Application of Personas in User Interface Design for Educational Software

Ursula Dantin

Department of Information Systems and Operations Management
School of Business, The University of Auckland
Private Bag 92019, Auckland, New Zealand

u.dantin@auckland.ac.nz

Abstract

In the current cautious business climate, development of large new software has become rarer as the cost pressure has increased. User interfaces (UI), the most important part of a system for end users and critical for system success, remain notoriously user-unfriendly. This makes it imperative to identify practical tools that small software projects can use to help them maximise UI design quality while minimising cost.

This research looks at two similar small systems and investigates the usefulness of the concept of personas for UI design evaluation. Following the Goal-Directed Design approach, personas are defined along classes of users. The tasks of each persona are established via user-centred requirements. Each task is then performed with consideration to established usability heuristics. Overall, the question is whether the initial use of personas in the design phase might have resulted in fundamentally different UI choices.

It was concluded that the approach of identifying personas and performing their tasks in evaluating the UIs of both software systems was most definitely a process that helped introduce clarity and a form of accountable reasoning into the UI evaluation process. For both systems, it could be reasoned that the UI design would have been fundamentally different in some aspects if personas were used. However, personas were less helpful when it came to diagnose and describe user frustrations that had not so much to do with UI design as such, but with general usability issues. In this case they were mainly related to discrepancies and inconsistencies in the underlying business logic or simply bad programming, pointing to deficiencies in other areas of the software development life cycle (SDLC).

Keywords: persona, user interface design, educational software, e-learning.

1 Introduction

User interface (UI) design seems still to be a neglected part of the software development life cycle (SDLC). But also is it usually the last step in designing new software, with the well documented difficulties of IS projects to stay on time and on budget, this is when the developers are running out of both (Paynter *et al.*, 2001). As a consequence, UIs are notoriously user-unfriendly. Good experts on UI design are rare and expensive because the skill set and expertise in this area cannot be developed on a broad base of working experience. Especially in small software development (SD) projects, UIs are designed when the software is already in the final stages of development to accommodate the logic of the underlying code rather than an identified set of user needs. The "entire development process has to be turned around so that it starts with user needs and ends with engineering". (Norman, 1998)

2 Research Method

Heuristics to evaluate UI design quality and, consequently, design better UIs are for example proposed by Nielsen (www.useit.com). (Appendix 9.2) In SD, it is generally argued that the UI design must be fully integrated into the SDLC and users must be considered right from the start of a design. Attempts to integrate UI design by developing integrated design tools relate usually to large expensive software projects. However, with the design of large new software becoming rarer in the current cautious business climate, what practical tools can small software projects use to help them maximise UI design quality while minimising cost?

In the SDLC teaching literature, prototyping seems the standard way to address UI design for smaller SD projects. However, Randolph (2004) suggests that the concept of personas, developed by Cooper (1999) in his Goal-Directed Design approach, could be useful in this environment. This tool could help developers of small software packages to start thinking about their users early in the SDLC and integrating the UI design without huge cost. User-centred requirements using personas are for example suggested by RedHat (www.redhat.com). (Appendix 9.1)

Randolph wants us to think of personas as "hypothetical users – fictional people who represent classes of users" (2004). To help the software designer make these personas real they are fully fleshed out with personal attributes and personal goals. This could also help with the often difficult task of capturing and considering non-system related issues and creating intrinsic value for users of a system. Not all the personas identified may be considered in a UI design, but each UI would be designed for at least one primary persona. Later, in the evaluation phase of the SDLC, Randolph suggests that this primary persona's needs and goals must be satisfied to declare the system a success.

Personas were developed following Randolph, using also RedHat's user-centred requirements (Appendix 9.1) to ensure all goals and interactions of the personas were captured. During the evaluation of the UI designs for each persona, the ten usability heuristics suggested by Nielsen (Appendix 9.2) were considered.

This research applied Randolph's persona approach to evaluate two existing small educational software packages used in the Business School at the University of Auckland, NZ. This type of software was chosen because it seems increasingly to be used in tertiary course administration worldwide. Competing applications have become available in recent years. But as their initial versions tended to be rather basic, most are currently experiencing further expansion of functionality and sophistication, which may include changes in UI design.

In the following chapter, we describe the two small educational applications chosen. In chapter 4, we describe the personas developed from the person profiles and informal interviews of staff using each system. They reflect different task sets and IS capabilities of the users. (Appendix 9.3, 9.4) We consider how each of these personas might use the system to do their tasks. This leads in chapter 5 to the description of user interface functions needed for user groups based on the personas identified in the previous chapter. In chapter 6, the existing UIs of the two applications are evaluated in some detail for their task logic and ease of use, i.e. user-friendliness for each persona. In the final chapter, we consider whether the initial use of personas in the UI design phase might have resulted in fundamentally different interface choices.

