https://doi.org/10.48009/1_iis_2005_10-14

# HANDS-ON PROTOTYPING IN SYSTEM ANALYSIS AND DESIGN

**Dr. Robert F. Zant, Illinois State University, rfzant@ilstu.edu**

## ABSTRACT

*The art of prototyping has evolved from the use of pen-and-paper layout charts to being the basis for Evolutionary Prototyping, a full methodology for developing software systems. While prototyping is discussed in introductory systems analysis and design courses, students gain a better appreciation of the technique by actually developing different types of prototypes. For example, prototyping is an excellent vehicle for demonstrating the overlapping of phases in the SDLC. This paper reviews the ways in which prototyping may be used in the system development life cycle and then presents examples of how prototyping can be infused into an undergraduate course in systems analysis and design. Students can develop discovery and user-interface prototypes using HTML and JavaScript modules. The development of functional prototypes using a scripting language and MS Access is also presented.*

**Keywords:** Prototyping, systems analysis, software development, systems engineering

## INTRODUCTION

Prototyping is now as fundamental to software development as it has been to systems development in other engineering fields. Satzinger, *et al*., state that prototyping is used "in almost every [software] development project in some way" [13, p. 137] and McConnell proposes that prototyping can be used "in one form or another on most kinds of [software] projects regardless of what other practices are used" [10, p. 573]. The use of prototyping has been developed to the level of being the basis for software development methodologies, such as the Evolutionary Prototyping lifecycle model [10, p. 147]. Sommerville states that the evolutionary prototyping methodology "is now the normal technique used for web-site development and e-commerce applications [14, p.175].

Prototyping is, thus, a very important topic for the Systems Analysis and Design course. However, the manner in which the prototyping concept is presented in systems development texts can be confusing to some students. It is the thesis of this paper that students gain a clearer understanding of the various roles of prototypes in the systems development life cycle (SDLC) if they are involved in the hands-on development of different types of prototypes. It is proposed that students in a Systems Analysis and Design course develop prototypes that are appropriate for use in the analysis phase of the SDLC and prototypes of the type that would be used in the design phase. Students should also develop an initial version of an evolutionary prototype.

## HISTORICAL PERSPECTIVE

Business computing began in earnest the late 1950s with the introduction of the IBM 1401 computer. The advent of the IBM 360 series of computers in the mid-1960s sparked rapid growth in the use of computers in business systems and consequently in the development of information systems. Dijkstra's famous 1968 letter on "GOTO-less" programming sparked great debate and inquiry into methodologies for developing programs and ultimately into methodologies for developing information systems [5]. Texts on the subject written in the 1970s presented what is today considered the Traditional Systems Development Life Cycle. The texts did not mention prototyping. Prototyping as a technique for software development entered the software engineering literature in the 1980s [1, 2, 9, & 12] but did not become a prominent topic in information systems texts until the 1990s. The development of web-based technologies and the increased emphasis on the rapid deployment of systems has fueled the growth in the use of software prototyping over the last decade.

## USE OF PROTOTYPING IN SOFTWARE DEVELOPMENT METHODOLOGIES

Software development methodologies may be broadly thought of as those that are specification based and those that are prototype based [14, p. 177]. In specification-based methodologies formal, detailed statements of the requirements for a system are developed. In prototype-based methodologies detailed specification statements are not developed. Instead, the specifications are embodied in the prototype.

### Specification-based Methodologies

Specification-based methodologies, such as traditional structured methodologies, use "throwaway" prototypes. Throwaway prototypes are developed for a specific, limited purpose and then discarded. This type of prototype can be used for different purposes in each of the phases of software development.

In the Planning Phase *Demonstration prototypes* are developed to portray the user interface of the proposed system. The purpose is to provide a "vivid demonstration" of proposed software to managers and customers to garner support for the project [10, p. 591].

In the Analysis Phase *Discovery prototypes* are used as an aid in determining user requirements. Allowing users to experiment with a prototype interface helps to refine requirements that were specified as the basis for the prototype's construction and elicit new requirements that may have been overlooked. *Feasibility prototypes* also are used in the analysis phase to evaluate new technology such as new software development tools or aspects of the proposed system architecture.

