Métodos de Desenvolvimento de Software (MDS) 2016/2017

Vasco Amaral
vma@fct.unl.pt

Lecture 4

Process Models

The Software Process

- The set of activities and associated results which produce a software product
- The sequence of steps required to develop and maintain software
- Sets out the technical and management framework for applying **methods**, **tools** and **people** to the software task
- Definition:
  - The Software Process is a description of the process which guides software engineers as they work by identifying their roles and tasks.

The Software Process

- Fundamental Process Activities
  - Software Specification
  - Software Development
  - Software Validation
  - Software Evolution

Software Process: Software Specification (or requirements engineering)

- Understand and define what services are required from the system and identify constraints to the system's operation and development
- Leads to a requirements document which is the specification of the system
  - Customers and end-users get high level statements
  - System developers get the detailed system specification

Software Process: Software Specification (or requirements engineering)

Main phases:

- **Feasibility Studies** (leads to feasibility report)
- **Requirements Elicitation** and **Analysis** (may develop System Model and prototypes)
- **Requirements Specification** (leads to User and system requirements document)
- **Requirements Validation** (checks realism,consistency and completeness)

Software Process: Design and implementation

- Architectural Design
- Abstract Specification
- Interface Design
- Component Design
- Data Structures Design
- Algorithm Design

Software Process: Software Validation

- **Verification: are we building the system right?**
  - Look at the system's specification
- **Validation : are we building the right system?**
  - Meets the expectation of the customer

Stages:

- Component (or unit) Testing
- System Testing
- Acceptance Testing

Characteristics of a good process

- Understandability
- Visibility
- Supportability
- Acceptability
- Reliability
- Robustness
- Maintainability
- Rapidity

Steps in a Generic Software Process

- Project Definition
- Requirements Analysis
- Design
- Program Implementation
- Component Testing
- Integration Testing
- System Testing
- System Delivery
- Maintenance

Project Activities (1)

- **Project Definition**
  - States the purpose of the project
  - Makes initial decision on political and technical feasibility of the project
- **Requirements Analysis**
  - High level definition of the functionality of the system, primarily from the point of view of the users
- **Design**
  - Looks at the software requirements of the system and the architecture of the system
  - Lower level design activities - data structures, interface representations, procedural (algorithmic) details

Process Activities (2)

- **Program Implementation**
  - Writing or generating the code to build the system
- **Component Testing**
  - Testing of the individual components while they are being built and after they have been completed
- **Integration Testing**
  - Testing of the way individual components fit together
- **System Testing**
  - Testing of the whole system usually in concert with the users (acceptance testing)

Process Activities(3)

- **System Delivery**
  - Implementation of the system into the working environment and replacement of the existing system
- **Maintenance**
  - Corrective
  - Adaptive
  - Perfective

