# CS 3300
# Intro to Software Engineering

## SOFTWARE ENGINEERING

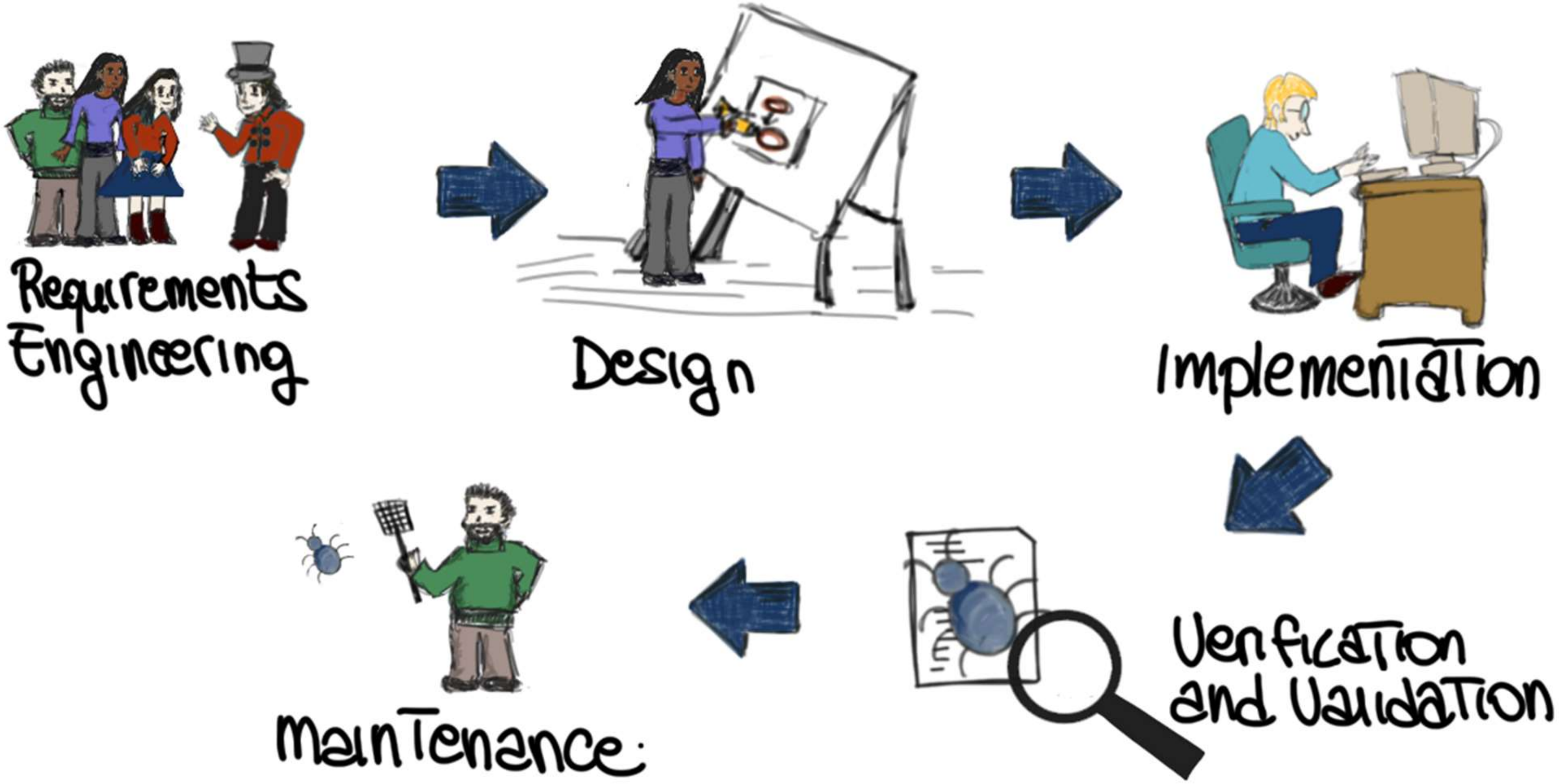
## LIFECYCLE MODELS

Mahdi Roozbahani

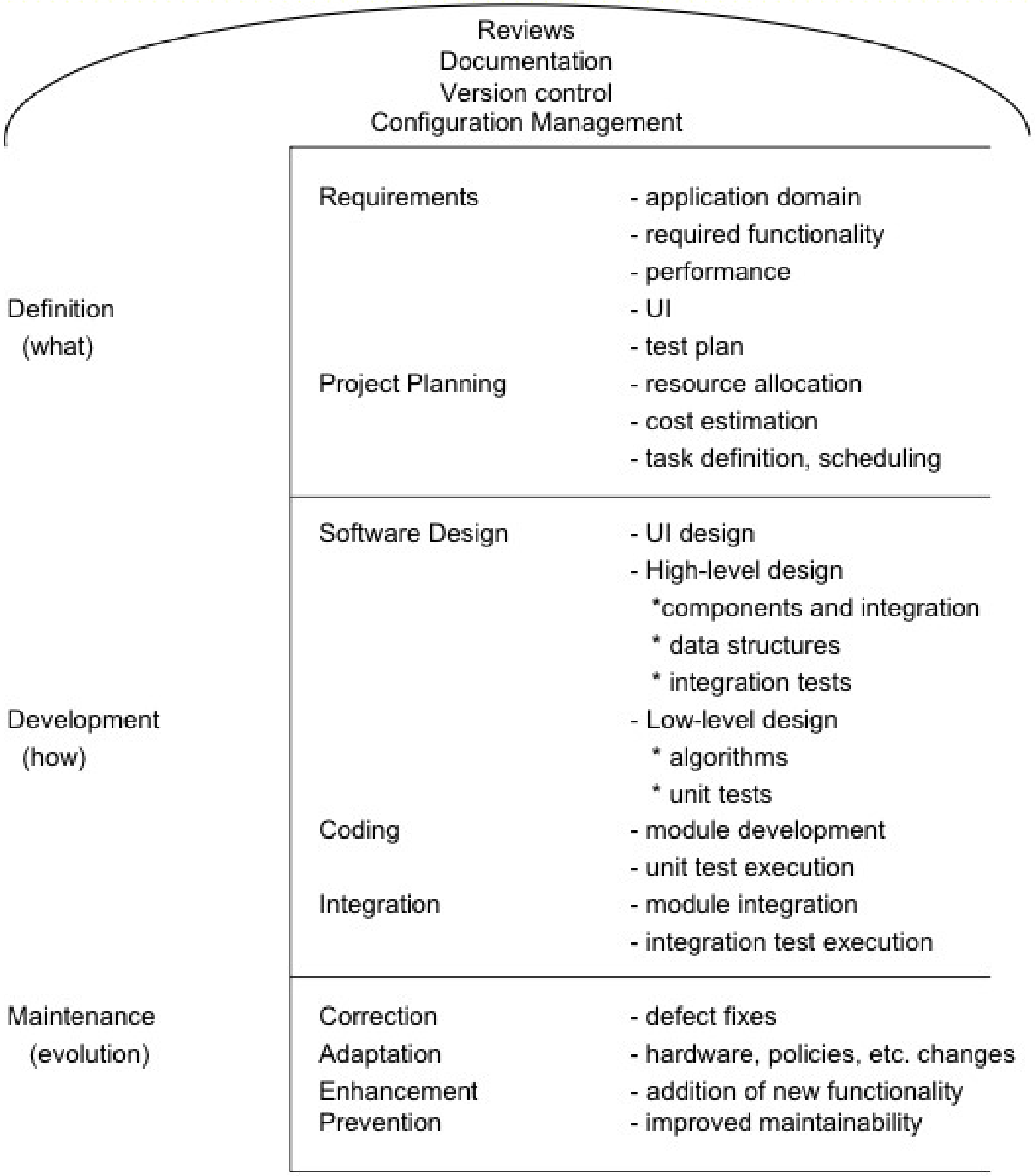
Slides are based on Alex Orso.

