Available online www.unicrossjournals.com

JOURNAL OF CONTEMPORARY RESEARCH (JOCRES)

RESEARCH ARTICLE VOL. 2 (1) ISSN:2814-2241

Date Accepted: 30th June, 2023

Pages 207 - 213

EXPLORING THE BENEFITS AND DEVELOPMENT ISSUES OF SOFTWARE SOLUTION PROTOTYPING

Fidelis I. Onah

ikonah80@yahoo.com

Department of Computer Science

Cross River University of Technology, Calabar

Abstract

The creation of reliable and efficient computer software that is acceptable to prospective users and stakeholders is a prominent feature in today's technology. As modern systems and computer products get more complex, the more the yearning for some kind of orderly development process that allow designers, users, developers and managers to have an early glimpse of the look and feel of the final product. In this research paper, some of the major issues in fabricating physical prototypes at all stages of new product design are assessed. Each of the phases in the development of prototype software is considered. The benefits and limitations of prototyping are investigated. The findings of an experimental evaluation of the prototype approach are presented. Finally, the guidelines for selecting a prototype are proposed. This paper is therefore a simple, practical assessment of the prototype approach to software applications development.

Keywords: Software solution, Prototyping, Prototype model, Benefits, Limitations

1. Introduction

Software (solution) development refers to the procedures involved in the creation of a computer application system. It can also be said to be the various steps a computer system passes through during its existence (Rice G. F. *et al*, 1988)

There are various software development life cycle (SDLC) models. Each of these models follows a series of steps in the development process. Examples include waterfall, spiral, Agile, V-shaped, Big Bang, Rapid Application, Joint Application Development and Prototyping models (Finoit, Inc., 2023). Irrespective of the life cycle model used, the different phases in the software development process include:

(1) Initial feasibility study – The system is conceived from a variety of sources; mostly from management.
(2) Analysis of software requirements – The systems analyst carried out a procedural study of the system's operations, and then prepares a problem definition so that a database can be built.
(3) Detailed specification of the software requirements.
(4) Software design – a set of specifications for the software describing exactly how the system will be built to meet the objectives established for it by whoever conceived the system.
(5) Programming – The systems implementer selects appropriate languages, tools and techniques used in the implementation.
(6) Testing the product
(7) Maintenance of the finished application (Finoit, Inc., 2023).

The determination of what the customer or user of a software package exactly requires has been one of the major problems of software engineering. For complicated and large systems

without manual or existing system, it is even more difficult to determine and exhaustively specify these users requirements.[2] Discussions on this issue have always centered on the importance of maintaining a dialogue between the designers and the customers. To meet these challenges, prototyping is strongly recommended (Mary Alavi, 1984).

1.0 What is Software Prototyping?

Software prototyping is the creation of a preliminary version of a software program being developed in order to allow certain aspects of the system to be investigated. A prototype may be defined as "an early sample, model, or release of a product built to test a concept or process" (Wikipedia, 2022; Wikipedia, 2023). This method models the software to be created through maintaining an interactive dialogue between the designer and the customer after the initial specification. Thus, intended users are allowed to gain a clearer idea of the nature of the services and give a feedback in terms of their needs and requirements. A prototype program is constructed that implements the dialogue as specified, but often without the full computation required in the complete program. The requirement specification for the system can then be updated to reflect this feedback and so increase confidence in the final system. Often the user finds that the dialogue implemented by the prototype is incomplete or difficult to use. Changes are made in requirements, and the prototype is modified to implement the improved dialogue. Ultimately, when the user is satisfied with the dialogue, then the computational subprograms are fully implemented so that correct results are produced (Dodd W. P., 1980).

As already pointed out, prototyping is the most efficient tool to use for new systems which cannot be specified in detail because the user is unable to define his needs. One reason for developing a prototype is that it is impossible to "get it right" the first time; and designers must plan to throw away the first product in order to develop an optimal finished product.

The degree of completeness and the techniques used in prototyping have been in development and debate since its proposal in the early 1970s.

2.0 Prototyping Process Model and Methodology

The diagram below (fig. 1) shows the phases of a typical prototyping process model.

