Agile Development of Interactive Software by means of User Objectives

Begoña Losada, Maite Urretavizcaya, Isabel Fernández de Castro
Dept. of Computer Languages and Systems
Faculty of Computer Engineering - University of the Basque Country
20001 San Sebastián
{b.losada, maite.urretavizcaya, isabel.fernandez}@ehu.es

***Abstract*—Agile methods, model-driven developments and user-centred design are three approaches widely accepted in the development of interactive software. In this paper we present InterMod, a new approach that integrates all three methods. The project planning is based on *User Objectives* and the process is organised as a series of iterations, where the work is distributed in different workgroups according to some developmental and integration activities, each one driven by models. The requirements are incrementally collected and evaluated with models based on user-centered design. To speed up this validation, we put forward the *SE-HCI model*, which enriches a human-computer interaction model with the semantics of the application and some basic characteristics of an abstract prototype. This allows gather and validate the requirements incrementally. Moreover, this iterative process speeds up the development and generates results from the project progress.**

Keywords-Software Engineering; Agile method; User-Centered Design; Model-Driven Development.

I. INTRODUCTION

Currently, Agile Methods (AM) and Model-Driven Development (MDD) are the predominant approaches in Software Engineering. AM are able to develop a software product incrementally and iteratively. They get feedback from the client at each incremental delivery and, as a result, adapt the development plan accordingly. Most studies report increased code quality when agile methods are used but they also report a lack of attention to design and architectural issues [1]; moreover, it must be noted that in the area of Software Engineering quality software comes from good design.

Current trends establish design as final product models characterised by iterative and incremental development while at the same time promoting formal development along the lines of traditional or waterfall methodologies. Some authors [2] point out that a drawback of MDD is that the models are difficult to maintain, because as a project progresses changes come up and new requirements are added.

On the other hand, in the area of Human-Computer Interaction, User-Centered Design (UCD) is the dominant approach. Under UCD, the end user is involved in the process of multidisciplinary development based on iterative design and evaluation so the designer understands the user's needs and tasks [3][4]. But, as is the case with traditional or heavyweight methodologies, with UCD all requirements must be gathered and evaluated before they are implemented [5][6][7].

To make up for the weaker aspects of these proposals, efforts are being made to integrate agile methods into both model-driven design [8][9], and into UCD [10]. However, due to the fact that a majority of software engineering development processes focus on software architecture, satisfactory integration has not yet been achieved. Therefore, we focus our efforts on integrating these three techniques and we base our methodology in user-centered models starting from requirements gathering.

The main contributions of the paper can be summarised as follows:

c1. We propose a new approach to improve *Software Development* by applying *User Centered* and *Model-Driven Development* in an *Agile* manner.

c2. A new integrated model, involved in a Model Driven Process, to support the project requirements, is presented: the SE-HCI model. It facilitates usability and other kinds of incremental evaluation, tested by a multidisciplinary team of developers and users, just as proposed by UCD.

c3. Finally, we present an agile methodology organised as a series of iterations by means of *User Objectives* (UO) as a new way to promote a correct development. This iterative approach guides the incremental development of software.

Our paper is structured as follows. Section 2 outlines the primary characteristics of agile methods and how they compare to the other abovementioned approaches. Section 3 presents our proposal, situating it in the context of related work. We explain phases and development activities of our approach and its model structure, especially for the requirements model, and we show graphically a project iteration example. Finally, we draw some conclusions and outline our future work.

II. AGILE SOFTWARE DEVELOPMENT: VIRTUES AND DEFECTS RELATIVE TO OTHER APPROACHES

Agile software development establishes the following as principles [11]: Individuals and interactions over processes and tools, working software over comprehensive documentation, customer collaboration over contract negotiation, and responding to change over following a plan.

 ISBN: 978-1-61208-165-6

This approach challenges waterfall or heavyweight methods in which one activity begins only when the previous one finishes and where extensive and well-founded documentation is required. The rationale behind these traditional methods is to reduce the number of corrections further on in the process and consequently reduce the cost of the project. However, in practice this type of planning fails, as it doesn't allow the changes that inevitably come up during development [12]. Because of this, versions of the agile philosophy such as eXtreme Programming (XP) [13], Scrum [14], Crystal [15], Feature Driven Development (FDD) [16], UP [17] and others, currently prevail.

