

SOFTWARE ENGINEERING

Software Life Cycle Models

The goal of Software Engineering is to provide models and processes that lead to the production of well-documented maintainable software in a manner that is predictable.



SOFTWARE ENGINEERING

Software Life Cycle Models

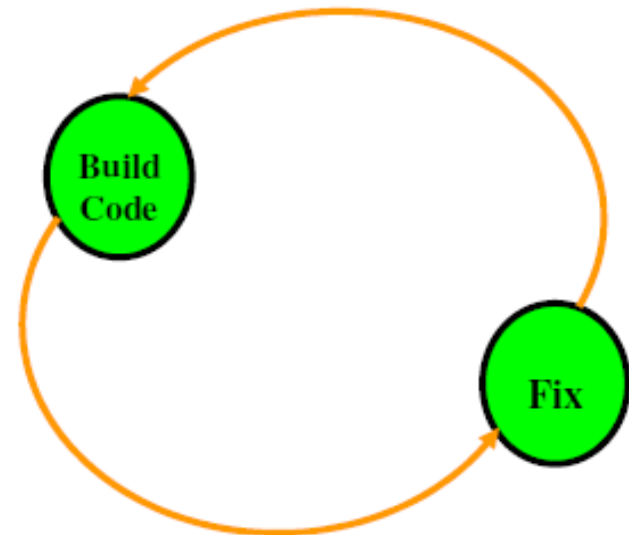
“The period of time that starts when a software product is conceived and ends when the product is no longer available for use. The software life cycle typically includes a requirement phase, design phase, implementation phase, test phase, installation and check out phase, operation and maintenance phase, and sometimes retirement phase”.



SOFTWARE ENGINEERING

Build & Fix Model

- ❖ Product is constructed without specifications or any attempt at design
- ❖ Adhoc approach and not well defined
- ❖ Simple two phase model



