

Modellistica Medica

Maria Grazia Pia

INFN Genova

Scuola di Specializzazione in Fisica Sanitaria

Genova

Anno Accademico 2002-2003

Lezione 18-19

The Unified Process

Static dimension

Glossary

- **UP (Unified Process) or USDP (Unified Software Development Process)**
 - a process model, developed by the “three amigos” (Booch, Jacobson, Rumbaugh)
 - a theoretical model, publicly available
- **RUP (Rational Unified Process)**
 - a process framework, based on the UP
 - theoretical basis identical to the UP
 - practical tools, guidance, documentation etc. on top of the UP
 - commercial product (we bought a license for Geant4-INFN in Genova)

What is the Rational Unified Process?

- A **model** for software process
- A **product** for software process

- The RUP can be customised
 - to tailor a software process appropriate to an organisation or a project
 - RUP provides guidance for customisation (*e.g. mini-RUP for small size projects*)
 - but it is up to us to define the process we want

The philosophical principles in the RUP

- A software project team should plan ahead
- It should know where it is going
- It should capture project knowledge in a storable and extensible form

The spirit of the RUP

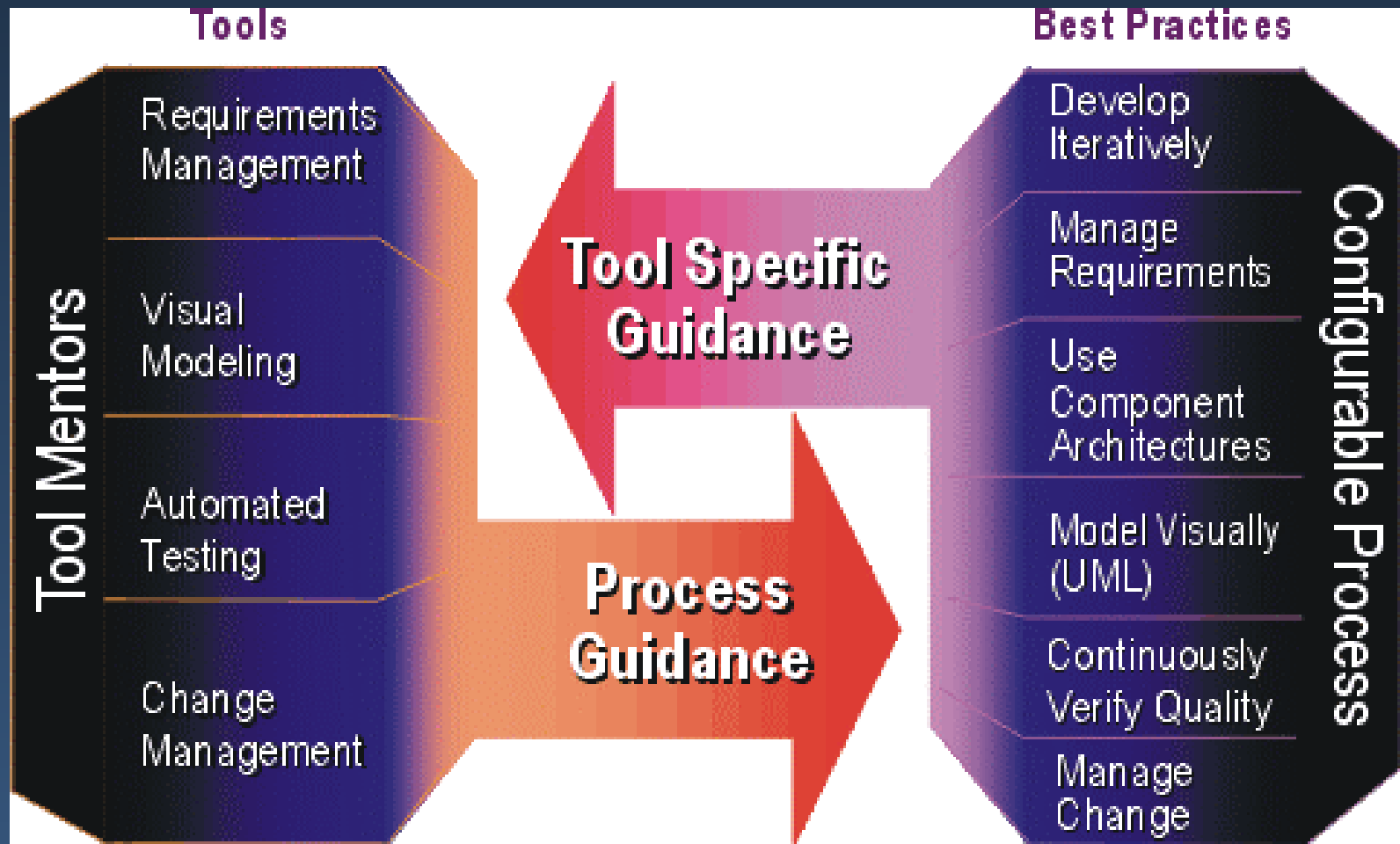
- Attack risks early and continuously... or they will attack you
- Ensure that you deliver value to your customers
- Stay focused on executable software
- Accommodate change early in the project
- Baseline an architecture early on
- Build your system with components
- Work together as one team
- Make quality a way of life, not an afterthought

a lot of common sense...

Best practices

RUP captures a set of “best practices” for software development teams

- now considered conventional wisdom by methodology practitioners



Other key principles

- Develop only what is necessary
- Focus on valuable results
- Minimize the production of paperwork
- Be flexible
- Learn from your mistakes
- Revisit your risks regularly
- Establish objective, measurable criteria for progress
- Automate what is human intensive, tedious and error prone
- Use small, empowered teams
- Have a plan

Major properties

- **Use case driven**
 - guided by interactions between the system and its users
- **Architecture centric**
 - founded on a defined architecture, with clear relationships among components
- **Iterative**
 - problem and solutions are organised into small pieces
- **Incremental**
 - each iteration builds incrementally on the previous one
- **Controlled**
 - control w.r.t. process: always know what to do next
 - control w.r.t. management: all artifacts and code under configuration management

The 10 essentials of RUP

- **Vision**
 - Develop a vision - analysing the problem, understanding stakeholder needs, defining the system
- **Plan**
 - Plan project schedule and resource needs, and track progress
- **Risks**
 - Identify and mitigate risks
- **Business case**
 - Determine whether or not the project is worth investing in
- **Process**
 - Adopt a process that fits your project
- **Architecture**
 - A systematic way to design, develop and validate a software architecture
- **Product**
 - Incrementally build and test the product
- **Evaluation**
 - Regularly assess results (focusing both on product and process problems)
- **Change requests**
 - Manage and control change
- **User support**
 - Provide assistance to the user

Why adopting the RUP?

