Chapter 1

Definition 1.0.1: Software process

- **Specification**: defining what the system should do
- **Design**: plans the overall architecture of the system including its modules, components and interfaces
- **Development/implementation**: s the implementation of the design
- **Verification and Validation**: checking that it meets the design specifications and does what the customer wants.
- **Evolution**: Checking that the system in response to changing customer needs
- **Documentation**: Model and describe software products
- **Planning**: Plan and manage the software development process

Note:-

It's generally better to spend more time at the beginning of the project planning it out and what to do than jump into coding. Make many tiny commits rather than one huge one. Implement tests early.

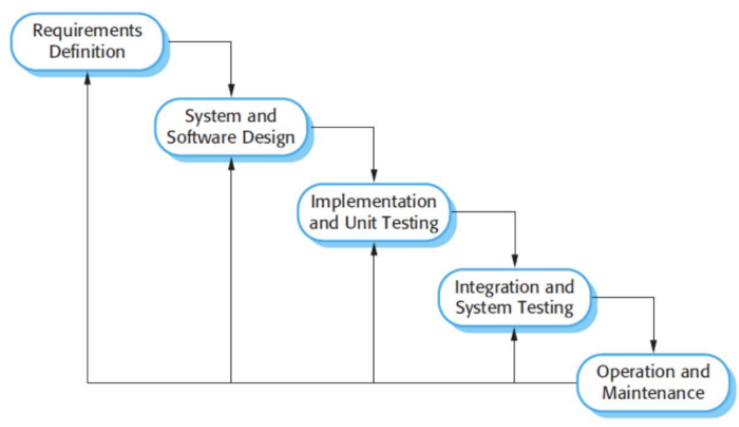
Corollary 1.0.1 Planning models

General Models

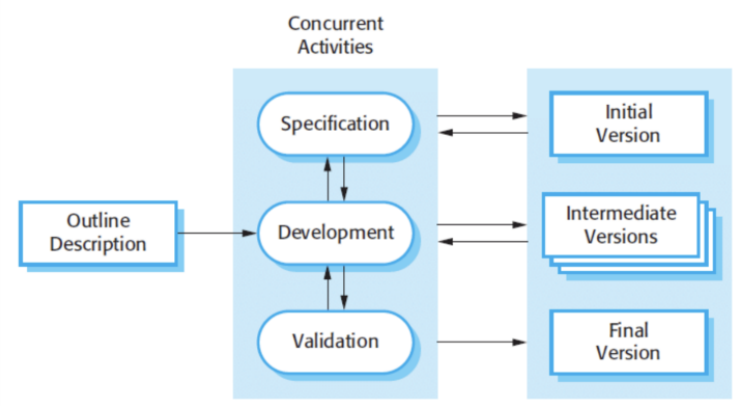
- Plan-Driven: Processes are planned in advance and progress is measured against the large plan.
- Agile: Planning is incremental and easier to pivot in accordance to changing requirements.

Specific Models

- Waterfall: Plan-Driven. Separates phases based on specification, development, validation, and evolution.



- Incremental: Specification, development and validation are interleaved. Can be either plan-driven or agile. Developed as a series of incremental versions with each version adding functionality to the previous.

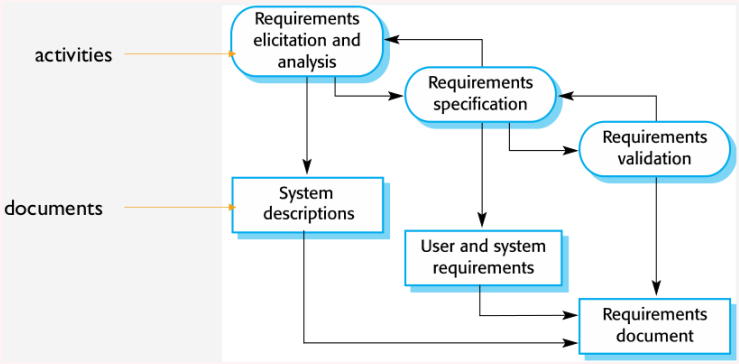


- Integration and Configuration: System is assembled from existing components and configured in accordance to specifications

How do we figure out what we are going to build?

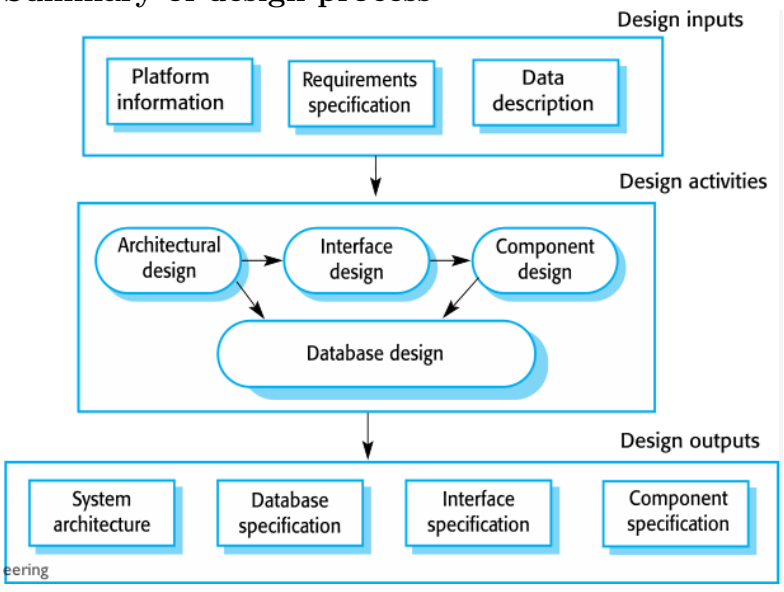
Definition 1.0.2: Requirements Engineering

- Elicitation and Analysis: What do the stakeholders expect from the system?