In the Design Phase *User-interface prototypes* are used to design the format of screens and reports and to design the human-computer dialogue. *Feasibility/Performance prototypes* are used to evaluate the design of system components such as database systems and schema, network architecture, algorithms, and system controls. These types of prototypes are particularly valuable when developing an innovative system or when using a new technology.

In the Implementation Phase *Feasibility/Performance prototypes* are used for the same purposes as in the design phase but would typically be more finely focused. Prototypes can also be used in the implementation phase to begin training users before the system is fully functional and to cross-test system components by comparing system test results to those obtained from corresponding prototypes.

### Prototype-based Methodologies

Prototype-based methodologies are centered on the development of an *evolutionary prototype* that would initially contain a minimal set of high priority features and then "is augmented and changed as new requirements are discovered" [14, p.174]. The prototype ultimately becomes the system. These methodologies perform "the analysis, design, and implementation phases concurrently and all three phases are performed repeatedly in a cycle until the system is completed" [4, p. 11]. Examples of this type of methodology are Boehm's Spiral Model and Rapid Application Development (RAD). In addition to developing an evolutionary prototype, prototype-based methodologies can incorporate throwaway prototypes much the same as the specification-based methodologies. In prototype-based methodologies throwaway prototypes (most likely *Feasibility/Performance prototypes*) would be constructed for special purposes as adjuncts to the evolutionary prototype.

## TREATMENT OF PROTOTYPING IN TEXTS

The treatment of the prototyping concept in systems development texts can be confusing to some students. The biggest problem is that, as shown above, throwaway prototypes are applicable throughout the software development process and prototypes with similar functions are used in different phases. For example, discovery prototypes and user-interface prototypes both deal with input and output, but with different focuses. Discovery prototypes use input and output in the analysis phase as a mechanism for eliciting and refining functional requirements; whereas, user-interface prototypes are used in the design phase to design actual screen formats and dialogues.

Second, because of the various ways prototyping may be employed in a software development life cycle, the topic typically appears several different times in a text. A discussion of prototyping will appear in multiple sections of a text which cover the various phases of the software development life cycle. In addition, prototyping as a methodology may also be included in a section of its own covering rapid software development.

Finally, while there is some consistency in the use of terminology, there is also considerable variation. In addition to the prototype classification scheme used above, a number or other terms have been used to describe various types of prototypes—*paper prototype* [12, p. 150]; *mock-ups* and *Wizard-of-Oz prototypes* [14, p.180]; and *patched-up, non-operational, first-of-a-series,* and *selected features prototypes* [8, p. 152]. The term *user-interface* prototype is sometimes used interchangeably with *discovery* prototype and, at least in one text, is also described as an *evolving* prototype.

The use of different types of prototypes for different functions, the similarity of some types of prototypes and the disjoint treatment of the concept in texts caused by its presentation within the context of various phases of the SDLC all contribute to students' confusion over the concept.

## HANDS-ON PROTOTYPING

As stated previously, it is the thesis of this paper that students gain a clearer understanding of the various roles of prototypes in the SDLC if they are involved in the hands-on development of various types of prototypes. It is proposed here that students in a Systems Analysis and Design course develop three types of discovery prototypes that would be applicable to the analysis phase of the SDLC, four types of user-interface prototypes that would be applicable to the design phase and at least one initial evolutionary prototype.

Three variations of discovery prototypes are to be developed. The first is a "paper mock-up" prototype. This type, in traditional methodologies, was called a "layout chart" (record layout chart, display layout chart, etc.).  While these are "low-tech," they have a long history of use in software development and "have been shown to be effective in helping users refine interface design and work through usage scenarios" [14, p.180]. The second type of discovery prototype is an "electronic mock-up." This type is developed using HTML forms (raw HTML or Dreamweaver, etc.) with no ACTION attribute. Thus, these are non-operational, as are the paper mock-ups. The electronic mock-up can be seen by students to be very similar in concept to the paper prototype but more "realistic." This highlights for the student the primary risk in using discovery prototypes. They tend to draw users and analysts into designing interfaces rather than defining requirements. DeMarco even recommended against using discovery prototypes in favor of using data dictionaries for this reason [3, p. 164]. The third type of discovery prototype to be developed contains multiple electronic mock-ups with navigation between the web pages. The primary purpose of this type of prototype is the validation of requirements. With this prototype the user can do a virtual walk-through of the identified functions for the system. This exercise aids users in revealing errors and omissions in the requirements that have been prototyped [14, p. 172].

