CSE-327
Software Engineering
Week-2

Reference book

- Chapter-4 : Software Engineering: A Practitioner's Approach – Roger S. Pressman (8th Edition)
- Chapter-2: Software Engineering – Ian Sommerville (9th Edition)

Software Process Model

- A software process model is a simplified representation of the software development process. The models specify the stages and order of a process.
- So, we can think of this as a representation of the order of activities of the process and the sequence in which they are performed.

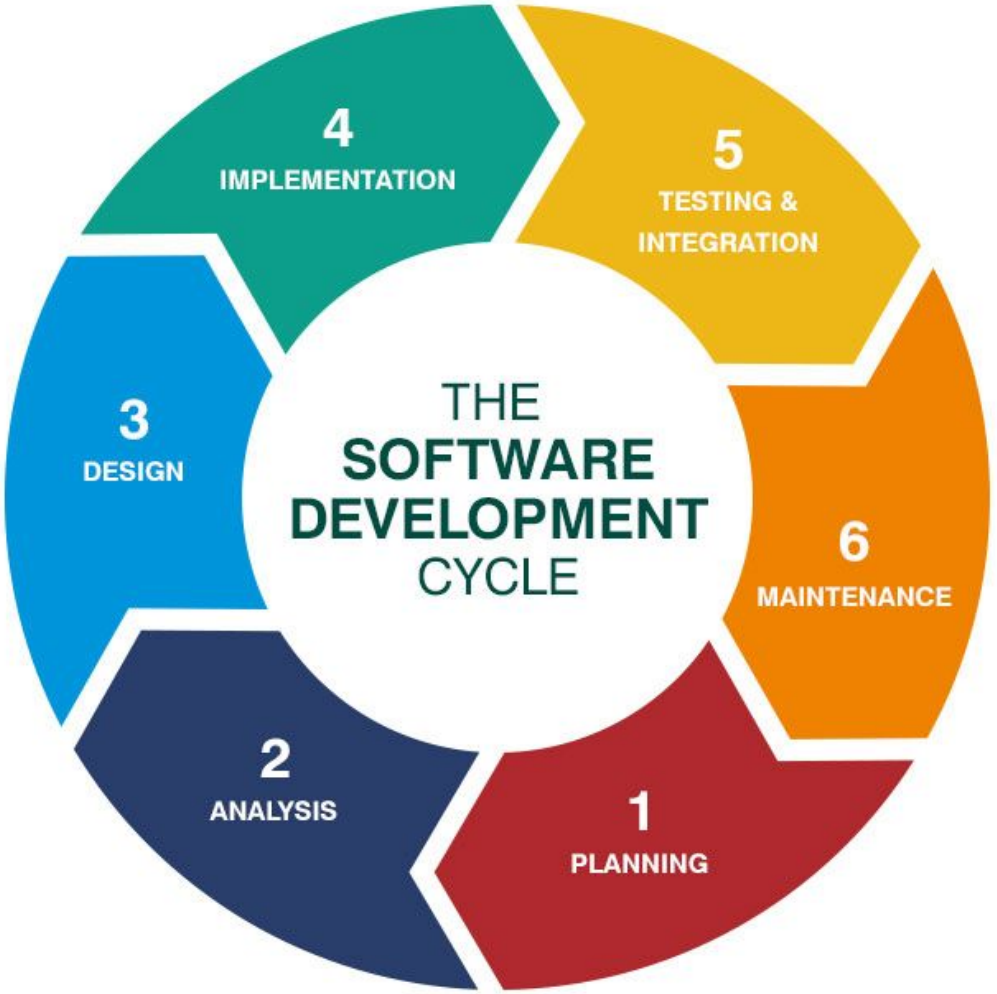
Example of SDLC Models

There are many kinds of process models for meeting different requirements. We refer to these as SDLC models (Software Development Life Cycle models). The most popular and important SDLC models are as follows:

- Waterfall model
- V model
- Incremental model
- RAD model
- Agile model
- Iterative model
- Prototype model
- Spiral model

SDLC

- The software deve
the cost-effective a
that the developm
build high-quality s
- The goal of SDLC is
through forward p
customer expectat
beyond.



SDLC model vs Software Process Model

Parameters	Life cycle model	Software process model
Definition	product states	activities performed
Relations	state transformed by activity	activity works on state
Generality	multiple life cycle models	unique for organization
Domain and scope	specific, domain-dependent	general, project-independent, organization-dependent
View	technical	management, technical and support
Focused on the object	product outputted by each phase	activity performed in each subprocess

Plan-driven vs Agile Process

- Software processes are categorized as either 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.
- In 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.