Software Process Model

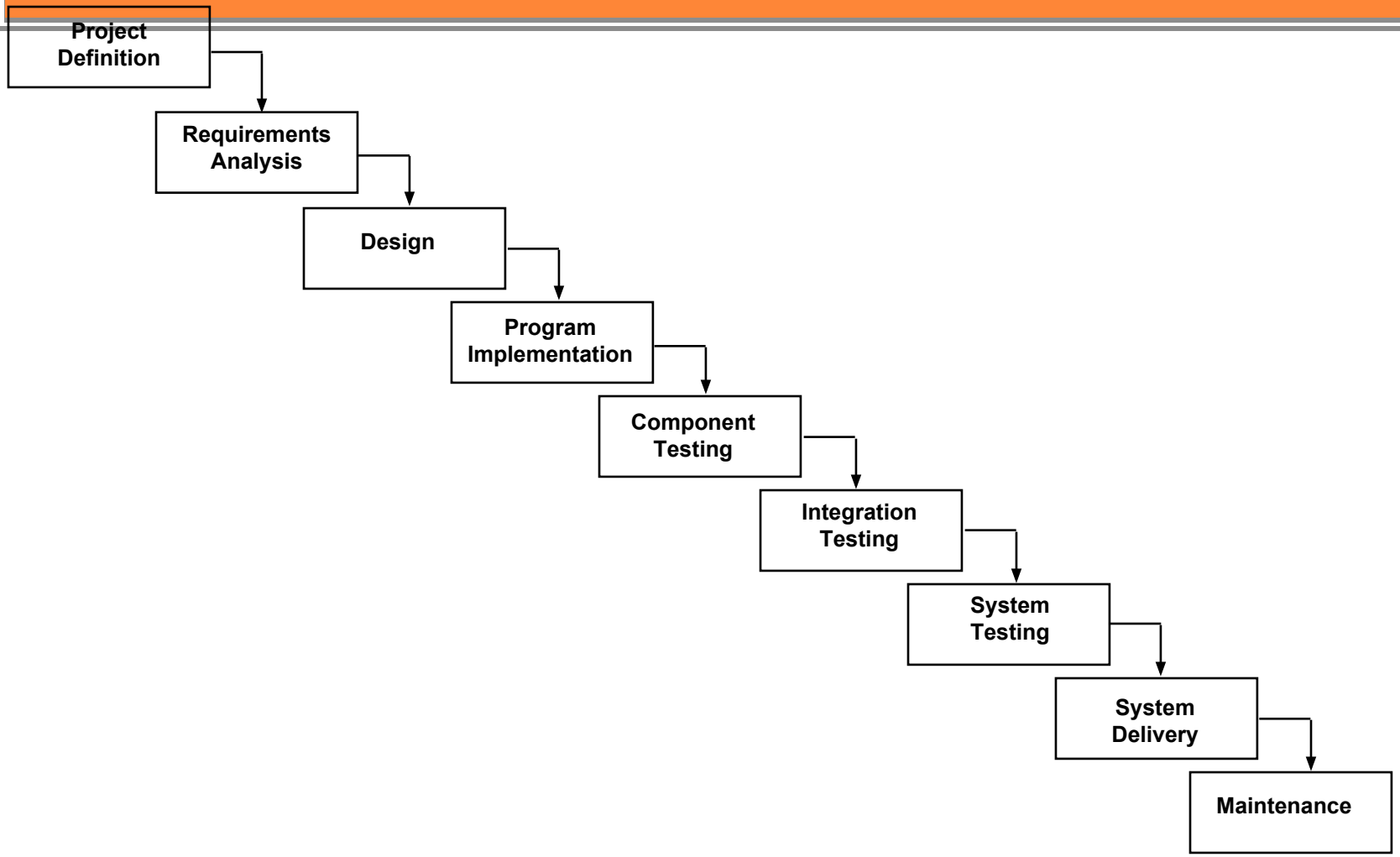
- Is a simplified description of a software process that presents one view of that process. Activities involve:
  - Workflow models
  - Dataflow or activity models
  - Role/action models

Rest of the Lecture Outline

- Traditional/Waterfall
- Incremental
- Prototyping
- Rapid Application Development (RAD)
- V-Model
- Evolutionary
  - Spiral
  - Component Assembly
- RUP and UML
- Agile Methods (e.g. XP)
- Formal Methods
- Fourth Generation Techniques

Prescriptive Models

The Waterfall Model



Waterfall Model - Plan driven

- Most widely used, though no longer state-of-the-art
- Each step results in documentation
- May be suitable for well-understood developments using familiar technology
- Not suited to new, different systems because of specification uncertainty
- Difficulty in accommodating change after the process has started
- Can accommodate iteration but indirectly
- Working version not available till late in process
- Often get blocking states