SOFTWARE ENGINEERING

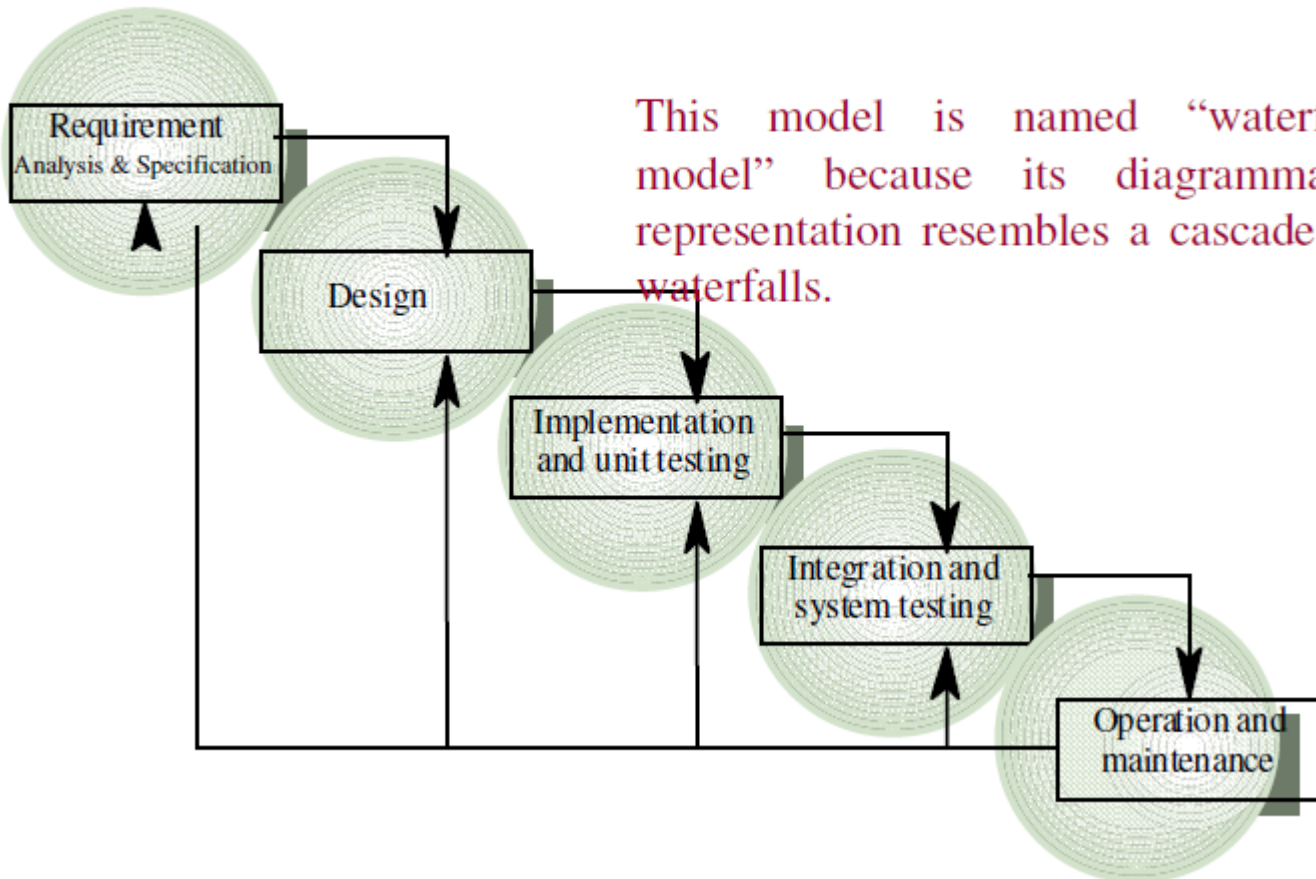
Build & Fix Model

- ❖ Suitable for small programming exercises of 100 or 200 lines
- ❖ Unsatisfactory for software for any reasonable size
- ❖ Code soon becomes unfixable & unenhanceable
- ❖ No room for structured design
- ❖ Maintenance is practically not possible



SOFTWARE ENGINEERING

Waterfall Model



SOFTWARE ENGINEERING

Waterfall Model

This model is easy to understand and reinforces the notion of “define before design” and “design before code”.

The model expects complete & accurate requirements early in the process, which is unrealistic



SOFTWARE ENGINEERING

Waterfall Model

Problems of waterfall model

- i. It is difficult to define all requirements at the beginning of a project
- ii. This model is not suitable for accommodating any change
- iii. A working version of the system is not seen until late in the project's life
- iv. It does not scale up well to large projects.
- v. Real projects are rarely sequential.



SOFTWARE ENGINEERING

Incremental Process Models

They are effective in the situations where requirements are defined precisely and there is no confusion about the functionality of the final product.

After every cycle a useable product is given to the customer.

Popular particularly when we have to quickly deliver a limited functionality system.



