

# RUPi – A Unified Process that Integrates Human-Computer Interaction and Software Engineering

Kênia Soares Sousa, Elizabeth Furtado  
Universidade de Fortaleza

Av. Washington Soares, 1321 – Edson Queiroz – Fortaleza, CE, Brasil  
[kenia@unifor.br](mailto:kenia@unifor.br), [elizabeth@unifor.br](mailto:elizabeth@unifor.br)

## Abstract

*This research has the main objective of presenting a study on the areas of Human-Computer Interaction (HCI) and Software Engineering (SE), focusing on the importance of integrating HCI in SE, more precisely in the Software Development Process (SDP). More specifically, it concerns the adaptation of the Rational Unified Process (RUP) towards having HCI aspects integrated into its main workflows, generating the Rational Unified Process for Interactive Systems, called RUPi. The RUP is a well-established SDP that intends to guarantee quality by controlling the project schedule, budget, communication, productivity, and trustworthiness. Meanwhile, the RUPi intends to guarantee usability, accessibility, and acceptability by focusing on the users and on their context of use, modeling users' tasks, considering guidelines during the SDP, and defining mechanisms to design the User Interfaces (UIs) and to test their usability.*

## 1. Introduction

Many SDP that exist nowadays guarantee quality for the developed system, focusing on schedule, budget, communication, and productivity. But on the other hand, few of them guarantee usability, that is, ease of learn and use. This way, they forget to focus on the users, in their tasks, in their contexts of use, and on the application of guidelines, which are recommendations concerning the usability of the system.

Many works as [1], [10], and [19] have shown that the integration of aspects related to HCI in the SDP brings improvements to the developed software, such as, higher user satisfaction derived from a user-centered SDP. Such integration aims to promote usability, accessibility, and acceptability for interactive systems.

In order to achieve these goals, this work presents a new SDP (RUPi<sup>1</sup>) [23] based on the existing one (RUP), but it has HCI concepts inserted into it. The RUP is a SDP that has many qualities that made it be chosen for this research, such as, being well-defined, applied worldwide, and flexible. The research on this SDP has the goal of acquiring knowledge on its workflows, as well as on the activities, artifacts and workers involved in each workflow. A *worker* performs activities and creates, modifies, and controls the artifacts. An *activity* defines the work that the workers perform in order to produce a valuable result to the project. An *artifact* is a tangible product that can be produced, modified, or used by a process. A *workflow* describes a group of activities that produce a result to the project and shows interactions among workers.

The HCI concepts studied in this research are able to contribute to the generation of usable systems when applied in a SDP. These HCI concepts are: human factors, guidelines, user-interfaces (UIs) for all, and the generation of model-based UIs. These concepts were chosen based on established literature that consider these aspects fundamental to be applied during the SDP.

Our strategy to demonstrate the importance of HCI concepts integration is the explanation of artifacts generated with the RUP, followed by the explanation and presentation of artifacts generated with the RUPi. The comparison between these artifacts will make it possible for us to envision the benefits of applying a SDP that includes HCI aspects.

In the RUPi, we created new workflow details, composed of a set of activities that will be included in the existing workflows. We inserted new workers and new artifacts to be produced by them while performing new activities, part of the newly inserted workflow details.

---

<sup>1</sup> The RUPi name was elaborated based on the UMLi[20], which stands for Unified Modeling Language for Interactive Systems, an extension of UML. UMLi addresses the modeling of complete interactive applications, specially their UIs.

The RUP will have its characteristics presented in the second section. Some aspects of HCI were selected and will be presented in the third section. Finally, these aspects will be included in the RUP, generating the RUPi, which will have its specific characteristics presented in the fourth section.

## 2. The Rational Unified Process

The RUP is a software development process concerned with the production of software that meets the needs of its end users within the predefined budget, schedule and with high quality. RUP is able to aggregate the best software development practices in a suitable way for a great number of organizations and projects. The six best practices are so-called because they are commonly used by successful organizations and for being able to mitigate the main causes of software development problems. The best practices applied in a SDP are the following: i) develop software iteratively; ii) manage requirements; iii) use component-based architectures; iv) visually model software; v) continuously verify software quality; and vi) control changes to software. [8]

The RUP architecture is divided in two dimensions. The first one represents the dynamic aspect of the process, and it is expressed in terms of phases: Inception, Elaboration, Construction, and Transaction. The second one represents the static aspect of the process, and it is expressed in terms of workers, artifacts, activities, and workflows.

