

Announcements

- Accept the invitation from openAI by tonight so that I can assign you to groups and group credits.
- Extra Credit Opportunity in today's class
- Project 1 Planning assignment due next Thursday.

CS3300 A Introduction to Software Engineering

Lecture 03: SDLC; Life Cycle Models

Dr. Nimisha Roy ▶ nroy9@gatech.edu

Traditional Software Development Phases



Requirements
Engineering



Design



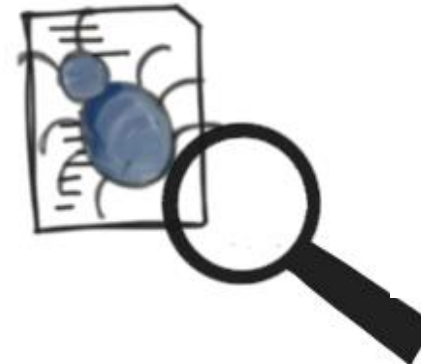
Implementation



Verification &
Validation



Maintenance



Software Development Phases: Semester Assignments



Requirements
Engineering

Project 1 Planning; Project 1 RE



Design

Project 1 Design



Implementation

Project 1 & 2 code, report, ppt



Verification & Validation

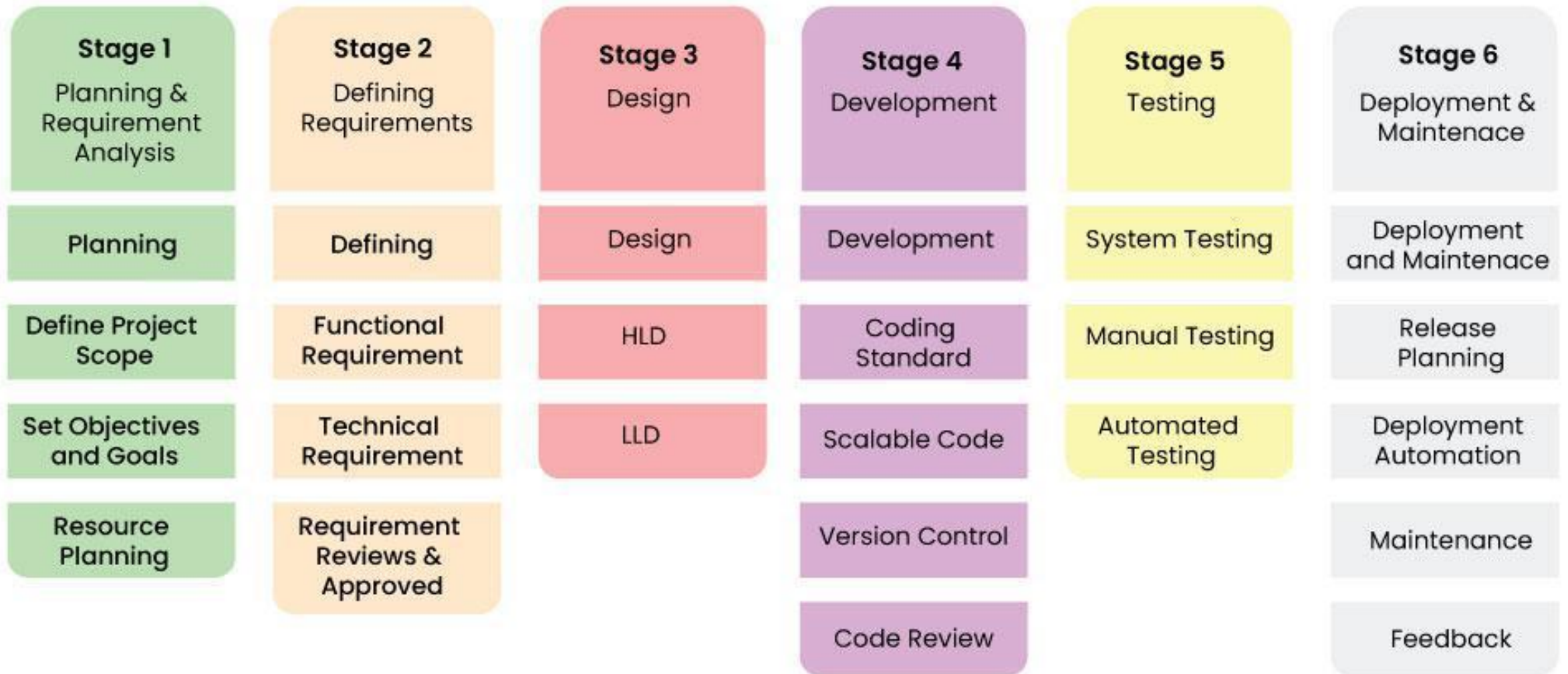
Project 2 Test



Maintenance



Software Development Life Cycle

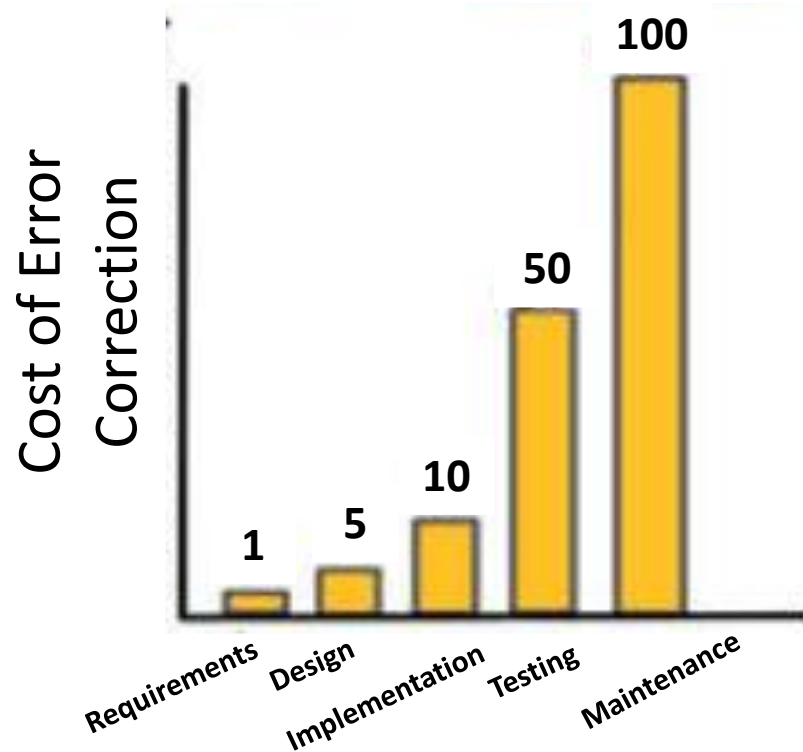


Requirements Engineering

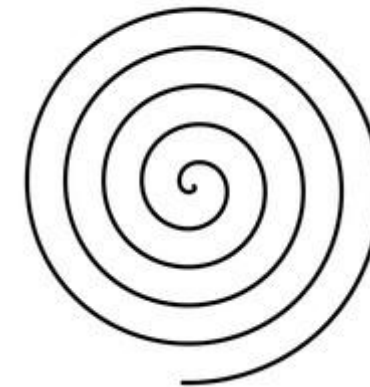


RE is the process of establishing the needs of stakeholders that are to be solved by software

