ECS505U
SOFTWARE ENGINEERING

MUSTAFA BOZKURT

LECTURER IN SOFTWARE ENGINEERING

Week 3

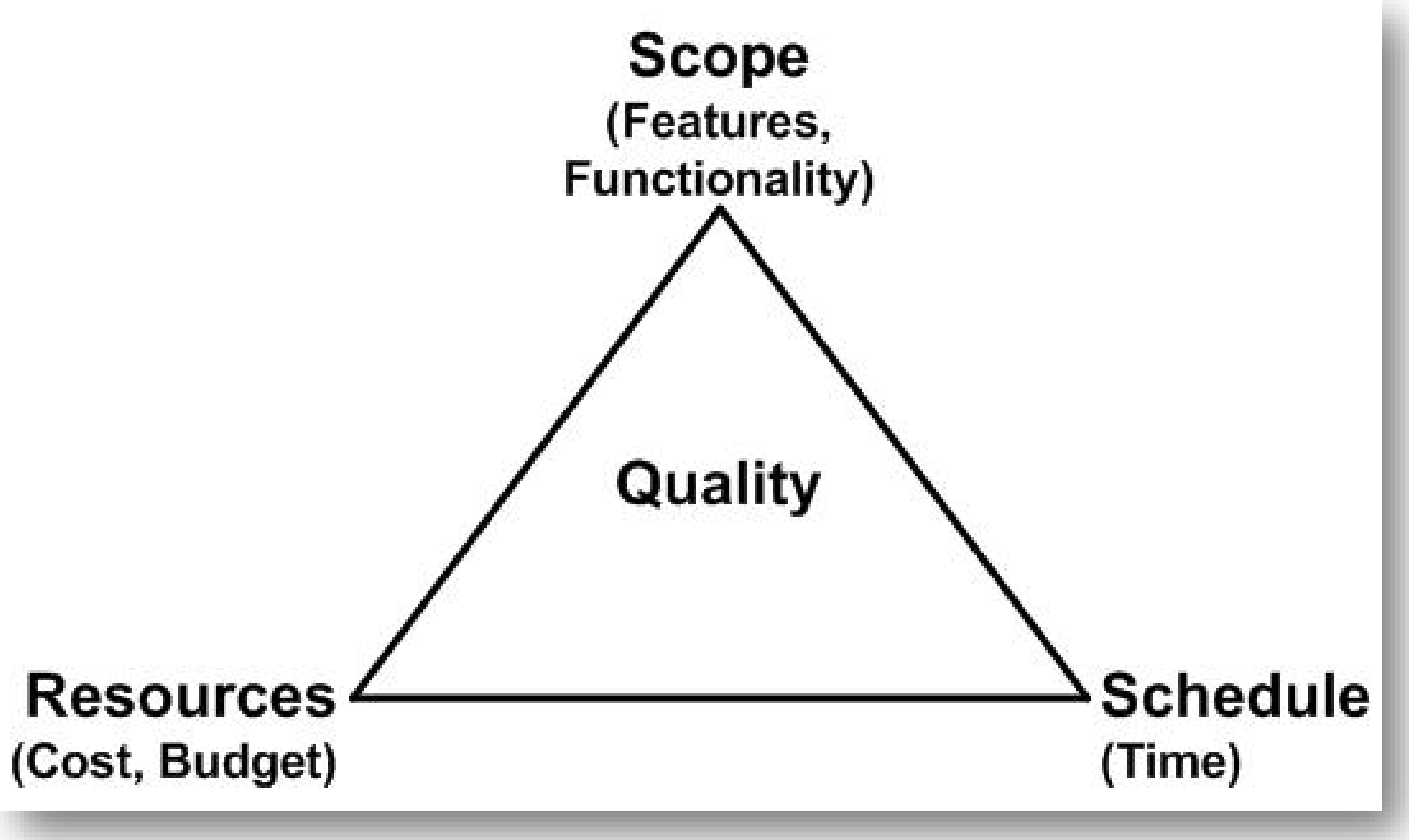
SOFTWARE LIFE-CYCLE PROCESSES

From Waterfall to Extreme Programming

LESSON OBJECTIVES

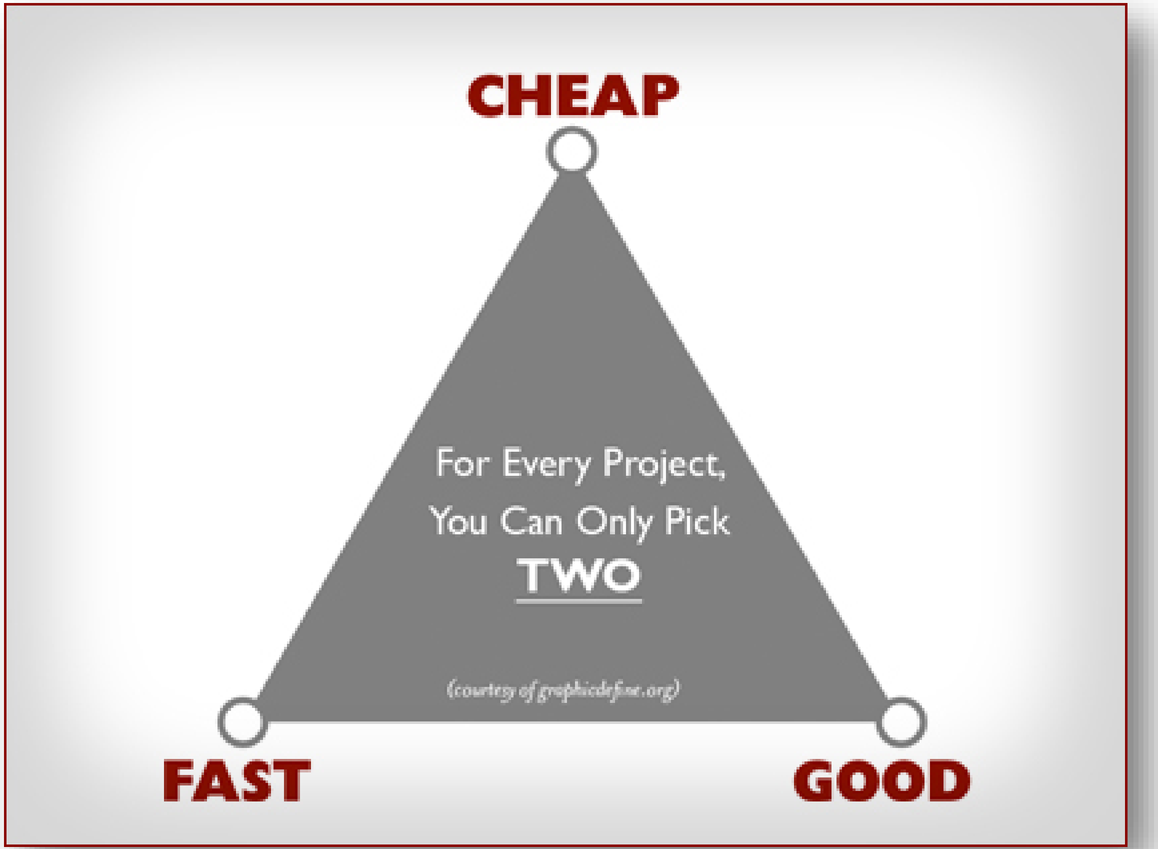
- Understand major activities of software projects
- Understand the place of these in different life-cycle models
- Understand the pros and cons of different life-cycle models
- Know enough about the incremental delivery and extreme programming to use relevant parts

THE IRON TRIANGLE OF QUALITY



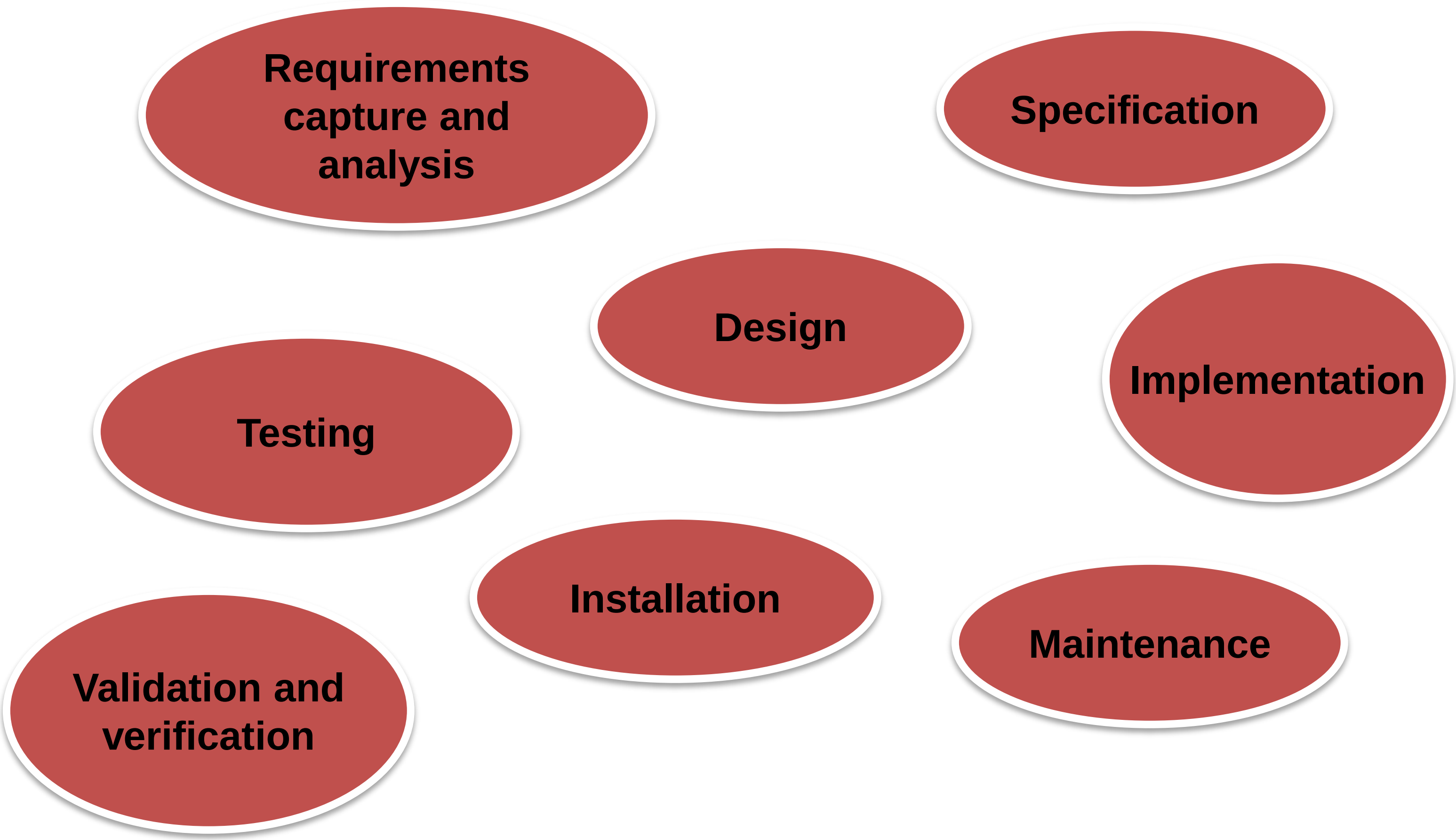
Produce quickly, with exceptional quality and at a reasonable cost