Concerning its dynamic aspect, each phase executes some activities of related workflows in order to have a product in the end of each iteration. In the *inception phase*, the focus is on understanding general requirements, and on the definition of the project scope. In the *elaboration phase*, the focus is on requirements, but some efforts are devoted to the production of an architecture prototype in order to minimize technical risks by trying to adopt new solutions and learning new tools and techniques. In the *construction phase*, the focus is on the design and implementation, when the initial prototype evolves into the first operational product. In the *transition phase*, the focus is on ensuring that the system has the correct level of quality to reach its goals, when the defects are corrected, users are trained, features are adjusted, and elements are added in order to deliver the final product.

Concerning its static aspect, a process describes who (workers) is doing what (artifacts), how (activities), and when (workflows).

Figure 1 depicts the relationship among the four of the RUP workflows and their specific artifacts and workers. Only these workflows were chosen because they are the most commonly applied phases in any SDP, especially in interactive systems design, such as in [12] and in [15].

| Workflows           | Workers                                                                                            | Artifacts                           |
|---------------------|----------------------------------------------------------------------------------------------------|-------------------------------------|
| Requirements        | System Analyst<br>Use-Case Specifier<br>User-Interface Designer<br>Requirements Reviewer           | Use Case Model                      |
|                     |                                                                                                    | Vision Document                     |
|                     |                                                                                                    | Supplementary Specification         |
|                     |                                                                                                    | Glossary                            |
|                     |                                                                                                    | Software Requirements Specification |
| Analysis and Design | Architect<br>Designer<br>Database Designer<br>Capsule Designer<br>Architecture and Design Reviewer | User-Interface Prototype            |
|                     |                                                                                                    | Use-Case Storyboard                 |
|                     |                                                                                                    | Software Architecture Document      |
|                     |                                                                                                    | Design Model                        |
|                     |                                                                                                    | Interfaces                          |
|                     |                                                                                                    | Analysis Model                      |
|                     |                                                                                                    | Real-Time System Artifacts          |
|                     |                                                                                                    | Use-Case Realization                |
|                     |                                                                                                    | Class Model                         |
|                     |                                                                                                    | Data Model                          |
| capsule             |                                                                                                    |                                     |
| Implementation      | Implementer<br>System Integrator<br>Architect<br>Code Reviewer                                     | Implementation Subsystems           |
|                     |                                                                                                    | Components                          |
|                     |                                                                                                    | Integration Build Plan              |
|                     |                                                                                                    | Implementation Model                |
|                     |                                                                                                    |                                     |
| Test                | Test Designer<br>Tester                                                                            | Test Plan                           |
|                     |                                                                                                    | Test Model                          |
|                     |                                                                                                    | Workload Model                      |
|                     |                                                                                                    | Test Results                        |

Figure 1 - Relationship among workflows, workers and artifacts in the RUP

The requirements workflow of the Rational Unified Process presents an approach to identify and control the system requirements. The analysis and design workflow of the Rational Unified Process shows an approach to translate the requirements into a specification that describes how to implement the system. The implementation workflow of the Rational Unified Process has an approach to implement systems by developing components, and incrementally integrating them, and producing prototypes to reduce risks. The test workflow of the Rational Unified Process presents an approach to evaluate the quality of the product, beginning early in the life cycle throughout the process until the evaluation of the final product delivered to users.

We verified some difficulties can be identified in the use of the RUP for developing interactive applications, such as: i) no consideration of human factors during the SDP; ii) no consideration of international and intercultural aspects during the UI design; iii) no consideration of user-centered models during the UI design and; iv) no consideration of UI customization based on the interaction technology and on the users. In order to attend these aspects, we propose the integration of HCI aspects that address the considerations mentioned above, resulting in the RUPi.

## 3. Human-Computer Interaction

HCI is becoming critical in the emerging information society because of the proliferation of interactive systems as tools for communication, collaboration, and social interaction among groups of people, and because human activities are increasingly

becoming mediated by computers, so computers are viewed as tools for productivity enhancement.

