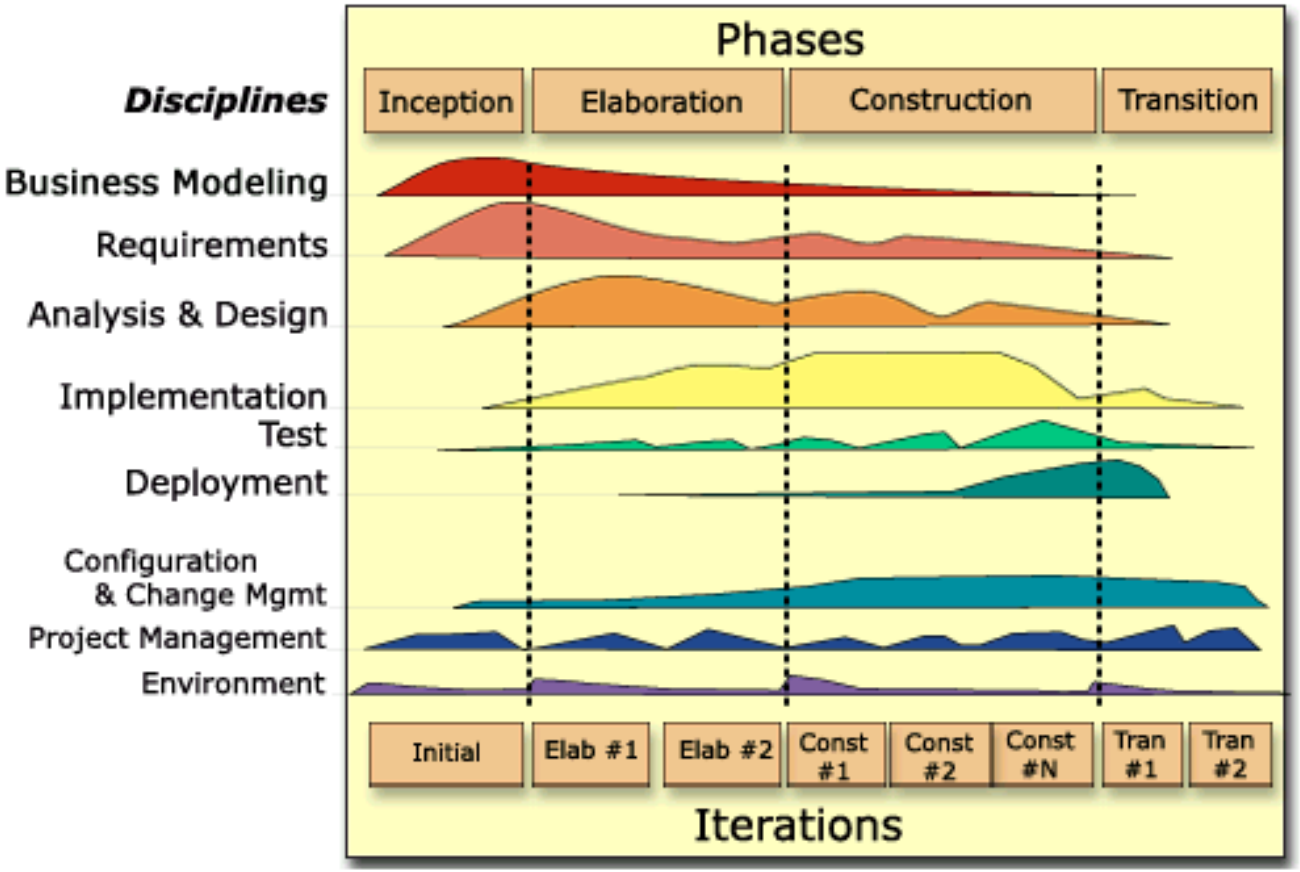
Génie logiciel

Philippe Dugerdil

10.10.2019

RUP

RUP

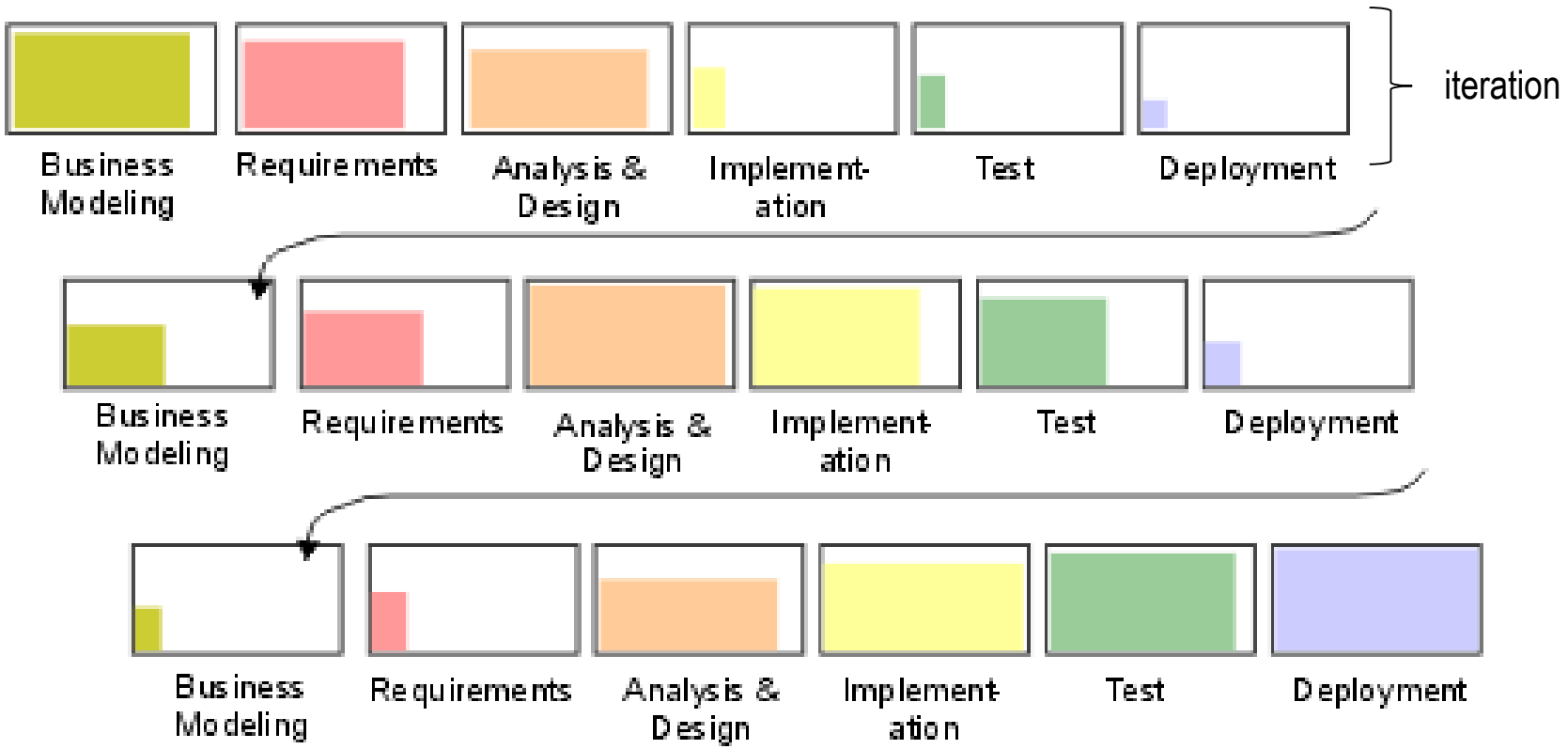


Source: RUP2000, Rational Software Corp, 2000 © Rational-IBM

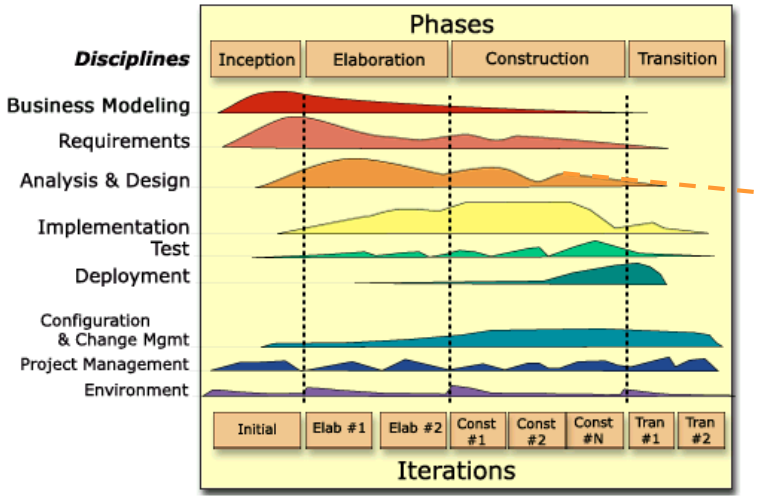
Origin

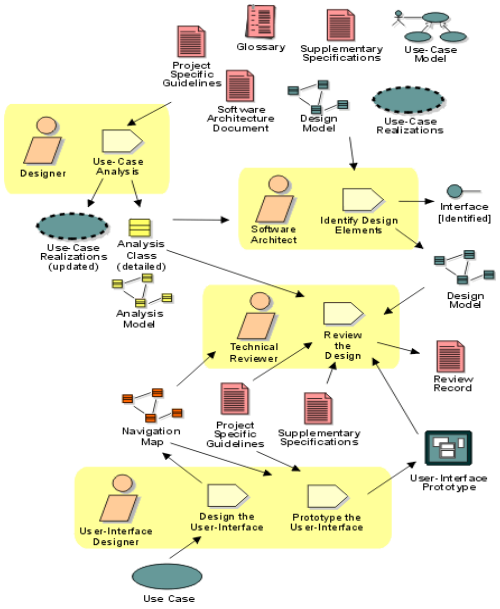
- Early 1990's - Objectory (Jacobson)
- 1999 - UP definition. Industrialization by Rational
- 2003 - Acquisition of Rational by IBM
- 2006 - OpenUP: Agile open source version
  - Ivar Jacobson, Grady Booch, James Rumbaugh - The Unified Software Development Process. Addison-Wesley 1999
  - Philippe Kruchten - The Rational Unified Process: An Introduction. Addison-Wesley 2003

