Software Life-cycle and Software Development Processes

Andrea Polini

Software Project Management
MSc in Computer Science
University of Camerino

Software Life Cycle

A Life Cycle refers to the states that a given entity can assume during the span of its life, from conceivement to dismissal

A software development process relates to strategies to be adopted to manage the life cycle of a software product. In particular it identifies when and who should do what to reach a given objective in the management of the life-cycle

Process Activities

Software processes can be distinguished on the base of the arrangements of the diffeernt typical development activities (workflow perspective) and the artefacts that these activities produce (data-flow perspective):

- Communication (to define what need to be done)
- Planning (risks and costs)
- Design (models of the systems)
- Construction (implementation of the system and properties verification)
- Deployment (production environment)

Communication

- To understand what the system should do
- Which are the constraints that need to be satisfied
- Generally two different kinds of specifications are derived
    - For the customer
    - For the developer

Communication

...continue

Requirements engineering generally includes 4 main sub-activities:

- Feasibility study
- Requirements elicitation and analysis
- Requirements specifications
- Requirements validation (realistic? consistent? complete?)

Design

To derive a description of:

- the software to be developed and its architecture
- the data that need to be exchanged
- componets composing the system
- interfaces among components
- algorithms

Design

Different approaches to design

- Agile methodologies reduce at minimum the quantity of time spent in modeling
- Structured models generally derive many models using some graphical notation (e.g. UML)

Modeling
...continue

Typical models developed using a structuraed approch:

- Object Model
- Sequence Model
- State Transition System
- Structural Models
- Data-flow models

Construction

In this case the objective is to derive the code for the system under development, and according to defined models

OO Programming
Design patterns

Construction

Verification and Validation

Objective is to check that the system satisfy its requirements and the needs of the user.
Tipically used techniques:

- Static (i.e. code inspection)
- Dynamic (i.e. Testing)

Typically testing is organized over more phases:

- Unit testing
- Integration testing
- System testing
- Alpha testing
- Beta testing

Maintenance

This refers to the organization of the activities concerning software already realised to customers

Evolution kinds:

- Corrective
- Adaptive
- Perfective

Software Process Models

A software development process is constituted by the activites carried on to build a software product

- No general solution! In general all the organizations develop their own software development process.
- Stong interrelation with people and internal organization

- Structured methods: sets of codified steps and rules which when applied, generate systems products (heavyweight methods)
- Agile methods: getting working applications delivered quickly and at less cost

Software Process Models

A software development process is constituted by the activites carried on to build a software product

- No general solution! In general all the organizations develop their own software development process.
- Stong interrelation with people and internal organization

- Structured methods: sets of codified steps and rules which when applied, generate systems products (heavyweight methods)
- Agile methods: getting working applications delivered quickly and at less cost

Software Process Models

Main models to be presented (Warning they are just models):

- Waterfall model
- Evolutionary
- Iterative
- Formal methods (cleanroom development, B method, Model Driven Development)

In reality adopted process are derived as a mix of the listed models

Waterfall model

Historically is considered the first process model (Royce 1970).
Development phases are structured according to the main activities:

- Requirements elicitation and analysis
- Software project modeling
- Implementation and unit tests
- Integration and system test
- Deployment and manteinance

Waterfall model

- to be more realistic the various activities are repeated till at a certain point when they are freezed - each activity provides artefacts to the next phases deriving needed documents, models, codebase, . . .
- once the activity is ended it is not anymore considered in the future
- in general it is highly desirable that phases overlap a bit to guarantee the flow of information

Main consegueces:

- requirements after a while are freezed, and not anymore modified
- The process becomes unresponsive to request of requirements modification
- Strongly documented software and document driven.

When can it be a useful option?

Prototyping

The product is developed at first trying to derive a first prototype.

- Successively through successive increments the system is derived till the objectives are met (Evolutionary prototyping)
- the prototype is trown-away and the development is restarted using any process model among the others. (Throw-away prototyping - do it twice)

Rationale:

- The prototype intends to support communication activities
- each increment includes phases in which the user is involved

Main consequences:

- Is difficult to throw-away a prototype
- system organization is generally not so much clear and clean as consequence of the continuous changes
- mainteinance could be harder

Prototyping - consequences

Plus:

- + Improved user involvement
- + Clarification of partially known requirements
- + Demonstration of the consistency and completness of a specification
- + Reduced need for documentation
- + Reduced maintenance costs
- + Feature constraint
- + Production of expected results

Drawbacks:

- - Users can misunderstand the role of the prototype
- - Lack of project standards
- - Lack of control
- - Additional expense
- - Machine efficiency
- - Close proximity of developers

Prototype characteristics

Prototype Extent

- Mock-ups
- Simulated interactions
- Partial working models: vertical/horizontal

What

- HCI
- Functionality

Iterative process models

Approach that tries to borrow the positive characteristics both from evolutionary approaches, and from the waterfall models.

