THE SOFTWARE PROCESS

Chapter 2

Software Engineering

Computer Science Engineering School

DSIC – UPV

Goals

- Define term "Software Process"
- Present main development process models that have been proposed
- Introduce the notion of methodology, presenting the main features of agile methodologies.

Contents

1. Introduction. The Software Process
2. LifeCycles
    - Classic or Waterfall
    - Classic with Prototyping
    - Automatic Code Generation
    - Incremental
    - Spiral
3. Methodologies
    - Agile Methodologies
    - RUP

References

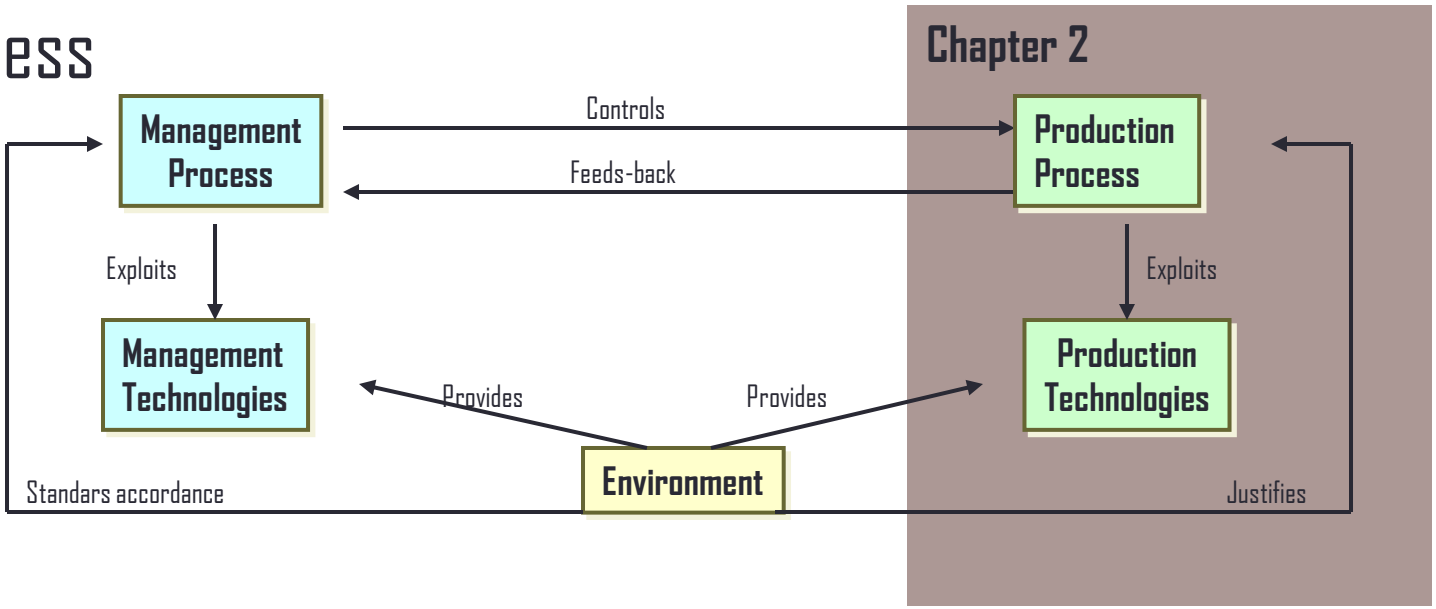
- Sommerville, I. Ingeniería del Software. (8ª ed.). Addison-Wesley, 2008
- Presman, R.S., Ingeniería del Software: un enfoque práctico (6ª ed.), McGraw-Hill, 2005
- Royce, W.W., Managing the Development of Large Software Systems: Concepts and Techniques. Proc WESCON, 1970
- Agresti, W.W. Tutorial: New Paradigms for Software Development. IEEE Computer Society Press, 1986
- Balzer, R., Cheatman, T.E. and Green, C., Software Technology in the 1990's: Using a New Paradigm, IEEE Computer, Nov. 1983, pp. 39-45
- McDermid, J. And Rook, P., Software Development Process Models. Software Engineer's Reference Book, CRC Press, 1993
- Boehm, B.W.. A Spiral Model of Software Development and Enhancement, IEEE Computer, pages 61-72, May 1988.

References

- Krutchen, P., *The Rational Unified Process- An Introduction*. Addison –Wesley, 1998
- Jacobson, G. Booch and J. Rumbaugh., *The Unified Software Development Process*, Addison-Wesley, 1999
- Beck, K., Extreme Programming Explained: Embrace Change. The XP Series. Addison-Wesley, 2000

The Software process

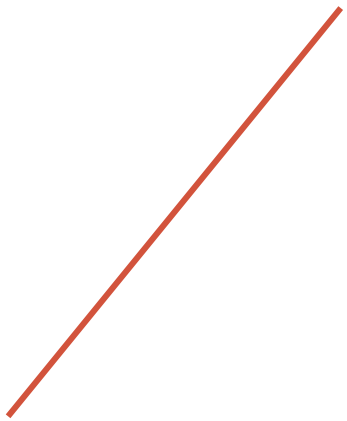
- It is a framework for the development of software
- In general the term "Software Process" is associated to the production process... but it includes the management process



The Development Process

- Collection of activities towards the development or evolution of software
- Also known as **Lifecycle**

- **Generic Activities** that are always carried out:
  - Specification
  - Development
  - Validation
  - Evolution