A. Agile processes and user-centered design

One of the important aspects of UCD is the collaboration between users and developers in building software solutions, each one bringing their experience to bear [18].According to Norman [19], it is first necessary to think about the needs of those who will be using the product that is being created in order to model that information, and then iteratively evaluate the product with users. Thus, the intention is to improve the product's usability such that it is easy to learn, it is easy to use, errors are reduced and users are satisfied, as defined in ISO standard 9241-11 [20].

Both proposals centre on the user/client and propose an iterative development process. These contrast with traditional architecture-based development processes, which are directed by the developers, who structure and control the users' activities.

Nevertheless, the differences between UCD and AM are great in terms of how they act and what their interests are [10]. On the one hand, the flexibility in action when faced with changes that the agile philosophy recommends is at odds with interface design prior to implementation (up-front), according to the principles of UCD. On the other hand, UCD develops a holistic product, while the agile process results in subproducts in an incremental process. And while agile methods focus on code development, UCD methods focus on the design of the interaction that users will engage in.

Finally, it must be noted that both approaches seek to satisfy the users' needs. However, in AM users are involved in checking that the functionality has been correctly implemented, while in UCD users give input regarding other aspects such as user satisfaction or efficiency of use for the whole application. UCD focuses on how end users work with the system, whereas AM is more concerned with how software should be built or how the process is managed.

B. Agile processes and model-driven development

In MDD, models serve principally as documentation and guidance for the subsequent implementation phase. Although building models is very useful in other areas of engineering, in Software Engineering there is great apathy toward building and using models. Many developers think that modelling demands the creation of excessive and extensive documentation, which ultimately is of little help when it comes time to implement and maintain the system [2]. This is because the changes that arise throughout development make these models difficult to update. In fact, many developers skip the model redesign phases and prefer to modify the code directly.

From this point of view, we have two issues that strongly conflict in software development. On the one hand, MDD needs to maintain model consistency as changes come up during application development. That is to say, our system will be more flexible if the model that represents it is an accurate and updated abstraction of itself [21]. On the other hand, due to unforeseen changes, AM perform modifications on the implementations that are not reflected in the designs. Therefore, if the constructed model does not correspond with reality and our code was initially generated from the model, this could spell failure for the project.

III. INTERMOD, AN INTEGRATED PROPOSAL

InterMod [22] is a methodology whose aim is to help with the accurate development of interactive software. Although it is suitable for use with web design, its utility is not restricted to just that area. Our latest studies have led us to place a new focus on the methodology by integrating an agile process with the other two philosophies namely, UCD and MDD, already present in our previous work. Also a new vision of the Requirements Models together with the SE-HCI model and the User Objective, to guide the process, are included in this paper.

Our proposal is the following: organise the project as a series of iterations, just as the agile methodologies do, and distribute the work in the iterations according to different developmental activities of the *User Objectives*. A *User Objective* (UO) is a user desire e.g. "*buying a t-shirt"* or "*reserving a meeting room in a workplace",* that can be achieved by one or more user functionalities. These are defined by means of the possibilities that the end user will perform in the application interface.

The Feature-Driven Design approach (FDD) [16] also uses MDD and divides the labour into different features (e.g. "*calculate the total of a sale"* or *" add a new customer to a customer list"*) to see measurable progress of the project. The functionalities implied in our UOs are always direct user's intentions, whereas the features can be user's or system's needs. In FDD, use cases obtain the features that allow the domain objects to be modelled (class diagram and the operations required in the system). However, our primary goal is not to model the domain objects but rather to model the tasks (user actions in the interface), navigation (action/reaction between the user and system) and presentation (visual aspects). We focus the development from the UCD perspective, as a new vision that obtains partially the interface before implementing the business logic. Once the objectives have been evaluated in terms of testing and usability of requirements, our proposal naturally ties in with the FDD perspective to model the domain objects.