Waterfall Model

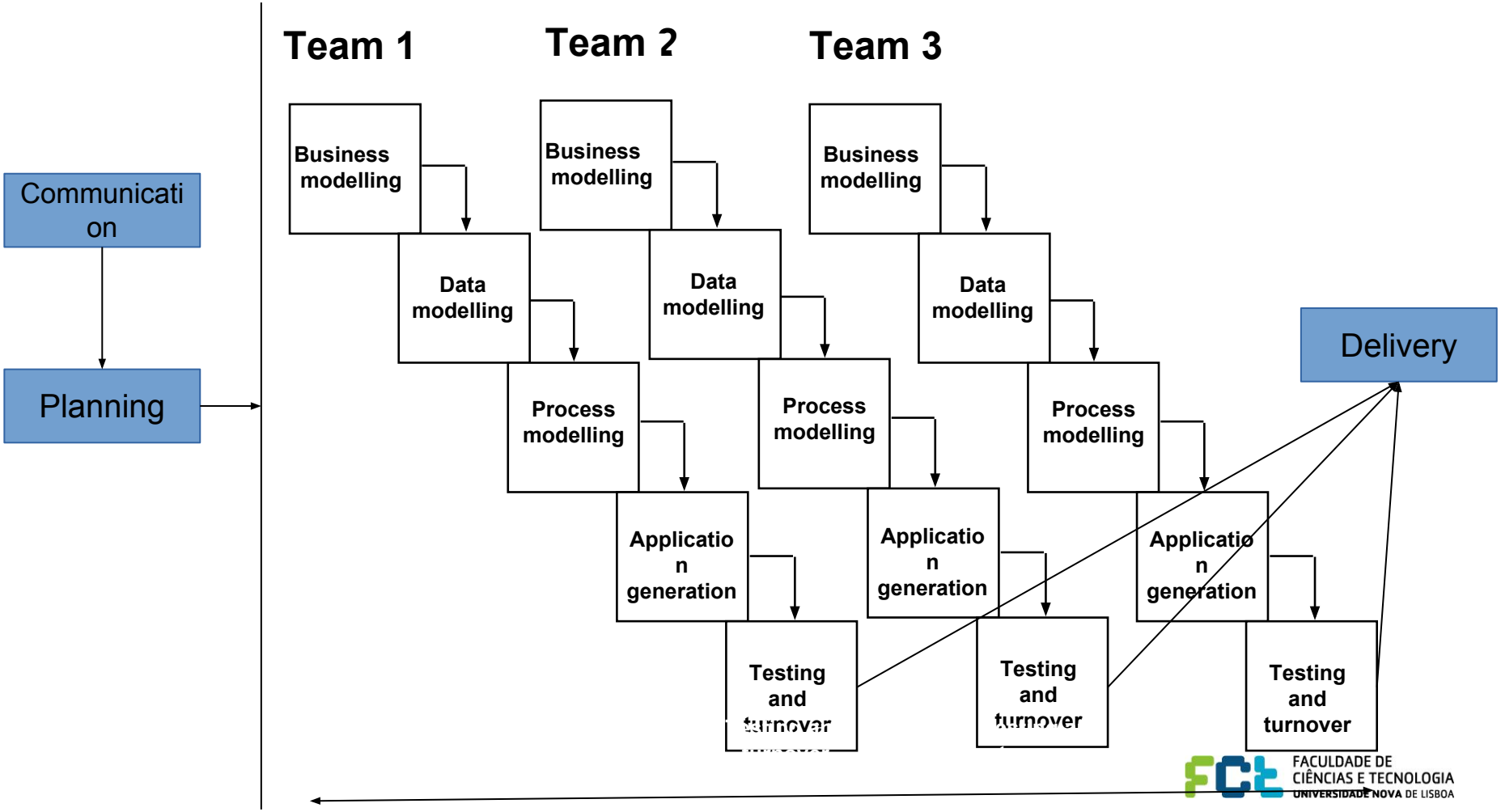
- However, it reflects the type of process model used in other engineering approaches
- Is still used when the software project is part of a larger system engineering project

- Adequate to Embedded, Critical and Large Systems

Rapid Application Development

- Similar to waterfall but uses a very short development cycle (60 to 90 days to completion)
- Uses component-based construction and emphasises reuse and code generation
- Use multiple teams on scalable projects
- Requires heavy resources
- Requires developers and customers who are heavily committed
- Performance can be a problem
- Difficult to use with new technology

