive Human-Computer Systems, TABLETOP 2007*, pp. 44-51.
2. Besacier, G. and Vernier, F. Toward user interface virtualization: legacy applications and innovative interaction systems. *Proceedings of the 1st ACM SIGCHI Symposium on Engineering interactive Computing Systems,* Pittsburgh, 2009, pp.157-166.
3. DViT Technology, available at: http://smarttech.com/DViT, last accessed June 18, 2009.
4. Gallardo, D.; Julia, C.F.; Jorda, S., "TurTan: A tangible programming language for creative exploration," *3rd IEEE International Workshop on Horizontal Interactive Human Computer Systems, TABLETOP 2008*, pp.89-92.
5. Hao Jiang; Wigdor, D.; Forlines, C.; Chia Shen, "System design for the WeSpace: Linking personal devices to a table-centered multi-user, multi-surface environment," *3rd IEEE International Workshop on Horizontal Interactive Human Computer Systems, TABLETOP 2008*, pp.97-104.
6. HP TouchSmart IQ770 PC datasheet, available at: http://www.hp.com/hpinfo/newsroom/press_kits/2007/ces/ds_pc_touchsmart.pdf, last accessed June 18, 2009.
7. Mansor, E.I.; De Angeli, A.; De Bruijn, O., "Little fingers on the tabletop: A usability evaluation in the kindergarten," *3rd IEEE International Workshop on Horizontal Interactive Human Computer Systems, TABLETOP 2008*, pp.93-96.
8. Microsoft Surface datasheet, available at: www.microsoft.com/surface, last accessed June 12, 2009.
9. Parente, P. and Clippingdale, B. Linux screen reader: extensible assistive technology. *Proceedings of the 8th international ACM SIGACCESS Conference on Computers and Accessibility*, Portland, 2006, pp. 261-262.
10. Rick, J.; Rogers, Y., "From DigiQuilt to DigiTile: Adapting educational technology to a multi-touch table," *3rd IEEE International Workshop on Horizontal Interactive Human Computer Systems, TABLETOP 2008*, pp.73-80.
11. SMART Table datasheet, available at: www2.smarttech.com/st/en-US/Products/SMART+Table, last accessed June 12, 2009.
12. Vandoren, P.; Van Laerhoven, T.; Claesen, L.; Taelman, J.; Raymaekers, C.; Van Reeth, F., "IntuPaint: Bridging the gap between physical and digital painting," *3rd IEEE International Workshop on Horizontal Interactive Human Computer Systems, TABLETOP 2008*, pp.65-72.
13. Wang, X.; Ghanam, Y.; Maurer, F, "From Desktop to Tabletop: Migrating the User Interface of Agile Planner," *Engineering Interactive Systems 2008 - The 2nd Conference on Human Centered Software Engineering,* pp. 263-267.