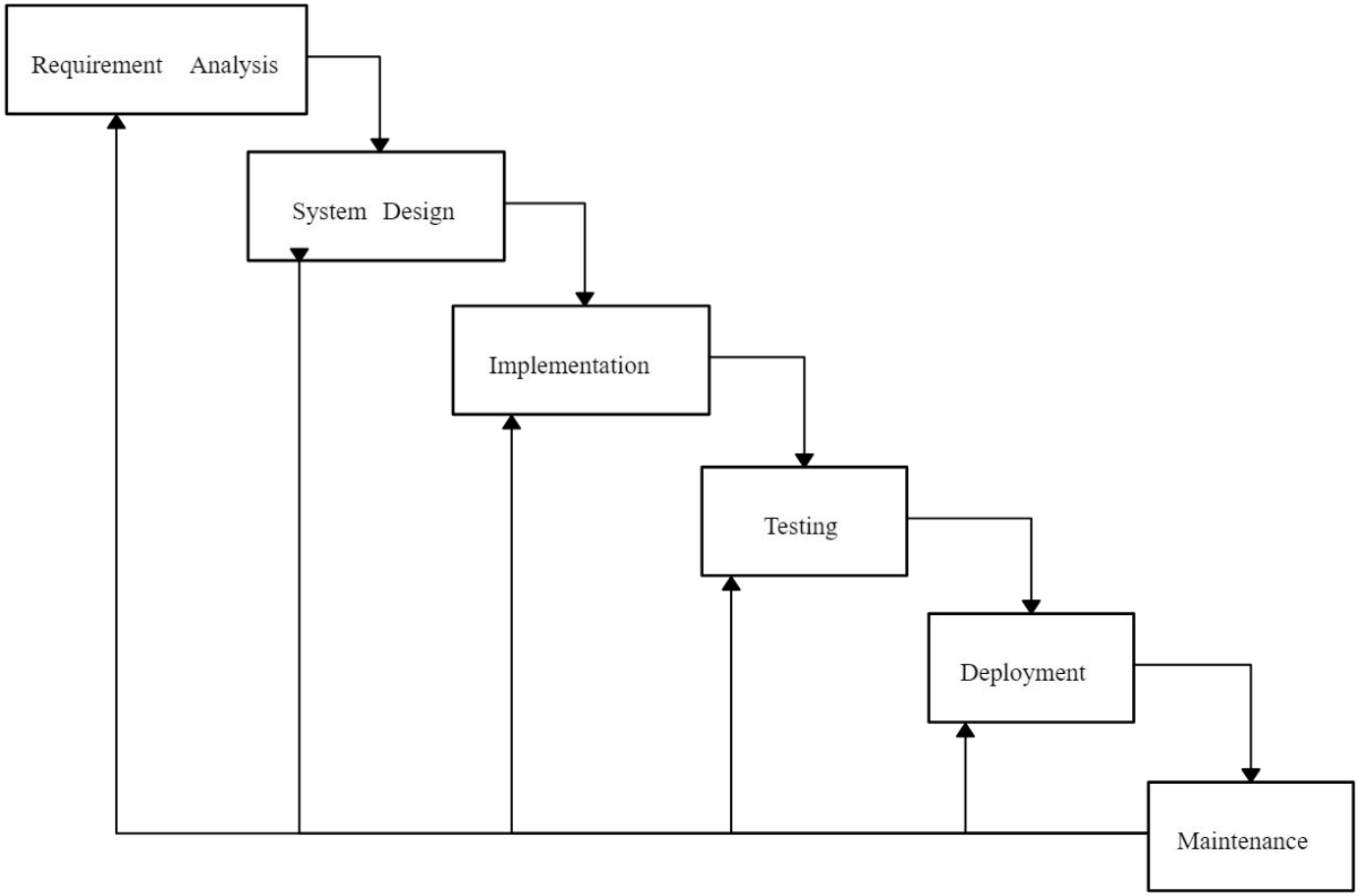
Plan-driven vs Agile Process

	Software Process Model	Advantages	Disadvantages
Plan Driven	• Waterfall • Incremental Development • Iterative development • Spiral Development • Prototype Model • Rapid Application Development	• Suitable for large systems and teams. • Handles highly critical systems effectively. • Appropriate for stable development environment. • Require experienced personnel at the beginning. • Success achieved through structure and order.	• Longer length in each iteration or increment. • Cannot accommodate changes any time. • Lack of user involvement throughout the life cycle of the product. • Costly for the dynamic development environment. • Assume that, future changes will not occur.
Agile	• Scrum model • Extreme Programming (XP) • Dynamic System Development Method • Kanban • Feature Driven Development	• Suitable for small to medium systems and teams. • Can accommodate changes at any time. • Effective for the dynamic development environment. • Required expert agile personnel throughout the life cycle. • Success achieved through freedom and chaos.	• Not suitable for large systems (except FDD). • Shorter length in each iteration. • Can accommodate changes at any time. • Costly for the stable development environment. • Assume that, frequent future changes will occur.

Waterfall Model

- The waterfall model is the oldest paradigm for software engineering.
- The waterfall model, sometimes called the classic life cycle, suggests a systematic, sequential approach to software development that begins with customer specification of requirements and progresses through planning, modeling, construction, and deployment, providing ongoing support of the completed software.

Waterfall Model



When to use Waterfall Model?

Waterfall Methodology can be used when:

- This model is used only when the requirements are very well known, clear and fixed.
- Requirements are not changing frequently
- Application is not complicated and big
- Project is short
- Environment is stable
- Technology and tools used are not dynamic and is stable
- Resources are available and trained

Advantages: Waterfall Model

- This model is simple and easy to understand and use.
- It is easy to manage due to the rigidity of the model – each phase has specific deliverables and a review process.
- In this model phases are processed and completed one at a time.
- Phases do not overlap.
- Waterfall model works well for smaller projects where requirements are very well understood

Disadvantages: Waterfall Model

- Once an application is in the testing stage, it is very difficult to go back and change something that was not well-thought out in the concept stage.
- No working software is produced until late during the life cycle.
- High amounts of risk and uncertainty.
- Not a good model for complex and object-oriented projects.
- Poor model for long and ongoing projects.
- Not suitable for the projects where requirements are at a moderate to high risk of changing.

Problem with Waterfall Model

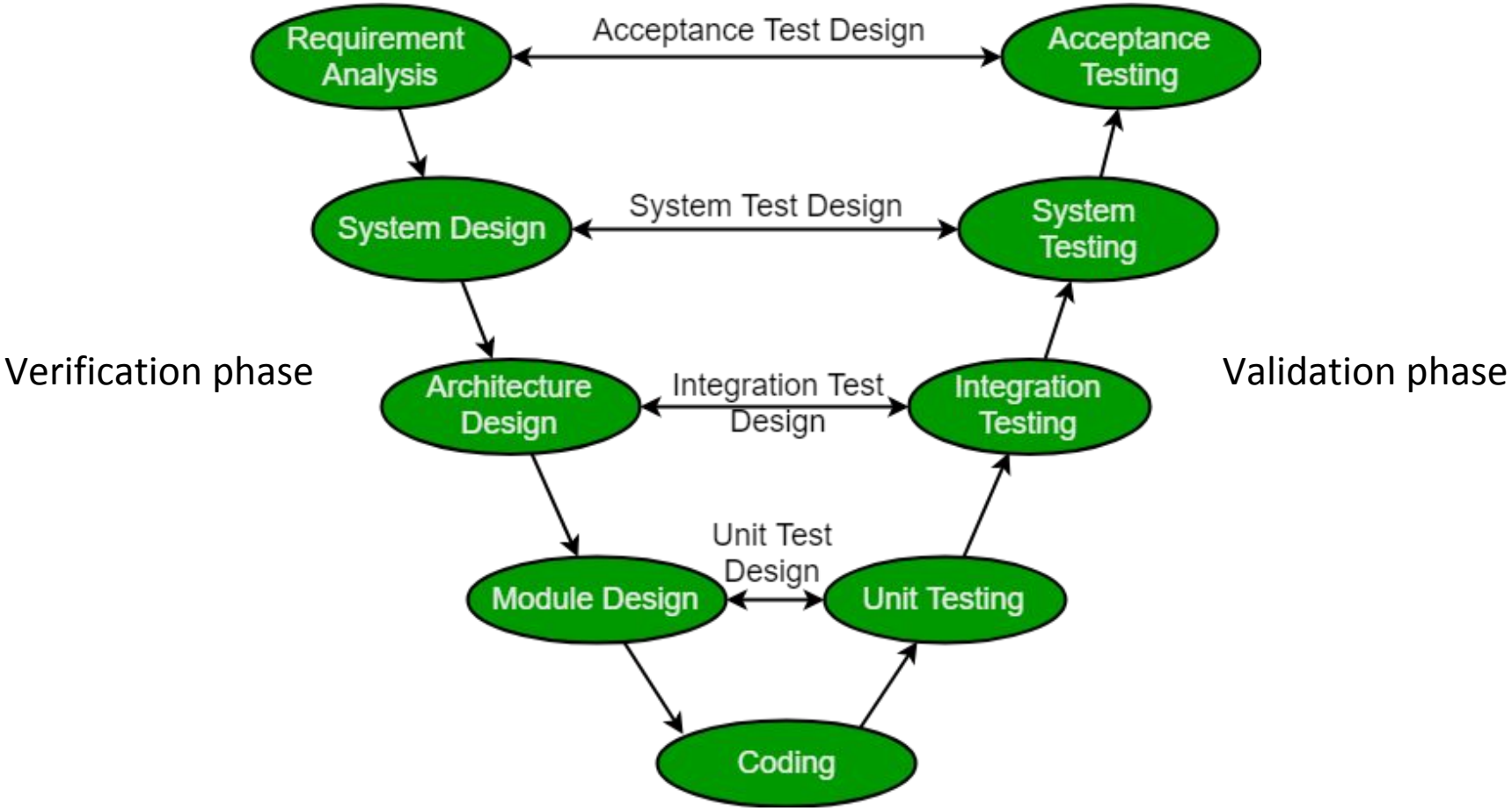
The problems that are sometimes encountered when the waterfall model is applied are:

- Real projects rarely follow the sequential flow that the model proposes.
- It is often difficult for the customer to state all requirements explicitly.
- The customer must have patience. A working version of the software will not be available until late in the project time span.

