Software Engineering

Week 3

Muhammad Daniyal Liaquat

Software processes and Structure

Software process

- A software process is a set of related activities that leads to the production of a software product.
- There are many different software processes but all must include four activities that are fundamental to software engineering:

  1. Software specification The functionality of the software and constraints on its operation must be defined.
  2. Software design and implementation The software to meet the specification must be produced.
  3. Software validation The software must be validated to ensure that it does what the customer wants.
  4. Software evolution The software must evolve to meet changing customer needs

Process descriptions

When we describe and discuss processes, we usually talk about the activities in these processes such as specifying a data model, designing a user interface, etc., and the ordering of these activities.

process descriptions may also include:

1. Products, which are the outcomes of a process activity. For example, the out come of the activity of architectural design may be a model of the software architecture.
2. Roles, which reflect the responsibilities of the people involved in the process. Examples of roles are project manager, configuration manager, programmer, etc.
3. Pre- and post-conditions, which are statements that are true before and after a process activity has been enacted or a product produced.

Example, before architectural design begins, a pre-condition may be that all requirements have been approved by the customer; after this activity is finished, a post-condition might be that the UML models describing the architecture have been reviewed

Plan-driven or agile processes.

- Plan-driven processes are processes where all of the process activities are planned in advance and progress is measured against this plan.
- Agile processes, planning is incremental and it is easier to change the process to reflect changing customer requirements.
- In Practice most practical processes include elements of both plan driven and agile approaches.
- There is no right or wrong processes.

Software Process Models

To solve actual problems in an industry setting, a software engineer or a team of engineers must incorporate a development **strategy** that encompasses the process, methods, and tools layers and the generic phases as described in earlier lectures

A Generic View of SE:

1. What is the problem to solved?
2. What are the characteristics of the entity that is used to solve the problem?
3. How will the entity be realized?
4. How will the entity be constructed?
5. What approach will be used to uncover errors that were made in the design and the construction of entity?
6. How will the entity be supported over the long term, when users of the entity request corrections, adaptations and enhancements?

6

Software Process Models

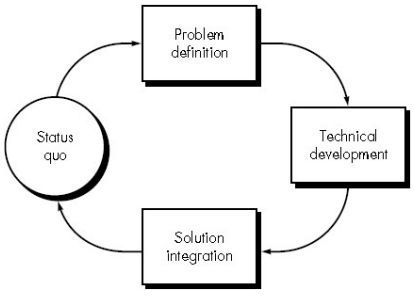
This **strategy** is often referred to as a *process model* or a *software engineering paradigm.*

A process model for software engineering is chosen based on the nature of the project and application, the methods and tools to be used, and the controls and deliverables that are required.

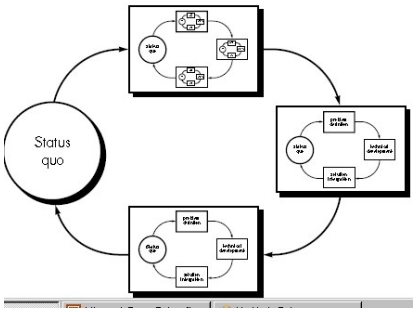
In an intriguing paper on the nature of the software process, Raccoon [RAC95] uses fractals as the basis for a discussion of the true nature of the software process.

7

Software Process Models



(a) The phases of a problem solving loop [RAC95]



(b) The phases within phases of the problem solving loop [RAC95]

8

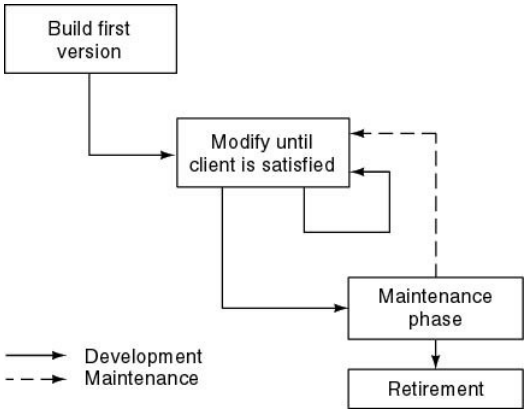
Software Process Models

There are a number of Software Development Lifecycle Models, each having its strengths and weaknesses and suitable in different situations and project types. The list of models includes the following:

- Build-and-fix model
- Waterfall model
- Incremental model
- Extreme programming
- Synchronize-and-stabilize model
- Spiral model
- Object-oriented life-cycle models

9

Build and Fix Model



It is unfortunate that many products are developed using what is known as the build-and-fix model.

