Software Prototyping: A Strategy to Use When User Lacks Data Processing Experience

Peter M Ogedebe, Babatunde Peter Jacob

ABSTRACT

A prototype is a working physical model of a system and it serves as a preliminary version of the system. One of the major challenges faced by developers is to know when to use prototype and when not to. This is so considering that there are many disagreements among developers on the proper use of prototyping. This study looked at the question “Is prototyping the best strategy to use when the user lacks data processing?” There is an accepted assumption about users by application developers; they think that users are naïve with little understanding of data processing. It is very important for developers to consider end users when developing systems; this is because the end users are the major beneficiary of the system. The paper concluded that the knowledge of Data processing will be very helpful in prototyping and lack of it does not mean that prototyping is the best strategy to use in software development.

Keywords: Prototyping, Users, Data Processing, Software engineering.

1. INTRODUCTION

A prototype is a working physical model of a system or a subsystem, a prototype serves as a preliminary version of the system or component from which requirements are extracted and on which subsequent versions are based. Sprague and McNurlin in Likin and Shoval (1987), defined prototyping as “an iterative process of creating quickly and inexpensively live and working models to test out requirements and assumptions…… If the process results in an actual production system it is called "add-on" or evolutionary prototyping. If, however, the prototype is discarded and a production system is constructed, it is called throw-away prototyping.”

Prototype comes in different forms, it can be paper-based or computer based. The process of developing a prototype is called prototyping.

There are 4 major prototyping methodologies which supplement the traditional systems in the development life cycle in use today. These are:-

- Illustrative: This produces only mockups of reports and screens.
- Simulated: This simulates some system functions but does not use real data or a database, model not implemented.
- Functional: performs some actual system functions and uses real data and/or a database, model not implemented.
- Evolutionary: produces model(s) that become part of the final operational system.

A prototype moves the developer and customer toward a quick implementation. It begins with requirement gathering. Nwaocha (2008) noted that meetings between developer and customer are conducted to determine overall system objectives and functional and performance requirements. The developer then applies a set of tools to develop a quick design and build a working model (the prototype) of some elements of the system. The customer or the user “test drives” the prototype, evaluating its function and recommending changes to better meeting customer needs. After repeated processes and iteration, an acceptable model is derived. The developer then moves to produce the prototype by applying many of the steps involved in the classic life cycle.

In object oriented programming, software engineering methodologies helps in creating rapid prototypes and programs by using library of reusable objects (ie data structures and associated procedures).

2. PURPOSE OF PROTOTYPING

Prototyping enable a software Engineer or a system developer to gather information about the user’s requirement by allowing the user to interact with the prototype. Therefore it serves a preliminary version of the system from which requirements are extracted and on which subsequent versions are based (umsl.edu, 2003). It has been observed 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 (Wikipedia, 2012) Crinnion (1991), asserted that:
• 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.

It is also believed that 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 mostly do calculations benefit little from prototyping. That prototyping could provide are too small (Crinnion, 1991).

It is also argued that Prototyping is especially good for designing good human-computer interfaces. Overmeyer in Wikipedia, (2012) stated that "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."

Advantages of prototype are many, it can be used as a communication tool between the developers and the users to overcome the problems of late delivery of software products, over – budgeting and production of product with less functionality than desired, all these have to do with requirement management practices. Prototyping may also help to overcome the problems of lack of user inputs, incomplete requirements and changing requirement.

Prototypes makes it easier for developers to make corrections and changes to the product being developed, this is usually based on the users feed back. Other benefits as reported by many systems development literature include:

• Provides a process to perfect the requirements definition.

• Provides a formal specification embodied in an operating replica.

• More enthusiastic and constructive end-user, customer participation in requirements activities.

• Improved morale of end-users, customers, and developers.

• Greater level of user satisfaction with systems development.

• Users better prepared for later stages of development due to familiarity with prototype.

• Delivery of early proof-of-concept.

• Prototype may be easily changed and even discarded.

• Allows productive work to proceed despite initial uncertainties.

• Demonstrates progress at an early stage of development.

• May provide early training for future users of the system.

• May produce some useful deliverables even if the project runs out of time or money.

• Should result in a product that is a better fit for the customer's requirements.

Systems produced through prototyping may be judged easier to learn and easier to use. On the over all, the cost of prototyping vary (UMSL.Edu, 2003).

Despite the many advantages and benefits of prototyping, many disadvantages and potential risks have been identified. The primary and major concern is the fact that users will again and again ask for changes. This is because as they reexamine the prototype they are likely to notice new features they like to include in the final product. Other major concerns that have been identified are (UMSL.Edu, 2003):

• Can result in unrealistic schedule and budget expectations.

• Iterative nature makes it difficult for management to plan and schedule.

• Can bias the system analysis process. If the prototype is computer-based manual alternatives are unlikely to be considered.