HCI concepts are becoming more important in the SDP because they focus the design process of an interactive system on the user. Besides, according to [11], “Almost half of the software in systems being developed today and thirty-seven to fifty percent (depending on life-cycle phase) of the efforts throughout the life cycle are related to the system's user interface.”

The HCI concepts studied and presented in this work are: human factors, guidelines, UI for all, and model-based UI generation.

### 3.1. Human Factors

Human factors study the way people perform their activities. It is important to include human factors since the beginning of the software development in order to reach human factors goals. These goals are: time to learn how to use the software; speed of performance of tasks using the software; rate of errors by users while performing their tasks; retention of knowledge on the software over time; and subjective satisfaction of users about the software. [17]

Designers have to be sensitive to human capacities and needs related to technology use. This attention is extremely useful in order to avoid users feeling frustration, fear, and failure when facing a complex system with incomprehensible terminology, or chaotic layout. In order to develop an interactive system that provides users the possibility of performing their tasks effectively, designers must consider the diversity of human physical and intellectual abilities, personalities, cultural backgrounds, and age constraints. In addition, they can consider some cognitive models to help identify the task of the system to be developed [4].

### 3.2. Guidelines

Guidelines are a set of advices that guide designers towards developing usable UIs. They are useful to provide solutions to design problems or to serve as a source of information necessary for unfamiliar design situations.

There are some principles that can be applied in any interactive system design in order to achieve users' satisfaction. This satisfaction can be achieved by “providing simplified data-entry procedures, comprehensible displays, and rapid informative feedback that increase feelings of competence, mastery, and control over the system” [17].

Some organizations adapt guidelines to suit the reality of their cultural needs and characteristics or to suit a single project requirements, producing a document composed of a collection of principles and rules that can be used to develop consistent UIs, called style guide.

Furtado [6] describes a collaborative process to define a style guide for an organization. This approach is implemented through a tool, which supports the participatory and evolutionary process of creation of a style guide. This approach for elaborating a style guide is structured with questions and answers and it can be used as a help if a developer encounters a problem when using a development tool or as documentation for new developers.

### 3.3. User Interfaces For All

In the information era, it is important to develop high-quality user interfaces, accessible and usable by a diverse user population with different abilities, skills, requirements, and preferences in a variety of contexts of use, and through a variety of different technologies, that is, user interfaces for all [18]. It is becoming compelling to design for the greatest possible user population.

One of the basic guidelines for HCI design is to study the user. Designers increasingly have to provide information to be used by diverse user groups, including people with different cultural, educational, training, and employment background, novice and experienced computer users, the very young and the elderly, and people with different types of disabilities.

The development of User interfaces for all can be achieved by applying international and intercultural, usability, accessibility, and acceptability methods.

- International and intercultural UIs provide availability of, and easy access to interactive systems among people for countries worldwide. In order to achieve this, user-centered design, globalization and intercultural issues are considered;
- In order to achieve usability, the interactive system should be easy to learn; efficient to use; easy to remember; have a low error rate; and pleasant to use;
- Information is accessible by all kinds of people, if careful attention is provided to their abilities, needs, and preferences during interactive systems development and;
- Acceptability is the possibility that users have to access, understand, and use interactive systems easily.

### 3.4. Model-Based User Interface Generation

The process of generating a UI based on models means that UI's characteristics and its functionalities are generated from specifications represented in different models.

These models can be the following: conceptual interface model, task model, user model, and context of

use model. The conceptual interface model defines the data that a user can view, access, and manipulate through a UI. The visualization can be displayed for the user in different media, e.g., text, sound, graphics, etc. The task model consists of a list of tasks that allows designers to envision what tasks users perform. The user model includes descriptions of users, and the way they perform their tasks. Users perform their tasks in a specific environment, which is represented by the context of use model [4]. Graphical scenarios can help designers understand different contexts of use where multiple types of users may carry out multiple tasks.

## **4. The Integration of Human-Computer Interaction in the Rational Unified Process**

### **4.1. Works with Human-Computer Interaction in Software Engineering**

Although many software engineers have not yet noticed the importance of HCI in the SDP, some HCI experts have been trying to demonstrate that the integration between these two areas can bring improvements, such as, higher user satisfaction derived from a user-centered SDP [7].