- Because
 - it summarizes the experience of documented “best practices”
 - it is a synthesis of methodologies developed by recognized gurus
 - it obeys to some “reasonable” common sense
 - it is flexible and can be easily (?) customized
 - [*add any other good reason here*]
- My own personal reason (*in addition to all the above*):
 - it is a **product**
i.e., a tool we can use to achieve our objectives

Why adapting the RUP?

- Because any organization has peculiar characteristics
- Because any project has peculiar characteristics

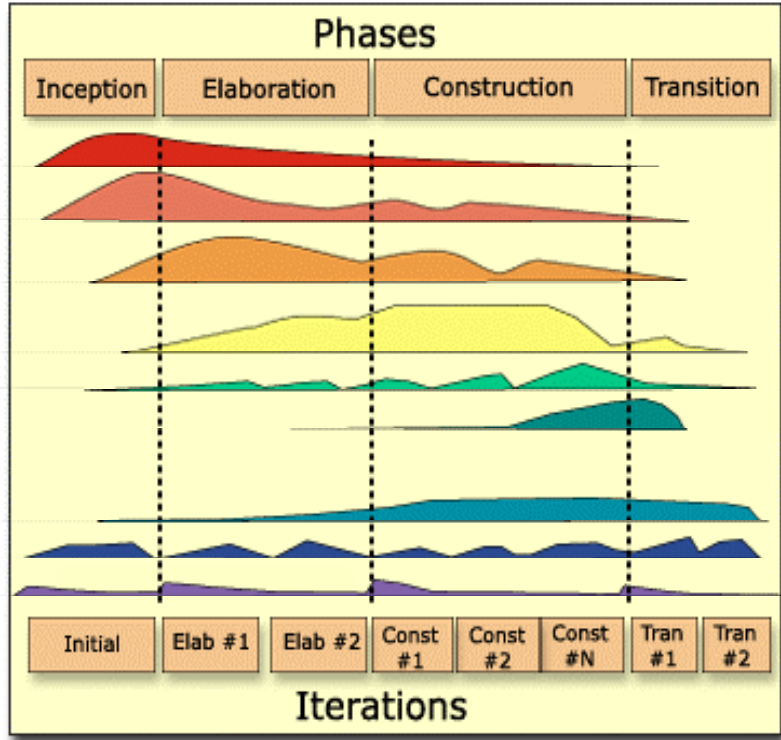
- Because all the above share common characteristics, needs, background etc.

- The RUP provides a framework where we can
 - exploit the commonality (reuse methods and tools across projects)
 - tailor the process to the individual needs of each project

Two dimensions

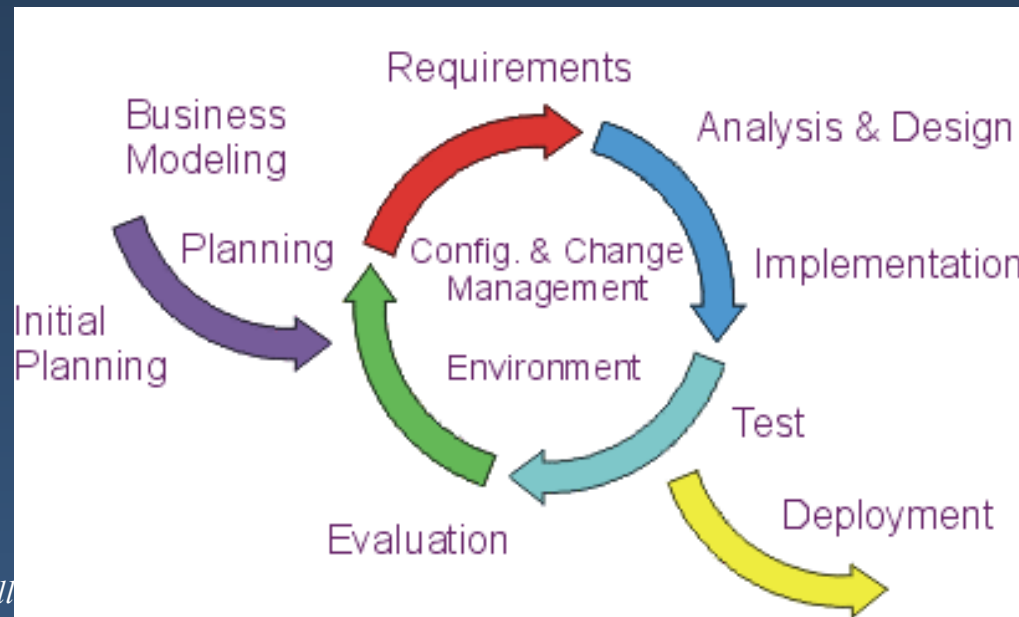
Content

- **Static** aspect of the process
- Described in terms of
 - disciplines
 - activities
 - artifacts
 - workers



Time

- **Dynamic** aspect of the process
- Expressed in terms of
 - cycles
 - phases
 - iterations
 - milestones