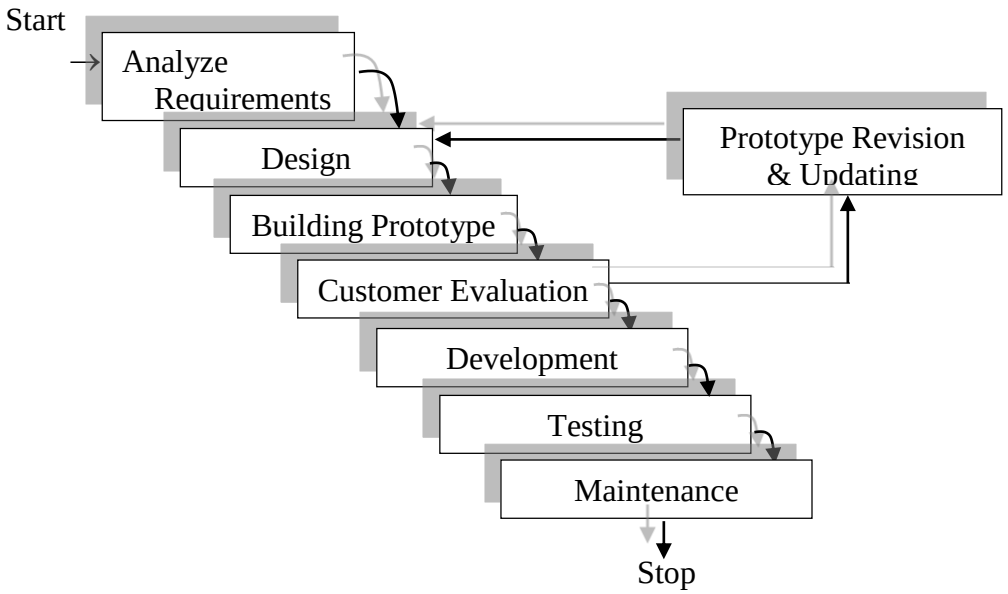

Fig. 1: The Phases of the Prototype Approach

The steps of the prototyping method are discussed as follows:

Step 1: Gather and analyze requirements

The designer contacts the end-users to learn their requirements, develop preliminary logical designs and write the system specifications. Since the designer will submit a working model of the systems for testing, the design can contain flaws or omissions which will be resolved later.

Step 2: Design

Software design is the specification or construction of a technical, computer-based solution for the business requirements identified in the requirement analysis. It accurately translates a customer's requirements into a finished product; and serves as a foundation for all development and maintenance phase steps that follow. Temporary forms are developed and input screen and databases assembled (Donald H. Sanders, 1985).

Step 3: Building Prototype

Prototype construction is the development, installation, and testing of system components (IT Chronicles Media Inc, 2023).

Step 4: Customer Evaluation

The prototype system is turned over to a limited number of end users for their testing and evaluation.

Step 5: Take a decision: Is the customer satisfied?

If Yes,

go to step 6

else

Refine prototype;

Loop back to step 2

The prototype is reworked (refined) based upon the experiences of end-user tests. If the prototype system design is maintained on the computer, changes in forms, input screens, and procedures can be made easily.

Revision and upgrading may be repeated several times until an acceptable system is developed.

Step 6: Development

This is the process of creating the computer software using one or more specific programming language(s) (IT Chronicles Media Inc, 2023).

Step 7: Testing

This ensures that the prototype application programs written and tested in isolation work properly when they are integrated into the total system. Errors, gaps, or missing requirements are identified here.

Step 8: Maintenance

Implemented systems and programs are usually subject to continual change and must therefore be maintained. This modification and improvement must be a cooperative effort between those served by the system or program (customers) and those responsible for maintaining it (Jeffrey L. Whitten Lonnie *et al*, 2001).

The prototyping methodology translates to the following algorithm (Fig. 2).