Workload in each discipline : depends on the progress through the project

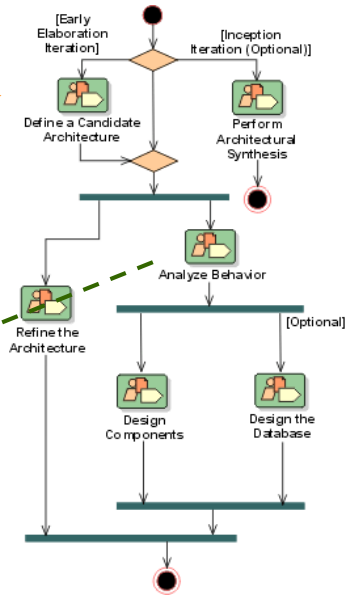


RUP is heavily documented





Workflow

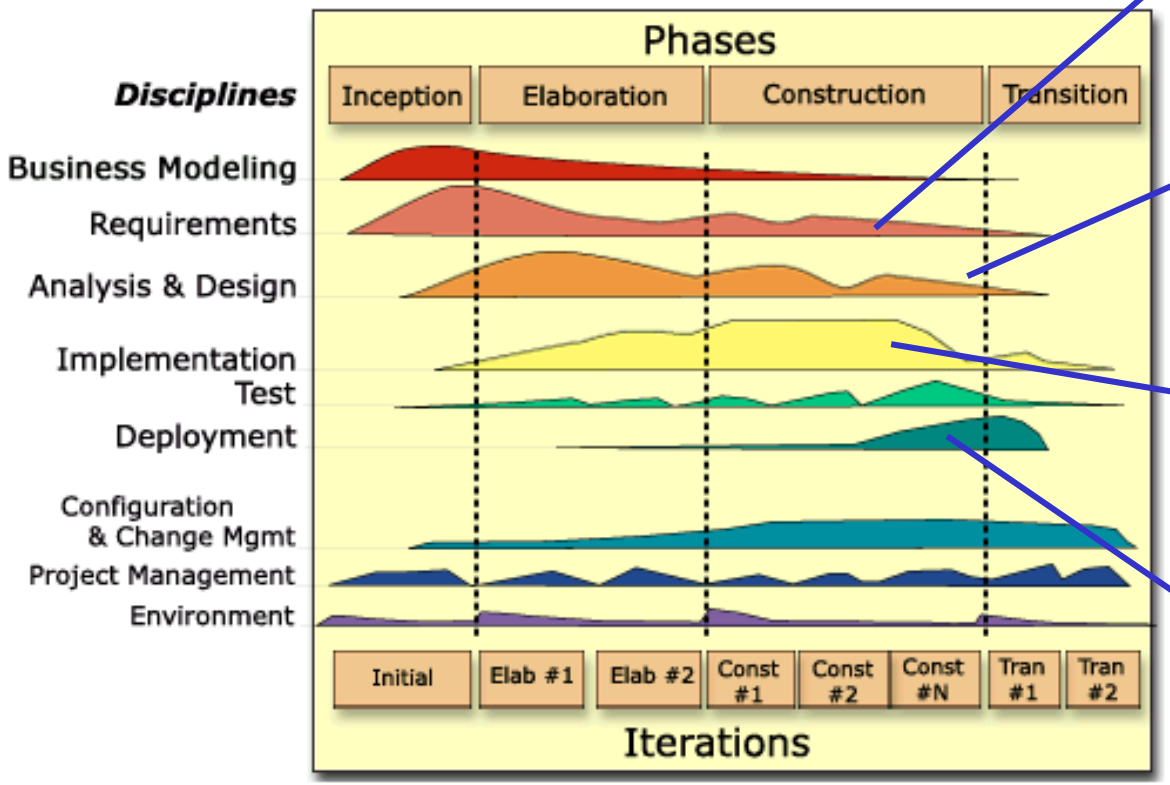


Features of RUP

6 best practices :

1. Develop software iteratively
2. Manage the requirements
3. Use component-based architectures
4. Visually model software
5. Control software quality constantly
6. Manage the software changes (version control)

Outcome of disciplines: models (UML)

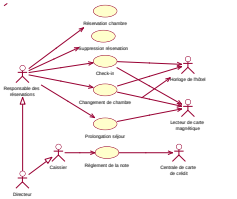


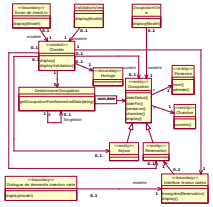
What to do?

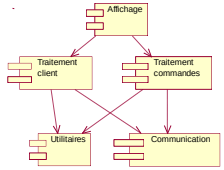
How to do it ?