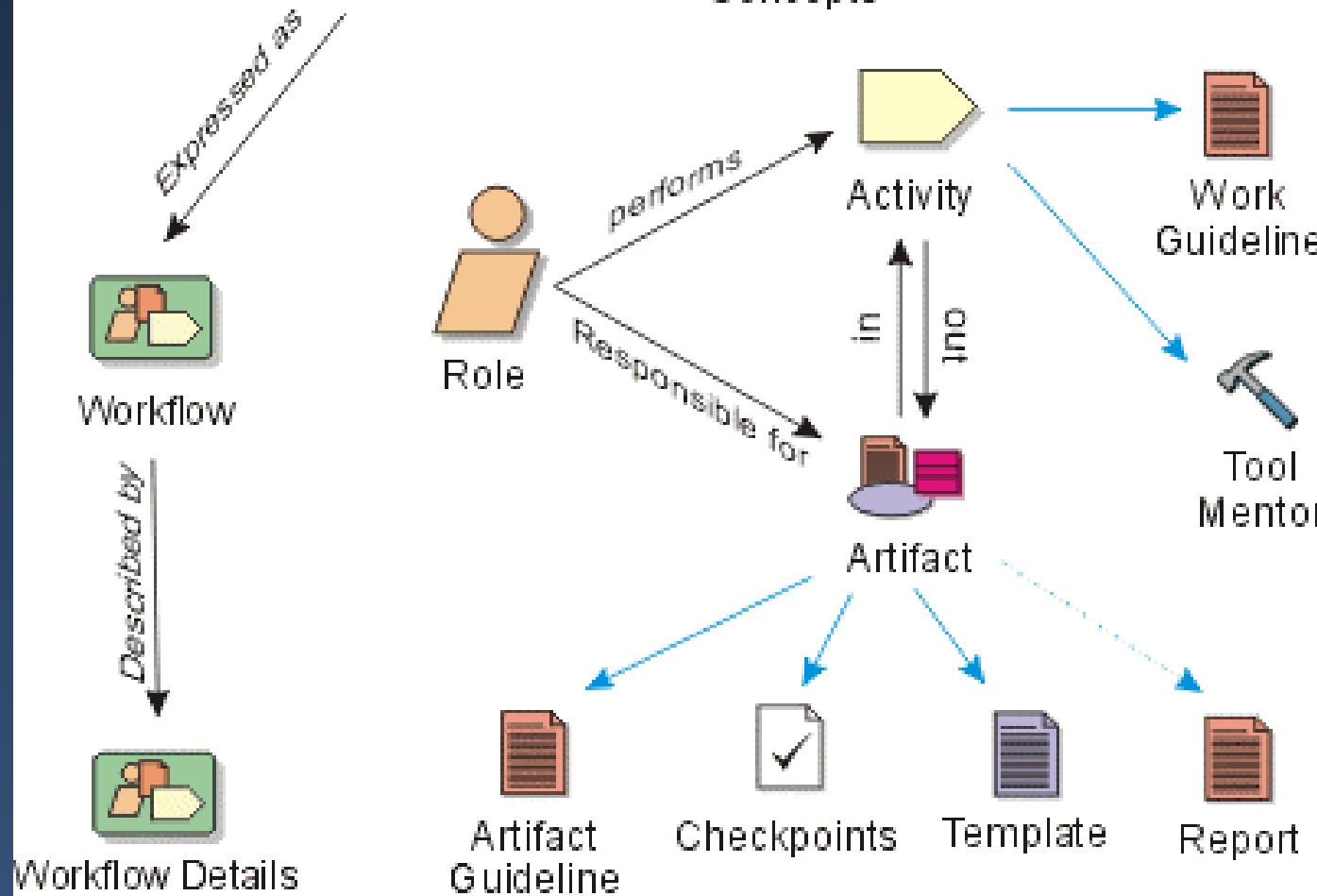
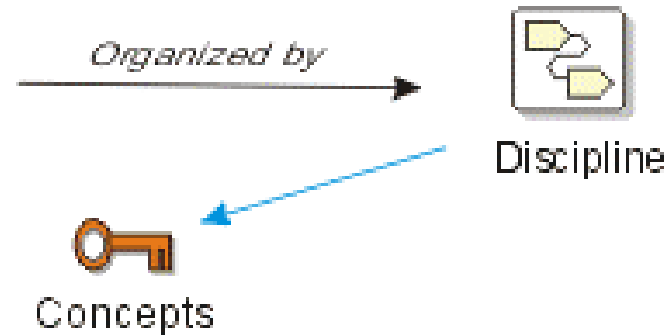
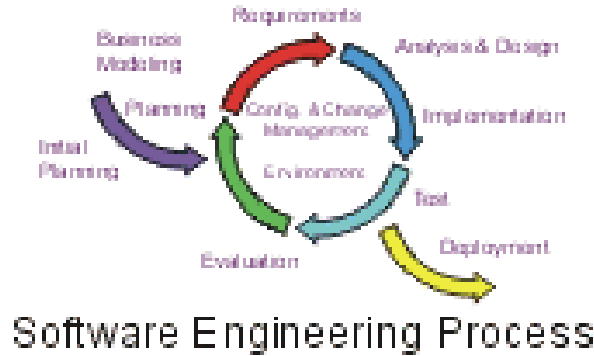
Key concepts of the RUP and their relationships

Focus on modeling



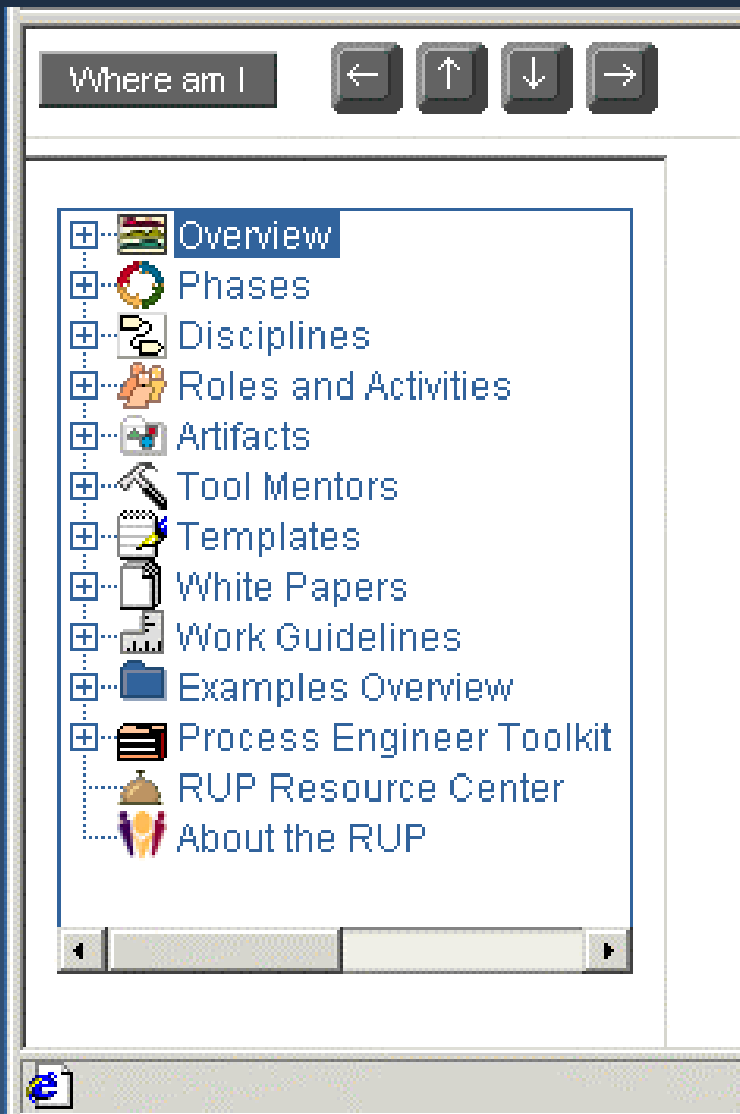
Architecture-centric process

Use case driven process



Refers to

The product



A web-enabled searchable knowledge base

- Extensive guidelines for all parts of the software life-cycle
- Tool mentors
- Templates
- Examples
- Development kit
 - to customise and extend the RUP

The static structure

- A process describes who is doing what, how and when
- RUP has 4 primary modeling elements:
 - roles (*who*)
 - activities (*how*)
 - artifacts (*what*)
 - workflows (*when*)
- These elements are grouped into disciplines

- **Role**

- Defines the behaviour and responsibilities of an individual or a team

- **Activity**

- A unit of work
- Has a clear purpose (usually creating or updating some artifacts)
- Examples: Plan an iteration, Review the design, Execute performance test etc.

