Building the User Interface: The Case for Continuous Development in an Iterative Project Environment

by Jim Conallen
Web Modeling Evangelist
Rational Software

The Rational Unified Process® (RUP®) is a software engineering process. It details an iterative approach to the software development lifecycle. Each iteration performs a set of loosely sequential activities that span multiple disciplines in the lifecycle. In the early iterations, most of the activities focus on management, requirements, and design. Later activities focus on implementation, testing, and deployment. One of the main goals of iterative development is to identify potential problem spots and fix or adjust for them early on, when it is cheaper. The development of the user interface (UI), or "user experience," is therefore an activity that is worked on throughout the entire process.

This article offers a counterpoint to Wayne Lee Jones' fine article on adopting the RUP in user interface development at Oracle Corporation. Wayne's position, in a nutshell, appears to be that user interface development should be considered a very separate activity and separate product of the overall development lifecycle. He argues that the side effects of introducing user interface development at the wrong time could adversely affect the development of the "right" solution. While recognizing that user interfaces are not monolithic and require iterative development, Wayne suggests a process that staggers user interface iterations one step behind the rest of the system. This is to ensure that the user interface is wholly dependent on the rest of the system and not the other way around.

While many of his points are valid and should be considered for many
projects, there are significant benefits attached to integrating user interface development with the entire development process, from early Inception to Transition (i.e., from project beginning to end). This article discusses some of the benefits of user interface development early in the development process, and the multiple roles these activities play in the overall development process.

Prototyping: A Brief Overview

Before launching into a discussion of the user interface and its relation to the RUP, let's take a quick look at the various ways software developers employ prototypes in the development process. The importance of the UI as part of the iterative process will then become clearer.

Prototyping is an activity in the development process that reduces risk. A prototype is a concrete example built by the team to answer a question or address an area of risk identified by the development team. Every prototype is created to test at least one hypothesis. The hypothesis might be architectural (will our Web services have the throughput necessary to meet the performance objectives?), or it might be related to usability (will the entry clerks understand the menu structure in the user interface?).

Prototypes fall into two major classes: 1) Exploratory Prototypes, designed to understand and answer questions about the some part of the system; and 2) Evolutionary Prototypes, designed to change along with the rest of the project and be incorporated into the system. The lifecycle of a prototype is wholly determined by its classification.

1. Exploratory Prototypes are usually short-lived prototypes created to quickly answer a question or address a specific risk point. They are either Structural or Behavioral. Structural prototypes are used to test architectural mechanisms and other structural concerns. Behavioral prototypes examine system behavior and include the usability of the user interface and external system integrations. Once these prototypes have validated or discredited their hypothesis, the prototype is thrown away.

2. Evolutionary Prototypes, on the other hand, are intended to be integrated with the rest of the evolving system. Like their exploratory cousins, they are still created to address a hypothesis; however, evolutionary prototypes are designed more carefully and in accordance with the rest of the system architecture, so that when completed, they can be incorporated into the larger system.

The Unique Role of the UI in the Development Process

The development of the user interface stands out as one of the more interesting activities of the development process, because its very goal actually changes during the development lifecycle. Early on, the development of the user interface is used (in part) as a tool or technique for gathering requirements. It is not until later on in the process that the requirements are analyzed and feedback from the prototypes is
incorporated into a suitable user interface design.

In the earliest iterations, user interface developers use prototypes and storyboards as means to elicit requirements from the stakeholders. The RUP recommends that simple user interface prototypes be created early in the development lifecycle. In addition, it states that,

... the main purpose of creating a user-interface prototype is to be able to expose and test both the functionality and the usability of the system before the real design and development starts. This way, you can ensure that you are building the right system, before you spend too much time and resources on development.

The term "test" in this context does not mean that the actual functionality of all layers of the application are executed as it would be in the final stages of acceptance testing. User interfaces are "tested" against the requirements for functionality, and in general for usability. Metrics such as number of clicks, or screens required to complete a scenario, are gathered and evaluated. Stakeholders validate screen definitions and identify missing or redundant information. The earliest prototypes are simple paper drawings or bitmaps that are assembled into storyboards and reviewed by the stakeholders.