THE REVISED IRON TRIANGLE



http://www.mopdog.com/wp-content/uploads/2009/09/revised-triangle.jpg

SOFTWARE LIFE-CYCLE PHASES



Requirements capture and analysis

Specification

Design

Implementation

Testing

Installation

Maintenance

Validation and verification

REQUIREMENTS CAPTURE AND ANALYSIS

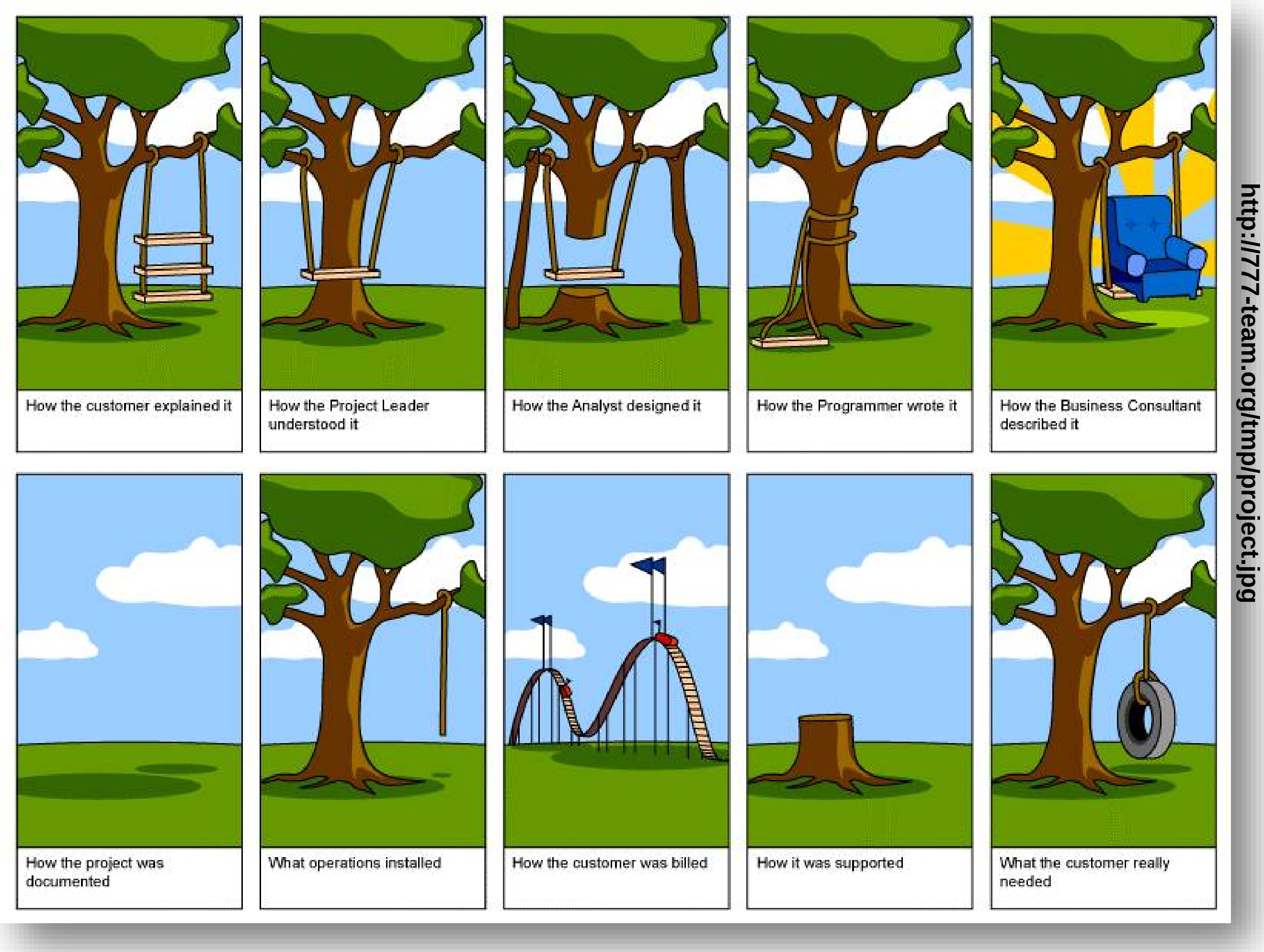
- Identify and agree general functional and non-functional requirements
- Prioritise requirements where possible
- Determine risk factors, cost analysis, development schedule
- Identify how customer might test completed system
- Identify likely changes and how to accommodate them

SPECIFICATION

- Says what the system does not how it does it
- Formal version of requirements specification, written by developer
- Developer identifies errors, omissions, or impracticalities in the requirements
- Crystallises precise functional and performance requirements
- Basis for formal contract, hence should contain sufficient explanation for customer to understand
- Should include detailed plan for acceptance test

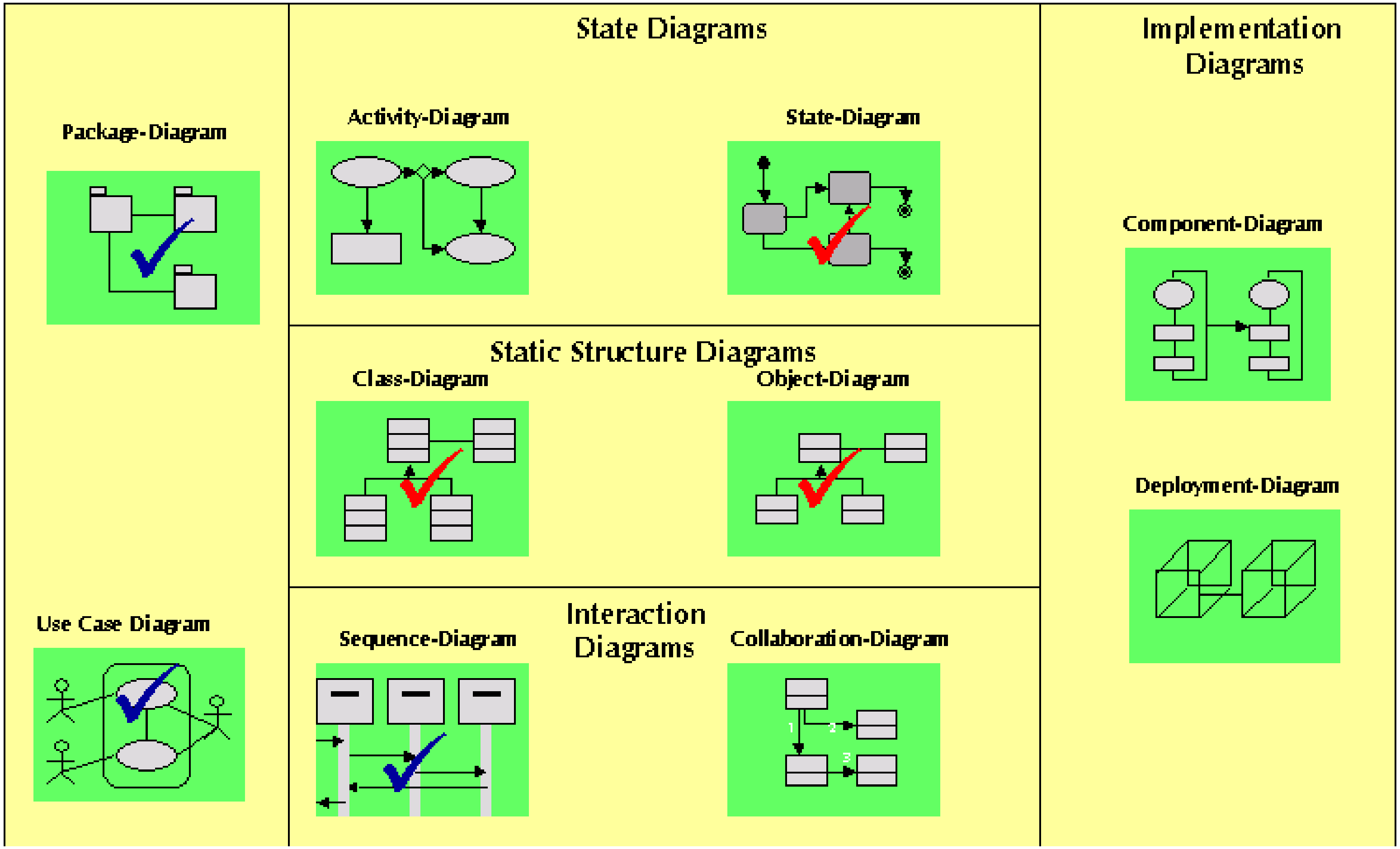
THE PROBLEM WITH SPECIFICATION

St


How the customer explained it
How the Project Leader understood it
How the Analyst designed it
How the Programmer wrote it
How the Business Consultant described it
How the project was documented
What operations installed
How the customer was billed
How it was supported
What the customer really needed

http://777-team.org/tmp/project.jpg

DESIGN



We will focus on object-oriented design using UML

IMPLEMENTATION

Process of expressing the detailed design in a programming language so that it will run on the target computer.