SOFTWARE ENGINEERING

Iterative Enhancement Model

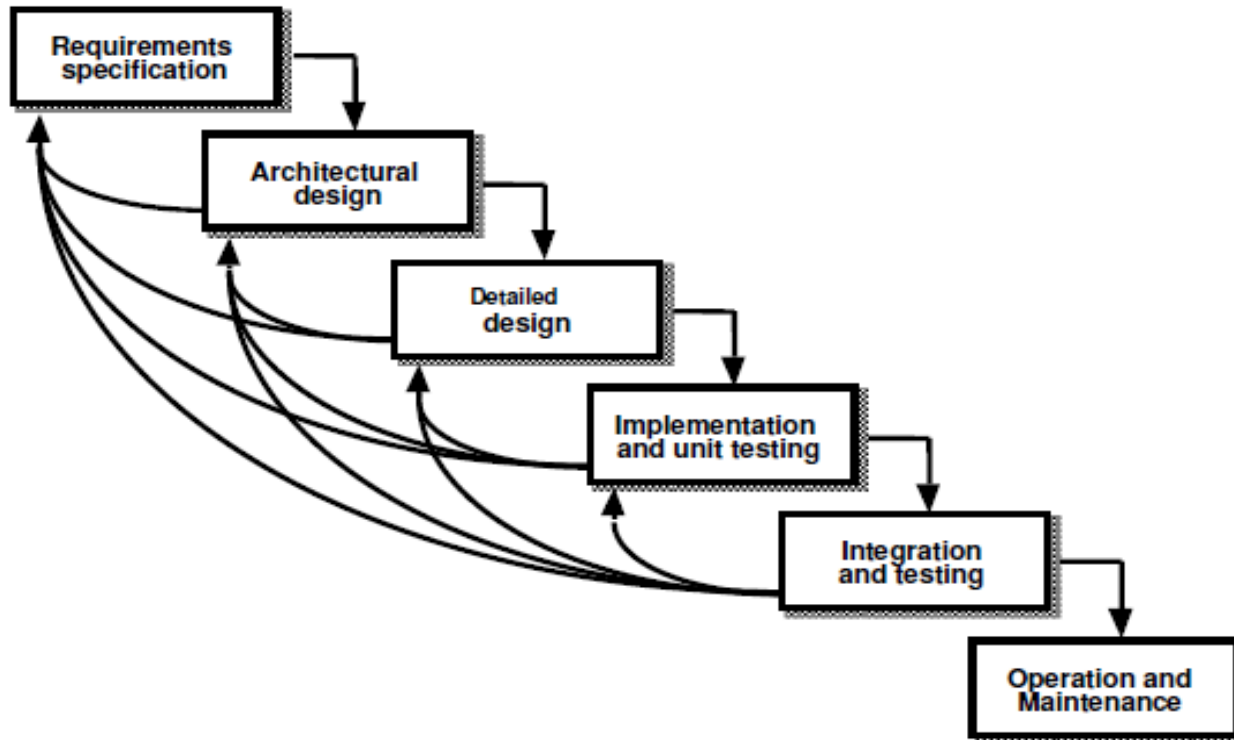
This model has the same phases as the waterfall model, but with fewer restrictions. Generally the phases occur in the same order as in the waterfall model, but they may be conducted in several cycles. Useable product is released at the end of the each cycle, with each release providing additional functionality.

- ✓ Customers and developers specify as many requirements as possible and prepare a SRS document.
- ✓ Developers and customers then prioritize these requirements
- ✓ Developers implement the specified requirements in one or more cycles of design, implementation and test based on the defined priorities.



SOFTWARE ENGINEERING

Iterative Enhancement Model



SOFTWARE ENGINEERING

Prototyping Model

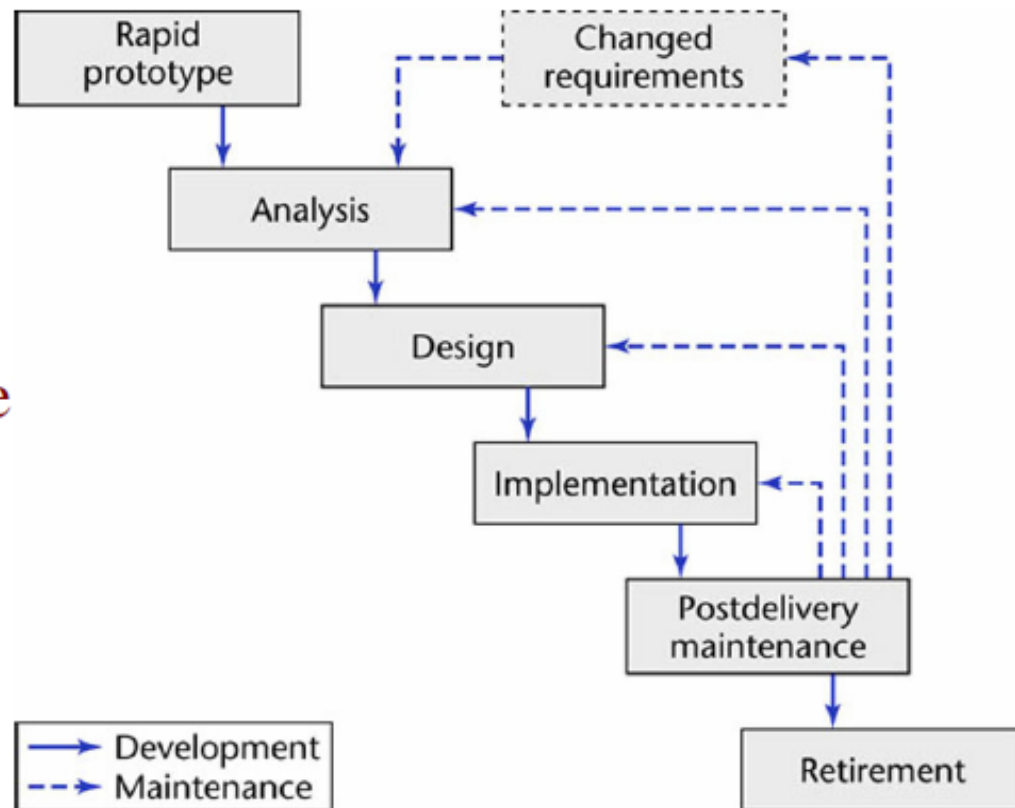
- The prototype may be a usable program but is not suitable as the final software product.
- The code for the prototype is thrown away. However experience gathered helps in developing the actual system.
- The development of a prototype might involve extra cost, but overall cost might turnout to be lower than that of an equivalent system developed using the waterfall model.



