Evolutionary Process Models

Evolutionary process models are iterative type models. Using these models, the developer can develop increasingly more complete versions of the software.

When are we going to use evolutionary process models?

- To build complex versions of the software.
- Requirements are not clear.

There are two common evolutionary process models.
1. Prototyping Model
2. Spiral Model

Prototyping Model

Prototyping is best suited for the software projects with the following cases.

- Customer state the problem with a set of general objectives, but does not identify detailed requirements for functions and features of the software.
- Developer unsure of the efficiency of the algorithmic, form of human machine interaction and development environment.

Prototyping can be used together with other models for elicitation of requirements. Prototyping paradigm helps us to better understand the requirements.

The prototyping paradigm begins with communication, developer meet with other stakeholders to define the overall objectives for the software, identify whatever requirements are known, and outline areas where further definition is required.

After completion of communication activity, quick plan, modelling and quick design takes place. A quick design focuses on a representation of the soft-ware that will be visible to end users, human interface layout or output display format. The quick design leads to the construction of a prototype. The prototype is deployed and evaluated by stakeholders; they provide feedback that is used to further refine requirements.

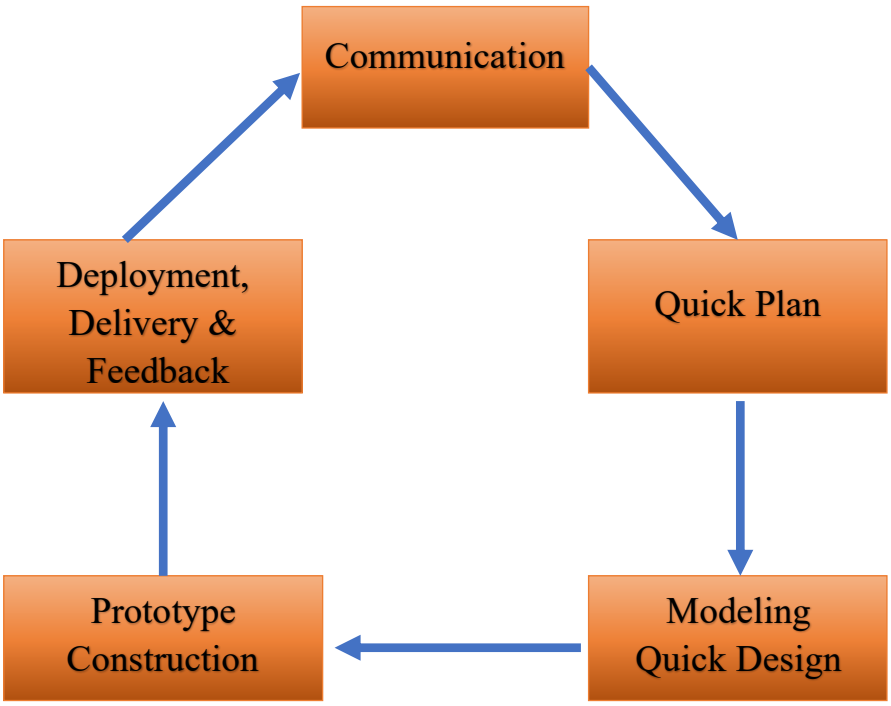

Figure: Prototyping Model

The prototype can serve as "the first system." Some prototypes are "Throw Away" while others also evolve and become part of 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:

- Since this methodology provides quick working version of the software, the users get a better understanding of the system being developed.
- Errors can be detected much easier.
- Quicker user feedback is available leading to better solution.
- Missing functionality can be easily identified.

Disadvantages:

- Implementation compromises often make in order to get a prototype working quickly.
- There is no guarantee of software quality as prototype rush to get working.

Spiral Model

Originally proposed by Barry Boehm [Boe88], the spiral model is an evolutionary software process model that couples the iterative nature of prototyping with the controlled and systematic aspects of the waterfall model.

Using the spiral model, software is developed in a series of evolutionary releases. During early iterations, the release might be a model or prototype. During later iterations, increasingly more complete versions of the software are produced.

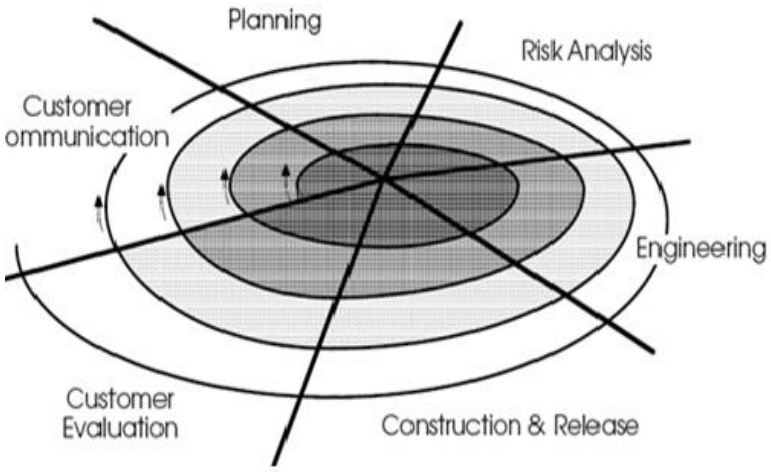

Figure: Spiral Model

A spiral model is divided into a set of framework activities defined by the software engineering team. The initial activity is shown from centre and developed in clockwise direction.

The first circuit around the spiral might result in the development of a product specification; subsequent passes around the spiral might be used to develop a prootype and then progressively more sophisticated versions of the software.

Advantages:

1. Unlike other process models that end when software is delivered, the spiral model can be adapted to apply throughout the life of the computer software.
2. The developer and customer better understand and react to risks at each evolutionary level.
3. Schedule and cost are more realistic.
4. Changes can be accommodated in the later stages of development.

Disadvantages:

1. If major risk is not discovered in early iteration of spiral, it may become a major risk in the later stages.
2. Each iteration around the spiral leads to more completed version of software. But it's difficult to convince to the customer that the model is controllable.
3. Cost of this approach is usually high.
4. Not suitable for low risk management.
5. Rules and protocols must be followed very strictly to implement the approach.

Text Books

1. Roger Pressman S., "Software Engineering: A Practitioner's Approach", 7th Edition, McGraw Hill, 2010.
2. Sommerville, "Software Engineering", Eighth Edition, Pearson Education, 2007

Web Links

1. https://ocw.mit.edu/courses/aeronautics-and-astronautics/16-355j-software-engineering-concepts-fall-2005/lecture-notes/cnotes2.pdf
2. https://cdn.shopify.com/s/files/1/0457/4009/7694/files/software_engineering_pdf_pressman_7th_edition.pdf
3. https://images.app.goo.gl/c9TJtajtyzgrLo1i6