VERIFICATION AND VALIDATION

Verification:

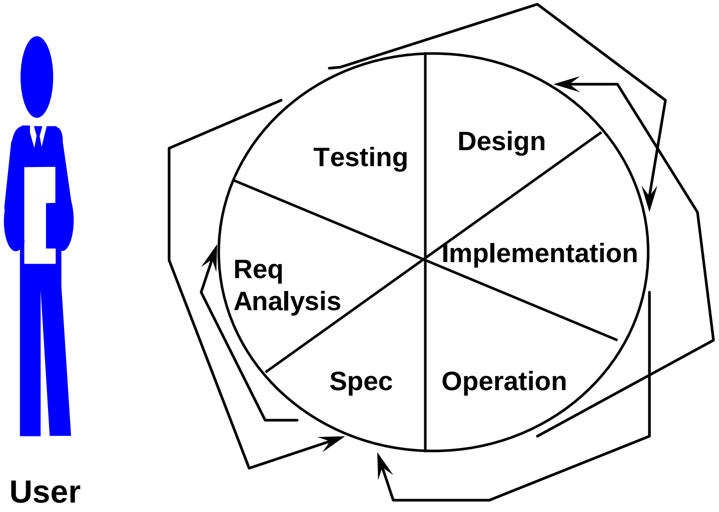
Are we building the product right?

Validation:

Are we building the right product?

VERIFICATION AND VALIDATION

Verification



User

Testing

Design

Implementation

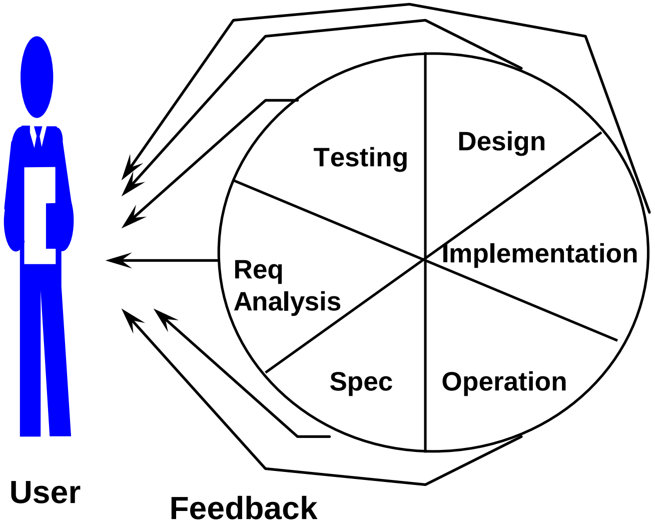
Req Analysis

Spec

Operation

Are we building the product right?

Validation



User

Feedback

Testing

Design

Implementation

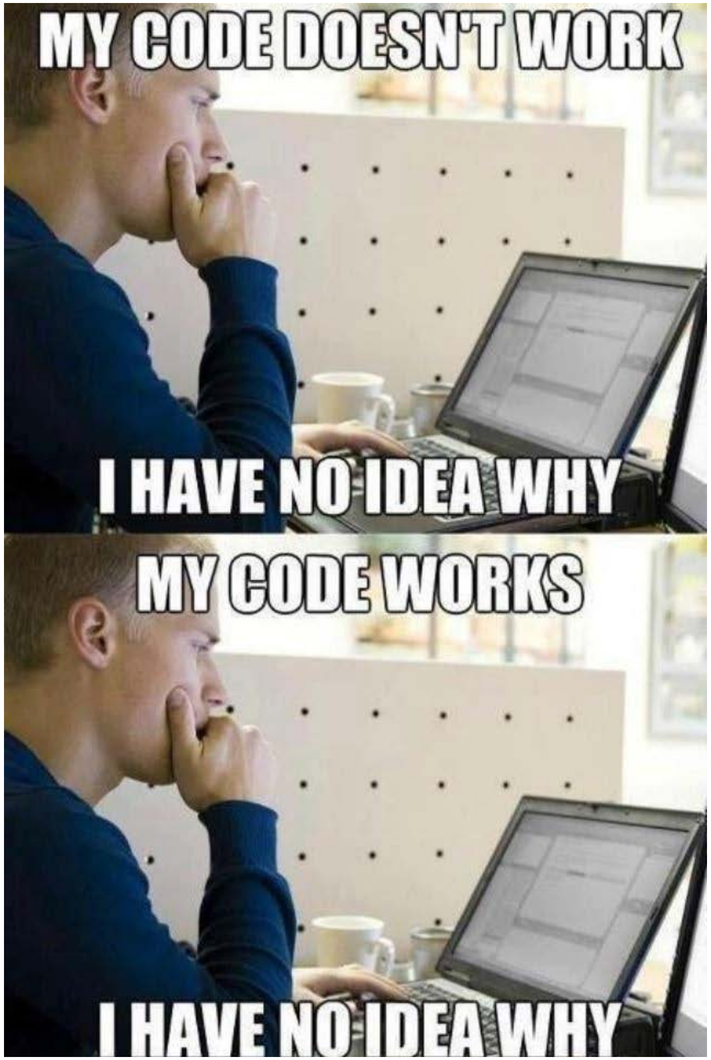
Req Analysis

Spec

Operation

Are we building the right product?

TESTING


MY CODE DOESN'T WORK
I HAVE NO IDEA WHY
MY CODE WORKS
I HAVE NO IDEA WHY

TESTING

Unit tests

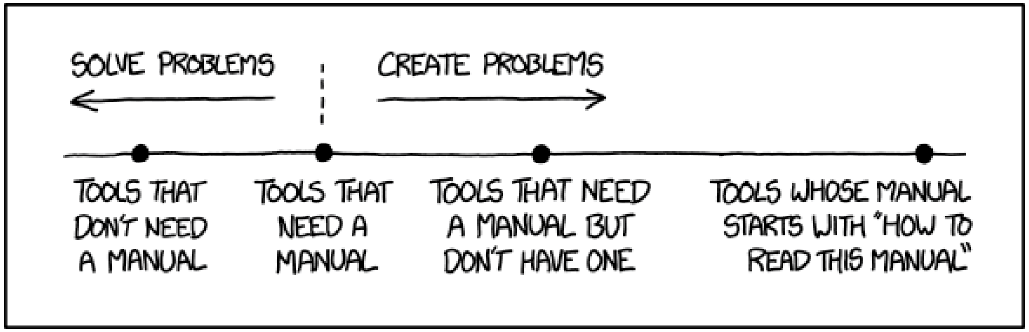
Integration tests

System and acceptance tests

Regression tests

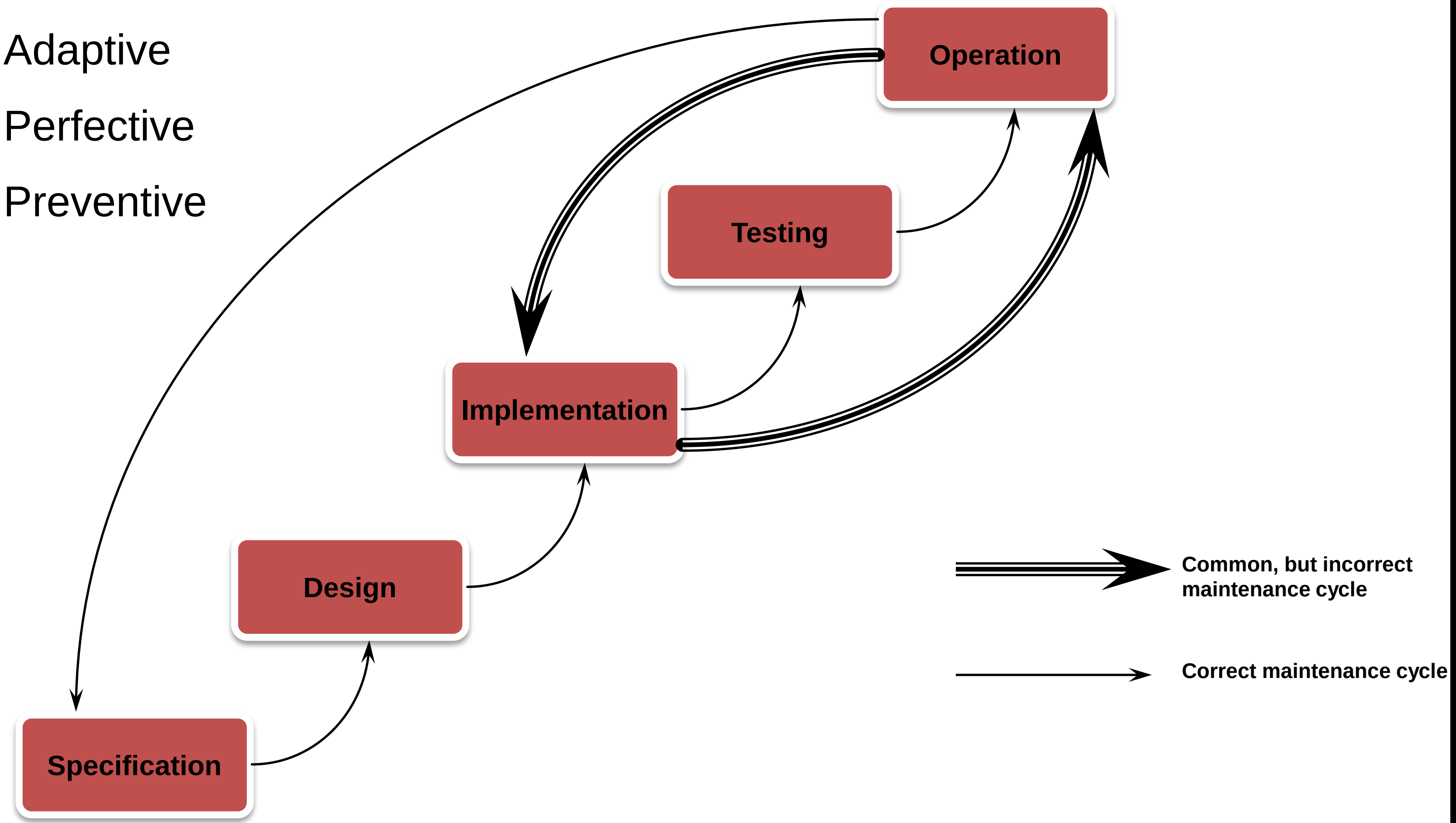
INSTALLATION

- Create working version on target machine
- Make available all support files and manuals
- Coordinate with hardware
- Rerun acceptance test
- Training and on-line support
- Evaluation of project


