Prototyping and Software Development Approaches

Mahil Carr
Department of Information Systems
City University of Hong Kong
83 Tat Chee Avenue
Hong Kong
Tel: (852) 2788 7540
Fax: (852) 2788 8694
E-mail: iscarr@is.cityu.edu.hk

Dr. June Verner
College of Information and Technology
Drexel University
4131 Chestnut St
Philadelphia PA 19104
USA
Email: june.verner@cis.drexel.edu

Abstract

Researchers have provided a number of different definitions, process models and classificatory schemes for both prototypes and prototyping approaches over the past two decades. Because there tends to be some confusion in the use of prototyping terms, in this review we attempt to place prototyping in context and delineate evolutionary prototyping approaches from other kinds development approaches that may have prototypes and prototyping strategies embedded within them. We consider what prototypes are, what the prototyping process is, and how software development approaches adopt prototyping for exploration, experiment or evolution. We provide a classification of the software development approaches that include prototyping of some kind. Within this discussion we review experimental prototyping, exploratory prototyping and evolutionary development.

KEYWORDS : prototyping, software development, prototyping definition and classification

1 INTRODUCTION

The classic waterfall model and its variations assume a software development project where work steps can be clearly detailed before they are executed. In an attempt to overcome the shortcomings of the waterfall model many new software development approaches such as iterative enhancement (Basili and Turner, 1975), rapid prototyping (Gomaa, 1983) evolutionary prototyping and incremental development (Floyd, 1984) have been suggested. Software development approaches incorporating prototyping have gained respectability as they have proved to be able to dynamically respond to changes in user requirements (Floyd, 1984) reduce the amount of rework required and help control the risk of incomplete requirements (Floyd, 1984). Researchers have also noted that prototyping enables us to partition the development process into smaller, easier to handle steps (Kaushaar and Shirland, 1985), is cost-effective (Boehm et al., 1984, Gordon and Bieman, 1994, Palvia and Nosek, 1990), improves communication between development personnel (Alavi, 1985) helps determine technical feasibility (Floyd, 1984), is a good risk management technique (Tate and Verner, 1990) and results in greater user involvement and participation in the development process (Naumann and Jenkins, 1982).

There is much confusion in the prototyping literature in the use of the prototyping terminology. Researchers have described various process models and classificatory schemes for prototypes and prototyping over the past two decades. The exact meaning of, and the differences between terms such as iterative, rapid, evolutionary, throwaway, interface, incremental, and mock up prototyping tends to be blurred. In this review we attempt to understand and place the terms in context.

Floyd's (1984) systematic review of 'prototyping', has been the foundation for much serious work on prototyping and therefore we use his schema to provide a skeletal framework into which we add adaptations and refinements.

Prototypes are used in many disciplines, for example engineers fabricate prototypes of products to explore and control uncertainty in the design of a product, or in order to explore difficulties in the production process before the eventual large scale manufacture of the product. The software industry has adopted this industrial technique to construct *prototypes* as models, simulations, or as partial implementations of systems and to use them for a variety of different purposes, e.g., to test the feasibility of certain technical aspects of a system, or as specification tools to determine user requirements.

Prototyping, on the other hand, can be viewed a 'process' (Floyd, 1984) which is either a well-defined phase within the software development life cycle, or is an 'approach' that influences the whole of it (Budde et al., 1992). The prototyping process can encourage the efficient development of applications by breaking a complex, and often ill-defined problem into several comprehensive yet smaller and simpler parts (Kaushaar and Shirland, 1985). A prototyping development approach can help build, and subsequently refine, a product to meet end-user or market expectations.(Gomaa H., 1983).

In this review we consider what prototypes are, what the prototyping process is, and how software development approaches adopt prototyping for exploration, experiment or evolution. We provide a classification of the software development approaches that include prototyping of some kind (see Figure 1 below).