V-Model

- The costs of fixing a defect increase across the development lifecycle. The earlier in life cycle a defect is detected, the cheaper it is to fix it.
- To address this concern, the V model of testing was developed where for every phase, in the Development life cycle there is a corresponding Testing phase

V-Model


Requirement Analysis
Acceptance Test Design
Acceptance Testing
System Design
System Test Design
System Testing
Architecture Design
Integration Test Design
Integration Testing
Unit Test Design
Module Design
Unit Testing
Coding
Verification phase
Validation phase

V-model

- As a software team moves down the left side of the V, basic problem requirements are refined into progressively more detailed and technical representations of the problem and its solution.
- Once code has been generated, the team moves up the right side of the V, essentially performing a series of tests (quality assurance actions) that validate each of the models created as the team moves down the left side.
- In reality, there is no fundamental difference between the classic life cycle and the V-model. The V-model provides a way of visualizing how verification and validation actions are applied to earlier software engineering work.

V-model

- Advantages of V-model:
  - Simple and easy to use.
  - Testing activities like planning, test designing happens well before coding. This saves a lot of time. Hence higher chance of success over the waterfall model.
  - Proactive defect tracking – that is defects are found at early stage.
  - Works well for small projects where requirements are easily understood.

V-model

- Disadvantages of V-model:
  - Very rigid and least flexible.
  - Software is developed during the implementation phase, so no early prototypes of the software are produced.
  - If any changes happen in midway, then the test documents along with requirement documents has to be updated.

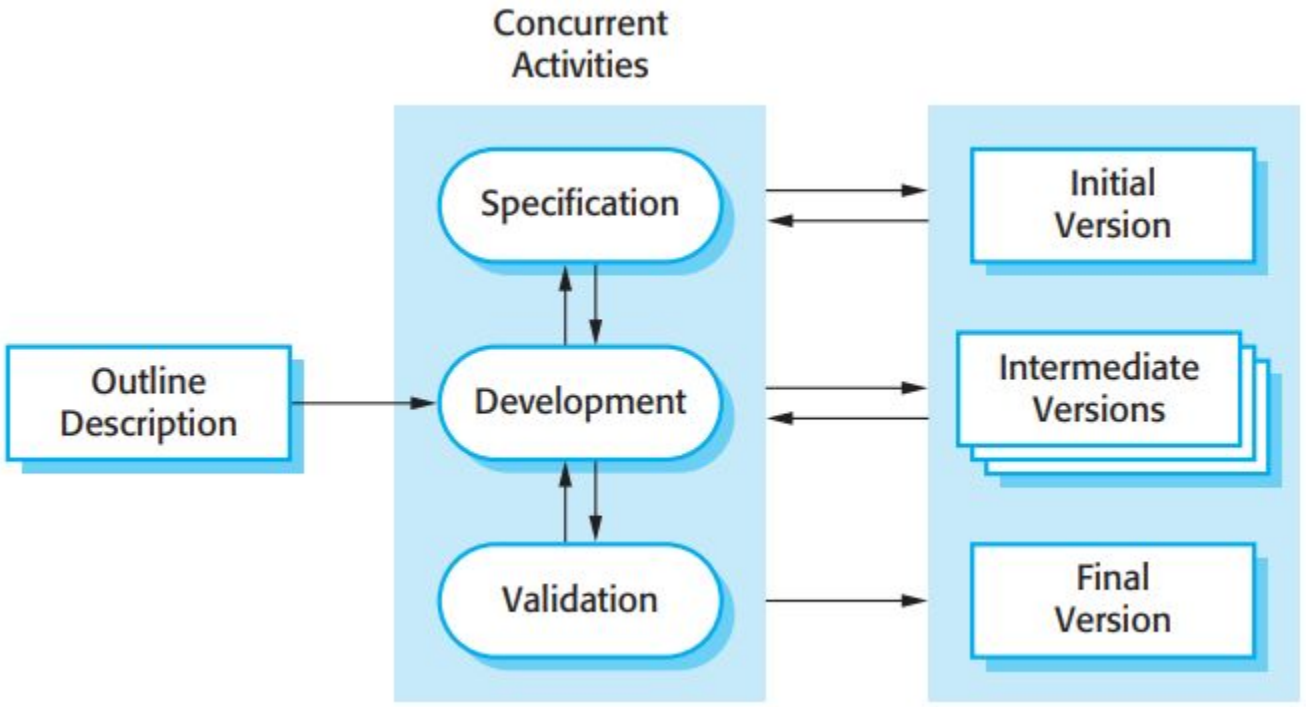
When to use the V-model

- The V-shaped model should be used for small to medium sized projects where requirements are clearly defined and fixed.
- The V-Shaped model should be chosen when sample technical resources are available with needed technical expertise.

Incremental Model

- The incremental model delivers a series of releases, called increments, that provide progressively more functionality for the customer as each increment is delivered.
- Multiple development cycles take place here, making the life cycle a “multi-waterfall” cycle.
- Rather than deliver the system as a single delivery, the development and delivery is broken down into increments with each increment delivering part of the required functionality.
- User requirements are prioritized and the highest priority requirements are included in early increments.

