MuiCSer: A Process Framework for Multi-Disciplinary User-Centred Software Engineering processes

Mieke Haesen[1], Karin Coninx[1], Jan Van den Bergh[1] and Kris Luyten[1]

[1] Hasselt University -tUL -IBBT, Expertise Centre for Digital Media,
Wetenschapspark 2, 3590 Diepenbeek, Belgium
{mieke.haesen,karin.coninx,jan.vandenbergh,kris.luyten}@uhasselt.be

Abstract. In this paper we introduce MuiCSer, a conceptual process framework for Multi-disciplinary User-centred Software Engineering (UCSE) processes. UCSE processes strive for the combination of basic principles and practices from software engineering and user-centred design approaches in order to increase the overall user experience with the resulting product. The MuiCSer framework aims to provide a common understanding of important components and associated activities of UCSE processes. As such, the conceptual framework acts as a frame of reference for future research regarding various aspects and concepts related to this kind of processes, including models, development artefacts and tools. We present the MuiCSer process framework and illustrate its instantiation in customized processes for the (re)design of a system. The conceptual framework has been helpful to investigate the role of members of a multi-disciplinary team when realizing artefacts in a model-based approach. In particular process coverage of existing artefact transformation tools has been studied.

Keywords: User-Centred Software Engineering, User-Centred Design, Process Framework

1 Introduction

The perceived quality of the user experience of an interactive application is well emphasized nowadays. It has raised attention from the HCI community for user-centred design (UCD) approaches. Key issues in UCD processes that contribute to the overall user experience with the resulting product are continuous attention for the end-user needs, iterative (and possibly incremental) design and development, and a dominant presence of evaluation with respect to external quality attributes such as usability, accessibility and apparent performance [9]. UCD approaches have proven their value for interactive systems development for new as well as for legacy systems. We have the impression, however, that redesign of legacy systems places higher demands on the process being used due to the need to capture existing knowledge and reuse requirements from existing documentation. Besides analysis and design artefacts such as diagrams and models related to the application back

end, the running system itself and manuals are valuable sources. Because diagrams and models related to the application back end (e.g. UML diagrams) often result from a software engineering (SE) process, there is a search to combine basic principles and practices from the SE domain and UCD approaches in order to define an overall process that fulfils the needs of a multi-disciplinary design team. We coin the processes that unite both HCI and SE perspectives as "User-Centred Software Engineering Processes" (UCSE processes).

Based on former research results, we explore extensions of model-based user interface development approaches to bridge the gap with SE approaches such as model-driven development. A model-based approach typically employs different types of models, thereby conveying enough information to generate the skeletons for concrete user interfaces. Models still tend to emphasize facilitating the more technical phases in application development over the creative design phase and overall development cycle. Overcoming these shortcomings in a unified HCI and SE approach, and paying attention to multi-disciplinary teams are a necessity to allow for a pragmatic approach and applicability of model-based techniques in real-world projects.

To accommodate for both flexibility in selecting the techniques for one particular UCSE process and consistency in models in consecutive developments, we prefer starting from a conceptual process framework rather than a single, exhaustively defined UCSE process. The conceptual process framework can be considered as a generic process that can be customized or instantiated for the specific design task at hand. Though UCD research in the HCI community is focused on processes, process frameworks are gaining importance in the software engineering community (e.g. The Eclipse Process Framework[1]). Therefore, we believe this approach is helpful to strive at the same time for practical processes for applied research and for a comparison and evaluation framework, driving research activities regarding models, development artefacts and tools.

In this paper, we present our proposal for a UCSE process framework and detail the tools, models and artefacts that support the approach. This process framework has been the basis for two process instances employed during case studies, which are also used to summarize some lessons we learned. A discussion of our current and future work, as well as conclusions are presented.

2 The MuiCSer Process Framework

Comparable to several UCD approaches, our process framework for Multi-disciplinary user-Centred Software engineering processes, MuiCSer, focuses on the end-user needs during the entire SE cycle in order to optimize the user experience provided by the software that is delivered. The user experience is typically determined by measuring the usability, accessibility, availability of required functionality etc. of the delivered application.