SOLVE PROBLEMS
CREATE PROBLEMS
TOOLS THAT DON'T NEED A MANUAL
TOOLS THAT NEED A MANUAL
TOOLS THAT NEED A MANUAL BUT DON'T HAVE ONE
TOOLS WHOSE MANUAL STARTS WITH "HOW TO READ THIS MANUAL"

MAINTENANCE AND SUPPORT

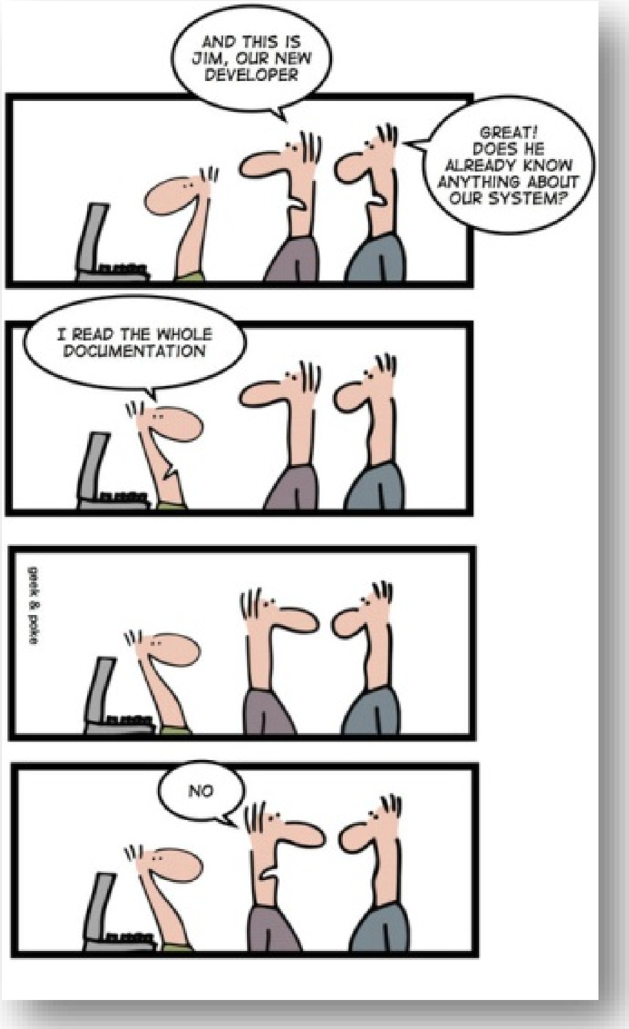
- Corrective
- Adaptive
- Perfective
- Preventive


Operation
Testing
Implementation
Design
Specification
Common, but incorrect maintenance cycle
Correct maintenance cycle

SOFTWARE DOCUMENTATION

Types of documentation:

- Requirements
- Architecture/Design
- Technical
- End user
- Marketing


AND THIS IS JIM, OUR NEW DEVELOPER
GREAT! DOES HE ALREADY KNOW ANYTHING ABOUT OUR SYSTEM?
I READ THE WHOLE DOCUMENTATION
geek & poke
NO

http://media.tumblr.com/tumblr_m8pz5fKnq21rps1w0.jpg

LIFE-CYCLE TYPES

- Build-and-fix
- Waterfall
- Rapid prototyping
- Incremental development
- Formal transformations model
- Spiral model
- Unified process (UP)
- Extreme programming (XP)

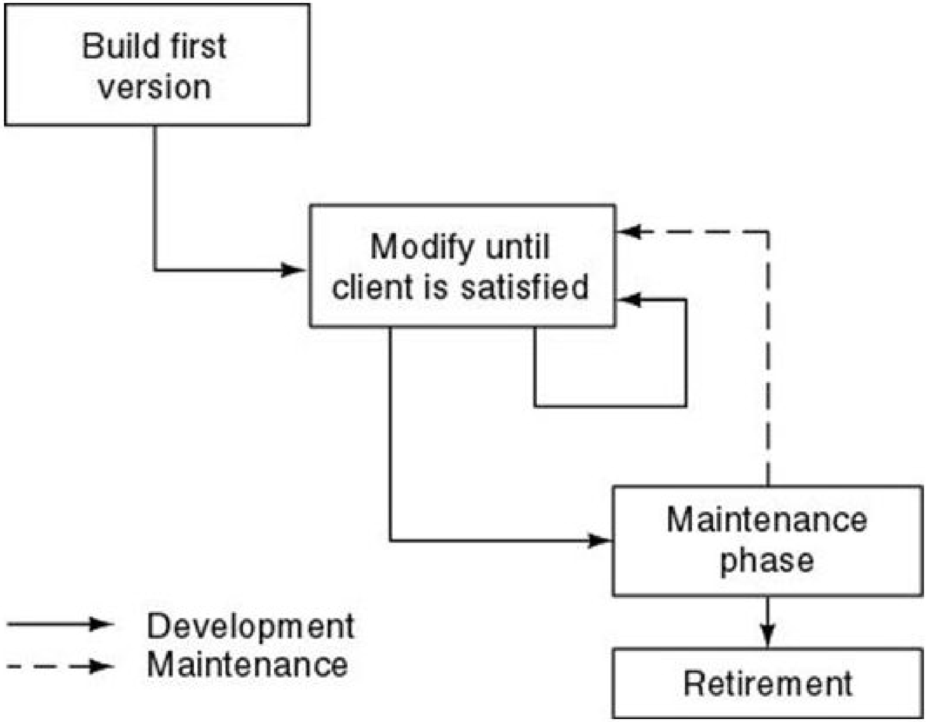
BUILD-AND-FIX MODEL

Problems

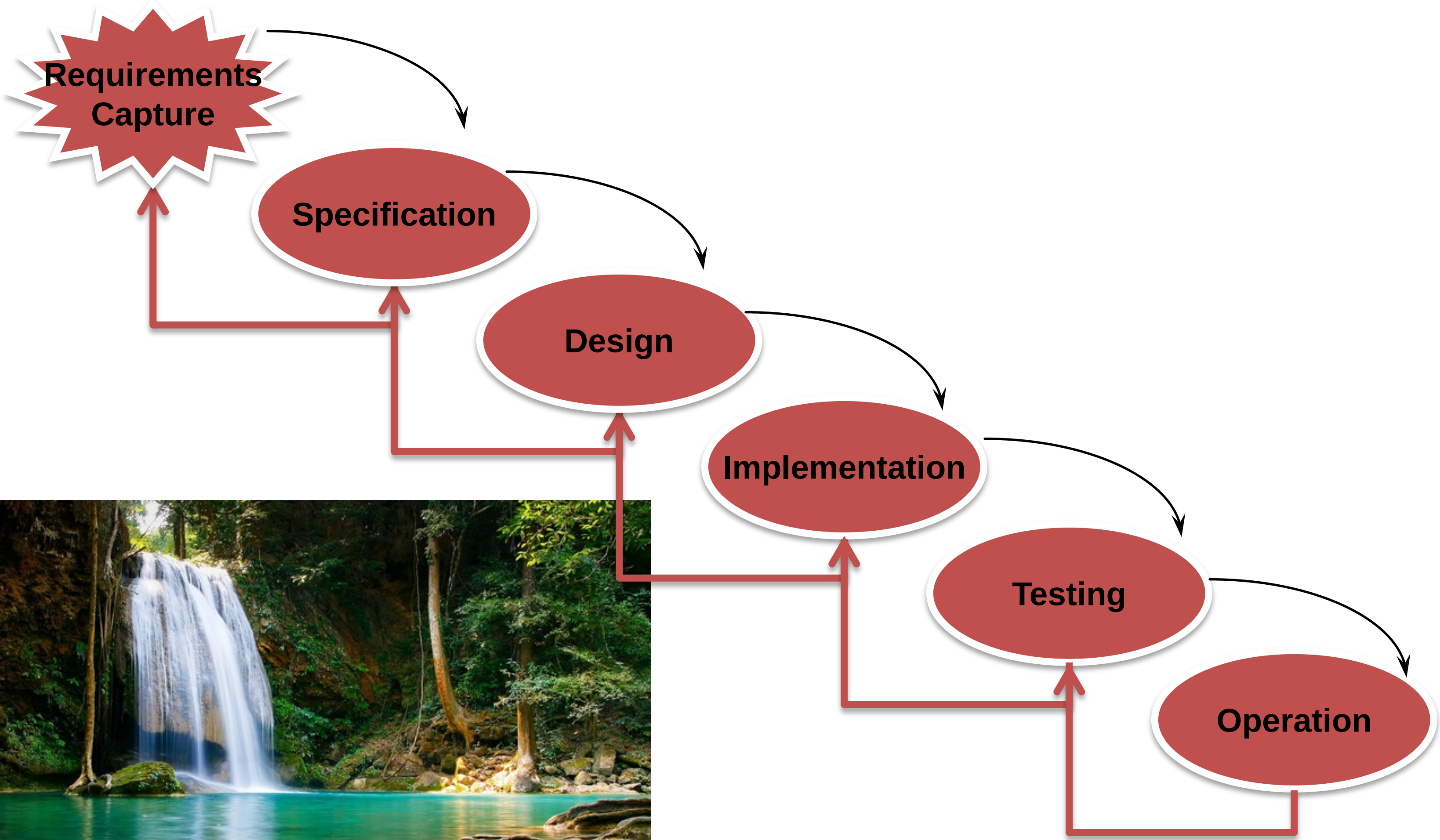
- No specifications
- No design

Totally unsatisfactory

- High cost
- Difficult maintenance



WATERFALL MODEL



WATERFALL MODEL

Advantages:

Enforces disciplined approach

- Documentation for each phase
- Products of each phase checked for QA

Maintenance is easier

- Every change reflected in the relevant documentation

WATERFALL MODEL

Disadvantages:

- Working version of the software will not be available until late in the project time-span
- Specifications are long, detailed
- “Blocking states” – some project team members must wait for other team members to complete dependent tasks

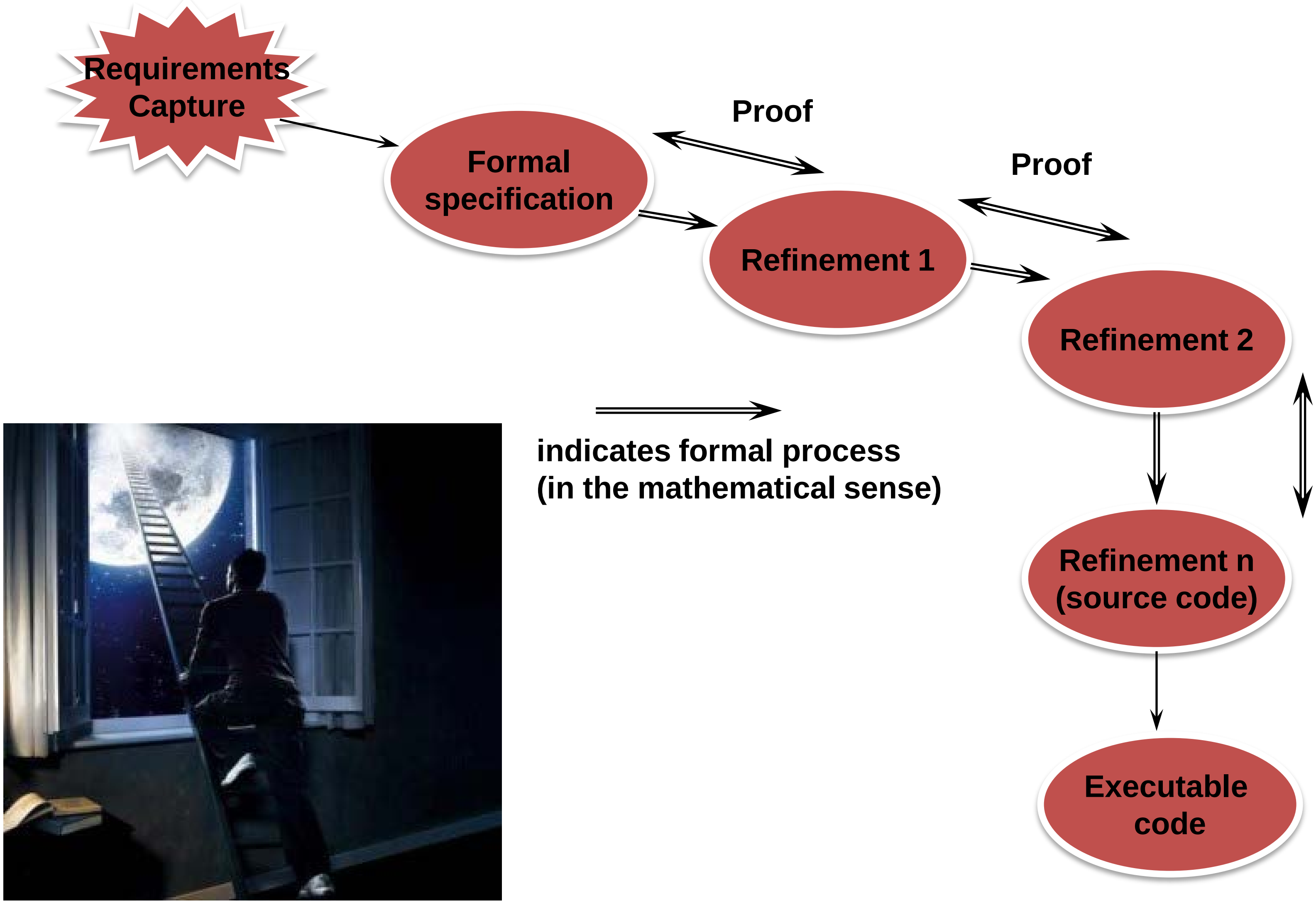
WATERFALL MODEL

A common feature of waterfall projects is that they tend to fail.

18% cancelled, 53% late, over budget or descoped[1].

[1] CHAOS Demographics and Project Resolution 2004, Standish Group

FORMAL TRANSFORMATION MODEL



indicates formal process (in the mathematical sense)

FORMAL TRANSFORMATION MODEL

Advantages:

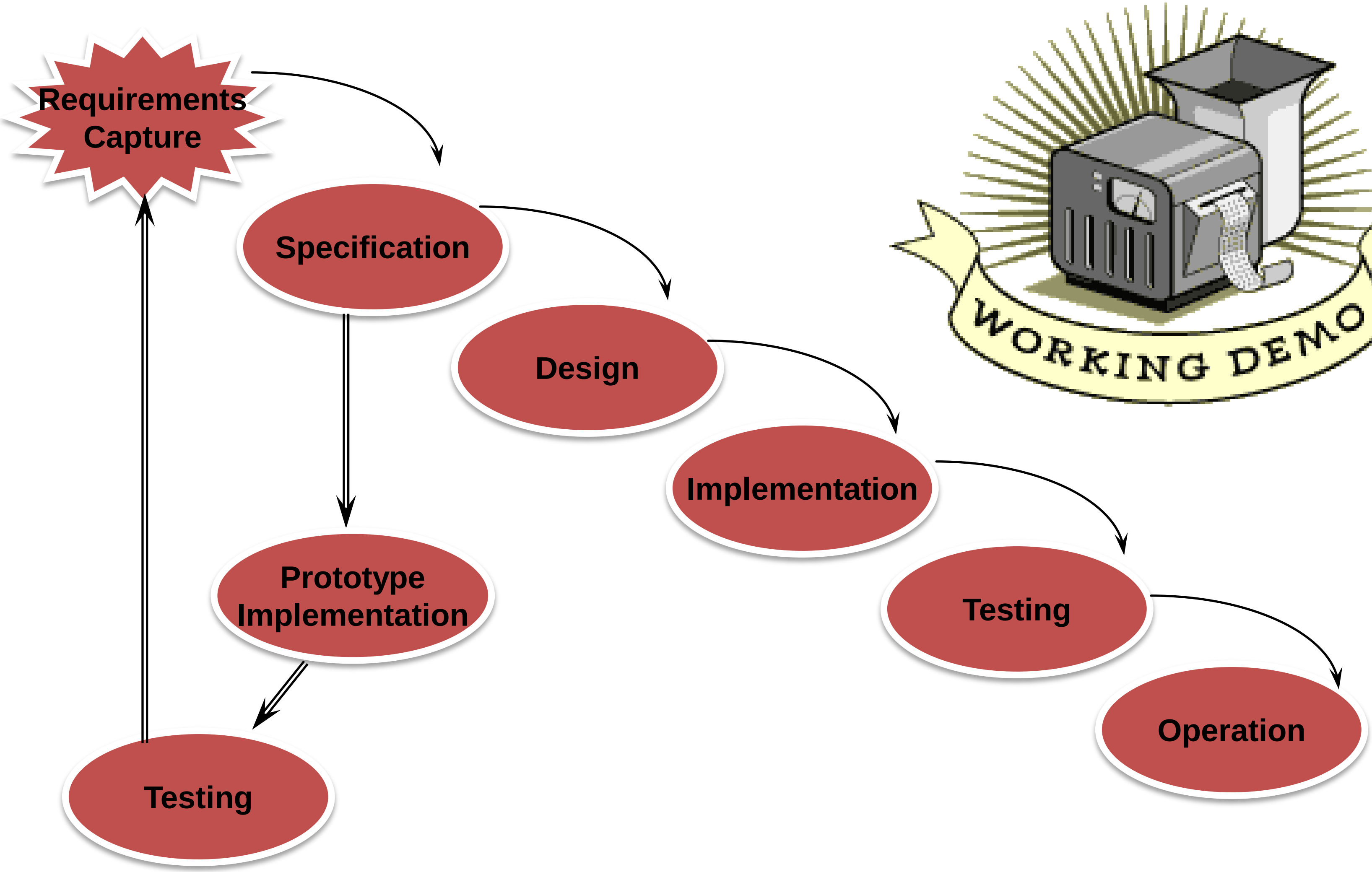
- Very precise
- Often used in safety critical systems.

FORMAL TRANSFORMATION MODEL

Disadvantages:

- Considerable expertise is required.
- Extra level of complexity!
- It is very costly and might become error prone with excessive verification (due to errors in transformation).

RAPID PROTOTYPING



RAPID PROTOTYPING

Advantages:

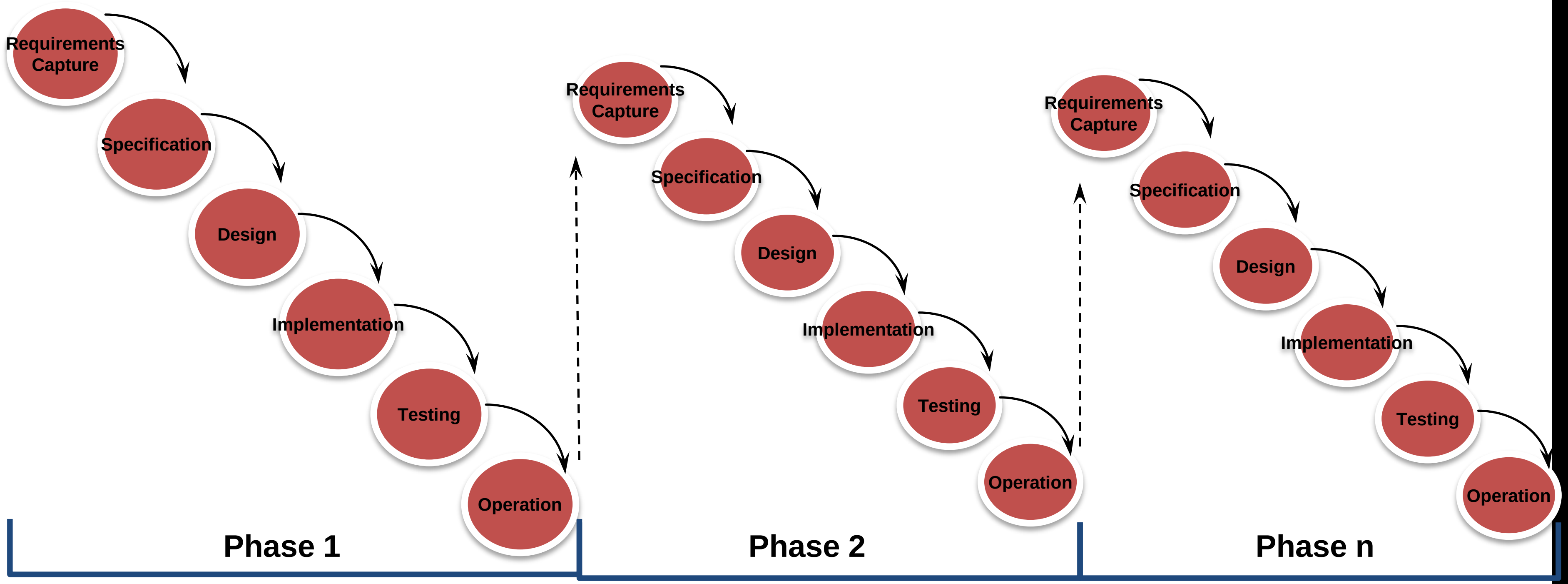
- No specifications!
- Speed and accuracy.