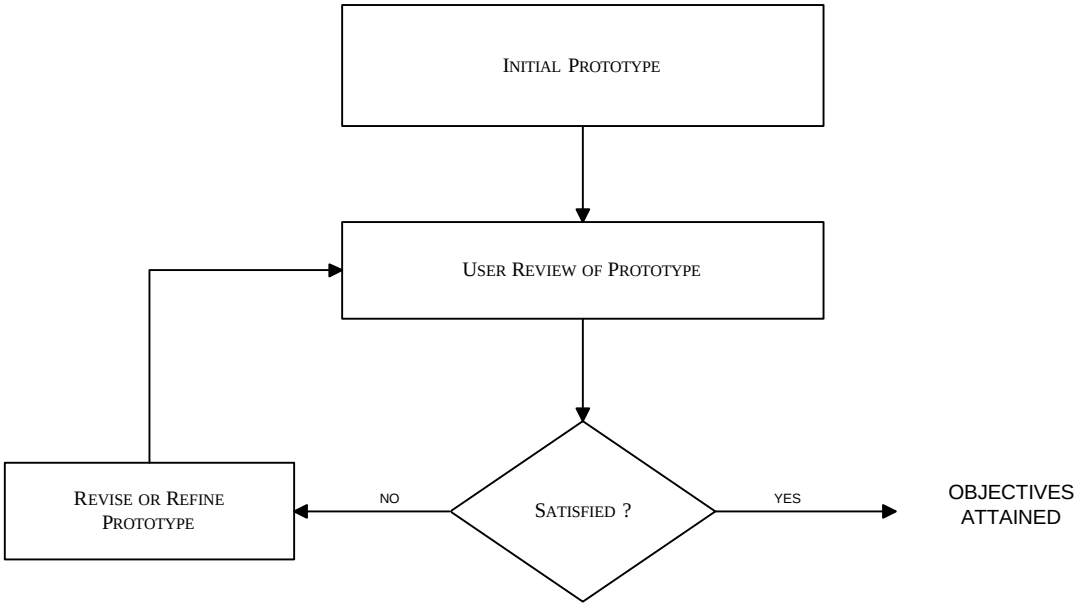

Figure 1: The Prototyping Process

Within this discussion we review:

1) the experimental prototyping approach;

2) exploratory approaches including rapid throwaway prototyping and the spiral model; and,

3) evolutionary software development including both incremental development and evolutionary system development.

2 PROTOTYPES

Prototypes are ‘instruments’ used within the software development process and different kinds of prototypes are employed to achieve different goals. The ‘product’ prototype has been variously defined within the prototyping literature and an early definition is that of Naumann and Jenkins (1982) who consider an information systems prototype to be:

> *“... a system that captures the essential features of a later system, is the most appropriate definition of a prototype. A prototype system,*

intentionally incomplete, is to be modified, supplemented, or supplanted."

Budde *et al*. (1992) classify along two different vectors - according to the purpose they fulfil and according to the manner of their construction. Four types of prototypes: presentation prototypes, prototype proper, breadboard prototypes and pilot system prototypes are identified according to the different tasks they accomplish. The *presentation prototype* is one which is presented to prospective clients by a software manufacturer in order to convince them of the feasibility of a new project. As such it is their first impression of the future system. The *prototype proper* is constructed and tested, to clarify user needs, while the actual information system is under construction. The *breadboard prototype* is used mainly by development staff to ascertain the feasibility of certain technical aspects of the system. A *pilot system prototype* is a type of prototype which constitutes the core of an application system. After one or more iterations of evolutionary prototyping a pilot system prototype reaches enough "sophistication" to become the final system.

Budde *et al*. (1992) also delineate the construction techniques employed in prototyping as either horizontal or vertical prototyping. Prototypes can be developed technically in different ways. Software construction can be seen as the design and implementation of a number of different software "layers" from the user-interface to database query language or the operating system. Horizontal prototyping involves the building of specific layers such as the user-interface alone. Horizontal prototyping is also called mock-up prototyping. Vertical prototyping involves the building of selected parts of the target system through all layers. Vertical (or functional) prototyping is used when particular aspects of functionality need to be demonstrated(Budde *et al.*, 1992, Lichter *et al.*, 1994).