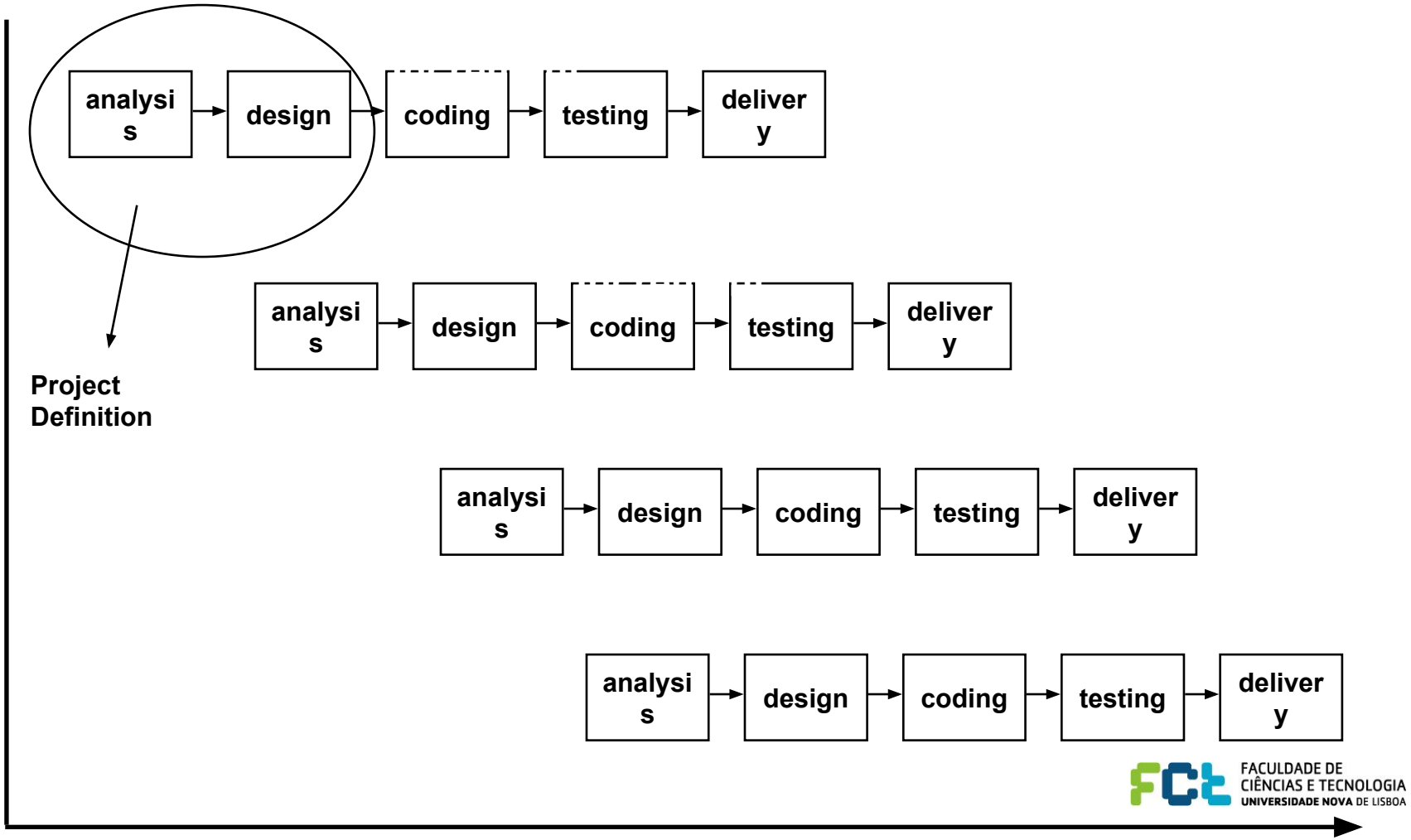
Rapid Application Development (RAD)