InterMod has four main steps, i.e. the initial step *Analyse Overall Project* takes place at the project beginning, and then an iterative process with three steps follows: *Build User Objectives List*, *Plan Parallel Iteration* and *Perform Iteration Activities*.

 ISBN: 978-1-61208-165-6

A. InterMod steps: Activities and Models

Fig. 1 shows a scheme of the InterMod process and the models associated with it.

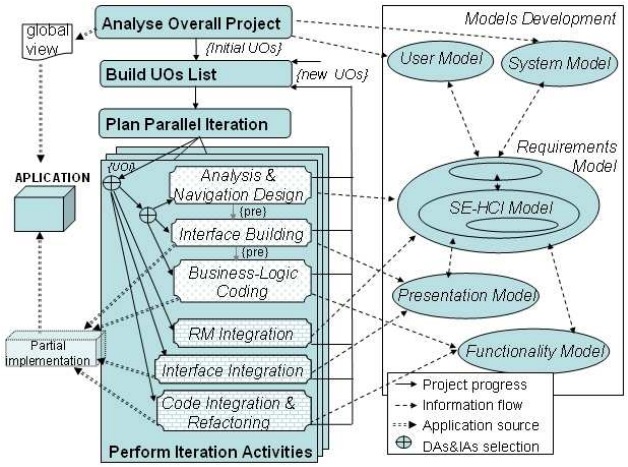

Figure 1. InterMod process and development activities

At the beginning of the project, it is necessary to analyse it as a whole in order to determine: (a) what the starting UOs are, and (b) the design decisions that will guide and give coherence to this iterative and incremental process. InterMod proposes the **Analyse Overall Project** step to achieve these challenges. The starting UOs (such as those most important or needed), together with a provisional general menu incorporating some functionalities, provide the global view of the application. And this analysis draws up the models that help to collect the defining characteristics of the system type (e.g. device type, security, window size, colour, logo, etc) and those of the user (e.g. colour preferences, font, size, some limitations as colour blindness, deafness, vision loss, etc). These characteristics are collected in the *System Model* and the *User Model* respectively. All developments in the project will inherit, supplement or extend these models in order to guide and ensure coherence throughout the entire application.

The application requirements are incrementally collected during the progressive UO List construction. Each iteration begins with a revision of the UOs list. The ***Build User Objective List*** step updates the list with the new UOs derived either from previous UO developments or from the new needs of the project. It is possible that a UO breaks on two or more new UOs because of its complexity; on the contrary, some UOs may be merged in one new integrated UO because of its simplicity. That is, the UOs included in the list may be modified, in the sense of agile methodologies [23], through the different evaluations undertaken by developers and users, or by the continuous meeting among members of the same and different teams.

In order to achieve a UO, different activities must be realised. The next step, ***Plan Parallel Iteration***, decides for the current iteration:

a) what UOs to develop

b) what activities to make for those UOs

c) how to distribute these different activities to the workgroups (if there is more than one).

The iteration ends with the **Perform Iteration Activities** step. Each workgroup performs the activities established in its plan.

InterMod has two kinds of **Activities**: Developmental Activities and Integration Activities.

The **Developmental Activities** (DAs) associated with each UO are strongly related:

- A1.Analysis and Navigation Design
- A2. Interface Building
- A3. Business-Logic Coding.

Just as UCD recommends, before coding a relevant UO, its interface must be validated. However, unlike UCD, it is not required that the complete application interface be developed before moving to the implementation of the business logic; instead this approach stays framed in the development of one or several UO groups. That is, each UO requires the three DAs to be developed but a prerequisite relation must be done A1< A2< A3 ('<' means prerequisite). A1 has not got any prerequisite activity. A DA of a User Objective is possible to deal with if and only if the UO is in the UO list and its prerequisite is achieved.