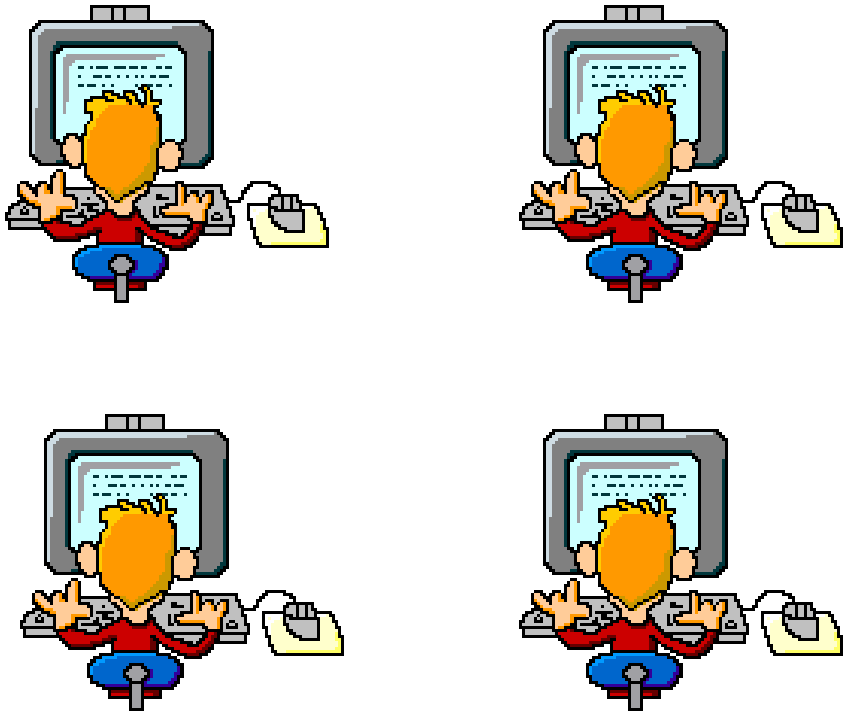


- Analysis
- Design
- Implementation
- Testing
- Maintenance

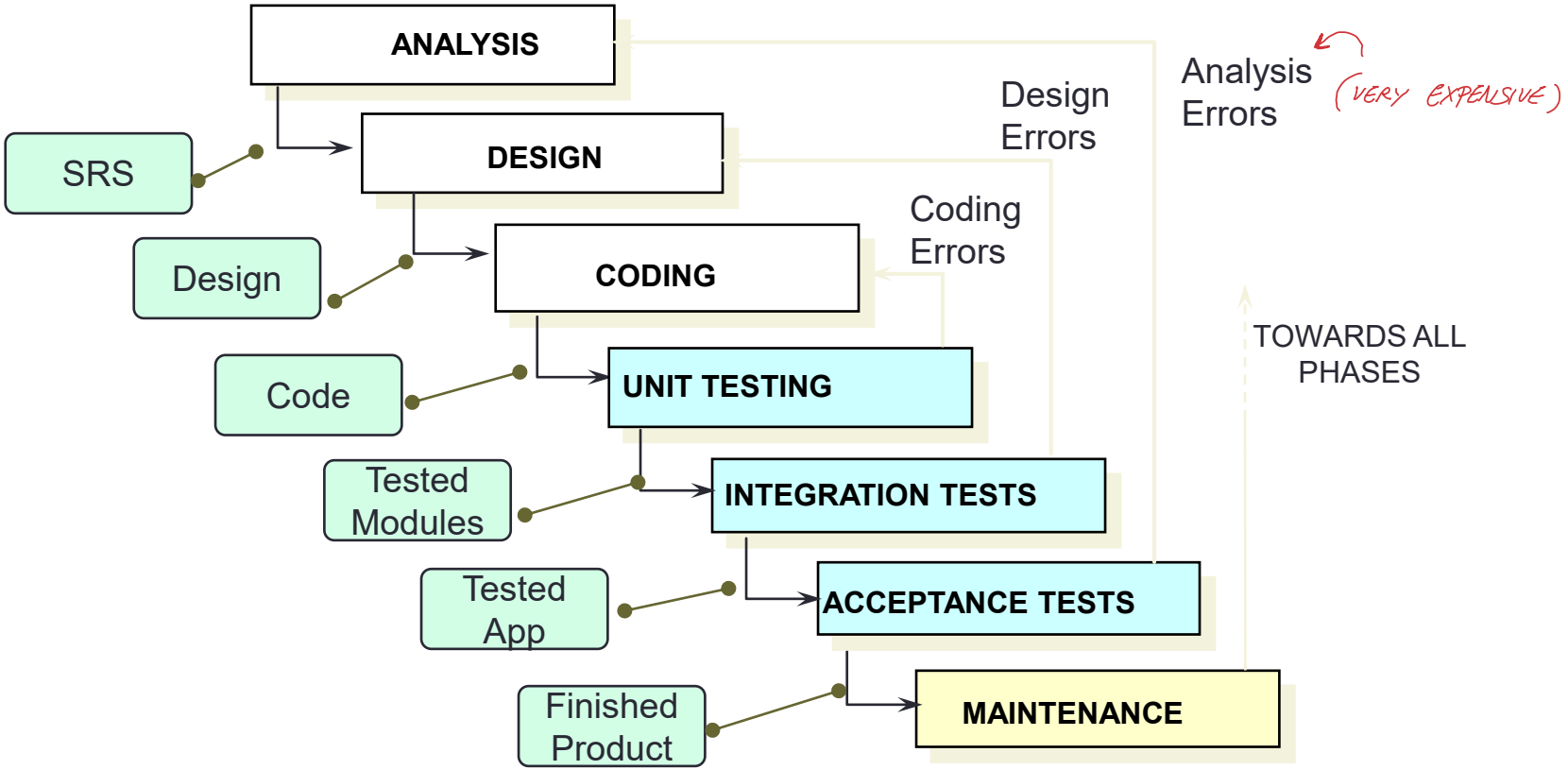
Lifecycle Models

- *Code-and-fix* : Only 1 phase. Used by telco.
- Classic or Waterfall : 3 phases (ANALYSIS, DESIGN, CODING). In each phase, design an artifact.
- Classic with prototyping
- Automatic Code Generation
- Evolutionary Models:
  - Incremental
  - Spiral

Code and Fix

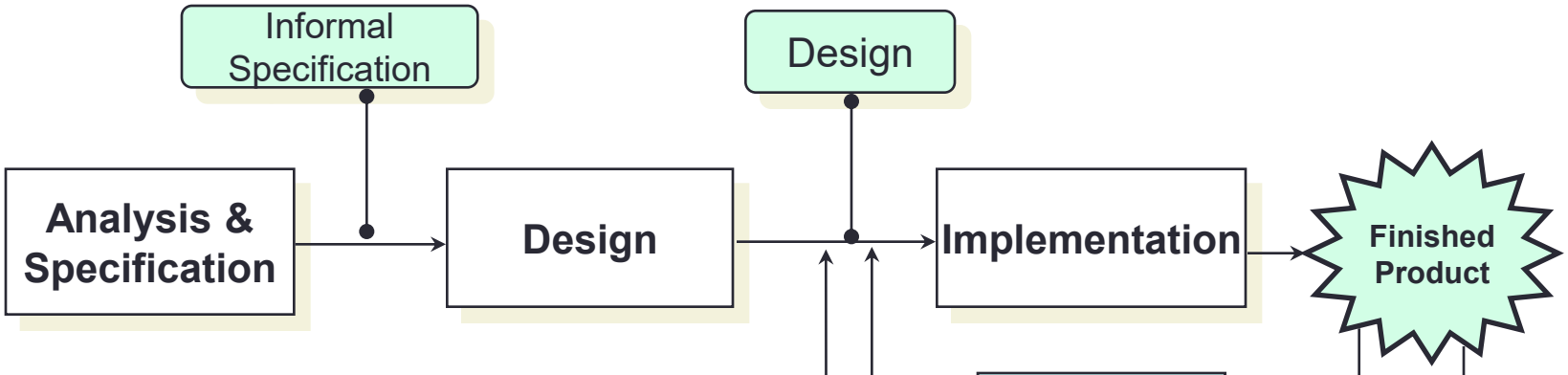


Classic or Waterfall



Classic or Waterfall

- In practice this model is "distorted" and all the validation and maintenance is performed on the source code.