Incremental Development Model



Incremental Model: How does it work

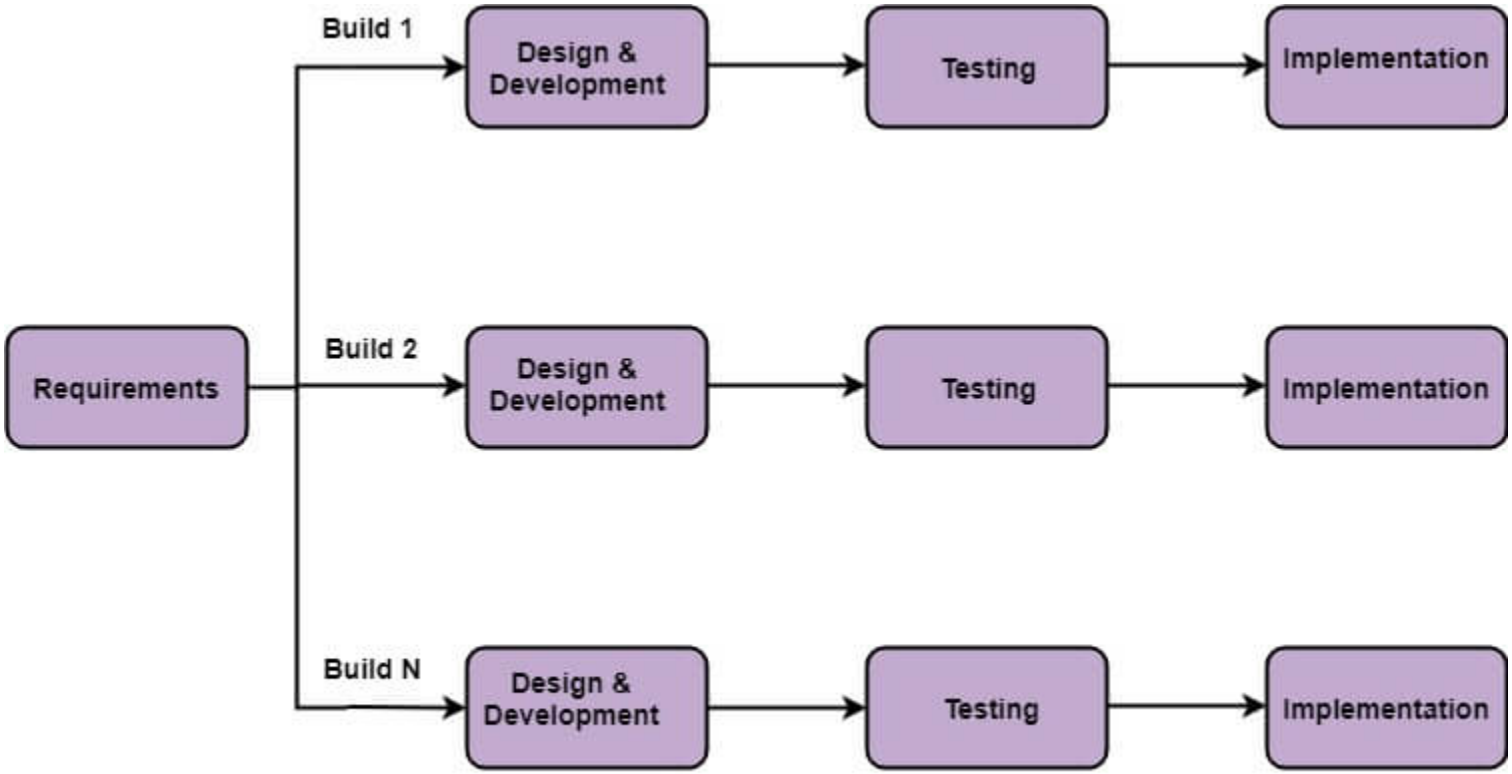


Fig: Incremental Model

Incremental Model

- For example, word-processing software developed using the incremental paradigm might deliver basic file management, editing, and document creation functions in the first increment;
- more sophisticated editing and document creation capabilities in the second increment;
- spelling and grammar checking in the third increment;
- and advanced page layout capability in the fourth increment.

Incremental Model

- When an incremental model is used, the first increment is often a core product. That is, basic requirements are addressed but many supplementary features remain undelivered.
- The core product is used by the customer (or undergoes detailed evaluation). As a result of use and/or evaluation, a plan is developed for the next increment.
- The plan addresses the modification of the core product to better meet the needs of the customer and the delivery of additional features and functionality.
- This process is repeated following the delivery of each increment, until the complete product is produced.

Advantages of Incremental Model

- Generates working software quickly and early during the software life cycle.
- This model is more flexible – less costly to change scope and requirements.
- It is easier to test and debug during a smaller iteration.
- In this model customer can respond to each built.
- Lowers initial delivery cost.
- Easier to manage risk because risky pieces are identified and handled during it’d iteration.

Disadvantages of Incremental Model

- Needs good planning and design.
- Needs a clear and complete definition of the whole system before it can be broken down and built incrementally.
- Total cost is higher than waterfall.

When to use Incremental models?

- Requirements of the system are clearly understood
- When demand for an early release of a product arises
- When software engineering team are not very well skilled or trained
- When high-risk features and goals are involved
- Such methodology is more in use for web application and product based companies
- A new technology is being used

Problem with Incremental Model

From a management perspective, the incremental approach has two problems:

- The process is not visible. Managers need regular deliverables to measure progress. If systems are developed quickly, it is not cost-effective to produce documents that reflect every version of the system.
- System structure tends to degrade as new increments are added. Unless time and money is spent on refactoring to improve the software, regular change tends to corrupt its structure. Incorporating further software changes becomes increasingly difficult and costly.

Reuse-oriented Software Engineering

- In the majority of software projects, there is some software reuse.
- This often happens informally when people working on the project know of designs or code that are similar to what is required.
- They look for these, modify them as needed, and incorporate them into their system.
- Reuse-oriented software engineering has the obvious advantage of reducing the amount of software to be developed and so reducing cost and risks.
- It usually also leads to faster delivery of the software.

Reuse-oriented Software Eng.

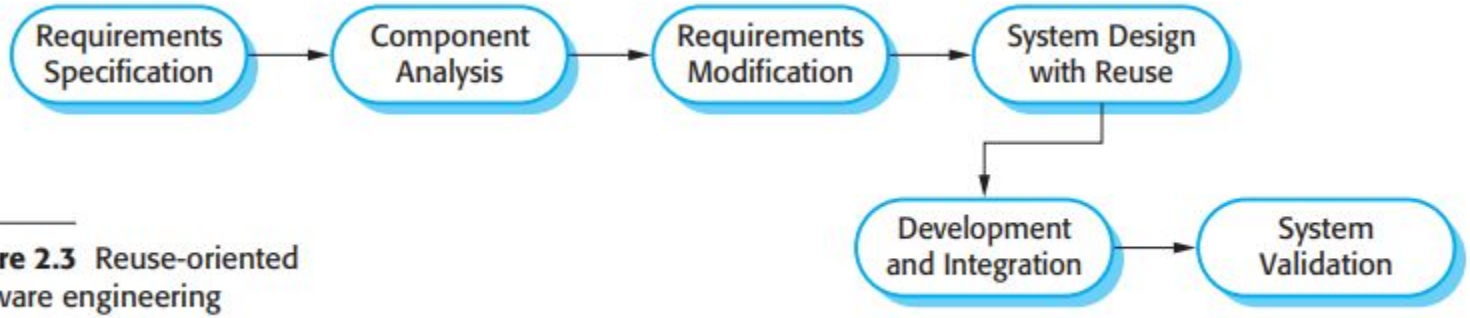


Figure 2.3 Reuse-oriented software engineering

Evolutionary Process Models

- Evolutionary models are iterative.
- They are characterized in a manner that enables you to develop increasingly more complete versions of the software.