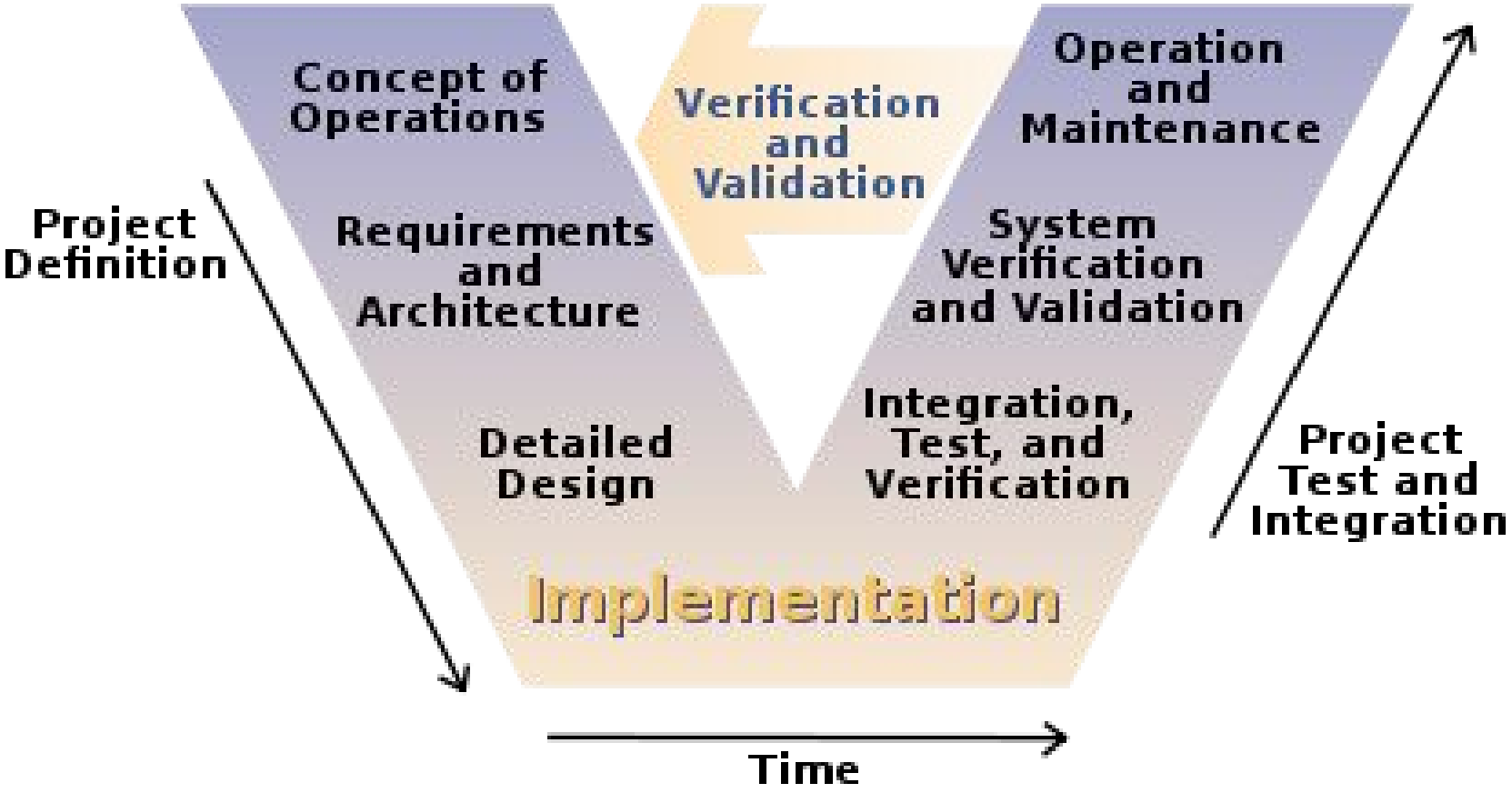
Rapid Application Development (RAD)

- For large projects requires sufficient human resources to create the right number of RAD teams
- If the system cannot be modularized, building the components necessary for RAD will be problematic
- If high performance is an issue, requiring to tune the several system components, it may not work
- It may not be appropriate when technical risks are high

Incremental Development



V-Model



*taken from the wikipedia

V-Model

Pros:

- Minimizes project risks - due to the explicit concerns:
  - Verification (Am I doing things right?)
  - Validation (Am I doing the right thing?)

- Improvement and guarantee of quality
- reduction of total cost over the entire project and systems life cycle

Cons:

Apart from the mentioned benefits it suffers from the same problems of the watterfal model

Incremental Development

- Applies an iterative philosophy to the waterfall model
- Divide functionality of system into increments and use a linear sequence of development on each increment
- First increment delivered is usually the *core product,* i.e only basic functionality
- Reviews of each increment impact on design of later increments
- Manages risk well
- Extreme Programming (XP), and other Agile Methods, are incremental, but they do not implement the waterfall model steps in the standard order

Incremental Process Model

- Combines elements of the waterfall applied iteratively
- Each linear sequence delivers deliverable operational product increments
- The first increment is a core product where requirements are met but supplementary features remain undelivered
- Ideal when the staff is unavailable for a complete implementation
- Increments can be planned to to manage technical risks

Incremental Process Model

- Positive:
  - Cost of implementing is reduced
  - easier to get customer feedback
  - early delivery and deployment of useful software
- Negative:
  - The process is not visible
  - system structure tends to degrade (Agile methods propose to often refactor)