- **Artifact (= work product)**

- Examples: a Design model, a Model element (e.g. a use case), a Document, source code, executable etc.

- **Workflow**

- A sequence of activities that produces a result of observable value
- Can be expressed in UML as a sequence diagram, a collaboration diagram, an activity diagram
- Not to be interpreted literally as a programme for people to be followed exactly and mechanically!

Core process workflows

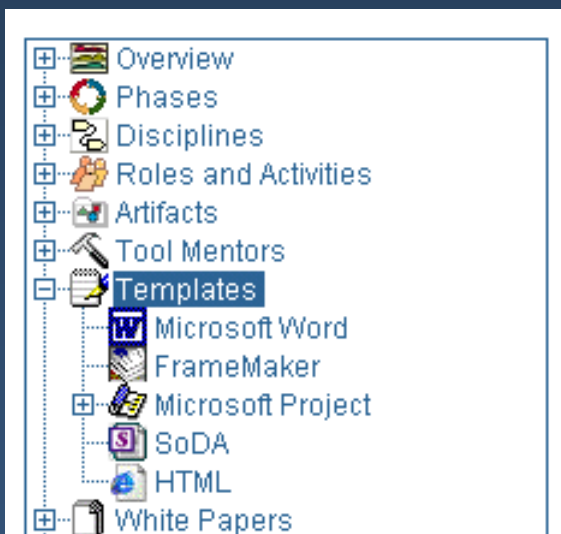
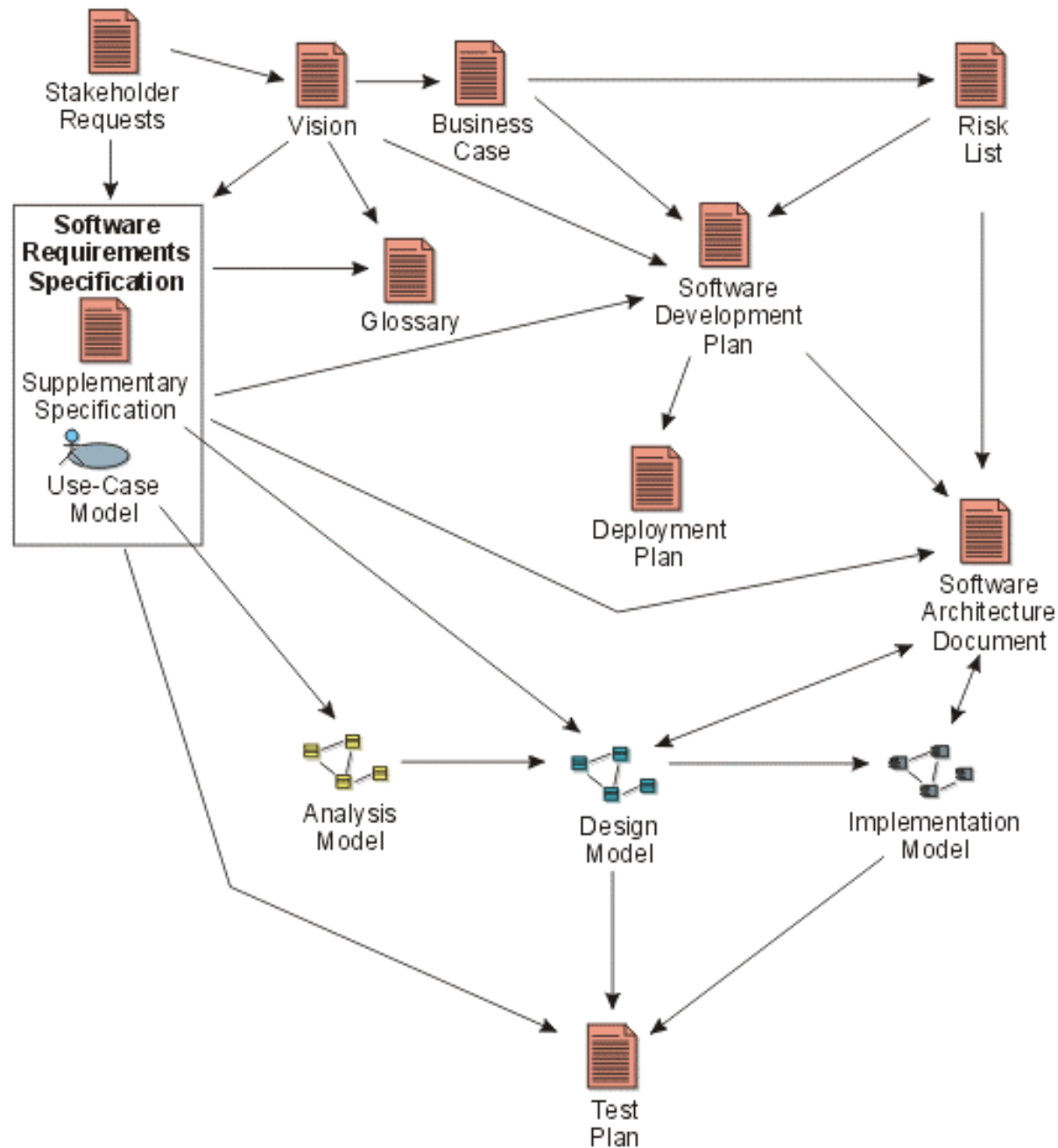
- 6 core engineering workflows
 - Business modeling workflow
 - Requirements workflow
 - Analysis and design workflow
 - Implementation workflow
 - Test workflow
 - Deployment workflow
- 3 core supporting workflows
 - Project management workflow
 - Configuration and change management workflow
 - Environment workflow
- Workflows are revisited again and again throughout the life-cycle
 - various emphasis and intensity at each iteration

Major artifacts in the RUP

The RUP has provision for > 100 artifacts

We don't want all of them...