[1] http://www.eclipse.org/epf/

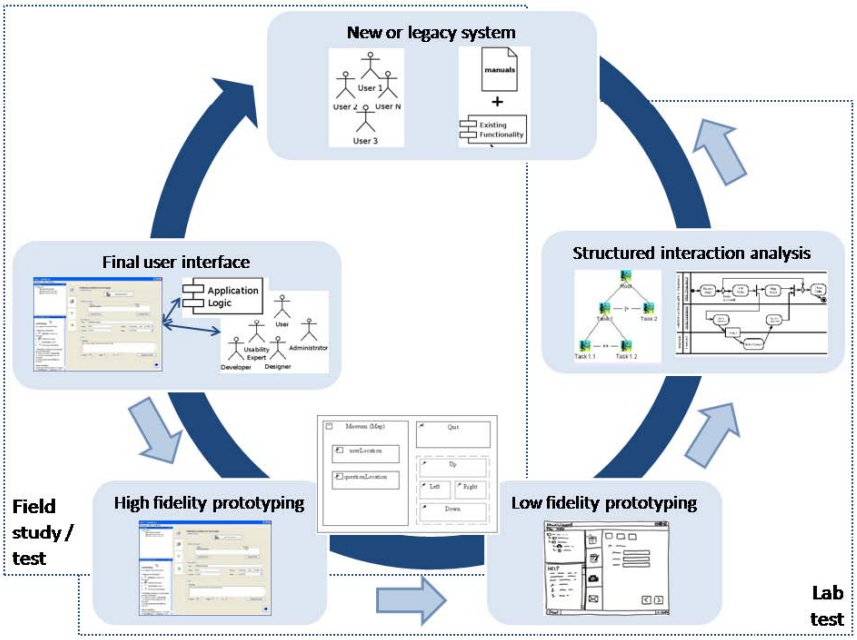

Fig. 1. Our MuiCSer process framework. The dark arrow indicates the overall design and development direction. The light arrows indicate feedback from evaluation, verification and validation efforts.

Based on our experiences and observations when working with multi-disciplinary teams, we are gradually introducing model-based processes in applied research and software development projects. Our conceptual process framework embodies UCD with a structured Agile Software Engineering (ASE, [11]) approach and organizes the creation of interactive software systems by a multi-disciplinary team. We will support different models throughout processes that are derived from the framework, where each model describes a specific aspect of an interactive system and represents the viewpoint of one or more specific roles in the multi-disciplinary team. The need for communication with end-users or customers results in additional models or artefacts (e.g. low-fidelity and high-fidelity prototypes) on top of the commonly used models in a model-driven approach. This has also a positive effect on the visibility and traceability of the processes that are based on our process framework, in particular when artefacts are stored in a central repository: the models and artefacts describe the status of the system being designed at various stages, support the design decisions made during these processes and are ready for use in the next iteration.

Fig. 1 gives an overview of the proposed process framework. Existing UCD approaches such as GUIDE [23], Effective Prototyping [1] and Rapid Contextual Design [8] can be represented using this framework. Likewise, when projects are carried out following a UCSE approach, the approach that is used, can be seen as a process that is created according to the MuiCSer process framework. Both functional

and non-functional requirements are tackled by the process framework and unlike traditional software engineering processes, it supports processes with a continuous and smooth integration between user interface design and software development. The next paragraphs discuss the properties of the process framework we propose in more detail.

MuiCSer processes typically start with an analysis phase in an initial iteration where the users tasks, goals and the related objects or resources that are important to perform these tasks are specified. If the user experience of a legacy system needs to be optimized, the functionality of such a system can be often found in existing manuals and also contributes to the analysis. Several notations are used to express the results of the analysis phase: HCI experts take a user-centred approach and commonly use domain-specific notations to express the task model and use personas to identify the user characteristics that are important. The software engineer specifies the required behaviour of the system with use cases and behaviour diagrams. Although the relationship between both is clear, linking them in an engineering process remains a difficult issue. However, when a process framework helps to define what artefacts are important in which stages and how progress from abstract to concrete models can be realized, this helps to identify, create and relate the required models in each stage.

During the structured interaction analysis, the results of the analysis are used to proceed towards system interaction models and presentation models. These models are often expressed using the UML notation, thus keeping in pace with the traditional SE models.

Since both user needs and functional information are specified, they can both serve as input for the low-fidelity prototyping stage, as is shown in Fig. 1. User interface designers create mockups of the user interface, based on the information contained in the task and interaction models, while using design guidelines and their experience. In subsequent phases, low-fidelity prototypes are transformed into high-fidelity prototypes, which on their turn evolve into the final user interface while each stage is related to the artefacts created in a previous stage.

By evaluating the result of each stage, the support for user needs and goals and the presence of required functionality is verified. If possible, an evaluation with target users can be very useful to get feedback from the end-user directly. Because most of the artefacts do not present a fully functional system, part of the testing takes place in a usability lab. To evaluate some advanced prototypes, field tests can examine the user interface in more realistic situations. If the results of a phase are not suited (e.g. too complex) to involve an end-user during evaluation, it is still necessary to evaluate, verify or validate the models or prototypes, e.g. in meetings with domain experts or by performing an expert evaluation.

3 Tools and Models

In this section we discuss to what extent MuiCSer can be covered by existing tools for the creation and transformation of artefacts and in what stages tool-support should be improved. The current use of tools also reveals how the collaboration within multi-

disciplinary teams is supported. Besides the discussion of tools, this section gives an overview of models that can be used in processes based on MuiCSer.