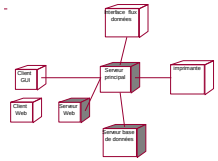
Do it

Deliver



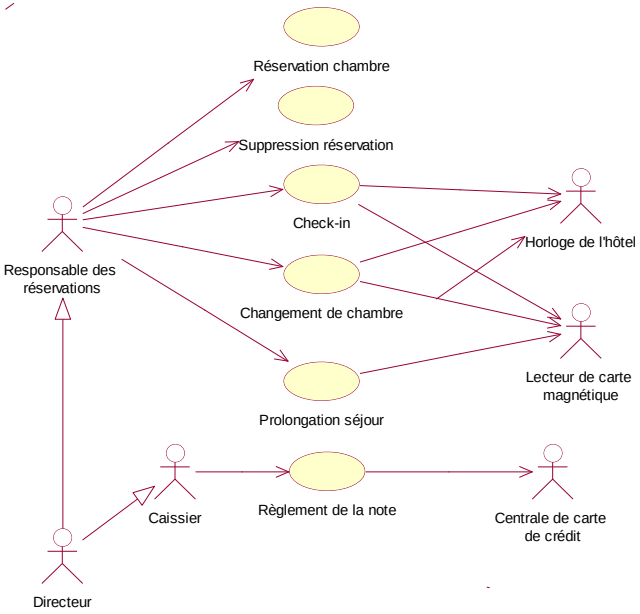


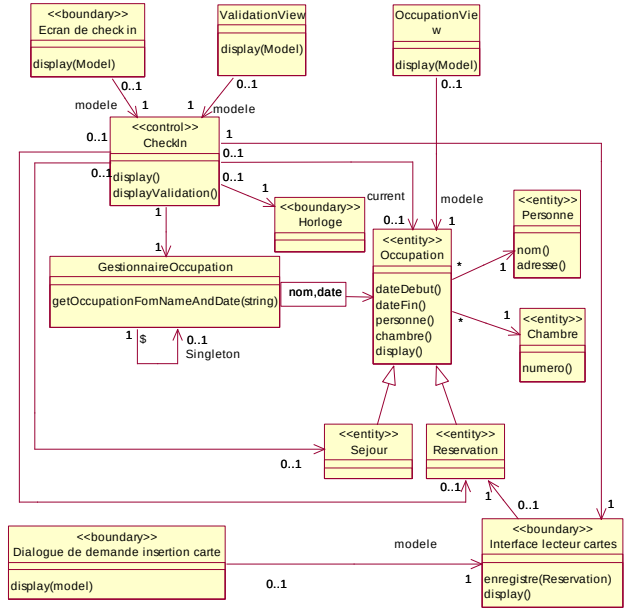


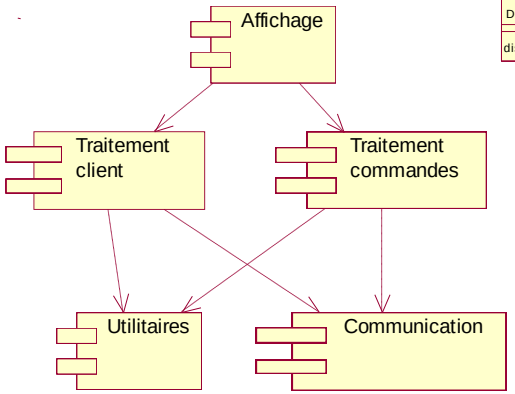


D'après: RUP2000, Rational Software Corp, 2000

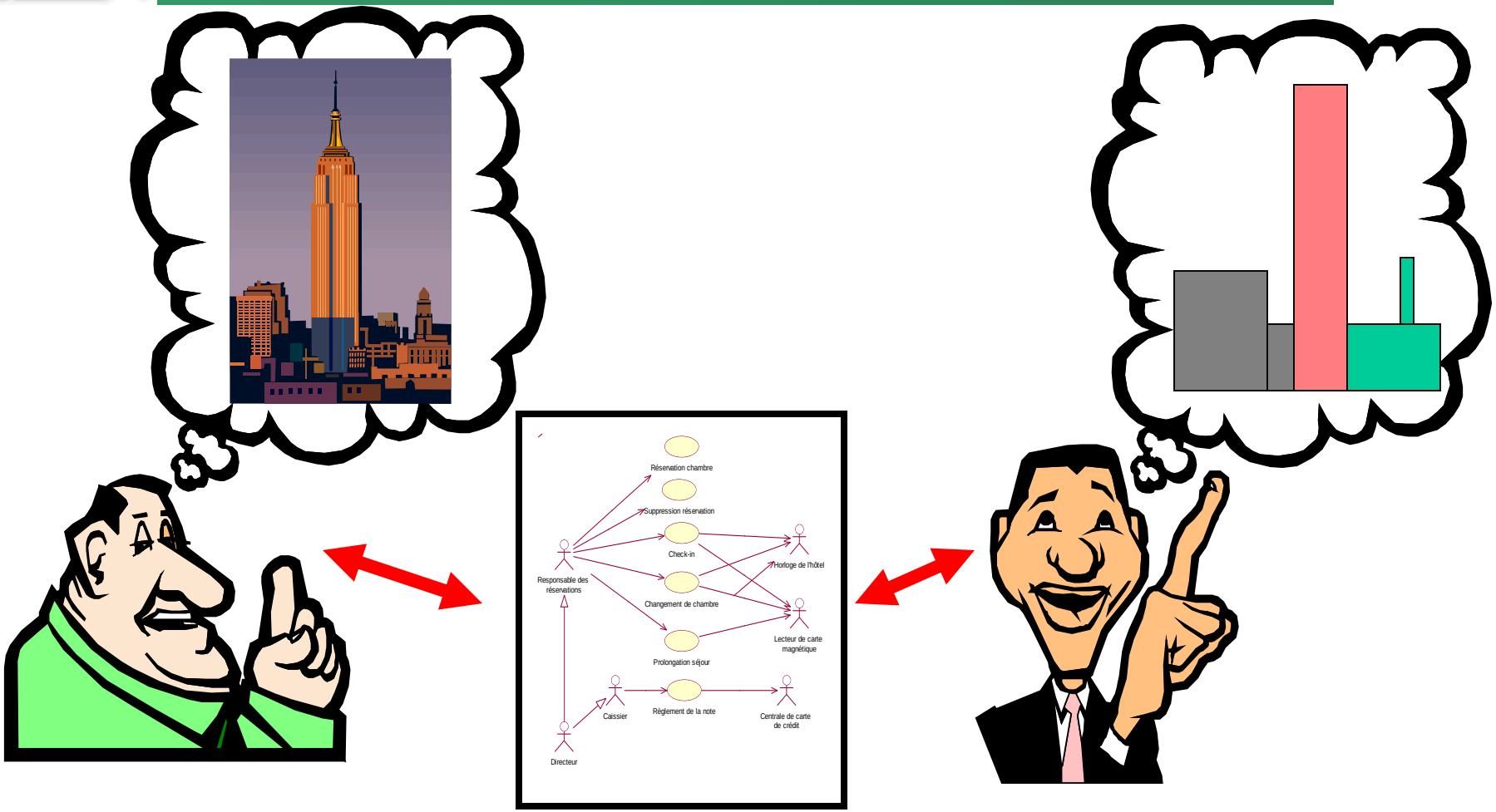
Role of UML models : communication



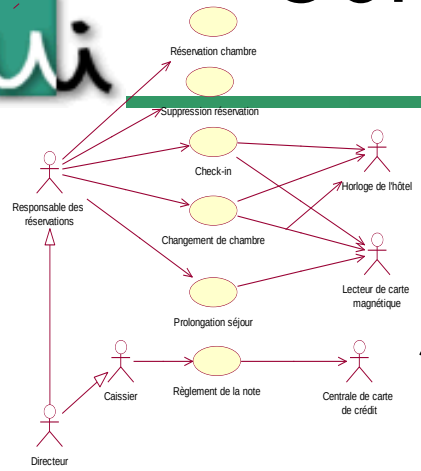


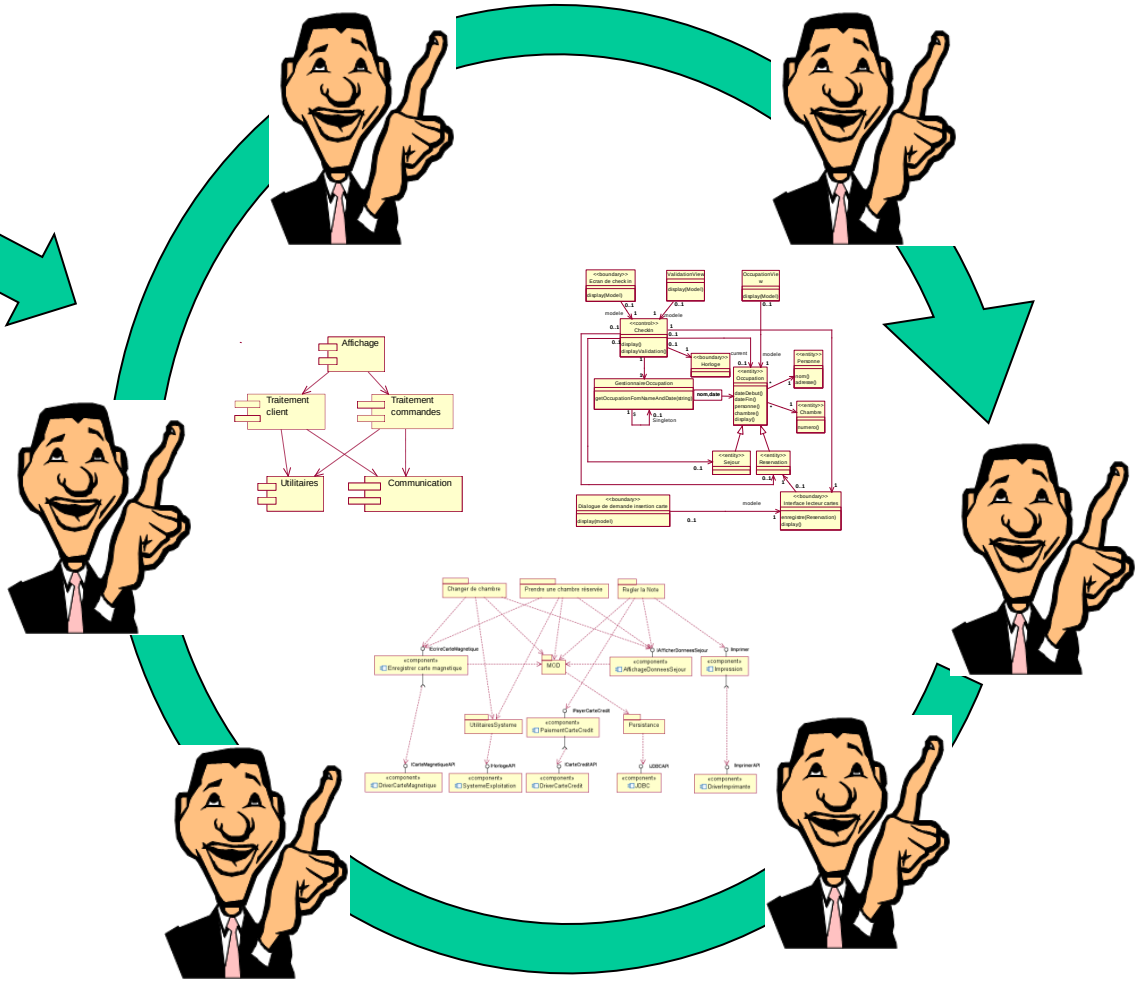


Communication with the customer

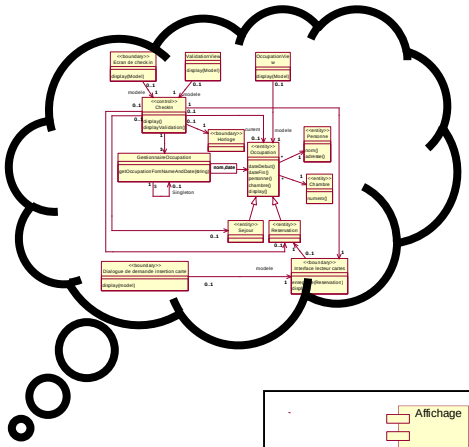


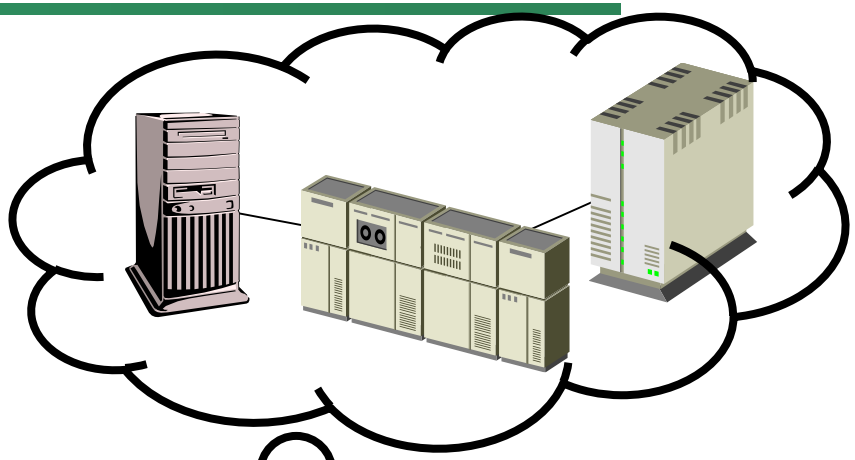
Communication within the team

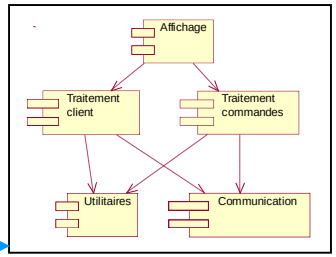


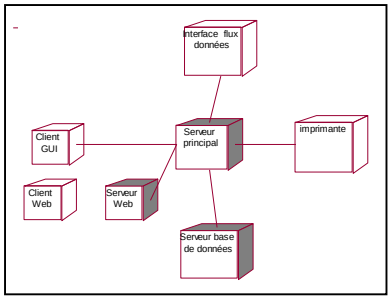


Communication with operations

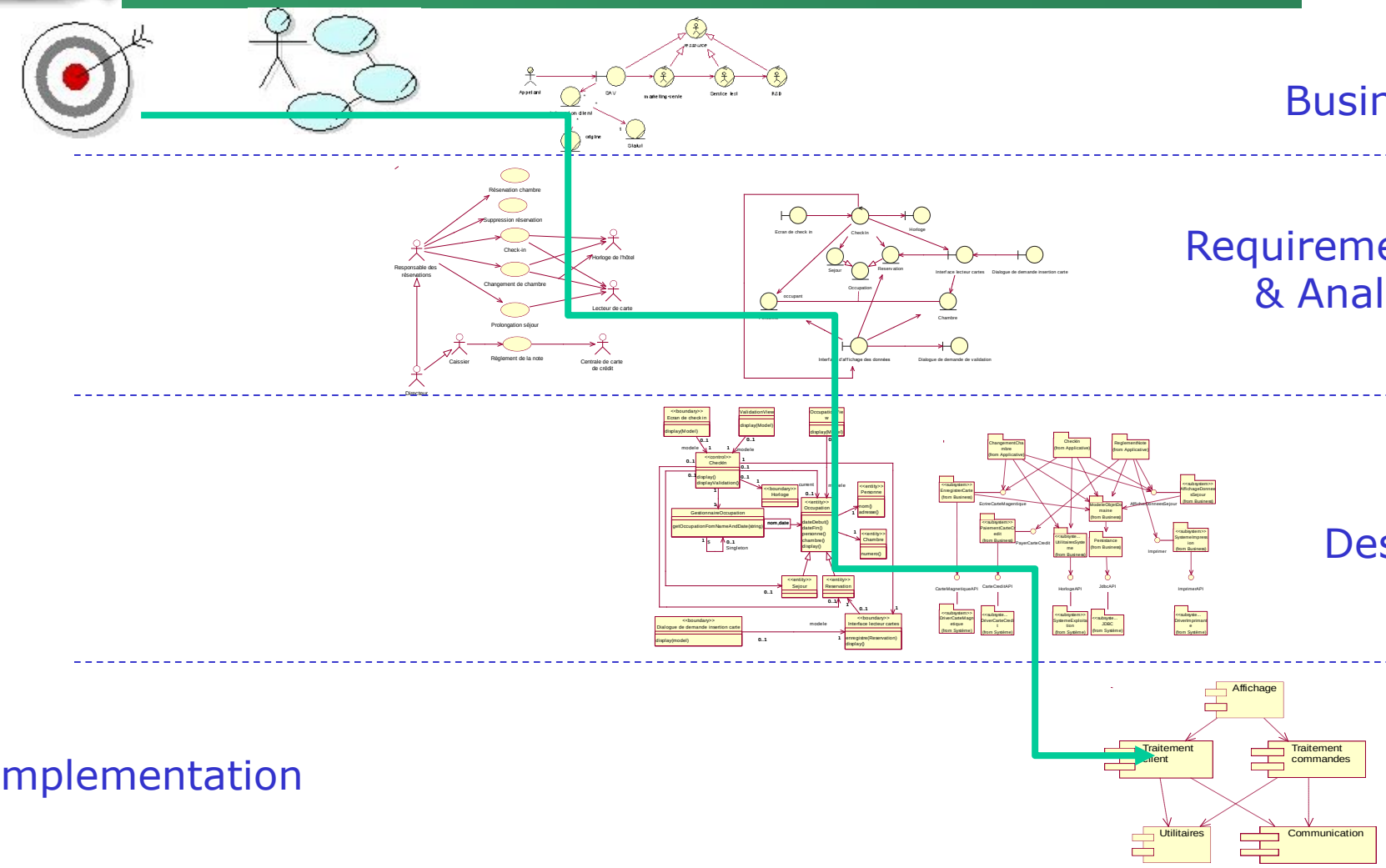








The models are traceable



RUP is risk centered

Risks : “ The events which might occur which would decrease the likelihood that we will be able to deliver the project with the right features, the requisite level of quality, on time and within budget.” [IBM]

In short: **any situation that would lead to project failure**.

- But…what is a project failure ?
- Who defines (evaluates) it ?

Risk list: features of a risk

For all risks:

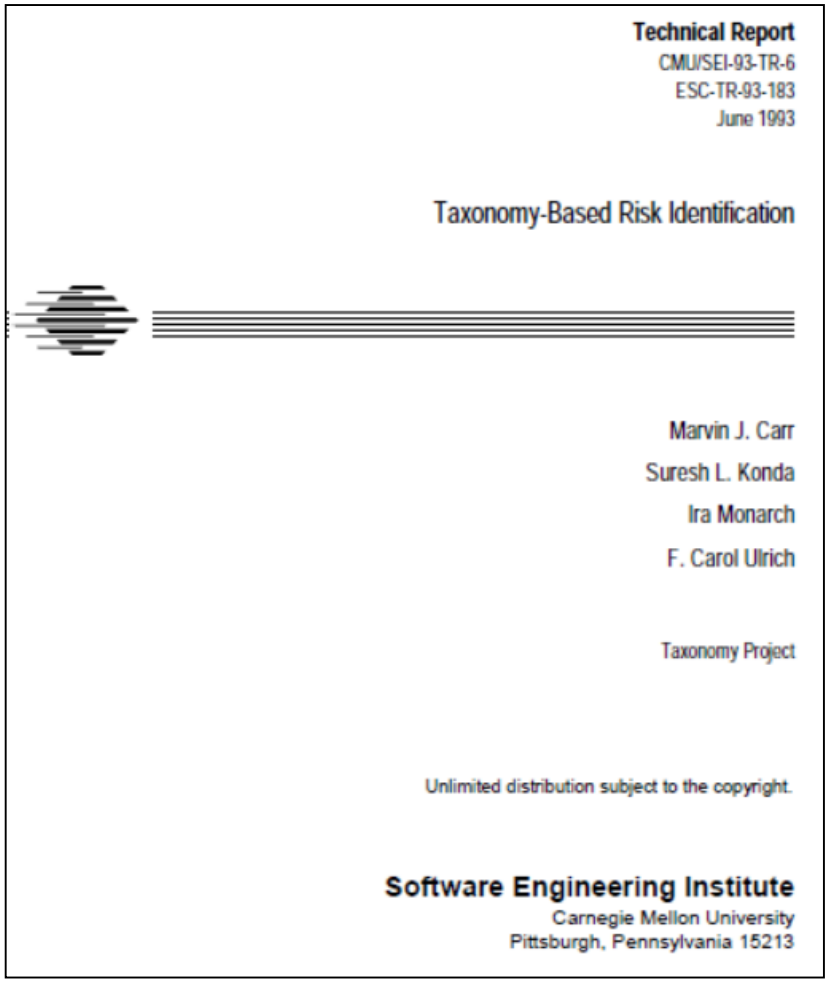
1. Risk priority indicator
2. Risk description
3. Impact on the project
4. Risk occurrence indicator
5. Risk mitigation strategy
6. Alternative plan in case of trouble

Main software risk factors

Source of Risk	Risk Management Techniques
1. Personnel shortfalls	• Staffing with top talent; key personnel agreements; team-building; training ; tailoring process to skill mix; walkthroughs.
2. Schedules, budgets, process	• Detailed, multi-source cost and schedule estimation; design to cost; incremental development; software reuse; requirements descoping; adding more budget and schedule; outside reviews.
3. COTS, external components	• Benchmarking; inspections; reference checking; compatibility prototyping and analysis
4. Requirements mismatch	• Requirements scrubbing; prototyping; cost-benefit analysis; design to cost; user surveys
5. User interface mismatch	• Prototyping; scenarios; user characterization (functionality; style, workload); identifying the real users
6. Architecture, performance, quality	• Simulation; benchmarking; modeling; prototyping; instrumentation; tuning
7. Requirements changes	• High change threshold: information hiding; incremental development (defer changes to later increments)
8. Legacy software	• Reengineering; code analysis; interviewing; wrappers; incremental deconstruction
9. Externally-performed tasks	• Pre-award audits, award-fee contracts, competitive design or prototyping
10. Straining computer science	• Technical analysis; cost-benefit analysis; prototyping; reference checking

Source: Center for Software Engineering, University of Southern California.

