RAPID PROTOTYPING FOR SOFTWARE PROJECTS WITH USER INTERFACES

Ali Tizkar SADABADI[1], Naser M. TABATABAEI[2]
[1]State Engineering University of Armenia (SEUA), Department of Computer Systems and Informatics [2] Seraj Higher Education Institute, Iran
al_tz2@yahoo.com, n.m.tabatabaei@gmail.com

Key words: rapid prototyping, software, development process,

Abstract: *Rapid prototyping is a process for creating a realistic model of a product's user interface. A rapid prototyped user interface is easy to change and gets customers involved early in the design of the product. To prototype successfully, you should pick a rapid prototyping tool that meets your needs, form a small prototyping team, get lots of customer feedback, and iterate until customers are delighted with your user interface. A prototype typically implements only a small subset of the features of the eventual program, and the implementation may be completely different from that of the eventual product. Prototyping has several benefits: The software designer and implementer can obtain feedback from the users early in the project. The client and the contractor can compare if the software made matches the software specification, according to which the software program is built. It also allows the software engineer some insight into the accuracy of initial project estimates and whether the deadlines and milestones proposed can be successfully met. The degree of completeness and the techniques used in the prototyping have been in development and debate since its proposal in the early 1970's.*

1. INTRODUCTION

The process of prototyping involves the following steps:

1.Identify basic requirements: Determine basic requirements including the input and output information desired. Details, such as security, can typically be ignored.

2.Develop Initial Prototype: The initial prototype is developed that includes only user interfaces.

3.Review: The customers, including end-users, examine the prototype and provide feedback on additions or changes.

4.Revise and Enhancing the Prototype: Using the feedback both the specifications and the prototype can be improved. Negotiation about what is within the scope of the contract/product may be necessary. If changes are introduced then a repeat of steps #3 ands #4 may be needed.

2. ELEMENTS OF SUCCESSFUL RAPID PROTOTYPING

Successful rapid prototyping is performed:

•*Quickly* – The first pass must be done quickly, and subsequent improvements should be incorporated immediately. While the prototype needs to give customers a realistic feel for the product, it does not need to include special graphics or computational algorithms that require a lot of time and effort to create.

•*Iteratively* – The prototyped user interface is reviewed, commented upon, improved, and reviewed again in a repeating cycle. No one creates a perfect design the first time. This iterative cycle allows you to gradually improve the user interface. These cycles can be completed more quickly if the prototype is easily changed.

•*Using domain experts* – Ideally, the prototype should be built by a domain expert.

Domain experts are familiar with the user – his or her job, expectations, requirements, jargon, and priorities. These people may have done the user's job in the past. Domain experts can do the best job of incorporating user requirements into the prototype. If your prototyping tool is too difficult for the domain expert to use, make sure that the domain expert works closely with the programmer.

3. TRADITIONAL DEVELOPMENT PROCESS VERSUS RAPID PROTOTYPING PROCESS

The traditional process used to develop a product follows the general steps shown in Figure 1. During Step 1, "Analyze Proposed System," marketing and planning identify a customer need and determine whether the company can develop a product that will profitably meet that need. In Step 2, "Specify Requirements," marketing and planning draft general requirements for the proposed product. In Step 3, "Design System," development writes detailed specifications for the proposed product. In Step 4, "Develop System," development creates the product. In Step 5, "Release Product," the company releases the product.

There are two obvious differences between the traditional product development process and the rapid prototyping process shown in Figure 1. These differences are customer involvement and iterative design. Customers are involved only indirectly at the beginning of the traditional process, when marketing and planning specify requirements. In rapid prototyping, customers are involved directly throughout the development process. Also, the traditional process goes from requirements, to design, to development in a fixed series of steps. In rapid prototyping, the process is iterative. This makes it easier to change or add requirements that will make the product more popular with customers.

There are two obvious differences between the traditional product development process and the rapid prototyping process shown in Figure 1. These differences are customer involvement and iterative design. Customers are involved only indirectly at the beginning of the traditional process, when marketing and planning specify requirements. In rapid prototyping, customers are involved directly throughout the development process. Also, the traditional process goes from requirements, to design, to development in a fixed series of steps. In rapid prototyping, the process is iterative. This makes it easier to change or add requirements that will make the product more popular with customers.