3 PROTOTYPING PROCESS MODELS

The essential characteristic features of prototyping are invariant although many researchers have outlined the prototyping process. (Nauman and Jenkins, 1982, Boar, 1986, Stephens and Bates, 1986) In this section, a general outline of the characteristics of the prototyping process is discussed and the major features identified by researchers are considered.

Floyd (1984) describes the prototyping process as consisting of functional selection, construction, evaluation and further use. Those functions that are to be prototyped are selected and a prototype is constructed. This prototype is evaluated and the prototype is further used for outlining specification or as a part of the new system. Naumann and Jenkins (1982) characterise prototyping as a four step, iterative procedure involving users and developers:

1) user's basic needs are identified;

2) a working prototype is developed;

3) the working prototype is then implemented and used;

4) the prototyping system is revised and enhanced.

This process undergoes several iterations and steps three and four are repeated until the user accepts the system. Boar (1986) describes the prototyping life cycle as identifying basic needs, developing a working model, demonstrating the model to all people affected by it in order to solicit requirements, and implementing revisions or enhancements with successive iterations until all participants are satisfied with the functionality. Stephen and Bates (1990) note the difficulty of formulating a universal definition for prototyping. They identify common prototyping characteristics:

1) a high degree of user evaluation which substantially affects requirements, specifications or design, and

2) initiates a learning process for users and developers of the system.

We identify the essential features of the prototyping process with the following characterisation. The prototyping process consists essentially of several iterative cycles. The initial prototype - an executable software model is constructed based on either an initial selection of functions or on user's needs that have been identified. A demonstration or actual usage of the prototype allows review by users. User's suggestions, criticisms and enhancements result in revision of the prototype. This cycle is continued for each revised version. The prototyping process is completed when the desired objective is attained (See Figure 2).

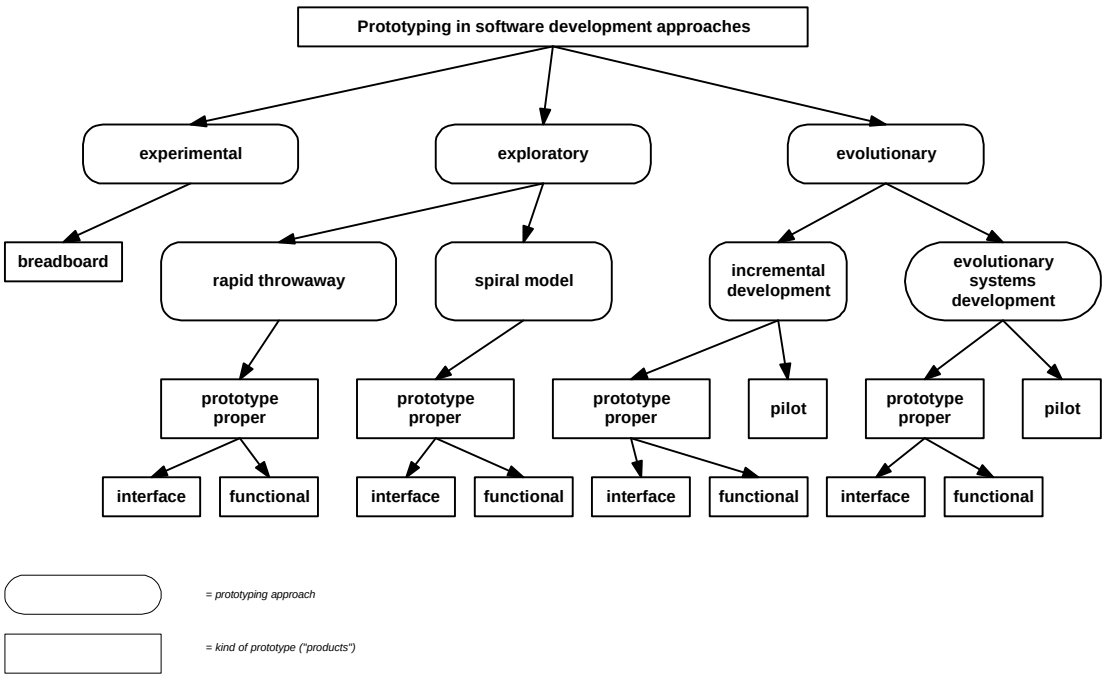

Figure 2: Prototyping in Software Development Approaches

4 SOFTWARE PROTOTYPE DEVELOPMENT APPROACHES

Floyd (1984) categorises software development approaches that employ prototyping as being based on one of three goals - exploration, experiment and evolution. An