RAPID PROTOTYPING

Disadvantages:

- Client disappointment!
- Hard to focus developers to implement a prototype.
- Legal issues might arise.

ITERATIVE DEVELOPMENT



“You should use iterative development only on projects that you want to succeed”

Martin Fowler, 2000

ITERATIVE DEVELOPMENT

Advantages:

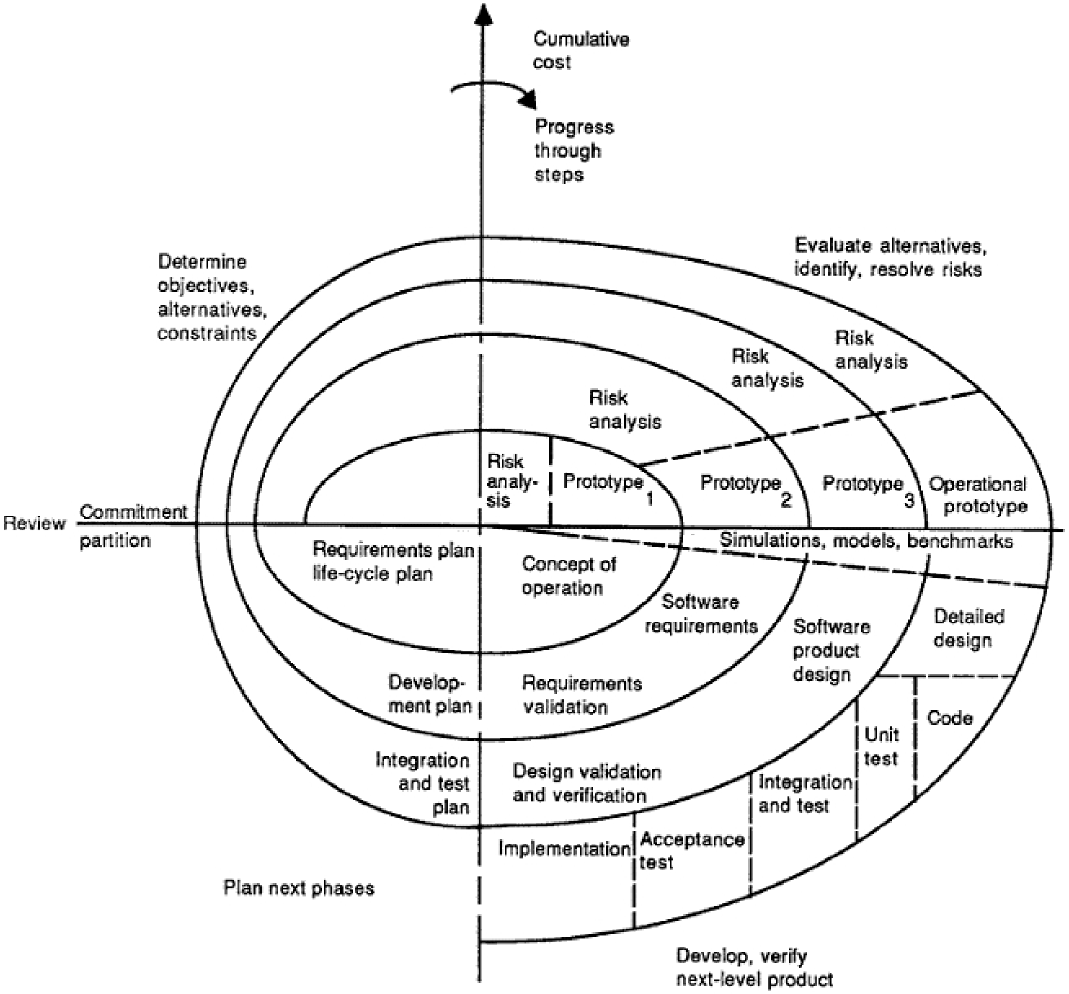
- Customer feedback due to early working software.
- Requirements and specifications in phases.
- Step-by-step improvement makes easy to track defects.

ITERATIVE DEVELOPMENT

Disadvantages:

- Without careful planning early design decisions can cripple future evolutions.
- Difficult to switch methodologies and languages.
- Not having all requirements might cause problems.

SPIRAL MODEL



SPIRAL MODEL

- Determine objectives
- Specify constraints
- Generate alternatives
- Identify risks
- Resolve risks
- Develop & verify next-level product
- Plan next iteration

SPIRAL MODEL

Advantages:

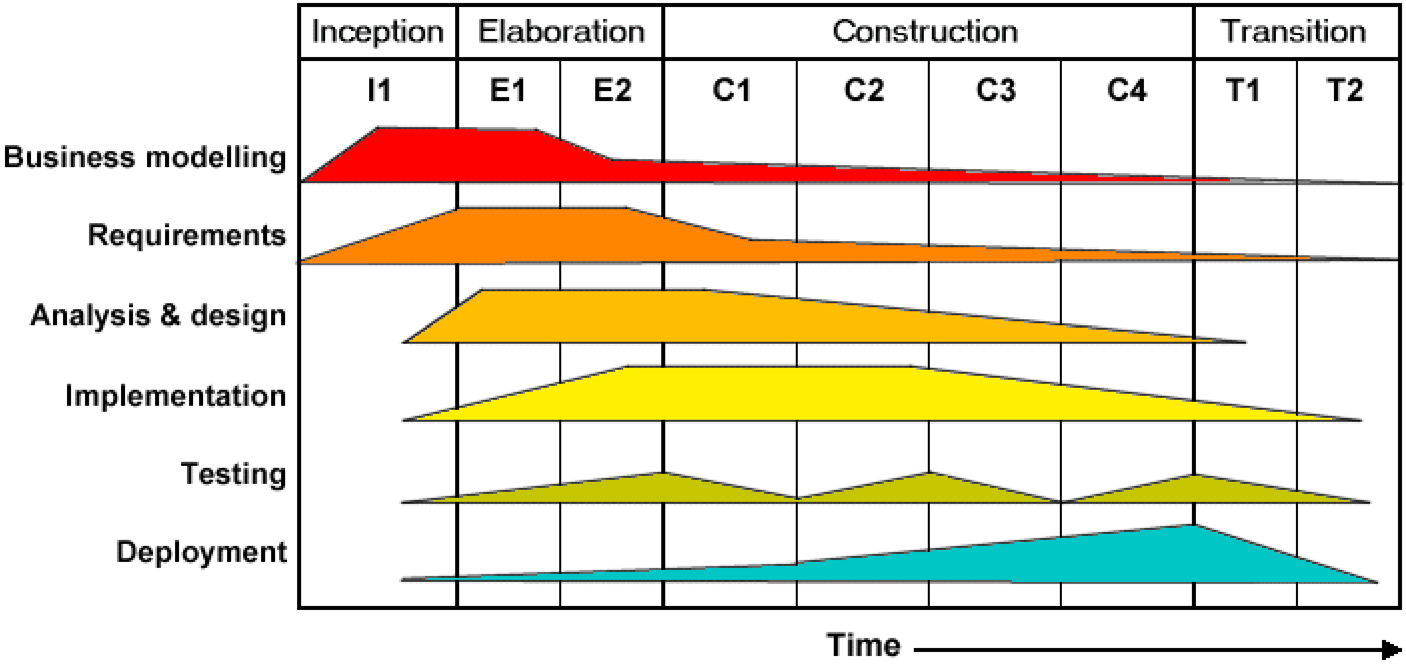
- Realistic approach for large-scale projects.
- Prototyping to reduce risk.
- Iterative and incremental approach.
- Reduced risk.

SPIRAL MODEL

Disadvantages:

- Requires considerable expertise in risk-assessment.
- Not proven enough due to less employment!

UNIFIED PROCESS OUTLINE



The balance between workflows is different in different phases

UNIFIED PROCESS (UP)

- Current state-of-the-art methodology
- Initially developed by designers of UML
- Structures project as a number of phases
- Each phase contains several iterations
- Different workflows (activities) are performed in each iteration

UNIFIED PROCESS

Inception:

- Establish the project scope and boundary conditions
- Outline the use cases and key requirements for design tradeoffs
- Outline one or more candidate architectures
- Identify risks
- Prepare a preliminary project schedule and cost estimate

Elaboration:

- Capture requirements
- Design software architecture
- Plan for construction phase

Construction (use UML diagrams)

Transition

- Deployment of initial release

UNIFIED PROCESS

Advantages:

- Well-documented and complete methodology
- Adapts easily to changing requirements
- Reduced integration time and effort
- Higher level of reuse

UNIFIED PROCESS

Disadvantages:

- The process is too complex
- Not natural way of developing code
- Disorganised development
- Continuous integration might be challenging

UML AND THE UNIFIED PROCESS

UML diagrams can be used in conjunction with many different processes (or even in the absence of a formal process) however there is a close fit between UML and the UP.