Benner [1] presents the integration of scenarios in the software development process in order to better describe properties of the domain, define system requirements, and evaluate design alternatives. The intention to integrate scenarios in the software development process is devoted to better represent situations of users performing tasks while interacting with an interactive system. Having scenarios as artifacts, generated by specialized tools, can be helpful to guide designers during the UI generation and to evaluate usability.

Tavares [19] presents the integration of UI design mechanisms in the software development process by using UML diagrams and scenarios. The intention is to apply modeling techniques during requirements analysis in order to design the UI. The modeling techniques proposed are: scenario generation; task modeling; UML diagrams generation, such as, use case model, activity diagram, interaction diagram, class diagram; and a formalism related to the designer message specification language, called LEMD, which specifies the UI according to the semiotics engineering approach. These modeling techniques are applied in order to generate prototypes to be validated by users.

Long [10] presents the integration of human factors with SE by applying guidelines in a design method based on Denley's work [2]. It is also presented the application of a design method, called a Method for Usability Engineering (MUSE), based on Lim and Long's work [9]. The first phase involves: collecting information

about users and their tasks, and producing a task model of the system to be designed. The second phase involves: eliciting users' needs and defining the application domain, preparing the conceptual design of the system considering users' needs and the application domain as a task model. The third phase includes: preparing the interaction design that represents human behaviors and considers the style guide, the users' needs and the application domain.

None of the works mentioned above considers the main HCI concepts during the entire development process, such as, human factors, guidelines, and user interfaces for all.

### **4.2. RUPi Characteristics**

The six best practices adopted by the RUPi are:

- Develop software iteratively – HCI aspects can be applied since early in the lifecycle until the product is ready for release, what raises the quality of the final product. Prototypes can be generated by applying the results of modeling techniques starting in the requirements workflow.
- Manage requirements - Analysis of the users' tasks, the environment where they are performing their tasks, and the devices that they are using, guaranteeing that the software accommodates users' reality.
- Use component-based architecture - Modular architecture definition, but, more specifically, it suggests the adoption of the n-tier architecture. They provide the software to be divided in independent layers, what provides higher reusability and flexibility.
- Visually model software – The use of user, task, context, and domain models. These models can be directly aided by graphical scenarios.
- Continuously verify software quality - Quality assessment also starts early in the lifecycle and focuses on areas of highest risk. RUPi focuses on usability, accessibility, and acceptability.
- Control changes to software - RUPi proposes maintainability efforts towards the models defined during requirements elicitation.

The RUPi process structure is divided in two dimensions: the dynamic and the static dimension. The phases in the dynamic dimensions are the same ones defined in the RUP. What changes is the application of HCI concepts in every phase and their evaluation in every milestone. The elements in the static dimension, which are workers, activities, artifacts, and workflows, have the same definitions and purposes as the ones defined in the RUP. What changes is the content of each workflow, that is, which activities will be performed and which artifacts

will be produced to accommodate HCI aspects and who will be the workers.

The RUPi has four workflows, which are based on the RUP workflows and were adapted by the inclusion of HCI concepts, described in section 3. The four chosen workflows for RUPi are: requirements, analysis and design, implementation, and test. This addition of HCI concepts is done in order to perform activities that produce artifacts that consider:

- human factors, considered during the development of an interactive system to help users perform their tasks effectively;
- usability requirements, used to decide which UI option is most suitable to the performance of a specific task by a particular user;
- accessibility and acceptability issues, applied during the definition of guidelines in order to accommodate characteristics of all different kinds of users;
- usability evaluation, used to test the usability of an interactive system, considering users' characteristics, tasks performed, environment of work, and technology used and;
- graphical scenarios, generated to graphically represent specific situations experienced by users while interacting with the system.

Figure 2 depicts the relationship among the four of the RUPi workflows and their specific artifacts and workers.

| Workflows           | Workers                 | Artifacts                              |
|---------------------|-------------------------|----------------------------------------|
| Requirements        | Ergonomist              | Scenario                               |
|                     | Human Factors Expert    | User Model                             |
|                     | User-Interface Designer | Task Model<br>User-Interface Prototype |
| Analysis and Design | User-Interface Designer | Task Model<br>MIC<br>Class Model       |
|                     | Implementer             | Prototypes                             |
| Test                | Test Designer           | Test Model                             |

Figure 2 - Relationship among workflows, workers and artifacts in the RUPi

Following, the RUPi workflows are presented.