Let me illustrate the importance of user interface testing as a means of eliciting and validating requirements. I recall one of my early experiences as a user interface developer. We were building a call center application and had been working on the customer management module. After a couple of weeks working in a small team of three (stakeholder, logical data modeler, and user interface designer), I was quite proud of the user interface mock-up I created. I showed it to the rest of the members on our team, and they were both impressed. But after a few minutes the stakeholder asked how she could enter in the second customer address. The logical data modeler and I looked at each other in confusion. During all the weeks we had worked with the stakeholder to understand and model the requirements of the customer management module, no one had ever expressed this need to associate multiple addresses to customer records. Not until the stakeholder looked at and examined the user interface did this very significant requirement surface. Since we were in the early iterations of the project, this oversight was barely noticed and easily corrected. Had it been discovered much later during beta testing, its impact would have been more significant.

The Importance of Modeling

When the requirements solidify and an appropriate architecture gets defined, a subtle change in focus takes place. The focus of the user-experience team changes from discovering requirements to defining and building the "right" user interface. As Constantine and Lockwood state: "Of the heart and substance of the system itself, users see nothing. To the users, the user interface is the system." Certainly, the look and feel of a user interface can determine the success or failure of an application. But how has this area of development become such a critical aspect of system
design? It is important to remember that, when developing systems with user (human) interfaces, the user interface is all that the user "sees" of the system. For the project manager, any disconnect between the designer's view of the system and the actual user's is likely to involve some aspect of the UI. Which is why models are just as important for the UI as they are for most activities of the software development process. Users have certain mental models of the systems they interact with. These models are based on what they expect the system to do, and are usually founded on their professional experiences. Legacy systems have a strong influence on a user's mental model of the knowledge they are working with, so to understand the user's perspective, it is important for the user interface team to understand relevant legacy or companion systems.

The development team, too, uses models. Throughout the development process, many models (use case, analysis, design, test, deployment, security, process, etc.) are developed and refined. These models represent the system that the team wants to build, is building, and has built. The trick to building a system that is "intuitive" to the user and stakeholders is ensuring that the user's mental models of what the system should be doing actually match the models that describe the system under development.

The user interface activities of the development process are all oriented toward making these models compatible and consistent with the user. UI activities iteratively refine a shared understanding of the "real" requirements, then progressively help the evolution of a suitable interface that matches the user's mental model of the system with the entities and processes that are embodied in the actual deployed system.

**A Summary of RUP User Interface Roles/Activities**

Currently, Rational Developer Network offers several RUP Plug-Ins that address user interface development either directly or indirectly. Any detailed discussion of user interface development specified in these is, of course, beyond the scope of this article; however, it is possible to outline and summarize some of their common points.

For most development teams, we can identify three distinct roles that are directly involved with user interface development. These roles can be performed for small efforts by the same individual, and for larger efforts by separate teams.

- **Information Architect** -- responsible for the structuring the information presented to the user. The information architect develops the navigation map (set of screens or forms that a user navigates through).
- **User Interface Designer/Developer** -- responsible for the development of the prototypes and the actual user interface (screens and forms) that interact with the rest of the system.
- **Graphic Designer** -- responsible for the artwork and color schemes used in the user interfaces.
Other roles and responsibilities may be appropriate for specific development environments; however, these three cover the essence of the user interface team's roles. The RUP defines four phases in the software development lifecycle in which user interface activities occur; each phase includes activities performed by user interface team members, as we shall see below.

**Inception**

The overriding goal of the Inception phase is to achieve concurrence among all stakeholders on the lifecycle objectives for the project. This involves the establishment of the project's scope and vision. The information architect and user interface designer/developer are busy understanding and eliciting the requirements for the system. Low fidelity prototypes of the user interface are created early on, not only to elicit and validate requirements from the stakeholders, but also to establish a baseline for the user interface.