Cost of Late Correction



Elicitation



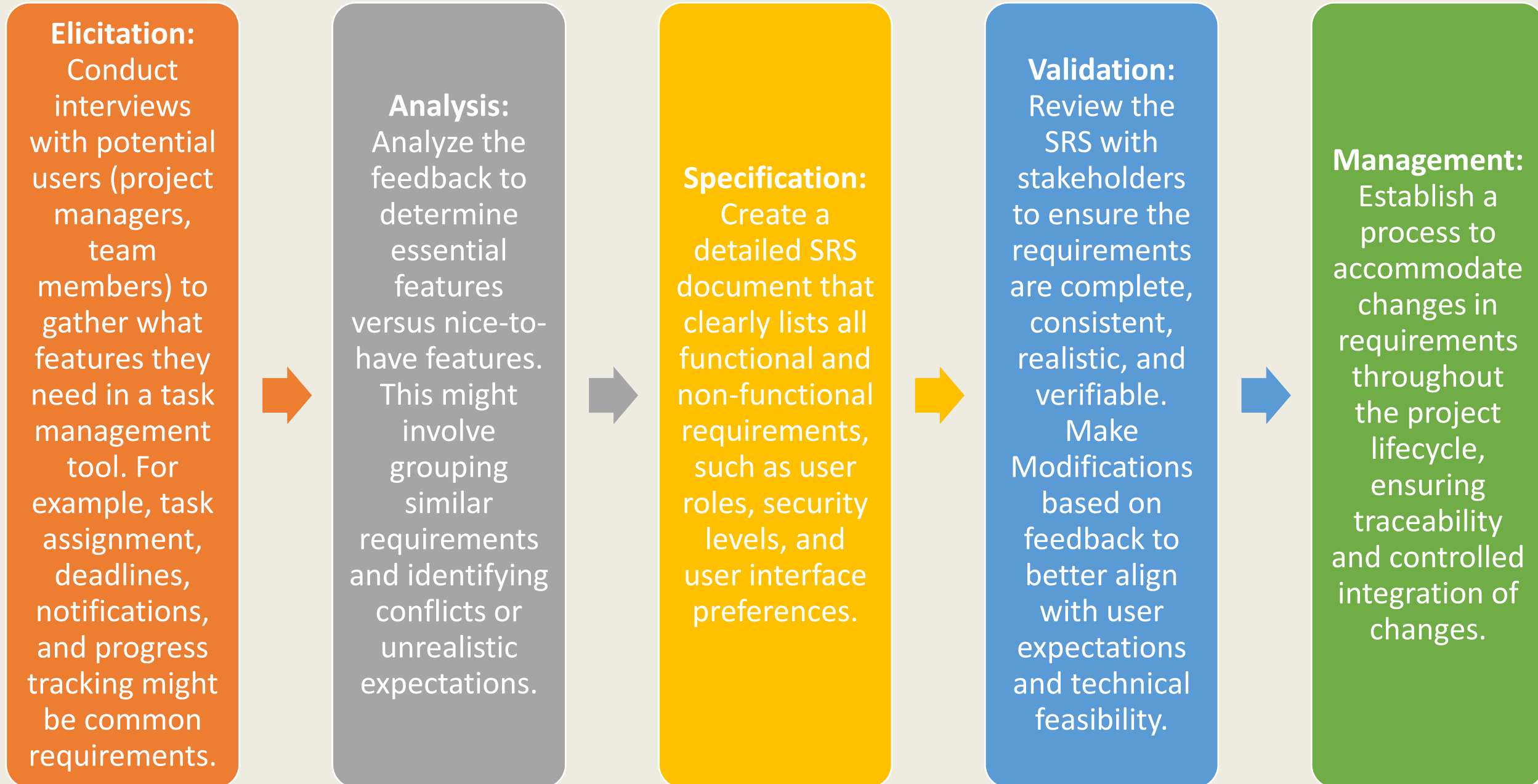
Management

Analysis

Validation

Specification

RE Example: Task Management Software



Design



SRS (Software Requirements Specification) is a reference for software designers to come up with the best design for the software.

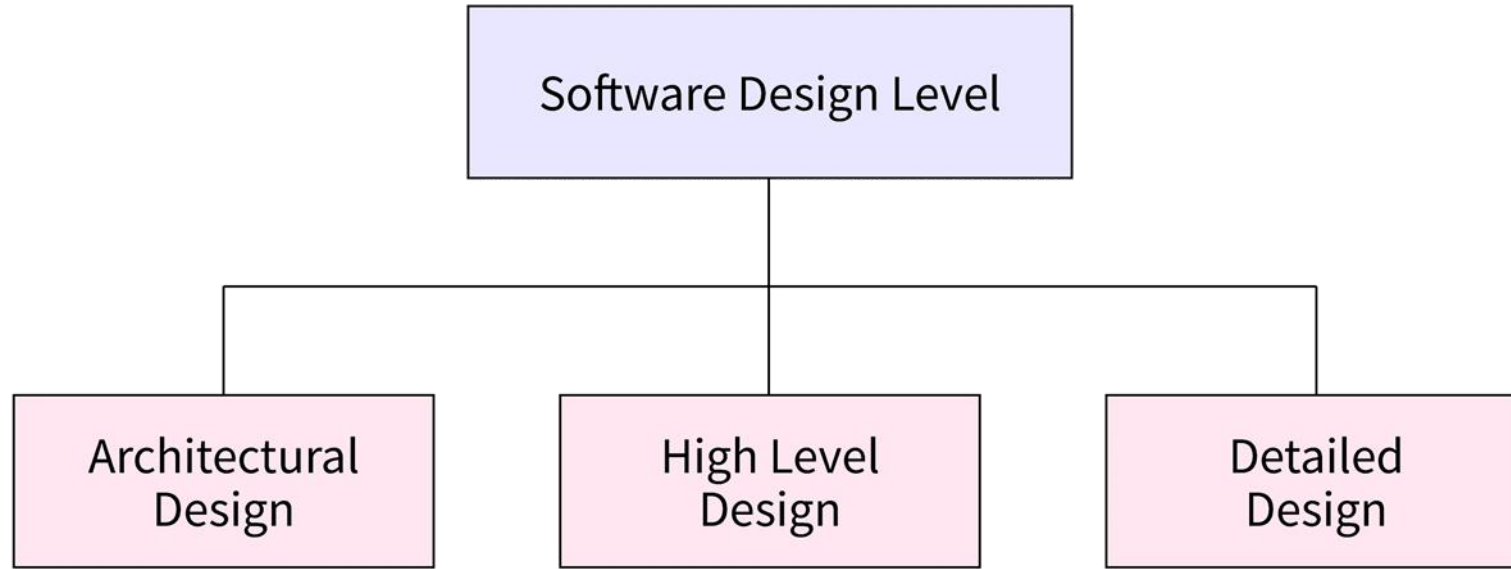


Multiple designs for the product architecture are present in the Design Document Specification (DDS).