The Requirements workflow of the RUPi contains the workflow details of the RUP requirements workflow, and adds some more:

- user modeling, specifies the users who will use the system. This specification details information about: general knowledge, level of experience performing their tasks, using a computer, or using a similar system, preferences related to how data is displayed on the screen, physical and psychological abilities, role in the organization, working method, etc. Figure 3 depicts the user model for a project management software, in

which we present four kinds of users with their specific characteristics;

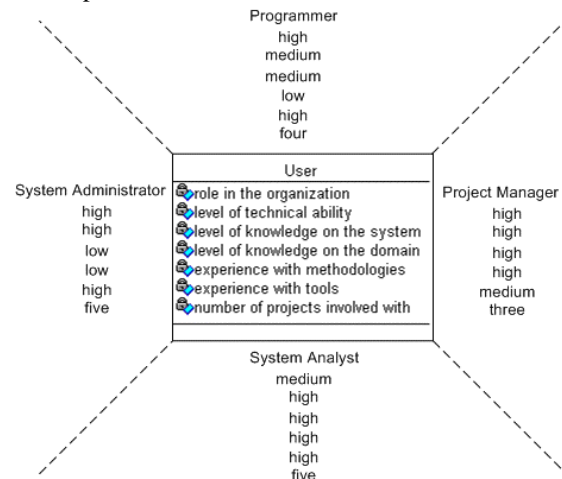


Figure 3 – User model for the project management software

- task modeling, defines the tasks performed by users and the cognitive characteristics necessary for users to perform them. In this work, we used the MAD formalism [16], in which a task is divided in two or more subtasks that are performed in order to fulfill a certain objective. This model is organized in hierarchical levels that link the tasks and organize their performance in: sequential, parallel, simultaneous, alternative, optional, or recurrent;
- context of use modeling, includes definition of the environment where the users are located in order to perform their tasks. This model can be enhanced with the generation of scenarios that are helpful to anticipate risks, prepare prototypes, and verify the accessibility and acceptability issues. Figure 4 depicts a scenario showing the library of a software development organization, where a developer is searching for a book [4];

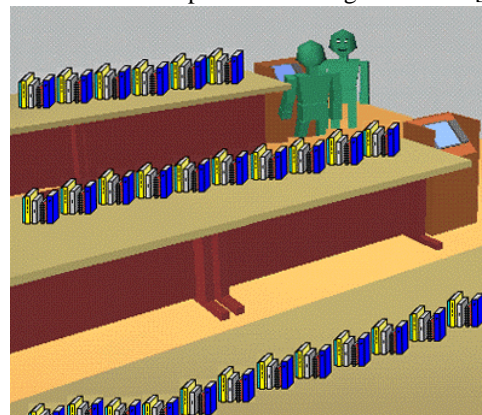


Figure 4 – Graphical scenario for an intranet development project

- domain modeling, specifies the aspects of the work to be performed with or without the system;
- definition of guidelines based on the users, their tasks, their working environment, and the system domain and;
- definition of usability requirements, related to users' satisfaction and the performance of the system. These requirements can be gathered by analyzing the user, the task, and the context of use models.

The Analysis and Design workflow focuses on transforming requirements into a design specification useful for the implementation workflow. This is achieved by defining the software architecture, designing components and the database. This workflow can be incremented with the following workflow details:

- Refinement of the task model performed in the previous workflow by considering characteristics of different kinds of users and different interaction styles;
- UI conceptual design, represents the interactive part of the system by graphically representing the navigational structure of the system. To choose the best option, the designer should associate usability requirements to the design options, as shown in figure 5, and discuss them with the users to come to a decision consensus and;

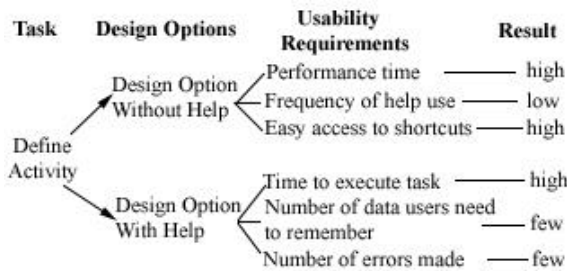


Figure 5 – Usability requirements related to design options

- class model, which is the foundation for the implementation of the project. Considering the UI conceptual design and domain modeling, we can define the class model in a n-tier architecture because the information is already separated. In such a class model, the first layer represents the interface, the second one is the control layer, the third is the business logic layer, and the fourth one is the data layer.

