CS3300 Introduction to Software Engineering

Lecture 06: Life Cycle Models

Nimisha Roy ▸ nroy9@gatech.edu

Slides adapted from Alessandro Orso

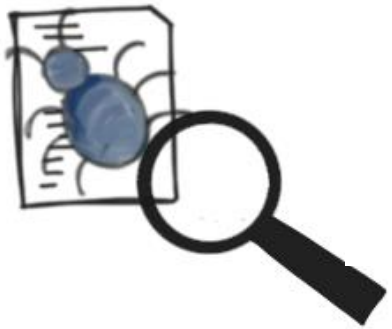
Traditional Software Phases

Requirements Engineering

Design

Implementation

Verification & Validation

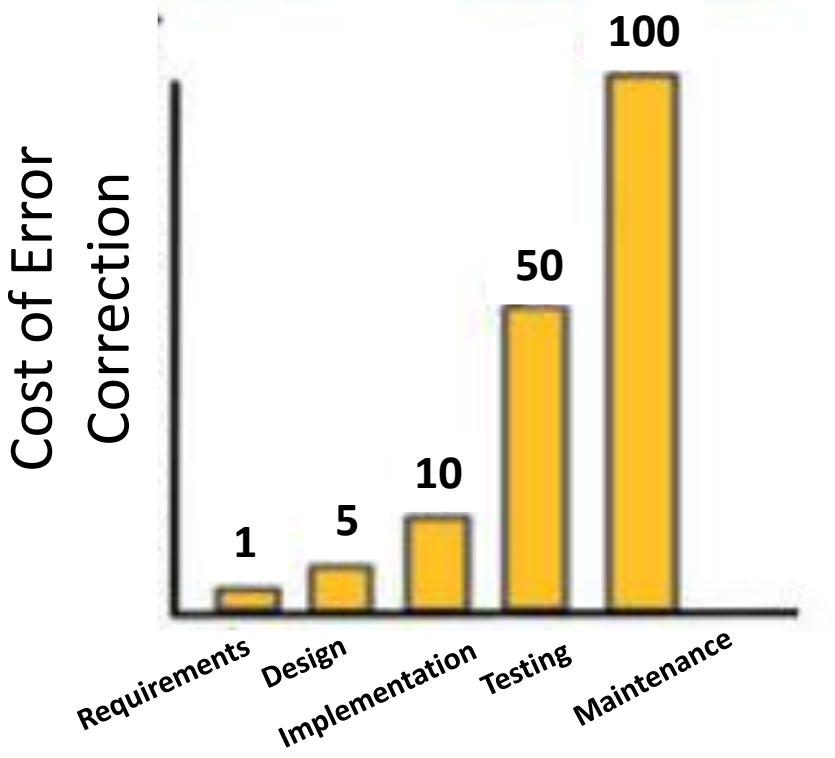


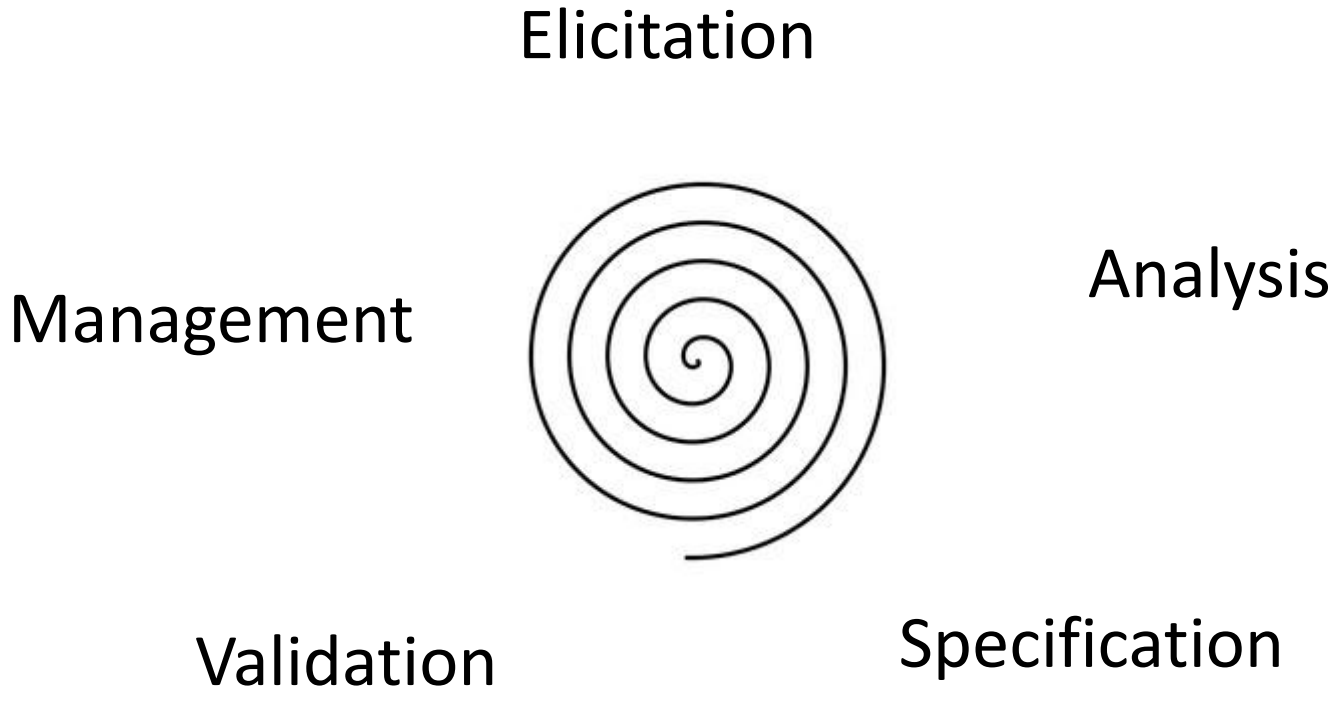
Maintenance

Requirements Engineering

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

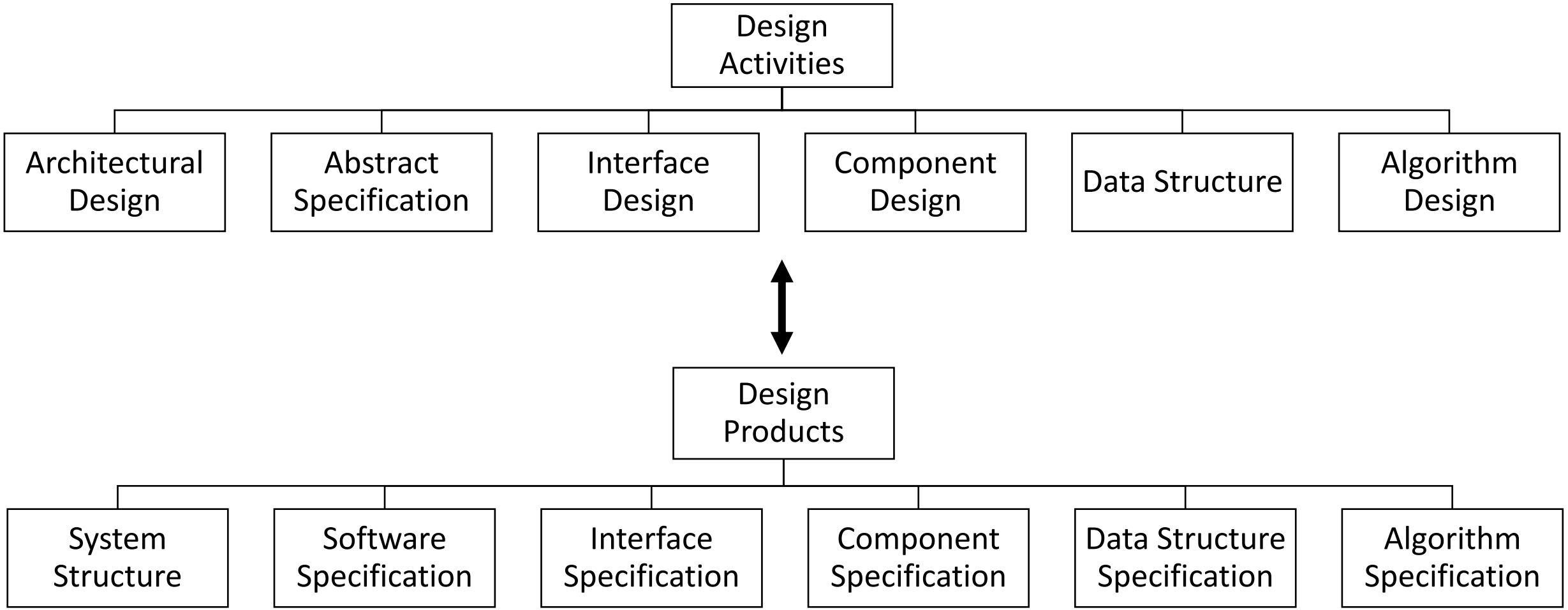
Cost of Late Correction





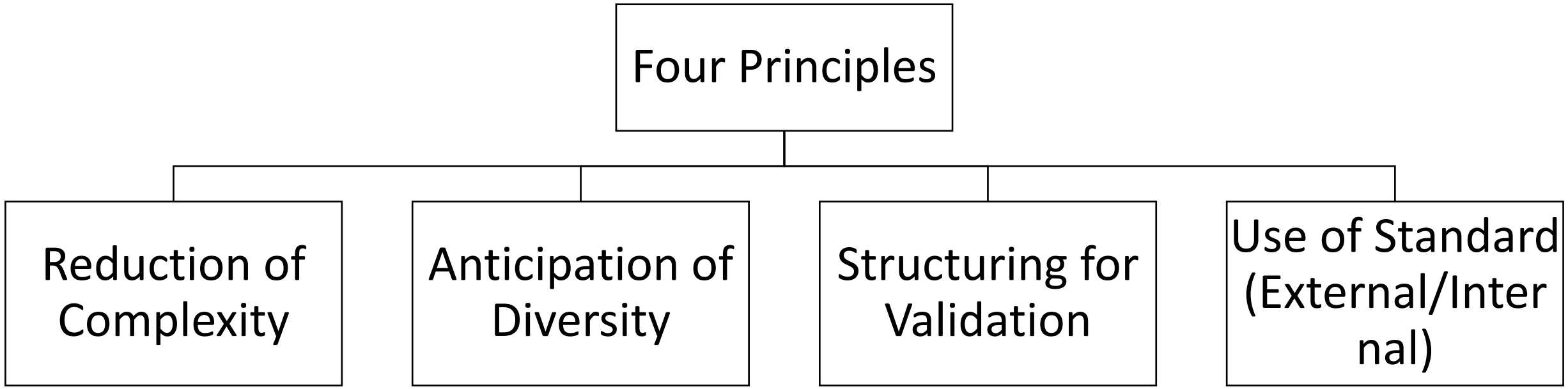
Design

Phase where requirements are analyzed in order to produce a description of the internal structure and organization of the system. Basis for construction of the actual system



Implementation

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

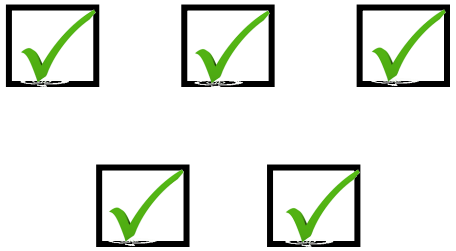


Verification & Validation

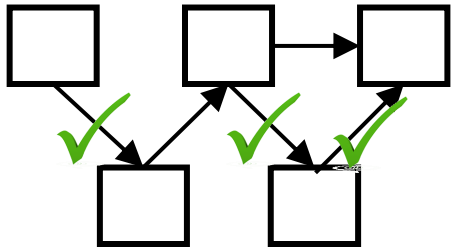
Phase that aims to check that software system meets its specifications and fulfils its intended purpose

Verification: did we build the system right?
Validation: did we build the right system?