# TRADITIONAL SOFTWARE PHASES

# TRADITIONAL SOFTWARE PHASES



# SE PHASES



Reviews
Documentation
Version control
Configuration Management

| | | |
|---|---|---|
| Definition<br>(what) | Requirements | - application domain<br>- required functionality<br>- performance<br>- UI<br>- test plan |
| | Project Planning | - resource allocation<br>- cost estimation<br>- task definition, scheduling |
| Development<br>(how) | Software Design | - UI design<br>- High-level design<br>*components and integration<br>* data structures<br>* integration tests<br>- Low-level design<br>* algorithms<br>* unit tests |
| | Coding | - module development<br>- unit test execution |
| | Integration | - module integration<br>- integration test execution |
| Maintenance<br>(evolution) | Correction | - defect fixes |
| | Adaptation | - hardware, policies, etc. changes |
| | Enhancement | - addition of new functionality |
| | Prevention | - improved maintainability |

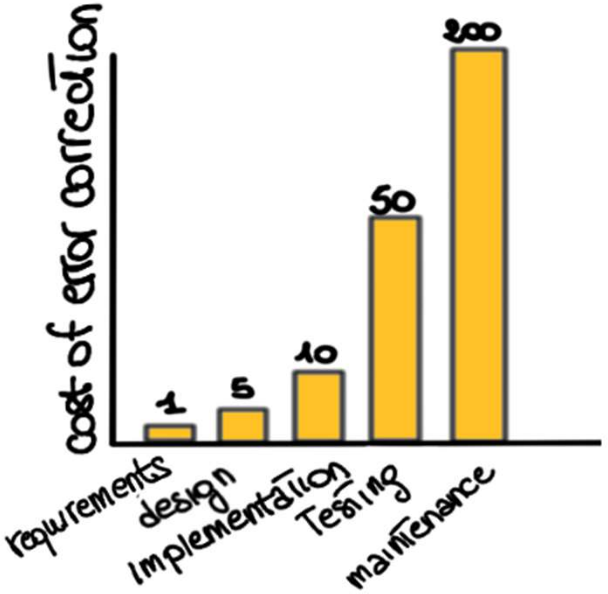
# Umbrella Activities

- Reviews - assure quality
- Documentation - improve maintainability
- Version control - track changes
- Configuration management - integrity of collection of components