10

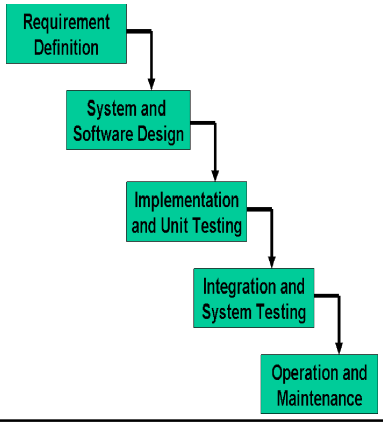
Build and Fix Model

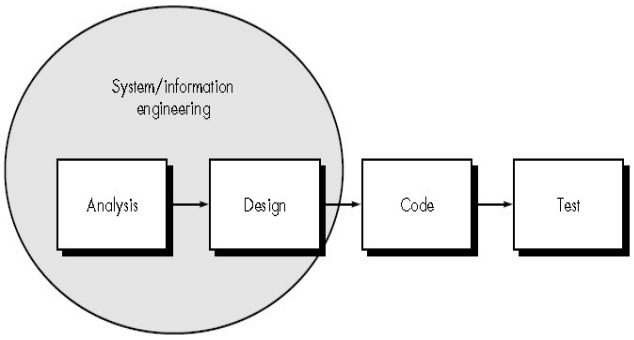
- In this model the product is constructed without specification or any attempt at design.
- The developers simply build a product that is reworked as many times as necessary to satisfy the client.
- This model may work for small projects but is totally unsatisfactory for products of any reasonable size.
- The cost of build-and fix is actually far greater than the cost of properly specified and carefully designed product.
- Maintenance of the product can be extremely in the absence of any documentation.

11

Linear Sequential Model/ waterfall model

Also called as classic life cycle or the waterfall model, the linear sequential model suggests a systematic, sequential approach to software development that begins at the system level and progresses through **analysis, design, coding, testing, and support**.





12

Linear Sequential Model/ waterfall model

The principal stages of the model map directly onto fundamental development activities.

It suggests a systematic, sequential approach to software development that begins at the system level and progresses through the analysis, design, coding, testing, and maintenance.

In the literature, people have identified from 5 to 8 stages of software development

The five stages above are as follows:

1. **Requirement Analysis and Definition:** What - The systems services, constraints and goals are established by consultation with system users. They are then defined in detail and serve as a system specification.

13

Linear Sequential Model/ waterfall model

2. **System and Software Design:** How – The system design process partitions the requirements to either hardware of software systems. It establishes and overall system architecture. Software design involves fundamental system abstractions and their relationships.
3. **Implementation and Unit Testing:** - How – During this stage the software design is realized as a set of programs or program units. Unit testing involves verifying that each unit meets its specifications.
4. **Integration and system testing**: The individual program unit or programs are integrated and tested as a complete system to ensure that the software requirements have been met. After testing, the software system is delivered to the customer.
5. **Operation and Maintenance**: Normally this is the longest phase of the software life cycle. The system is installed and put into practical use. Maintenance involves correcting errors which were not discovered in earlier stages of the life-cycle, improving the implementation of system units and enhancing the system's services as new requirements are discovered.

- In principle, the result of each phase is one or more documents, which are approved. No phase is complete until the documentation for that phase has been completed and products of that phase have been approved. The following phase should not start until the previous phase has finished.

14

Linear Sequential Model/ waterfall model

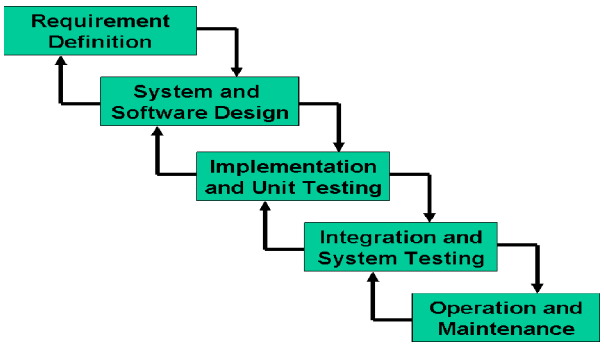
Real projects rarely follow the sequential flow that the model proposes.

In general these phases overlap and feed information to each other hence there should be an element of iteration and feedback.

A mistake caught any stage should be referred back to the source and all the subsequent stages need to be revisited and corresponding documents should be updated accordingly.

15

This feedback path is shown in the following diagram



Because of the costs of producing and approving documents, iterations are costly and require significant rework

16

Incremental Models

In the incremental models, as opposed to the waterfall model, the product is partitioned into smaller pieces, which are then built and delivered to the client in increments at regular intervals.

Since each piece is much smaller than the whole, it can be built and sent to the client quickly.