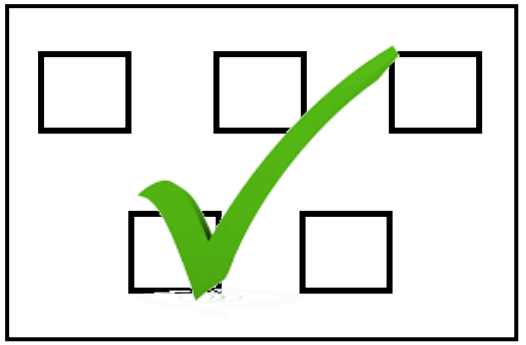
Unit



Integration

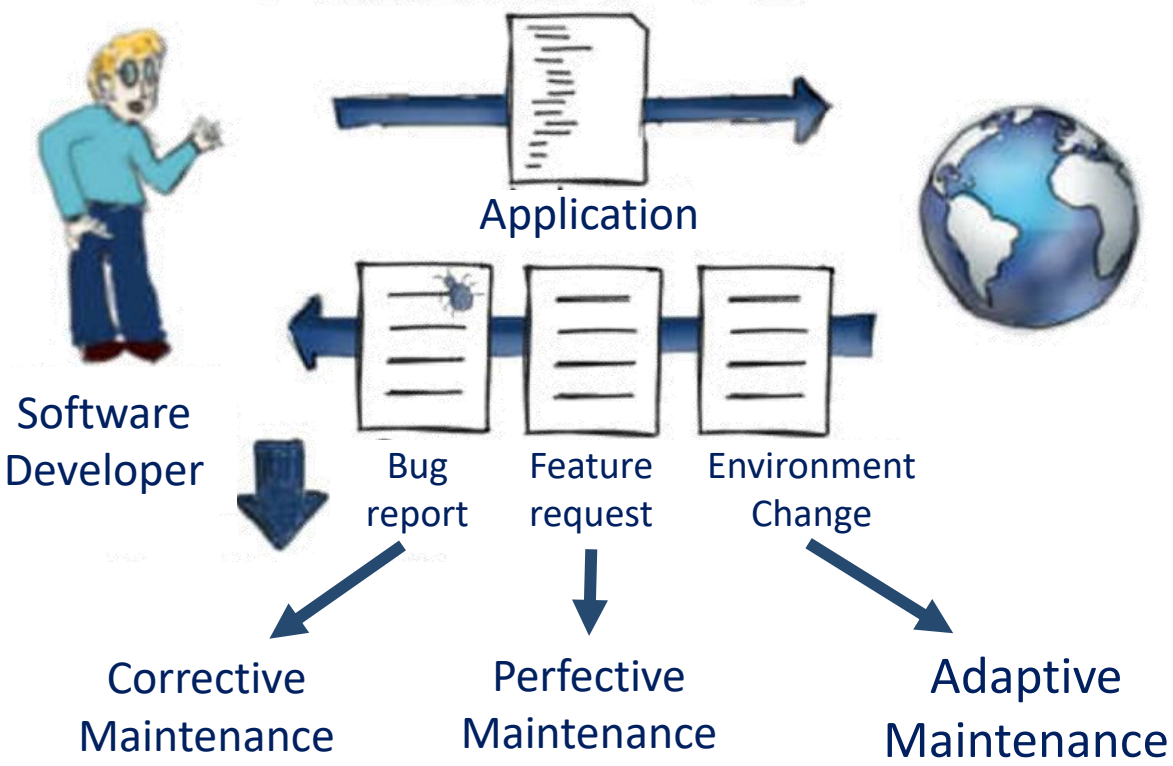


System



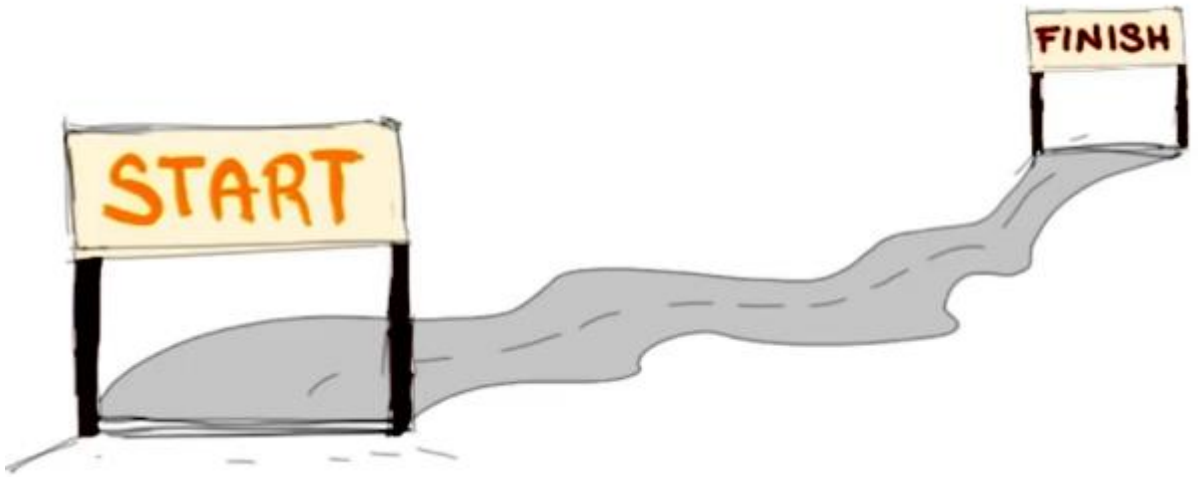