Four variations of the types of prototypes used in design are developed. The first is a refinement of the electronic mock-up developed for analysis. In this case, though, the purpose is to design system components. So, the prototype is used to compare alternate designs, such as, different layouts, color schemes, etc. This is easily accomplished by comparing the prototypes developed by different students. The mock-up is then extended to incorporate design decisions on data formats with client-side editing being added using a package of Javascript functions that is available on the web [6]. This allows comparison of different designs such as editing data input item by item or editing an entire form or doing both [11].

The third type of design prototype focuses on prototyping a database system. MS Access is used by students to define tables, relationships, and data formats. Data can then be added and retrieved from the database to test the design's ability to fulfill the system's functional specifications.

The fourth type of design prototype adds real interactive capability to the prototype by using a set of Javascript functions developed by the author. The Javascript package provides support for temporary data storage (in a Javascript array) for a simple database schema. Support for multiple tables is provided along with functions for storing, updating, and deleting tuples. Tuples can be retrieved by primary key or non-primary key attributes. With this package, students can produce a working system prototype on a client computer without the need for a server.

Evolutionary Prototyping methodology is demonstrated by using the free version of a commercially available server-side scripting system [7] along with HTML, Javascript, and MS Access. The scripting system is easy to install: simply copy a CGI program to a web server and define a web directory to contain the prototype's files. In fact, students are encouraged to install the system on their own computer.

The prototype developed by the student will consist of HTML pages, script files, and an MS Access database. All three types of files are placed in the same web directory. The scripting language is easy to use and provides an SQL interface to the database and commands to produce formatted displays. But, it is also a commercial-level language offering support for encryption and session controls. This provides the opportunity for the student to construct an initial, limited prototype and then add more features in a second cycle as happens in evolutionary development.

## SUMMARY

Over the last decade prototyping has become a very valuable technique in software development methodologies. It is important that information systems students understand the various types of prototypes used and the roles they play in a SDLC. This paper suggests that this goal is better achieved by having students develop various types of prototypes. A set of prototypes to be developed as assignments in an undergraduate course in systems analysis and design and the software required for their implementation was presented.

## REFERENCES

1. Boar, B. H. (1984). *Application Prototyping*, Wiley
2. Boehm, B. W. (1988). A Spiral Model of Software Development and Enhancement, *Computer*, May, 61-67
3. DeMarco, T. (1978). *Structured Analysis and System Specification*, Prentice-Hall
4. Dennis, A., Wixom,, B. H., & Tegarden, D. (2002). *Systems Analysis and Design*, Wiley
5. Dijkstra, E. W. (1968) GOTO Statement Considered Harmful, *Communications of the ACM*, March, 147-148.
6. Form Validation, Retrieved April 20, 2005, from http://developer.apple.com/internet/webcontent/validation.html
7. Harris, R. (2005). ODBscript, Retrieved April 20, 2005, from http://odbscript.com
8. Kendall, K. E. & Kendall, J. (2003). *Systems Analysis and Design*, Prentice Hall
9. Lantz, K. E. (1987). *The Prototyping Methodology*, Prentice-Hall
10. McConnell, S. (1996). *Rapid Development*, Microsoft
11. Poley, S. (2005). Javascript form validation – doing it right, Retrieved April 20, 2005, from http://www.xs4all.nl/~sbpoley/webmatters/formval.html
12. Pressman, R. S. (1987). *Software Engineering*, McGraw-Hill
13. Satzinger, J. W., Jackson, R. & Burd, S. (2004). *Systems Analysis and Design*, Course Technology
14. Sommerville, I. (2001). *Software Engineering*, Addison Wesley.