The Implementation workflow focuses on the codification and integration of components. The generation of the physical UI can be enhanced by the application of guidelines. Guidelines principles are interpreted considering the users, the tasks they perform, and the environment where they interact in order to select and apply them correctly. A software tool supports designers and developers during the development process,

among other goals, to achieve usability. There are some tools that intend to help in the application of guidelines during the generation of UIs, such as SEGUIA [22] and SIERRA [21]. These tools are useful because of the absence of consistency among guidelines and the lack of a uniform structure among guideline documents. These shortcomings make it hard to organize guidelines and inter-relate guideline documents.

As a consequence of correctly applying guidelines during the implementation workflow, the physical UI is generated in a way that accommodates users' expectations towards usability, acceptability, and accessibility, thus, their satisfaction.

Figure 6 shows an adaptation in the UI towards the steps taken by users to achieve their goal while performing a set of tasks. The options in the menu offer a list of workers' productivity and a list of similar activities related to prior projects in order to help the user define the schedule for a specific activity. This adaptation is helpful for novice users, since it does not expect users to have enough experience in project management, nor to directly define the schedule for an activity.

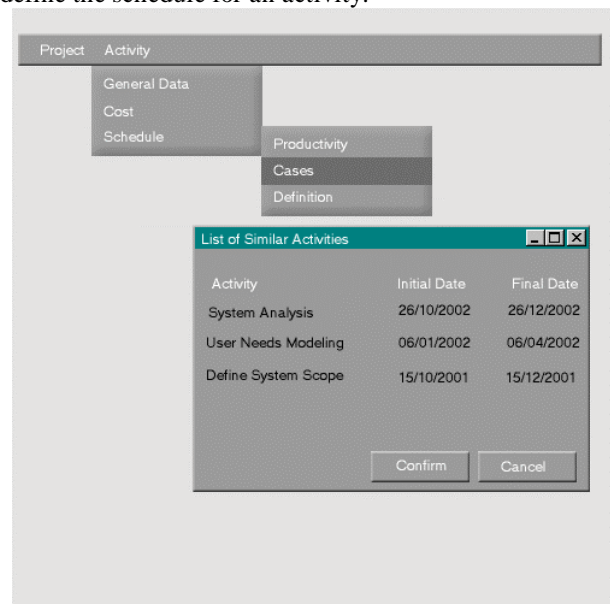


Figure 6 – Prototype with activity list as help

Although some UI adaptation aspects demonstrate the differences in applying one of the two SDP, there are some aspects, related to the advantages of applying HCI in a SDP, which are not visually noticed. Such advantages are related to user satisfaction, user participation during the SDP that leads to more precision in the accommodation of users' requirements, risks anticipation derived from task modeling, among other benefits related to the cognitive level.

The Test workflow focuses on the quality of the product since early in the SDP. Its main activities are verifying the integration of components, correcting the implementation of requirements, and correcting detected

defects. The main workflow detail to be included in this workflow is primarily concerned with usability aspects of the system developed, which is usability evaluation. Usability evaluation [14] should be a concern during this workflow since it involves analysis of a system in the users' point of view. That is, this evaluation takes into consideration: users' characteristics, the tasks they perform, the environment in which they work, and the technology they use. Despite of the evaluation method applied, these aspects are important to predict the usability of a system. Evaluation tests can be composed of test cases that describe what aspects to be tested in the system. In the RUPi, these aspects are closely related to usability requirements defined in the requirements workflow. An example of a RUPi test case contains a list of usability attributes defined according to the usability requirements, the specification of the measured value, and the expected value for each tested attribute, as depicted in figure 7. The test case prepared using the RUP concerns functionality attributes.