Maintenance

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

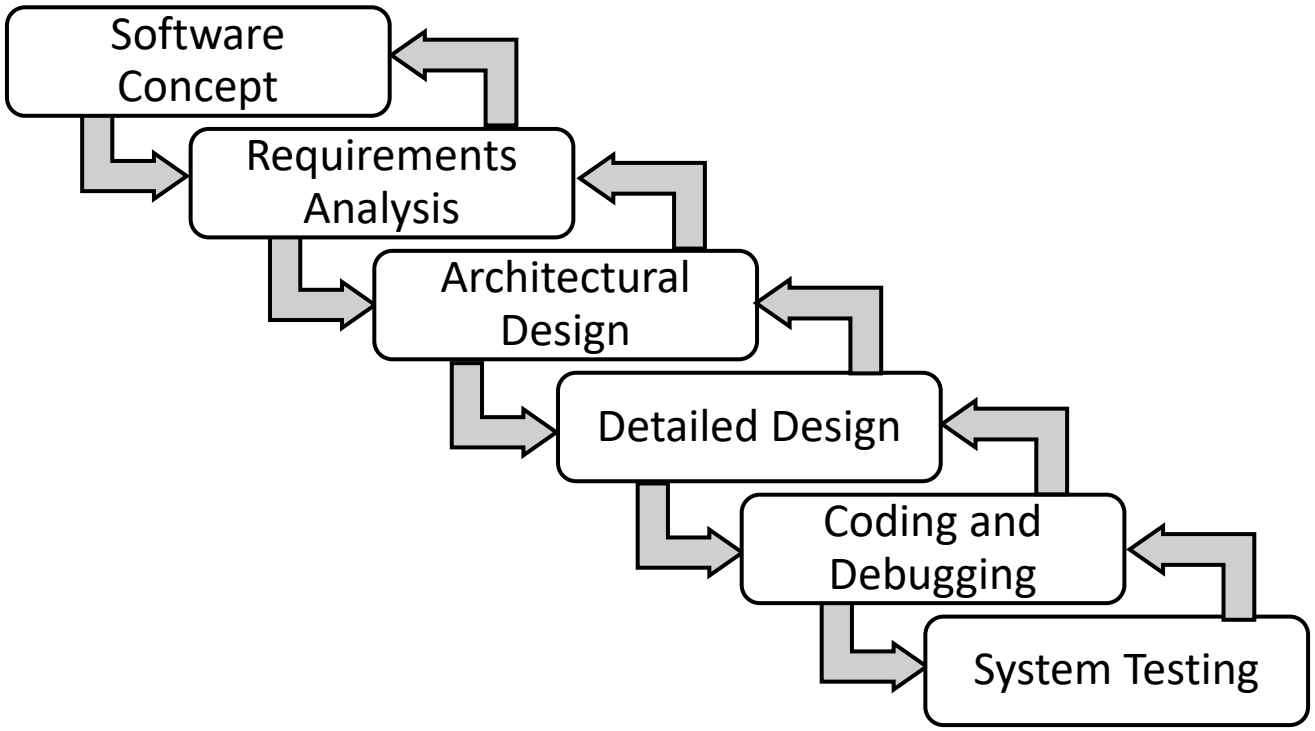
Software Process Model/ Life Cycle Model