Table 1. An association of tools that can be used to support MuiCSer and their accessibility for different roles in a multi-disciplinary team.

		Tools																					
		Word processor [1]	Presentation [1]	Spreadsheet [1]	Drawing [1]	Paper [1]	PDF viewer [1]	Paint program [1]	Simple programming [1]	HTML (site) editor [1]	Animation tool [1]	Advance programming [1]	CTTE [17]	TaskSketch [3]	Vista Environment [2]	CanonSketch [3]	Teresa [18]	SketchiXML [7]	Damask [12]	GrafiXML [16]	Gummy [14]	IntuiKit [4]
Roles in a multi-disciplinary team	End-user					✓												✓				
	Purchaser, manager of user	✓	✓	✓	✓	✓	✓															
	Application domain specialist	✓	✓	✓	✓	✓	✓															
	Systems analyst, systems engineer, programmer	✓	✓	✓	✓	✓	✓		✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Marketer, salesperson	✓	✓	✓	✓	✓	✓															
	UI designer, visual designer	✓	✓	✓	✓	✓	✓	✓		✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Human factors and ergonomics expert, HCI specialist	✓	✓	✓	✓	✓	✓															
	Technical author, trainer and support personnel	✓	✓	✓	✓	✓	✓	✓														

3.1 Artefact transformation tools

The process framework described in the previous section has been used in practice to support several real-life cases. During the execution of the MuiCSer processes to develop these cases, some of which will be explained more into detail further in this paper, we observed what tools members of the project team used to contribute in the

different stages of these processes. This information in combination with literature that describes tools that fit in this process gave rise to Table 1. This table presents different roles which can be part of a multi-disciplinary team [1, 8, 9] and the tools associated with the role. The table shows that the leftmost tools are widespread and accessible for different roles of the multi-disciplinary team, which is confirmed by Campos and Nunes in [3].

Table 2. Overview of artefacts supported by artefact transformation tools.

		Artefacts										
		Scenario	Use case	Task model	Task Oriented Specification	Task Flow	Domain model	Activity Diagrams	User Interface Architecture	System Architecture	Abstract UI	Concrete UI
Artefact transformation tools	CTTE [17]	✓	✓	✓								
	TaskSketch [3]		✓			✓		✓		✓		
	Vista [2]			✓	✓		✓		✓			
	CanonSketch [3]						✓				✓	✓
	Teresa [18]			✓							✓	✓
	SketchiXML [7]			✓			✓				✓	✓
	Damask [12]											✓
	Gummy [14]											✓
	GrafiXML [16]											✓
	IntuiKit [4]											✓

Table 2 provides an overview of a selection of these tools and their applicability for creating artefacts that are used in the HCI engineering process. We use the term *artefac*t *transformatio*n tool to describe a tool that can be used by two or more different roles and supports relating two artefacts or models. Such a tool allows to progress the design and development of an interactive system involving different roles, often by providing different views on the same artefact or model. The ways in which a tool can manipulate, create relations or transform between artefacts and models are summarized in [5].

Mapping these tools on the stages of MuiCSer (Fig. 1) results in the time-line shown in Fig. 2. Most tools that are suitable for interactive, incremental and multi-disciplinary user-centred processes are artefact transformation tools which comes as no surprise. Fig. 2 also shows that it is possible to combine two or three tools to cover most stages of MuiCSer. While Teresa [18] can be used to model tasks of a multi-

platform application and generate a system task model, an abstract user interface and a concrete user interface, Gummy [14] can be used by designers to add creative aspects to the medium-to high-fidelity prototypes for multi-platform user interfaces.

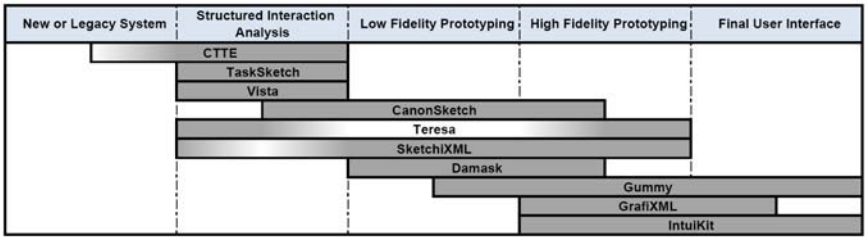

Fig. 2. A timeline presenting the stages of MuiCSer and how artefact transformation tools can be mapped on it.