3 Software evaluated

In this chapter, we describe the two small educational applications. The two small software packages used to follow up on the usefulness of personas in UI design were chosen because they are both concerned with similar aspects in managing tertiary courses.

3.1 Cecil

Cecil is a custom-designed enterprise learning management system developed and used at the University of Auckland, New Zealand (www.cecil.auckland.ac.nz). Of the more than 32,000 students enrolled at this university, during 2003 27,500 students were enrolled in courses that were using Cecil in some form. This was an increase from approximately 22,000 in 2002, and 15,000 in 2001. Cecil has two completely different interfaces, on the one hand for the students and on the other hand for the staff administering the courses. Both interfaces are web-based. Students and staff can access the system on campus as well as over any Internet connection. The number of staff involved in teaching these courses and using the administration interface of Cecil increased even more dramatically than the student numbers from just 500 in 2001 to 1,000 in 2002, to 1,400 in 2003. The reason is that the advantages of electronic course administration were first recognised by large courses. Small courses with a higher staff/student ratio adopted Cecil later. The turnover of course administration staff is high due to the nature of a university with many senior students working as temporary part-time staff. There will be a large number of first time users for both interfaces each semester.

For this study, we are only concerned with the administration interface of Cecil. The main features that are used at the moment on this interface seem to be:

- Course details, such as course objectives and staff contact details, are made accessible to students.
- Course materials such as lecture slides, PDF files, website links, and even video files of taped lectures are made available on the student interface for viewing or download.
- Announcements which can be posted on the student interface of a course and simultaneously emailed to the students' university mailboxes.
- Final course grades for students, usually composed of marks for single activities like tests, assignments, and exams, are stored on the Cecil administrator interface by the course coordinator or lecturer and accessed by the central university registry.

Cecil has further facilities that especially ease the management of large courses:

- A streaming facility, where students can enrol into tutorials or lab classes where seats are limited, or where they can form project groups.
- A facility to create and electronically administer multi-choice tests. The tests are marked automatically. This is already efficient for courses of 20-30 students, but even more so for large courses of several hundred students.
- All marks for partial assessments during a course can be managed and stored on the Cecil administrator to be added and even scaled in the end.

The administrator interface has four staff roles (coordinator, lecturer, tutor, and marker) reflecting the different possible job titles of staff involved in a course. At the moment there is only one small difference in the interface capabilities: Only coordinators can add somebody to the staff of a course. In all other aspects, the interface look and feel, and capabilities are identical. To help staff getting through the most common course administration tasks each course has a 'Task pad' with six wizards. The task wizards are:

- Create New Announcement (= send a message to students)
- Put Files Online (= make material/files available)
- Create New Activity Session (= create the course structure)

- Course Import Wizard (= re-use an existing course structure)
- Create Question (= create multiple choice questions for on-line tests and exams)
- Create New Test (= create the test itself using the questions)

3.2 Turnitin

Turnitin, created by the iParadigms team, is used “by thousands of institutions in over fifty countries” and recognised “worldwide as the standard in online plagiarism prevention” (www.turnitin.com).

Like Cecil, Turnitin has a student interface and a staff interface. Students can check their own work for plagiarism (if this feature has been enabled) and make necessary changes before officially submitting their paper to the course. A peer review facility can also be set up by the instructor of a course for students to assess each others work. We are not investigating this area of Turnitin, but are solely interested in the course administration interface of the product. Turnitin offers, apart from student and staff, a third user type – the administrator. This gives each participating institution the option to appoint one person to be the overall Turnitin administrator to assert a certain level of control over the access to Turnitin within their own organisation. Because this function is not involved in any actual work in any course we ignore this role for the purpose of the comparison.

Turnitin offers the following staff features:

- Plagiarism prevention (=identification of copied material)
- GradeMark (= an advanced grading tool)
- GradeBook (= assignment and grade administration)
- Digital Portfolio (= document archiving of submitted assignments)

The iParadigms team is working towards integration of Turnitin with existing student administration systems like Blackboard.

4 Application of personas

In this chapter, we describe the personas developed from the person profiles and informal interviews of staff using each system.

The administrative job structure for course delivery at the University of Auckland was analysed and informal discussions were conducted with real users from various departments. The user profiles, user responses on the tasks performed and how they used each system were the basis for the development of the personas and their associated tasks. The blueprint for the personas was adopted from Pind (2001). The responses of the interviewees on what they were happy/unhappy with were later used in assessing the user-friendliness of the existing system.

The administrative structure can be different for each course. This is up to individual departments or even lecturers. It means that a “tutor” can have a different job description and tasks/authority in different courses. The task structure of large courses is usually more differentiated, even hierarchical, compared to a course with small student numbers. The complexity of the task structure affects the analysis of Cecil because it offers a sophisticated administration tool for most aspects of a course. Turnitin is only concerned with the management of student assignment documents, their check for plagiarism and the recording of marks. As a result, we considered more personas for Cecil than for Turnitin.