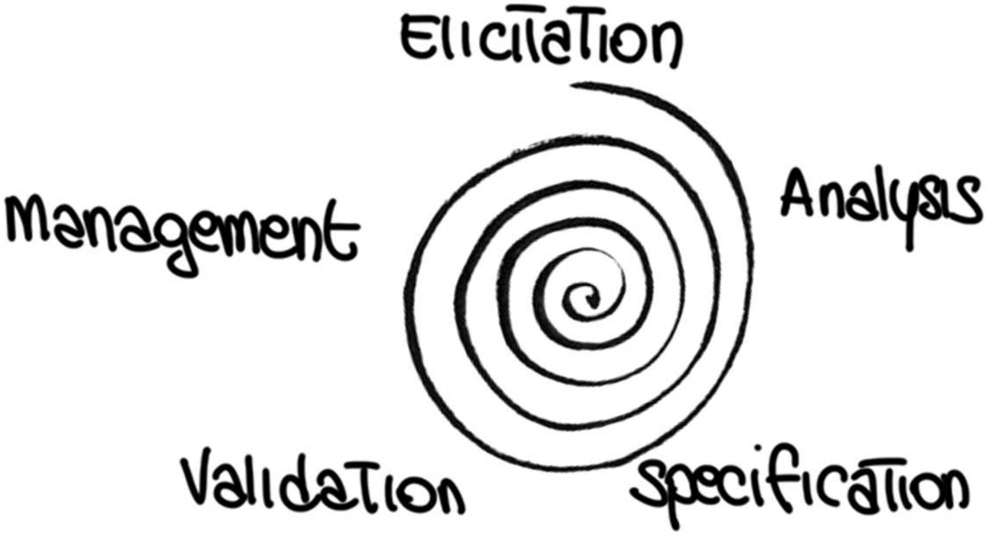
# REQUIREMENTS ENGINEERING

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

# REQUIREMENTS ENGINEERING

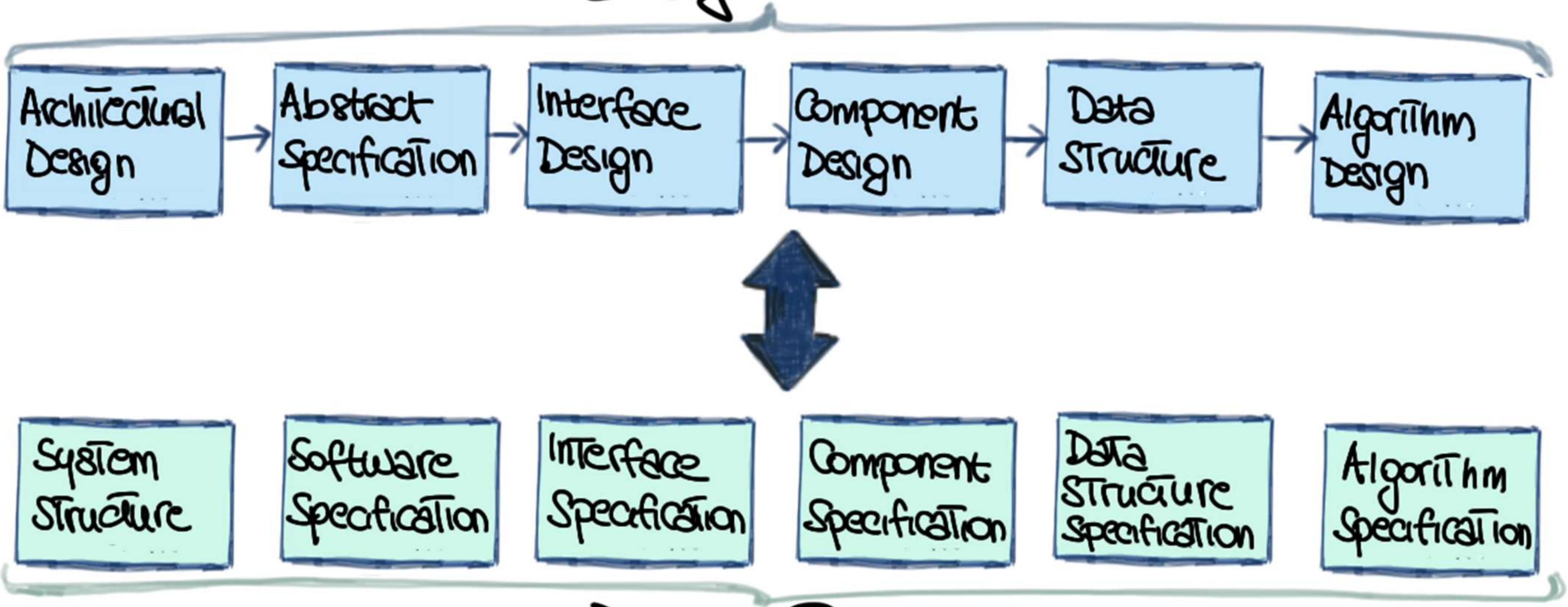


cost of late correction



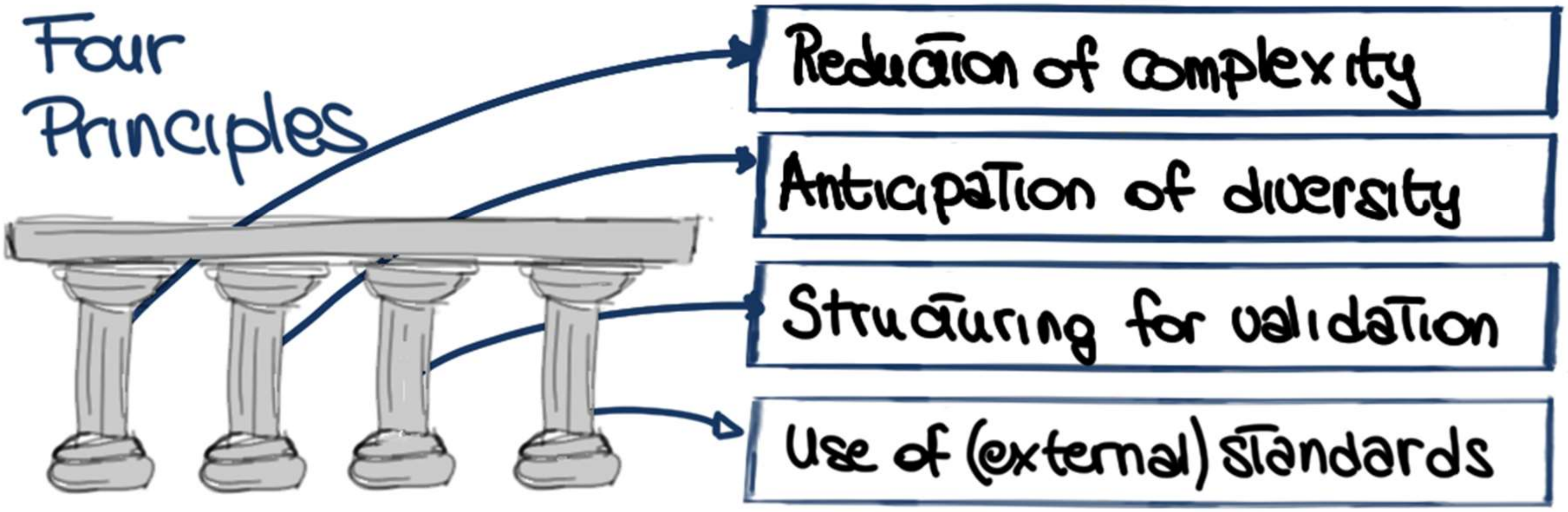
# DESIGN

Design Activities



Design Products

# IMPLEMENTATION



Four Principles

- Reduction of complexity
- Anticipation of diversity
- Structuring for validation
- Use of (external) standards

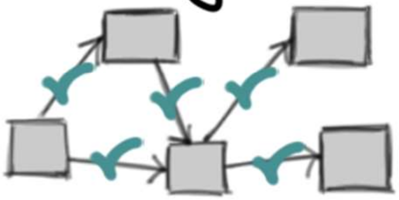
# VERIFICATION & VALIDATION

Validation : did we build the right system?