The overview of tools in Fig. 2 also reveals that there is little tool support for the transformation of the results of user studies into structured models. Furthermore, when a new iteration takes place after a final user interface is deployed, there is no single tool that completely covers MuiCSer. The main drawbacks of most of these tools are their inaccessibility for non-experts and their relative immaturity for real-world software development processes. Several of the aforementioned tools are being increasingly used in industrial projects, so we expect this situation will improve rapidly. SketchiXML for instance is already suitable to be used by a wider range of roles including designers and end-users [7]. Gummy supports the roles of software developers and designers but this tool is gradually being extended to be used by application domain specialists [13].

The following describes different models being created, changed and transformed during the execution of MuiCSer processes in order to support a smooth transfer towards the final user interface. The models and tools discussed in the remainder of this section are not required. They provide a clear idea of how MuiCSer processes can be instantiated with concrete models, notations and tools.

3.2 Structured Interaction Analysis

Task models are frequently used to specify requirements for an application from a user's point of view. Most task models have an hierarchical structure, allowing a gradual refinement of the high-level tasks and goals into fine-grained actions and activities. A task specification for a system can be found by transforming the requirements text and the scenarios of the personas into a hierarchical task model with temporal operators, such as the ConcurTaskTrees notation. Although this step is not automated, the expert performing this step uses a set of (informal) rules and is supported by a tool such as CTTE [17].

This task model can be related to other user interface and software engineering models expressed using e.g. UML diagrams, which are widely known by software analysts and programmers. These user interface models can provide an alternative

view on the information captured in the task model [20, 24] or additional information [21, 24].

3.3 Low-Fidelity Prototyping

Since the creativity of designers and other members of a multi-disciplinary team may influence the user experience in positive way, MuiCSer does not imply the use of specific tools or technologies to create low-fidelity prototypes. The first prototypes can be created using pencil and paper or using a tool. Tools such as SketchiXML [7] or CanonSketch [3] have the advantage that they provide support for the transition to high-fidelity prototypes. This ability to make the transition from low-fidelity to high-fidelity using these tools and notations is illustrated by the drawing between the low-fidelity and the high-fidelity stage in Fig. 1.

3.4 High-Fidelity Prototyping

For the high-fidelity prototyping stage, design and development tools that support serialisation of the user interface design to (high-level) XML-based languages are preferred. This allows more rapid prototyping of user interfaces that support a common set of tasks. Tools such as Gummy [14] or GrafiXML [16] even have specific support for adapting the designs to different platforms, screen sizes or in general different contexts of use. A loose coupling with the application logic is preferred to enable reuse.

3.4 Final User Interface

To speed up development of the final user interface and to make it as flexible as possible, we preferably reuse as much as possible of the developed artefacts, such as the XML-based high-fidelity prototypes and even selected models. A flexible user interface management system allows the use of these models at runtime. Coupling for example the task model to the user interface descriptions allows to check for task coverage of the user interface and even selection of a subset of features for certain users while ensuring that all remaining tasks are still valid. Using these artefacts in the final user interface also ensures that they are still available and up-to-date for the development of future increments.

4 Case Studies

We explain how MuiCSer can be used by describing two MuiCSer processes that are customized for two cases, carried out within the VIP-lab project [6] The first case study concerns the redesign of a legacy system while the second case study presents the approach that has been used for the design of a new system. The project team was not limited to computer scientists but also psychologists and social scientists were

involved and in some cases a graphic designer. Fig. 3 shows an overview of the MuiCSer processes that are employed for these cases. For sake of clarity of the presentation and to allow comparison, both processes are shown as a linear path without emphasis on the intra-and inter-stage iterations.

4.1 NewsWizard

When a reporter is on location, he or she not only has to write an article. The biggest challenge is often to configure a network connection to send the article to the editorial staff. The NewsWizard prototype, developed in this case study, should ease the job of a journalist on location by guiding him / her while making the appropriate network connection and sending the article(s).

As recommended by MuiCSer, first the legacy system has been explored. Manuals of the existing editor to write and send articles have been studied and the system was demonstrated to the project team. Next journalists and photographers were observed and interviewed by social scientists while they were collecting information and sending it to the editorial office. Besides the comparison of the job of a contemporary journalist and a photographer, this contextual inquiry resulted in primary and secondary personas [22] and scenarios (Fig. 3 I.a, tool: word processor and PDF viewer).

At the second stage of this process which concerned the structured interaction analysis, some task models were created by developers using the Hierarchical Task Analysis (HTA) and CTT notation (Fig. 3 I.b, tool: drawing tool and CTTE). The verification of these task models was carried out during meetings with the project team. The social scientists checked consistency with the observations, the personas and the scenarios while the computer scientists examined the technical possibilities and representatives of the news publishing agency verified the design according to the needs of the journalists and their own expectations. The task models were refined within two iterations. The threshold for progression is the agreement of the domain experts and stakeholders on structure and content of the task model, scenarios and personas.