4.1 Cecil users

A large course (over 850 students per semester) was used to explore a differentiated task structure. In this case, the course coordinator is the overall administrator of the course. S/he would like to control read/write permissions for her/his course on an individual staff and time basis. It means that two “tutors” may have different types of access at different times during a course. This enables the coordinator to respond flexibly to administration demands as they arise. Using the electronic administration capabilities of Cecil can result in a big efficiency gain and improved communication with students.

Courses with 20-40 students are considered small. The main difference to the large courses is that the lecturer is the administrator of the course and has few, if any, assistants. The number of students is small enough for personal contact to be established easily. The electronic administration of communication and tests via Cecil does not lead to a big efficiency gain compared to existing, more traditional methods. It replicates these traditional methods rather than replacing them with something more efficient.

4.2 Cecil personas

- Lecturer (super user)
- Course Coordinator (constant user)
- Lecturer small courses (casual user)
- Lecturer large courses (technophobe - reluctant user)
- Tutor (infrequent user with limited authority)

These personas could be further reduced. The super user is in a sense a combination of a course coordinator and a lecturer of small courses. The difference is that s/he chooses to use many features in Cecil on principle, because s/he is an early adopter even though they bring no considerable time saving for his/her smaller courses. (Appendix 9.3)

4.3 Turnitin users

We look at courses with word based assignments like essays and programming code as part of the course assessment. In this case, it does not matter whether the course has large or small student numbers. Either way, it must be established whether the assignment handed in is the student’s own work. The annotations and rubrics facility is convenient to use as feedback tool to students for a course of any size. However, for large courses it may make more sense than for small courses to use the marks facility of Turnitin and export the assessment marks later into the internal marks administration of the course (like Cecil or Blackboard).

4.4 Turnitin personas

- Instructor (full access user)
- Teaching Assistant (infrequent user with limited authority)

It is up to the instructor to decide how much authority the teaching assistant should have. In the extreme, the access rights would be identical and we would effectively have just one persona. (Appendix 9.4)

5 Proposed user interfaces

Now, we describe user interface functions needed for user groups based on the personas identified in the previous chapter.

By looking at the tasks and needs of the personas identified, it seemed possible to further group these personas. This could be seen as an attempt to create what Randolph (2004) calls primary personas with one distinct UI for each. For each software examined however, it was felt that moving away from the personas identified to just primary personas would mean loosing too much valuable information. The picture would become too generalised and peculiarities of certain personas would be lost. It seemed a case of forcing a "one size fits all" which often really results in nobody being satisfied.

5.1 Cecil user interfaces

From the personas identified, it seems possible to define at least two distinct user groups, warranting two distinct user interfaces for Cecil.

One group would include the lecturers with large courses, technophobes, and the tutors. This group is not very comfortable with a complex system and user interface. They want to infrequently perform some core tasks, like sending an announcement or making lecture slides available, without having to wade through a labyrinth of clicks and screens. They are not interested in student administration tasks because they can delegate these, are not allowed to do them, or they cannot cope yet with the complexity of the task.

The other group would be the people involved in administration of larger courses and special projects, and people interested in using the technology. Especially for large courses of several hundred students it makes sense to use the on-line test facilities, the marks administration, announcements and bulk email communication with students. With big groups like this personal contact with students is limited. Electronic student administration and communication are a good way of reducing the workload and insuring equitable access for students to resources.

Staff running small courses fall somewhere in between. They could use all the same facilities as are used for the large courses but they could manage without Cecil. With small student numbers, personal contact can be readily guaranteed and even manual administration of marks during the semester would be possible. So it really depends on the individual lecturer which user level s/he wants to join. The super user persona would be an example of a staff member who has chosen to use the full facilities even for smaller courses without coordinator support. This persona could even jump between both groups and interfaces depending on the task at hand.

5.1.1 Interface 1

The interface for the first group, the non-confident user with limited tasks should be easy, clean, non-cluttered with only one window visible at any time and a clear sequencing of screens. This kind of interface is usually supplied by wizards. The tasks for this group are:

- Making simple announcements
- Making materials available using a simple course structure
- Final marks to go electronically to the Registry office

5.1.2 Interface 2

These personas are technically confident and tend to use the system frequently which means they can cope with more complex interfaces, with multiple windows open at the same time. They may have the system running all day and want to switch ad hoc between tasks.

The interface for the second group could be concerned with:

- All electronic student course administration tasks
- Adding and deleting students
- Making materials available using a complex course structure
- Sophisticated communication
- Full marks administration
- Running on-line tests
- Ability to check if test is actually working (trial runs)
- Checking what students can see and access.