→ Bad, as the code ends up not representing the Analysis phase notes / specifications

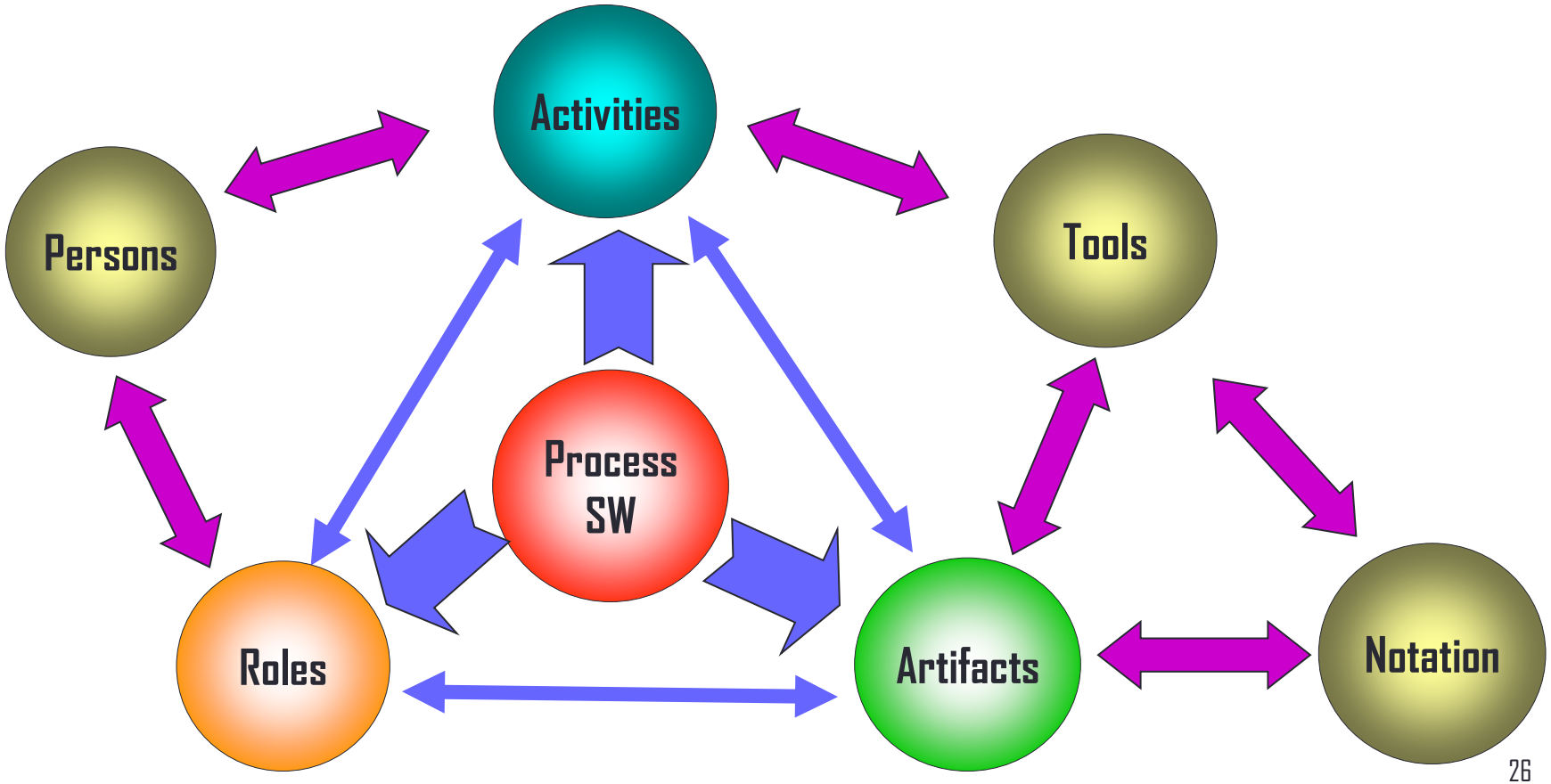
⇓

IDEA: Involve the customer in each phase ⇒ BUT ⇒ You won't see anything until all phases are done ⇒ Create a

Methodology

Difference between lifecycle model & methodology
- lifecycle model: defines the general, abstract phases to follow
- methodology: must define ALL details (tools, artifacts generated, organization, goals...)

- In a software development project, the methodology defines: Who/ What / How / When



Methodology

- Defines an explicit process of software development

  (its goal is the formalization of activities related with the elaboration of information systems)

- This process must be:
  - Reproducible
  - Defined
  - Measurable with respect to performance
  - Subject to Optimizations
  - ...

Methodology

There is no universal software methodology.

Structured methodologies

Object oriented methodologies

Rational Unifying Process

RUP

→ Heavy, must produce LOTS of artifacts even before starting coding
→ Not good for small teams.

Traditional methodologies **vs.** *Agile methodologies*

RUP **XP**

Agile Methodologies

Agile Methodologies appreciate:

- The individual and the interactions within the development team more than the activities and the tools
- The development of software that works rather than obtaining a good documentation ⇒ Minimalistic approach wrt modelling and documentation of the system

(each 2 weeks customer sees the development)

- The collaboration with the customer rather than the negotiation of a contract
- The fast response to changes rather than following a strict planning

http:\\www.agilealliance.com

Agile Methodologies

Principles of Agile Methodologies (1/2)

1.- The main priority is to satisfy the customer with early and continuous releases of usable software.

2.- Welcome changes. Agile processes apply updates for the customer to remain competitive.

3.- Release the developed software frequently and with the shortest possible interval of time between releases

4.- Business people and developers work together as a team in a project

5.- Build project driven by personal motivations. Provide the environment that people need and trust them.

Agile Methodologies

SCRUM: stablishes what face-to-face meetings should be done

Principles of Agile Methodologies (2/2)

6.- Face to face dialogue is the most efficient and effective method to communicate information within a development team

7.- Developed software is the first metric of progress

8.- Agile processes promote a bearable development. Funding entities, developers and users are capable of keeping a peaceful ambient

9.- The continuous attention to technical quality and good design increases agility

10.- Simplicity is key

11.- The best architectures, requirements and designs arise from the organization of the team ≈ Best arch arise from analysing NON-Functional requirements

12.- At regular intervals, the team reflects about how to be more effective and how to synchronize and adjust their work. (ex: SCRUM meetings)

Agile Methodologies

- Comparative

Agile Methodology	Non Agile Methodology
The customer is part of the Development team (*on-site*)	The customer interacts with the team By means of meetings
Small teams (< 10 members) Working at the same place	Large teams
Few artifacts	More artifacts
Few roles	More roles
Less emphasis on the architecture X	The architecture is essential

ARCHITECTUAL DESIGN is key in agile aswell !!