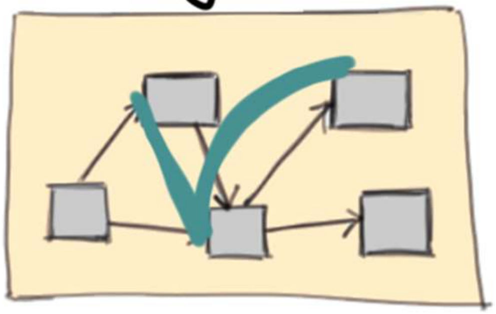
Verification: did we build the system right?

Unit

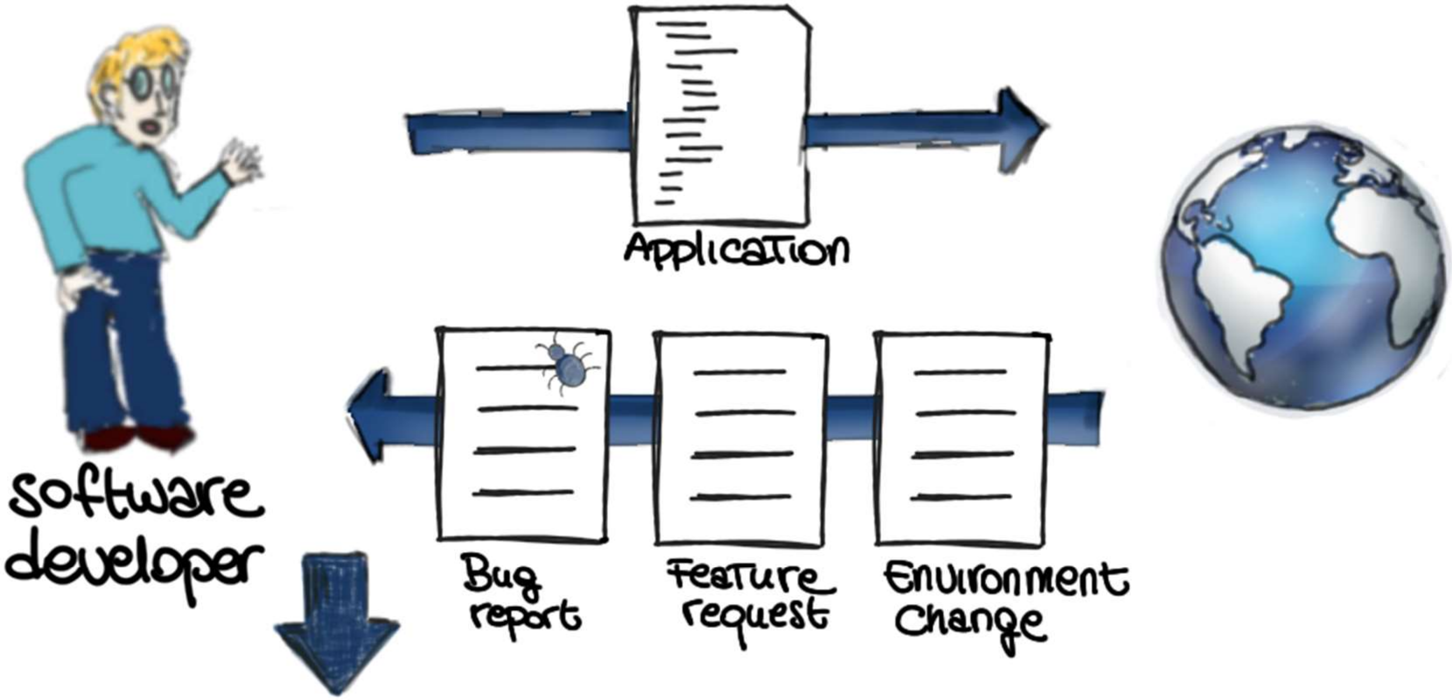
Integration



System

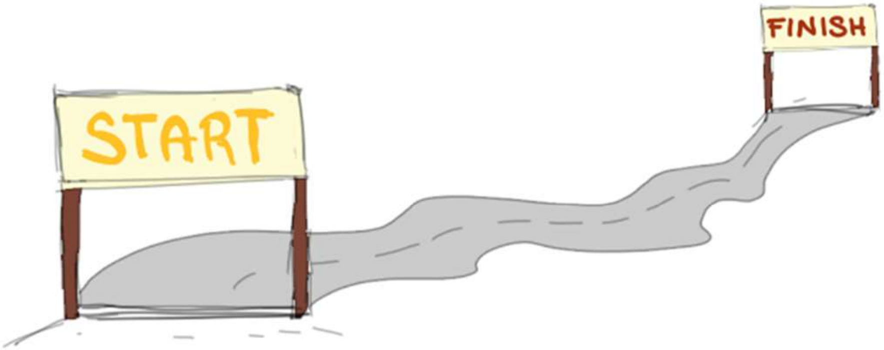


# MAINTENANCE

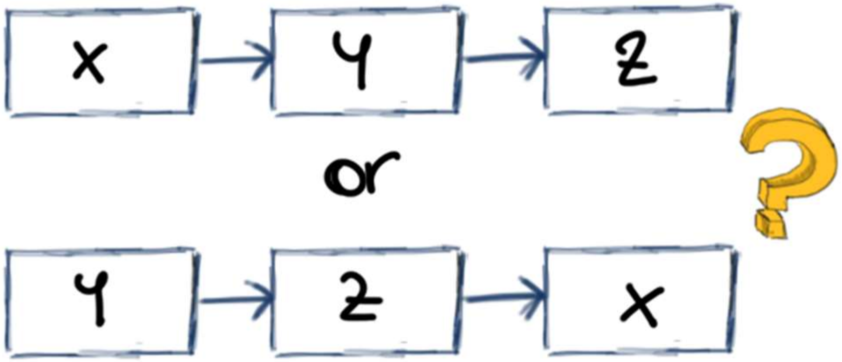


Corrective maintenance
Perfective maintenance
Adaptive maintenance

