# Intelligent Working Environments for the Ambient Classroom

Maria Korozi[1], Stavroula Ntoa[1], Margherita Antona[1], and Constantine Stephanidis[1,2]

[1] Foundation for Research and Technology – Hellas (FORTH),
Institute of Computer Science, GR-70013 Heraklion, Crete, Greece
cs@ics.forth.gr
[2] University of Crete, Department of Computer Science

**Abstract.** This paper introduces a suite of Window Managers purposed for the technologically enhanced classroom. The overall objective is to instantiate a common look and feel across various classroom artifacts, thus providing a unified working environment for the students and teachers. To achieve optimal interaction and application display, the workspaces for each artifact are designed keeping in mind both the platform's characteristics and the user's requirements. The usability evaluation of the developed system is reported.

**Keywords:** window managers, education, smart classroom, pervasive user interfaces.

## 1 Introduction

The benefits of using Information and Communication Technologies (ICTs) in education have been in the foreground for several years, as they are acknowledged to be potentially powerful tools for educational change and reform. The students benefit particularly from the use of ICTs, since the access to educational information is unlimited, the learning environment is enriched, active learning and collaboration is promoted, and motivation to learn is enhanced. In the recent past, learning with the use of ICT was strongly related to concepts such as distance learning, educational games, intelligent tutoring systems and e-learning applications.

As a result, the notion of smart classrooms became prevalent in the past decade [9] and has drawn the attention of many researchers. Many have envisioned a smart classroom that consists of technologically enhanced artifacts purposed to serve teachers' and students' needs. Such artifacts could replace the traditional student and teacher desks as well as the classroom boards, in order to support the processes of disseminating and receiving knowledge.

The diversity of characteristics among the potential classroom artifacts outlines the need for a unified work environment for both students and teachers, and points out the importance of optimal interaction with the available applications that may travel among the various artifacts (i.e., desk, board).

To address the aforementioned issues, this paper introduces the classroom Window Managers (cWMs) that host educational applications and facilitate learning activities.

C. Stephanidis (Ed.): Universal Access in HCI, Part I, HCII 2011, LNCS 6765, pp. 381–390, 2011.

A common look and feel is instantiated across the various classroom artifacts, thus transforming the classroom into a unified environment rather than a group of isolated units. In cooperation with the PUPIL system [4], which is a framework that includes among others a library of adaptive educational-oriented components, the cWMs instruct the applications to adapt accordingly to the needs of the targeted classroom artifact. Therefore, the developers are disengaged from having to build different versions of each application and the use of cWMs ensures their portability.

The next sections present some related work in this domain, outline characteristics of the proposed Window Managers, describe the heuristic evaluation performed and finally summarize with conclusions and future work.

## 2 Related Work

During the past decade, research focused on augmenting the classroom environment by assisting the teaching process, recording lecture activity, enabling collaboration among students, supporting remote student telepresence, and encouraging active student's participation and contribution in the learning process.

The system reported in [3] aims to improve the teaching experience by offering automated device control and creating a suitable environment for each detected situation. For example, when an instructor logs on to the classroom computer, the system infers that a computer-based lecture will be given, automatically turns off the lights, lowers the screen, turns on the projector, and switches the projector to computer input. Similarly, Y. Shi et al. in [8] try to facilitate the lecturers by providing alternatives for command execution that do not require being in direct contact with the computer. In this approach, the teacher can use a traditional laser pen to indicate and circle a point of interest, while a camera detects the user's intention and sends a mouse click message to the computer.

In order to endorse students' collaboration and active participation in the learning process H. Breuer et al. [2] developed a platform to support the programming of distributed applications running on different platforms. In their approach, applications running on different platforms and sharing an object will have automatically the status of the object synchronized. An interactive whiteboard application is offered, named "DeepBoard", that implements a gesture-based interaction paradigm and a hierarchical semantic to store and retrieve data, created on the fly, by free-hand writing. Furthermore, collaboration among students is also supported in the Smart Classroom proposed by S. Yau et al. [10], in which each student owns a situation-aware PDA. Students' PDAs dynamically form mobile ad hoc networks for group meetings. Each PDA monitors its situation (locations of PDAs, noise, light, and mobility) and uses the related information to trigger communication activities among the students and the instructor for group discussion and automatic distribution of presentation materials.