This DDS is assessed by market analysts and stakeholders. After evaluating all the possible factors, the most practical and logical design is chosen for development.

Design



The **architectural design** characterizes the software as a system with numerous interconnected components. The designers acquire an overview of the proposed solution domain at this level.

The **high-level design** deconstructs the architectural design's 'single entity-multiple component' notion into a less abstract perspective of subsystems and modules, depicting their interaction with one another

Each module is extensively investigated at this level of software design to establish the data structures and algorithms to be used. The outcome of all stages is documented in DDS. It defines the logical structure of each module as well as its interfaces with other modules.

RE and Design Example: Task Management Software

Architectural Design:

Define the overall structure of the system. For this task management software, you might decide on a web-based architecture with client-server model where the server handles logic and database interactions, and the client provides interactive user interfaces.



High-Level Design:

Break down the architecture into major components or modules such as User Management, Task Management, Notification System, and Database.

Define the relationships and data flow between these modules.

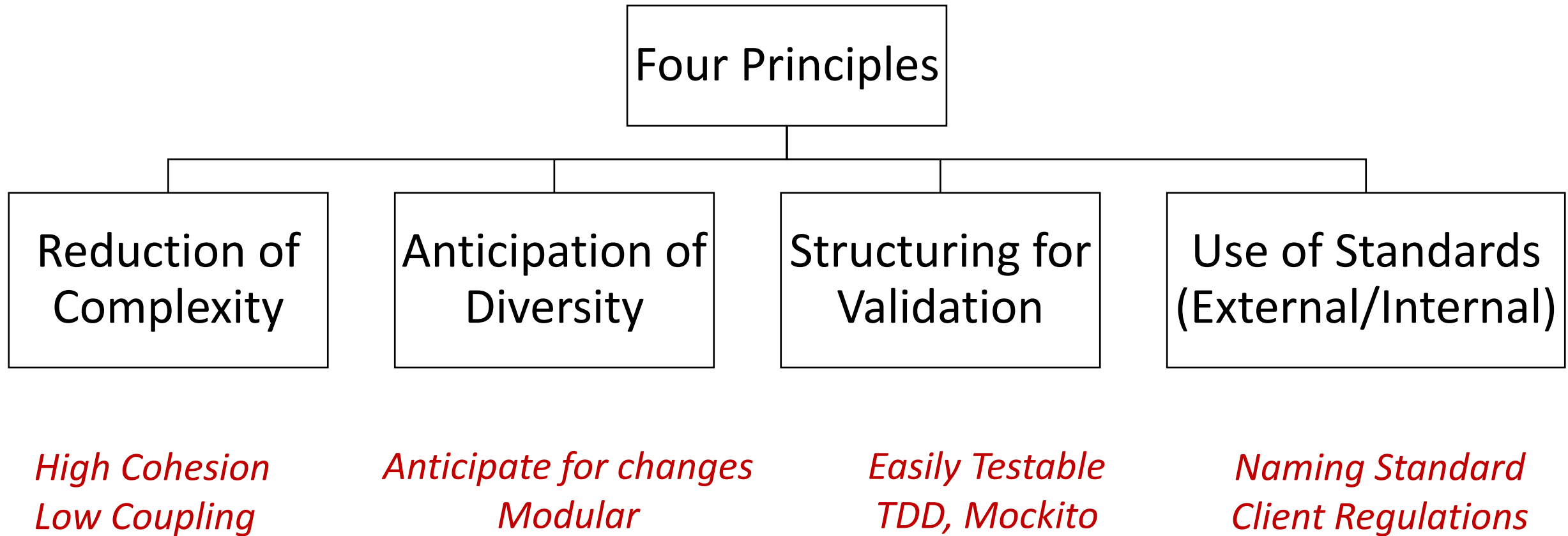


Detailed Design: Focus on the specifics of each module. For instance, the Task Management module might involve detailed designs of the database schema for tasks, classes, and methods to handle task creation, updates, and queries. Interfaces for each module should also be defined to ensure they can interact seamlessly.

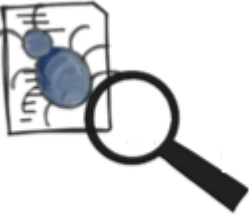
Implementation



Phase where we take care of realizing the design of the system and create a natural softer system



Verification & Validation

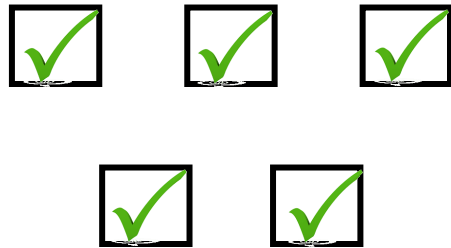


Phase that aims to check that software system meets its specifications and fulfils its intended purpose

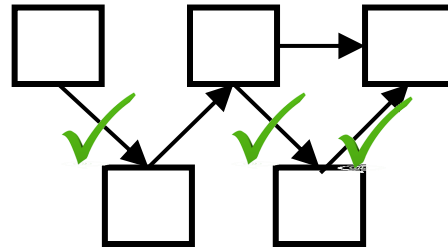
Verification: did we build the system right?

Validation: did we build the right system?