Functions:

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

Waterfall Method



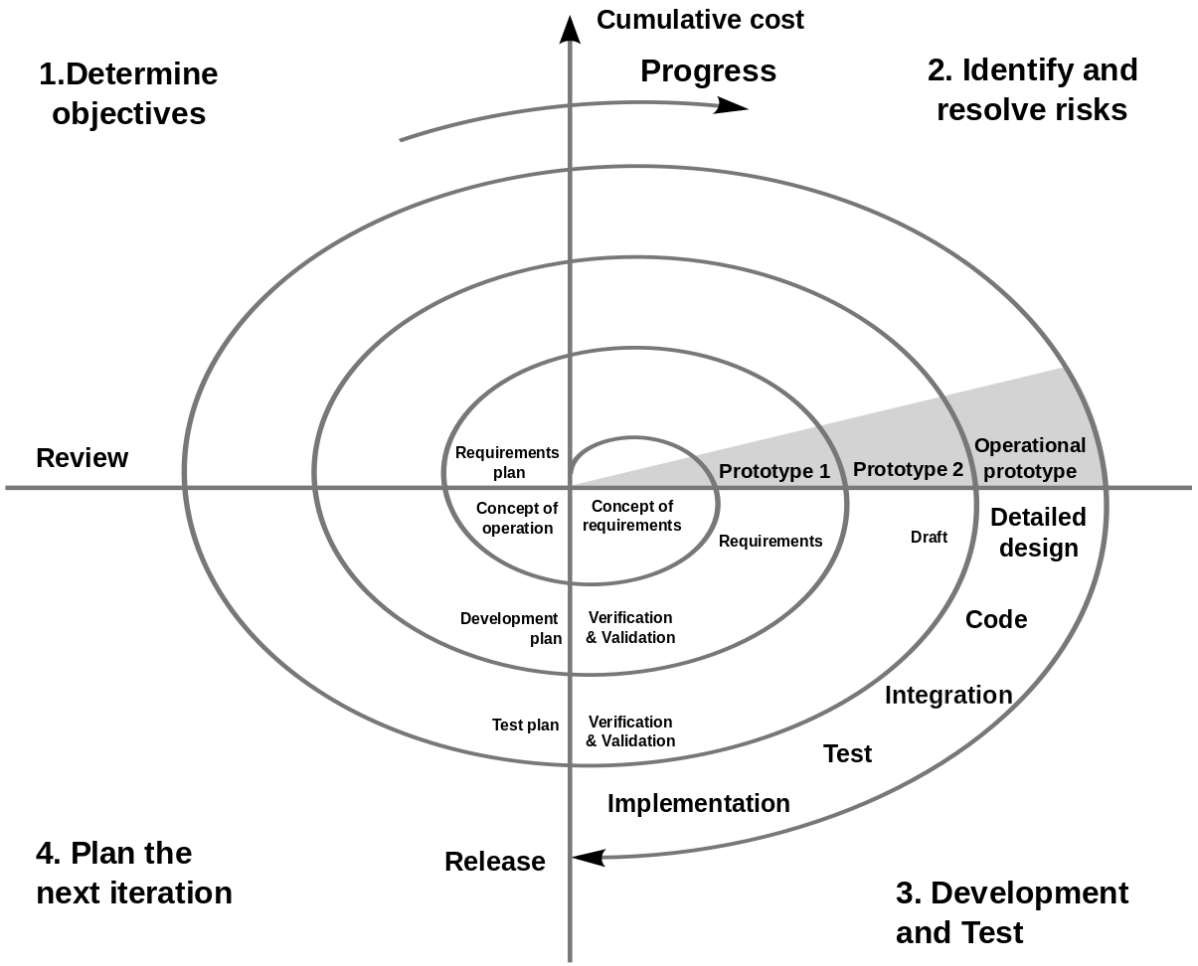
Early Error Detection

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
- 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