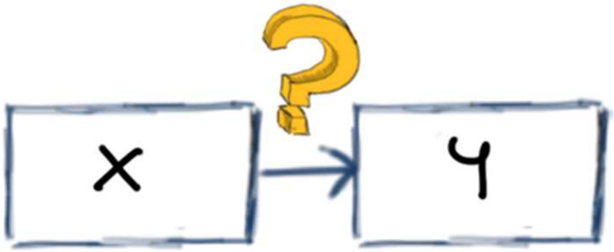
# SOFTWARE PROCESS MODEL



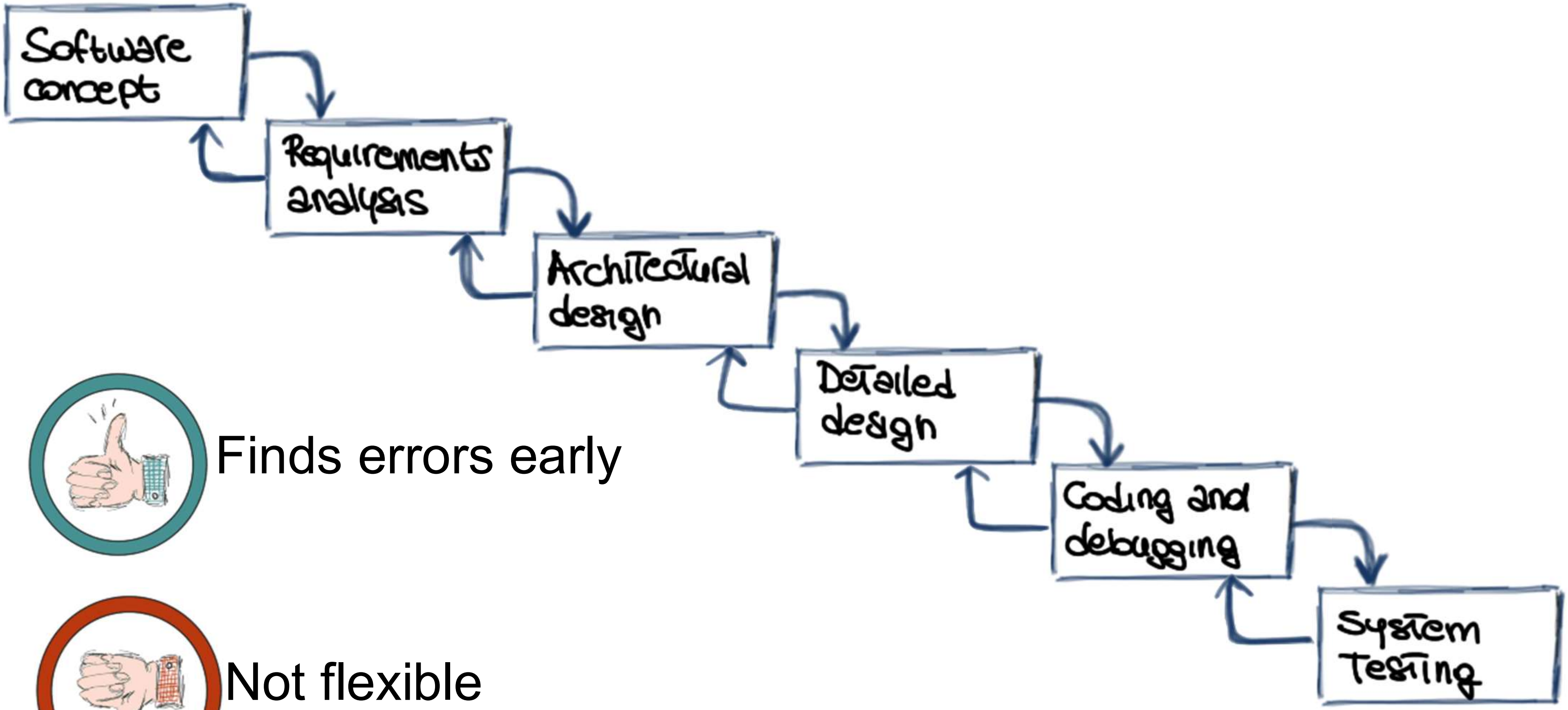
Determine the order



Establish The Transition criteria



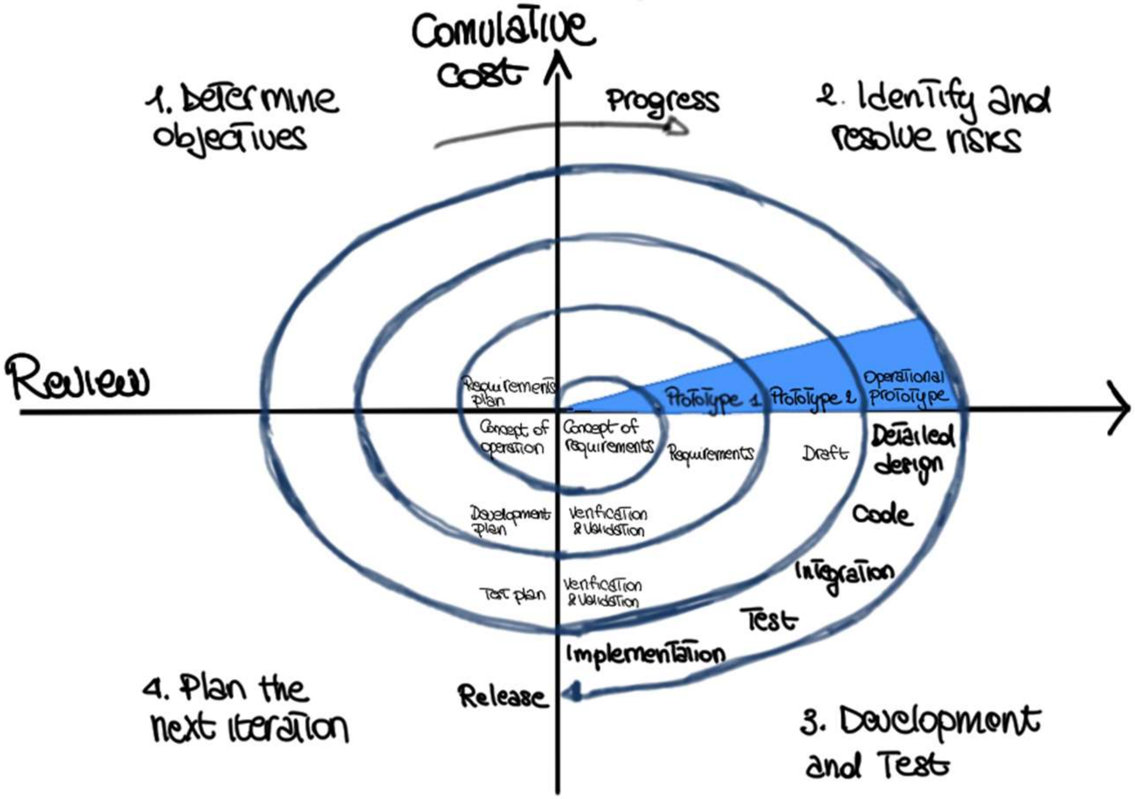
# WATERFALL



👍 Finds errors early

👎 Not flexible

# SPIRAL



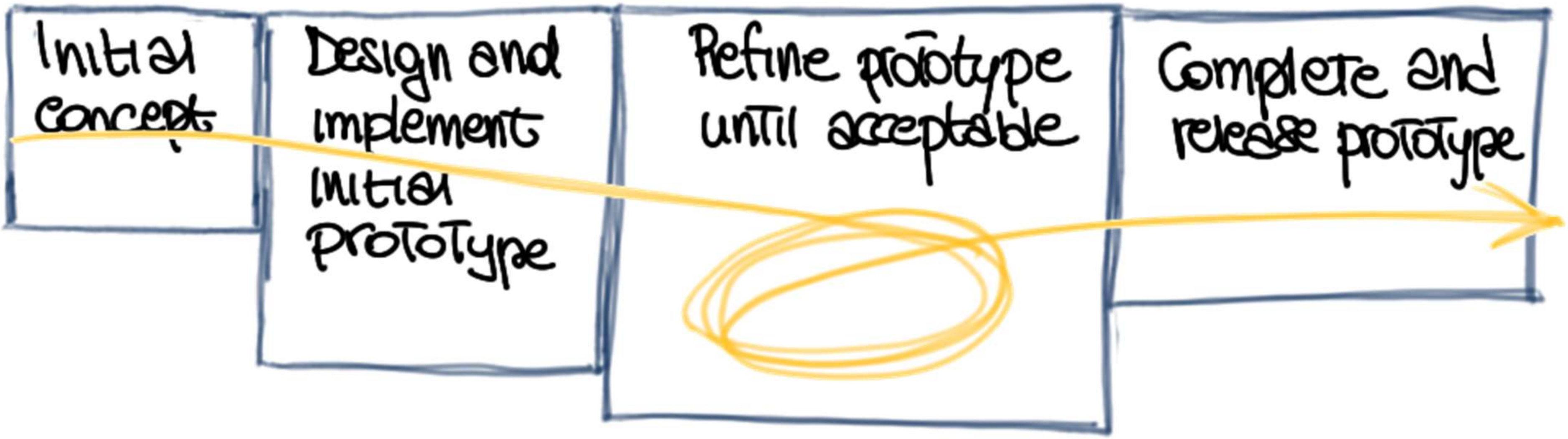
👍

- Risk reduction
- Functionality can be added