By putting together the results of the user and task analysis and the structured interaction analysis, it became clear that journalists mainly experience problems when they need to send an article on location. Consequently a user interface in wizard-style was designed to collect articles and pictures (in case the journalist is not accompanied by a photographer), followed by sending the data successfully. The relations between the task model and the low-fidelity prototype on paper were determined manually and the prototype was checked for completeness with respect to the task model during meetings, similar to the meetings held during the structured interaction analysis stage.

In order to have a prototype that could be evaluated by journalists in a usability lab, soon the low-fidelity prototype of NewsWizard evolved into a high-fidelity prototype (Fig. 4, tool: advance programming tool). Although this was done manually, there is a clear one-to-one mapping from each component in the low-fidelity prototype to each component in the high-fidelity prototype. By consequence, the high-fidelity prototype is also complete with respect to the task model. In three iterations and increments the

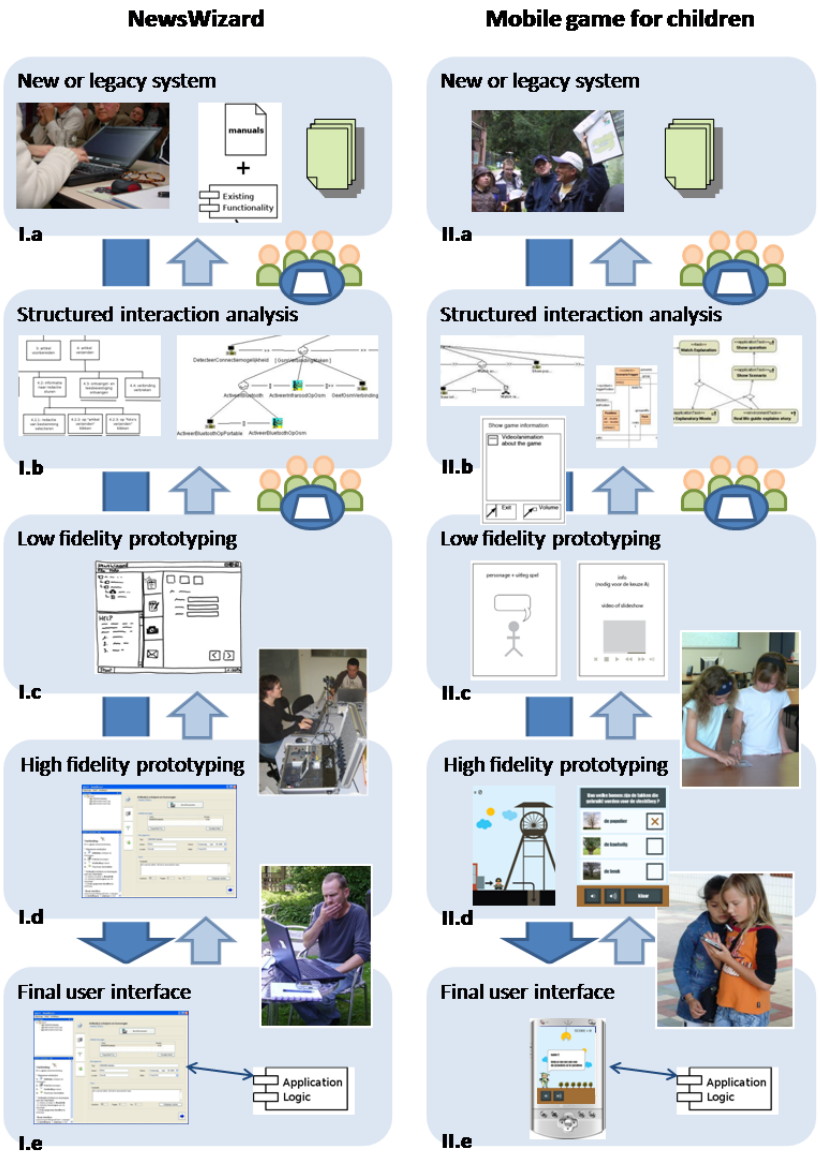

Fig. 3. MuiCSer process instances of the NewsWizard and the mobile game for children. In both processes the verification between steps a and b, and steps b and c was done during brainstorm meetings within the multi-disciplinary team, while the evaluation in later stages involved end-users during lab and field tests.

NewsWizard prototype was developed and functionality was added. After each iteration and increment the UI was evaluated by journalists in a portable usability lab (Fig. 3 I.d). In order to evaluate the prototype in the natural environment of a reporter, some field tests were carried out (Fig. 3 I.e). During the field tests, the participating journalists were observed and interviewed while accomplishing a realistic assignment on location using NewsWizard. The general observations showed that the use of NewsWizard was much more intuitive than using the existing system. Most of the journalists confirmed that in the future they would rather send articles from location instead of going back to their desk if they could use the NewsWizard application.