Spiral Method

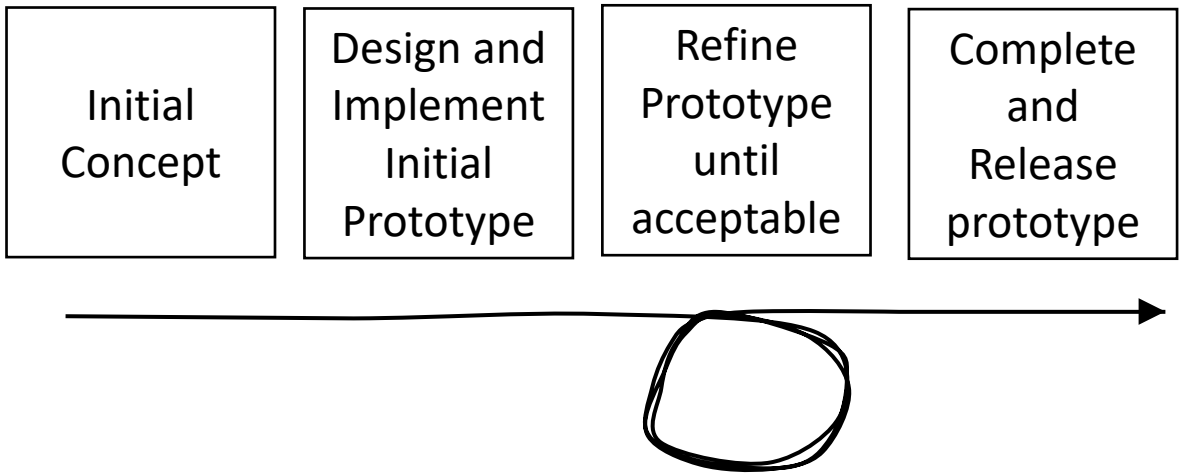
Incremental risk-oriented lifecycle model with 4 main phases



Risk Reduction
Functionality can be added
Software produced early, Early feedback

Specific Expertise
Highly dependent on risk analysis
Complex, Costly

Evolutionary Prototyping

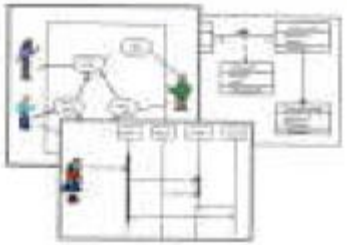


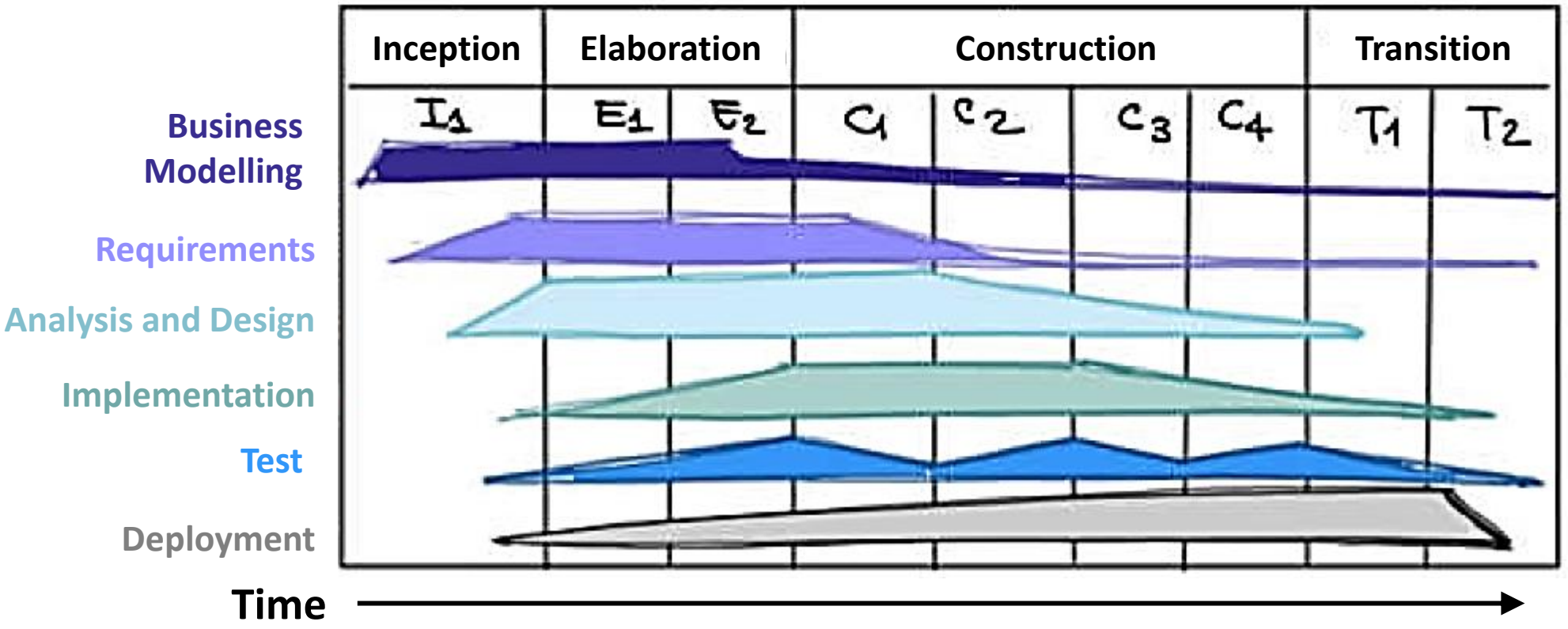
Immediate feedback
Helps Requirements understanding