- Software produced early

👎

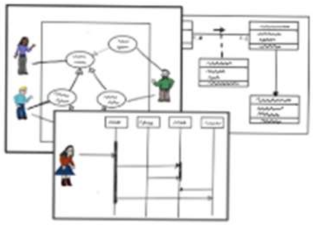
- Specific expertise
- Highly dependent on risk analysis
- Complex

# EVOLUTIONARY PROTOTYPING

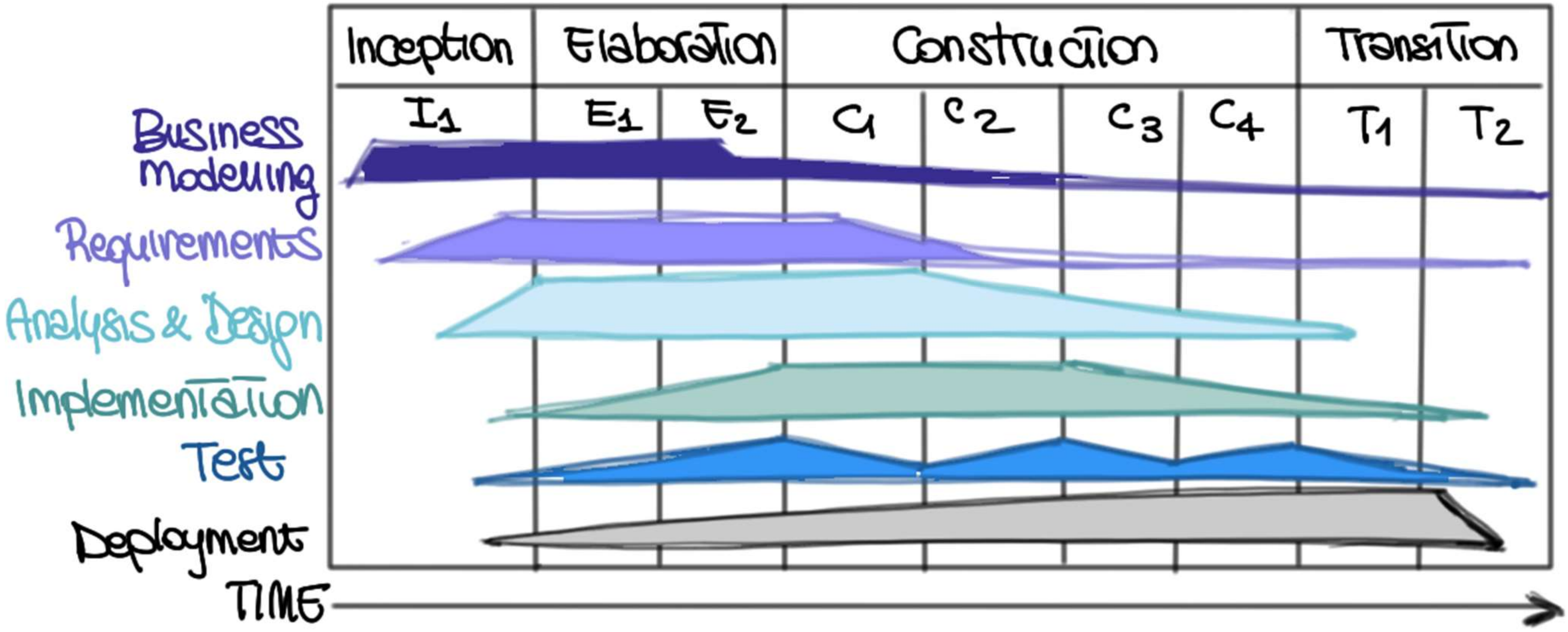

Initial concept
Design and implement initial prototype
Refine prototype until acceptable
Complete and release prototype