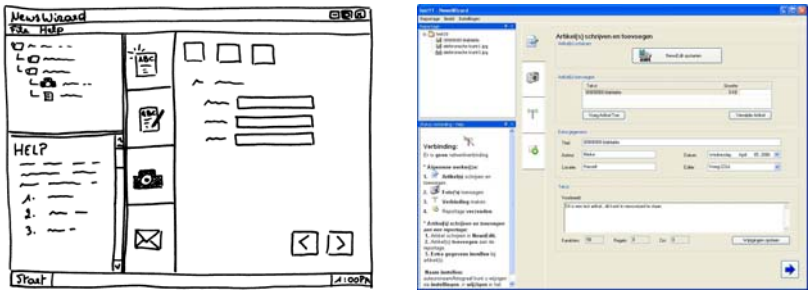

Fig. 4. Low-and high-fidelity prototype of the NewsWizard interface. The main part of the user interface concerns the wizard. The user can navigate between steps using arrow-buttons or tab pages.

4.2 Mobile game for children

A second case study concerns the development of a prototype for a mobile game, and was carried out in collaboration with local cultural and tourist organizations. The goal of this game for children is to make educational excursions more interesting and informative.

Since a new system had to be developed in this case study, it was impossible to examine manuals and existing functionality. Mainly results from a user and task analysis could contribute to the structured interaction analysis. During the user and task analysis school groups were observed and interviewed while they were visiting museums and zoos. It turns out that the addressed target users prefer being guided throughout the visit in a narrative style, based on a story they can identify themselves with. After several brainstorm sessions, the multi-disciplinary team including a graphic designer and representatives of cultural and tourist organisations, came up with two game concepts for a PDA application (Fig. 3 II.a, tool: word processor and PDF viewer). The goal of one game is to save the trees in a nature resort, while the other game challenges children visiting a mine museum to help a mine worker to have a safe working day. Scenarios ensured that all team members had the same understanding of the game to be designed (tool: word processor and PDF viewer).

The game scenarios proved to be very useful to structure the user tasks and to create a task model using the CTT notation (Fig. 3 II.b, tool: CTTE). Even though

both games are totally different, the same user interface components would be necessary. This resulted in the decision to create a general framework containing the application logic for both games.

Besides the task model, other HCI engineering models were created to present the relation between the user interface and the application logic (Fig. 3 II.b, tool: drawing tool). The application model ensured the application logic would be suitable for both games. The system interaction model, based on the user task and application model gives an overview of the flow of actions carried out by the system and the user. The abstract presentation model, is based on the preceding models and represents the user interface components, which can be used in a Canonical Abstract Prototype (CAP) [10]. This CAP (Fig. 5, tool: CanonSketch) is a first graphical representation of the functional parts of the user interface, independent of the content or the story that would be used in the game. During the verification of the models, the scenarios were used to ensure the models did meet the requirements of the game. After the computer scientists created these models, the task was handed over to the graphic designer. He translated the CAPs into some low-fidelity prototypes, which evolved into a design of the prototypes for both games (Fig. 5, tool: paint program) after adding layout and style information.

Fig. 5. Three levels of prototypes for one specific screen. From left to right: a Canonical Abstract prototype, a low-fidelity prototype and a high-fidelity prototype.

In order to get some early feedback of the end-users the prototypes were interactively tested in a lab environment with materials similar to what is being used in participatory approaches such as PICTIVE [19] (Fig. 3 II.d). The tests showed children were amused by the game, but revealed problems concerning the size and behaviour of buttons and the content.

Based on the test results, the design of the user interface was adjusted (tool: animation tool), while the models of the structured interaction analysis were used for the development of the application logic of the game (tool: advance programming tool). The resulting high-fidelity prototypes were evaluated by children in a nature resort and a mine museum. During these field tests few user interface problems were detected, so we may conclude that the model-based approach, and the evaluation in early stages influenced the high-fidelity prototype in a beneficial way.

5 Lessons Learned

The case studies presented in section 4 were carried out using MuiCSer processes. In the NewsWizard case a MuiCSer process was used for the redesign of an existing system, while the second case study concerned the design of a new system. In both case studies we experienced that it was hard to structure the information to get a complete overview of the user needs. Since the usage of personas and scenarios implies partially structured narrative information, it was necessary to transform the information into some task models. These task models made it possible to abstract the most important goals of the future prototype. By doing so, some information contained in the personas and scenarios could be overlooked. Therefore, the task models were evaluated during meetings with the computer scientists and team members with other roles.