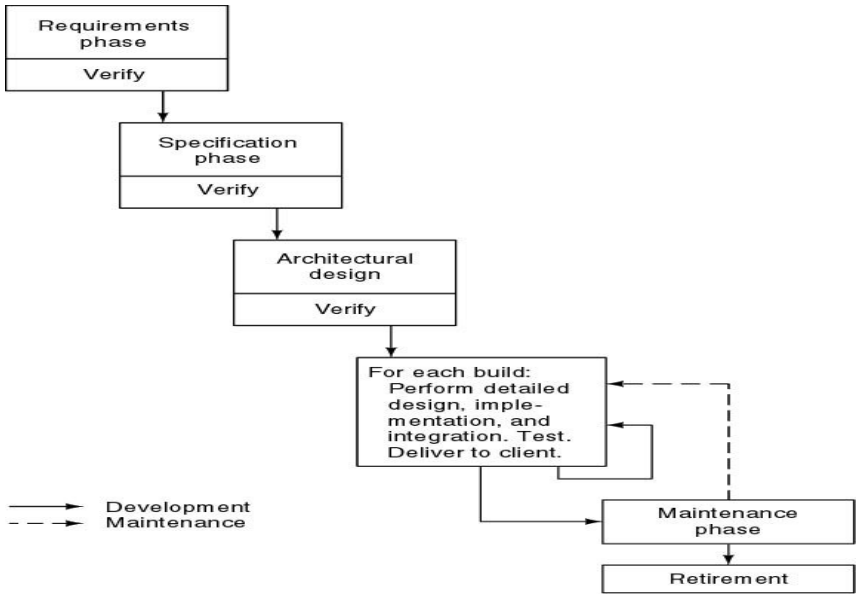
This results in quick feedback from the client and any requirement related errors or changes could be incorporated at a much lesser cost.

It is therefore less traumatic as compared to the waterfall model. It also required smaller capital outlay and yield a rapid return on investment.

However, this model needs and open architecture to allow integration of subsequent builds to yield the bigger product. A number of variations are used in object-oriented life cycle models.

17

Incremental Models



18

Variations of Incremental Models

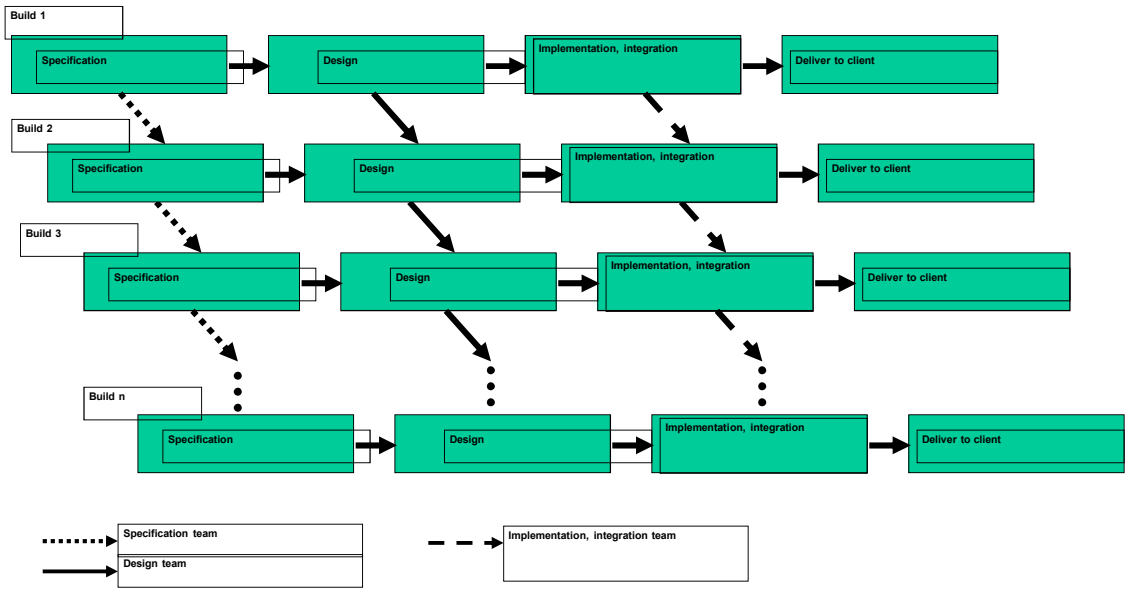
There are two fundamental approaches to the incremental development. In the first case, the requirements, specifications, and architectural design for the whole product are completed before implementation of the various builds commences

In a more risky version, once the user requirements have been elicited, the specifications of the first build are drawn up. When this has been completed, the specification team turns to the specification of the second build while the design team designs the first build. Thus the various builds are constructed in parallel, with each team making use of the information gained in the all the previous builds.

This approach incurs the risk that the resulting build will not fit together and hence requires careful monitoring.

19

Risky Version of Incremental Model



20