Difficult to Plan
Can deteriorate to code-and-fix

- Ideal when not all requirements are well-understood. System keeps evolving based on customer feedback
- Developers start by developing the parts of the system that they understand, instead of developing a whole system.
- Partial system is then shown to the customer and the customer feedback is used to drive the next iteration, in which either changes are made to the current features or new features are added.
- Either the current prototype is improved, or the prototype is extended.

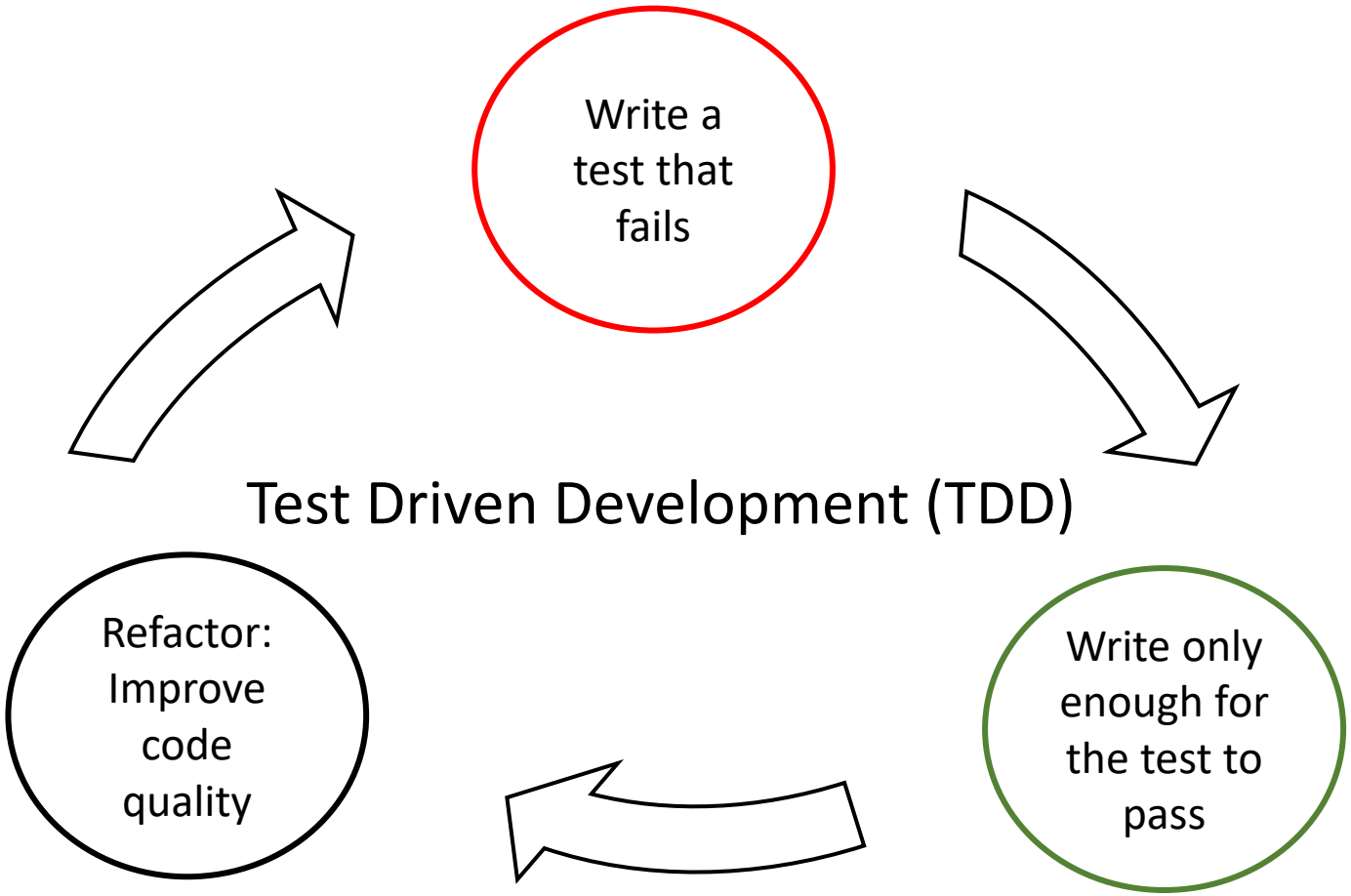
Rational Unified Process (RUP)