exploratory prototyping approach is mainly undertaken to elicit or clarify user requirements. Exploratory prototyping helps developers gain insight into the users work tasks and problems, and helps to crystallise hazy user perceptions and needs into the requirements for an initial system. Experimental prototyping provides users with alternative possibilities, helps assess the feasibility of the future system in terms of performance and other technical aspects, when requirements are known. Evolutionary prototyping is an approach that allows flexibility in the software development process so that it can adapt to changing organisational contexts. Here, software development process is not seen as an isolated self-contained project but as something that continuously evolves.

4.1 Exploratory Approach

Software development approaches often need to explore requirements or certain details before or during development in a traditional fashion. There are two important approaches that adopt this way - rapid throwaway prototyping and the spiral model (Boehm, 1988), which are discussed below. Rapid throwaway prototyping explores completeness of specifications and the spiral model attempts management of risk using prototypes in the software development process.

Rapid throwaway prototyping helps us to ascertain a complete set of user requirements. During the requirements stage of a conventional software development life cycle model a “quick and dirty” (hence rapid) partial implementation of the system is built (Gomaa, 1983). Poorly understood requirements are usually implemented first[15]. While the prototype is constructed quality factors like efficiency, maintainability, portability, documentation and completeness are not considered. (Keus, 1982) Potential users provide feedback to the developers by reviewing the

prototype which is used to refine the software requirements specification. The developers then proceed to design and implement the system once there is agreement between the users and developers on the requirements. The prototype is normally discarded though sometimes the code may be reused (Davis et al., 1988). Rapid prototyping is also known as throw-it-away prototyping (Ince and Hekmatpour, 1987, Tate, 1990) A throwaway prototype may be embedded within any phase of a waterfall development approach but predominantly it is used within the requirements elicitation/analysis phases or the design/construction phases (Verner and Cerpa, 1996).

Boehm's (Boehm, 1988) *spiral model* further extends the approach taken by Gomaa (1983) and employs prototypes in various phases of the waterfall model in his spiral model. The software process begins with a concept document and progresses in cycles to coding of individual programs in the spiral model. Each cycle involves the same sequence of steps; of determining objectives, alternatives and constraints that resolve risks by the use of prototypes, simulations etc. and proceeding to the next cycle based on how the alternatives were resolved. The spiral model embeds prototypes for risk resolution in the process of software development. Though Boehm (1988) calls his prototypes 'evolutionary' he uses the term in the sense of a 'prototype proper' rather than as 'pilot prototype'.

4.2 Experimental Approach

In this approach, prototypes are built so that the feasibility of proposed solutions to problems can be examined by means of experimental use. The prototype could be a partial functional simulation demonstrating certain aspects of the system, a full

functional simulation which makes available all the functions of the proposed system, a mock-up interface or a skeletal program showing the structure of the system (Tate, 1990, Mayhew and Dearnley, 1987).

4.3 Evolutionary Software Development Approach

Information systems are embedded in an organisational context. As the organisation surrounding the system evolves, new requirements emerge. The implementation of a new system can also transform the usage context and therefore new requirements arise (Floyd, 1984) The evolutionary prototyping approach results in the gradual development of systems and allows for, or adapts to, changes that take place within an organisation as a result of either the operationalisation of the system, or externally induced changes to the organisation (Ince and Hekmatpour, 1987) In contexts of high uncertainty (and unlike contexts with high certainty where waterfall models are employed), the evolutionary prototyping approach dynamically responds to changes in user needs and accommodates subsequent unpredictable changes in requirements, as the development process progresses (Pape and Thoresen, 1992).