The abovementioned approaches envision the "smart" classroom as an environment that assists the instructor and enables students' remote access, collaboration and active participation during lectures. However, cWMs in collaboration with the PUPIL system approach this concept from the students' perspective. The student is provided with a

collection of workspaces tailored to the needs of the available artifacts (i.e., desk, board) that facilitate studying and transform boring school "chores" into pleasant activities.

## 3 Classroom Window Managers

A window manager is the system software that controls the placement and appearance of windows within a windowing system in a graphical user interface, and most of them are designed to provide a desktop environment. The cWMs are engaged with an additional task as well, namely to ensure the optimal display of the launched applications according to the characteristics of the targeted classroom artifact.

The window managers introduced here are employed in the technologically-enhanced ambient classroom, where every artifact incorporates situation-aware functionality offered by the ClassMATE platform [5]. The cWMs were developed following a user centered design process (UCD) and focus on the students' needs gathered through a requirements elicitation process. Furthermore, each cWM has unique characteristics, as it is purposed for a specific artifact (i.e., student's desk, classroom board) trying to create an ideal workspace environment. Table 1 lists the available artifacts.

**Table 1.** The classroom artifacts supported by the classroom Window Managers

| Artifact | Characteristics | Resolution | Use |
|---|---|---|---|
| AmIDesk | Vision-based touch-enabled device | 1600x600 | desk |
| SmartDesk | Touch screen device | 1440x900 | desk |
| AmIBoard | Vision-based touch-enabled device | 1920x1200 | board |
| SmartBoard | Touch sensitive interactive whiteboard | 1024x768 | board |

### 3.1 Core Modules

The cWMs incorporate mechanisms that transform the educational applications for optimal display in the various artifacts. To this purpose, the applications should be defined following a particular structure that is automatically translated by the cWMs, composing the appropriate application version.

A generic application layout contains: (i) the content, (ii) a primary navigation area, visualized as tab items, for navigating through the different content categories and (iii) a secondary navigation area to facilitate navigation among the various subcategories of the selected content category. In order to conform to the aforementioned specification, every cWM-enabled application defines its skeleton using the following XAML elements:

(i) **App:** denotes the root of the Application Tree hierarchy, which subsequently defines a single application (e.g., the Multimedia application).

(ii) **Applet:** denotes a distinct content category (e.g. the Image Displayer or the Video Displayer of the Multimedia application). At runtime every Applet is translated into a Tab item and populates the primary navigation area.

(iii) **AppletComponent:** designates a sub-division of an Applet and at runtime it is translated into a menu item of the secondary navigation area. An illustrative example is the Image Displayer of the Multimedia application, which can be divided into the "Book Images" and "Wikipedia Images" subcategories.

A designer uses the aforementioned elements to build the appropriate hierarchy and create the Application Tree. Subsequently, each AppletComponent can be populated using either native WPF or PUPIL widgets, thus building the UI of the application. The following example presents the Application Tree of a Multimedia application that can on the one hand host images from different sources and on the other hand display videos relevant to these images.

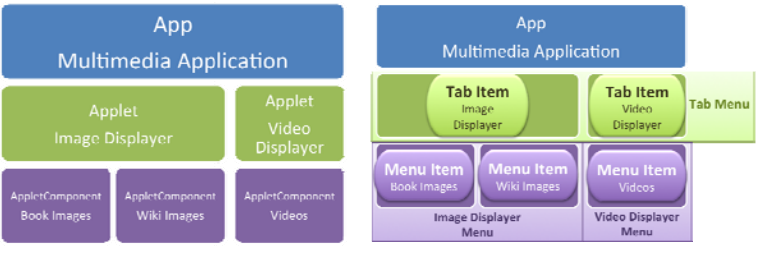


**Fig. 1.** The Application Tree of a Multimedia application

Figure 1 is a graphical representation of the application layout composition according to the Application Tree. The composition process consists of the following steps:

- The Applet elements generate the application's tab items and menus – one menu for each tab item
- The AppletComponent elements generate menu items that populate the respective menus
- The actions of the generated tab and menu items are defined so that each tab item activates a menu and each menu item displays the appropriate content
- The children of AppletComponent elements are appropriately adapted to the artifact needs to result in optimal content presentation.