• Working prototypes may lead management and customers to believe that the final product is almost ready for delivery.

• People can begin to think of the prototype as the solution.

• The excellent (or disappointing) performance characteristics of prototypes may mislead the customer.

• Prototypes generally lack security, auditing, and other controls, and data integrity may be difficult to ensure.

• Often inefficient and difficult to maintain.

• Tendency not to document.
Customers may not be prepared to provide the level or frequency of feedback required for iterative prototyping. Prototyping is applicable only to a limited class of problems. It is valuable when heavy human machine interaction occurs. It is far less beneficial for large batch oriented processing or embedded process control applications.

3. END USER PROTOTYPE

This paper sets out to address the hypothesis, “Software prototyping: A strategy to use when user lacks data processing experience”. In developing any system, it is always very important to consider the end user who is always the major beneficiary of the product. Many system development methods usually assume equality among users in terms of data processing understanding; however microcomputers and information centres have helped to upgrade many users from naïve to sophisticated users. This assumption was challenged by Pliskin and Shoval (1987) in their paper “End user prototyping, sophisticated users supporting system development.” They asserted that users should no longer be considered universally ignorant of information systems. They also advised that IS developers are to take advantage of the increased sophistication of users. Wasserman in Pliskin and Shoval (1987) noted that user dissatisfaction is a major issue of concern in System development life cycle.

It is believed that most users today have moved from naïve to at least computer literate individuals. This is true of most graduates of educational environments who are familiarized with computer in their institutions, Rockart & Flannery (1983). There are a number of categorization schemes for users, Rockart & Flannery proposed a six category view of users, these include:

- None programmers end users;
- Command level users;
- End user programmers;
- Functional support personnel;
- End user computing (EUC) support personnel and
- Data processing (DP) Programmers.

Marins in Pliskin and Shoval (1987) had 3 categories of users, these are:

- Non Programming end users;
- DP Amateurs and;
- DP Professionals.

Pliskin and Shoval used a two dimensional framework to illustrate this.

Table 1: Showing Categories of users.

<table>
<thead>
<tr>
<th>Programming Skills</th>
<th>DP Professionalism</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>IS Builders</td>
</tr>
<tr>
<td>No</td>
<td>Professionals</td>
</tr>
</tbody>
</table>

The goal of user centered design according to Gould and Lewis (1985), Kara (1997) is the development of usable system. There are many approaches to user involvement. However Gould and Lewis have presented the following principles:

- Early focus on users and task.
- Empirical Measurement.
- Iterative design.

Most users are regarded as sophisticated because many of them have had formal computer training as graduates of engineering and business schools or they have been able to overcome computer illiteracy. Most of them use technology to support their work at office. The advantage of this trend is the reduction of application development backlogs and its maintenance loads (Martin, 1982). Rivard and Huff (1984) have observed that without real DP credentials, end user programmers produce user developed applications (UDA) that are deficient in documentation, backup security and integrity. Users and builders are the two roles that are usually mentioned in relation to prototyping, Jenkins (1985). The architect is IS professional who is responsible for prototyping the users requirement to ease communication with the builder, Mason & Carey (1983). The use of an architect suggest that perhaps that the user is necessarily unsophisticated. End user prototyping (EUP) is defined as the “iterative, quick and inexpensive creation (by a sophisticated user) of live, working models to define and test system requirements.

4. WHAT IS DATA PROCESSING?

Data processing is the collection and manipulation of items of data to produce meaningful information (French, 2006). The term according to French is also more commonly associated with specialist business tasks of this nature such as sales order processing, purchase ledger processing and payroll processing. Most modern data processing utilize
electronic means today. Hence the term electronic data processing (EDP) is directly linked with the term Information technology. Information Technology as defined by the Information Technology Association of America (ITAA) (enotes.com, 2011) is “the study, design, development, implementation, support or management of computer-based information systems, particularly software applications and computer hardware.” French also asserted that although EDP is part of IT, it is not sensible to think of EDP as being a water tight compartment.

Data processing is also viewed as changing one format of data into another format for better maintenance as well as effective analysis and study. According to Studio Access (2009), “It is a process of converting data into information as well as it can convert information into data, for example consumers in many cases give their opinions (data) from different sources. Information system can use this data as input to produce information as output, this activities is data processing.”

Data processing is very important in market research, most business gathers abundant data and analyse these data to get information that will help them increase their profit and cut down cost. Data utilized from this process are presented in a methodical fashion to make it easy to understand, analyse and act upon. The different approaches to data processing as enumerated by studio Access (2009) are:

- Editing:- This involve deciding which data is significant after accumulating data from the different sources.
- Coding:- This involve the aligning of data into a particular system to make it easy to comprehend, this method can also be referred to as netting or bucketing.
- Data Entry:- This is the entering of data into software that does the tabulation. This is done usually after decision has been made on a code validation. This is the second phase of cleaning, it involves a thorough quality check, and data is double checked to ensure that the procedure is done correctly.
- Tabulation:- This is usually the final step in making the product, data is tabulated in orderly format to ensure that a thorough analysis is done.