Based on the concept of iteration (the specification evolves together with the system development):

- Incremental releases
- Spiral model

Incremental releases

- Development proceeds according to many small “waterfall” in a row
- For each iteration a working system is released. The system will be used by the customer
- Successive iterations intend to introduce novel functionality
- Once started the iteration cannot be stopped
- Each iteration should target an increment corresponding to something “manageable in isolation” (20.000 LOC?)

Incremental release

Pros

Main advantages (somehow we have a prototype):

- Particularly effective when a small team is constituted
- Customers do not have to wait for the final release to use the system
- Rick of failure is reduced
- The most important funtionality are tested more

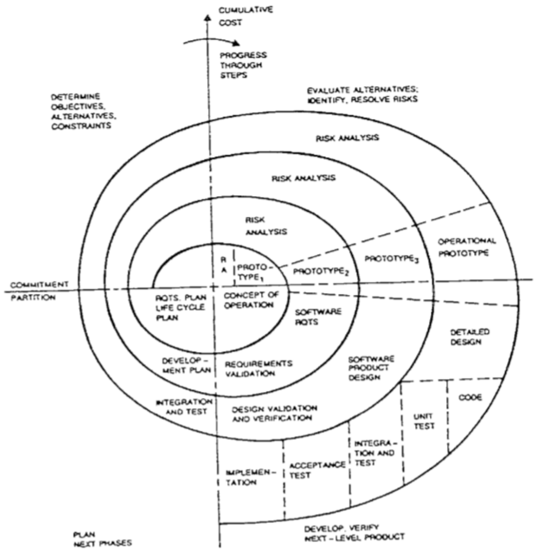
Spiral model

Proposed by Boehm in 1988 it introduced a risk evaluation activity in each iteration

It is generally represented as a plan for which each quarter includes activities related to:

- the specification of the objectives and identifications of risks for the next iteration
- Manage risks
- Development and validation
- Planning for a new iteration

Spiral Model



Model Driven Development

Development of a software system evolve as result of continuous refinement of software models. For each refinement step additional information are provided till the level of detail if that of a selected programming language.

- CIM (Requirements model)
- PIM (High level design)
- PSM (System design that includes information about the platform and the deploying environment)

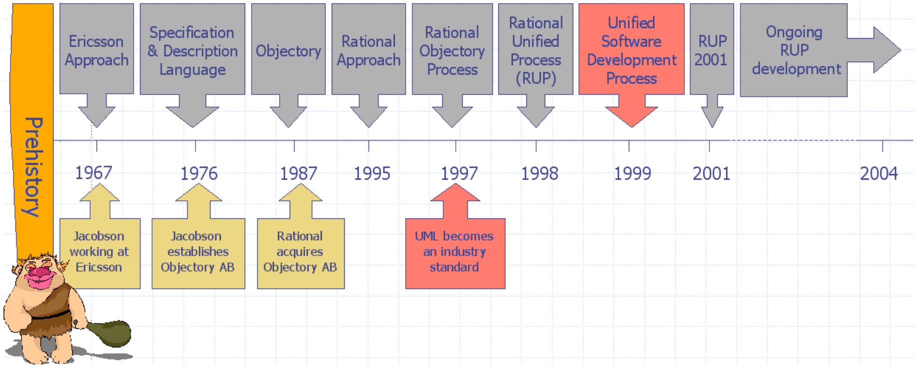
The Unified Process

The Unified Process

The Unified Software Development Process (Unified Process - UP) is more than just a process and includes most of the PM activities

- UP it is a “standardized” industrial process to develop software systems:
    - Reference process in conjunction with modeling activities using UML
    - Introduced in the book “The Unified Software Development Process”
- Main characteristics:
    - Risk driven
    - Driven by the Use cases
    - Architecturally grounded
    - Incremental and iterative process
- the UP is a generic software development process that has to be adapted to each single project:
    - Internal standards, Document formats, tools, databases . . .
- Rational Unified Process (RUP) it is a specific instance of the UP:
    - It is a product of the Rational Corporation (IBM)

