SNS COLLEGE OF ENGINEERING

Kurumbapalayam(Po), Coimbatore – 641 107
Accredited by NAAC-UGC with 'A' Grade
Approved by AICTE, Recognized by UGC & Affiliated to Anna University, Chennai

Department of Information Technology

Object Oriented Software Engineering

Evolutionary Process Models

Prepared By
R.Vaishnavi.,AP/IT
SNSCE.

Evolutionary Process Models

- Business and product requirements often change as development proceeds, making a straight line path to an end product unrealistic; In such case, the iterative approach needs to be adopted. Evolutionary process model is also called as iterative process model
- Evolutionary models are iterative. They are characterized in a manner that enables you to develop increasingly more complete versions of the software.

Prototyping

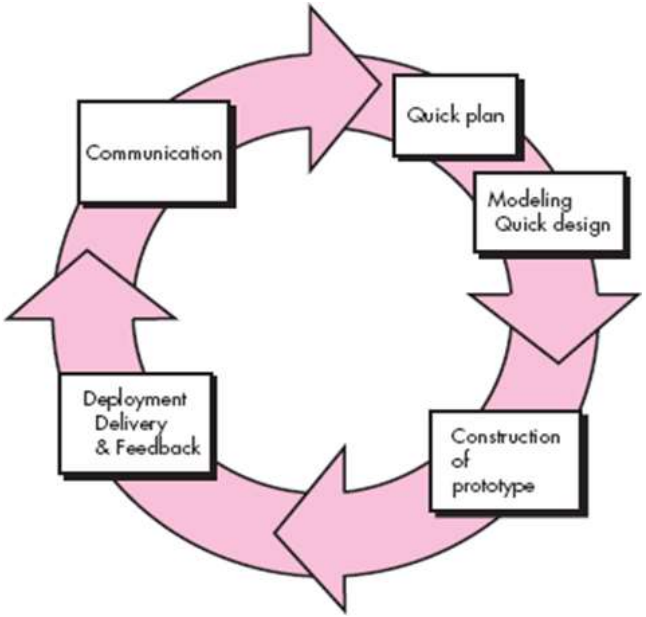
- Software prototyping, refers to the activity of creating prototypes of software applications, i.e., incomplete versions of the software program being developed.
- It is an activity that can occur in software development and is comparable to prototyping as known from other fields, such as mechanical engineering or manufacturing.

When we can choose Prototype:

- A customer defines a set of general objectives for software, but does not identify detailed requirements for functions and features.
- The developer may be unsure of the efficiency of an algorithm, the adaptability of an operating system.
- When requirements are fuzzy

- Although prototyping can be used as a stand-alone process model, it is more commonly used as a technique that can be implemented within the context of any one of the process models.
- The prototyping assists you and other stakeholders to better understand what is to be built when requirements are fuzzy.

- The prototyping paradigm begins with communication. You meet with other stakeholders to define the overall objectives for the software, identify whatever requirements are known, and outline areas where further definition is mandatory.
- A prototyping iteration is planned quickly, and modeling (in the form of a “quick design”) occurs. A quick design focuses on a representation of those aspects of the software that will be visible to end users (e.g., human interface layout or output display formats).



- The quick design leads to the construction of a prototype. The prototype is deployed and evaluated by stakeholders, who provide feedback that is used to further refine requirements.

- Iteration occurs as the prototype is tuned to satisfy the needs of various stakeholders, while at the same time enabling you to better understand what needs to be done.

- Ideally, the prototype serves as a mechanism for identifying software requirements.
- If a working prototype is to be built, you can make use of existing program fragments or apply tools (e.g., report generators and window managers) that enable working programs to be generated quickly.
- In most projects, the first system built is barely usable. It may be too slow, too big, awkward in use or all three.
- There is no alternative but to start again, smarting but smarter, and build a redesigned version in which these problems are solved.

- The prototype can serve as “the first system.” The one that Brooks recommends you throw away. But this may be an idealized view. Although some prototypes are built as “throwaways,” others are evolutionary in the sense that the prototype slowly evolves into the actual system.
- Both stakeholders and software engineers like the prototyping paradigm.
- Users get a feel for the actual system, and developers get to build something immediately.

Advantages:

- Users are actively involved in the development
- Since in this methodology a working model of the system is provided, the users get a better understanding of the system being developed.
- Errors can be detected much earlier.
- Quicker user feedback is available leading to better solutions.
- Missing functionality can be identified easily.
- Confusing or difficult functions can be identified Requirements validation, Quick implementation of, incomplete, but functional, application.

Disadvantages:

- Stakeholders see what appears to be a working version of the software, unaware that the prototype is held together haphazardly, unaware that in the rush to get it working you haven't considered overall software quality or long-term maintainability.
- Software engineer make implementation compromises in order to get a prototype working quickly.
- An inappropriate operating system or programming language may be used simply because it is available and known; an inefficient algorithm may be implemented simply to demonstrate capability.

Usage of prototyping:

- Although problems can occur, prototyping can be an effective paradigm for software Engineering.
- The key is to define the rules of the game at the beginning; that is, all stakeholders should agree that the prototype is built to serve as a mechanism for defining requirements.
- It is then discarded (at least in part), and the actual software is engineered with an eye toward quality.