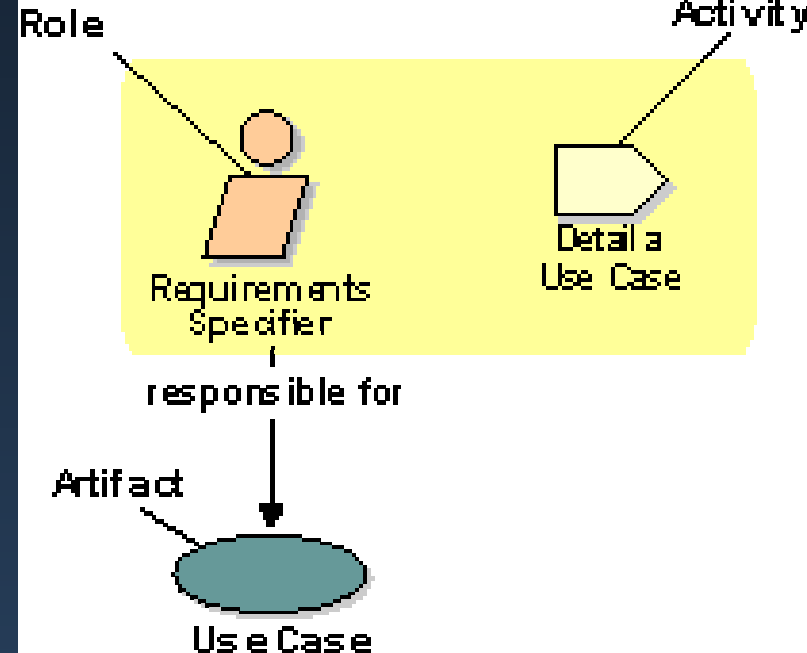
Critically selecting which ones are relevant to us is part of the customization



Roles





















A role is an abstract definition of a set of activities performed and artifacts owned























Role sets in RUP: {
Analysts
Developers
Testers
Managers
Other roles



- Roles are typically realized by an individual or a team
- A project team member typically fulfills many different roles
- Roles are not individuals
 - they describe how individuals behave and what responsibilities they have
- Roles have a set of cohesive activities that they perform
- Activities are closely related to artefacts
- Not to be forgotten among the roles: *stakeholders*

Mentors and guidelines

- [-]  Tool Mentors
 - [+]  Rational Suite AnalystStudio
 - [+]  Rational Administrator
 - [+]  Rational ClearCase
 - [+]  Rational ClearQuest
 - [+]  Rational Process Workbench
 - [+]  Rational RequisitePro
 - [-]  Rational Rose
 -  Detailing Business Workers and
 -  Finding Business Actors and Use
 -  Detailing a Business Use Case
 -  Finding Business Workers and E
 -  Structuring the Business Use-Case
 -  Finding Actors and Use Cases
 -  Detailing a Use Case
 -  Structuring the Use-Case Model
 -  Capturing the Results of Use-Case
 -  Creating Use-Case Realizations
 -  Documenting the Deployment Model
 -  Documenting the Process View

- [+]  Overview
- [+]  Phases
- [+]  Disciplines
- [+]  Roles and Activities
- [+]  Artifacts
- [+]  Tool Mentors
- [+]  Templates
- [+]  White Papers
- [-]  Work Guidelines
 -  Assessment Workshop
 -  Business Object Modeling Workshop
 -  Development Case Workshop
 -  Requirement Workshop
 -  Use-Case Workshop
 -  Use-Case Analysis Workshop
 -  Interviews
 -  Brainstorming and Idea Reduction
 -  Storyboarding
 -  Role Playing
 -  Review Existing Requirements
 -  Fishbone Diagrams
 -  Pareto Diagrams
 - Reviews

Templates

- Rose examples and templates
- SoDA templates
 - help automate documentation
- MS Word templates
 - assist documentation in all workflows
- MS Project plan templates

Requirements: concepts Requirements: guidelines

Requirements Management

Requirements

Traceability

Types of Requirements

Use-Case View

User-Centered Design

Activity Diagram in the Use-Case Model

Actor-Generalization

Actor

Boundary Class

Communicate-Association

Extend-Relationship

Include-Relationship

Requirements Management Plan

Software Architecture Document

Software Requirements Specification

Use Case

Use-Case Diagram

Use-Case-Generalization

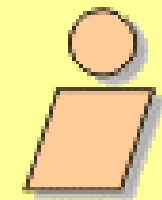
Use-Case Model

Use-Case Package

Use-Case Storyboard

User Interface (General)