3.1. Types of prototyping: Throwaway prototyping

Throwaway or Rapid Prototyping refers to the creation of a model that will eventually be discarded rather than becoming part of the finally delivered software. After preliminary requirements gathering is accomplished, a simple working model of the system is constructed to visually show the users what their requirements may look like when they are implemented into a finished system.

Rapid Prototyping involved creating a working model of various parts of the system at a very early stage, after a relatively short investigation. The method used in building it is usually quite informal, the most important factor being the speed with which the model is provided. The model then becomes the starting point from which users can re-examine their expectations and clarify their requirements. When this has been achieved, the prototype model is 'thrown away', and the system is formally developed based on the identified requirements.

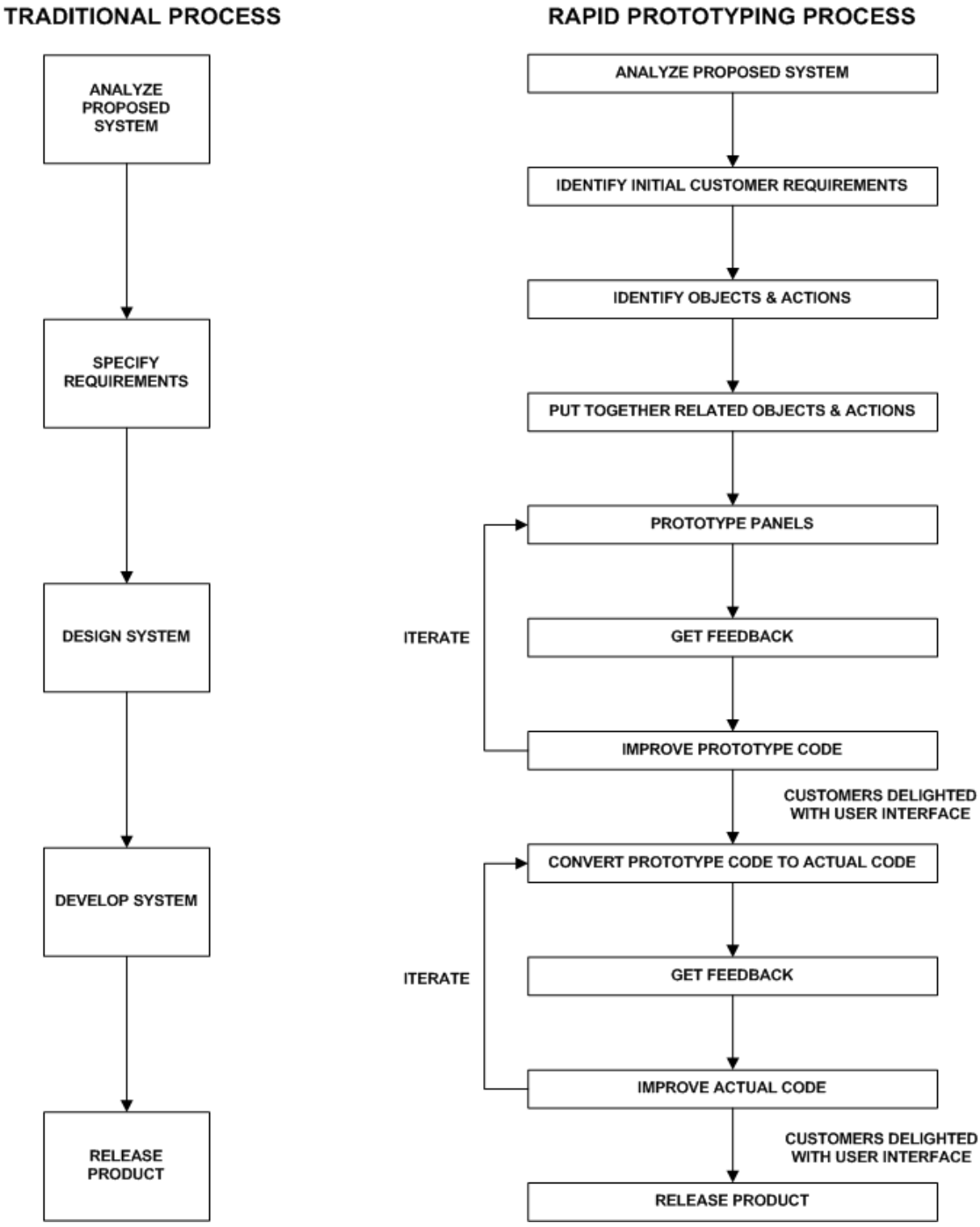

Figure 1. Traditional and Rapid Prototyping Product Development Processes

The most obvious reason for using Throwaway Prototyping is that it can be done quickly. If the users can get quick feedback on their requirements, they may be able to refine them early in the development of the software. Making changes early in the development lifecycle is extremely cost effective since there is nothing at that point to redo. If a project is changed after a considerable work has been done then small changes could require large efforts to

implement since software systems have many dependencies. Speed is crucial in implementing a throwaway prototype, since with a limited budget of time and money little can be expended on a prototype that will be discarded.