The personas in this group will need access to the tasks of Interface 1 as well. But these users might appreciate a more sophisticated approach to task capabilities. They may want to email an announcement just to a single or sub-group of the students. They may want to use discussion groups to enable communication in large courses. They publish a plethora of course material and may want to make materials available under various sub-headings or folders to help students cope with information overload on their screens. They want to administer their marks completely on Cecil with all partial results like lab tests accumulating over the duration of a course. To help with decisions about scaling, they want to get statistical reports and graphs, maybe even use what-if scenarios.

5.2 Turnitin user interfaces

From the personas identified, it seems possible to work with just one user interface for Turnitin.

Apart from the setup of new users, classes, and assignments, the tasks performed could be identical. It depends more on the division of responsibilities in each course. To have one interface and use passwords for the protection of sensitive task allows for full flexibility in each individual course situation.

5.2.1 Interface

Because the interface should cater for all users, it should be easy to use and not cluttered. The user should be able to step through a task intuitively. The tasks for the interface are:

- Setup of courses/classes and assignments
- Process student submissions (text)
- Look at reports

6 Evaluation

In this chapter, the existing UIs of the two applications are evaluated in some detail and compared to the proposed UIs from the previous chapter.

Evaluations of the existing UIs of the two applications were performed by stepping through the various tasks identified for the personas. (Appendix 9.3, 9.4) Nelson's heuristics for good UI design were taken into account. The existing UIs were evaluated for their task logic and ease of use, i.e. user-friendliness for each persona. Is the step-through for each task logical and easy to follow for the persona? To what extent are the persona's needs and goals satisfied? Does the access to tasks granted by the system reflect the authority of this persona in the organisation's administration hierarchy? Overall, the question is whether the initial use of personas in the UI design phase might have resulted in fundamentally different interface choices.

The existing Cecil administration UI follows closely the design rules of Microsoft Windows. This is not only evident in the way the course structure is visualised in the left hand window, but goes right through to the choice of colours and icons used for generic buttons. Turnitin takes more a "one-window" approach. The window design is mainly consistent with a web site approach. It does not follow Microsoft Windows design conventions but uses a custom design in white / light grey / light blue with important information standing out in various shades of red.

Turnitin has a task-sensitive help; Cecil only has a search facility by key words for various help documentation. Both have downloadable manuals.

6.1 Cecil

Cecil in its current form seems to provide interface options for both groups – basic and advanced users.

6.1.1 Interface 1

The basic Interface 1 is contained in the wizards accessible by large symbols on the task pad of each course. They cover the design and setup of tests as well as the basic tasks identified here for Interface 1. Small problems exist with the lack of meaningful explanations in some wizards. This will result in possible stress for new users and technophobes. Frequent users have taken that hurdle and remember simply how to work through the wizard. They usually even do not notice the lack of explanations anymore. The same applies for strange explanations clearly written by non-native English speakers and small inconsistencies in naming conventions between the wizards and the non-wizard areas.

A particular oddity seems to be that the last "next" in most wizards is not marked as the final important step which actually invokes the action. Instead, a message box comes up only afterwards telling what the system has just done. All one is left with is to click the finish button to close the wizard. Following usual wizard design, the last "next" button should be labelled differently plus an additional "yes/no" message should come up to ask for a final confirmation before the task is actually performed. Ideally, there should even be an "undo" facility.

To get the final student marks into Cecil is quite easy but still requires insider knowledge because the existing wizard style interface is hidden behind a right-click and the whole process consists of two independent steps in Cecil. A wizard guiding through the whole process and residing on the task pad would be a big improvement.

What Cecil cannot provide so far is an effective limitation of access to particular tasks. A tutor for example cannot have access to record students' marks just for a particular test. They will be able to access all student data (including personal records) and change all marks in the course. As a consequence, some coordinators or lecturers do not give their tutors access to the Cecil administrator. This means they are not able to delegate simple tasks like writing routine announcements and making lecture slides available.

6.1.2 Interface 2

The advanced Interface 2 can be seen as the whole of the Cecil administrator. It consists of a window to the left showing the course structure and usually two more windows to the right showing details using an additional tab structure where necessary (sometimes even at the top and bottom of a sub-window) to display rich information. Sometimes, the structure becomes even more complex with pop-up windows accessible on double click. From this pop-up window the task can be accessed, again from within a double window and/or tab structure. In principle, this structure follows Windows conventions and should be familiar to anybody with intermediate computer literacy. But the screen appears cluttered and the structure may become less experienced users of the system. Not all windows visible are refreshed when necessary. Drill down is inconsistent and not in all cases possible from all windows. The user cannot easily remember the navigation the next time the task is required. Tasks performed in pop-up menus cannot be interrupted to quickly check on something else within Cecil. In some cases, the user must have performed a certain task before a wizard is started. The system only prompts the user halfway through the task sequence in the wizard on the need for this task, without giving the opportunity to actually do it at that point. As a result, the task sequence must be aborted and the initial task must be performed somewhere else. Then the user must start the wizard again and can finally complete the task sequence.