| Usability Attribute      | Measured Value        | Expected Result |
|--------------------------|-----------------------|-----------------|
| Performance time         | Time to complete task | 40 sec          |
| Frequency of help use    | Number of accesses    | 4 of 10         |
| Easy access to shortcuts | Time to complete task | 28 sec          |
| Memorability             | Time to complete task | 40 sec          |
| Number of errors made    | Percentage of errors  | 10,00%          |

Figure 7 – Test case related to usability attributes

#### 4.4. RUPi Workflow Analysis

Analyzing the application of HCI aspects in these four workflows, we can notice the difference in the artifacts created by RUP and by RUPi. In the latter, there are more artifacts that consider the domain of work, the users, the tasks they perform, the devices they use, and the environment where they work. In addition to that, guidelines and usability requirements are considered in order to accommodate any specificity derived from the HCI aspects analyzed by the development team.

The main intention of this section is to differentiate the artifacts produced by RUPi from the ones from RUP, and especially to show the advantages achieved by the SDP from the RUPi artifacts.

In the RUP requirements workflow, the main artifact is the use case model. In the RUPi, the main artifacts are the user model and the task model oriented by the use case model. These models contribute to the generation of a UI that is suitable to the users' reality, opposed to being too generic to an extent that does not support users with their tasks.

In the RUP analysis and design workflow, the primary artifact is the design model that contains the class model. In the RUPi, the class model is designed according to the n-tier architecture, which allows this model to be

reused by other systems in a more flexible manner. Besides that, there is an analysis of UI conceptual design options based on usability requirements. This analysis makes the decision process more concrete because it is based on HCI aspects and not on personal opinions.

In the implementation workflow, the most important artifact is the system. The UI designed following workflow details from RUPi present adaptation in the semantic level, which is related to adapting the UI towards the steps taken by users to achieve their goal while performing a set of tasks; and adaptation in the syntactic level, which is related to adapting the UI towards visualization aspects, such as, menu option, font size, font colors, background colors, etc. The UI designed following workflow details from RUP do not present any level of adaptation. It is noticeable that the RUPi's UI is more usable than the other one because of its possibility of customization.

In the test workflow, the key artifact is the test model, which contains test cases. The RUP's test cases verify and validate the system functionality, while RUPi's test cases focus on usability.

The application of RUPi will prove to achieve the goals mentioned above along with usability goals. These usability goals have proven to bring benefits, such as ease to learn and use the system, reducing training time and the number of errors, consequently, reducing costs and time spent in the correction of defects.

#### 5. Conclusion

This research demonstrated the need for the integration of HCI concepts into a SDP because of the importance of concentrating on the users, their tasks, their environment, and the technologies they use.

The main contribution of this work is the definition of RUPi and its first workflows by taking into account HCI concepts.

Each RUPi workflow suggests the generation of some artifacts that intend to aid the development team in developing an interactive system that considers human factors, usability requirements, accessibility and acceptability issues, usability evaluation, graphical scenarios, among other aspects.

The future of this work is to define the other five RUPi workflows. In addition to that, we intend to develop a module for a project management software. This module will guide the development team in generating a UI by applying those HCI concepts integrated into the workflows. The automation of this integration will help in making it more easily applicable in many organizations.