- Immediate feedback
- Helps requirements understanding

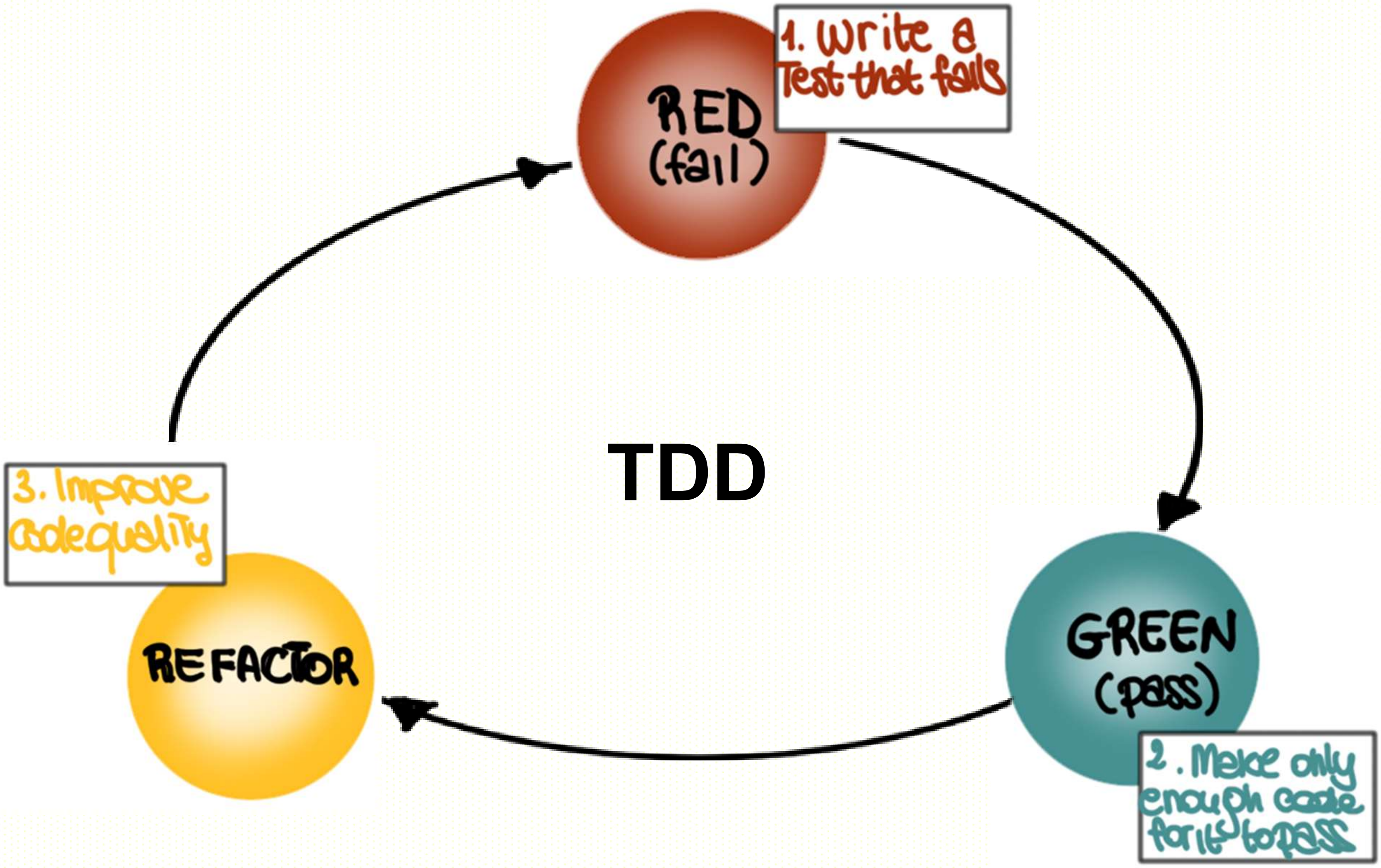
- Difficult to plan
- Can deteriorate to code-and-fix