Source for risk identification : SEI



Technical Report
CMU/SEI-93-TR-6
ESC-TR-93-183
June 1993

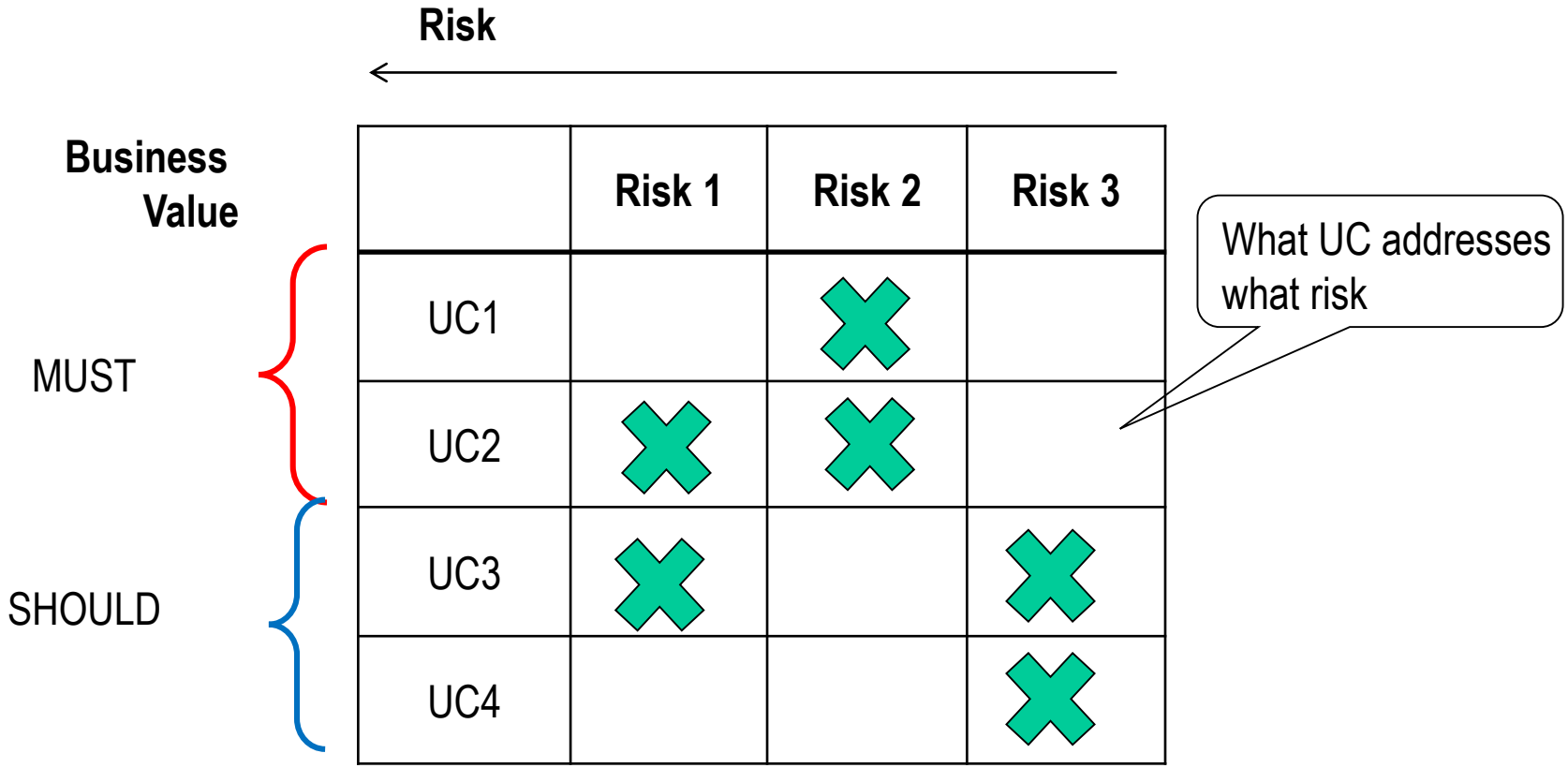
Taxonomy-Based Risk Identification

Marvin J. Carr
Suresh L. Konda
Ira Monarch
F. Carol Ulrich

Taxonomy Project

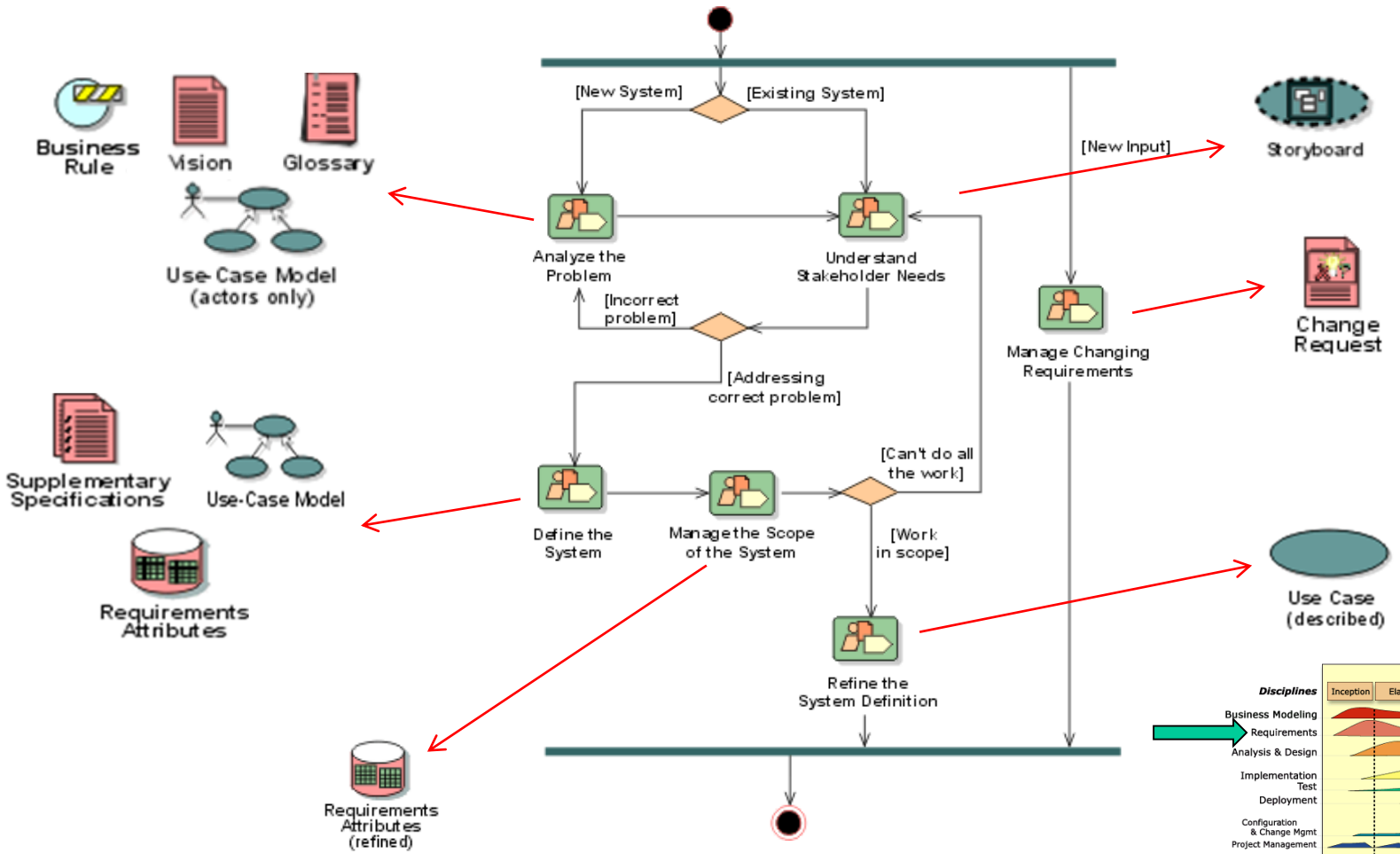
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

Scheduling the use-cases: business value and risks



The first UC to develop would be UC2, then UC1, then UC3 and finally UC4

Requirements tasks and key documents



UP phases and use-cases

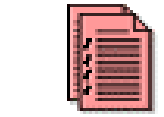
- Select the most critical use-cases
- Design the flows that expose the risks
- Analyze and prototype to eliminate risks

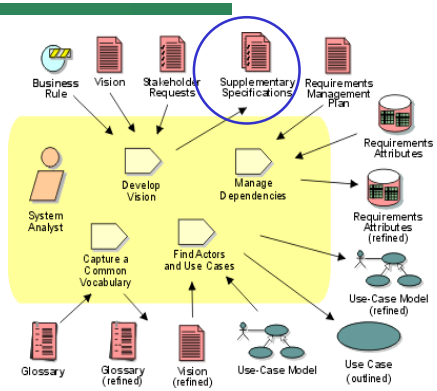
Inception

- Choose the use-case in descending order of criticality =f(risk & business value)
- Complete the flow of the chosen UC
- Analyze and develop the corresponding increment

Elaboration

Supplementary specifications





1. Legal & administrative constraints
2. **Non-functional requirements**
3. General functional specifications
4. Technical constraints (OS, plateform, hardware, COTS, Legacy,…)

« Ultimately, the non functional requirements are every bit as important to the end user community as are the functional requirements. »

[Krutchen Ph. – The Rational Unified Process. An Introduction. Addison-Wesley Inc. 2000.].

Managing the requirements

- List of requirements
- Features of each requirement, examples:
  - Origin
  - Date
  - Status (new, accepted, canceled, in progress, implemented,..)
  - Workload
  - Risk
  - Who submitted it?
  - Priority
  - Person in charge

Simple requirement mgt tool

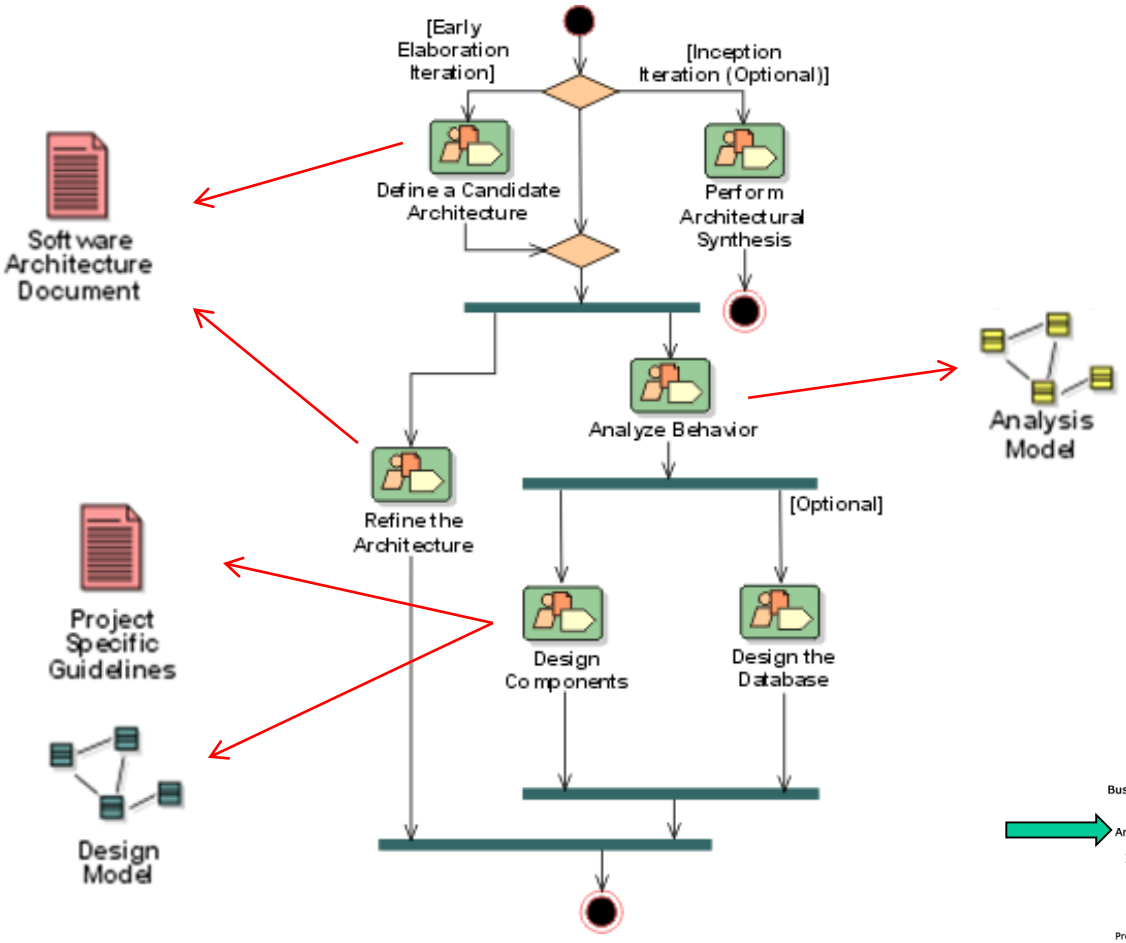
Qui	Quand	Specification	Delai	priorité	Risque	Charge	Responsable	Etat