Another strength of throwaway prototyping is its ability to construct interfaces that the users can test. The user interface is what the user sees as the system, and by seeing it in front of them, it is much easier to grasp how the system will work.

It is asserted that revolutionary rapid prototyping is a more effective manner in which to deal with user requirements-related issues, and therefore a greater enhancement to software productivity overall. Requirements can be identified, simulated, and tested far more quickly and cheaply when issues of evolvability, maintainability, and software structure are ignored. This, in turn, leads to the accurate specification of requirements and the subsequent construction of a valid and usable system from the user's perspective via conventional software development models.

Prototypes can be classified according to the fidelity with which they resemble the actual product in terms of appearance, interaction and timing. One method of creating a low fidelity Throwaway Prototype is Paper Prototyping. The prototype is implemented using paper and pencil, and thus mimics the function of the actual product, but does not look at all like it. Another method to easily build high fidelity Throwaway Prototypes is to use a GUI Builder and create a click dummy, a prototype that looks like the goal system, but does not provide any functionality.

Not exactly the same as Throwaway Prototyping, but certainly in the same family, is the usage of storyboards, animatics or drawings. These are non-functional implementations but show how the system will look.

In summary, this approach to protyping is constructed with idea that it would be discarded and financial system would be built from the scratch.The steps in this approach are:

1. Write prelim requirements
2. Design the prototype
3. User experiences/uses the prototype, specifies new requirements.
4. Writing final requirements
5. Developing the real product

3.2.Types of prototyping: Evolutionary prototyping

Evolutionary Prototyping (also known as breadboard prototyping) is quite different from Throwaway Prototyping. The main goal when using Evolutionary Prototyping is to build a very robust prototype in a structured manner and constantly refine it. "The reason for this is that the Evolutionary prototype, when built, forms the heart of the new system, and the improvements and further requirements will be built.

When developing a system using Evolutionary Prototyping, the system is continually refined and rebuilt.

"…evolutionary prototyping acknowledges that we do not understand all the requirements and builds only those that are well understood."

This technique allows the development team to add features, or make changes that couldn't be conceived during the requirements and design phase.

For a system to be useful, it must evolve through use in its intended operational environment. A product is never "done;" it is always maturing as the usage environment changes…we often try to define a system using our most familiar frame of reference---where we are now. We make assumptions about the way business will be conducted and the technology base on which the business will be implemented. A plan is enacted to develop the capability, and, sooner or later, something resembling the envisioned system is delivered.

Evolutionary Prototyping have an advantage over Throwaway Prototyping in that they are functional systems. Although they may not have all the features the users have planned, they may be used on an interim basis until the final system is delivered.

"It is not unusual within a prototyping environment for the user to put an initial prototype to practical use while waiting for a more developed version…The user may decide that a 'flawed' system is better than no system at all."

In Evolutionary Prototyping, developers can focus themselves to develop parts of the system that they understand instead of working on developing a whole system.

To minimize risk, the developer does not implement poorly understood features. The partial system is sent to customer sites. As users work with the system, they detect opportunities for new features and give requests for these features to developers. Developers then take these enhancement requests along with their own and use sound configuration-management practices to change the software-requirements specification, update the design, recode and retest

3.3. Types of prototyping: Incremental prototyping

The final product is built as separate prototypes. At the end the separate prototypes are being merged in an overall design.

4. CONCLUSIONS

It has been argued that prototyping, in some form or another, should be used all the time. However, prototyping is most beneficial in systems that will have many interactions with the users.

It has been found that prototyping is very effective in the analysis and design of on-line systems, especially for transaction processing, where the use of screen dialogs is much more in evidence. The greater the interaction between the computer and the user, the greater the benefit is that can be obtained from building a quick system and letting the user play with it.