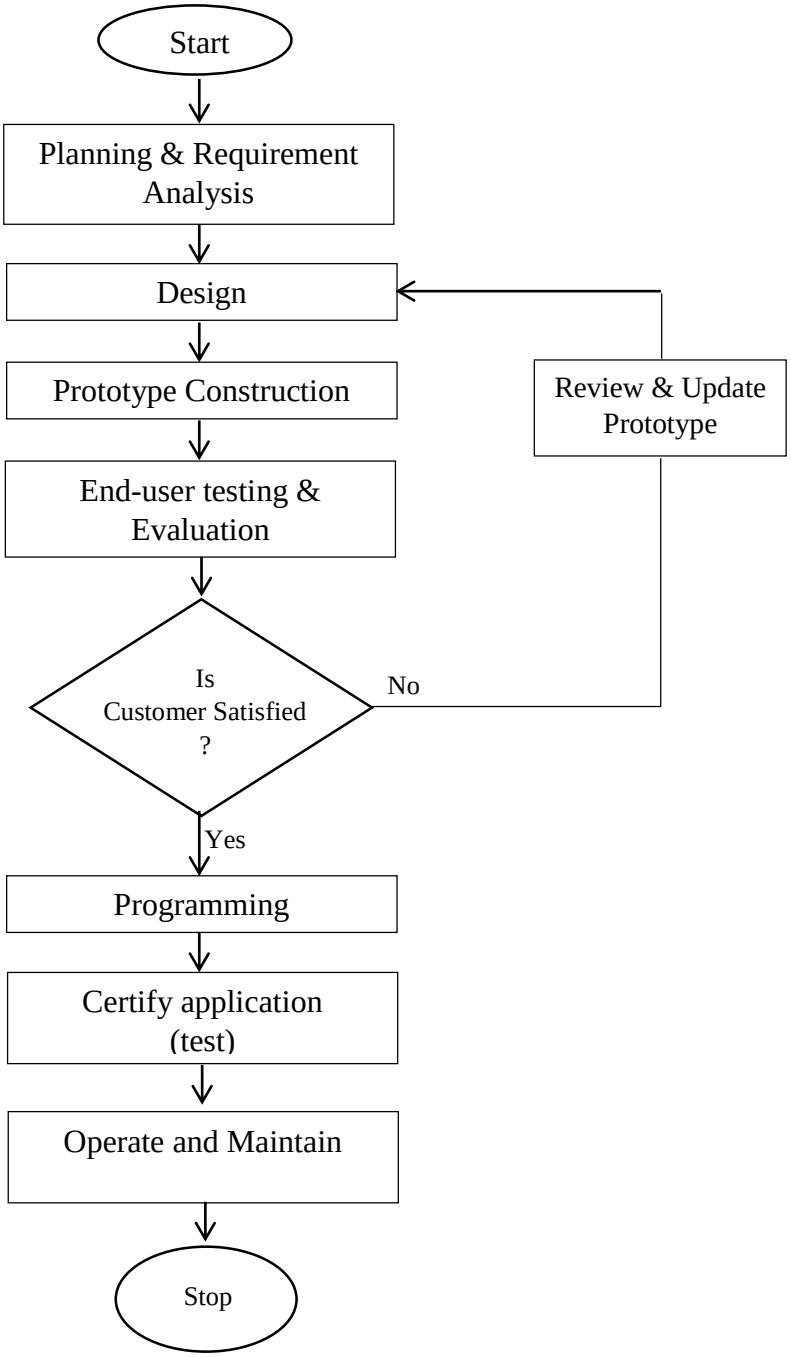

Fig. 2**:** Algorithm of the Prototyping Method

3.0 Evaluation of the Prototyping Model

3.1 Experimental Results

Based on the experimental evaluation of the prototype approach (Wilkinson G. G. and Winter flood A. R., 1991), the following are the findings:

1. Prototyping users are usually more satisfied with their level of participation in the design process and perceive less conflict between users and designers.
2. Prototyping users have more favourable perceptions and attitude towards the design process relative to users of other life-cycle models.
3. Users of a prototype have a higher ease of communication with the designers and have a higher understanding of the design project.

4. Prototype designers have more difficulty managing and controlling the design process than designers of other life-cycle models.
5. The designers employing prototype perceive a higher degree of change in user specification during the design process.

There is a higher utilization of the system designed by the prototyping approach than the system designed by other life-cycle approaches.

3.2 Benefits of Prototyping

The main benefits of prototyping to the educational, financial, transportation, government, manufacturing and health services industries, etc. include:

1. **Better communication and rapport:** Prototyping provide a working model of the system; so users get a better understanding of the system being developed and its intended purpose before resources are used for the development of the entire system. Better understanding between users and developers lead to a better working relationship between them. This results in final product that is more likely to satisfy users' desire for look, feel and performance.
2. **Early feedback:** Prototyping allows software designers to get valuable feedback (reviews) from the users (in terms of their needs and requirements) early in the project development process. The feeling of apathy towards the system by the users is removed by users' active participation and commitment. In this way, developers can steer the product in the right direction.
3. **Reduced time and costs of development:** Early determination of what the user really wants before resources are committed to the project can result in faster and less expensive software. Flaws (or defects) detected later in the development process cost exponentially more to implement.
4. **Validation before development:** Customers can validate the prototype at the earlier stage and provide their inputs and feedback. So, developers can have some insight into the accuracy of initial project estimates and whether the deadlines and milestones proposed can be successfully met. Potential problems and opportunities can respectively be detected and exploited as well (Ruchi Goel, 2022).
5. **Clarification of user set goals:** Identifying the prospective users' set goals reveals to designers how usable and valuable their product are to the end users in real-world scenarios. Missing, confusing or difficult functionalities can be identified easily and fixed or clarified to address users' pain points; and ensure that the final product is not only beautiful on the outside, but functional and enjoyable on the inside.

