



*Accelerate Development
Reduce Time to Product
Automate Critical Tasks*

White Paper:

A Picture is Worth 1000 Words: Reduce Software Project Risk Through Prototyping

The ASHVINS GROUP, Inc.
6161 Blue Lagoon Drive, #340
Miami, FL 33126
305-264-4442
www.ASHVINSGroup.com

GSA Contract #: GS-35F-0561T
CAGE #: 49ZF1
D-U-N-S #:10-252-0355

INTRODUCTION

Carefully thought out and documented requirements are the back bone of a high quality systems development project. No single view of the requirements for an IT systems project provides a complete understanding of them. Combinations of textual and graphical requirement representations are required to fully understand the scope of an intended system. Prototyping is recognized as an effective, clear method of visually describing and demonstrating a project's desired functionality and features.

This paper explores the utilization of prototyping to model the requirements in order to effectively communicate requirements among project participants and bridge vocabulary barriers among different team members. The information provided will offer insight into the prototyping process to explain what makes a good prototype and why prototypes are important. Details will be provided to help readers incorporate prototyping into their development process and decide on the best type of prototype to build. And finally, prototype evaluation techniques will be reviewed and helpful tips will be provided to assure readers maximize the success of their prototypes and avoid pitfalls.

PROTOTYPING: WHAT AND WHY

Project leaders frequently experience the frustration of discovering a requirements problem late in the Software Development Process or after the application is delivered. The earlier you find problems in a software project, the better off you are. The cost to fix an error found during requirements or early design phase is orders of magnitudes less to correct than the same error found during testing (for study information see "Evaluating Software Architectures: Methods and Case Studies"; By Paul Clements, Rick Kazman, Mark Klein). Software prototyping offers the end user a chance to see how the system will be implemented before coding formally begins. Early feedback from users ensures that requirements have been properly understood by the software developers and full project team.

Prototyping is a development methodology based on building and using a model of a system to facilitate the system design and implementation. As with any other methodology, it is a tool that when used effectively will teach, measure, compare and organize. Prototypes are used for three major purposes:

- *As a requirements tool to clarify and complete the requirements.* A prototype is an excellent way to eliminate ambiguity and incompleteness in the requirements. Giving users a prototype can help point out problems with requirements before the application is constructed.
- *As a design tool to explore design options and alternatives.* Prototypes help system architects evaluate approaches, survey user interface options, mock up navigation and optimize system usability.
- *As a preliminary iteration of an application that may possibly be used to grow into the final application.* Prototypes may be used as a construction tool that becomes a subset of the product. Developers may begin with a prototype as a baseline which is then expanded into a completed product.

Project Managers, end users and other non-technical project stakeholders find that prototypes provide a way to look at the requirements in a meaningful way without the technical lingo used by software

developers. The prototype allows them to work with a tangible model as they review the functionality specified in the requirements. Software developers and architects use the prototyping process as a means to create a skeletal version of the system so that the prototype becomes the vehicle for designing the final version. This allows both developers and end users a visual format for communicating the look and feel of critical feature so that nothing is lost in implementing the requirements. Depending on the nature of the project, the prototype may or may not be “throwaway” coding. The prototype may serve as the first iteration of the product. Project complexity, schedule and budget may dictate how elaborate or rough the prototyping exercise may be.

INCORPORATING THE PROTOTYPE INTO THE DEVELOPMENT PROCESS

The prototyping process is part of Requirements Development. The first three phases, of the four phase process, for Requirements Development should be completed before the prototype is built.

- *Phase 1*- requirements elicitation: the process of identifying and understanding the needs and constraints of the different users for a proposed software system.
- *Phase 2*- requirements analysis: the step to create an understanding of the functionality, features, limitations and constraints of an intended system.
- *Phase 3*- requirements specification/documentation: a documented agreement between the intended system users and developers that thoroughly describes the system to be built.

The prototype is part of the last phase, verification.

- *Phase 4*- requirements verification: confirming step to assure that the requirements are fully understood and accepted by users, developers and other project team members.

It is important to note that prototyping does not replace phases 1-3 in Requirements Development. An entry condition required to start prototyping is the completion of a written requirements specification. The developers must know enough about the system in order to build a usable initial model than can be used to challenge the requirements for clarity, value and completeness. The different phases work together systematically to create the best possible system requirements that will be used to support the coding and testing efforts.

DECIDING ON A PROTOTYPING STYLE

Before the project begins, the project team must assess what type or style of a prototype will be created. They must decide if the prototype will be used for evaluation purposes only or if it will be used as an initial iteration of the application that will evolve as the project continues. General types of prototyping include: 1- throwaway prototyping (exploratory prototyping) and 2- evolutionary prototyping.

Throwaway or exploratory prototyping refers to the creation of a model that will eventually be discarded rather than become part of the finally delivered software. And since the project team will discard the prototype, it is built as rapid and cost effective as possible. The goal is to be quick and to forego robustness, reliability, performance and long term maintainability. A simple working model of

the system is constructed to visually demonstrate what the requirements may look like when they are implemented into a finished system and to help uncover gaps in the requirements. The largest advantage to a throwaway prototype is that it can be done quickly. An additional advantage 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. By seeing it in front of them, it is much easier to understand how the system will work.

In contrast, the evolutionary prototype provides a solid architectural foundation for building the product incrementally. The main goal when using evolutionary prototyping is to build a very robust prototype in a structured manner and constantly refine it. Therefore, an evolutionary prototype takes longer to create than a throwaway prototype that addresses the same functionality. The evolutionary prototype forms the baseline of the new system, and the improvements and further requirements will be built using the prototype. Evolutionary prototyping acknowledges that the project team does not yet understand all the requirements. The prototype is built using only those requirements that are well understood at the time. This technique allows the development team to add features, or make changes as the requirements are better defined. It creates “pilot” software that can evolve through use in the intended operational environment.