Furthermore, to assure a correct incremental progress of the project, some **Integration Activities** (IAs) are needed:

- I1.Requirement Models (RM) Integration
- I2.Interface Integration
- I3.Code Integration & Refactoring

A restriction is necessary for controlling the correct development of an IA. Thus, it is possible to carry out an IA I_k (K=1..3) for a concrete UO_j (j=0..n) if and only if the UO_j is the fusion of two UOs belonging to the UO List and the DAs A_k of these fused UOs are already made. To ensure consistency in the final application, evaluations of the incrementally obtained products as well as heuristic and metric evaluations are included in all activities.

All iterations are guided by the same action plan that divides the work according to the activities of different UOs, in such a way that each DA will be next driven by models and all the integration processes can lead to the revision and modification of these models. Even during final integration of the software there may be revisions of all models and new UOs can be created.

The activities of **analysis and navigation design** and **RM integration** deals with the *Requirements Model* (RM), which includes the *Semantically Enriched Human-Computer Interaction* (SE-HCI) *model* (more detail in section III.B). In the **Interface Building activity**, the *Presentation Model* is created for a UO previously designed and evaluated, and the **Interface Integration activity** fuse together the *Presentation Model* of some UOs. The *Presentation Model* of a specific UO settles the graphical elements and others

 ISBN: 978-1-61208-165-6

characteristics gathered from the *Requirements Models*. There are several languages for modeling user interfaces widely used and tested, such as XIML[24] or UIML [25], and they may be used to reflect this model. Finally, the **Business–Logic Coding** and **code Integration & Refactoring activities** deal with the *Functionality Model* that guides the implementation in a particular programming language. This model inherits the behaviour characteristics from the UO *Requirements Models* evaluated in the first activity. UML or SysML [26] are alternative languages typically used to represent this model.

B. The SE-HCI model in a Model Driven Process

We propose interactive software development based on user-centered models generated and evaluated during the project, following the Object Management Group's Model Driven Architecture proposal [27].

For each UO the designers involved in the ***Analysis & Navigation Design*** **activity** formalise the established Requirements Models (RMs): *Task Model* and *SE-HCI Model* (see Figure 2.).

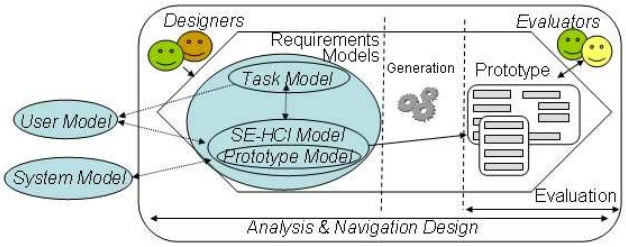

Figure 2. The SE-HCI model involved in a Model-driven process

The *Task Model,* which is a classic element in Model-Based User Interface Development [28][29][30], describes user performance in completing each task. The concept that is the basis of RM is the **Task**, which allows user performance to be captured. This concept is complemented by the **ordering** of tasks (Sequential, Indifferent, Choice, Concurrent), **iteration** which establishes whether it is compulsory to carry out a task and how many times it is necessary to do so (Unitary, Optional, Repetitive) and **hierarchy,** which correctly places the task in the complete set of tasks. That is, the *Task Model* defines the semantic aspects of the application to be associated (see Fig. 3).

The *SE-HCI Model,* which incorporates information from the *User* and S*ystem Models,* is an abstract description constructed over the *Task Model*. The *SE-HCI Model* is the core of our proposed methodology, and it not only gathers the requirements from the *Task Model* but it also incorporates three essential aspects. The first two are behaviour aspects and the third, visual aspect (see Fig. 3):

1) The system direct communication with the user. The description of both the actions that users and the system can carry out at the user interface level (who performs the intervention: usr/sys), during an interactive session [28], and their possible temporal relations are here included. That is, it generates those communications in which the system directly communicates with the user by displaying an error window or a simple message. This means that the system's operations on other elements in the application's environment, such as a database, won't be expressed in the model since they will not be involved in any direct communication with the user.