AGILE MODELS

- Agile movement proposes alternatives to traditional project management.
- Agile approaches are typically used in software development to help businesses respond to unpredictability.

DID I REMEMBER TO TELL YOU BEFORE YOU FINISHED THE CODING THAT THE USER'S SPECIFICATIONS CHANGED?
Dilbert.com DilbertCartoonist@gmail.com
AAAIII-YIIII-YIIII-YIIII-YIIIIII!!!!!!
5-16-11 ©2011 Scott Adams, Inc. Dist. by Universal Uclick.
SO, NO-ISH?
BAM! BAM! BAM!

AGILE MODELS

Philosophy

- Encourages customer satisfaction and early incremental delivery of the software
- Small highly motivated project teams
- Informal methods
- Minimal software engineering work products
- Overall development simplicity

Development guidelines

- Stress delivery over analysis and design
- Active and continuous communication between developers and customers

AGILE MODELS

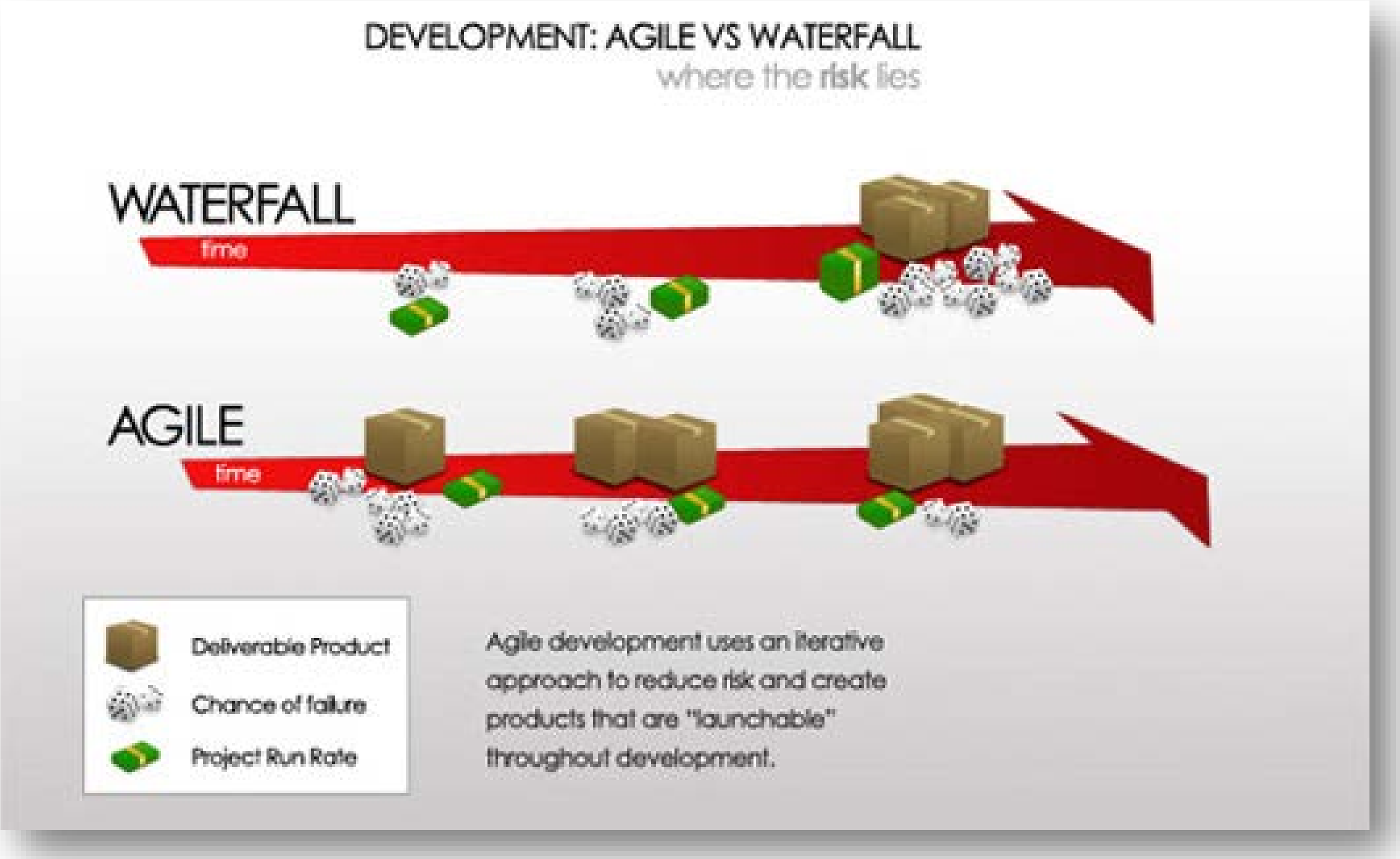
The Manifesto for Agile Software Development

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

AGILE MODELS

WE'RE GOING TO TRY SOMETHING CALLED AGILE PROGRAMMING.
www.dilbert.com scottadams@aol.com
THAT MEANS NO MORE PLANNING AND NO MORE DOCUMENTATION. JUST START WRITING CODE AND COMPLAINING.
11-26-07 © 2007 Scott Adams, Inc./Dist. by UFS, Inc.
I'M GLAD IT HAS A NAME.
THAT WAS YOUR TRAINING.

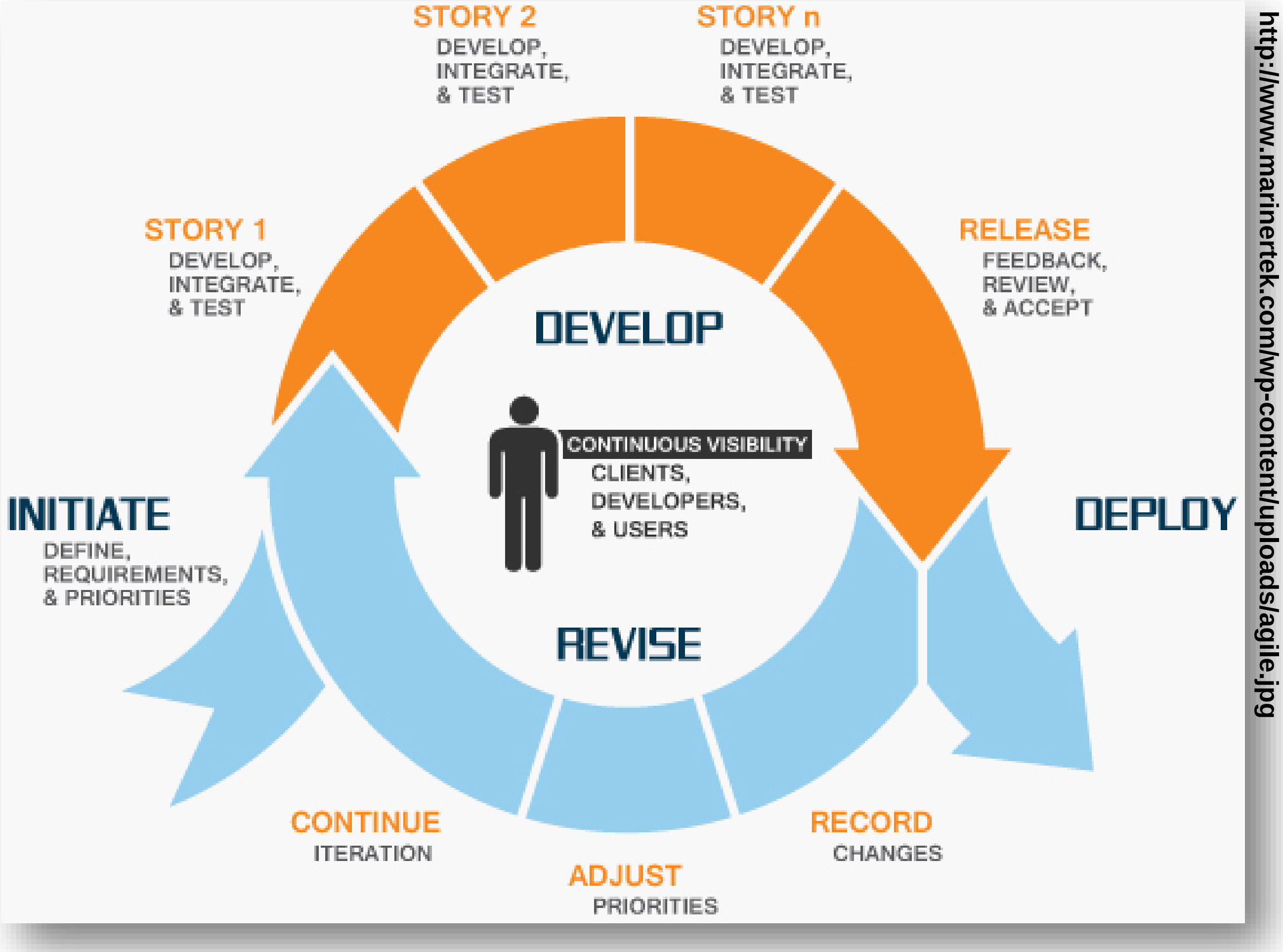
AGILE MODELS



DEVELOPMENT: AGILE VS WATERFALL
where the risk lies

Agile development uses an iterative approach to reduce risk and create products that are "launchable" throughout development.