Agile Methodologies

- Comparative

Agile Methodology	Non Agile Methodology
Heuristics (adaptable to changes, previous experience)	Rigurous
Tolerant with updates	Resistant to updates
Internally imposed (by the team)	Externally imposed
Less controlled process, with Few principles	Highly controlled process with many Policies and norms
No traditional contract or at least very flexible ×	There is a prefixed contract

We need a contract, although it may be more flexible

Main Agile methodologies

- Extreme Programming (XP) http://www.extremeprogramming.org
- SCRUM https://www.scrum.org/
- Feature-Driven Development (FDD) http://www.featuredrivendevelopment.com
  - http://csis.pace.edu/~marchese/CS616/Agile/FDD/fdd.pdf
  - https://apiumhub.com/tech-blog-barcelona/feature-driven-development/
- Crystal Methods http://alistair.cockburn.us/Crystal+methodologies
  - http://www.devx.com/architect/Article/32836/0/page/2
  - https://activecollab.com/blog/project-management/crystal-methods
  - https://www.researchgate.net/publication/234820806_Crystal_clear_a_human-powered_methodology_for_small_teams
- Adaptive Development Software (ADS) http://www.adaptivesd.com
- Dynamic Systems Development Method (DSDM) http://www.dsdm.org
  - https://www.agilebusiness.org/what-is-dsdm
- Lean Development (LD) http://www.poppendieck.com

ANNEX - Extreme Programming (XP)

Kent Beck, Ward Cunningham y Ron Jeffries

www.extremeprogramming.org

www.xprogramming.com

- Design for dynamic environments
- Ideal for small teams (<= 10 coders)
- Strongly oriented towards coding
- Emphasis on informal and verbal communication
- Other values: simplicity, feedback and courage

XP

Development Cycle

Stories, Iterations, Versions, Tasks and test cases

✓ The customer selects the **next version** to be built, choosing the **functional features** that he considers more valuable (known as **Stories**) from a set of possible stories, being informed about *costs* and the required *time* of their implementation.

✓ Coders **convert stories** into **tasks to be done** and then convert **tasks** into a **set of test cases** to demonstrate that the tasks have been completed.

✓ Working with a teammate, the coder **runs the test cases** and **updates the design (evolution)** trying to keep it simple.