Floyd (1984) identifies incremental development and evolutionary system development as two distinct approaches that employ prototypes. During incremental development the system evolves gradually in partial increments against the backdrop of an overall long range development strategy. The fundamental difference between evolutionary development and incremental development is in system design; while software design evolves in evolutionary development, there are no changes to the design during incremental development (Ince and Hekmatpour, 1987). Whilst incremental development assumes that most user needs are known at the beginning, in evolutionary system development, the prototype is

built in an area where overall the requirements are not well understood (Davis et al., 1988). Initial development, however, starts in an area where the requirements are known. The iterative enhancement approach and the incremental development approach are closely related approaches to prototyping though here we clarify the differences.

Basili and Turner (1975) define the *iterative enhancement* method as

> *"a practical means of using a top-down, step-wise refinement approach to software development. A practical approach to the problem is to start with a simple initial subset of the problem and iteratively enhance existing versions until a full system is implemented. At each step of the process, not only extensions but also design modifications can be made. In fact, each step can make use of stepwise refinement in a more effective way as the system becomes better understood through the iterative process."*

The iterative enhancement technique presumes that there is an initial "project list" which is achieved in stages, beginning with the implementation of a simple subset and undergoing successive refinements to the software as the development goes on until a "final implementation is developed that meets the project specifications" (Basili and Turner, 1985). Whilst the software design can undergo modification, we should note that the project specifications themselves do not change 'stepwise'.

Incremental development is a method in which a partial system is constructed from an overall system design. Functionality and increased performance are slowly added to the partial system. A number of partial and increasingly complete versions

of the system are produced. Each version adds a part of the original design (Ince and Hekmatpour, 1987). The system is deliberately built in a manner that facilitates the incorporation of additional requirements.

Evolutionary system development, is adopted in a context of a dynamic and changing environment. This type of development necessities a sequence of cycles of (re-) design, (re-) implementation and (re-) evaluation without any attempt to capture a complete set of requirements in advance (Floyd, 1984). A prototype of partially known requirements is implemented first. When users use the prototype a fuller understanding of the requirements is gained. New requirements are then implemented. The final system evolves in a continuous fashion (Davis *et al*., 1988). Each successive prototype explores new user needs, and refines functionality that has already been implemented. While the sequence of prototypes are perhaps converging approximations towards a moving target they may never hit it, as the requirements are subject to continual change (Tate, 1990). Therefore, the system is always in continuous evolution and there is no such distinct phase as maintenance. (Floyd, 1984).

5 CONCLUSION

There is much confusion in the prototyping literature regarding the usage of various terms relating to prototyping. We have provided a framework within which the terms relating to prototyping can be properly located. The framework integrates prototypes, the prototyping process and approaches to software development that use prototyping. Software development approaches use prototyping and prototypes for a variety of different purposes. Prototypes are essentially the products of the prototyping process and there are different kinds of prototypes dependent on how

they are used or constructed. Essentially prototypes are used for three main purposes in software development - exploration, experimentation and evolution. Exploratory approaches use prototypes for finding requirements early and to make complete users incomplete knowledge of their requirements. Rapid throwaway prototyping and Boehm's spiral model fall within this classification. Experimental approaches use prototypes to investigate certain kinds of feasibilities or possibilities within the process of development. The feasibility investigated may be technical, increasing efficiency, etc. The evolutionary approaches use the philosophy of the prototyping process itself and employ it as the methodology for the software development process as a whole. Incremental development and evolutionary system development can both be viewed as types of evolutionary software development approaches.

Whilst the distinctions made here are necessarily for classificatory purposes, in practice it has been found that one or more kinds of prototypes may be employed within a particular prototyping development approach as demonstrated by the case studies of Lichter *et al.* (1994) and there are relationships between some development approaches and particular kinds of prototypes. The evolutionary system development approach produces pilot prototypes and the experimental approach at times uses breadboard prototypes. Finally, the boundaries between experimentation and exploration are blurred and evolution may initially include both experimentation and exploration (Floyd, 1984).