  ✔ Prototyping Model
  ✔ Spiral Model

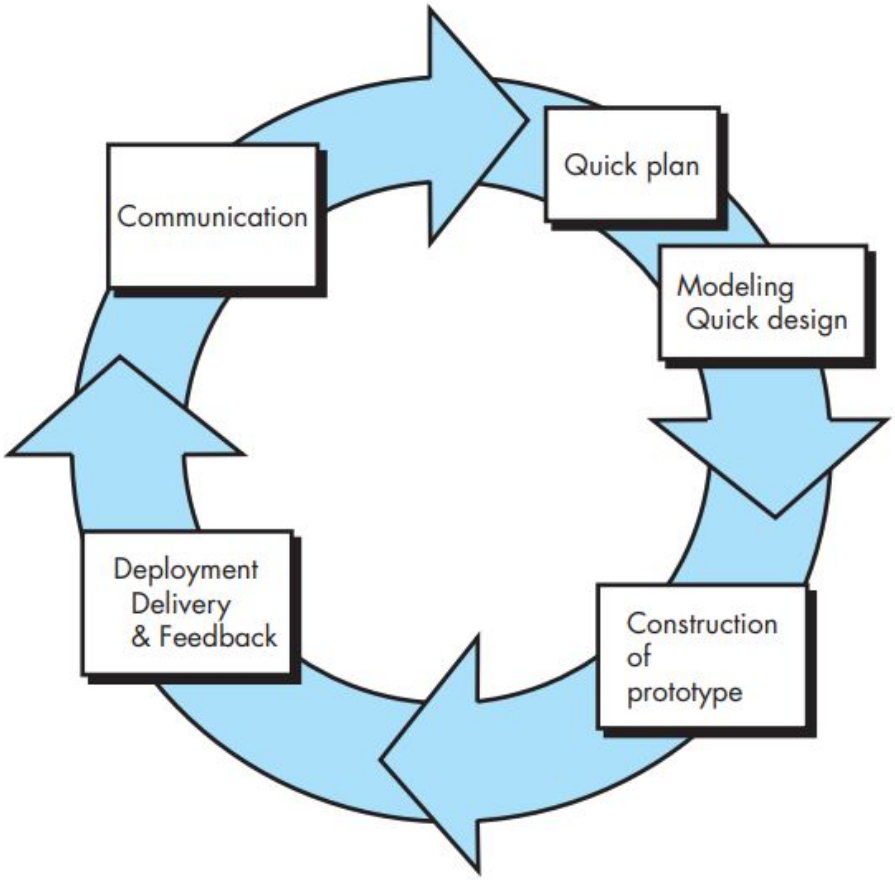
Prototyping Model

- A prototype is an initial version of a system used to demonstrate concepts and try out design options.
- It also creates base to produce the final system or software.
- It works best in scenarios where the project's requirements are not known in detail.
- It is an iterative, trial and error method which takes place between developer and client.

A prototype can be used in:

- The requirements engineering process to help with requirements elicitation and validation;
- In design processes to explore options and develop a UI design;
- In the testing process to run back-to-back tests.

The Prototyping Paradigm



Development: Prototyping

Construction of prototype

- May be based on rapid prototyping languages or tools
- May involve leaving out functionality
  - Prototype should focus on areas of the product that are not well-understood;
  - Error checking and recovery may not be included in the prototype;
  - Focus on functional rather than non-functional requirements such as reliability and security

Throw Away: Prototyping

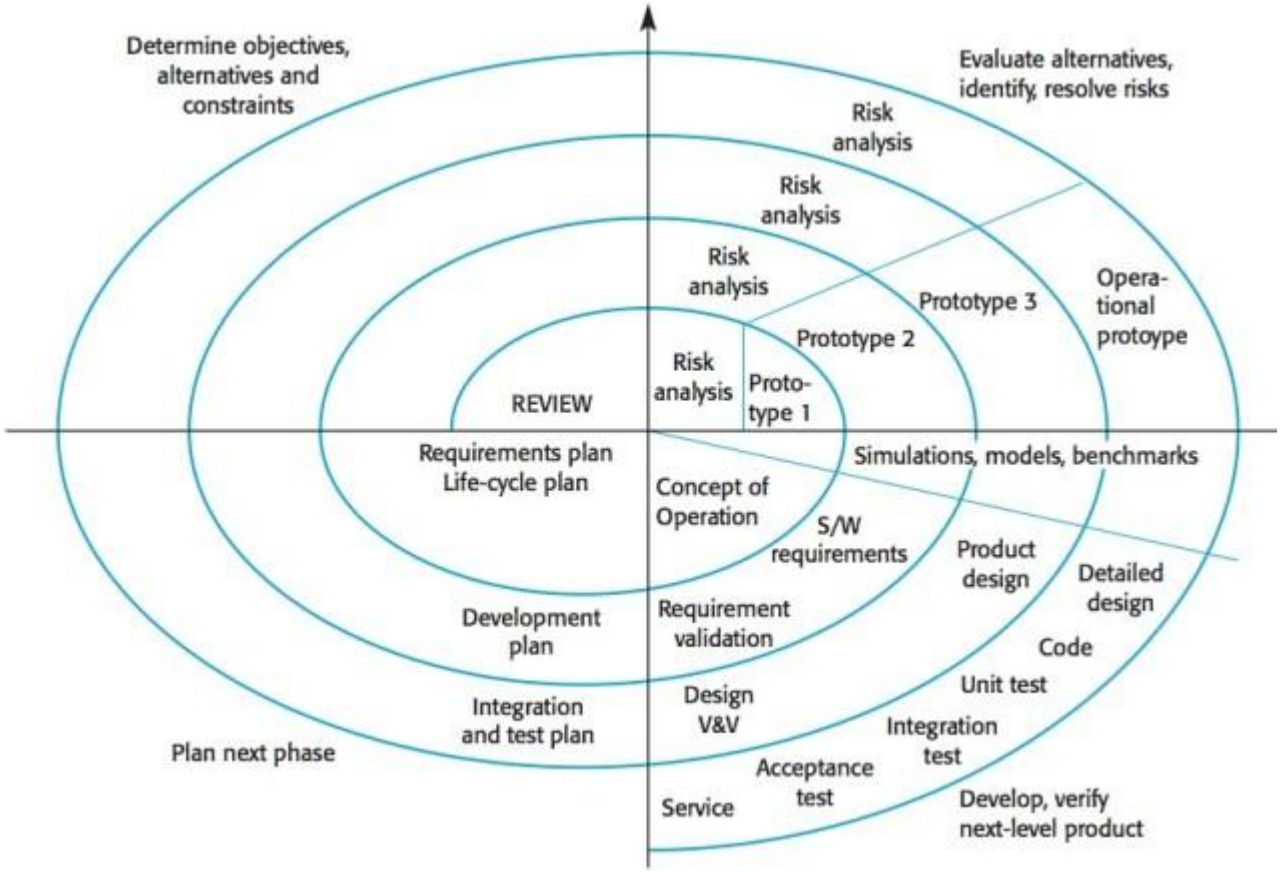
Prototypes should be discarded after development as they are not a good basis for a production system:

- It may be impossible to tune the system to meet nonfunctional requirements;
- Prototypes are normally undocumented;
- The prototype structure is usually degraded through rapid change;
- The prototype probably will not meet normal organizational quality standards.

Spiral Model

- Process is represented as a spiral rather than as a sequence of activities with backtracking.
- Each loop in the spiral represents a phase in the process.
- Risks are explicitly assessed and resolved throughout the process.
- Spiral Model is a risk-driven software development process model. It is a combination of waterfall model and iterative model.