The cWMs support application layouts that can be customized at design time through a number of available options of the Application Tree (e.g., menu orientation, tab orientation and style, etc.); however, device characteristics may lead to their modification at run time. For instance, consider a designer who creates an application having in mind the AmIDesk artifact. The designer selects the horizontal version of the menu and places it above the main content. The same application, if launched on the SmartBoard, will have a different layout where the menu will maintain its horizontal orientation but it will be placed below the main content to be reachable by younger students. The variety of placement alternatives for every main UI component (menu, tabs, main content and application previews) results in more than one layouts for each artifact (AmIDesk, Netbook, SmartBoard, etc.).

As far as the content UI is concerned, the WPF WrapPanel container was selected as the default container to host all the included widgets in order to take advantage of all available space that an application might exploit. In more details, the WrapPanel is used to position child elements in sequential position from left to right, breaking content to the next line when it reaches the edge of its parent container. According to this behavior, two widgets displayed next to each other in a big screen would be rearranged in a smaller screen so that the second widget goes to the next line.

Nevertheless, a system that only supports UIs where all the contained UI elements wrap in the available space does not provide an optimal solution. The designers should be given more flexibility when building the UIs, without compromising the application's usability and appealing appearance when traveling among artifacts.

To address these issues, the group and breakLine designer elements were introduced. The group element enables the application designer to denote that all the contained widgets must always be placed on the same line. However, forcing the display of a number of widgets in a single line entails the risk of hiding some of them due to space limitations. In such case, since the designer's directive was single line occupancy, the strategy is to scale down the contained widgets to fit the designated area without hiding any of them. On the contrary, the use of the breakLine element forces the following widget to be displayed on the next line.

### 3.2 AmI Desk and Smart Desk Window Managers

The AmIDesk [1] and SmartDesk Window Managers have similar characteristics, since they are both purposed for the student's classroom desk. Their key feature is to display two applications simultaneously to enable student's parallel work. In addition, they support working area organization by offering application sliding from left to right and vice versa. Since both the AmIDesk and SmartDesk artifacts (see Figure 2) offer widescreen resolutions, any application that needs to display additional information, such as history or help, can take advantage of the extra horizontal space and launch a new application next to it with the relevant content.

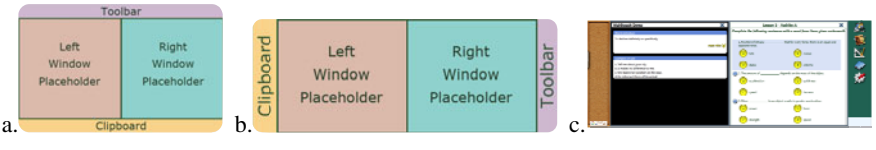


**Fig. 2.** The window manager layouts for the SmartDesk (a) and the AmIDesk (b). Screenshot of the AmIDesk window manager with two launched applications (c).

**Toolbar.** The Toolbar area is a specific region offered by the AmIDesk and SmartDesk Window Managers where useful shortcuts are displayed. The Personal Area shortcut is always visible, allowing the student to easily browse through educational/ personal material. The lower area of the Toolbar is populated with course related applications, when a shortcut is selected the respective application must be launched.