AGILE MODELS



http://www.marinertek.com/wp-content/uploads/agile.jpg

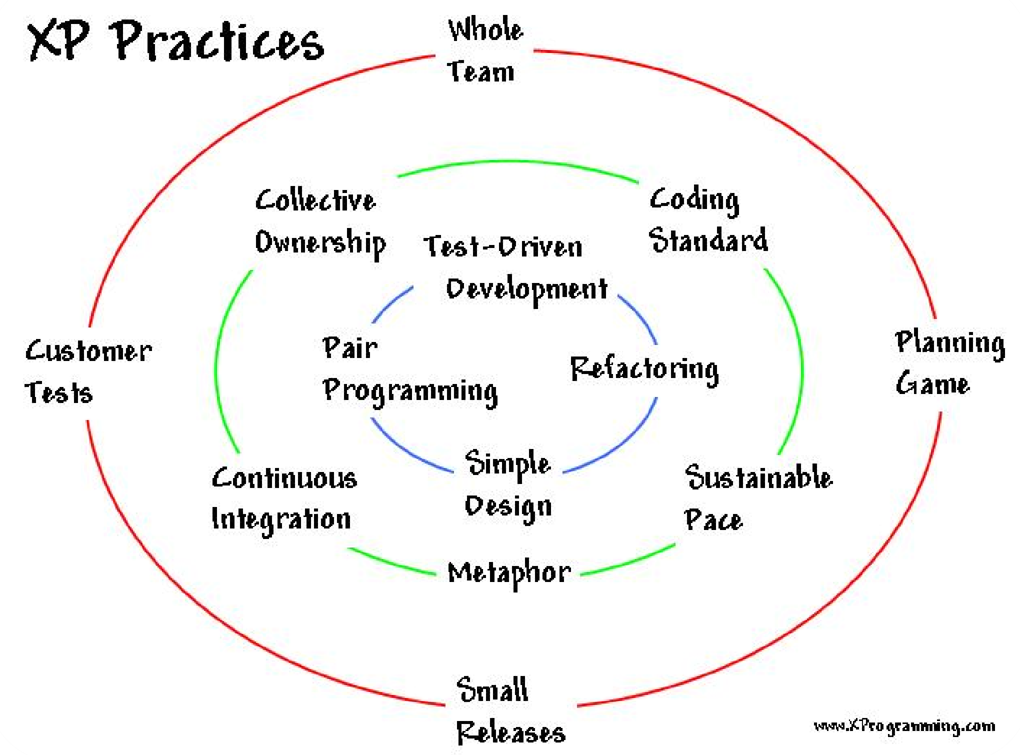
AGILE MODELS

There are many agile process models

- Extreme Programming (XP)
- Adaptive Software Development (ASD)
- Dynamic System Development Method (DSDM)
- Scrum
- Crystal
- Feature Driven Development (FDD)
- Agile Modeling (AM)

EXTREME PROGRAMMING: PRINCIPLES

- Plan to release in small increments
- Test first
- Keep it simple
- Own it collectively
- Code to standards
- Integrate continuously
- Refactor
- Program in pairs



EXTREME PROGRAMMING: PLANNING AND MANAGEMENT

- Initial brief ‘prototype’ phase
- Quickly determine scope of next release
- Put simple system into production quickly, then release new versions on a short cycle.
- Keep meetings short but frequent
- Involve the customer throughout
- Don’t burn out
- Embrace change – it will happen anyway

EXTREME PROGRAMMING: TESTING

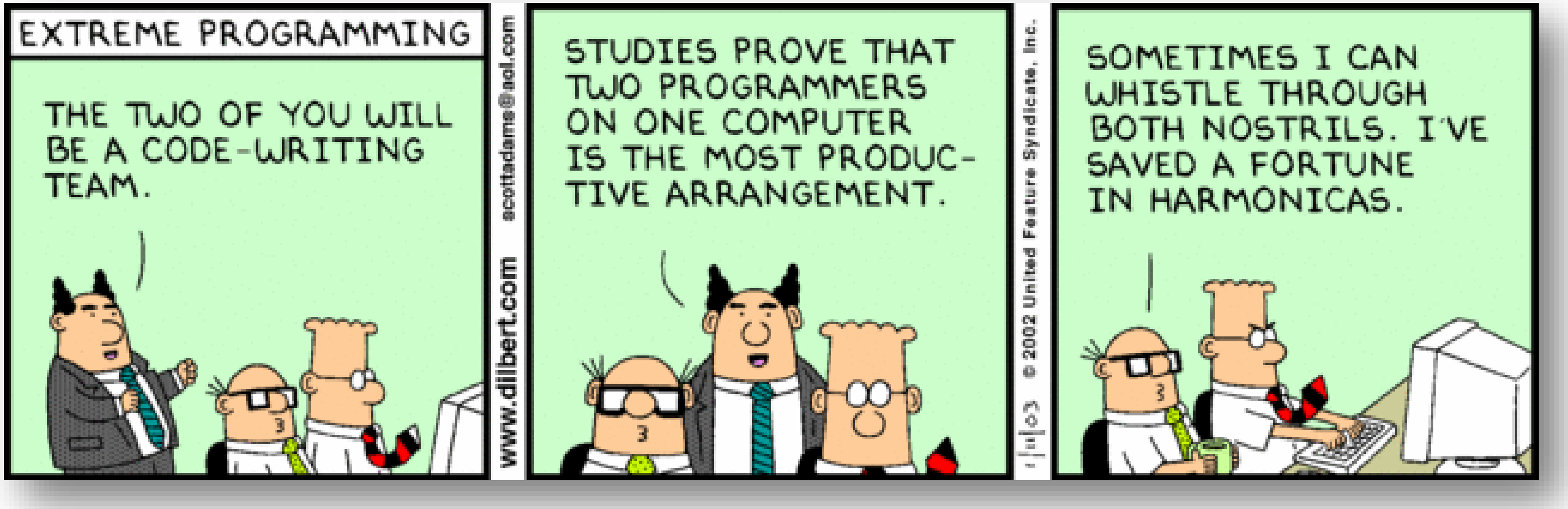
- Write unit tests for each method even before you start coding
- Tests provide a definition and documentation of the required behaviour
- Integrate and build the system every time a task is completed

EXTREME PROGRAMMING: KEEPING IT SIMPLE

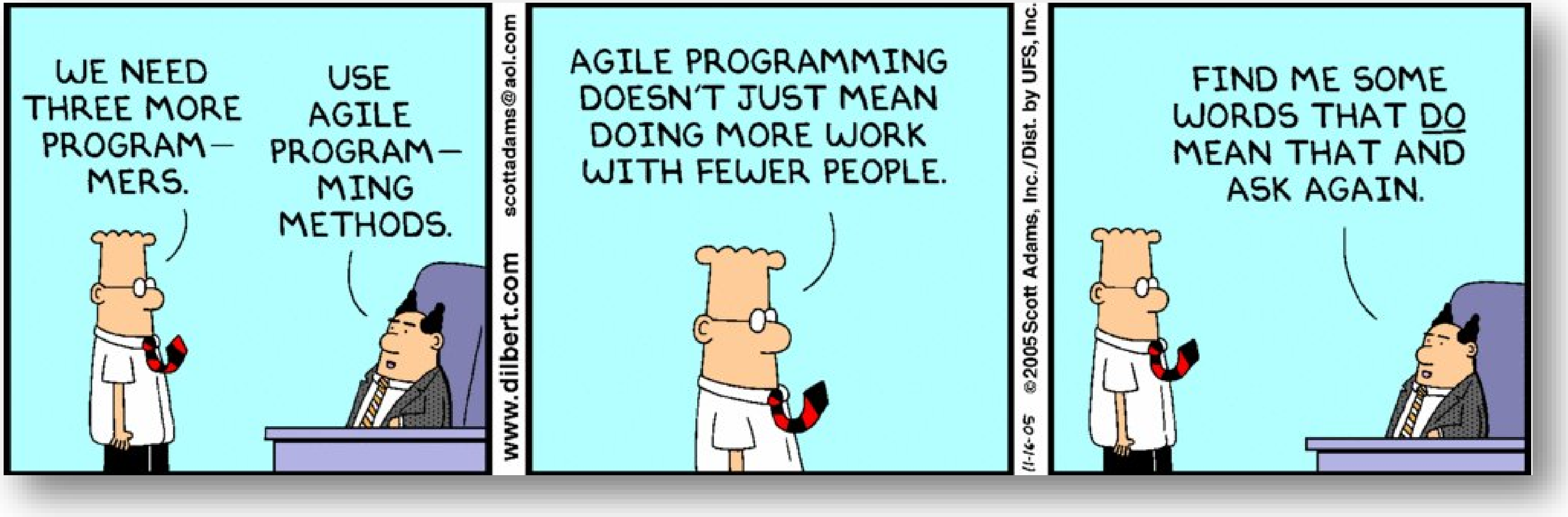
- System should be designed as simply as possible at any given moment
- Extra complexity is removed as soon as it is discovered
- Look for well known design patterns
- Refactor

EXTREME PROGRAMMING: CODING

- All production code written by two programmers at one machine
- Anyone can change any code anywhere in the system at any time
- All code written to agreed standard that emphasizes communication throughout


EXTREME PROGRAMMING
THE TWO OF YOU WILL BE A CODE-WRITING TEAM.
scottadams@aol.com
www.dilbert.com
STUDIES PROVE THAT TWO PROGRAMMERS ON ONE COMPUTER IS THE MOST PRODUCTIVE ARRANGEMENT.
© 2002 United Feature Syndicate, Inc.
1/11/03
SOMETIMES I CAN WHISTLE THROUGH BOTH NOSTRILS. I'VE SAVED A FORTUNE IN HARMONICAS.

EXTREME PROGRAMMING


WE NEED THREE MORE PROGRAMMERS.
USE AGILE PROGRAMMING METHODS.
scottadams@aol.com
www.dilbert.com
AGILE PROGRAMMING DOESN'T JUST MEAN DOING MORE WORK WITH FEWER PEOPLE.
©2005 Scott Adams, Inc./Dist. by UFS, Inc.
11-16-05
FIND ME SOME WORDS THAT DO MEAN THAT AND ASK AGAIN.

...REAL SOFTWARE LIFE-CYCLE?

Requirements capture	Lone genius has bright idea
Coding	Friends drop by with more bright ideas
Test and trial	Someone in accounts wants to play with it
User documentation	Someone in accounts can't remember how to drive it
Architectural design	You must be joking
Detailed design	" "
Functional specification	'Manager wants to know what you've been doing for the past year'
Requirements definition	'Manager wants to know what xxxxx use it is'
Maintenance	Impossible

LESSON SUMMARY

- Some kind of defined life-cycle needed
- Classic waterfall is dead
- Different life-cycle models contain common key activities.
- Prototyping and evolutionary development approaches allow more user feedback
- Extreme programming is suitable for group project