Spiral Model



Sectors: The Spiral

- Objective setting
  - Specific objectives for the phase are identified.
- Risk assessment and reduction
  - Risks are assessed and activities put in place to reduce the key risks.
- Development and validation
  - A development model for the system is chosen which can be any of the generic models.
- Planning
  - The project is reviewed and the next phase of the spiral is planned.

When to use the Spiral Model?

- A Spiral model in software engineering is used when project is large.
- When releases are required to be frequent, spiral methodology is used.
- When creation of a prototype is applicable
- When risk and costs evaluation is important
- Spiral methodology is useful for medium to high-risk projects
- When requirements are unclear and complex, Spiral model in SDLC is useful
- When changes may require at any time
- When long term project commitment is not feasible due to changes in economic priorities

Advantages of Spiral Model:

- Software is produced early in the software life cycle.
- Risk handling is one of important advantages of the Spiral model, it is best development model to follow due to the risk analysis and risk handling at every phase.
- Flexibility in requirements. In this model, we can easily change requirements at later phases and can be incorporated accurately. Also, additional Functionality can be added at a later date.
- It is good for large and complex projects.
- It is good for customer satisfaction. We can involve customers in the development of products at early phase of the software development. Also, software is produced early in the software life cycle.
- It is suitable for high risk projects, where business needs may be unstable. A highly customized product can be developed using this.

Disadvantages of Spiral Model

- It is not suitable for small projects as it is expensive.
- It is much more complex than other SDLC models.
- Too much dependable on Risk Analysis and requires highly specific expertise.
- Difficulty in time management. As the number of phases is unknown at the start of the project, so time estimation is very difficult.
- Spiral may go on indefinitely.
- End of the project may not be known early.
- It is not suitable for low risk projects.
- May be hard to define objective, verifiable milestones. Large numbers of intermediate stages require excessive documentation.

Task 1

Suppose you are the team leader of a software company. Under your supervision, a team has been formed. Now, one of your clients wants to develop antivirus software from your company. This software needs to identify potential threats and take the necessary actions. So, your team needs to release the software with the necessary precautions. Several times of the testing process to be applied to make this software operational. The client is not technically equipped.

- Predict which process model will be selected for this development with reasons and also demonstrate the whole building process?

Task 2

Suppose you are the team leader of a software company. Under your supervision, a team has been made. Now, a client wants to develop a security software from your company. This software is based on artificial intelligence. So, your team needs to release the software with the necessary precautions.

- Predict which process model will be selected for this development with reasons

Task 3

Suppose you are the project manager of a software company. Under your supervision, a team has been assigned. Now, one of your clients wants to develop an e-learning based mobile application from your company. So, your team needs to collect sample data from different educational institutions and then develop along wise. Several times of testing process need to be applied to make this software operational. This mobile application should be able to generate performance of progress report.

- Predict which process model will be selected for this development with reasons and also demonstrate the whole building process.

What is “Agility”?

- Effective (rapid and adaptive) response to change
- Effective communication among all stakeholders
- Drawing the customer onto the team
- Organizing a team so that it is in control of the work performed

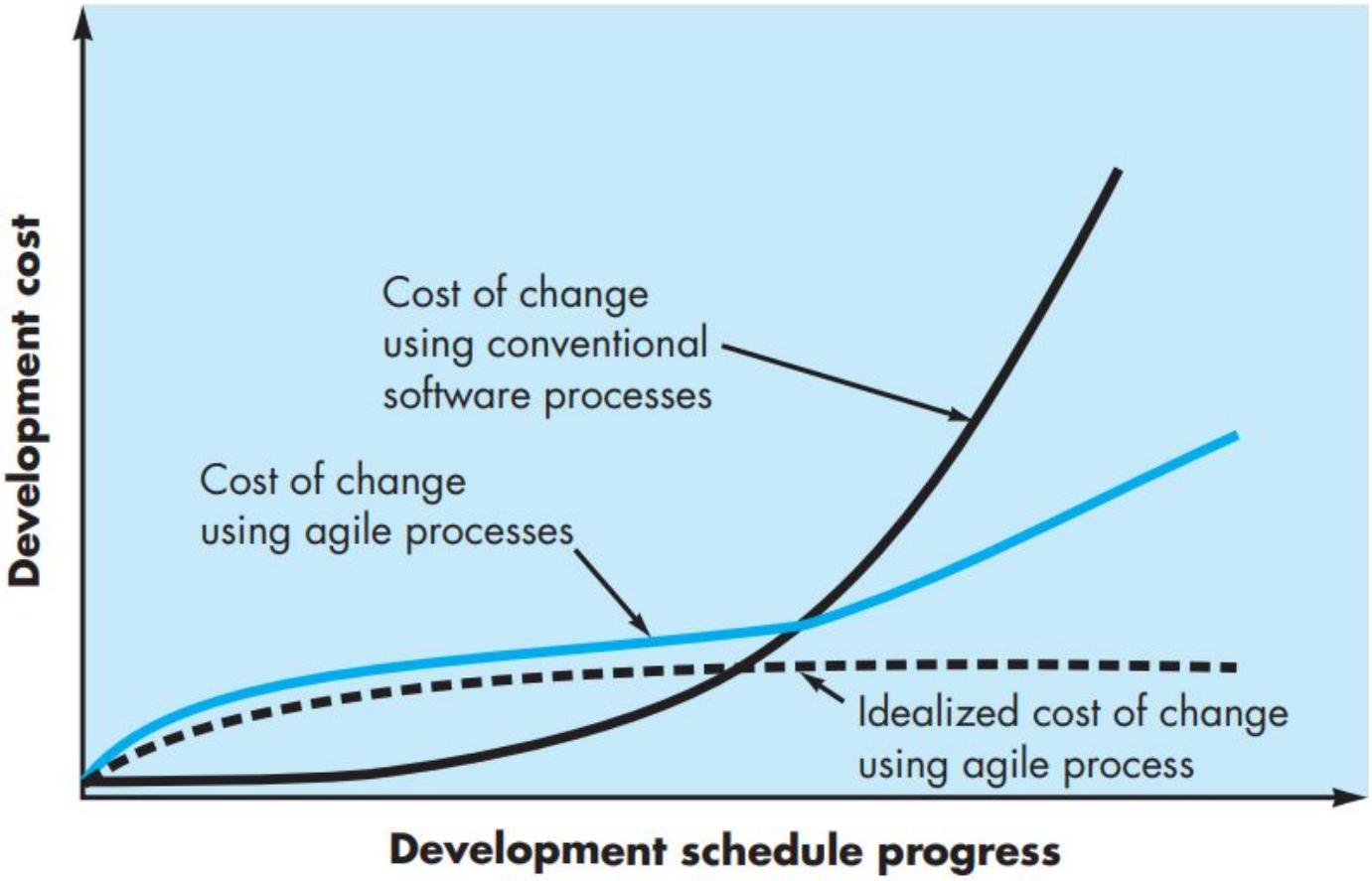
Yielding …

- Rapid, incremental delivery of software

Cont.