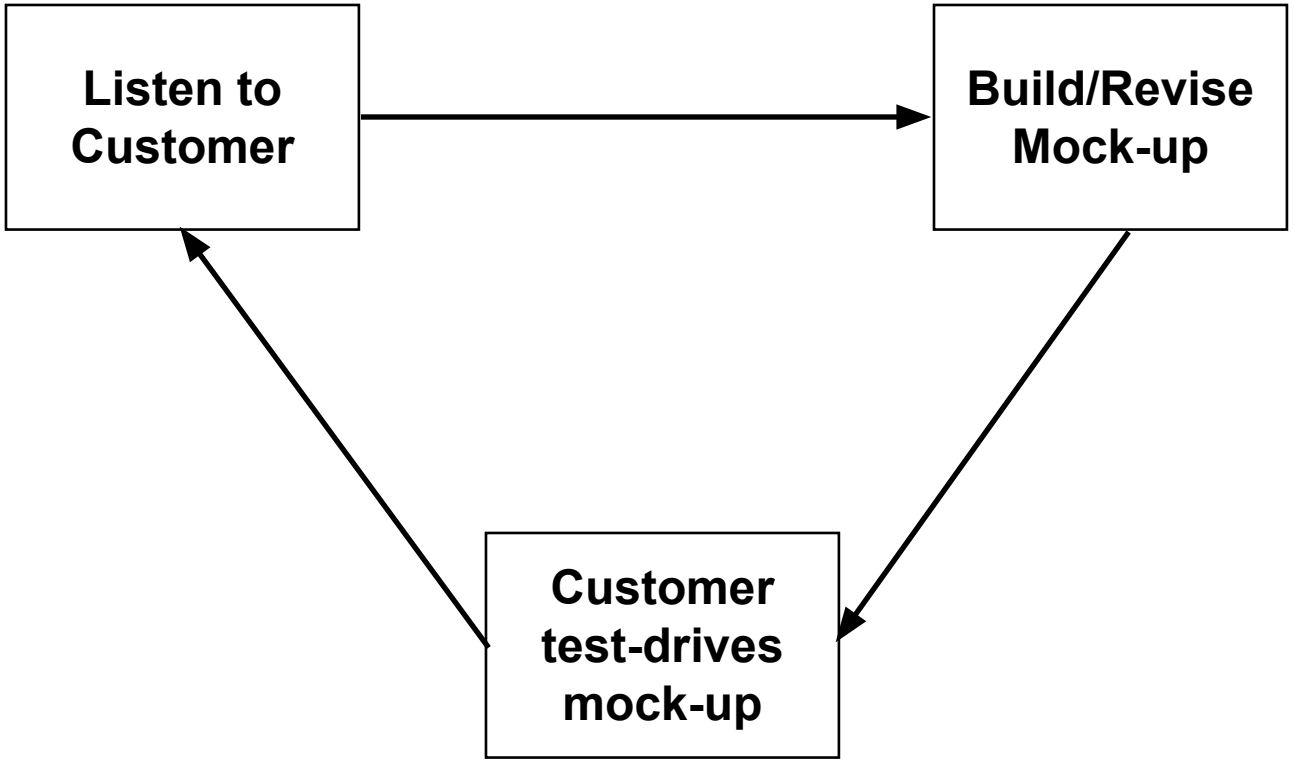
Evolutionary Process Models

- Exploratory development – work with the costumer to explore their requirements and deliver a final product, starting by the parts of the system that are understood. New proposed features are added.
- Throwaway prototyping - help on better understand the requirements and get a better requirements definition. The prototype concentrates on poorly understood requirements

Prototyping

- Specifying requirements is often very difficult
- Users don't know exactly what they want until they see it
- Prototyping involves building a mock-up of the system and using to obtain for user feedback
- Closely related to what are now called "Agile Methods"

Prototyping



Prototyping

- Ideally mock-up serves as mechanism for identifying requirements
- Users like the method, get a feeling for the actual system
- Less ideally may be the basis for completed product
  - prototypes often ignore quality/performance/maintenance issues
  - may create pressure from users on deliver earlier
  - may use a less-than-ideal platform to deliver e.g Visual Basic - excellent for prototyping, may not be as effective in actual operation

Prototyping

- The process is not visible – it is not cost effective to produce documents that reflect every version of the system
- Systems are often poorly structured – incorporating changes becomes increasingly costly and difficult
- Not adequate for large, complex long-lived systems with different teams developing different parts
- Difficult to establish a stable system architecture
- Usually should be used mixed together with waterfall: evolutionary approaches for uncertainties in specification (e.g. user interface) and waterfall for parts well understood.