3.3 Limitations of Prototyping

1. It is difficult to fill all the requirements of the prototype software initially. Too much dependence on prototype can lead to insufficient requirement analysis. When purpose and scope are ill-defined by the user, there is potential for misunderstanding.
2. Limited in capabilities: Users may sometimes expect the performance of certain features included in a prototype to be the same as the final system. If those features are removed from the specification for the final system, it can lead to conflict. Unrealistic user expectations may result in disappointments.
3. Prototypes are difficult to manage and control because the form of the evolving systems, number of revisions to the prototype and some user requirements are

unknown at the beginning. And it is difficult to predict how the system will work after development.

4. **Expansion in scope:** Prototyping large software application is difficult because it is not clear how aspects of the system to be prototyped are identified and boundaries set. The complexity of the system may increase as the scope expands be4yond original plans.
5. **Extra costs and development time:** Prototyping can lead to changes in requirements and development methodologies. This suggests throwing away parts of the prototype and redesigning them; which incurs extra cost. When companies need to retain and/or retool due to change in development methodologies, the start-up costs for building a prototyping development team may be high. The time and cost saving benefits of prototyping can be lost if sophisticated software prototypes are employed.
6. **New bugs:** Modifying an existing system may introduce new bugs in parts that used to work correctly. It may also require more testing and debugging as the prototype is extended to the full system than if all parts of the system is designed and coded at one time.

3.4 Criteria for Selecting the Prototyping Method

The following guidelines for selecting a prototyping strategy are proposed:

1. Prototyping should be employed only when users are able to actively participate in the project.
2. Prototyping effort should be undertaken by designers and users who either have the experience or are given training on the use and purpose of prototyping.
3. Prototypes should become part of the final system only if the developers are given access to prototyping support tools.
4. If experimentation and learning are needed before there can be full commitment to a project, prototyping can be successfully used (Bill C. Hardgrave, Rick L. Wilson and Ken Eastman, 1999).

4.0 Conclusion

This research paper has looked at the benefits, limitations and development issues of prototype application software.

Building an operational prototype offers us the opportunity to maintain the customer-designer dialogue after the initial specification unlike other life cycle approaches. Thus, intended users are allowed to gain a clearer idea of the nature of the services and give a feedback in terms of the needs and requirements. The requirement specification for the system can then be updated to reflect this feedback and so increase confidence in the final system (Mary Alavi, 1984).

Prototyping is recommended when user requirements are ambiguous, and where there is need for experimentation before resources are committed to the development of a full scale system[5]. It was obvious that the price paid for not prototyping, in terms of lack of confidence, etc., may outweigh the cost of prototyping when viewed (quantified) in monetary terms (Mary Alavi, 1984). But prototyping should be undertaken by designers and users who are well informed about it; otherwise frequent alterations in user requirements may frustrate designers if they are not prepared to accept such changes and view them positively (Mary Alavi, 1984).

References

Bill C. Hardgrave, Rick L. Wilson and Ken Eastman (1999). *Toward a Contingency Model for Selecting an Information System Prototyping Strategy*. Journal of Management Information Systems, Vol. 16, No. 2

(Fall, 1999). Retrieved from https://shareok.org on 05/06/2022.

Dodd, W. P. (1980). *Prototype Programs*. The Computer Journal. Vol. 13 (2). https://ieeexplore.ieee.org.

Donald, H. Sanders (1985). *Computers Today*, Third Edition. McGraw-Hill, Inc. pg. 471-479.

Finoit, Inc. (2023). *SDLC Models & Methodologies*. 6565 N MacArthur Blvd, STE 225 Irving, Texas, 75039, United States. Retrieved from https://www.finoit.com on 05/06/2022

IT Chronicles Media Inc. (2023). *What is Software Development?* Retrieved from https://itchronicles.com on 05/06/2022.

Jeffrey L. Whitten Lonnie, D. Bentley and Kevin C. Dittman (2001). *Systems Analysis and Design Methods*, 5th Edition. The McGraw-Hill Companies, 1221 Avenue of the Americas, New York, NY, 10020.

Maryan Alavi (1984). *Prototype Information Systems*. Communications of the ACM, p. 27.

Rice G.F., Turner W. S. and Cashwell, D. F. (1988). *Systems Development Methodology*. Revised Edition, North-Holland/American Elsevier.

Ruchi Goel (15th April, 22). *Why is Prototyping Important?* Retrieved from https://blog.zipboard.com on 30/05/2022.

Wikipedia (2023). *Prototype*. Retrieved from https://en.m.wikipedia.org on 03/06/2022

Wilkinson, G. G. and Winterflood, A. R. (1991). *Fundamentals of Information Technology*. John Wiley and Sons Ltd. Pg. 105.

.