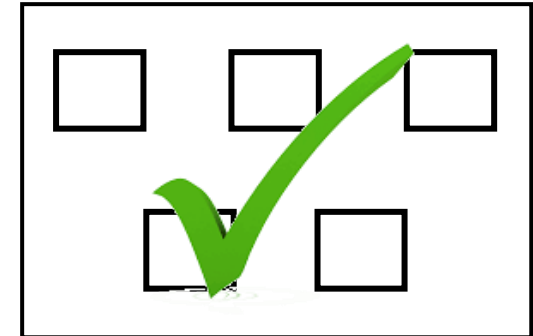
Unit



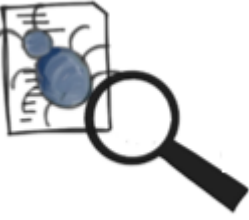
Integration



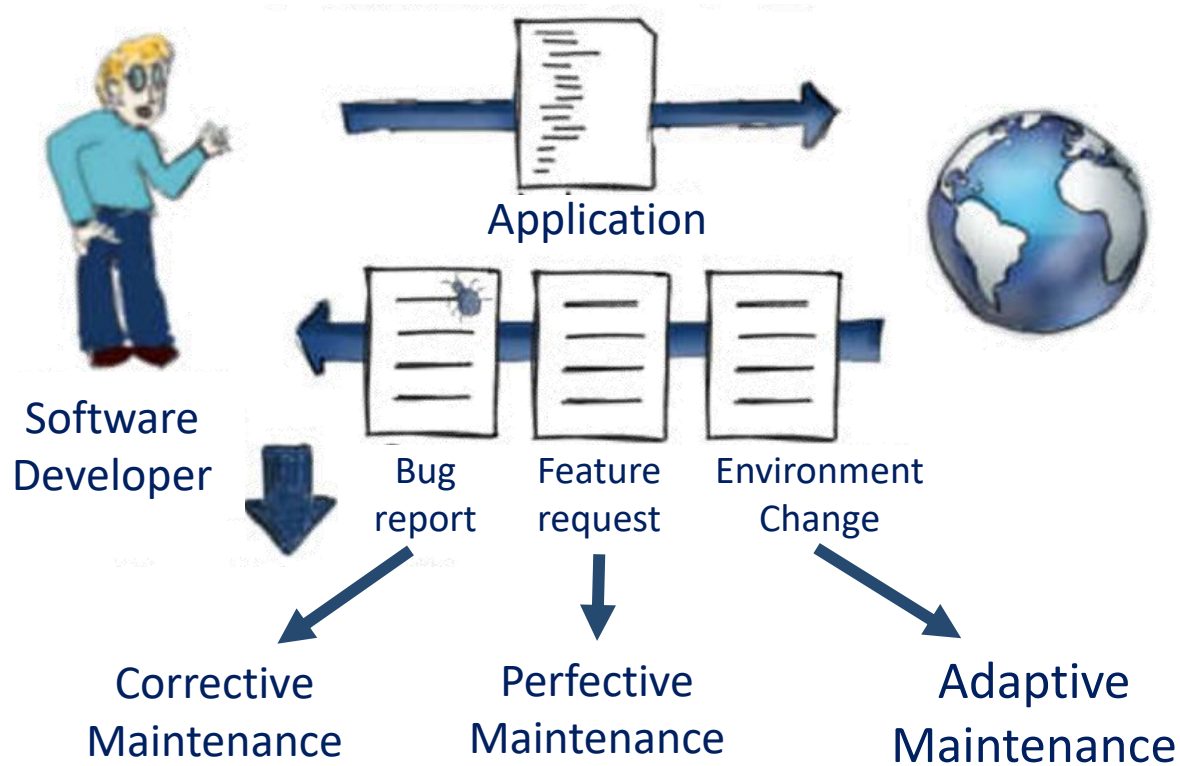
System



Maintenance

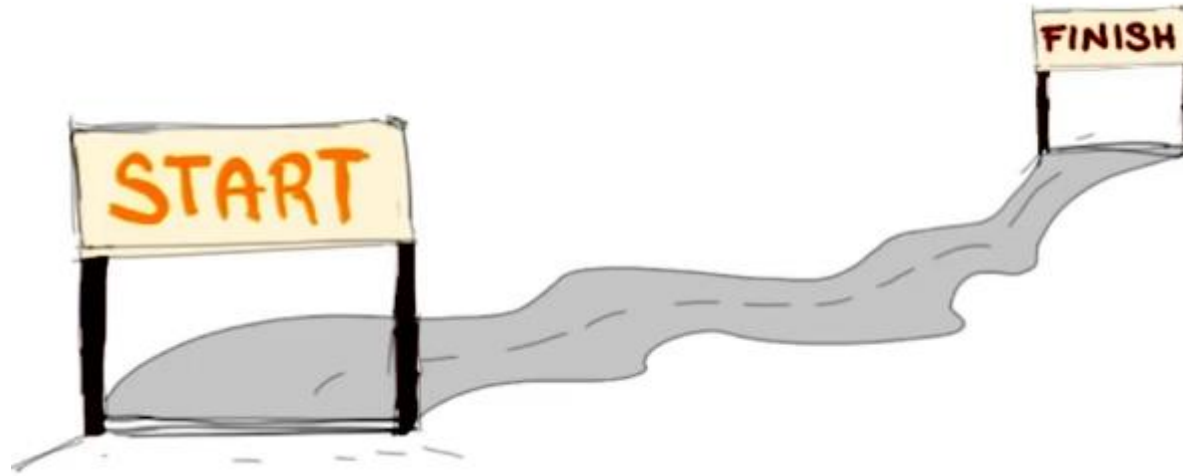


Once Software released to final users and in operation, many things can happen: environment change -new libraries, new systems, additional functionality requests, bug reports



- Maintenance is a fundamental and expensive phase
- *Regression testing* – retesting a modified version of software before release, no introduction of new errors

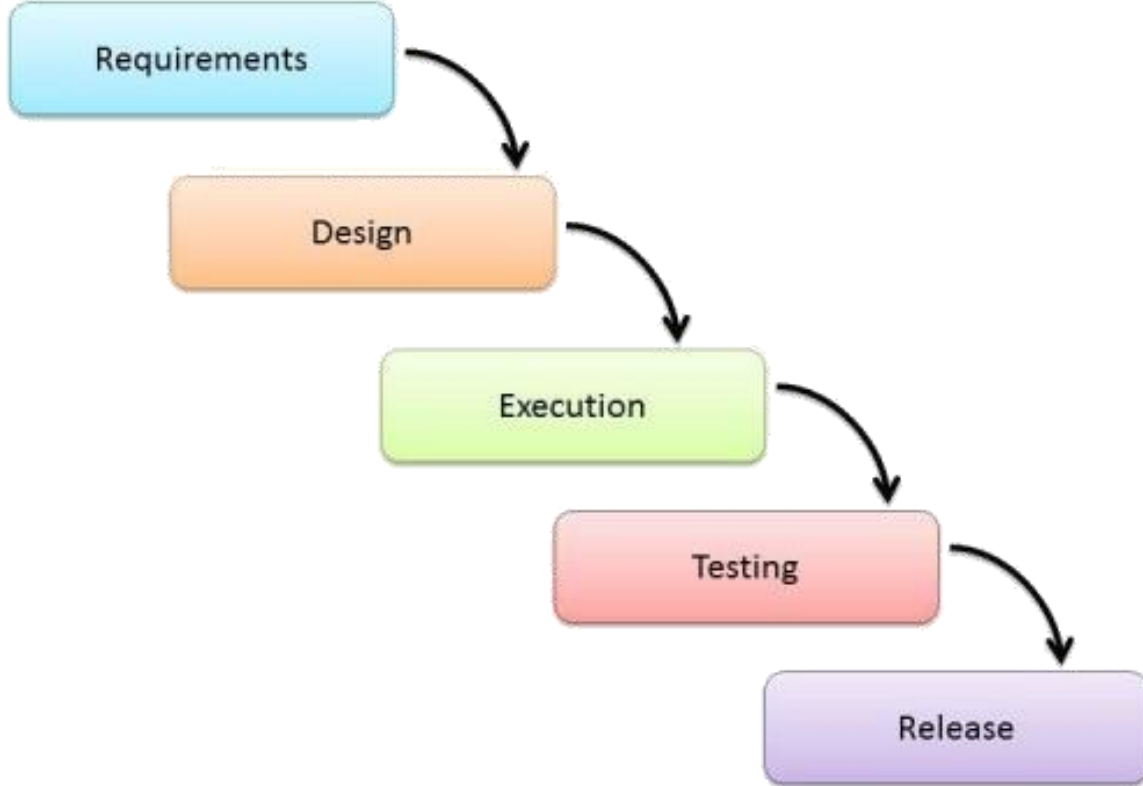
Software Process Model/ Life Cycle Model