Requirements: activities and roles



System Analyst



Develop Requirements Management Plan



Develop Vision



Elicit Stakeholder Requests



Capture a Common Vocabulary



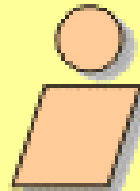
Find Actors and Use Cases



Manage Dependencies



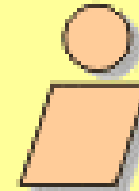
Structure the Use-Case Model



Software Architect



Prioritize Use Cases



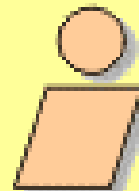
Requirements Specifier



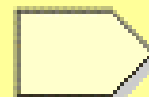
Detail a Use Case



Detail the Software Requirements



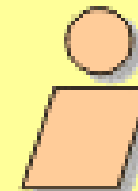
User-Interface Designer



Model the User-Interface



Prototype the User-Interface

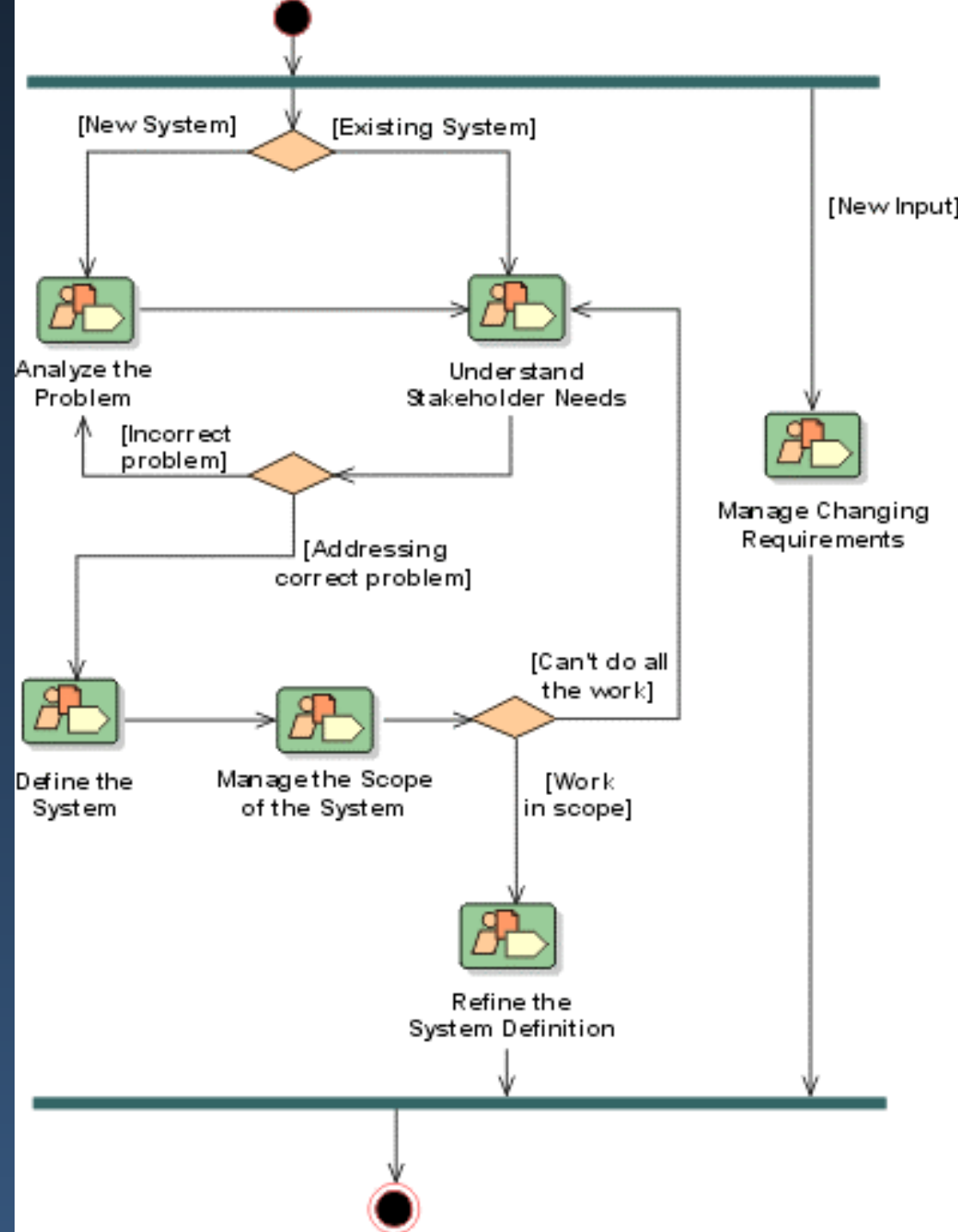


Requirements Reviewer

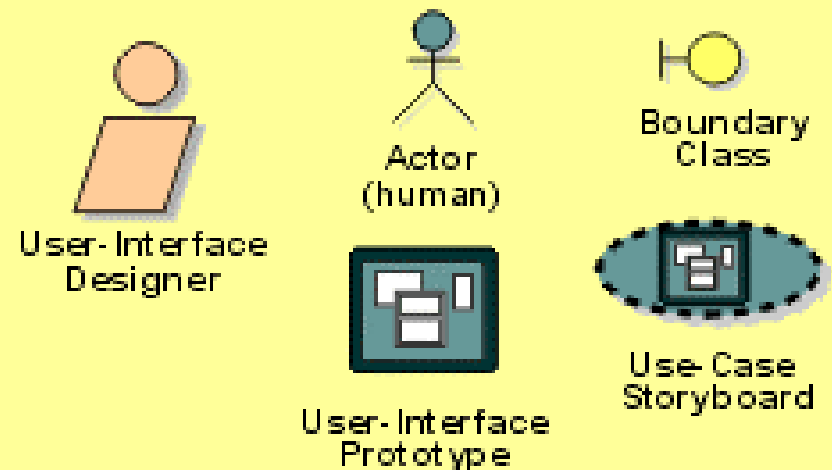
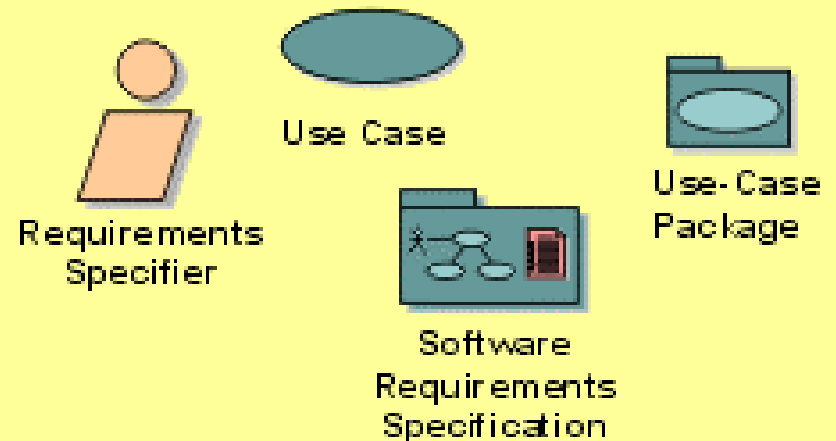
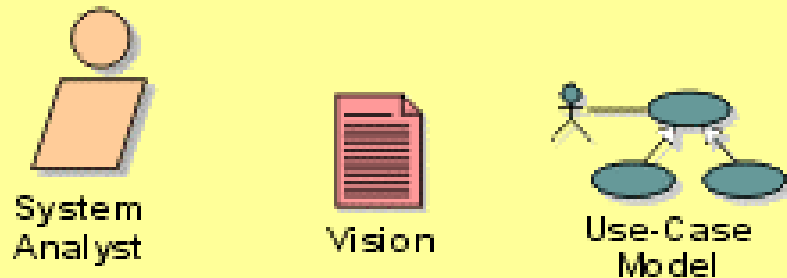
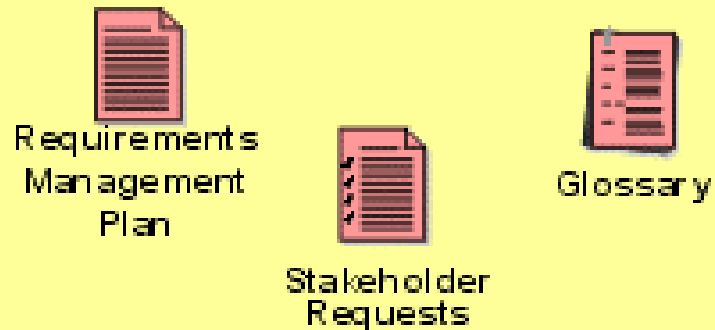