The file structure in the left window is the only way to access the contents. Staff members can access in this way course contents of courses they were involved with in previous years. Over time, this can build up to a considerable amount of data. Since the system has no other search facility, the user can only get to a piece of information if they can remember exactly in which folder, within which course, within which year and semester the piece of information they want to access is stored.

Problems for users seem to occur from a lack of clarity about the steps involved in a complex task they wish to perform or where the functionality is 'hidden'. Some tasks can be accessed through a right mouse click. Another similar task may only be accessible within a particular tab. Functionality on the main menu bar is limited and not consistently available. This could be an indication for a system that is still under development towards full functionality. 'Copy' and 'Paste' are for example sometimes possible using the shortcut keys, but not, or only partially, available on the main menu or with right clicks. To rename a file or folder is sometimes possible, sometimes not. When a test question is copied into a completely different folder, a rename to "copy of …" is always enforced. This is inconsistent with Windows principles and inconvenient for most users. However, there is also no facility to backtrack to where the copy came from which makes version control problematic.

Some tasks, like creating a number of streams and making them available for students to book into a lab, require multiple steps hidden behind various right clicks and tabs. The logic is complicated to remember and inconsistent with the business logic of all users interviewed. To successfully perform the task is made even more difficult, since it is usually done only at the start of a course. So, when the next semester or year comes around, the task sequence will need to be re-discovered by trial and error or by reading a help file.

Some concepts in the underlying business logic of Cecil, like the difference between "copy" and "link" when importing a course structure from an existing course in a new course, are not sufficiently explained, even in the wizard. Documents for importing into the course must first be saved onto an A, C, or D drive because Cecil cannot access the network.

Many problems advanced users experience are not so much related to the user interface but to the underlying business logic of Cecil, and affect the usability of the system. At the University of Auckland the rules on "how we do things" are as diverse as the courses on offer since the internal course administration Cecil is trying to support is up to each department or even lecturer. Like in any other business environment, users get frustrated and often refuse to use a system if it requires them to change the way they "do things". Options for doing things differently are not available or hidden somewhere, frustrating less experienced or infrequent users. On the other hand, when the business logic is changed to accommodate some users, established users are often taken by surprise when the system suddenly "does weird things" and they cannot perform a task the way they used to do it or cannot do it any longer when an update is implemented. One can argue that this is really not strictly a problem of user interface quality as such but more one of general system usability. Both are related to user satisfaction within ongoing software development management.

Another issue falling more into the general area of usability and not strictly UI design is the sometimes annoying, even confusing, lack of proper window content update. In the chosen UI design with its multiple windows it happens quite often that old content is left in one of the windows when the user has moved already on to another task or area of content within the main window. In a sense, all windows usually displayed are hierarchical, like Windows Explorer, with the left window displaying the file structure and the right window(s) displaying the contents of whatever is selected in the left window. In Cecil, this update of the right windows, when selecting another file in the left window, does not always happen. Either the old contents turns to "gobbledygook" or what is much worse, it remains unchanged and accessible. This means the user really still works in the old environment when s/he actually thinks s/he has moved on to a different place. This can easily remain unnoticed at first with contents and internal structure of courses being very similar over the years, as well as between some courses.

The walk-through exercises for all personas, as well as the feedback from real users highlighted for Cecil many frustrations that users experience. They are not so much related to UI design but to usability of a system. It became clear that most frustrations users - mostly advanced and super-users – experienced, were related to issues like incorrect or inconsistent business logic within the programming or simply bad programming (faults or bugs). These users were very vocal about their frustrations with insufficiently tested updates, hidden changes to the business logic within an update, and good and much used features like hidden wizards suddenly missing in an updated version. In general, these were complaints about a lack of communication and consultation between users and developers. This is especially surprising in the case of Cecil, as both groups are belonging to the same tertiary organisation, some even work in the same building. This seems to indicate that an academic environment can have the same problems in this area as commercial enterprises.

6.2 Turnitin

Turnitin currently provides one consistent interface. The set up of the structure is protected by password. It seems that the look of the interface can be customised to a certain extent. Each user can specify for example the maximum number of items that should be displayed on a page. The customisation of the turn-around time for the plagiarism check is not regarded as a UI feature.

6.2.1 Interface 1

At the upper left side is a kind of page title "Now viewing:…", followed by an instruction / help area similar to wizard walk-through explanations. Applicable tasks are located at the top of each window as links and/or tabs. Above the window are more tabs to change quickly to other areas of Turnitin. At the top of the screen, the company logo as well as user info tabs and the logout remain constant through navigation. Information is accessed with web links that allow access into the layers of course related information. Selective views are made accessible via drop-down choices where applicable. Some links generate a separate browser window, but this is usually only used to display more detailed information. An actual student assignment for example appears in a separate browser window.