Functions:

- Order of activities
- Transition Criteria between Activities
- What should we do next and for how long?

Waterfall Method



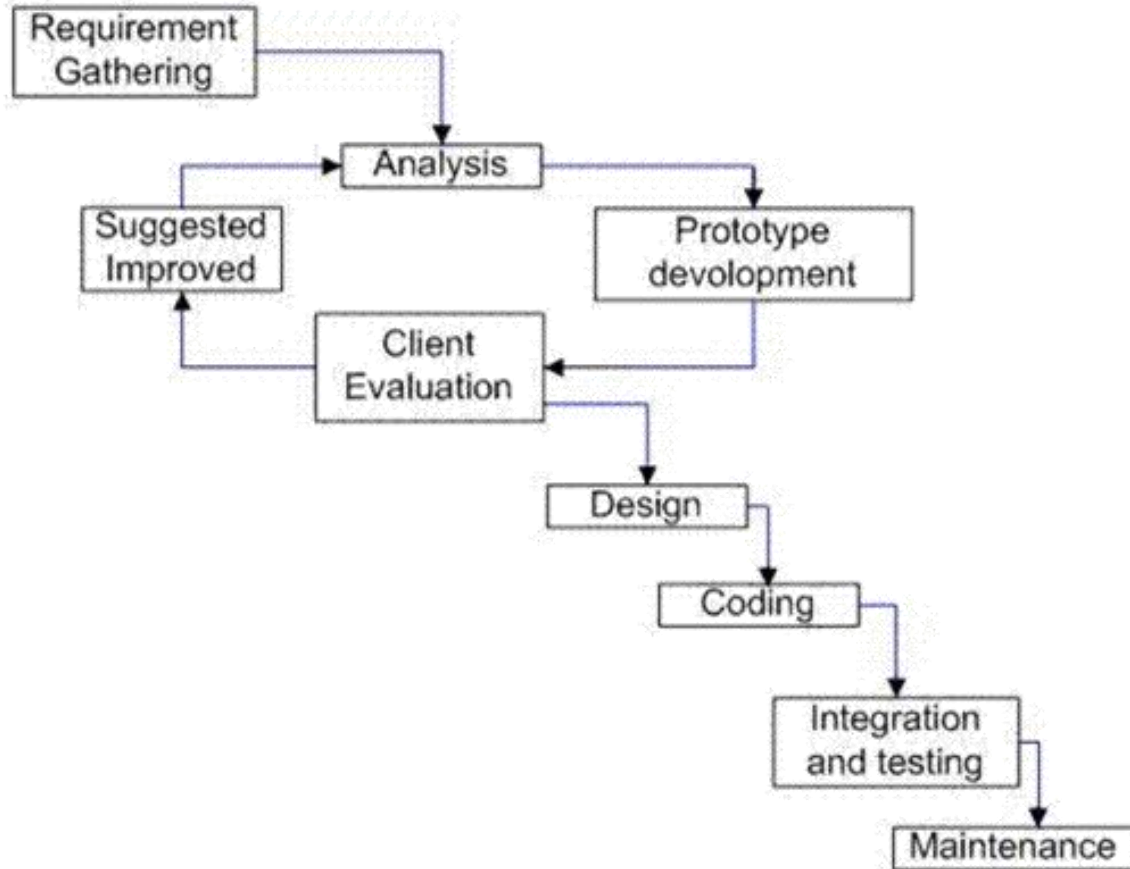
Early Error Detection



No Flexibility

- Project progresses in an orderly sequence of steps
- Pure Waterfall model performs well for software products with a stable product definition- well known domain, technologies involved, Request for Proposals (RFP)
- Waterfall method finds errors in early local stages
- Not flexible- not for projects where requirements change, developers not domain experts, or technology used are new and evolving