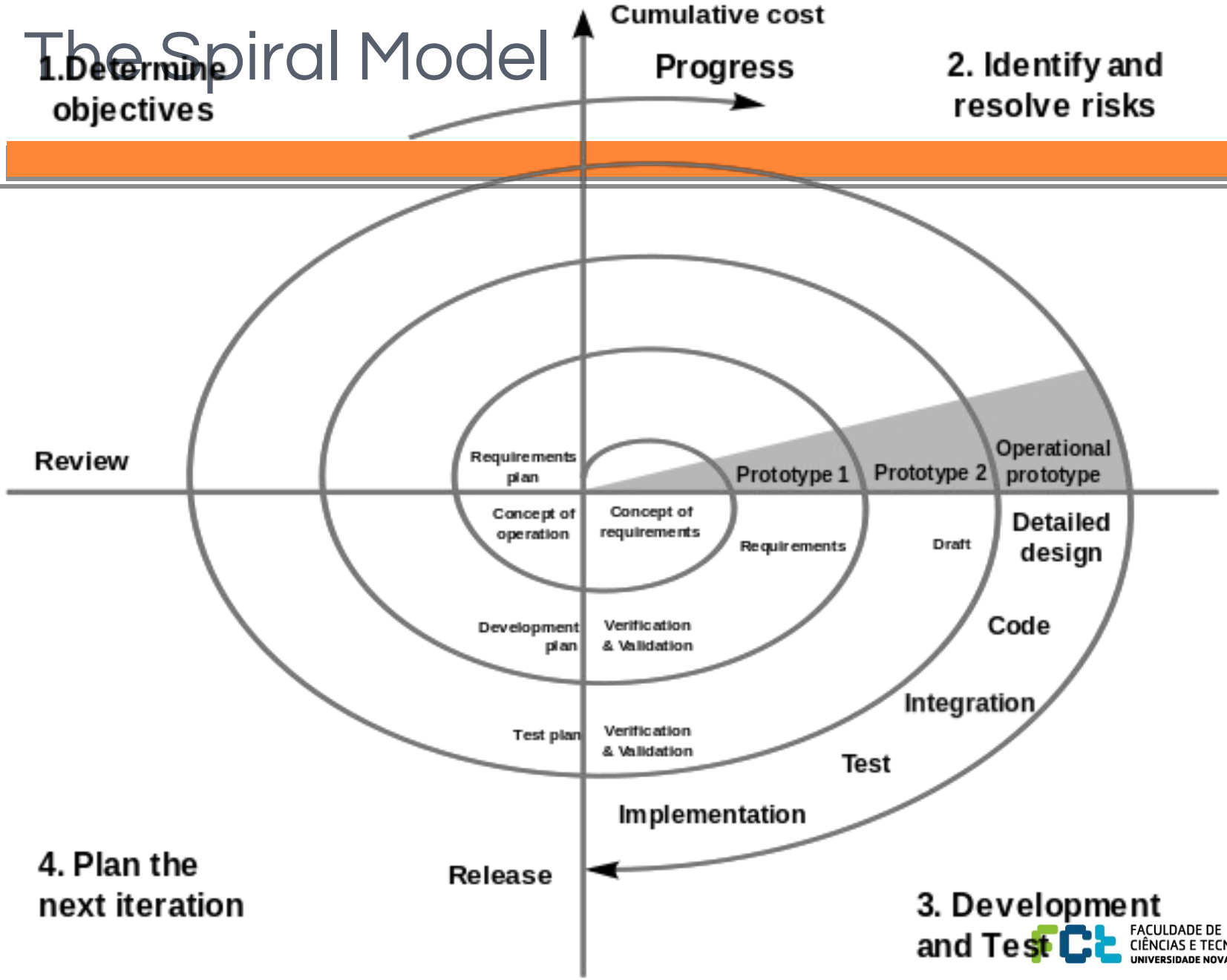
Process Iteration

- The specification is developed in conjunction with the software
- There is no complete system specification until the final increment is specified.
- Requires new form of contract that large costumers and Government agencies may find difficult to accommodate

The Spiral Model

- Development cycles through multiple (3-6) task regions (6 stage version)
  - customer communication
  - planning
  - risk analysis
  - engineering
  - construction and release
  - customer evaluation
- Incremental releases
  - early releases may be paper or prototypes
  - later releases become more complicated
- Models software until it is no longer used