The choice on the maximum number of items to appear on a page can potentially make the structure appear too deep and require too many navigation steps if the number is set too low. If it is set too high, the page can become cluttered and difficult to read.

Class materials and announcements can be made available to students on a calendar. They can be uploaded from any place on the Intranet. This is different to Cecil which can only access local drives such as A, C, D drive. However, the library/archive contents in Turnitin cannot be rolled over into another course.

The process of setting up a Teaching Assistant is not intuitive. Individual students can not be assigned to a specific teaching assistant once students have been imported into the course. Student documents cannot be moved into another course if they were submitted incorrectly. The processing options for student work between 'archive' and 'delete' seem confusing, as is the Library / Archive areas.

The walk-through exercises for all personas, as well as the feedback from real users of Turnitin highlighted not as many frustrations that users experienced. This will in part relate to the fact that Turnitin tasks are less complex.

7 Conclusion

It seems that the approach of identifying personas and performing their tasks in evaluating the UIs of both software systems was most definitely a process that helped introduce clarity and accountable reasoning into the UI evaluation process. It is felt that considering usability heuristics such as those of Nielsen could be less fuzzy if the time was invested to first identify the personas for a system. The usability heuristics then are applied for each of these personas. This combination was used here and resulted in a good understanding of the UI quality issues for both applications. The UI evaluation can be summarised as follows:

The Cecil interface tries to cater for users with more basic levels of capability by providing wizards for core tasks. These wizards have some design flaws. Still, the full interface with its multiple windows is accessible to all users and can confuse or intimidate novice users. The possibility of differentiated access rights for special groups, such as tutors for example, is not meaningfully developed yet. If personas had been used for the interface development, the need for at least two distinctive interfaces (basic – advanced) might have been clearer in the mind of the developers. It might have also been clearer that the majority of the personas/users are not highly computer literate and would prefer to use the basic interface. This could have resulted in the main interface having more guided step-through tasks and a simpler interface with less tabs and windows on any given screen.

Turnitin takes the "keep it simple" road with its predominantly on-window-only approach. This more uncluttered screen, together with a generous amount of help information provided, makes it a more easily accessible system. The help information is unobtrusive and should not annoy experienced users. The only complaint here might be that the single window approach requires too much clicking and moving through multiple screens to navigate to a particular point in the system layers. The persona approach could have resulted in more clarity on the need for limited, customised access to teaching assistants. Turnitin relies on a small obscure preference feature (number of items per page) for some screen customisation.

In conclusion, we believe that the UI design of both systems would have benefited from using the persona approach as proposed by Randolph. The UI evaluations performed, using personas in combination with Nielsen usability heuristics, was not time consuming and required no any additional software applications. This suggests that it is indeed an inexpensive yet effective option for UI design of small software applications. Even after implemention, personas can be a valuable tool to assess usability and pinpoint areas for improvement.

8 References

Carroll, J.M. (2002): *Making Use: Scenario-Based Design of Human-Computer Interactions*. San Francisco, Morgan Kaufmann Publishers.

Cooper, A. and Saffo, P. (1999): *The Inmates are Running the Asylum. Why high tech products drive us crazy and how to restore the sanity*. SAMS.

Nielsen, J. (1994): Heuristic evaluation. In *Usability Inspection Methods*. NIELSEN, J., and MACK, R.L. (eds). New York, NY, John Wiley & Sons.

Nielsen, J (1993): *Usability Engineering*. San Diego, Academic Press.

Norman, D.A. (1998): *The Invisible Computer. Why Good Products Can Fail, the Personal Computer Is So Complex, and Information Appliances Are the Solution,* MIT Press.

Paynter, J., Ahmed, M.D., Everett, A., and Llanes, V. (2001): An Analysis of Software Development Project Management Failure: Two New Zealand Cases. In *Business Case Studies in Operations Management*. 152-161. BATLEY, T. (ed). Auckland, NZ, Prentice Hall for Pearson Education.

Pind, L. (2001) (Ed.): *Personas*, http://ccm.redhat.com/user-centered/personas.html. Accessed 12 Aug 2003.

Pind, L. (2001) (ed): *User-Centered Requirements*, http://ccm.redhat.com/user-centered/user-centered-requirements.html. Accessed 12 Aug 2003.

Pressman, R. (2001): *Software engineering: A practitioner's approach*. Boston, McGraw Hill.

Randolph, G. (2004): Use-Cases and Personas: A Case Study in Light-Weight User Interaction Design for Small Development Projects. *Informing Science Journal* **7**: 105-116.

Smith, R. S. (2003): *Using Scenarios to Gather Requirements*, http://www.sonoma.edu/csse/edtech/560/scenarios.pdf. Accessed 13 Aug 2003.

Goal Directed Design: http://www.cooper.com/content/why_cooper/modeling.asp. Accessed 13 Aug 2003.