By carrying out different case studies we had the opportunity to fine-tune the approach in our multi-disciplinary team. In the NewsWizard case study it became clear that task models were understandable for all team members and thus could be evaluated during meetings. On the other hand computer scientists experienced that the information of task models was insufficient for the development of the high-fidelity prototypes. During the structured interaction analysis and prototyping of the mobile game, models presenting the links between the user interface and the application logic were helpful to get more insight into the functional requirements. Furthermore, these models evolved gradually into a first graphical representation, the CAP, which was also presented to the graphic designer.

The low-fidelity prototypes of both case studies were created by putting together the artefacts of earlier stages in MuiCSer. The design of the first prototypes was discussed and evaluated during meetings attended by the multidisciplinary team.

End-users were asked to participate in the evaluation of high-fidelity prototypes. Our experience from other case studies learned us that field tests give more information on the entire user experience. By evaluating a prototype in the natural environment of the end-user, a broader user experience is taken into account and context dependent actions can be observed.

When comparing the processes shown in Fig. 3 we discover that both are in line with the MuiCSer framework from the start where the user studies take place, until the high-fidelity prototyping phase. Several artefacts were created as a result of the process stages. This illustrates the fact that the MuiCSer framework suggests some models and artefacts, but that the design team decides about the particular results for the customized process at hand. All artefacts proved useful to convert artefacts in the next phase. The conversion of these artefacts required some human intervention that is difficult or impossible to automate.

The creation, evaluation, verification and validation of the artefacts, was carried out using several tools. The computer scientists and designers used CTTE, CanonSketch, drawing tools, animation tools and advance programming tools for the development of HCI models and coded prototypes. Widespread tools such as pencil and paper, a word processor and a PDF viewer were useful for the other artefacts as the entire project team, including representatives of the participating companies, was familiar with these common tools.

6 Ongoing and Future Work

The process framework introduced in this paper has been tested on software projects of limited complexity and, by consequence, with a development team of limited size. Although our tests did not include any larger software projects, customized processes derived from this framework should be flexible enough to support the increased complexity and team size, partly because parameters such as size of increments, number of iterations, specific models and artefacts are decided about when instantiating the process from the framework. Currently we are investigating how a process instantiated by MuiCSer can be used to model and design adaptable user interfaces for heterogeneous environments [15].

One of the main advantages of the openness of the framework with respect to specific techniques is that different domain experts can use their own notations to create models which can relate to models of other domain experts, in order to obtain a complete and usable interactive system with respect to the requirements. We are testing this conceptual framework for processes supporting multi-disciplinary teams in various application domains, requiring different experts to collaborate. Besides the relationship with existing UCD processes, we will investigate how software engineering processes fit into our framework. These research activities, including application of derived processes and generalization of existing processes for comparison, will give rise to enhancements or extensions of the framework.

Central storage of models and artefacts as well as manual and system-guided transitions between these products turn out to be key factors for the efficiency of the processes and acceptability by the design team. Therefore, the design and creation of a flexible user interface management system (UIMS) that is able to use XML-based user interface descriptions and models is an integral part of our current work [25]. In order to support this UIMS we plan to gradually improve the relation between the different types of artefacts. The combination of HCI models and UML models contributes to a smooth integration of the user interface and application logic. Putting forward the combination of models explicitly also prevents mismatches between the functionality provided by the application logic and the functionality accessible through the user interface.

7 Conclusions

In this paper we introduced MuiCSer, a novel process framework, practicing Multi-disciplinary User-Centred Software engineering in such ways that methodologies used by developers as well as the creativity of developers are included and a positive user experience is more likely to be obtained. Each iteration of a MuiCSer process produces one or more prototypes to enhance the visibility of this process and to allow continuous user involvement and evaluation. Through the case studies, we found the explicit support for multi-disciplinary teams in our process framework one of the strong points of our approach. The definition of the framework stimulates the use of customized processes that pay explicit attention to consistency of design and development artefacts throughout the different cycles of the process.

Multi-disciplinarity has been a focus in the current instantiations of the MuiCSer framework and will get additional attention in future research activities in this area. Extending and fine tuning the framework by deriving new and existing processes, will make it a better reference for process comparison and evaluation. Together with the user-interface management system being developed, this will encourage systematic studies of requirements for supporting tools for UCSE processes.

Acknowledgments. Part of the research at EDM is funded by the ERDF (European Regional Development Fund) and the Flemish Government. The VIP-lab project (4-BMG-II-2=37), is financed by the "Interreg Benelux-Middengebied" authorities and co-financed by Province of Limburg (B), Province of Limburg (NL), Ministry of Economic Affairs (NL) and Ministry of Flemish Government/Economic Affairs (B). The *MuiCSer* Process Framework is also based on our experiences in IWT projects Participate (with Alcatel-Lucent) and AMASS++ (IWT 060051).