Throwaway and evolutionary prototyping can be combined in a software development project. The project team may use a series of throwaway prototypes to refine the requirements until they get to the point of well understood functionality. Then this knowledge may be implemented incrementally through an evolutionary prototyping sequence. What the project team *cannot* do is turn the low quality of a throwaway prototype into a functional evolutionary prototype. Evolutionary prototypes should be built on a stable and sustainable architecture.

BUILDING THE PROTOTYPE

An executable prototype may not be required to understand requirements. Throwaway prototypes can be created on paper to make them in a fast, inexpensive and low tech way. Paper prototypes can help users and developers determine if they have reached a shared understanding of the requirements. Sketching displays and outlining user responses on paper will quickly enhance communication in the project team and allow them to iterate through requirements. Paper prototypes are an excellent precursor to building electronic throwaway prototypes. If electronic prototypes are desired, several tools are available to quickly create a representation of the software functionality. These include:

- Programming languages such as Visual Basic
- Scripting languages such as PHP, Perl, ASP
- HTML
- Prototyping toolkits, screen painters and GUI builders

If the project team decides to build an evolutionary prototype, then the developers must use the same development tools selected for the project.

EVALUATING THE PROTOTYPE

Drafting a series of scripts that guide the users through the steps of the prototype will improve the effectiveness of prototype evaluation. These scripts will help guide and direct the users to cover all areas of the prototype and is a supplement to the general question “tell me how you like this prototype”. The scripts should require that the user perform specific tasks and highlight areas of the application that are most uncertain. After users perform the tasks, the following questions may provide additional insight:

- Is the implementation of the functionality what you expected?
- Is functionality missing or are any unnecessary functions present?
- Are there any alternate paths or error conditions that the prototype does not address?
- Is the navigation intuitive?
- Are the screen layouts and/or graphics useful?

It is important that the prototype users represent typical users of the intended system. The evaluation should include both inexperienced and power users. In addition to providing scripts, the project team reviewers can also watch users as they randomly work with the prototype to determine if they easily navigate the screens or if they appear to struggle. Reviewers must avoid the temptation to coach the users as they work with the prototype and encourage them to document all comments and suggestions.

PROTOTYPING SUCCESS FACTORS AND PITFALLS

The following **Success Tips** are offered to help effectively utilize prototyping:

❖ **Success Tip #1:** Include Prototyping in the Project Plan
Schedule time to prepare, evaluate and modify prototypes.

❖ **Success Tip #2:** Plan to Develop a Series of Prototypes
Prototypes are rarely complete enough on the first try.

❖ **Success Tip #3:** Don't Prototype Requirements Already Understood
If the requirement is clear, don't waste time- focus on the unclear items.

❖ **Success Tip #4:** Use Feasible Data in the Prototype Screens
The users who evaluate prototypes can be distracted by data that is not feasible and fail to focus on the important parts of the prototype.

❖ **Success Tip #5:** Don't try to unnecessarily perfect prototypes.
Prototypes are not intended to be the completed application; they should develop as quick and cheap as possible.

The primary **Pitfalls** associated with prototyping include:

■ **Pitfall #1:** Expecting a Prototype to Replace Written Requirements,
Written requirements that are clear, concise and testable are critical to the success of the project.

■ **Pitfall #2:** Using a Throwaway Prototype in the Product.

Throwaway prototypes are not designed to become part of the completed application; they are not robust or stable.

- **Pitfall #3** Assuming the Product is Done When the Prototype Works.
It may look like the real thing, but it cannot be complete if it is a prototype.

CONCLUSION

Prototyping can shorten development schedules, increase end user satisfaction, reduce requirements errors and overall help to produce high quality applications. Prototypes make the requirements tangible and help users visualize how the application will be implemented in order to clarify ambiguity in the requirements and to close gaps in described functionality. Successfully applying prototypes in a project will enhance communication between users and software developers and help refine system concepts. Prototypes contribute to the development process in the critical final phase of Requirements Development, requirements verification. Throwaway and Evolutionary prototypes can be combined or used separately for the project depending on the complexity of the application. Keep prototypes simple, flexible and extensible to derive the most benefits. Seek different levels of users (inexperienced and power users) to evaluate the prototype and provide feedback to the development team.

By instilling the prototype Success Factors and avoiding the Pitfalls as summarized in the previous section, the overall risk of project delays and budget overruns can be mitigated early during the Requirements Development process. Utilizing prototypes to communicate requirements proves that a picture can indeed be worth 1000 words.

ABOUT THE ASHVINS GROUP

The Ashvins Group Inc. is a team of IT consultants who design, develop, test and implement custom software and data management projects. The Ashvins Group is a Florida corporation since 2000 and classified as a Women Owned Small Business. We offer Full Life Cycle Application Development and Validation services with a specialty in healthcare, medical and clinical data management applications. Our expertise includes FDA and HIPAA compliant applications development.

The Ashvins Group applies disciplined application development industry standards combined with our unique Rapid Development Methodology (RDM) and project management practice to achieve client goals. As Development Accelerators, the Ashvins Group will enable clients to meet their critical objectives effectively while reducing time to implement and managing expenses. We utilize controlled processes and apply rigid documentation practices commensurate with healthcare industry compliance requirements and regulations. For additional information see our website at www.ashvinsgroup.com and contact us at info@ashvinsgroup.com.