- Agile methods have been very successful for some types of system development:
  - Product development where a software company is developing a small or medium-sized product for sale.
  - Custom system development within an organization, where there is a clear commitment from the customer to become involved in the development process and where there are not a lot of external rules and regulations that affect the software

Agility and The Cost of Change



Agility Principles

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

4. Business people and developers must work together daily throughout the project.

5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

Agility Principles

7. Working software is the primary measure of progress.

8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefi nitely.

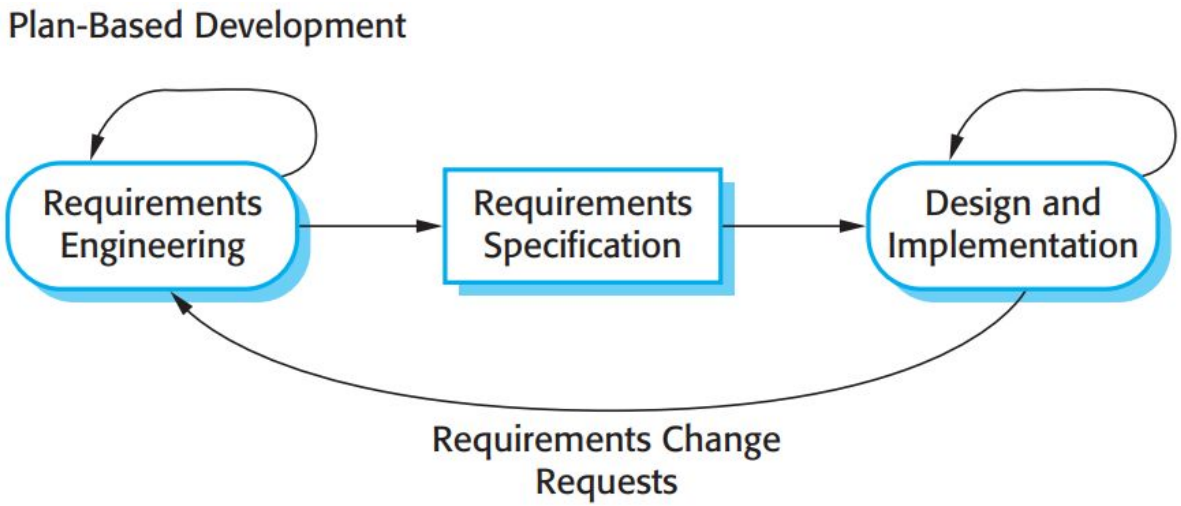
9. Continuous attention to technical excellence and good design enhances agility.

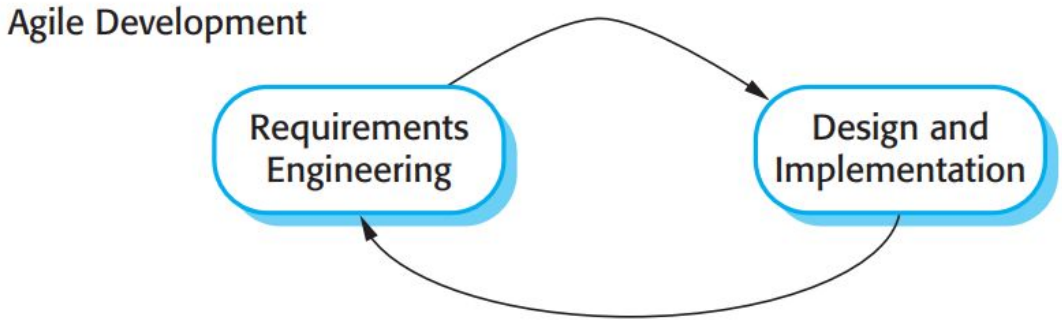
10. Simplicity—the art of maximizing the amount of work not done—is essential.

11. The best architectures, requirements, and designs emerge from self-organizing teams.

12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Plan-driven and Agile Specification





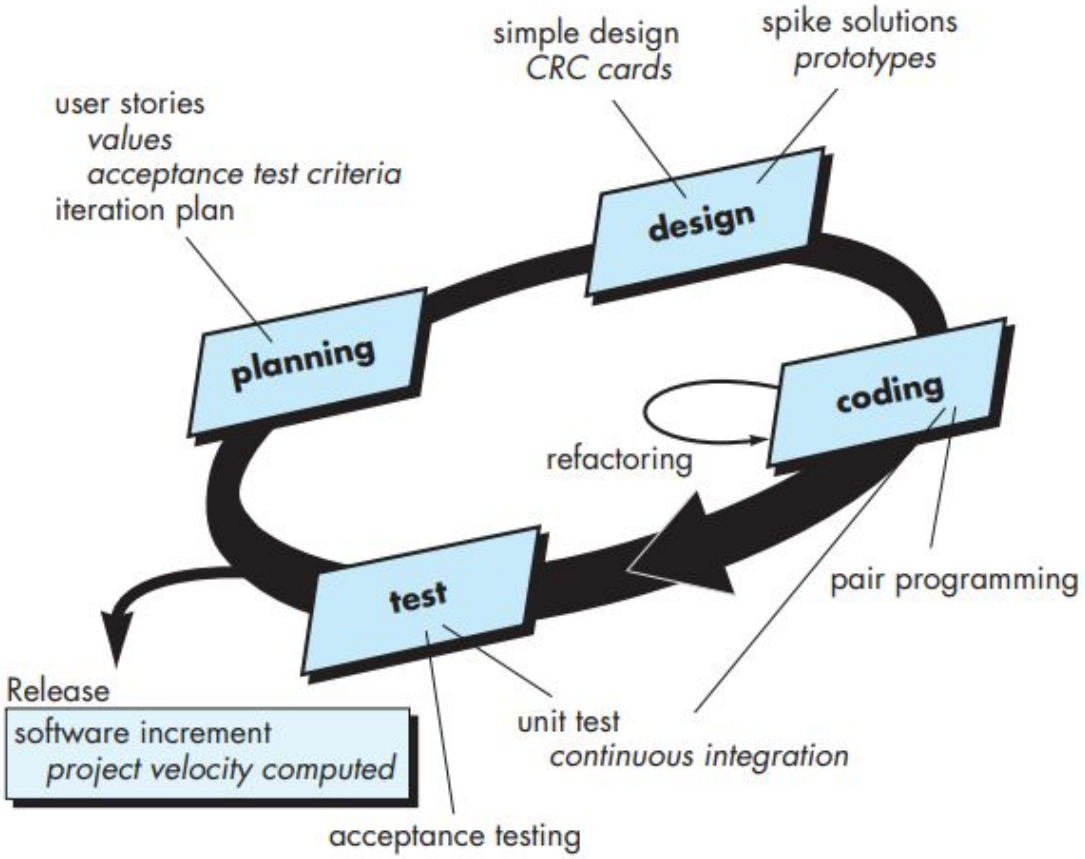
Plan-driven and agile development

- In a plan-driven approach, iteration occurs within activities with formal documents used to communicate between stages of the process. For example, the requirements will evolve and, ultimately, a requirements specification will be produced. This is then an input to the design and implementation process.
- In an agile approach, iteration occurs across activities. Therefore, the requirements and the design are developed together, rather than separately.