XP

Planning

Small deliverables

40 hours weeks

Simple design

Coding in pairs

Laboratory

tests

Collective ownership

Metaphore

Refactoring

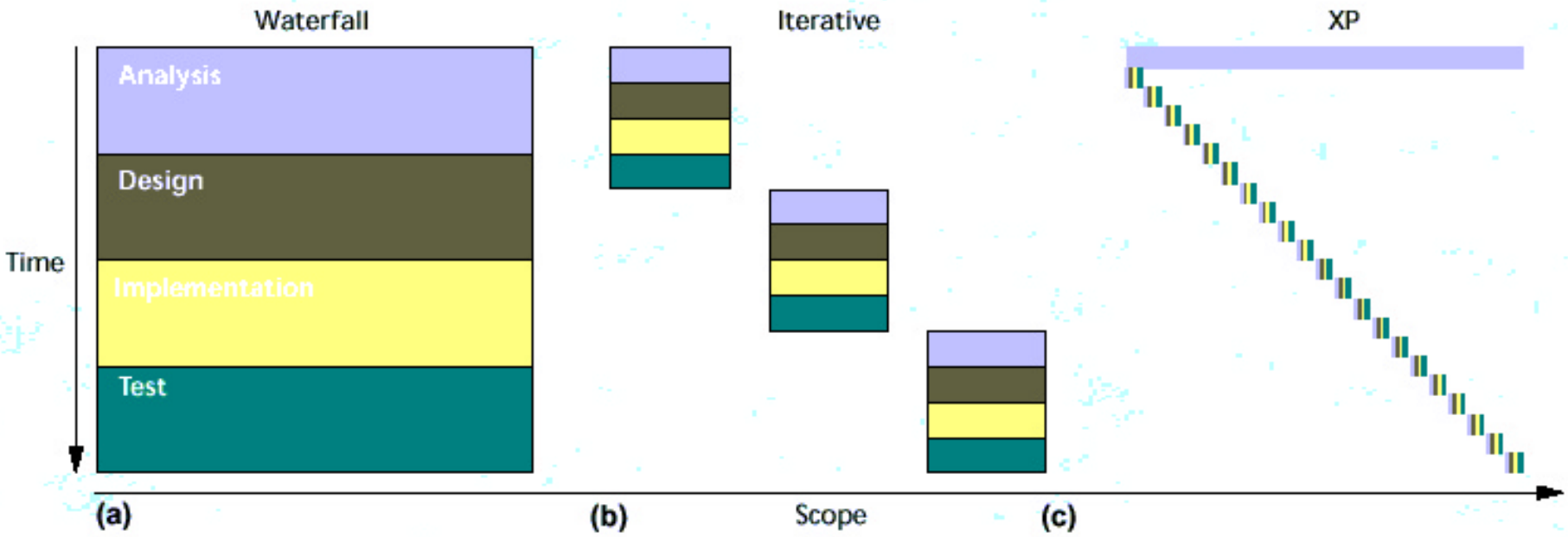
The customer always with the coder

Continuous integration

Coding standards

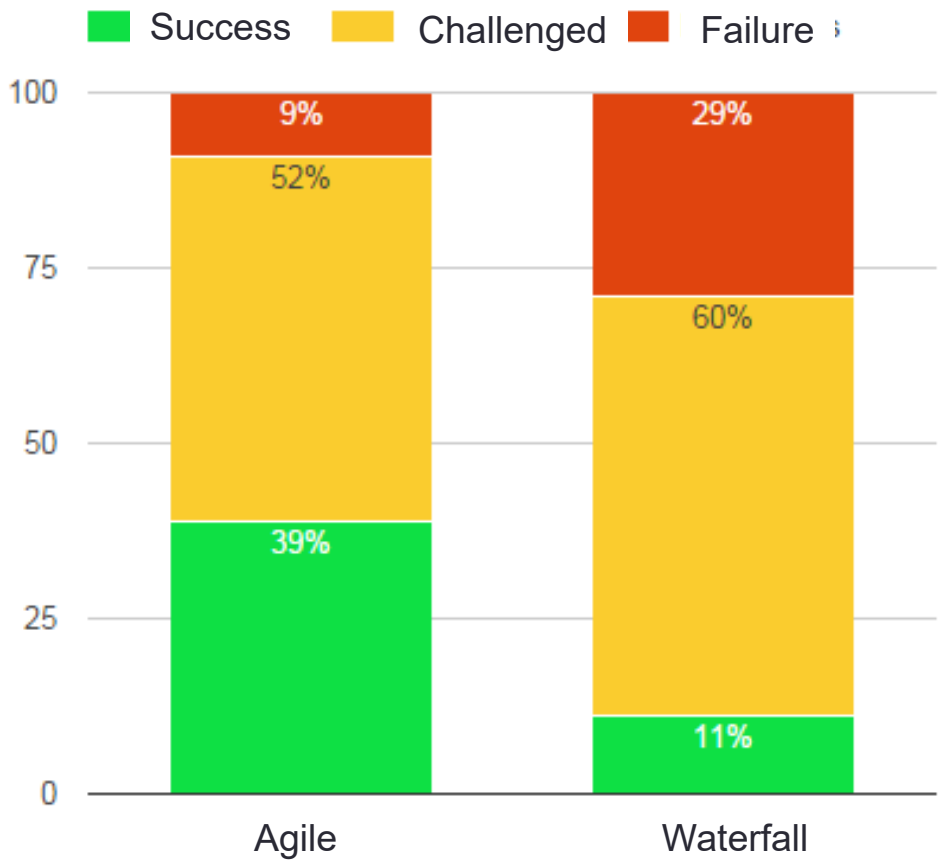
XP

Comparative



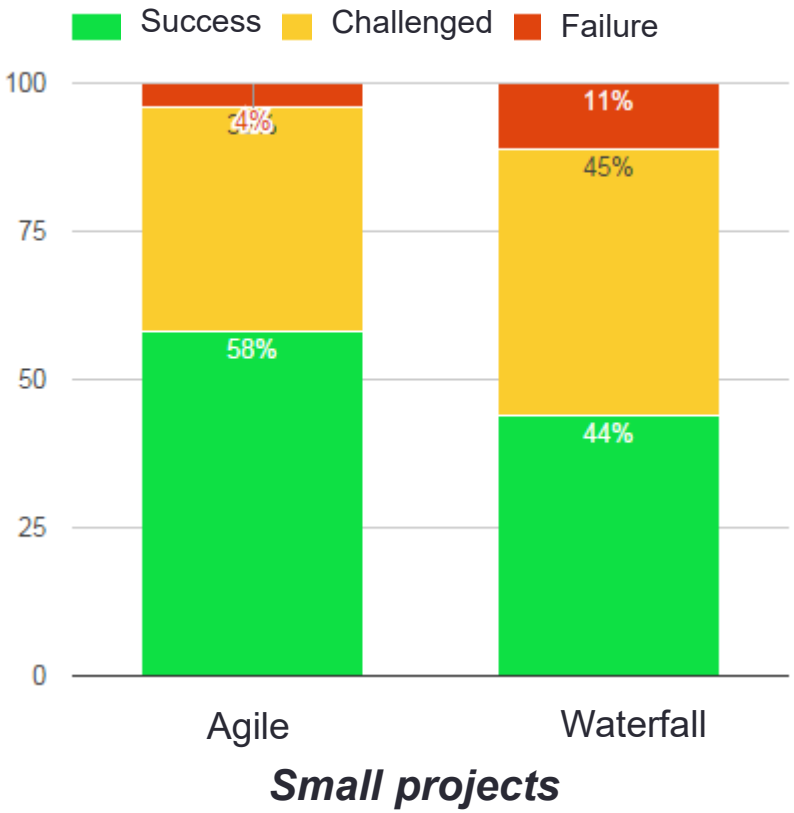
Agile vs. Waterfall

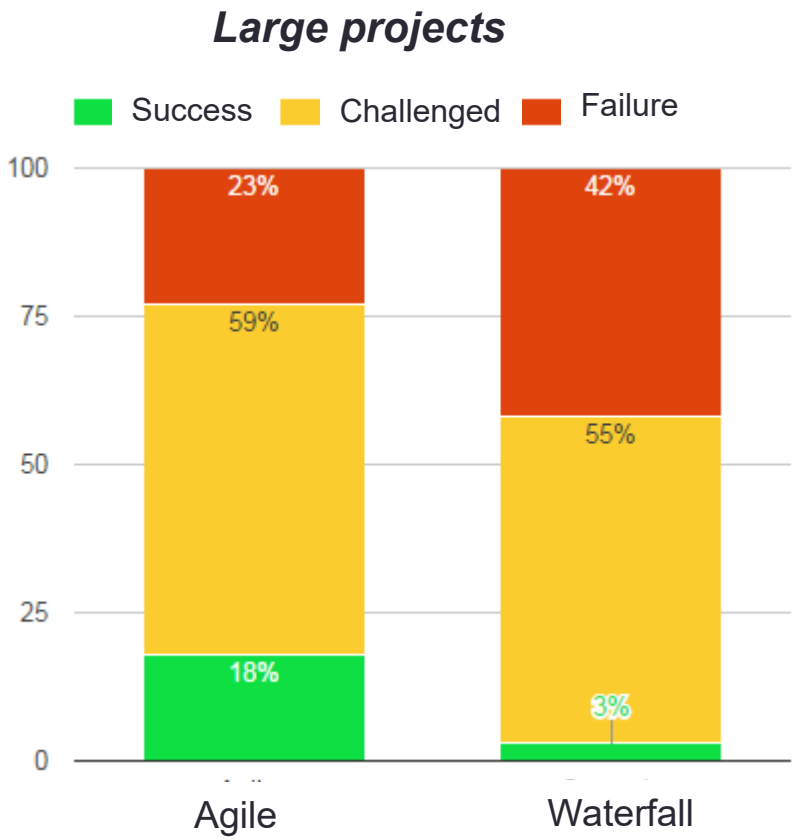
Success based on methodology
2011-2015



http://www.laboratorioti.com/2016/05/16/informe-del-caos-2015-chaos-report-2015-bien-mal-fueron-los-proyectos-ano-2015/

Agile vs. Waterfall





http://www.laboratorioti.com/2016/05/16/informe-del-caos-2015-chaos-report-2015-bien-mal-fueron-los-proyectos-ano-2015/

ANNEX - Rational Unified Process (RUP)

Software development process
(Rational – IBM)