Evolutionary Prototyping



Immediate feedback
Helps Requirements understanding



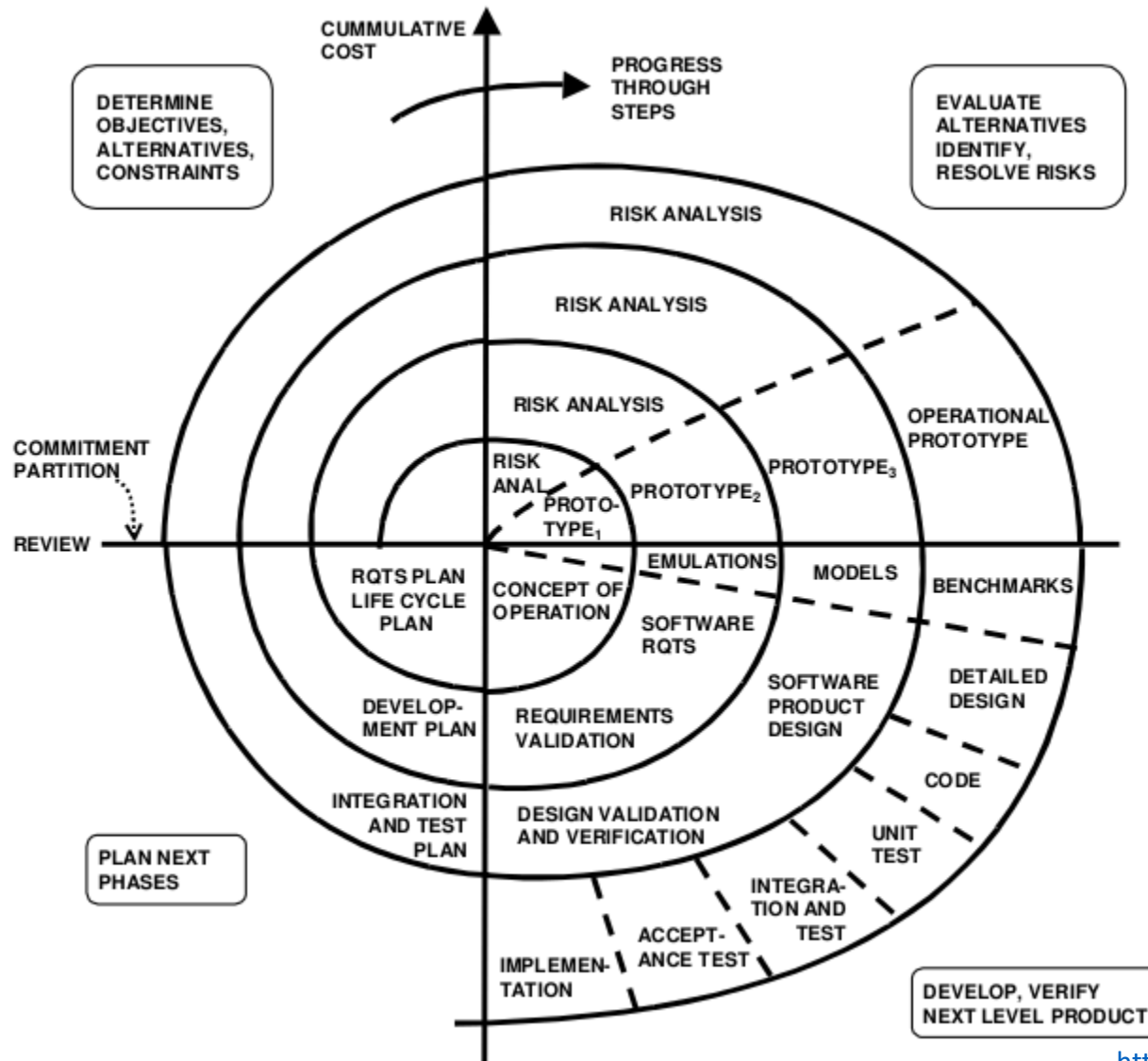
Difficult to Plan
Can deteriorate to code-and-fix

- Prototypes that evolve into the final system through an iterative incorporation of user feedback.
- Ideal when not all requirements are well-understood. System keeps evolving based on customer feedback

Spiral Method



Incremental risk-oriented lifecycle model with 4 main phases



Risk Reduction

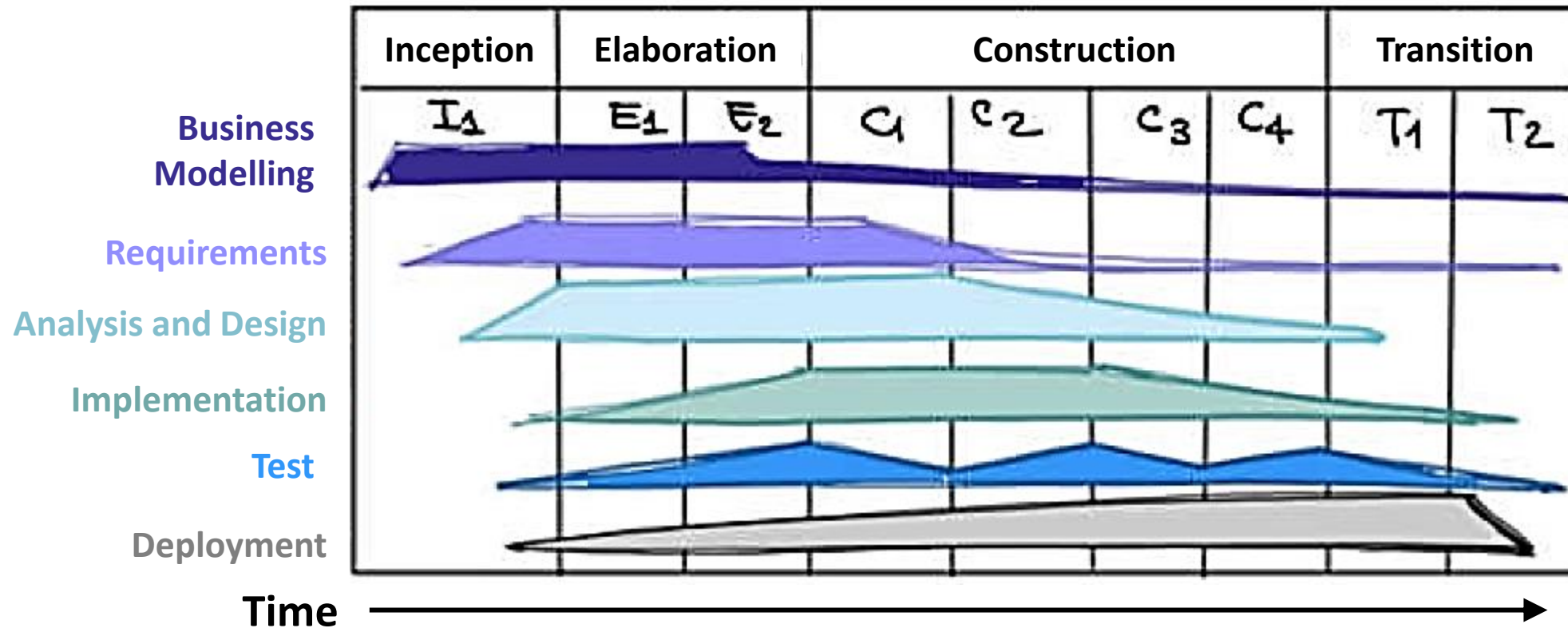
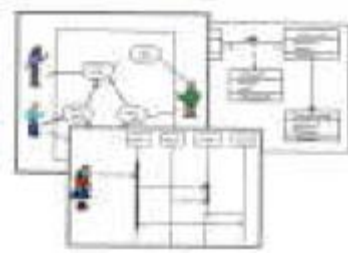
Functionality can be added
Software produced early, Early feedback