SOFTWARE ENGINEERING

Prototyping Model

- Linear mode
- “Rapid”



SOFTWARE ENGINEERING

Evolutionary Process Models

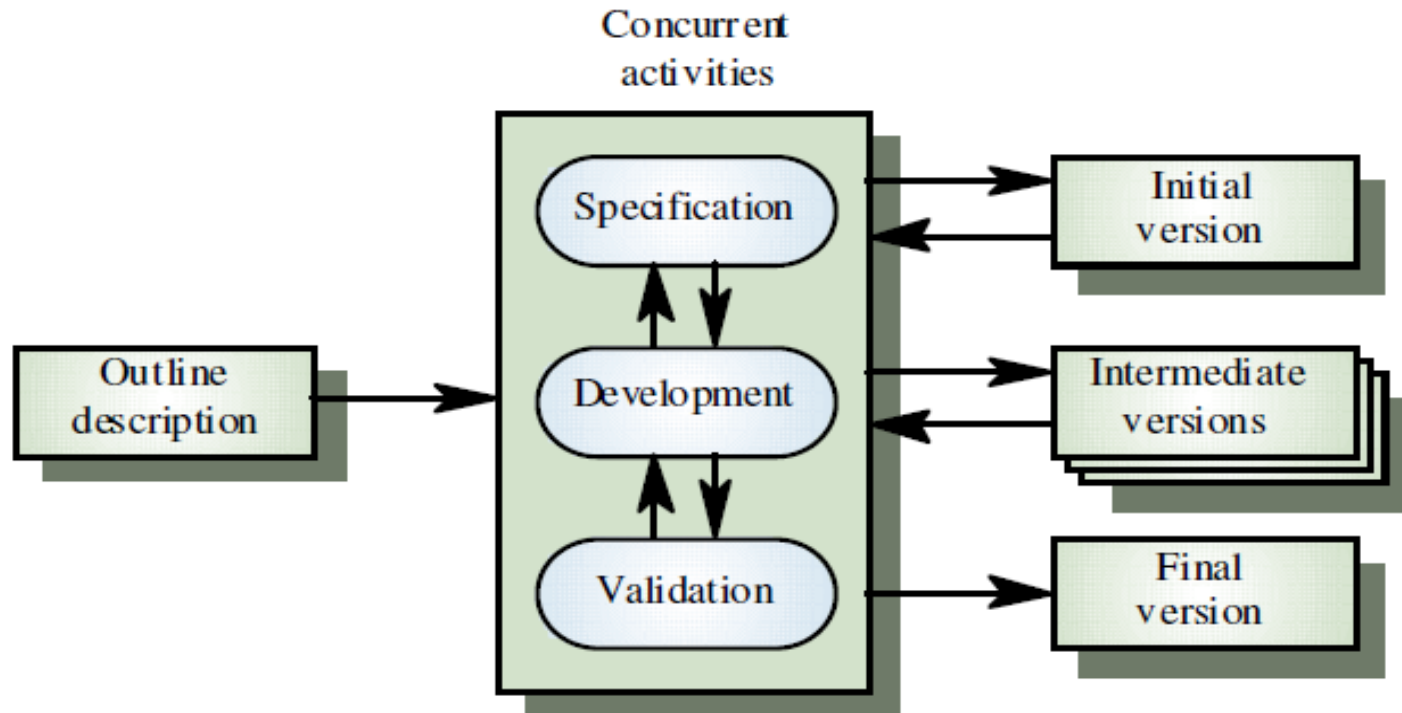
Evolutionary process model resembles iterative enhancement model. The same phases as defined for the waterfall model occur here in a cyclical fashion. This model differs from iterative enhancement model in the sense that this does not require a useable product at the end of each cycle. In evolutionary development, requirements are implemented by category rather than by priority.