Historical perspective

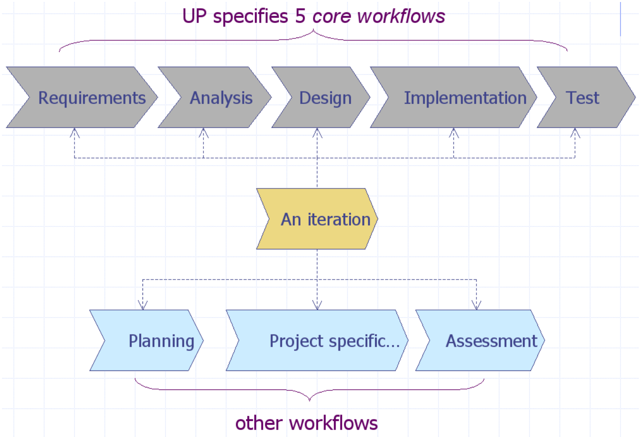

Prehistory
Ericsson Approach
Specification & Description Language
Objectory
Rational Approach
Rational Objectory Process
Rational Unified Process (RUP)
Unified Software Development Process
RUP 2001
Ongoing RUP development
1967
1976
1987
1995
1997
1998
1999
2001
2004
Jacobson working at Ericsson
Jacobson establishes Objectory AB
Rational acquires Objectory AB
UML becomes an industry standard

Iterations

- Each iteration can be considered as a small development project for which the following activities are considered:
    - Planning
    - Requirements
    - Analysis
    - Design
    - Development
    - Test and Integration
    - Validation and Internal or External release
- The final product is derived as a sequence of iterations
- iterations can superimpose each others
- iterations are organised in 4 different phases
- iterations are "timeboxed" and they should last for a short period ridotta (2-3 weeks)

Workflows for UP iterations

Each iteration can include all the activities foreseen by the workflow. Depending from the phase in which the iteration is located the different activities will have a different weight



Analysis and Design

- Analysis emphasizes the investigation of the problem and corresponding requirements. It does not target the definition of a solution
- Design intends to define a conceptual solution that satisfies the requirements.
- Implementation describes the complete project and it is codified in the selected programming language and technologies

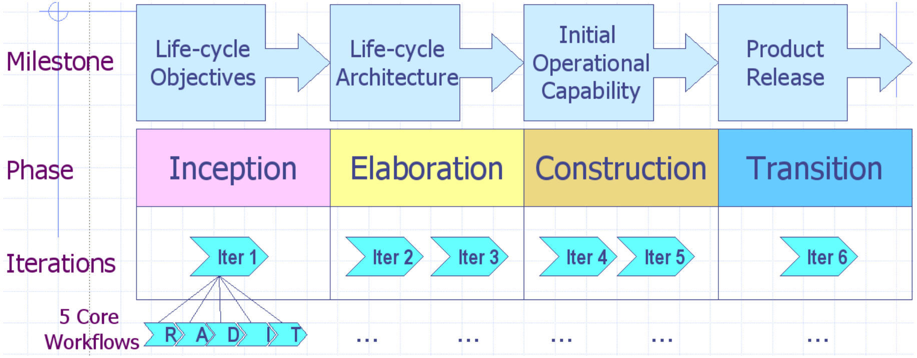
Example: consider a system for the management of a restaurant and identify relevant and general concepts

$$\text{Reality} \xrightarrow{OOA} \text{analysis classes} \xrightarrow{OOD} \text{design classes} \xrightarrow{OOP} \text{implementation classes}$$

Baselines

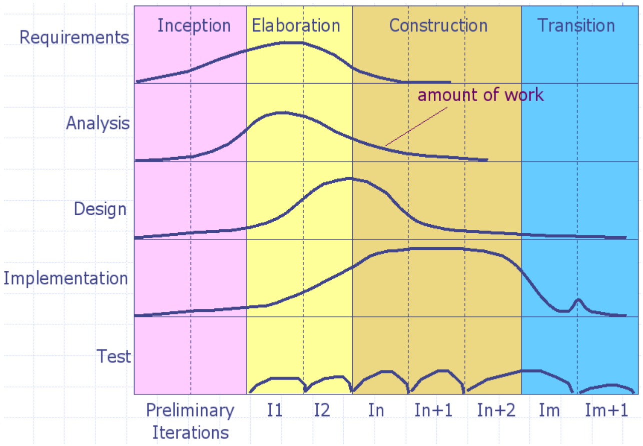
- Each iteration generate a “baseline”
- A baseline can be characterized by the set of artefacts revised and approved that:
  - Provide a common baseline for further development
  - Can be modified adopting a formal procedure
- An increment is given by the difference between the baseline resulting from the iteration and the one resulting from the previous iteration