**Clipboard.** The Clipboard temporarily holds instances of applications - e.g., one school hour- and disposes its contents before the next class (the student will be asked whether to save the pinned application instances in the Personal Area). As soon as an application is pinned to the Clipboard, its state is stored and the student can restore it by selecting the respective application thumbnail. Subsequently, a student can pin an application instance in order to access it later. The Clipboard permits browsing the pinned thumbnails through scrolling. When a thumbnail is pressed, it disappears from the Clipboard and the relevant application is either launched if suspended, or brought to front displaying the data recalled through the pinned instance. Finally, the pinned items can be removed by striking through them (right to left) and for mass removal an "Unpin All" button is provided.

**PieMenu.** The Pie Menu (Figure 3a) is a round menu that appears when the student touches the screen at the same point continuously for more than 400 milliseconds, displaying a number of options. The Pie Menu allows reorganizing the launched applications. The menu includes on the one hand commands such as sliding applications from left to right and vice-versa, and on the other hand commands such as sending application instances to the Clipboard, the class board and the teacher. When the Pie Menu is activated the student can either execute a command by sliding a finger over the desired option, or deactivate the popup menu by lifting the finger from the screen. When the menu is activated, only one sliding option is displayed according to its position, so if the menu is launched on the left part of the screen the "slide right" is visible. These two options appear at the left and right of the Pie Menu respectively, fulfilling the natural mapping design principle. Accordingly, the "send to class board" option was placed at the top center of the round menu to take advantage of the physical board's placement in a classroom. When an application instance is sent to the class board, the same application is launched at the board artifact, displaying the same data for all the students. Finally, the "send to teacher" option is visible only when the selected application is an exercise that needs to be submitted to the teacher.

**Application Switcher.** The Application Switcher (Figure 3b) is a utility activated when the student touches a specific area on the screen (down right corner for the AmIDesk and up right corner for the SmartDesk). It displays thumbnails previewing the launched applications and the student can select one or two applications to view. When an application thumbnail is touched, the Application Switcher collapses and the respective application(s) come to front. When active, the Application Switcher can be closed by touching anywhere else on the screen.

**Placement Decision Making Component.** The fact that the AmIDesk and SmartDesk Window Managers support the presentation of two applications simultaneously raised the need for a Placement Decision Making Component. This component decides the placement (left or right) of each newly launched application, ensuring that an application and its additional information will be launched next to each other. Furthermore, the algorithm prioritizes the most recently used application by not placing the new one on top of it, preserving the student's working area on focus.

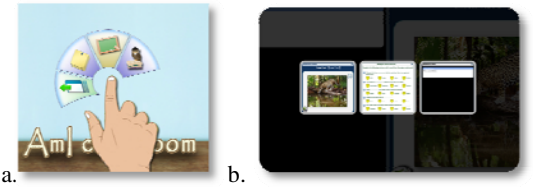

**Fig. 3.** The Pie Menu (a) and the Application Switcher (b)

**Glass Input Handler.** The Glass Input Handler is a transparent mechanism that supports the Pie Menu commands of the AmIDesk and SmartDesk Window Managers. The concept of this mechanism was based on Java's Glass Pane [7], which is like a sheet of glass over all the UI elements intercepting input events. Similarly, the Glass Input Handler is handling the touch events transparently and either propagates them to the respective applications or launches the Pie Menu. When a "touch down" event (the user's finger contacted the touch-sensitive screen) event is raised, it must be checked whether the student was aiming an application or intending to activate the Pie Menu. Since a single "touch down" event indicates that the student may intend to activate the Pie Menu, the Glass Input Handler is responsible to monitor the sequence of the upcoming events and decide whether to simply propagate them or launch the Pie Menu.

**Application Layout.** The AmIDesk and SmartDesk artifacts support various layout alternatives that can be specified at design time during the Application Tree construction or at runtime according artifact characteristics (see Figure 4).

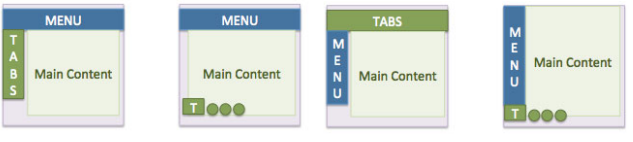


**Fig. 4.** Layout alternatives for the desk window managers

### 3.3 Smart Board and AmI Board Window Manager

The SmartBoard and AmIBoard Window Managers are intended for a board artifact. They both provide the Toolbar and Sidebar features, while the AmIBoard Window Manager additionally supports a layout that integrates application-related previews to utilize the extra space offered by its higher resolution (see Figure 5).