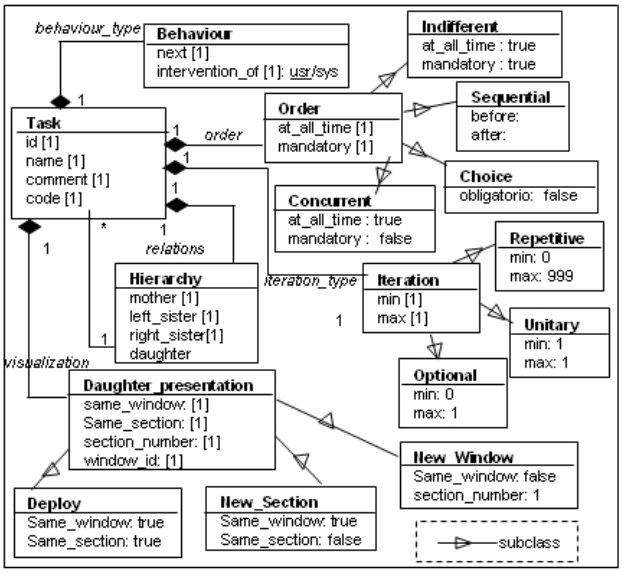

Figure 3. The Requirement Meta-Model

2) The descriptions of the correct interactions, taken from the *Task Model*, as well as the incorrect ones. Both types of interactions express the different application runs (next task to perform). That is, this model represents the semantics of the application through interface navigation.
3) The basic visual characteristics, such as colours, sections, button types, etc. The *SE-HCI* incorporates a *Prototype Model* that gathers these aspects, some of that are assumed from the *User* and *System Models.*

Different techniques can be used to implement this specification. Fig. 4 shows a graphical example of the SE-HCI for the development of a website; it has been made with a HTA technique [31] (some symbols express the characteristics of the tasks). In this case, we use a XML format to express that SE-HCI specification.

In line with user-centered designs, our proposal stresses, like Hix's model [32], the integration of the evaluation process at all stages of the lifecycle rather than just at the end as is the case in the classic cascade lifecycle. The *RMs* make it possible to quickly produce incremental prototypes by adapting the design according to the modifications prompted by both user and software developer evaluations. Similar to our proposal, Propp and his colleagues [33] start with task models in the process of developing interactive applications and they then define the navigational structure, the creation of an Abstract User Interface (AUI) that is independent of the device, and one or more Concrete User Interfaces (CUI).

 ISBN: 978-1-61208-165-6

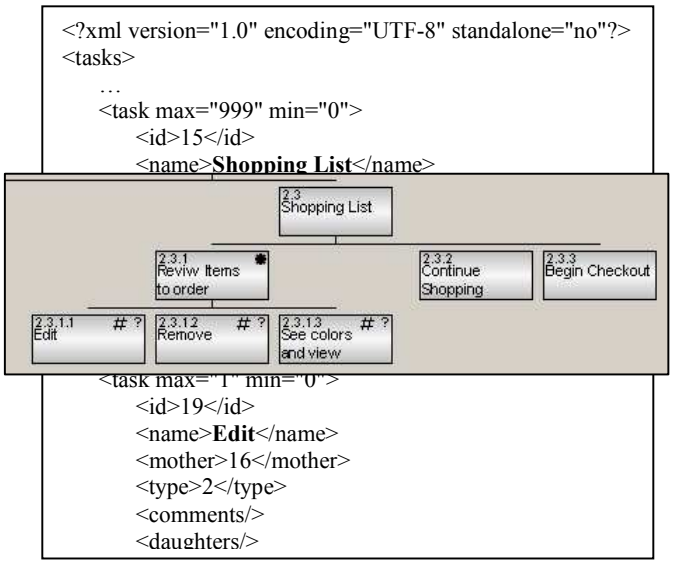

Figure 4. Snapshot of the XML description and Task Hierarchy of an application development

During the development process they perform several usability evaluations. We propose the evaluation of the requirements involved in the SE-HCI with an abstract prototype (Fig. 5) created automatically by transforming the *SE-HCI model*. From this point, the evaluation can be carried out jointly by the designers, customers and developers. According to Wiegers [34] we think that it's hard to visualise exactly how software will behave by reading textual requirements or studying analysis models. Users are more willing to try a prototype than to read a document. Wiegers says: "A prototype is useful for revealing and resolving ambiguity and incompleteness in the requirements."