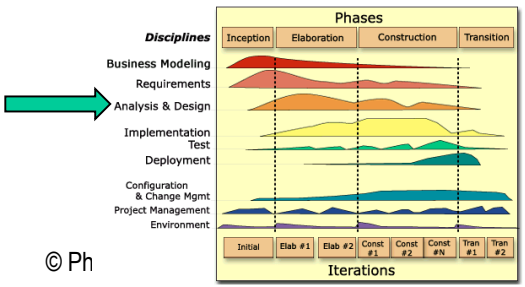
Customer

Project mgr

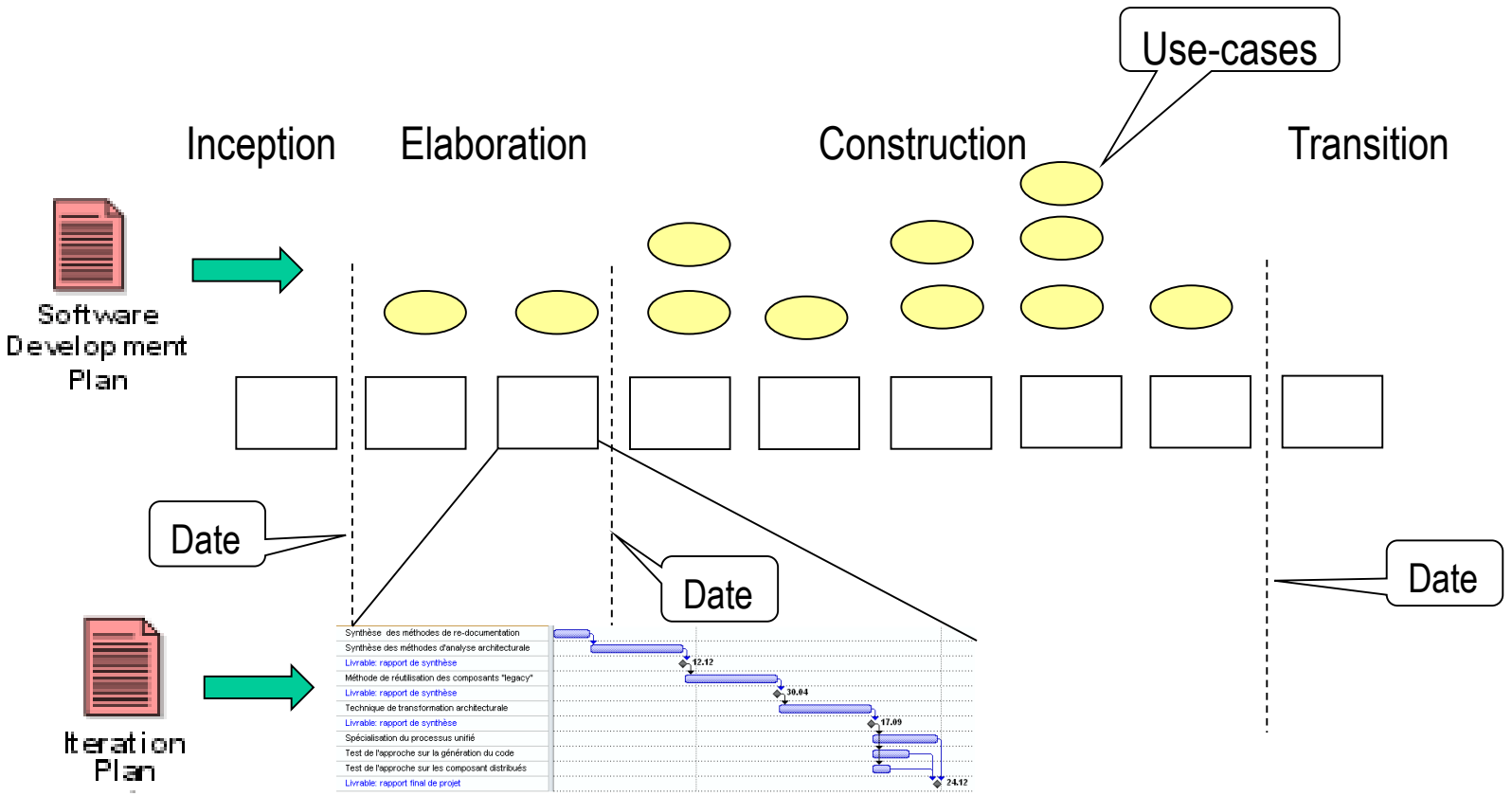
Requirements: UC, BR, NFR, technical constraint, bug description,…

Analysis & design tasks and key documents



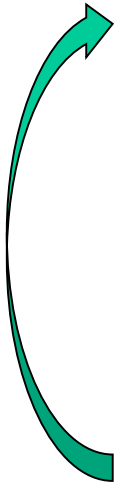


RUP planning



Iteration length = f(requirements)

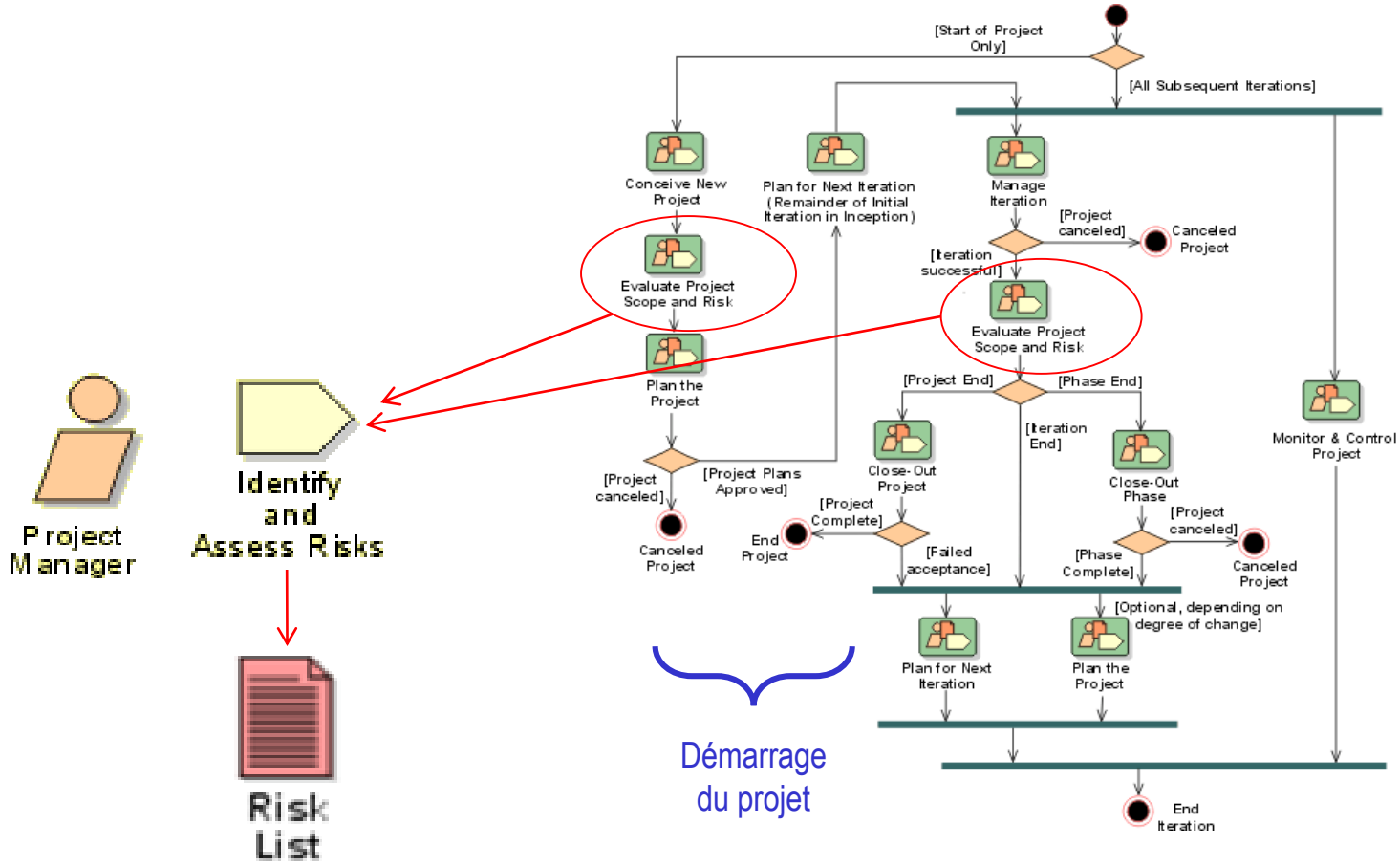
Incremental requirement specification



1. Inception: vision, BR, glossary, identification of the use-cases & supplementary specs
2. Schedule the use-cases in the project plan
3. Select the use-case for the next iteration
4. Write down their flows (or the relevant ones)
5. Estimate the workload and plan the next iteration
6. Assess the iteration and possibly update the use-cases

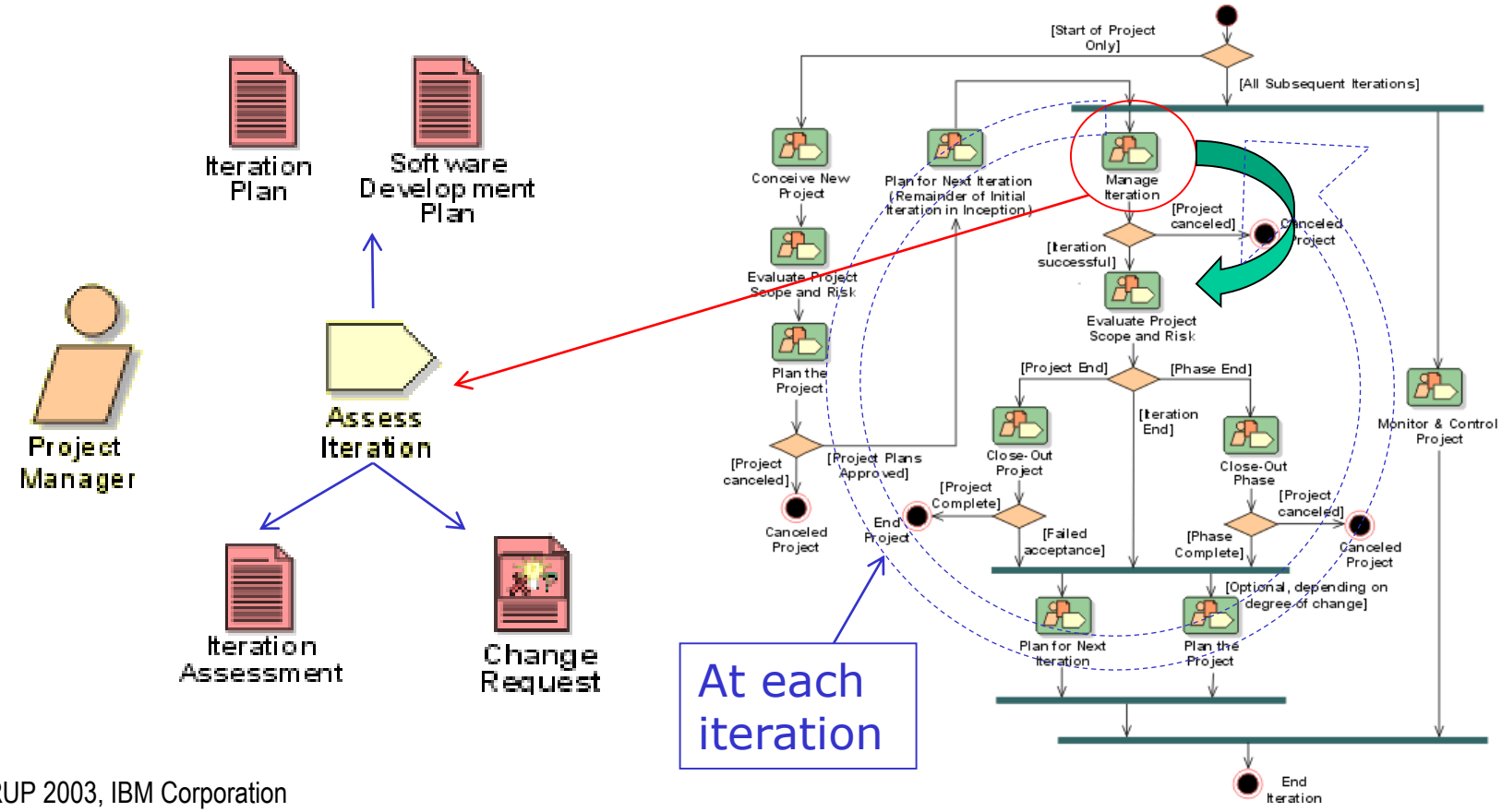
Iteration planning

Eliminate risks at each iteration



Source: RUP 2003, IBM Corporation

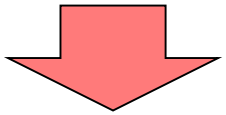
Iteration review : iteration assessment



Source: RUP 2003, IBM Corporation

Iteration assessment (review)