**Application Launcher.** As a board artifact is aware of the current course, it was necessary to determine an appropriate placeholder for displaying course-related application shortcuts. The concept of placing these shortcuts on the Toolbar area was discarded, as it was quite likely that some of them might become unreachable for younger learners. For that reason, a new module was introduced, the Application Launcher, which displays a grid of application shortcuts following a bottom-up placement algorithm.

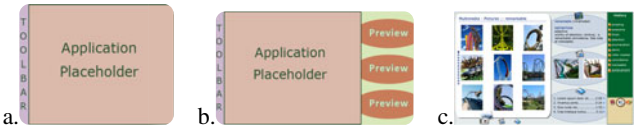


**Fig. 5.** The window manager layouts for the SmartBoard (a) and the AmIBoard (b). Screenshot of the AmIBoard window manager displaying the Multimedia application (c).

**Toolbar.** The Toolbar items of a board artifact are the Application Switcher shortcut, the Application Launcher shortcut, and the application close control. The Board, as opposed to the Netbook artifact, does not display information concerning an individual student, thus the Personal Area shortcut was redundant. Taking into consideration the Toolbar's vertical placement, its contained items anchor at the bottom to be reachable by all students.

**Previews.** Previews take advantage of the AmIBoard extra space. They display, in parallel to the currently selected content, snapshots of other information available through the same application. The designer can easily employ this feature by selecting the appropriate preview-enhanced layout that gives users a glimpse through the application's categories/subcategories. Figure 5c presents such a layout for an enhanced Dictionary Application. While the students observe images related to the word "remarkable", the preview placeholders may display the definition of the word and/or some relevant videos and audio. When the student or teacher points to one of the preview areas the respective category/subcategory is displayed.

**Application Layout.** The application layouts supported by the board Window Managers are depicted in Figure 6. Their key feature is that the Tab and Menu placeholders are lower in the screen, thus making them easily accessible to younger students.

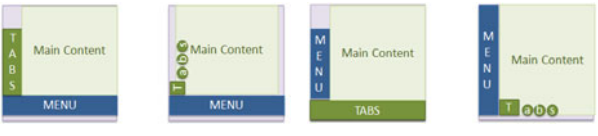


**Fig. 6.** Layout alternatives for the board window managers

## 4 Heuristic Evaluation

As the development of the Window Managers had advanced, a number of educational applications were build (e.g., Book Application, Multimedia Application) in order to evaluate the cWMs. A heuristic evaluation [6] of the SmartDesk Window Manager was conducted.

The evaluation of the SmartDesk was performed by five evaluators. The usability issues discovered where noted down by an observer, who was monitoring the evaluation progress. The most important discovered issues are:

I1. The Pie Menu should not be activated when no applications are launched
I2. Application Switcher should not be activated when no applications are launched
I3. The application slide should be performed by direct gestures
I4. Applications should allow maximization
I5. When an application is restored from the Clipboard it is unpinned. However, applications should remain pinned to the clipboard until the user decides otherwise.
I6. Some Pie Menu Icons were not intuitive

The chart presented in Figure 7 illustrates the heuristic evaluation results, ordered according to their severity. The most severe usability issues that were ranked higher than "major" usability problems (>3) are marked in red color. These findings are the most imperative to fix. The issues ranked higher than "minor" (>2) and which are represented in the figure by the orange-colored bars are next in turn to be fixed. Issues marked in blue color range from "just aesthetic" problems (severity rating 1) to "minor" problems (severity rating 2) and will be fixed before the next system version becomes available.