The Spiral Model



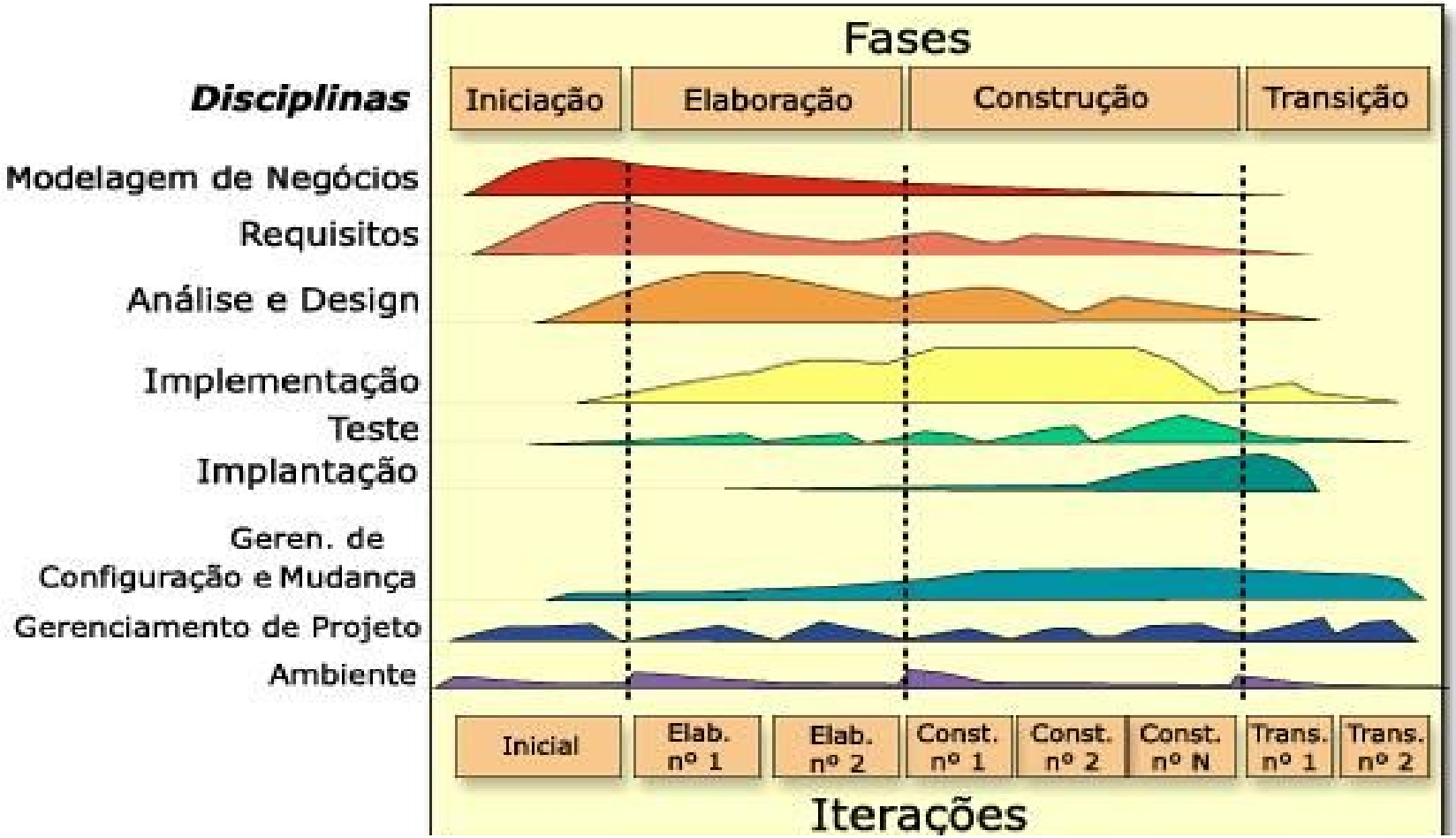
The Spiral Model

- Not a silver bullet, but considered to be one of the best approaches
- Is a realistic approach to the problems of large scale software development
- Can use prototyping during any phase in the evolution of product
- Requires excellent management and risk assessment skills

RUP – Rational Unified Process

- A framework for object-oriented Software Engineering using UML
- Use-case driven, architecture-centric, iterative and incremental software process

RUP - Phases


Fases
Disciplinas
Iniciação
Elaboração
Construção
Transição
Modelagem de Negócios
Requisitos
Análise e Design
Implementação
Teste
Implantação
Geren. de Configuração e Mudança
Gerenciamento de Projeto
Ambiente
Inicial
Elab. nº 1
Elab. nº 2
Const. nº 1
Const. nº 2
Const. nº N
Trans. nº 1
Trans. nº 2
Iterações

Best Practices

- Develop Software iteratively
- Manage Requirements
- Use Component Base Architectures
- Verify Software Quality
- Control changes to Software

To be continued…