This model is useful for projects using new technology that is not well understood. This is also used for complex projects where all functionality must be delivered at one time, but the requirements are unstable or not well understood at the beginning.



SOFTWARE ENGINEERING

Evolutionary Process Model



SOFTWARE ENGINEERING

SPIRAL MODEL

Important software projects have failed because project risks were neglected & nobody was prepared when something unforeseen happened.

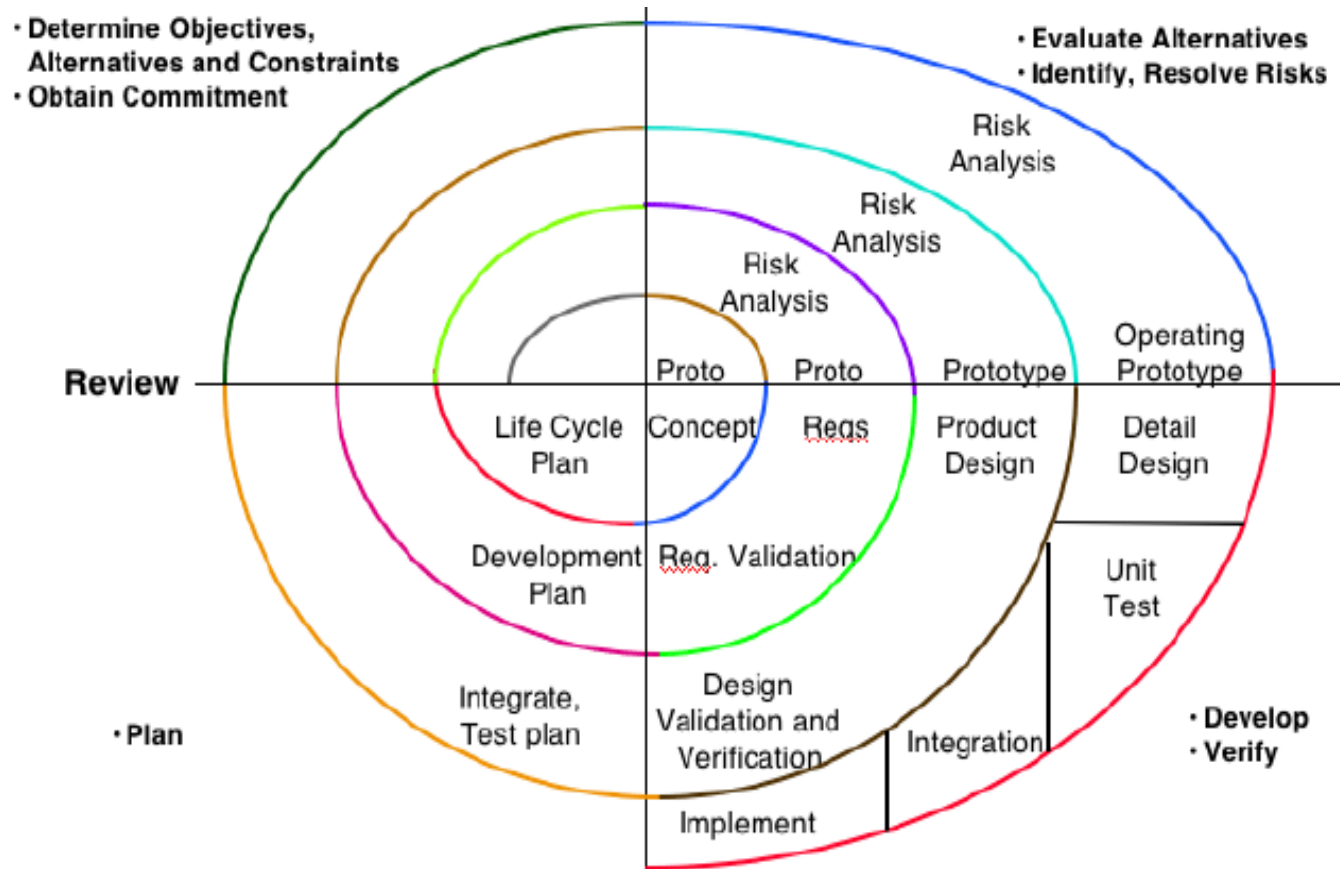
Barry Boehm recognized this and tried to incorporate the “project risk” factor into a life cycle model.