Specific Expertise

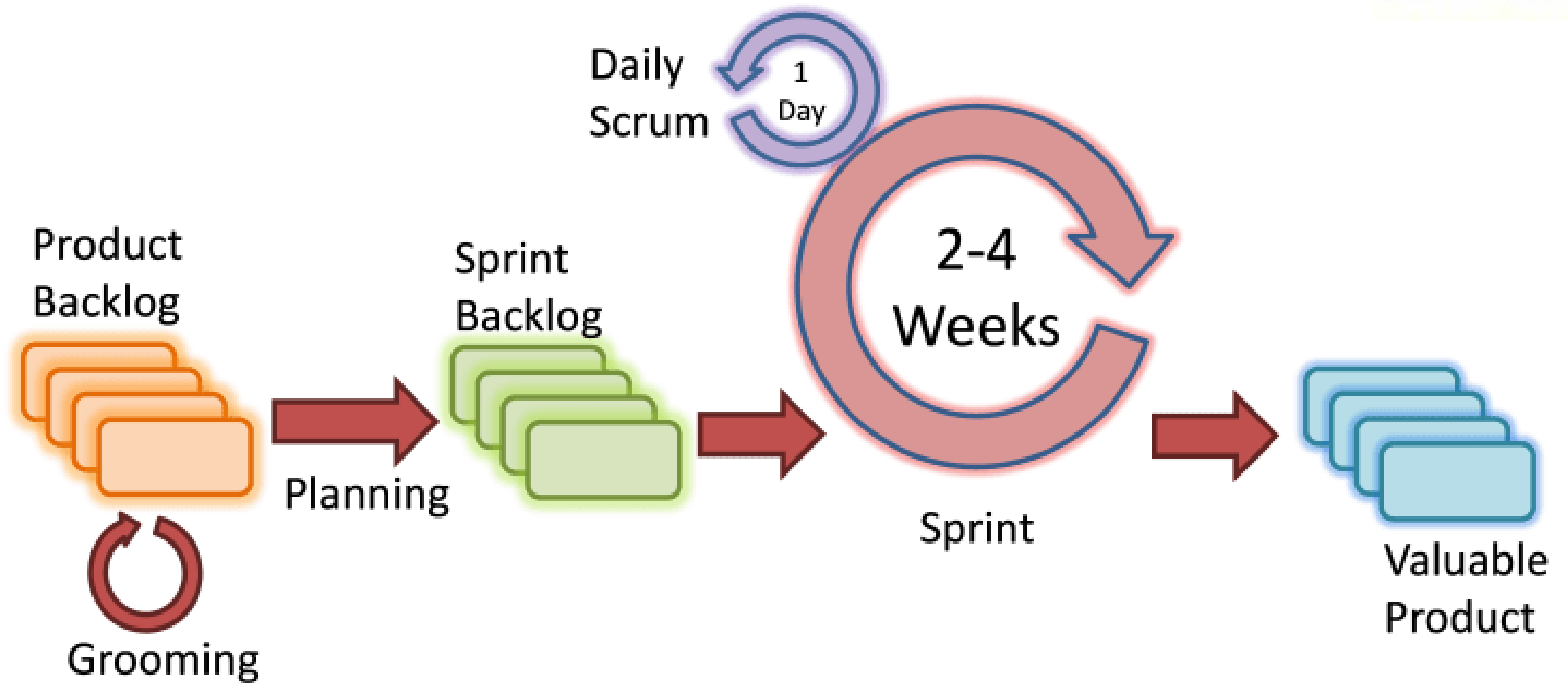
Highly dependent on risk analysis
Complex, Costly

Rational Unified Process (RUP)



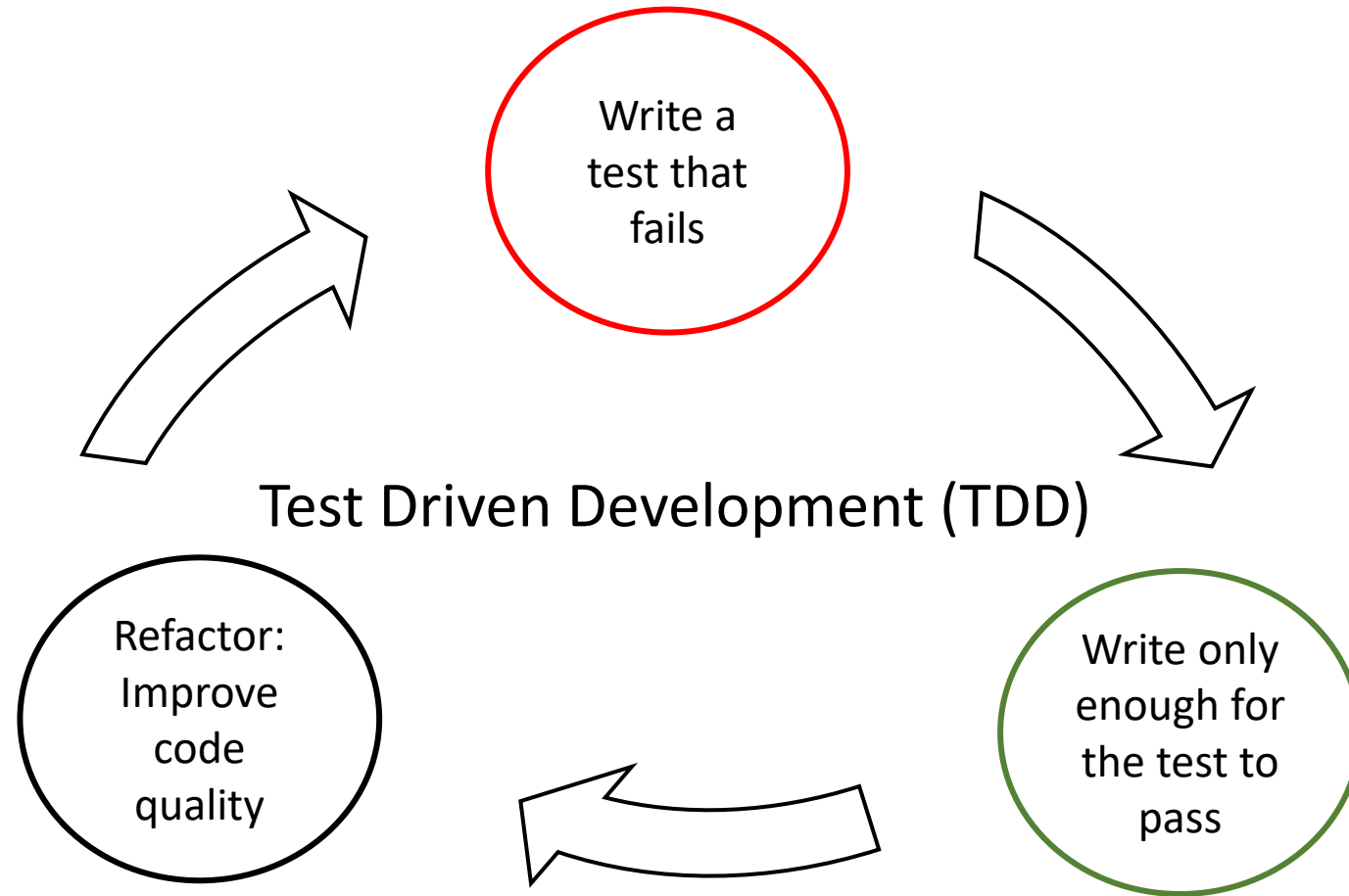
- Popular Process based on UML. Works iteratively, performs 4 phases in each iteration
- Inception phase: Scope the system - Scope of project, domain, initial cost, budget estimates
- Elaboration phase: domain analysis and basic architecture
- Construction phase: Bulk of development
- Transition: From development to production, available to users

Agile - Scrum



Highly iterative and incremental development process

Agile - XP



Highly iterative and incremental development process

Other Agile Methodologies

Kanban: Simplest in IT World;
May Pose time related problems



Some industry-based examples

Waterfall

Military And Aircraft Programs Where Requirements Are Declared Early On And Remain Constant



Evolutionary Prototyping

- **Company:** Broderbund Software.
- **Project:** The creation of the original "Prince of Persia" video game. The initial version of the game was created and then improved upon based on feedback and playtesting.