UP structure

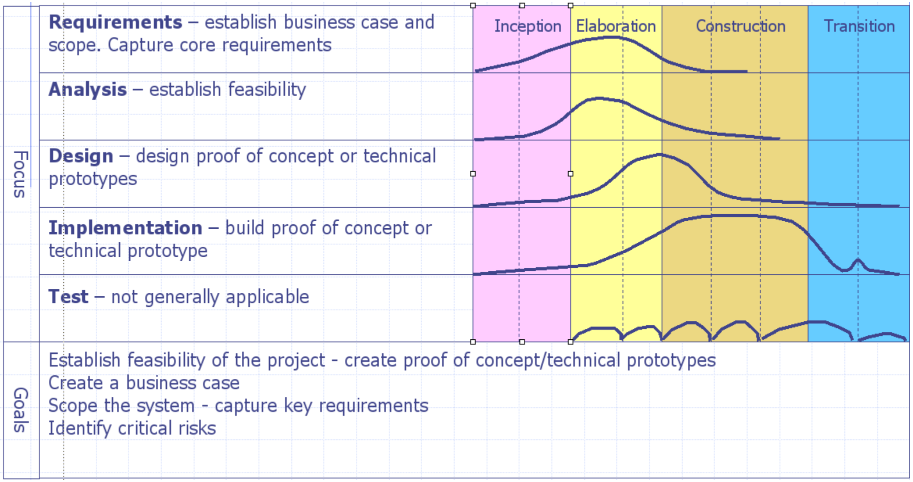


- Depending from the dimension of the project each phase can include more iteration for each phase of the project
- A phase ends with a milestone
- Each pahse is characterized by: objectives, focus of the workflows, milestones

Phases and workflows in UP



Inception



Requirements – establish business case and scope. Capture core requirements

Analysis – establish feasibility

Design – design proof of concept or technical prototypes

Implementation – build proof of concept or technical prototype

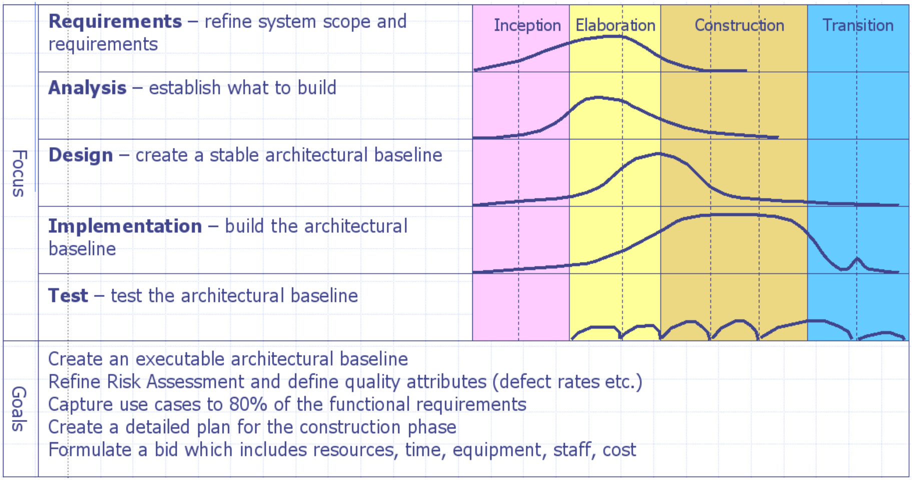
Test – not generally applicable

Establish feasibility of the project - create proof of concept/technical prototypes
Create a business case
Scope the system - capture key requirements
Identify critical risks

Inception

- Conditions to be satisfied and artefacts:
  - The scope of the system has been defined (document reporting the main requirements)
  - The context of the project has been clarified (fundamental requirements have been explored and main stakeholders have been contacted – Defined Use cases 10-20 %)
  - An architectural “vision” has been defined
  - A first attempt to identify and assess risks
  - First cost assessment and time schedule has been defined (Project plan)
  - A business case has been identified and advantages have been highlighted
  - Project feasibility has been confirmed (possible usage of prototypes)