Uses UML as modelling language

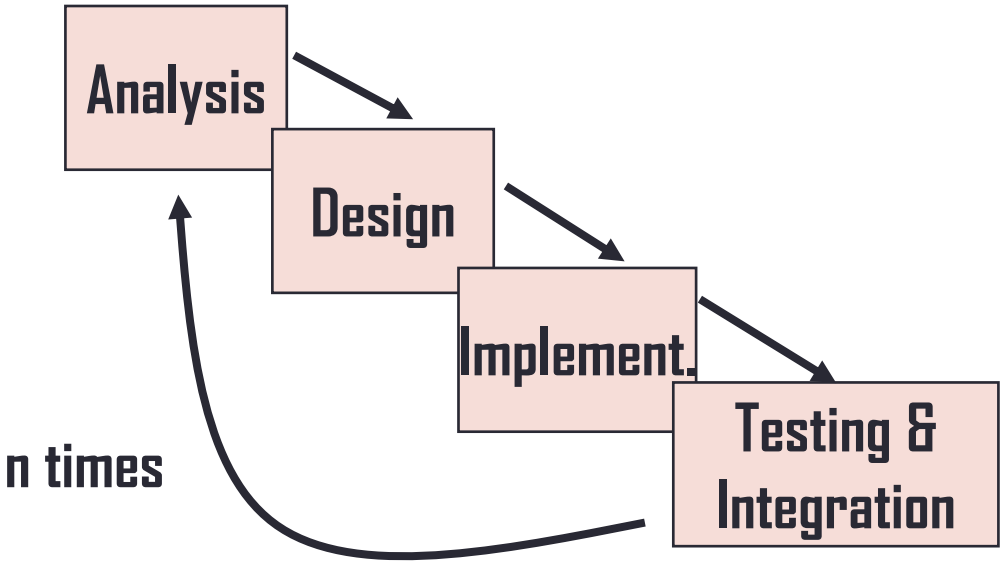
Features:

- *Use cases driven process*: from specification to maintenace
- *Iterative and incremental process:* iterations depending on the importance of use cases and the study of risks.
- *Architecture centered process*: reusable and serving as a guide towards the solution

RUP

- Iterative and Incremental

  Activities are performed in a mini-fall with a limited scope (the goals of the iteration)

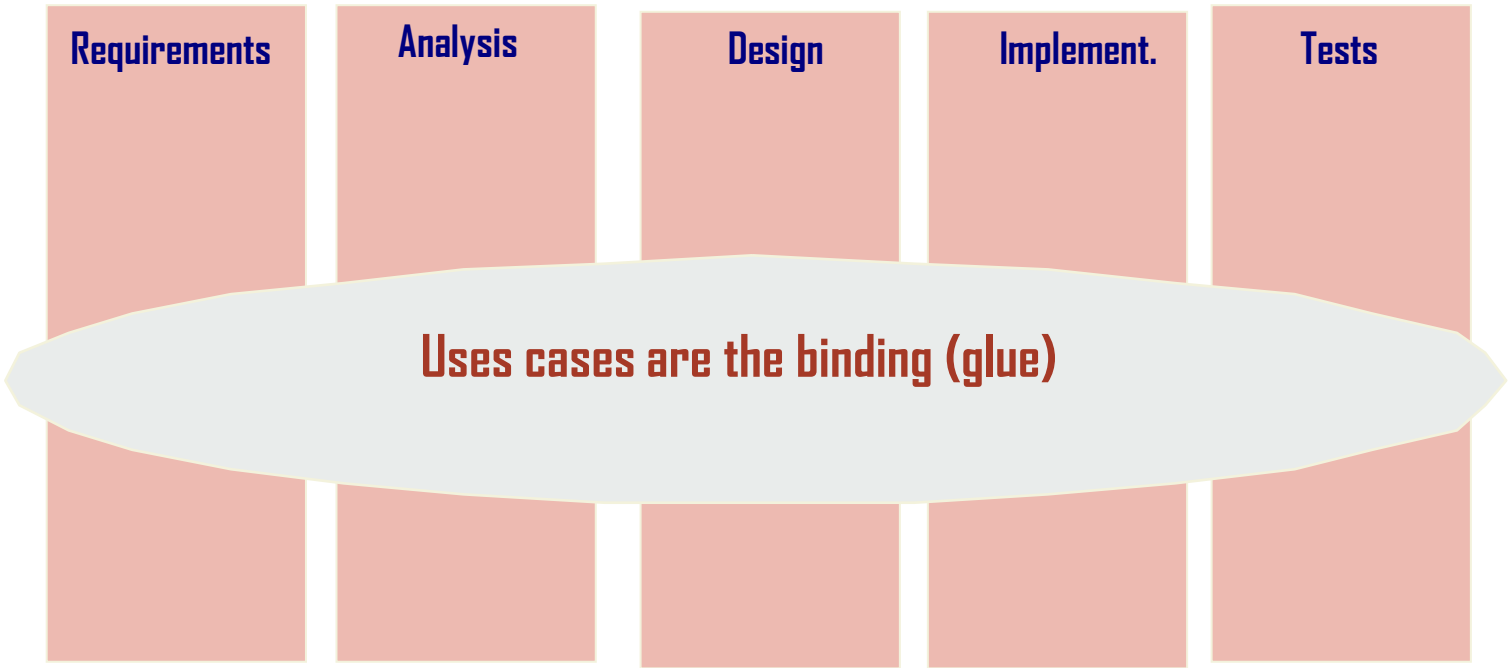


ACTIVITIES OF THE ITERATION

- Plan iteration (risks)
- Analysis of Use cases and Scenarios
- Design of Architectural choices
- Implementation
- Tests
- Integration
- Evaluation of release
- Preparation of release