- 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



Highly iterative and incremental development process

Choosing the right Software Process Model

Requirements Understanding

Expected Lifetime

Risk

Schedule Constraints

Interaction with Management/Customers

Expertise

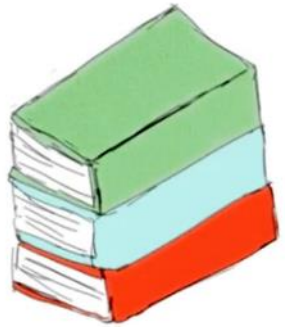
As much influence over a project's success as any other major planning decision

Lifecycle Documents

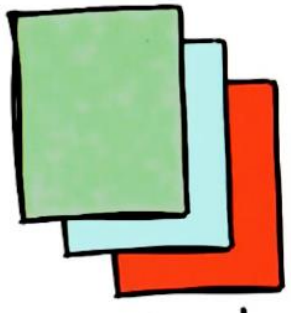
Documenting the activities carried out during the different phases of the lifecycle is a very important task.

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

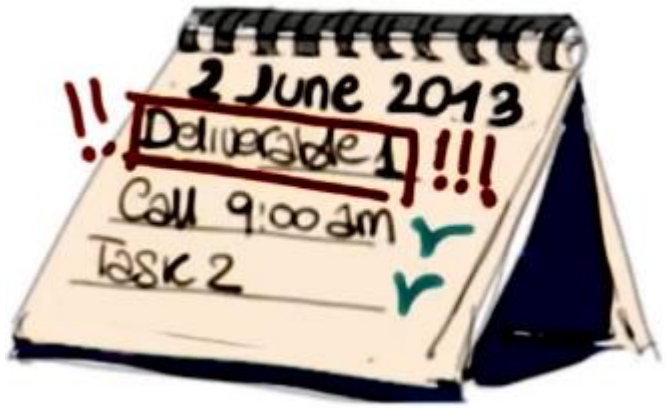
Classic Mistakes : People

Heroics

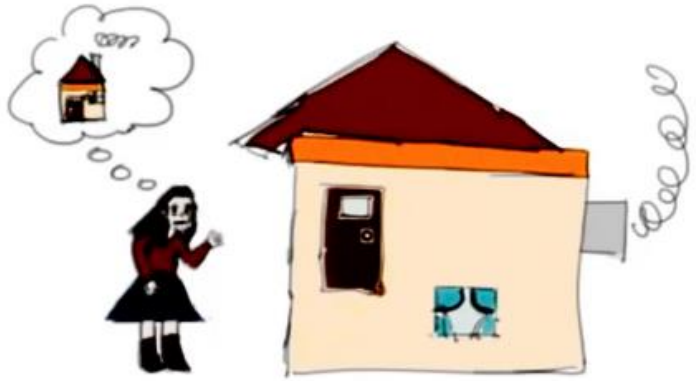
Work Environment

People Management

Classic Mistakes : Process



Schedule Issues

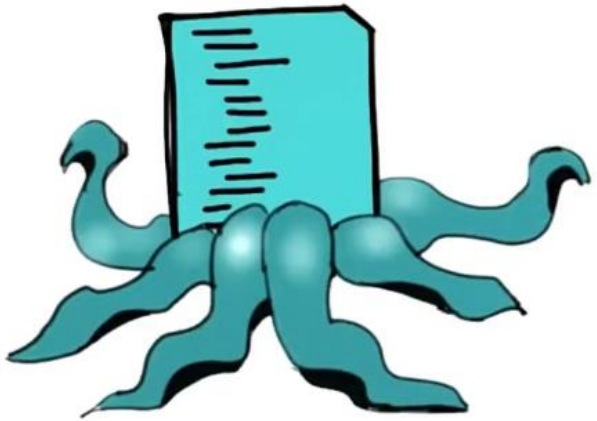


Planning Issues

Failure

Classic Mistakes : Product

Gold Plating of Requirements

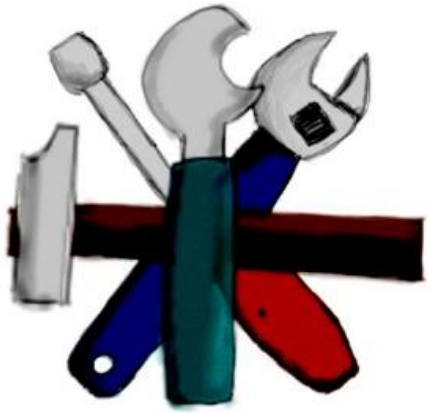


Feature Creep

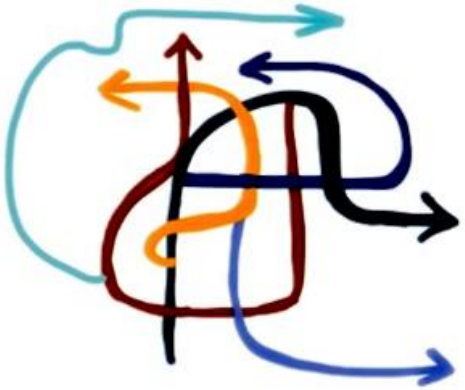
Research ≠ Development

Classic Mistakes : Technology

Silver-Bullet Syndrome



Switching Tools



No version control