Figure 5. An Abstract Interface with simple menus and buttons

C. *Iterations in InterMod. A general example*

In this section we explain an iteration progress of a project. In order to facilitate and simplify the general example comprehension, we represent graphically ***Activities*** as shown in TABLE I. As above mentioned, each ***Developmental Activity*** (DA) is driven by models: A1- Requirements Models, A2- Presentation Model and A3- Functionality Model. And each ***Integration Activity*** (IA) is involved in models integration: I1- Requirements Models, I2- Presentation Model and I3- Functionality Model.

TABLE I. INTERMOD ACTIVITIES

Development Activities	Graphical representation		Integration Activities
A1. Analysis & Navigation Design			**I1**. RM Integration
A2. Interface Building			**I2**. Interface Integration
A3. Bus.-Logic Coding			**I3.** Code Integration & Refactoring

Fig. 6 shows a snapshot of the Project Progress State and the Plan obtained for the Parallel Iteration after some iterations (iteration i).

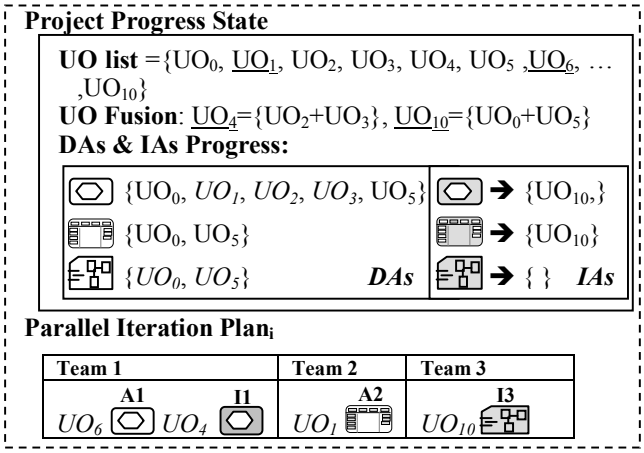

Figure 6. A Snapshot of the Project Progress with InterMod

Three aspects characterise the state of the project: the UO list, the UOs fusion list and the UOs progress according to the ***Activities*** (DAs or IAs) performed. This iteration begins with the **Build UO List** step to revise the UOs list. After that, the process goes on with the **Plan Parallel Iteration** step where the project members have decided:

a) The UOs to perform (underlined in Fig.6 - UO list and UO Fusion),

b) The activities for these UOs, which have been selected taking into account their prerequisites.

c) The distribution into three teams, as follows:

- The first team takes responsibility for two activities: **A1** activity for UO_6 (in the UO list) and **I1** for UO_4. As it is shown, UO_4 is the fusion of the objectives 2 and 3 (UO Fusion in Fig. 6). The I1 Activity is possible because the progress of the project assures that both, the RMs of UO_2 and UO_3 are already validated (see A1 list in "DAs & IAs Progress" in Fig.6).
- The second team builds the interface (A2) for the UO_1 whose prerequisite is reached (UO_1 is in the A1 list).

- Meanwhile, team 3 must integrates and refactors the code referred to UO_{10} that is composed of the objectives 0 and 5 that have been already coded.

The evolution of a UO is not predictable. In each work meeting project members will select the best UOs activities to do. Fig. **7** presents two different possible evolutions of UO_{12} which is composed of {UO_4, UO_6}.