Style guidelines are created, and the overall user experience (look and feel) is examined by the stakeholders. The information architect is concerned with understanding the structure of the information exchanged between the system and its users. Each user role (Actor) could potentially have a different user experience.

**Elaboration**

The goal of the Elaboration phase is to baseline the architecture of the system to provide a stable basis for the bulk of the design and implementation effort in the Construction phase. During Elaboration the navigational map -- the set of screens or user interface forms that the user experiences while interacting with the system -- is defined. This map names and describes each unique screen presented to the users. By the end of this phase, most of the screen's details (fields and actions) have been defined.

For some systems, the prototypes may have evolved into mockups using some of the target architecture's elements. For example, by the Elaboration phase, the paper or bitmap screens used during Inception may have evolved into HTML pages or Visual Basic forms. These prototypes still do not need to connect to any "real" business logic underneath; they are used to understand and obtain feedback from the stakeholders. If any show signs of potential risk (i.e., suggest that implementation will be especially complicated or require unproven technology), they are targeted for early development during Construction.

**Construction**

The goal of the Construction phase is to clarify the remaining requirements and complete the development of the system based upon the baselined architecture and user experience model. The user experience model contains the navigational maps and screen definitions that have been completed for all the architecturally significant parts of the system. Any additional screen definitions and flows are completed in this phase. By the
end of Construction, a functioning application should be ready for beta testing.

**Transition**

The focus of the Transition phase is to ensure that software is available for its end users. The Transition phase can span several iterations; it includes testing the product in preparation for release and making minor adjustments based on user feedback. These adjustments are especially important to the user interface team in Web application development, since this may be the first time real network issues begin affecting the performance of the application. By the time this phase is reached, new Web browsers or versions may have entered the market that were not available during Inception or Elaboration. These will require special attention by the user interface and architecture team.

**Cautions and Concerns**

Doing early prototyping and eliciting requirements with the user interface is not without its drawbacks. Some managers expect functional prototypes to be deployed as an early release of the system. For evolutionary prototypes this might be feasible (although not recommended). A more serious and subtle problem with early prototyping is the danger of unduly influencing the stakeholders with presentation details that may hinder, rather than facilitate, discovery of the system's requirements. For example, if the user interface team prepares a demo or shows a prototype of a similar application to the stakeholders before they've had a chance to interact with them, this can create false impressions and lead to misunderstandings and faulty assumptions about the project at the outset. Deciding when to begin prototyping and to what degree requires careful consideration; this is where an experienced information architect or user interface expert can be important.

These issues can be mitigated by the slow introduction of low fidelity prototypes (hand drawn) at first. Leaving the deeper issues of the look and feel and the user experience metaphors to the information architect and not presenting them to the stakeholders until the end of the Inception phase or the beginning of the Elaboration phase will help mitigate any unproductive influence. By the end of Elaboration, however, the candidate "user experience," together with the architecture, should be well established and reviewed by all stakeholders.

**Summary**

For most applications, the user interface is a very important and critical component. Its development occurs throughout the project lifecycle and requires some unique skills that until recently have not been a staple in the typical software development team. The user interface also has the unusual property of serving multiple purposes in the development process. Early on, it is used as a tool for eliciting and validating system requirements with the stakeholders. As the requirements are better understood, the UI is elaborated and refined to provide the most efficient interaction between a system and its users. Treating the user interface as
a single, autonomous component that is wholly dependent on the invisible business logic of the system does a disservice to both the development of the business logic and the user interface.

Notes

1 See "The Role of Functional Prototyping in the Development Process" on Rational Developer Network (registration required):

2 "Information Architecture" (IA) is a relatively new term for the area of specialty devoted to understanding the structure and presentation of information in user interfaces. This area of specialty has gained prominence with the emergence of Web applications. A good resource site for information architecture is http://www.infodn.com/index.shtml


4 The User Experience Plug-In and Creative Design Plug-In (coming soon) are available on Rational Developer Network: http://www.rational.net (registration required).

For more information on the products or services discussed in this article, please click here and follow the instructions provided. Thank you!