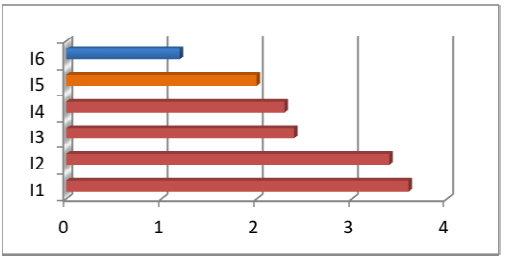


**Fig. 7.** Heuristic Evaluation results

## 6 Conclusions and Future Work

The main objective of this work was to lay the foundation towards the realization of a unified working environment inside the ambient classroom, where the students are constantly supported during their everyday educational activities. Each classroom artifact is equipped with an appropriate workspace (Window Manager) keeping in mind its special characteristics and the user's goals of use. Since the cWMs were developed following a user centered design process, the user needs and goals became clearer. On the other hand, the feedback received through the heuristic evaluation led to the discovery of significant usability issues.

Additional steps should be taken to fully support the initial concept. The first step is to conduct a full-scale user-based evaluation of the currently developed cWMs, in order to acquire additional feedback from the end-users. The evaluation experiment should include scenarios that engage all the supported artifacts, so as to record and analyze their acceptance by users in real-life situations. Furthermore, since most students nowadays possess numerous mobile devices (i.e., laptop, PDA), such devices should be incorporated in the enhanced classroom. Finally, teacher's support should

be included to assist the "maestro" of the class. A completely different requirements' elicitation process will be followed, as the teacher's goals differ from the student's goals, while the component-based architecture of the existing system will facilitate the integration of additional software (e.g., teacher-oriented applications) with minimal cost.

## References

1. Antona, M., Margetis, G., Ntoa, S., Leonidis, A., Korozi, M., Paparoulis, G., Stephanidis, C.: Ambient Intelligence in the classroom: an augmented school desk. In: Karwowski, W., Salvendy, G. (eds.) Proceedings of the 2010 AHFE International Conference (3rd International Conference on Applied Human Factors and Ergonomics), Miami, Florida, USA, July 17-20 (2010)
2. Breuer, H., Baloian, N., Sousa, C., Matsumoto, M.: Interaction design patterns for classroom environments. In: Jacko, J.A. (ed.) HCI 2007. LNCS, vol. 4553, pp. 163–172. Springer, Heidelberg (2007)
3. Cooperstock, J.: Classroom of the Future: Enhancing Education through Augmented reallity. In: HCI International, pp. 688–692 (2001)
4. Korozi, M.: PUPIL: pervasive UI development for the ambient classroom (Master's thesis), (2010), e-Locus, `http://elocus.lib.uoc.gr/dlib/a/e/2/metadata-dlib-81a07682706c2163d8f582245fd9edfd_1288689489.tkl`
5. Leonidis A.: ClassMATE: Classroom Multiplatform Augmented Technology Environment (Master's thesis), (2010), Available from e-Locus, `http://elocus.lib.uoc.gr/dlib/d/3/6/metadata-dlib-d19ded9b6c0938f4723672b56b78ebe2_1288598057.tkl`
6. Nielsen, J.: `http://www.useit.com/papers/heuristic/`
7. Oracle, The Java Tutorials - How to Use Root Panes, `http://download.oracle.com/javase/tutorial/uiswing/components/rootpane.html`
8. Shi, Y., Xie, W., Xu, G., Shi, R., Chen, E., Mao, Y., Liu, F.: The smart classroom: Merging technologies for seamless tele-education. IEEE Pervasive Computing, 47–55 (April-June 2003)
9. Xu, P., Han, G., Li, W., Wu, Z., Zhou, M.: Towards intelligent interaction in classroom. In: Stephanidis, C. (ed.) UAHCI 2009. LNCS, vol. 5616, pp. 150–156. Springer, Heidelberg (2009)
10. Yau, S.S., Gupta, S.K.S., Karim, F., Ahamed, S.I., Wang, Y., Wang, B.: Smart Classroom: Enhancing Collaborative Learning Using Pervasive Computing Technology. Engineering, 1–9 (2003)