Elaboration

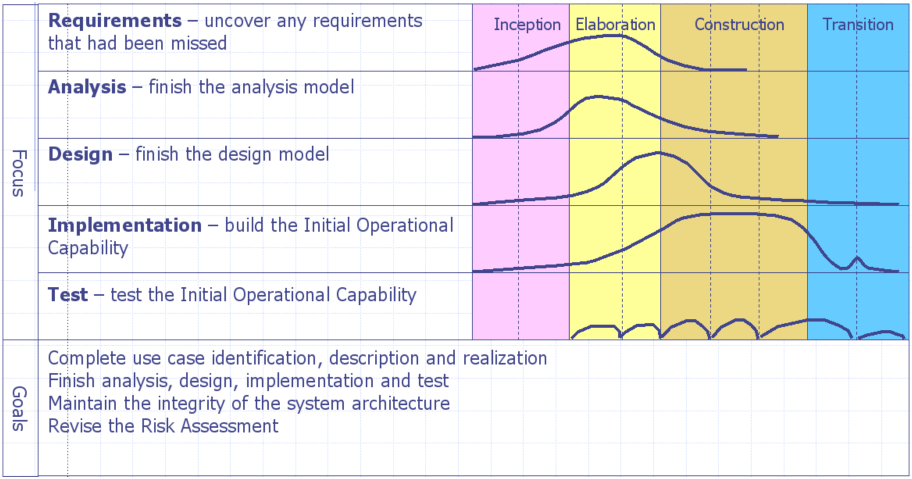


Focus	**Requirements** – refine system scope and requirements
	Analysis – establish what to build
	Design – create a stable architectural baseline
	Implementation – build the architectural baseline
	Test – test the architectural baseline
Goals	Create an executable architectural baseline Refine Risk Assessment and define quality attributes (defect rates etc.) Capture use cases to 80% of the functional requirements Create a detailed plan for the construction phase Formulate a bid which includes resources, time, equipment, staff, cost

Elaboration

- Conditions to meet and artefacts:
  - The baseline can be run
  - Executable baseline demonstrates that risks have been identified and partially solved (UML structural models, UML behavioural models, Use cases)
  - Risks revisions
  - Overall vision have been stabilized (80 % of the use cases have been defined)
  - Parties have approved the project planning
  - Detailed planning to derive realistic estimates for costs, time and required
  - Signed contract

Construction



Requirements – uncover any requirements that had been missed

Analysis – finish the analysis model

Design – finish the design model

Implementation – build the Initial Operational Capability

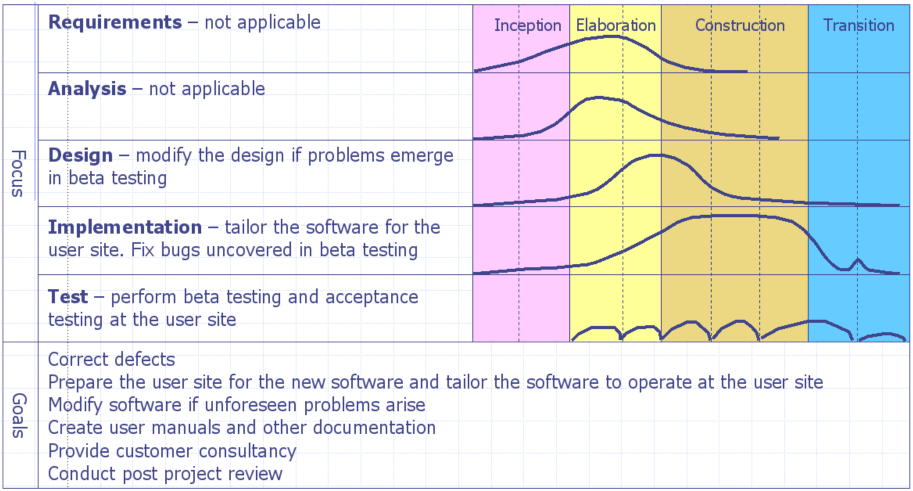
Test – test the Initial Operational Capability

Complete use case identification, description and realization
Finish analysis, design, implementation and test
Maintain the integrity of the system architecture
Revise the Risk Assessment

Construction

- Conditions to meed and artefacts:
    - Software of high quality ready to be released (software product, UML models, test suites)
    - Checking of estimates and motivations for divergences
    - Deployment manual and release description

Transition



Focus	**Requirements** – not applicable
	Analysis – not applicable
	Design – modify the design if problems emerge in beta testing
	Implementation – tailor the software for the user site. Fix bugs uncovered in beta testing
	Test – perform beta testing and acceptance testing at the user site
Goals	Correct defects Prepare the user site for the new software and tailor the software to operate at the user site Modify software if unforeseen problems arise Create user manuals and other documentation Provide customer consultancy Conduct post project review

Transition

- Conditions to meet and artefacts:
  - Beta testing has been performed, needed modifications have been addressed, final users are happy with the release of the software
  - The systems is actually working
  - Supporting and evolution strategies have been defined and agreed among the parties