- Specification: Defining requirements in detail.

Summary of design process



Chapter 2

Project planning

Definition 2.0.1: Waterfall model

Characterized by:

- Sequential: one phase after another
- Phase-based: one phase at a time
- Document-driven: Transition between phases results in the production of formal documentation

Note:-

Main problem is the lack of feedback until it's fully built, increases the risk of misunderstanding about requirements

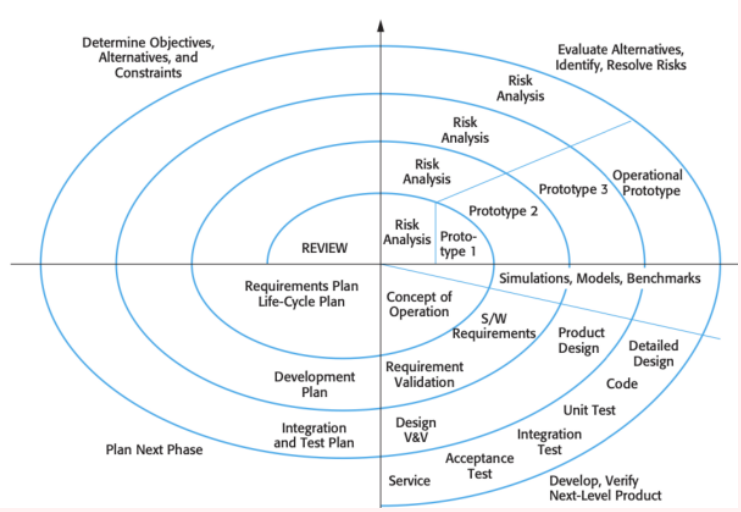
Pros

- Widely used with systems with clear user requirements
- The output of the previous phase is input to the next phase
- Work is performed within one monolithic cycle

Cons

Definition 2.0.2: Spiral Model

Plan starts out vague and high-level, then is prototyped and iterateratively tested until its specifics form



Pros

- Widely used for system with unclear user requirements
- Use for large complex systems
- One iteration encompasses one or more acivities in the SW development
- Work is performed in iterations, enhancing the existing versions until the system is complete
- Change is easier to adapt to

Cons

- Not efficient for small projects
- More management attention is required
- Not all requirements are gathered up front for the entire lifecycle thus design issues may arise

Definition 2.0.3: Incremental Model

Instead of providing the singular delivery such as in waterfall, incremental delivery delivers several incremental features which are a subset of the entire system. **Pros**

- Customers can see the value being added every delivery cycle
- Retains flexibility while communicating progress and what is yet to be built
- Lower risk of overall project failure
- Higher priority systems can receive more testing
- Better for most businesses, e-commerce, and personal systems

Cons

- Difficult to implement for *replacement* systems as increments have less functionality than the system being replaced
- Problems for system architechture since not all requirements are collected upfront for the entire software lifecycle
- If there is a change in developers they may have a hard time understanding the current stage of progress and where to go

Note:-

Typically Iterative and Incremental models are combined due to the multiple opportunities for checking that the product is in line with requirements

Definition 2.0.4: Agile

- Program specification, design, and implementation are inter-leaved
- Focuses more on the software rather than "design" of it
- Developed as a series of incremental versions with stakeholders involved in version specification and evaluation

Principles

- Customer involvement - The customer must be closely involved throughout the development process. Customers provide and prioritize system requirements and evaluate the system.
- Incremental delivery - The software is developed in increments with the customer specifying the requirements in each increment.
- People not process - The skills of the development team should be recognised and exploited. The team should be left to develop their own ways of working without prescriptive processes.
- Embrace change - Expect the system requirements to change and design the system to accommodate these changes.
- Maintain simplicity - Focus on simplicity in both the software being developed and in the development process used. Actively work to eliminate complexity from the system.

Corollary 2.0.1 SCRUM

There are 3 phases in SCRUM

1. The initial phase is an outline planning phase where you establish the general objectives for the project and design the software architecture.
2. This is followed by a series of sprint cycles, where each cycle develops an increment of the system.
3. The project closure phase wraps up the project, completes required documentation such as system help frames and user manuals and assesses the lessons learned from the project.

Definition 2.0.5: Extreme Programming (XP)

- Requirements are specified using scenarios which are composed of tasks
- Programming pairs develop tests for each tast *before* writing the code. All tests must be run for every build and only accepted if all tests are successful
- Customers are responsible for defining scenarios which is then broken down into tasks

Principles

- Incremental planning - Requirements included in a release are determined by the time available and their relative priority.
- Small Releases - The minimal useful set of functionality that provides business value is developed first. Releases of the system are frequent and incrementally add to the first release.
- Simple Design - Enough design is carried out to meet the current requirements and no more.
- Test first development - An automated unit test framework is used to write tests for a new piece of functionality before that functionality itself is implemented.
- Refactoring - All developers are expected to refactor the code continuously as soon as possible code improvements are found. This keeps the code simple and maintainable.
- Pair Programming - Developers work in pairs, checking each other's work and providing the support to always do a good job.
- Collective Ownership - The pairs of developers work on all areas of the system, so that no islands of expertise develop and all the developers own all the code. Anyone can change anything.
- Continuous Integration - As soon as work on a task is complete it is integrated into the whole system. After any such integration, all the unit tests in the system must pass.
- On-site Customer - The customer should be available full time for the use of the XP team. The customer is a member of the development team and is responsible for bringing system requirements to the team for implementation.