Result of current iteration (deliverables, costs, duration, quality, risks)



- Comparison of deliverables, cost & duration with iteration's plan
- Identify what risks have been lowered, removed, added, increased.
- Decide on the changes in the project if needed
  - Plan the changes in the next iterations
- Update the project plan
- Build the detailed plan of the next iteration

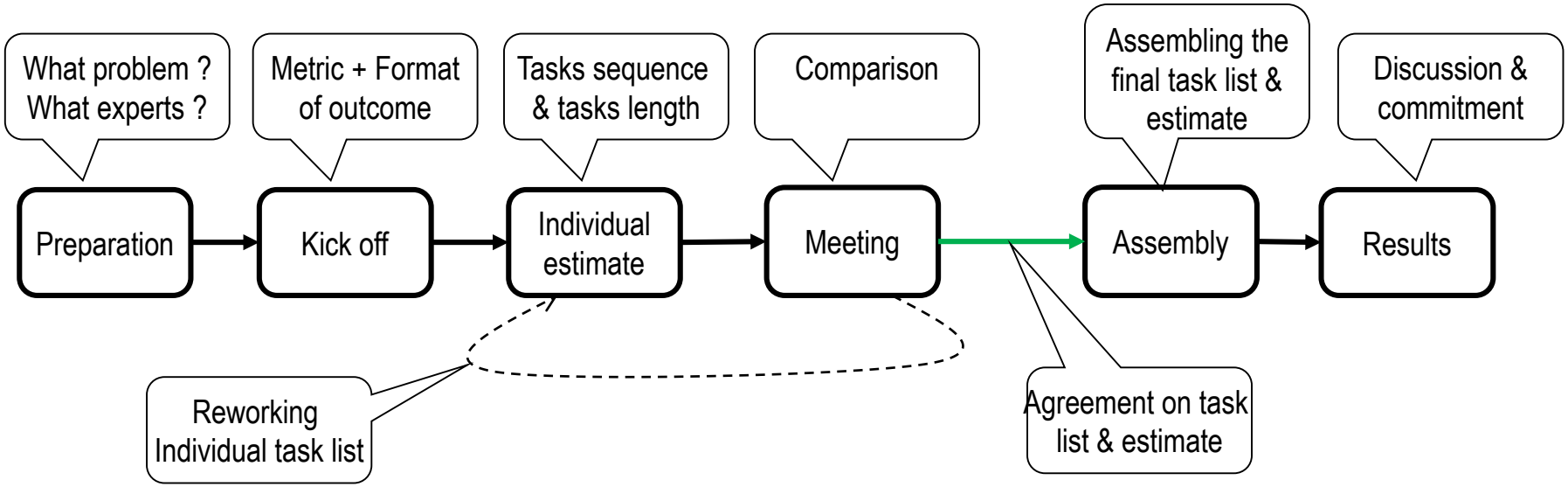
Updated risk list

Updated project plan

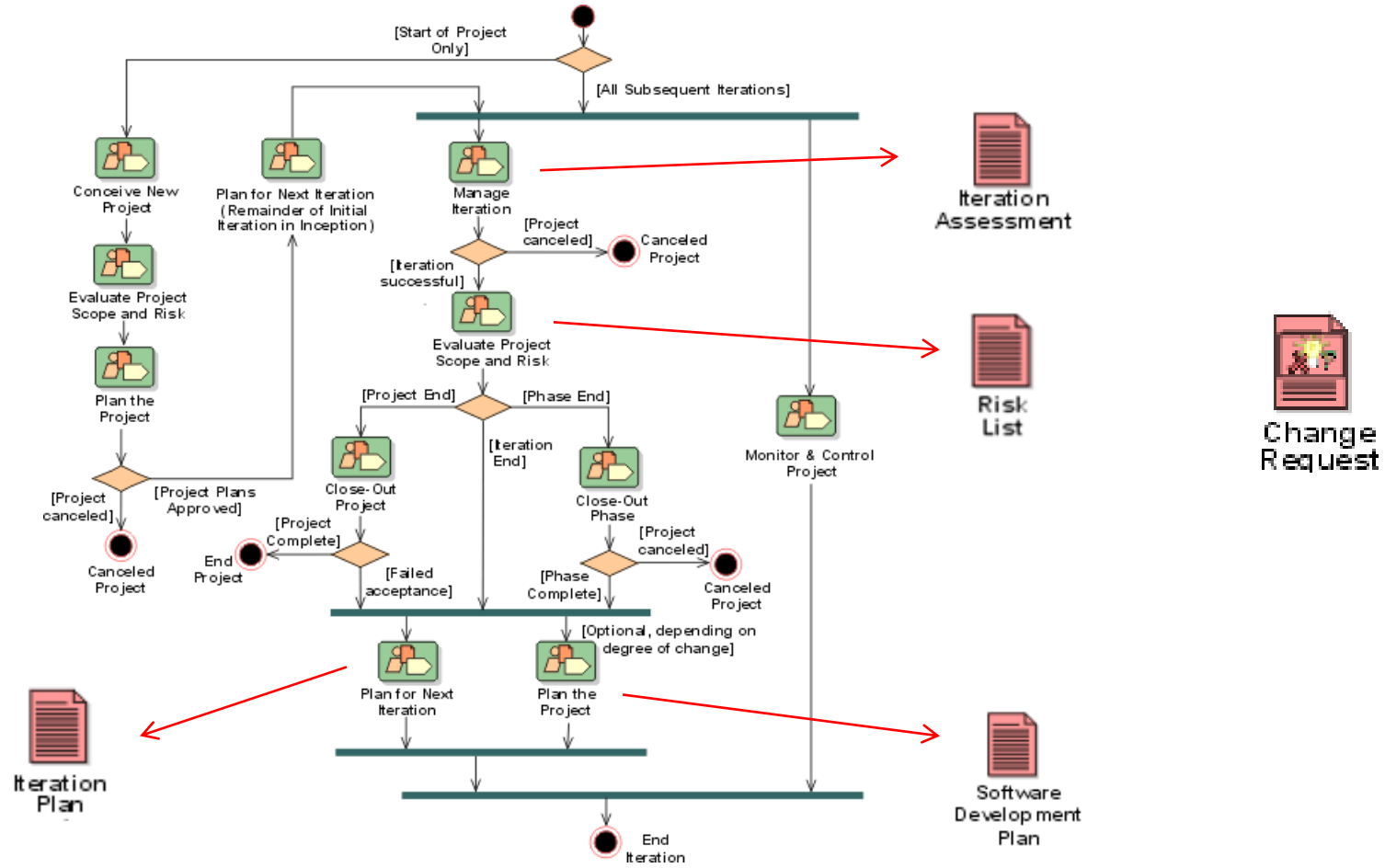
Next iteration's plan

Workload estimate : Delphi method

- Individual evaluation by a group of experts
- Synthesize the evaluations
- Group discussion to reach consensus



Key documents of the project management tasks



Typical number of iterations

Degrees of Iteration in Different Projects

Cycle Iteration Count	Total Iterations in Cycle	Number of Iterations per Phase [I, E, C, T]
Low	3	[0, 1, 1, 1]*
Typical	6	[1, 2, 2, 1]
High	9	[1, 3, 3, 2]
Very high	10	[2, 3, 3, 2]

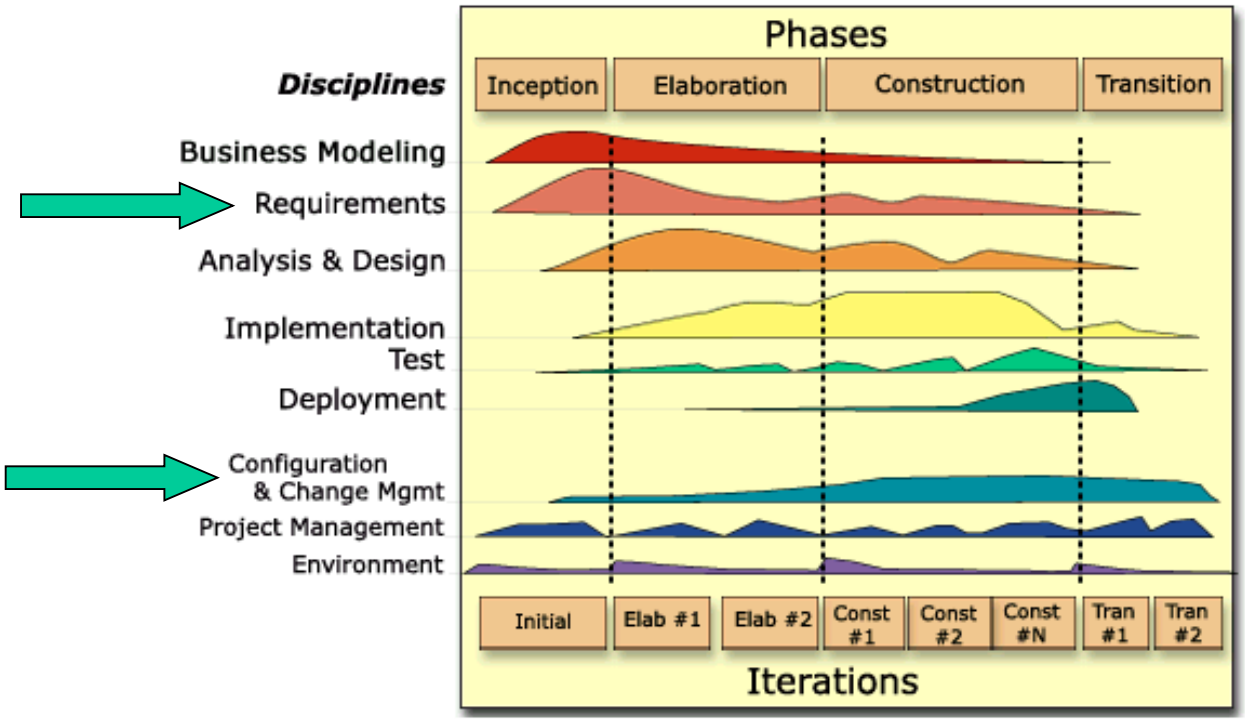
** [0, 1, 1, 1] denotes: Inception 0 iterations; Elaboration 1 iteration; Construction 1 iteration; Transition 1 iteration. Note that "0" does not mean that no work is being performed in that phase (Inception, for example), but rather that no software is being built.*
So, in general, plan to have between three and ten iterations. Observe, though, that the upper and lower bounds connote unusual circumstances; *most development cycles require six to eight iterations.*