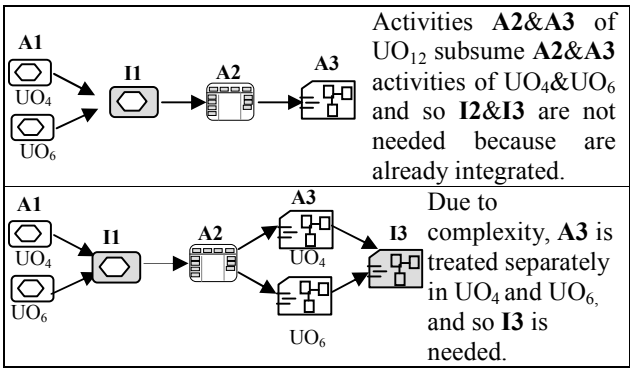

Figure 7. A Snapshot of two UO_{12} possible evolution

When the Plan is ready, the teams go on with the activities assigned. After each iteration is completed, the process repeats:

- **Step2- Build of UO list**- All teams contribute with their work and evaluation results to the list actualization.
- **Step3- Plan Parallel Iteration**- Taking into account the prerequisites of the activities and the project needs, the distribution is carried out.
- **Step4- Perform Iteration Activities-** Each team makes their activities and the process goes again to the Step2 until the application is completed.

IV. CONCLUSION AND FUTURE WORK

In this article we presented a new vision of the InterMod methodology, a proposal integrating three philosophies: UCD, MDD and AM. From the point of view of agile methods, our work is organised in a series of iterations in which the user objectives (UO) to be dealt with are developed. This iterative process speeds up the development and generates results from the project progress. InterMod proposes some developmental and integration activities driven by models to achieve the UOs In the first analysis, the initial user objectives are obtained and then, the different activities to achieve these UOs are distributed among the workgroups. Each iteration is open to include new user objectives, whether obtained through previous refinements or through evolution or alterations during the agile development of the application itself. The possibility to distribute the work in parallel increases the speed of resolution, although the process itself requires integration points to ensure consistency.

The SE-HCI model is the core of our proposal models architecture. It is involved in a Model Driven Process that obtains an abstract prototype created automatically by transforming the *SE-HCI model*. This prototype allows the evaluation of the requirements and facilitates the end user's participation, as recommended by UCD and AM. Early evaluations of the requirements reduce the number of the corrections further on in the process and therefore, reduce its cost.

This process allows for the gathering and validation of the requirements incrementally. Because of this agile approach, InterMod, unlike UCD, does not require the complete development of the application interface before the implementation of the business logic, but assures usability.

The new InterMod methodology has been refined in parallel with the development of a demonstrator. A small initial set of UOs has evolved to a complex system. It has been carried out by means of UO creation, development and integration processes. This make us think of the scalability and practicability properties of the proposed methodology. However these aspects have not been treated in this paper as a deeper work needs to be done.

We are currently working on reusing models. It should be understood in the broadest sense of the word. A UO model can be defined once in a project, but it can be reused at different points in the project. Similarly, a model developed in previous applications can be reused in a current project. Thus, a model can be converted into a pattern or a solution to a design problem. That is to say, we believe that it is important to value the possibility of creating patterns, in order to facilitate and speed up design processes.

ACKNOWLEDGMENT

This work has been partially supported by TIN2009-14380 and DFG 157/2009.

REFERENCES

[1] Mcbreen, P., "Questioning Extreme Programming", Pearson Educ., Boston, MA, USA 2003

[2] Ambler, S., "Debunking Modeling Myths", http://www.ambysoft.com/onlineWritings.html (Last Access: August 2011).

[3] Norman, D.A. and Draper, S.W., "User-Centered System Design: New Perspectives on HCI", Lawrence Erlbaum Associates, Inc, Mahwah, NJ, USA, 1986.

[4] Vredenburg, K., Isensee, S., and Carol Righi, C. , "User-Centered Design: An Integrated Approach", Prentice Hall, 2001.

[5] Norman, D., "Why doing user observations first is wrong". Interactions 13, 4, 2006, pp.50--63

[6] Cooper, A. and Reimann, R.: About Face 2.0, "The Essentials of Interaction Design", JohnWiley & Sons, Inc., Indianapolis, Indiana, USA, 2003

[7] Constantine, L. and Lockwood, L.: Software for Use, "A Practical Guide to the Models and Methods of Usage-Centered Design". ACM Press, Addison-Wesley Co., 1999

[8] Robles, E., Grigera, J., and Rossi, G., " Bridging Test and Model-Driven Approaches in Web Engineering", in: Gaedke M., Grossniklaus M, Díaz O. (eds.) ICWE 2009. LNCS, vol. 5648,. Springer, Heidelberg 2009, pp. 136--150

 ISBN: 978-1-61208-165-6

[9] Ambler, S.W., " The object primer: agile modeling-driven development with UML 2.0." Cambridge University Press, Cambridge 2004

[10] Ferreira, J., "Interaction Design and Agile Development, A Real- World. Pers.", Ph. D. 2007.