# RATIONAL UNIFIED PROCESS (RUP)

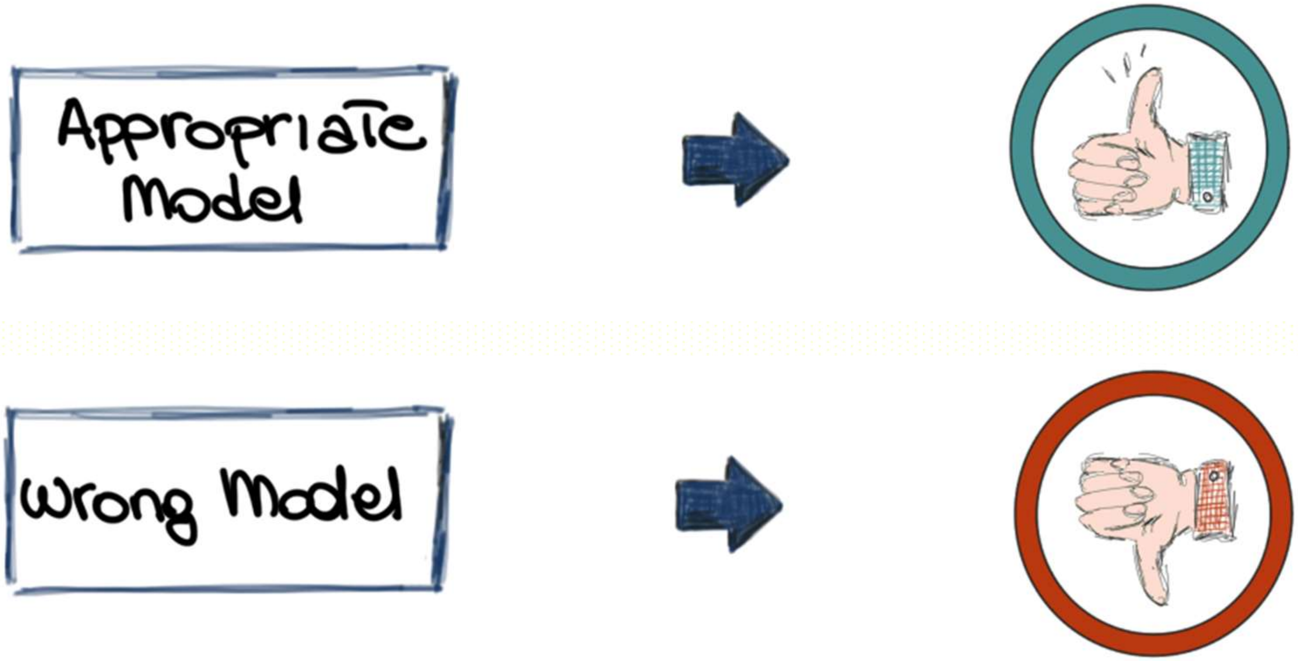


# AGILE


RED
(fail)
1. Write a
Test that fails
GREEN
(pass)
2. Make only
enough code
for it to pass
REFACTOR
3. Improve
codequality

TDD

# CHOOSING A MODEL



# CHOOSING A SOFTWARE PROCESS MODEL

Which of the following models is most suitable to develop a software control system ?

[ ] Pure waterfall

[ ] TDD

[ ] Evolutionary prototyping

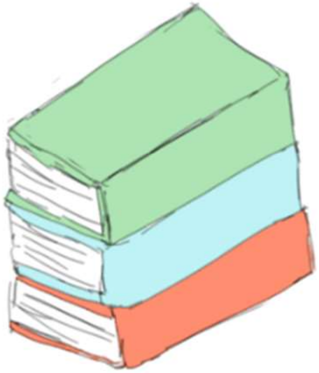
Which model is the most suitable if you except midcourse corrections ?

[ ] Pure waterfall

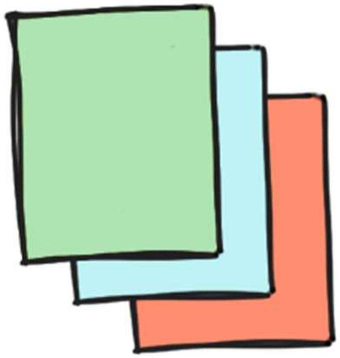
[ ] Spiral

[ ] Evolutionary prototyping

# LIFECYCLE DOCUMENTS



IEEE Documents Definition



Light-weight documents

# CLASSIC MISTAKES

From: Steven C. McConnell
Rapid Development
Classic Mistakes Enumerated

# CLASSIC MISTAKES: PEOPLE

# CLASSIC MISTAKES: PEOPLE

Heroics

# CLASSIC MISTAKES: PEOPLE

Heroics

Work environment

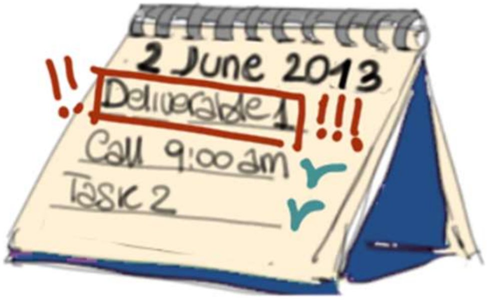
# CLASSIC MISTAKES: PEOPLE

Heroics

Work environment

People management

# CLASSIC MISTAKES: PROCESS



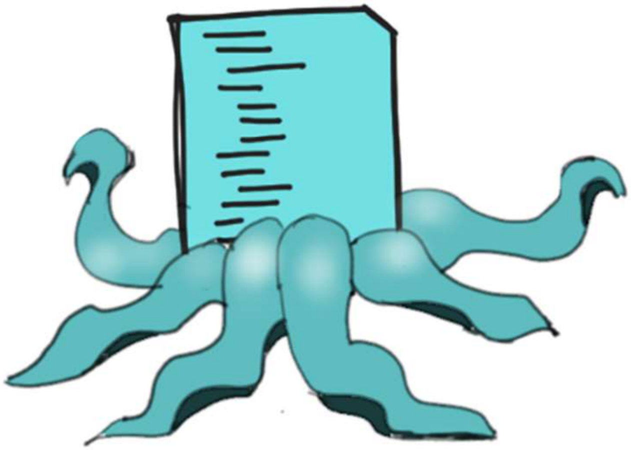
Scheduling issues

Planning issues

Failures

# CLASSIC MISTAKES: PRODUCT

Gold plating

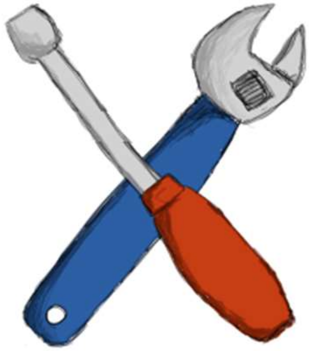


Feature creep

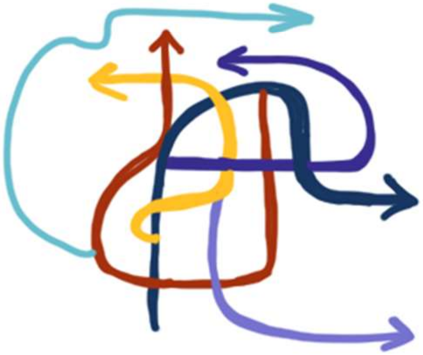
Research ≠ Development

# CLASSIC MISTAKES: TECHNOLOGY

silver- bullet syndrome



switching tools



No version control

# But first… pop quiz!

Questions about the first reading:

1. What is a software lifecycle?
2. What are the main purposes of a lifecycle model?
3. Mention one positive and one negative aspect of the waterfall model.
4. Give an example of a situation where you used the code-and-fix model (or where it would be appropriate to use it, if you never used it).
5. Mention one possible modification of the waterfall model that could help address the problem of changing requirements.
6. What is the difference between staged delivery and evolutionary prototyping?
7. Among the ones you know, which lifecycle model is the most effective in producing software of high quality, at a low cost, and within you schedule constraints?