Some industry-based examples

Spiral

- [NASA's space shuttle program](#) in the 1970s
- [Gantt Chart Software](#) – GanttPRO



Agile

- **Apple, IBM, Microsoft, and Procter & Gamble**
- **Cisco:** defects were reduced by 40% when compared to waterfall
- **Barclays:** 300% increase in throughput
- **Panera Bread:** 25% increase in company sales
- **PlayStation Network:** Saved the company \$30 million a year

Choosing the right Software Process Model



Requirements
Understanding



Expected
Lifetime



Risk



Schedule Constraints



Interaction with
Management/Customers



Expertise

As much influence over a project's success as any other major planning decision

Industry Standards: Factors affecting choice of project LCM

Degree of
Project
Complexity

Work/Time
Flexibility

Project Focus/
Client
involvement

Size of
organization

Role
Specialization

Budget










<https://asana.com/resources/project-management-methodologies>

<https://thedigitalprojectmanager.com/projects/pm-methodology/project-management-methodologies-made-simple/>

Industry Standards: Factors affecting choice of project LCM

Factors	Waterfall	Evolutionary Prototyping	Agile Methodologies	Spiral
Unclear User Requirements	Poor	Good	Excellent	Excellent
Unfamiliar Technology	Poor	Excellent	Poor	Excellent
Complex System	Good	Excellent	Poor	Excellent
Reliable System	Good	Poor	Good	Excellent
Short time schedule	Poor	Good	Excellent	Excellent
Strong Project Management	Excellent	Excellent	Excellent	Excellent
Cost Limitation	Poor	Poor	Excellent	Poor
Visibility of stakeholder	Good	Excellent	Excellent	Excellent
Skills Limitation	Good	Poor	Poor	Poor
Documentation	Excellent	Good	Poor	Good
Component Reusability	Excellent	Poor	Poor	Poor

Industry Standards: Most Popular Methods

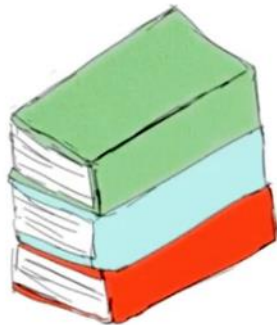
-  **Agile** – collaborating to iteratively deliver whatever works
-  **Scrum** – enabling a small, cross-functional, self-managing team to deliver fast
-  **Kanban** – improving speed and quality of delivery by increasing visibility of work in progress and limiting multi-tasking
-  **Scrumban** – limiting work in progress like Kanban, with a daily stand up like Scrum
-  **Lean** – streamlining and eliminating waste to deliver more with less
-  **eXtreme Programming (XP)** – doing development robustly to ensure quality
-  **Waterfall** – planning projects fully, then executing through phases
-  **PRINCE2** – controlled project management that leaves nothing to chance
-  **PMI's PMBOK** – applying universal standards to Waterfall project management

Lifecycle Documents

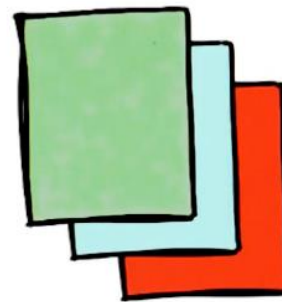
Documenting the activities carried out during the different phases of the lifecycle is a very important task.

Can be used for different purposes like:

- Communicate details of the software systems to different stakeholders
- Ensure the correct implementation of the system
- Facilitate maintenance and so on.



IEEE Documents



Light-weight Documents

Classic Mistakes : People



Heroics



Work Environment



People Management

Classic Mistakes : Process



Schedule Issues



Planning Issues

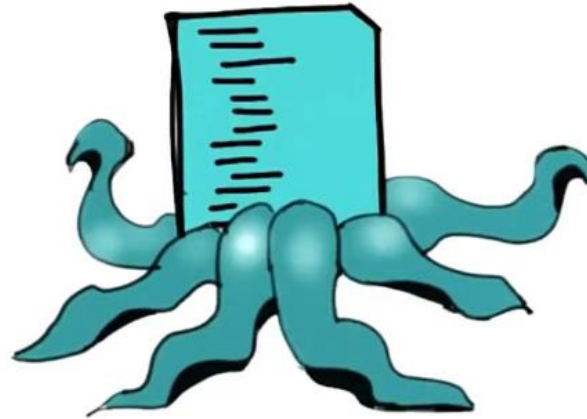


Failure

Classic Mistakes : Product



Gold Plating of Requirements



Feature Creep

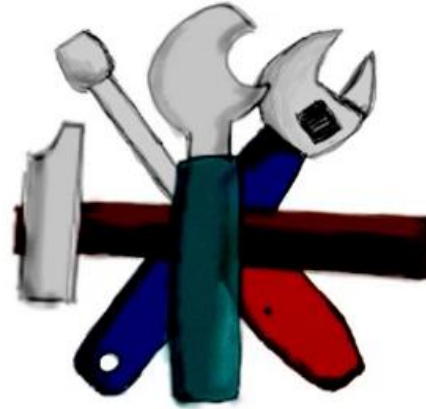


Research \neq Development

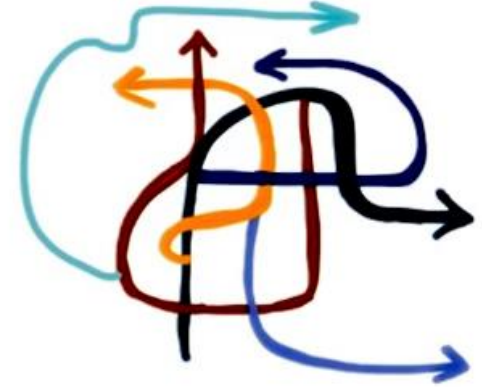
Classic Mistakes : Technology



Silver-Bullet Syndrome



Switching Tools



No version control

Quizizz