Source: https://www.ibm.com/developerworks/rational/library/2831.html

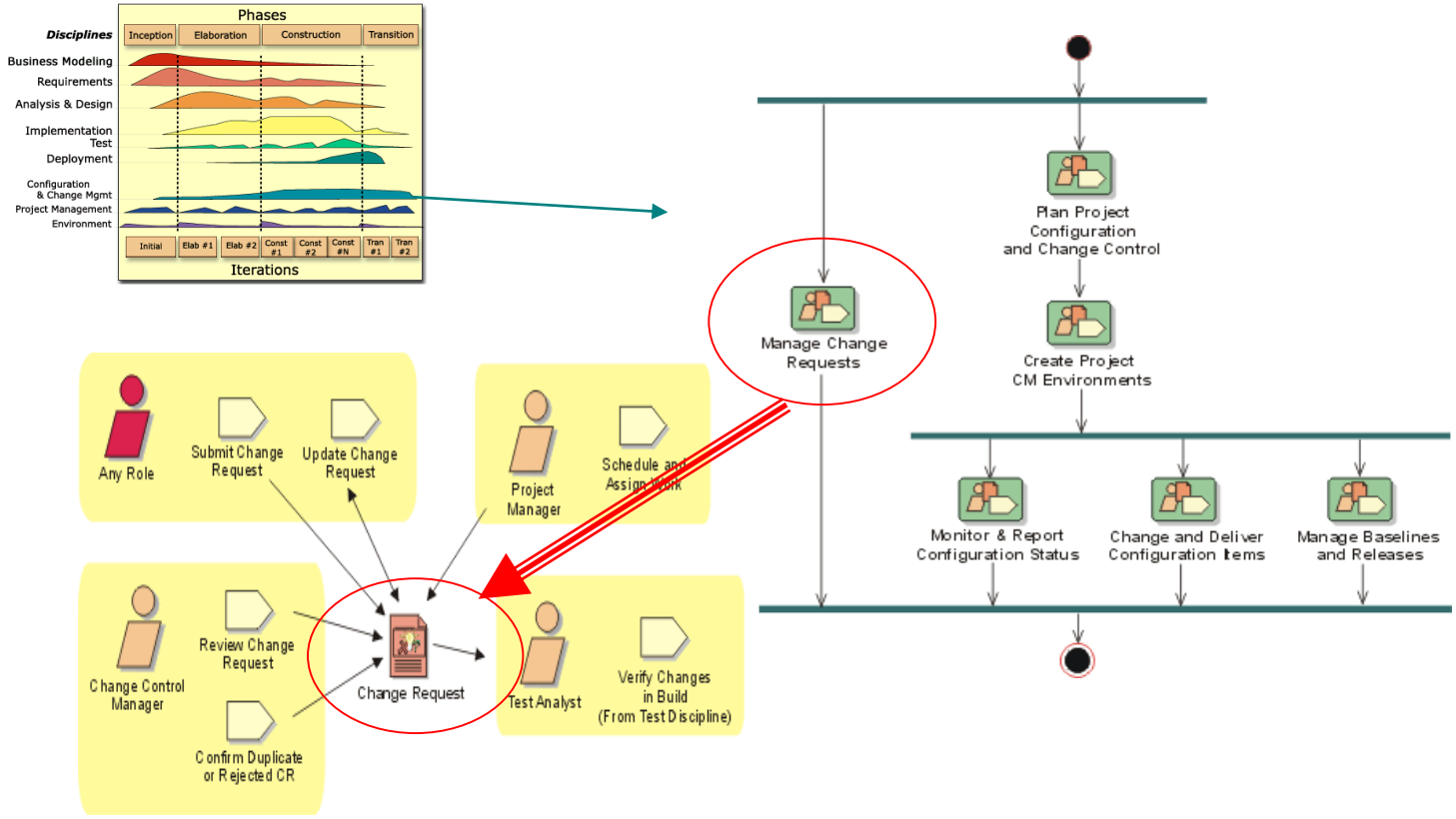
Change management

Manage the impact on the requirements

Manage change requests



Change management is a discipline in itself

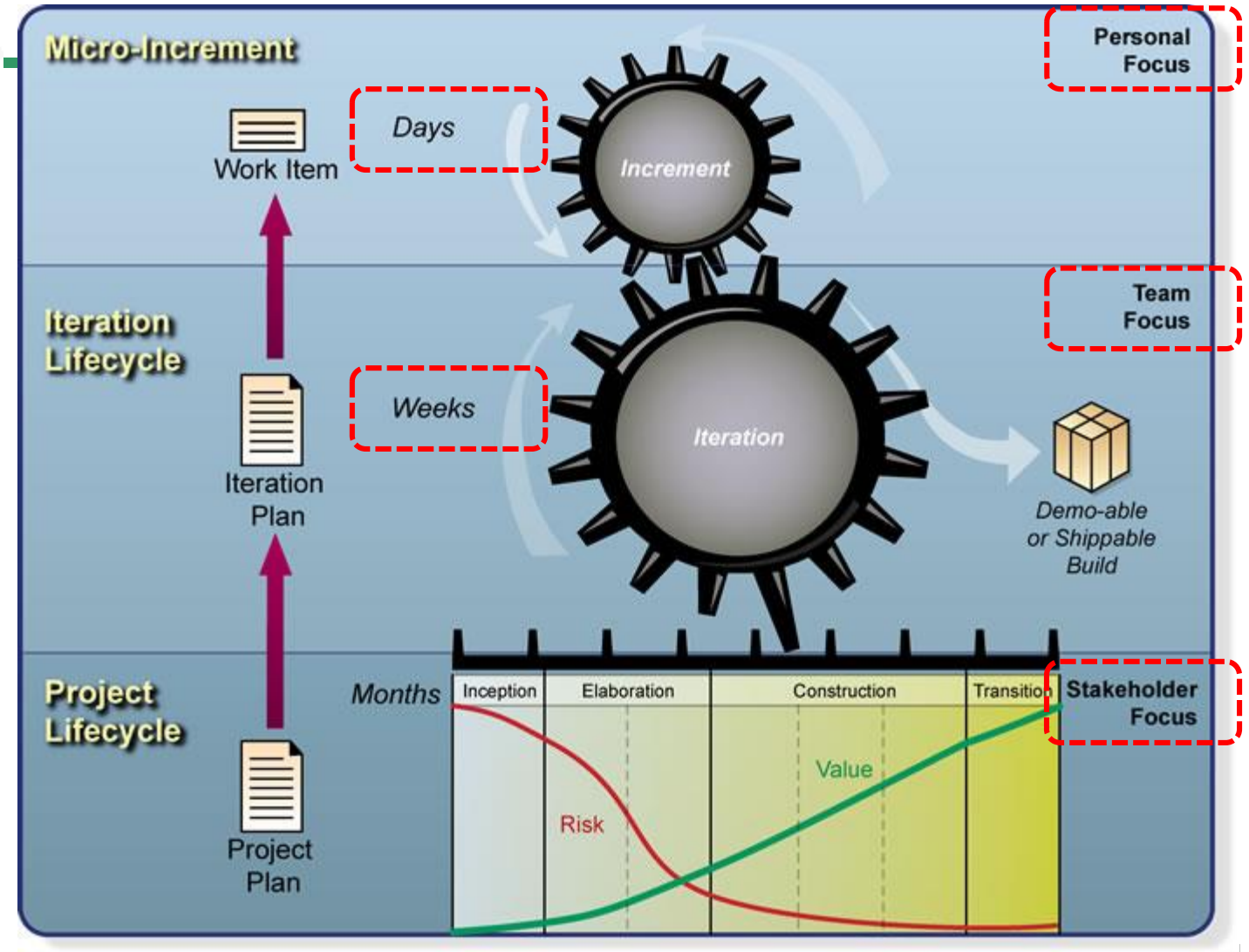


Images: Copyright (c) 1987-2003, Rational Software Corporation

Ce qu'il faut retenir

- Processus formalisé avec rôles et planning
- Orienté sur la gestion des risques
- La modélisation est centrale (traçabilité)
- Découpé en phases avec milestones
- Phase d'inception : go/nogo
- Les itérations sont de longeur variable
- La documentation est un livrable comme le code

Open UP


Micro-Increment
Personal Focus
Days
Work Item
Increment
Iteration Lifecycle
Team Focus
Weeks
Iteration
Iteration Plan
Demo-able or Shippable Build
Project Lifecycle
Months
Inception
Elaboration
Construction
Transition
Stakeholder Focus
Value
Risk
Project Plan

Open UP principles

- **Collaborate to align interests and share understanding.**
  - This principle promotes practices that foster a healthy team environment, enable collaboration and develop a shared understanding of the project.
- **Balance competing priorities to maximize stakeholder value.**
  - This principle promotes practices that allow project participants and stakeholders to develop a solution that maximizes stakeholder benefits, and is compliant with constraints placed on the project.
- **Focus on the architecture early to minimize risks and organize development.**
  - This principle promotes practices that allow the team to focus on architecture to minimize risks and organize development.
- **Evolve to continuously obtain feedback and improve.**
  - This principle promotes practices that allow the team to get early and continuous feedback from stakeholders, and demonstrate incremental value to them.

Open UP deliverables $\subset$ RUP deliverables