6 REFERENCES

1. Royce, W.W. (1970) Managing the Development of Large Software Systems: Concepts and Techniques, in *Proceedings WESCON*, August.
2. Basili, V.R. and Turner, A.J. (1975) Iterative Enhancement: A Practical Technique for Software Development, *IEEE Transactions on Software Engineering* **1**, 4, 390-396.
3. Gomaa, H. (1983) The Impact of Rapid Prototyping on Specifying User Requirements, *ACM SIGSOFT Software Engineering Notes* **8**, 2, 17-28.
4. Floyd, C. (1984) A Systematic Look at Prototyping, in: Budde, R., Kuhlenkamp, K., Mathiassen, L. and Zullighoven, H. (Eds.) *Approaches to Prototyping*, Springer-Verlag: Heidelberg, 1-17.
5. Tate, G. (1990) Prototyping: Helping to Build the Right Software, *Information and Software Technology,* **42**, 4, 237-244.
6. Kaushaar, J.M. and Shirland, L.E. (1985) A Prototyping Method for Applications Development End Users and Information Systems Specialists, *MIS Quarterly*, **9**, 4, 189 -197.
7. Boehm, B.W., Gray, T.E. and Seewaldt, T. (1984) Prototyping Versus Specifying: A Multiproject Experiment, *IEEE Transactions on Software Engineering*, **SE-10**, 4, 290 -402.
8. Gordon, V.S. and Bieman, J.M. (1994) Rapid Prototyping: Lessons Learned, *IEEE Software*, **12**, 1, 85-95.
9. Alavi, M. (1985) An Assessment of the Prototyping Approach to Systems, *Communications of the ACM*, **27**, 6, 556-564.

10. Tate, G. and Verner, J. (1990) Case Study of Risk Management, Incremental Development and Evolutionary Prototyping, *Information and Software Technology* **42**, 4, 207-214.

11. Nauman, J.D. and Jenkins, M. (1982) Prototyping: The New Paradigm for Systems Development, *MIS Quarterly*, **6**, 3, 29-44.

12. Budde, R., Kautz, K., Kuhlenkamp K., Zullighoven, H. (1992) What is Prototyping?" *Information Technology & People*, **6**, 2-4, 89-95.

13. Boar, B. (1986) Application Prototyping: A Life Cycle Perspective, *Journal of Systems Management,* **47**, 2, 25-41.

14 Stephens, M.A. and Bates, P.E. (1990) Requirements Engineering by Prototyping: Experiences in Development of Estimating System, *Information and Software Technology,* **42**, 4, 253-257.

15 Davis, A., Bersoff, E.H. and Comer, E.R. (1988) A Strategy for Comparing Alternative Software Development Life Cycle Models, *IEEE Transactions on Software Engineering,* **14**, 1453-1461.

16 Keus, H.E. (1982) Prototyping: A More Reasonable Approach to System Development, *ACM SIGSOFT Software Engineering Notes*, **7**, 5.

17 Ince, D.C. and Hekmatpour, S. (1987) Software Prototyping: Progress and Prospects, *Information and Software Technology*, **29**, 1, 8-14.

18 Palvia, P. and Nosek, J.T. (1990) An Empirical Evaluation of System Development Methodologies, *Information Resources Management Journal*, **3**, 23-32.

19 Verner, J., Tate, G. and Cerpa, N. (1995) Prototyping: Some New Results, *Information and Software Technology*, **38**, 12, 743-755.

20 Boehm, B.W. (1988) A Spiral Model of Software Development and Enhancement, *IEEE Computer*, 26-47, May.

21 Mayhew, P.J. and Dearnley, P.A. (1987) An Alternative Prototyping Classification, *The Computer Journal*, **40**, 6, 481-484..

22 Pape, T.C. and Thoresen, K. (1992) Evolutionary Prototyping in a Change Perspective, *Information Technology and People,* **6**, 2-4.

24 Lichter, H., Schneider-Hufschmidt, M. and Zullighoven, H. (1994) Prototyping in Industrial Software Projects: Bridging the Gap between Theory and Practice, *IEEE Transactions on Software Engineering,* **20**, 11, 825-842.