Nielsen's usability heuristics: http://www.useit.com/papers/heuristic/heuristic_list.html. Accessed 31 Mar 2004.

User-Centered Requirements: http://ccm.redhat.com/user-centered/user-centered-requirements.html. Accessed 12 Aug 2003.

Cecil: http://www.cecil.auckland.ac.nz. Accessed 30 Mar 2004.

Turnitin: http://www.turnitin.com. Accessed 30 Mar 2004.

9 Appendices

9.1 User-centred requirements using personas

- What personas are going to use this software component?
- What are the goals of these personas when using this software?
- For each primary persona, write all relevant scenarios, each telling the story of the persona achieving a goal.
- For each scenario, determine the individual tasks involved.
- Do the matrix of tasks and scenarios.

Source: RedHat http://ccm.redhat.com/user-centered/user-centered-requirements.html. 12 Aug 2003.

9.2 Ten Usability Heuristics

These are ten general principles for user interface design. They are called "heuristics" because they are more in the nature of rules of thumb than specific usability guidelines.

Visibility of system status
The system should always keep users informed about what is going on, through appropriate feedback within reasonable time.

Match between system and the real world
The system should speak the users' language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.

User control and freedom
Users often choose system functions by mistake and will need a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue. Support undo and redo.

Consistency and standards
Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.

Error prevention
Even better than good error messages is a careful design which prevents a problem from occurring in the first place.

Recognition rather than recall
Make objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.

Flexibility and efficiency of use
Accelerators -- unseen by the novice user -- may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.

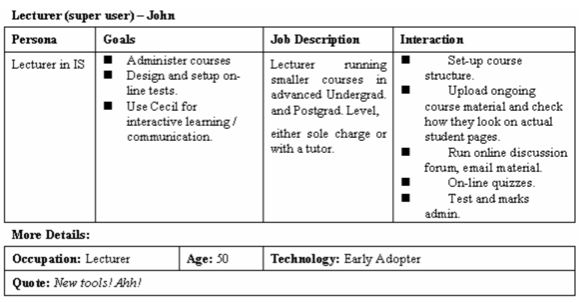
Aesthetic and minimalist design
Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.

Help users recognize, diagnose, and recover from errors
Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.

Help and documentation
Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large.

Source: Nielsen, J. http://www.useit.com/papers/heuristic/heuristic_list.html. 31 Mar 2004.

9.3 Cecil personas

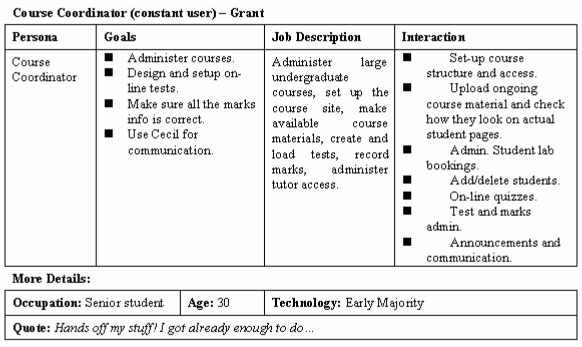


Lecturer (super user) – John

Persona	**Goals**	**Job Description**	**Interaction**
Lecturer in IS	■ Administer courses ■ Design and setup on-line tests. ■ Use Cecil for interactive learning / communication.	Lecturer running smaller courses in advanced Undergrad. and Postgrad. Level, either sole charge or with a tutor.	■ Set-up course structure. ■ Upload ongoing course material and check how they look on actual student pages. ■ Run online discussion forum, email material. ■ On-line quizzes. ■ Test and marks admin.

More Details:

Occupation: Lecturer	**Age:** 50	**Technology:** Early Adopter
Quote: *New tools! Ahh!*		

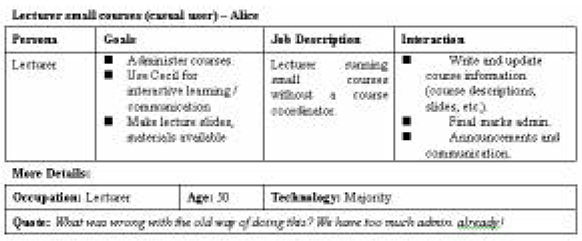


Course Coordinator (constant user) – Grant

Persona	**Goals**	**Job Description**	**Interaction**
Course Coordinator	■ Administer courses. ■ Design and setup on-line tests. ■ Make sure all the marks info is correct. ■ Use Cecil for communication.	Administer large undergraduate courses, set up the course site, make available course materials, create and load tests, record marks, administer tutor access.	■ Set-up course structure and access. ■ Upload ongoing course material and check how they look on actual student pages. ■ Admin. Student lab bookings. ■ Add/delete students. ■ On-line quizzes. ■ Test and marks admin. ■ Announcements and communication.