Review Requirements

Requirements: workflow

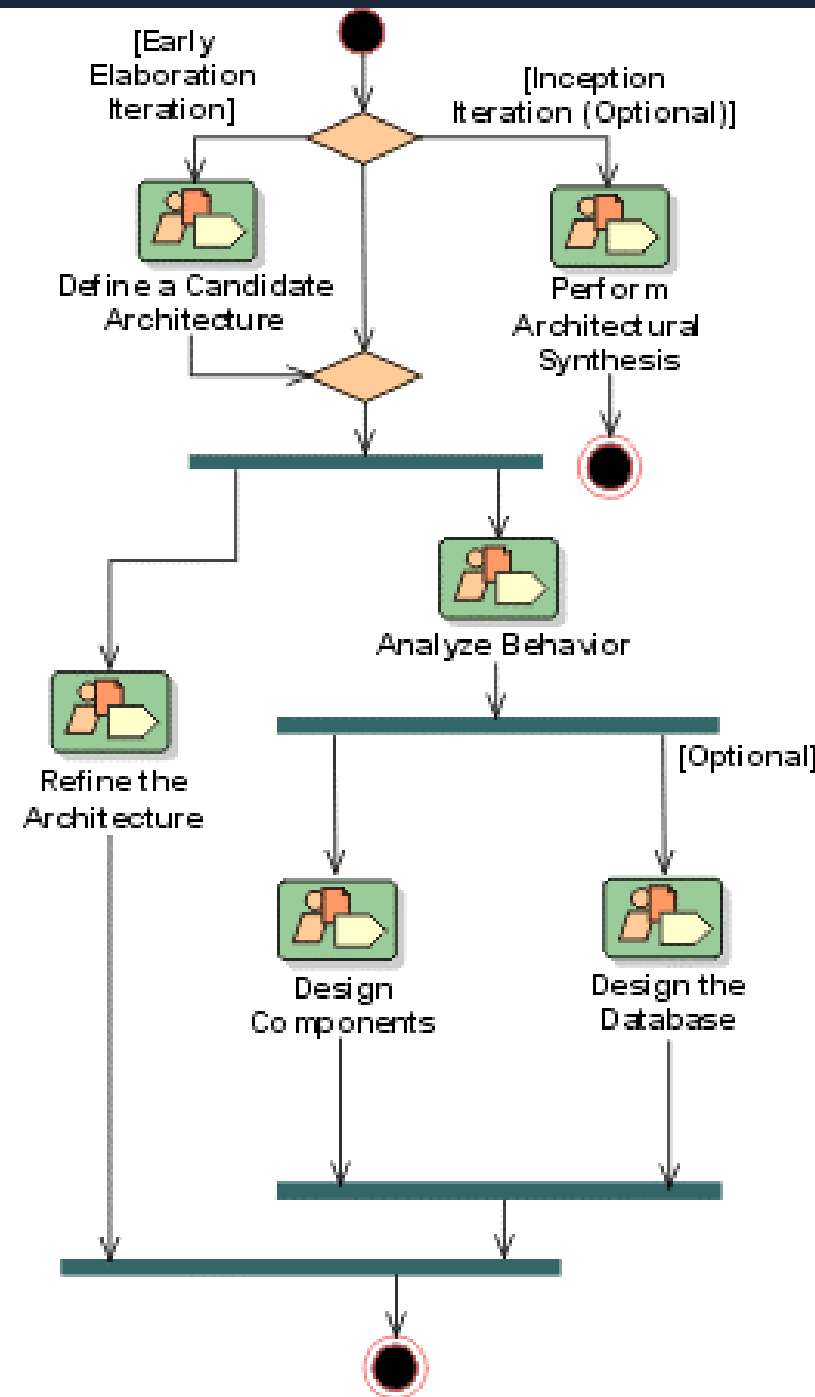


Requirements: artifacts



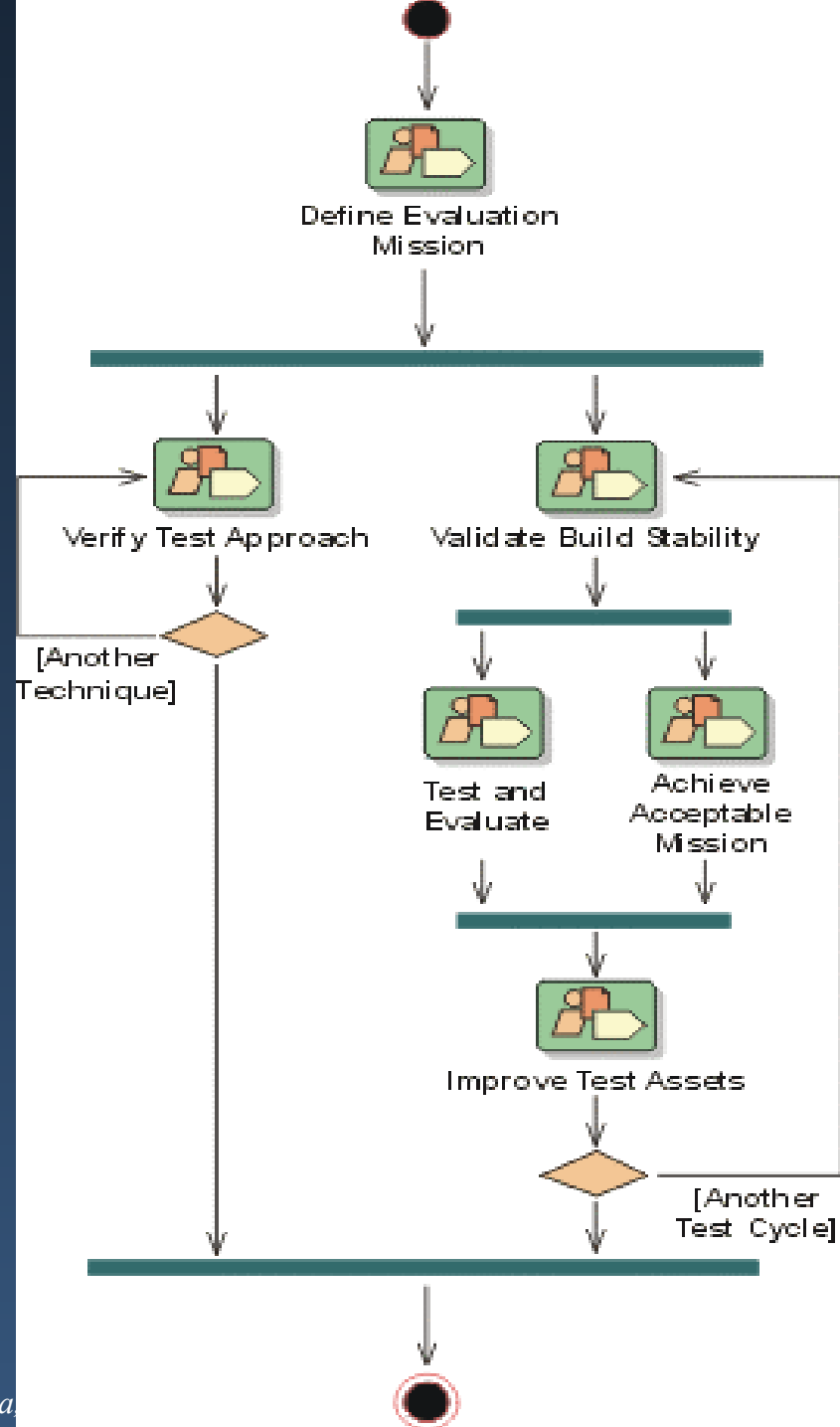
Analysis and designs concepts and workflow