RUP

- Use cases driven



Capture, clarify & validate use cases

Realize use cases

Verify whether use cases are satisfied

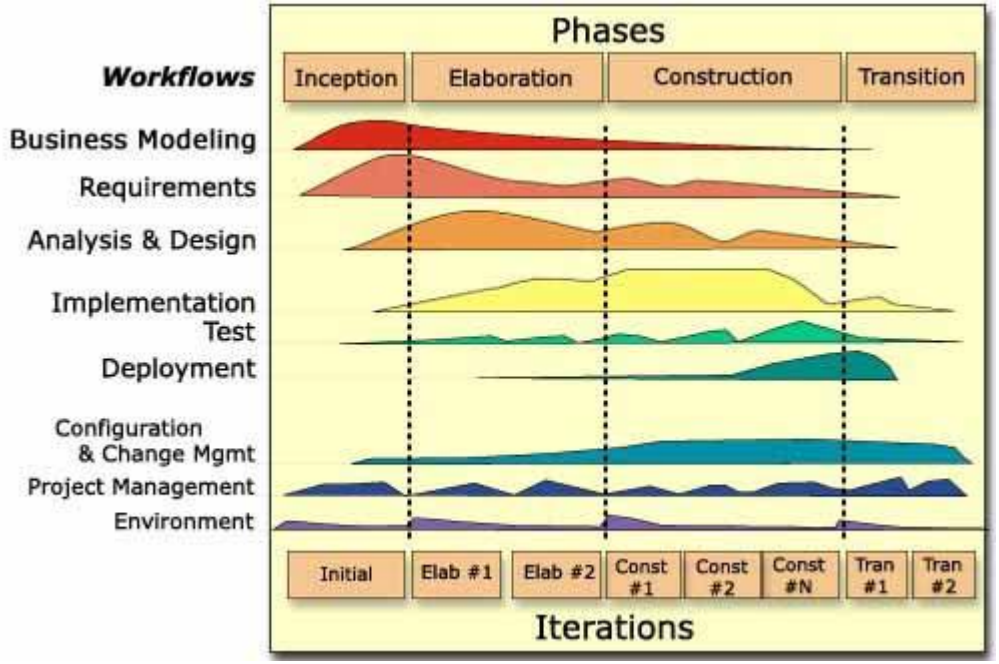
RUP

Dynamic View

Horizontal Axis: Time oriented organization

Static View

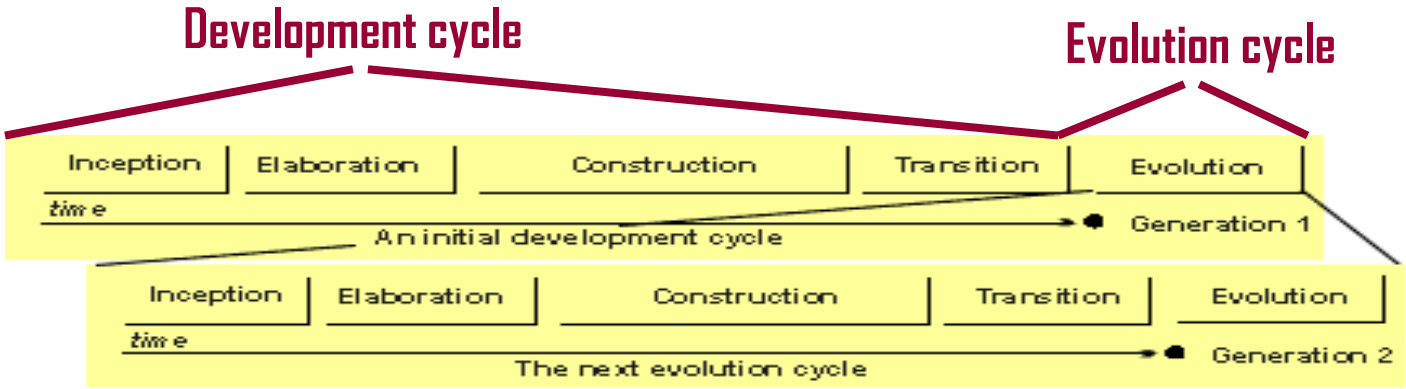
Vertical Axis:
Content oriented
organization

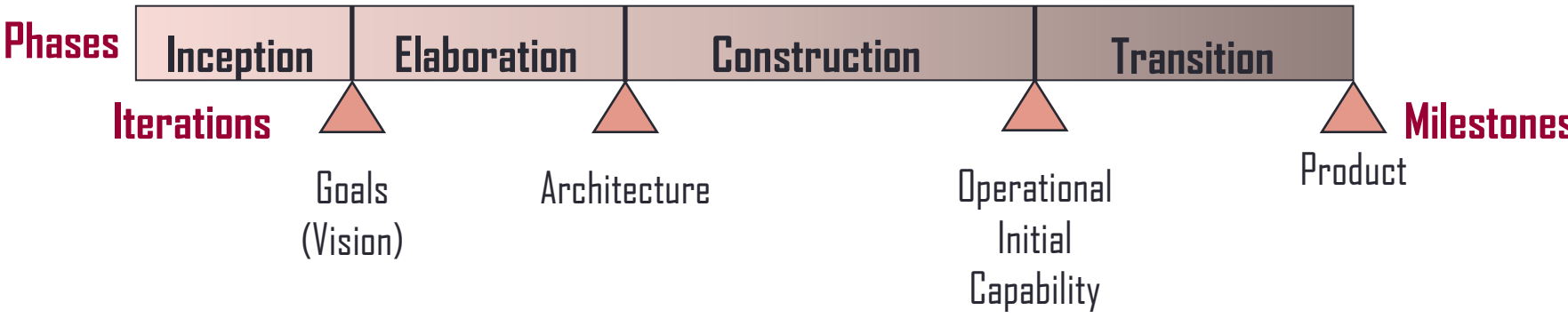


RUP

Dynamic View

- Cycles, Phases, Iterations and Milestones





RUP

Dynamic View

- Phases
  - *Inception(Opportunities Study)*
    - The scope and goals of the project are defined
    - The functionality and capabilities of the product are defined
  - *Elaboration*
    - The problem domain and the desired functionality are studied in depth
    - The basic architecture is defined
    - The project plan is defined according to the available resources