Data processing services is also seen as “giving out data from one data arrangement to another, data digitizing and data capturing. It also involves converting raw data entry into specified format of information. The quality of data entry form, word processing, image processing and data conversion may be improved by outsourced data processing” (Studio Access, 2009). Data processing outsourcing is very helpful in many BPO industries. Data processing outsource services include Data conversion, Data entry, survey processing, word processing, Database management, script processing, image processing and forms processing.

Factors that determine when to Prototype:

There are many variables that are used to determine when to prototype and when not to. In a study by Corey and Corey (1989), found out that prototype was the best strategy to use for Online development; all new development; when users expectations or requirement are not clear and when the appropriate tools are available. Hardgrave and Wilson (1993) in Hardgrave (1995) identified 18 guidelines or decisions variables that determine whether to prototype or not, these are:

- Clarity of requirement;
- Requirements Stability;
- System mode;
- Project duration;
- Innovation;
- Project size;
- Project impact;
- Performance requirement;
- User involvement;
- User’s experience with prototyping;
- User’s experience with computers;
- Number of users;
- Impact of users;
- Developer’s familiarity with application domain;
- Developer’s experience with prototyping.
- Management support;
- Project feasibility; and
- Tool availability.

Hardgrave also noted that ‘sophistication of user’, ‘computer literacy of user’, and ‘user’s DP experience were considered in the study’s respondents in the decision to use prototype. Users experience here refers to the user’s exposure to computers and computerized systems. Lack of automation experience is seen as a risk inducer.

It is believed that prototyping is best when a customer is able to define a set generic objective for software but is not able to identify detailed input processing, or output requirement. The developer may not also be sure of the efficiency of an algorithm, the adaptability of an operating system or the form a of human machine interaction that will take place (Pressman, 1987).

Prototyping makes it possible for the developer to create a model of the software he wants to built. The models in most cases can take any of these three forms; paper prototypes, working prototype and existing program.
Like the different approaches to software development, prototyping begins with requirement gathering. This involve meeting of the developer and customer to define the objectives, functions and performance requirements. The different steps involved in prototyping as identified by Pressman include:

- Requirement gathering.
- Quick design.
- Building of the prototype.
- Evaluation and refinement requirement.
- Engineer the product.

The requirements gathering involve the meeting of the developer and customer to define the overall objectives for the software; identify requirement that are known and outline areas where further definition is mandatory.

The quick design phase focuses on the representation of the aspects of software visible to the user; these include both the inputs, approaches and outputs formats. The quick design leads to the construction of a prototype, the prototype is evaluated by the customer/user and designer and it is used to refine requirement for the software to be developed. The prototype is tuned to satisfy the needs of the customer while at the same time it enable the developer to understand better what needed to be done.

5. SUMMARY

The four major prototyping methodologies which supplement the traditional systems developing are illustrative, simulated, functional and evolutionary methods user will Prototyping can be used as communication tool between the developers and the users to overcome the problems of late delivery of product. The major disadvantage of prototyping is that users will again and again ask for changes.

Many factors that have been highlighted as determinant on when to prototype are:

- Clarity of requirement; Requirements Stability; System mode; Project duration; Innovation; Project size; Project impact; Performance requirement; User involvement; User’s experience with prototyping; User’s experience with computers; Number of users; Impact of users; Developer’s familiarity with application domain; Developer’s experience with prototyping. Management support, Project feasibility, and Tool availability. Hardgrave and Wilson (1993) in Hardgrave (1995).

Since most users today have moved from naïve to at least computer literate individuals. Lack of Knowledge of data processing experience may not necessarily a requirement for the use of prototyping; however the knowledge of data processing may be helpful for the purpose of prototyping.

When a prototyping approach is used, it serves as a learning tool. Dearnly and Mayhew (1983) stated that a user with little or no previous contact with computers can gain valuable experience with a prototype otherwise difficult computer jargon can be simplified by illustration. Martins in Senn (1990) was quoted to have made one or most far reaching comments on the need to prototype in recent times thus “I have never seen an example of prototype being given to end users without end users changing it. So if you build a system without a prototype, you are probably going to build it wrong.”

6. CONCLUSION

Prototyping enables software Engineer or a system developer to gather information about users’ requirement by allowing the user to interact with the prototype. Since the ‘sophistication of user’, ‘computer literacy of user’, and ‘user’s DP experience’ are factors that affect the decision to prototype or not. The Knowledge of Data Processing will be very helpful in prototyping and lack of Data Processing Experience does not mean that prototyping is the strategy to use.

REFERENCES