Extreme Programming (XP)

- The most widely used agile process, originally proposed by Kent Beck
- Extreme Programming uses an object-oriented approach as its development paradigm and encompasses a set of rules and practices that occur within the context of four framework activities: planning, design, coding, and testing

The Extreme Programming Process



The Extreme Programming Process

- XP Planning
  - Begins with the creation of "user stories"
  - Agile team assesses each story and assigns a cost
  - Stories are grouped to for a deliverable increment
  - A commitment is made on delivery date
  - After the first increment "project velocity" is used to help define subsequent delivery dates for other increments

project velocity is the number of customer stories implemented during the first release.

The Extreme Programming Process

- XP Design
  - Follows the KIS (keep it simple) principle. A simple design is always preferred over a more complex representation
  - Encourage the use of CRC cards (class-responsibility-collaborator). CRC cards identify and organize the object oriented classes that are relevant to the current software increment.
  - For difficult design problems, suggests the creation of "spike solutions"—a design prototype
  - Encourages "refactoring"—an iterative refinement of the internal program design

Refactoring is the process of changing a software system in such a way that it does not alter the external behavior of the code yet improves the internal structure. It is a disciplined way to clean up code [and modify/simplify the internal design] that minimizes the chances of introducing bugs. In essence, when you refactor you are improving the design of the code after it has been written.

The Extreme Programming Process

- XP Coding
  - Recommends the construction of a unit test for a store before coding commences
  - Encourages "pair programming"

XP recommends that two people work together at one computer workstation to create code for a story. This provides a mechanism for real-time problem solving (two heads are often better than one) and real-time quality assurance (the code is reviewed as it is created).
For example, one person might think about the coding details of a particular portion of the design while the other ensures that coding standards (a required part of XP) are being followed.

The Extreme Programming Process

- XP Testing
  - All unit tests are executed daily

  "Fixing small problems every few hours takes less time than fixing huge problems just before the deadline."

  - "Acceptance tests" are defined by the customer and executed to assess customer visible functionality

Scrum

- Scrum is an agile software development method that was proposed by Jeff Sutherland and his development team in the early 1990s.
- Scrum defines a set of development activities:
- Backlog
  - a prioritized list of project requirements or features that provide business value for the customer.
  - Items can be added to the backlog at any time (this is how changes are introduced).
  - The product manager assesses the backlog and updates priorities as required.

Cont.

- Sprints
  - consist of work units that are required to achieve a requirement defined in the backlog that must fit into a predefined time-box (typically 30 days).
  - Changes (e.g., backlog work items) are not introduced during the sprint.
  - Hence, the sprint allows team members to work in a short-term, but stable environment.

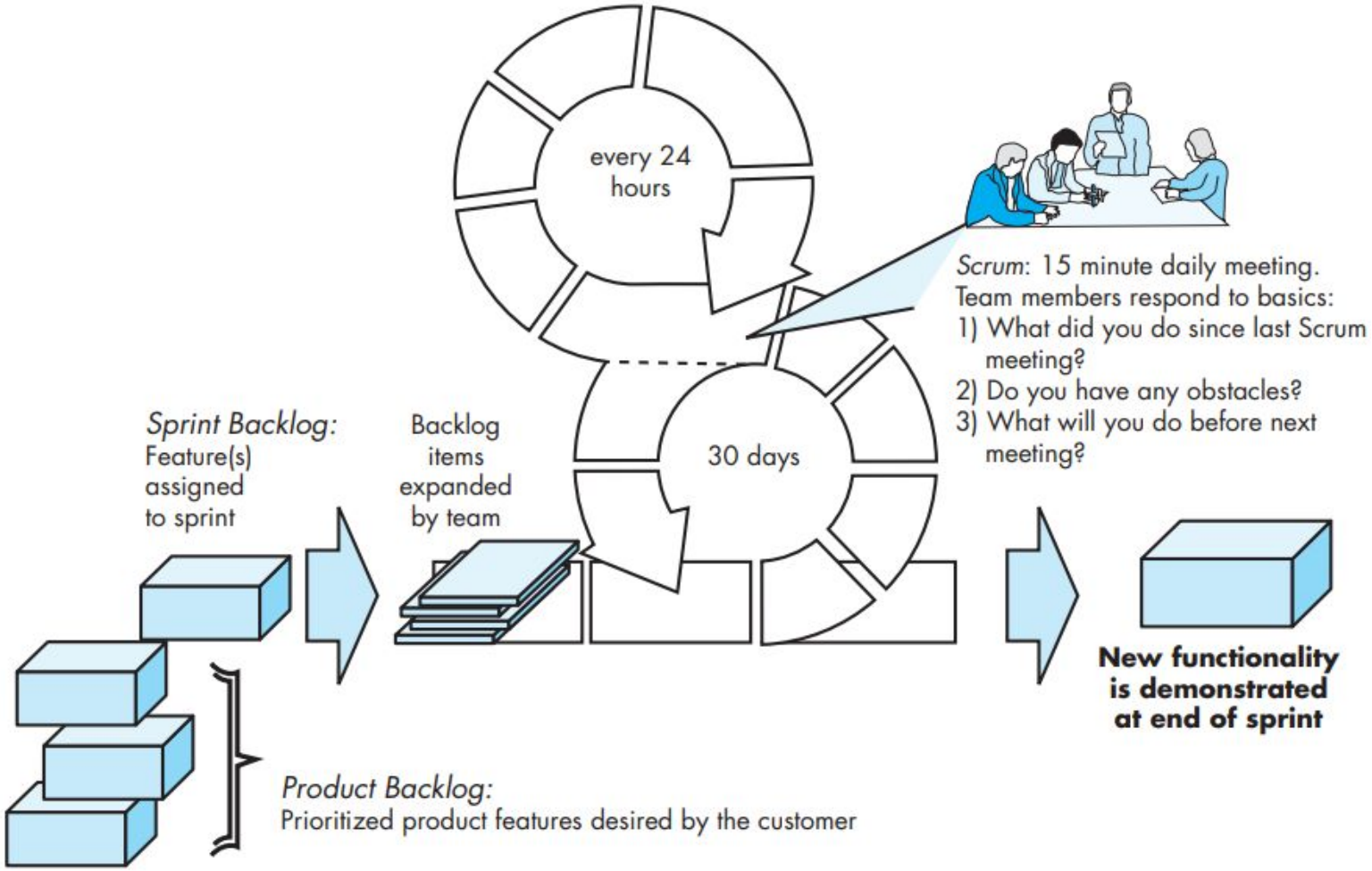
Cont.

- Scrum meetings—are short (typically 15-minute) meetings held daily by the Scrum team.
  - Three key questions are asked and answered by all team members:
    - What did you do since the last team meeting?
    - What obstacles are you encountering?
    - What do you plan to accomplish by the next team meeting?
  - A team leader, called a Scrum master, leads the meeting and assesses the responses from each person.
  - The Scrum meeting helps the team to uncover potential problems as early as possible.

Cont.

- Demos—deliver the software increment to the customer so that functionality that has been implemented can be demonstrated and evaluated by the customer

Scrum process flow



Task

- Advantages and disadvantages of XP
- Advantages and disadvantages of Scrum