RUP

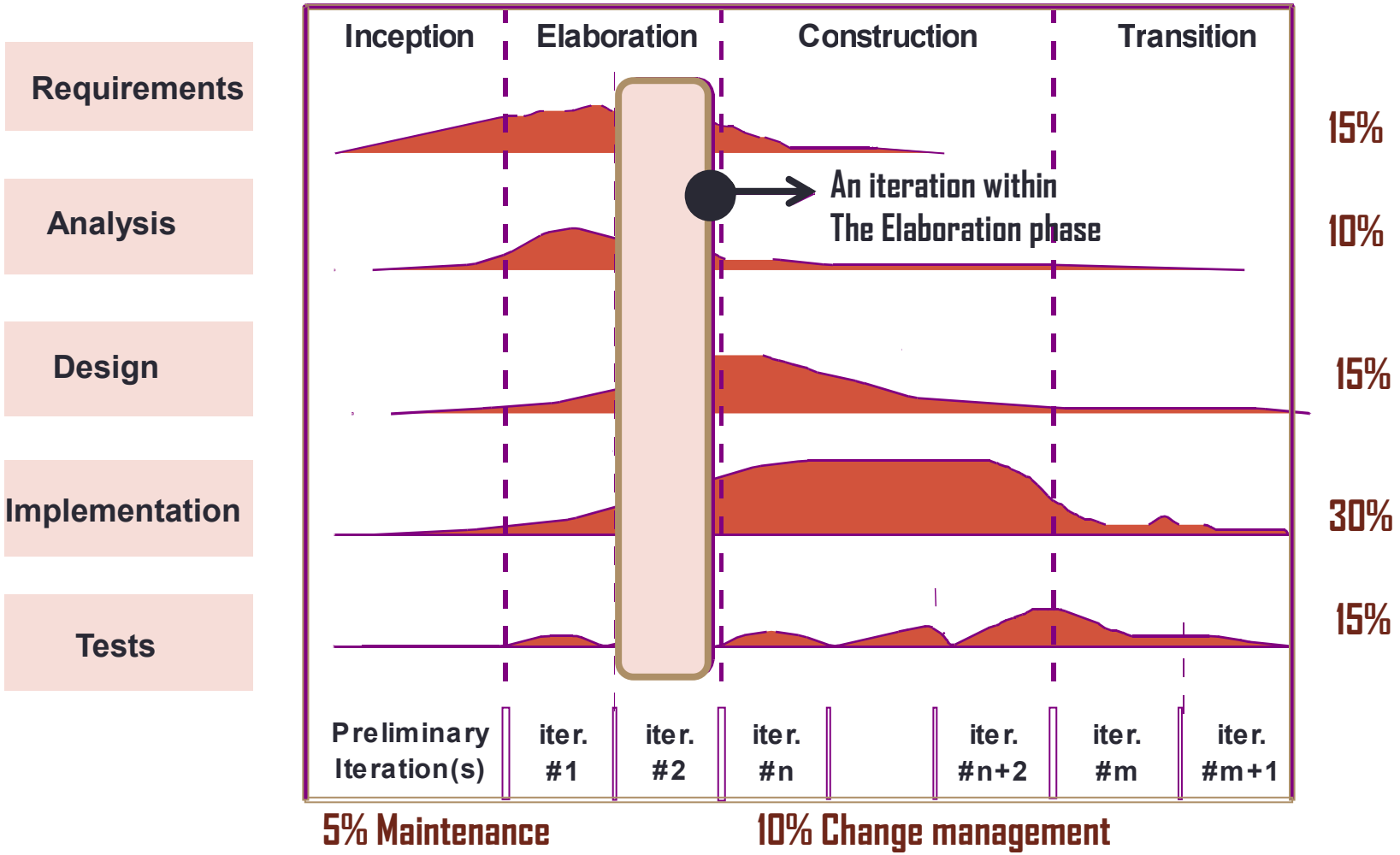
Dynamic View

- *Construction*
  - On each iteration analysis, design and implementation tasks are performed
  - The architecture is refined
  - An important part of the work is dedicated to coding and testing
  - The system and its use is documented
  - This phase provides a built product and a documentation

- *Transition*
  - The product is delivered to the user for its use
  - Marketing, packaging, installation, configuration, training, support and maintenance, ...
  - User, installation,... guides are completed and refined

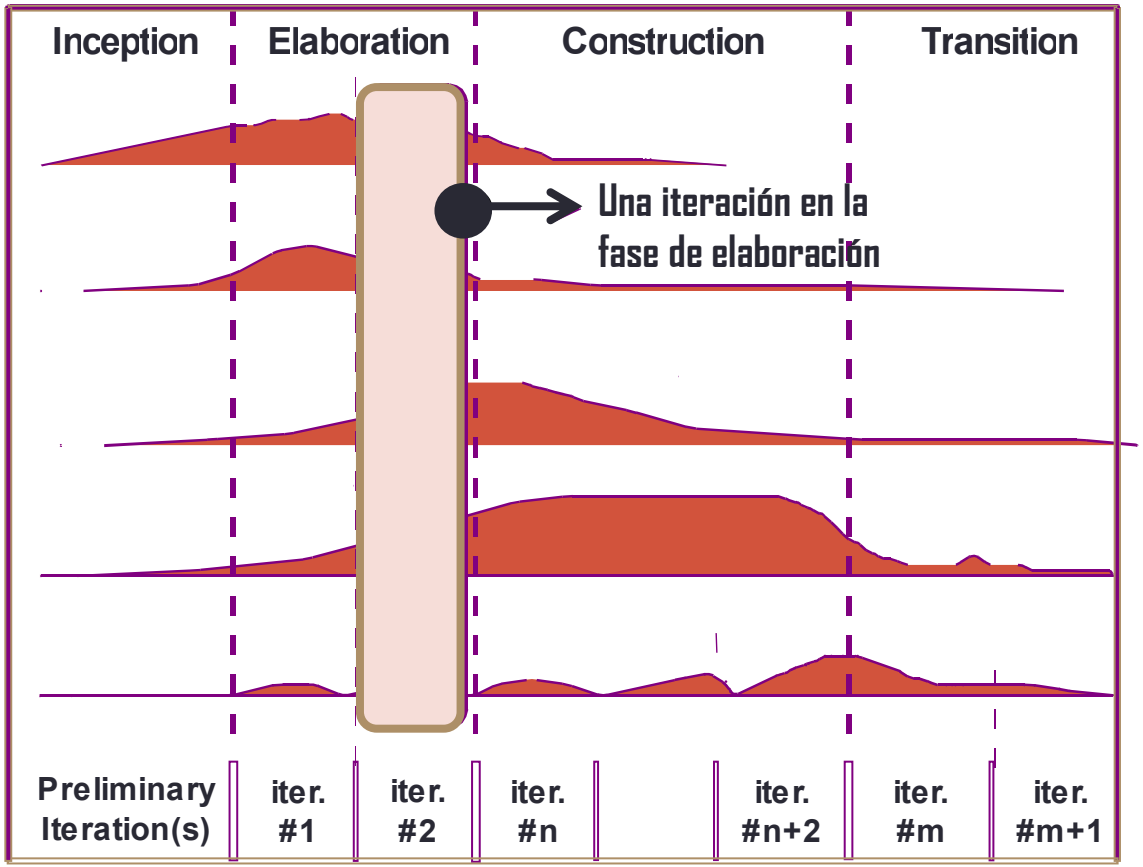
Dynamic View

RUP - *Distribution of **effort** with respect to activities*



Dynamic View

RUP - *Distribution of **effort** wrt phases*



Effort:	5%	20%	65%	10%
Duration:	10%	30%	50%	10%

RUP

Static View

- Workflows

Workflow	*Description*
Business Modelling	Business processes are modelled using business use cases
Requirements	Actors are defined that interact with the system and use cases are developed to model the requirements of the system
Analysis & Design	A design model is created using architectural models, component models, object models and interacion models.
Implementation	The different components of the system are structured and implemented. The automatic generation of code helps to speed up this process.
Tests	Testing is an iterative process that takes place simultaneously with the implementation. As soon as the implementation is finished the integration tests take place.
Deployment	A *release* (version) of the product is created, distributed to the users and installed in their workplace.

RUP

Static View

- Workflows

Workflow	*Description*
Configuration and Change Management	To manage changes in the system
Project Management	To manage the development of the system
Environments	Development of appropriate software development tools for development teams.