More Details:

Occupation: Senior student	**Age:** 30	**Technology:** Early Majority
Quote: *Hands off my stuff! I got already enough to do …*		

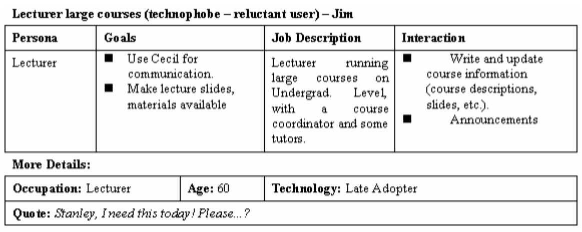


Lecturer small courses (casual user) – Alice

Persona	**Goals**	**Job Description**	**Interaction**
Lecturer	■ Administer courses. ■ Use Cecil for interactive learning / communication. ■ Make lecture slides, materials available	Lecturer running small courses without a course coordinator.	■ Write and update course information (course descriptions, slides, etc.). ■ Final marks admin. ■ Announcements and communication.

More Details:

Occupation: Lecturer	**Age:** 50	**Technology:** Majority
Quote: *What was wrong with the old way of doing this? We have too much admin. already!*		

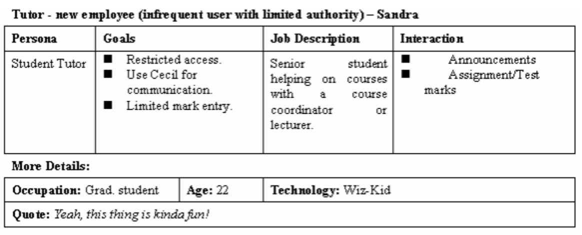


Lecturer large courses (technophobe – reluctant user) – Jim

Persona	**Goals**	**Job Description**	**Interaction**
Lecturer	■ Use Cecil for communication. ■ Make lecture slides, materials available	Lecturer running large courses on Undergrad. Level, with a course coordinator and some tutors.	■ Write and update course information (course descriptions, slides, etc.). ■ Announcements

More Details:

Occupation: Lecturer	**Age:** 60	**Technology:** Late Adopter
Quote: *Stanley, I need this today! Please…?*		



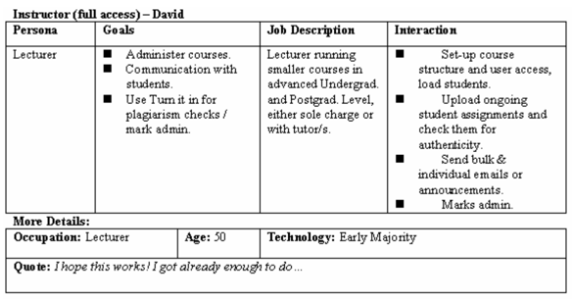
Tutor - new employee (infrequent user with limited authority) – Sandra

Persona	**Goals**	**Job Description**	**Interaction**
Student Tutor	■ Restricted access. ■ Use Cecil for communication. ■ Limited mark entry.	Senior student helping on courses with a course coordinator or lecturer.	■ Announcements ■ Assignment/Test marks

More Details:

Occupation: Grad. student	**Age:** 22	**Technology:** Wiz-Kid
Quote: *Yeah, this thing is kinda fun!*		

9.4 Turnitin personas

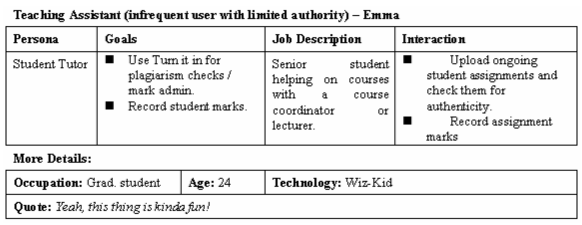


Instructor (full access) – David

Persona	**Goals**	**Job Description**	**Interaction**
Lecturer	■ Administer courses. ■ Communication with students. ■ Use Turn it in for plagiarism checks / mark admin.	Lecturer running smaller courses in advanced Undergrad. and Postgrad. Level, either sole charge or with tutor/s.	■ Set-up course structure and user access, load students. ■ Upload ongoing student assignments and check them for authenticity. ■ Send bulk & individual emails or announcements. ■ Marks admin.

More Details:

Occupation: Lecturer	**Age:** 50	**Technology:** Early Majority
Quote: *I hope this works! I got already enough to do …*		



Teaching Assistant (infrequent user with limited authority) – Emma

Persona	**Goals**	**Job Description**	**Interaction**
Student Tutor	■ Use Turn it in for plagiarism checks / mark admin. ■ Record student marks.	Senior student helping on courses with a course coordinator or lecturer.	■ Upload ongoing student assignments and check them for authenticity. ■ Record assignment marks

More Details:

Occupation: Grad. student	**Age:** 24	**Technology:** Wiz-Kid
Quote: *Yeah, this thing is kinda fun!*		