- Analysis Mechanisms
- Concurrency
- Deployment View
- Design and Implementation Mechanisms
- Distribution Patterns
- Events and Signals
- Layering
- Logical View
- Process View
- Representing Graphical User-Interfaces
- Representing Interfaces to External Systems
- Software Architecture
- Test-first Design
- Web Architecture Patterns



Test concepts and workflow

Acceptance Testing
Exploratory Testing
Key Measures of Test
Performance Testing
Product Quality
Quality Dimensions
Stages of Test
Structure Testing
Test Automation and Tools
Test-Ideas Catalog
Test-Ideas List
Test Strategy
The Lifecycle of Testing
Types of Test
Usability Testing



Using the RUP for SPI

- The RUP can be used as a tool for SPI
 - provides lot of guidance, mentoring etc.
- Assessments of the RUP against the two major SPI frameworks
 - CMM
 - ISO 15504
- Detailed papers (*useful reading...*)
- In most cases (in the areas where applicable) the RUP corresponds to achieving level 3 in either framework

RUP vs. XP

The exercise of comparison can be instructive

Huge amount of documentation available on XP
(it's fashionable...)

- The scope is different

- RUP: general process, can address both large scale projects as well as small ones, can be customised
- XP: for “small” projects

- The guidance is different

- Activities, work products etc. are often vague or implicitly assumed in XP

- The RUP can be customised into XP (dX)

The RUP as a process framework

- RUP is a process framework
 - can be adapted and extended to suit the needs of the adopting organization
 - make the process as lean as possible, still fulfilling its mission
 - complement the process with the best practices, specific rules and procedures of the organization
- RUP itself contains several variants
 - pre-packaged development cases for different types of organizations
- RUP contains guidelines for its modification and configuration
 - also a tool (RUP Builder)
- However the hard part is not changing the RUP
 - it is changing how we think about software development

Configuring and implementing the RUP

Configuring the RUP

- adapt the process product to the needs of the adopting organization
- modify the process framework
- the result of configuring is captured in a development case

Implementing the RUP

- change the organization's practice so that it routinely uses the RUP

Configuring the process

- Adaptation is possible at two levels
 - organization-wide process
 - project-specific process
- Modifying the online version
 - add, remove, expand, modify artifacts, activities, workers
 - add, remove, expand, modify step in activities
 - add guidelines, based on discoveries in past projects
 - tailor the templates
 - add tool mentors
- Hard changes
 - using a different process model
 - changing the core workflow structure
 - changes in process terminology

Guidelines for configuring the RUP

- It provides guidance to configure itself
- Sorting through the many artifacts, activities and roles
 - *do we need this one?*
- Build a framework first
 - address all the key elements of a process before focusing on a specific area
 - once the framework is in place, can address effectively a particular area
- Do not include activities and artifacts that cannot be clearly justified
- Minimize formal intermediate artifacts
- Use convenient formats
- Regularly re-visit the process
- Tailor while retaining best practices

Implementing the RUP step by step

- Assess the current state
- Set/revise goals
- Identify risks
- Plan the process implementation
- Execute the process implementation
- Evaluate the process implementation

Implementing the RUP: what the RUP says

Process changes are difficult and it may take time to see their true effects
A process change affects the individuals and the organization more deeply than changing technology or tools

- identify the opportunity and the benefits
- convey them clearly to the interested parties
- raise their level of awareness
- gradually change from the current practice to a new one

Areas to be addressed when implementing a process:

- the **people** and their competences, skills, motivation, and attitude
 - *everyone needs to be adequately trained and motivated*
- the supporting **tools**
- the software development **life cycle model**
 - *its organizational structure, underlying activities, practices, artifacts*
- the actual **description** of the software development process

Approaches to the implementation

• Typical approach

- configure the process and describe it in a development case
- first implement the process in a Pilot Project
- feedback from the Pilot project into the development case
- once the process is tested and verified, can rolled out to a broader group
- the Pilot Project is typically 10-15 people, 3-4 month duration

• Distributed approach

- several pilot projects in parallel

• Fast approach

- use the process and tools directly in actual projects
- no Pilot Project

• Careful approach

- the initial Pilot Project is successively refined in several pilot projects
- use of the new process spread slowly through small projects

Each approach has its pro and contra

Training and mentoring is essential in all cases

A development case

- A tailored, project-specific instance of the RUP
- Defines:
 - what will be developed
 - which artifacts are really needed
 - which templates should be used
 - which artifacts already exist
 - which roles will be needed
 - which activities will be performed
 - which guidelines, project standards and tools will be used

Success and failure factors

Top reasons for failure

- Failure to introduce process and tools incrementally
- Lack of management support
- Lack of buy-in from sponsors and stakeholders
- Unwillingness/incapacity for organizational change
- Vision and change drivers are unclear

Strategies for success

- Assess the project and the organization
- Implement process and tools incrementally
- Manage and plan
- Use mentors
- Distribute process ownership
- Be pragmatic
- Communicate
- Train people

*Based on experiences with large financial organizations
May not be the same in HEP ...*

Using the RUP

- The web application contains
 - process documentation
 - guidelines
 - templates
 - tool instructions
 - etc.
- Team members use the web application
 - to learn about the software development process
 - to access templates used to create process artifacts
 - to browse existing process artifacts