[11] Fowler,M. and Highsmith, J., " The agile manifesto, Software Development", 2001, pp 28--32

[12] Highsmith and J., Cockburn, A., "Agile Software Development: The business of innovation". Computer 34, 9, 2001, pp120—127

[13] Beck, K., "Extreme Proamming Explained-Embrace Change". Addison-Wesley, 2000.

[14] Schwaber, K. and Beedle, M. "Agile Software Development with Scrum", Prentice-Hall, 2002.

[15] Cockburn, A., "Agile Software Development", Addison-Wesley, 2002

[16] Palmer, S.M. and Felsing, J.M., "A practical guide to feature-driven development". Prentice-Hall USA, 2002.

[17] Jacobson, I., Booch, G., and Rumbaugh, J., "The Unified Software Development Process", Addison-Wesley, 1999.

[18] Robey, D., Welke, R., and Turk, D., "Traditional, iterative, and component-based development: A social analysis of software development paradigms", Information Technology and Management, Volume 2, Number 1, 2001, pp53-70

[19] Norman, D.A., "The invisible Computer", Cambridge M.A. MIT Press, 1998.

[20] ISO, (International Organization for Standardisation), 9241-11. Ergonomic requirements for office work with visual display terminals. Part 11: Guidance on usability, 1998.

[21] Eric Evans, Domain-Driven Design, " Tackling complexity in the heart of software", Addison Wesley, 2004

[22] Losada, B., Urretavizcaya M., and Fernández-Castro, I., " The InterMod Methodology: An Interface Engineering Process linked with Software Engineering Stages", In Macías,J.A., Granollers,T., Latorre,P.(eds). New Trends on Human-Computer Interaction: Reseach, Development, New Tools and Methods. Springer, 2009

[23] Larman, C., " Agile & Iterative development: A manager's guide". Addison-Wesley, 2004.

[24] eXtensible Interface Markup Language http://www.ximl.org/ (Last Access: August 2011).

[25] Abrams,M. and Helms, J., UIML Specification, 2002 http://www.oasis- open.org /committees /download.php /5937/uiml-core-3.1-draft-01-20040311.pdf (Last Access: August 2011).

[26] Nolan, B., Brown, B., Balmelli, L., Bohn, T., and Wahli, U., "Model Driven Systems Development with Rational Products". ibm.com/redbooks 2007

[27] Object Management Group. Model Driven architecture. Technical report, 2003 http://www.omg.org/mda (Last Access: August 2011).

[28] Paternò, F. "Model-Based Design and Evaluation of Interactive Applications", Springer-Verlag London, 1999

[29] Puerta, A., "A model based interface development environment" , IEEE Soft.Vol.14-4, 1997

[30] Limbourg, Q., Vanderdonckt, V., Michotte, B., and Bouillon, L., "USIXML: A Language Supporting Multi-path Development of User Interfaces" .LNCS , 3425, 2005, pp 200—220,

[31] Annet, J. and Duncan, K.D., "Task Analysis and Training Design", Occupational Psychology, vol. 41, 1967, pp. 211-221

[32] Hix, D., and Hartson, H.R., "Developing User Interfaces: Ensuring Usability Through Product and Process", John Wiley and Sons, New York NY, 1993.

[33] Propp, S., Buchholz, G. and Forbrig, P., "Integration of Usability Evaluation and Model-based Softwae Development", Journal Advances in Engineering Software. Vol. 40 Issue 12. 2009, pp 1223—1230

[34] Wiegers, K. E., "Software Requirements". Microsoft Press, 2003, pp 234--235

 ISBN: 978-1-61208-165-6