The result is the spiral model, which was presented in 1986.



SOFTWARE ENGINEERING

SPIRAL MODEL



SOFTWARE ENGINEERING

SPIRAL MODEL

The radial dimension of the model represents the cumulative costs. Each path around the spiral is indicative of increased costs. The angular dimension represents the progress made in completing each cycle. Each loop of the spiral from X-axis clockwise through 360° represents one phase. One phase is split roughly into four sectors of major activities.

- **Planning:** Determination of objectives, alternatives & constraints.
- **Risk Analysis:** Analyze alternatives and attempts to identify and resolve the risks involved.
- **Development:** Product development and testing product.
- **Assessment:** Customer evaluation



SOFTWARE ENGINEERING

SPIRAL MODEL

- An important feature of the spiral model is that each phase is completed with a review by the people concerned with the project (designers and programmers)
- The advantage of this model is the wide range of options to accommodate the good features of other life cycle models.
- It becomes equivalent to another life cycle model in appropriate situations.

The spiral model has some difficulties that need to be resolved before it can be a universally applied life cycle model. These difficulties include lack of explicit process guidance in determining objectives, constraints, alternatives; relying on risk assessment expertise; and provides more flexibility than required for many applications.



SOFTWARE ENGINEERING

Selection of a Life Cycle Model

Selection of a model is based on:

- a) Requirements
- b) Development team
- c) Users
- d) Project type and associated risk



SOFTWARE ENGINEERING

Based On Characteristics Of Requirements

Requirements	Waterfall	Prototype	Iterative enhancement	Evolutionary development	Spiral	RAD
Are requirements easily understandable and defined?	Yes	No	No	No	No	Yes
Do we change requirements quite often?	No	Yes	No	No	Yes	No
Can we define requirements early in the cycle?	Yes	No	Yes	Yes	No	Yes
Requirements are indicating a complex system to be built	No	Yes	Yes	Yes	Yes	No



SOFTWARE ENGINEERING

Based On User's Participation

Involvement of Users	Waterfall	Prototype	Iterative enhancement	Evolutionary development	Spiral	RAD
User involvement in all phases	No	Yes	No	No	No	Yes
Limited user participation	Yes	No	Yes	Yes	Yes	No
User have no previous experience of participation in similar projects	No	Yes	Yes	Yes	Yes	No
Users are experts of problem domain	No	Yes	Yes	Yes	No	Yes



SOFTWARE ENGINEERING

Based On Type Of Project With Associated Risk

Project type and risk	Waterfall	Prototype	Iterative enhancement	Evolutionary development	Spiral	RAD
Project is the enhancement of the existing system	No	No	Yes	Yes	No	Yes
Funding is stable for the project	Yes	Yes	No	No	No	Yes
High reliability requirements	No	No	Yes	Yes	Yes	No
Tight project schedule	No	Yes	Yes	Yes	Yes	Yes
Use of reusable components	No	Yes	No	No	Yes	Yes
Are resources (time, money, people etc.) scarce?	No	Yes	No	No	Yes	No