References

1. Arnowitz J., Arent M., Berger N.: Effective Prototyping for Software Makers (The Morgan Kaufmann Series in Interactive Technologies). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2006)
2. Brown J., Graham N., Wright T.: The vista environment for the coevolutionary design of user interfaces. In Proc. CHI '98, pp. 376--383, ACM Press/Addison-Wesley Publishing Co. New York, USA (1998)
3. Campos P. and Nunes N.J.: Practitioner tools and workstyles for user-interface design. IEEE Software, vol. 24(1), pp. 73--80 (2007)
4. Chatty S., Sire S., Vinot J., Lecoanet P., Lemort A., Mertz C.: Revisiting visual interface programming: creating GUI tools for designers and programmers. In Proc. UIST '04, pp. 267--276, ACM, New York, USA (2004)
5. Clerckx T., Luyten K., Coninx K.: The mapping problem back and forth: customizing dynamic models while preserving consistency. In Task Models and Diagrams for User Interface Design, pp. 33--42 (2004)
6. Coninx K., Haesen M., Bierhoff J.: VIP-lab: A virtual lab for ICT experience prototyping. In Proc. Measuring Behavior 2005, pp. 585--586 (2005)
7. Coyette A., Kieffer S., Vanderdonckt J.: Multi-fidelity prototyping of user interfaces. In Human-Computer Interaction - INTERACT 2007, 11th IFIP TC 13 International Conference, pp. 150--164 (2007)
8. Holtzblatt K., Burns Wendell J., Wood S.: Rapid Contextual Design. A How-To Guide to Key Techniques for User-Centred Design. Morgan Kaufmann Publishers (2005)
9. International Standards Organization: ISO 13407. Human Centred Design Process for Interactive Systems. Geneva, Swiss (1999)
10. Constantine L.: Canonical abstract prototypes for abstract visual and interaction design. In Proc. DSV-IS 2003, volume 2844 of LNCS, pp. 1--15. Springer (2003)
11. Larman C.: Agile and Iterative Development: A Manager's Guide. Addison-Wesley (2003)
12. Lin J., Landay J.A.: Employing patterns and layers for early-stage design and prototyping of cross-device user interfaces. In Proc. CHI '08, April (2008)
13. Luyten K., Meskens J., Vermeulen J., Coninx K.: Meta-GUI-builders: Generating domain-specific interface builders for multi-device user interface creation. In CHI '08: extended abstracts on Human factors in computing systems, New York, USA, ACM (2008)

14. Meskens J., Vermeulen J., Luyten K., and Coninx K.: Gummy for multi-platform user interface designs: Shape me, multiply me, fix me, use me. In Proc. AVI '08 (2008)
15. Meskens J., Haesen M., Luyten K, Coninx K.: User-Centered Adaptation of User Interfaces for Heterogeneous Environments, To appear in Advances in Semantic Media Adaptation and Personalisation - CRC Press Edited Book, (2008)
16. Michotte B., Vanderdonckt J.: A multi-target user interface builder based on UsiXML. In Proc. ICAS2008, Los Alamitos, USA, IEEE Computer Society Press (2008)
17. Mori G., Paternò F., Santoro C.: CTTE: support for developing and analyzing task models for interactive system design. IEEE Transactions on Software Engineering, vol. 28(8), pp. 797--813 (2002)
18. Mori G., Paternò, Santoro C.: Design and development of multidevice user interfaces through multiple logical descriptions. IEEE Transactions on Software Engineering, vol. 30(8), pp. 507--520 (2004)
19. Muller M.J.: Pictive – an exploration in participatory design. In Proc. CHI '91, pp. 225--231, ACM Press, New York, USA (1991)
20. Nobrega L., Nunes N.J., Coelho H.: Mapping ConcurTaskTrees into UML 2.0. In Proc. DSV-IS 05, vol. 3941, pp. 237--248, Springer (2005)
21. Nunes N.J., e Cunha J.F.: Towards a UML profile for interaction design: the wisdom approach. In The Unified Modeling Language. Advancing the Standard, vol. 1939, pp. 101--116, Springer (2000)
22. Pruitt J., Adlin T.: The Persona Lifecycle : Keeping People in Mind Throughout Product Design, Morgan Kaufmann (2006)
23. Redmond-Pyle D., Moore A.: Graphical User Interface Design and Evaluation. Prentice Hall, London (1995)
24. Van den Bergh J., Coninx K.: Towards Modeling Context-Sensitive Interactive Applications: the Context-Sensitive User Interface Profile (CUP). In Proc. SoftVis '05, pp. 87--94, New York, USA, ACM Press (2005)
25. Van den Bergh J., Haesen M., Luyten K., Coninx K., Notelaers S.: Toward Multi-disciplinary Model-Based (Re)Design of Sustainable User Interfaces. In Proc. DSV-IS 2008, Springer (2008)