## 6. References

- [1] BENNER, Kevin M. et al. **Utilizing Scenarios in the Software Development Process**. California: Elsevier Science, 1993.
- [2] DENLEY apud LONG, John. **Integrating Human Factors with Software Engineering for Human-Computer Interaction**. In: Seventh journal on Engineering of Human-Computer Interaction (IHM 1995). France: Cépaduès-Éditions, 1995, p.5.
- [3] DIX, Alan et al. **Human-Computer Interaction**. England: Prentice Hall, 1993.
- [4] FURTADO, Elizabeth. **Um Analisador Ergonômico de Interfaces Homem Máquina**. In: II Semana Universitária da UECE. Anais do VI Encontro de Iniciação Científica. Ceará, 1997.
- [5] FURTADO, Elizabeth; Sousa, Kênia. **An Ontology-Based Method for Universal Design of User Conceptual Interfaces Using Scenarios**. In: Task Models and Diagrams For User Interface Design (TAMODIA 2002). Bucharest:INFOREC, 2002.
- [6] FURTADO, Elizabeth; Sousa, Kênia; CÓLERA, César. **An Environment to Support Developers in Elaborating a Participatory and Evolutionary Help on Style Guide**. In: Seminário em Fatores Humanos para Sistemas de Computação, 2002, Fortaleza. Usabilidade: um direito. Fortaleza: Banco do Nordeste, 2002. p. 84 – 92.
- [7] HEFLEY, William et al. **Integrating Human Factors with Software Engineering Practices**. In: Human-Computer Interaction Institute Technical Reports, Pennsylvania, 1994. Available in: <http://reports-archive.adm.cs.cmu.edu/hcii1994.htm>. Accessed in 15 nov. 2002.
- [8] KRUCHTEN, Philippe. **The Rational Unified Process -An Introduction**. 2 ed. New Jersey: Addison-Wesley, 2000.
- [9] LIM; LONG apud LONG, John. **Integrating Human Factors with Software Engineering for Human-Computer Interaction**. In: Seventh journal on Engineering of Human-Computer Interaction (IHM 1995). France: Cépaduès-Éditions, 1995, p.7.
- [10] LONG, John. **Integrating Human Factors with Software Engineering for Human-Computer Interaction**. In: Seventh journal on Engineering of Human-Computer Interaction (IHM 1995). France: Cépaduès-Éditions, 1995.
- [11] MYERS; ROSSON apud HEFLEY, William et al. **Integrating Human Factors with Software Engineering Practices**. In: Human-Computer Interaction Institute Technical Reports, Pennsylvania, 1994, p.4.
- [12] NEWMAN, William M.; Lamming, Michael. **Interactive System Design**. England: Addison-Wesley, 1995.
- [13] NIELSEN, Jakob. **Usability Engineering**. California: Morgan Kaufmann, 1993.
- [14] PREECE, Jenny et al. **Human-Computer Interaction**. England: Addison-Wesley, 1994.
- [15] REDMOND-PYLE, David; Moore, Alan. **Graphical User Interface Design and Evaluation -A Practical Process**. England: Prentice Hall, 1995.
- [16] SEBILLOTTE, Suzanne. **Hierarchical Planning as a Method for Task Analysis: The Example of Office Task Analysis**. In: Behavior and Information Technology. v. 7, n. 3. 1988, p. 275-293.
- [17] SHNEIDERMAN, Ben. **Designing the User Interface: Strategies for Effective Human-Computer Interaction**. 3 ed. England: Addison-Wesley, 1998.
- [18] STEPHANIDIS, Constantine. **User Interfaces for All: New Perspectives into Human-Computer Interaction**. In: STEPHANIDIS, Constantine. (ed) **Universal Access in HCI - Towards an Information Society for All**. New Jersey: Lawrence Erlbaum Associated, 2001.
- [19] TAVARES, Tatiana A.; Leite, Jair C. **ARFDIU -Um método para Integrar Análise de Requisitos Funcionais com o Design de Interfaces de usuário Usando UML e outros formalismos**. In: V Symposium on Human Factors in Computer Systems (IHC 2002). Fortaleza: SBC, 2002, p. 313-323.
- [20] UMLi. **The Unified Modeling Language for Interactive Applications**. Available in: [http://www.cs.man.ac.uk/img/uml\\_i](http://www.cs.man.ac.uk/img/uml_i). Accessed in 30 jan. 2003.
- [21] VANDERDONCKT, Jean. **Accessing guidelines information with Sierra**. In: Proceedings of Fifth International Conference on Human-Computer Interaction (HCI International 1995). London: Chapman & Hall, 1995, p. 311-316.
- [22] VANDERDONCKT, Jean. **Assisting Designers in Developing Interactive Business Oriented Applications**. In: Proceedings of the Eighth International Conference on Human-Computer Interaction (HCI International 1999). Mahwah: Lawrence Erlbaum Ass., 1999, p. 1043-1047.
- [23] SOUSA, Kênia Soares; FURTADO, Elizabeth. **Integration of Human-Computer Interaction in a Software Development Process**. In: HCI INTERNATIONAL 2003, 2003, Creta. 2003.