Systems with little user interaction, such as batch processing or systems that mostly do calculations benefit little from prototyping. Sometimes, the coding needed to perform the system functions may be too intensive and the potential gains that prototyping could provide are too small.

Prototyping is especially good for designing good human-computer interfaces. "One of the most productive uses of rapid prototyping to date has been as a tool for iterative user requirements engineering and human-computer interface design.

4.1. Tools

Efficiently using prototyping requires that an organization have proper tools and a staff trained to use those tools. Tools used in prototyping can vary from individual tools like 4th generation programming languages used for rapid prototyping to complex integrated CASE tools.

4th generation programming languages like Visual Basic are frequently used since they are cheap, well known and relatively easy and fast to use. CASE tools, like the Requirements Engineering Environment are often developed or selected by the military or large organizations. Object oriented tools are also being developed like LYMB from the GE Research and Development Center.

4.2. Screen generators, design tools & Software Factories

Also commonly used are screen generating programs that enable prototypers to show users systems that don't function, but show what the screens may look like. Developing Human Computer Interfaces can sometimes be the critical part of the development effort, since to the users the interface essentially is the system.

Software Factories are Code Generators that allow you to model the domain model and then drag and drop the UI. Also they enable you to run the prototype and use basic database functionality. This approach allows you to explore the domain model and make sure it is in sync with the GUI prototype. Also you can use the UI Controls that will later on be used for real development.

4.3. Visual Basic

One of the most popular tools for Rapid Prototyping is Visual Basic (VB). Microsoft Access, which includes a Visual Basic extensibility module, is also a widely accepted prototyping tool that is used by many non-technical business analysts. Although VB is a programming language it has many features that facilitate using it to create prototypes, including:

•An interactive/visual user interface design tool.

•Easy connection of user interface components to underlying functional behavior.

•Easy to learn and use implementation language (i.e. Basic).

•Modifications to the resulting software are easy to perform.

4.4. Requirements Engineering Environment

"The Requirements Engineering Environment (REE), under development at Rome Laboratory since 1985, provides an integrated toolset for rapidly representing, building, and executing models of critical aspects of complex systems."

Requirements Engineering Environment is currently used by the Air Force to develop systems. It is an integrated set of tools that allows systems analysts to rapidly build functional, user interface, and performance prototype models of system components. These modeling activities are performed to gain a greater understanding of complex systems and lessen the impact that inaccurate requirement specifications have on cost and scheduling during the system development process. Models can be constructed easily, and at varying levels of abstraction or granularity, depending on the specific behavioral aspects of the model being exercised.

REE is composed of three parts. The first, called Proto is a CASE tool specifically designed to support rapid prototyping. The second part is called the Rapid Interface Prototyping System or RIP, which is a collection of tools that facilitate the creation of user interfaces. The third part of REE is a user interface to RIP and proto that is graphical and intended to be easy to use.

Rome Laboratory, the developer of REE, intended that to support their internal requirements gathering methodology. Their method has three main parts:

- Elicitation from various sources (users, interfaces to other systems), specification, and consistency checking;

- Analysis that the needs of diverse users taken together do not conflict and are technically and economically feasible;

- Validation that requirements so derived are an accurate reflection of user needs.

In 1996, Rome Labs contracted Software Productivity Solutions (SPS) to further enhance REE to create "a commercial quality REE that supports requirements specification, simulation, user interface prototyping, mapping of requirements to hardware architectures, and code generation…"This system is named the Advanced Requirements Engineering Workstation or AREW.

LYMB

LYMB is an object-oriented development environment aimed at developing applications that require combining graphics-based user interfaces, visualization, and rapid prototyping.

PSDL

PSDL is a prototype description language, to describe the real-time software

REFERENCES

[1].C. MELISSA MCCLENDON, LARRY REGOT, GERRI AKERS: "The Analysis and Prototyping of Effective Graphical User Interfaces", October 1996.

[2].D.A. STACY: "Lecture notes on Rapid Prototyping", University of Guelph. Guelph, Ontario, August, 1997.

[3].ALAN M. DAVIS: Operational Prototyping: A new Development Approach. IEEE Software, September 1992. Page 71

[4].JOHN CRINNION, "Evolutionary Systems Development, a practical guide to the use of prototyping within a structured systems methodology", Plenum Press, New York, pg. 18, 1991.

[5].LAWRENCE J. NAJJAR, "Rapid Prototyping", IBM Corp. Software Usability Department R16B Atlanta, GA 30328, April 1990.