INF5181: Process Improvement and Agile Methods in Systems Development

Lecture 02:

Processes and Process Modeling (Section A)

UNIVERSITETET I OSLO

Dr. Dietmar Pfahl

email: dietmarp@ifi.uio.no

Fall 2011

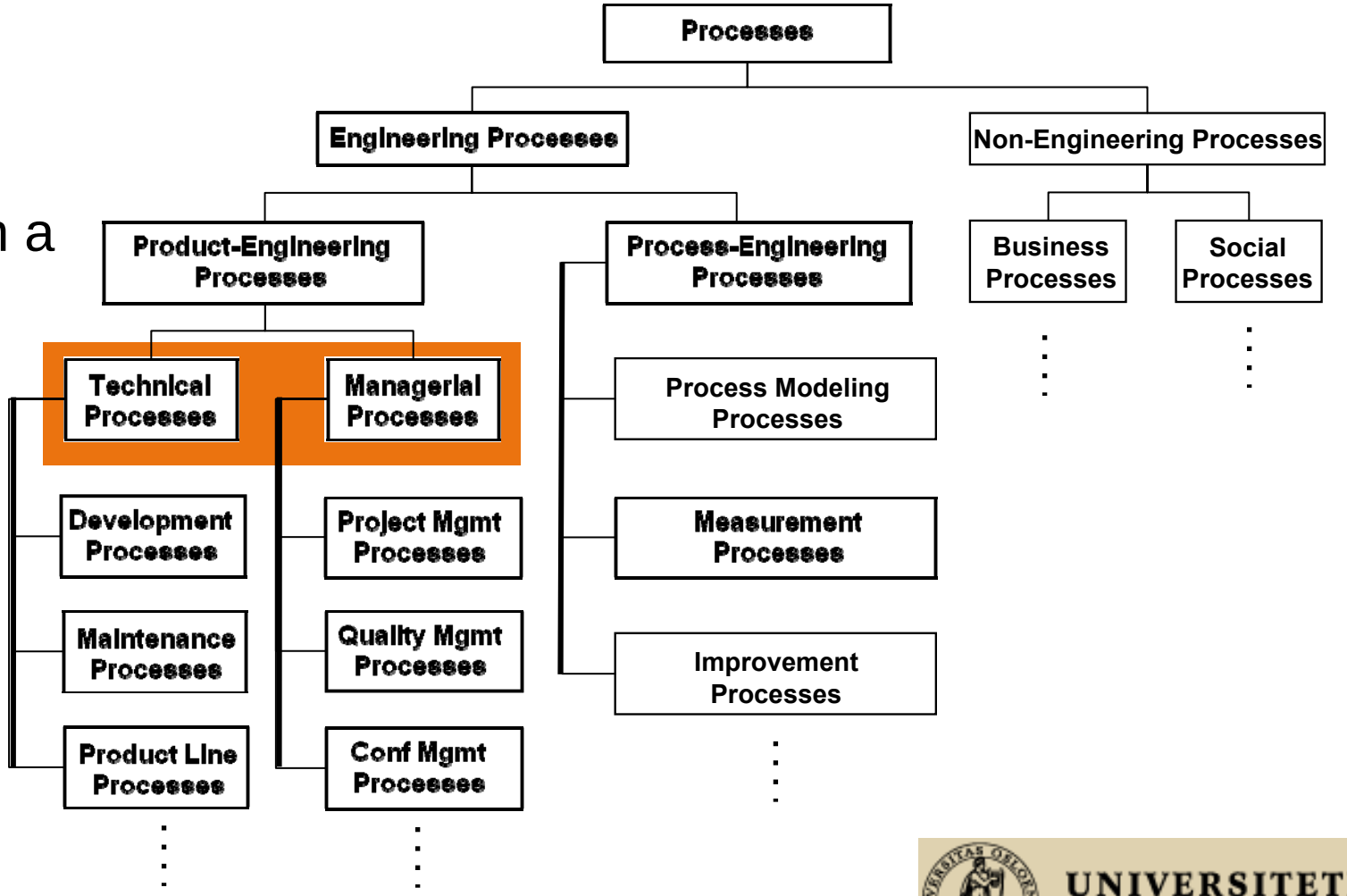
Structure of Lecture 02

- Hour 1:
  - Introduction into Process Modelling
  - Prescriptive Process Models
- Hour 2:
  - Process Families/Standards
  - Descriptive Process Modelling
- Hour 3:
  - Exercises

Process Taxonomy

H. Dieter Rombach, Martin Verlage, Directions in Software Process Research, Advances in Computers, Volume 41, Marvin V. Zelkowitz (Ed.), Pages 1-63, Academic Press, Boston, MA, 1995.

- What are typical processes in a software project?



UNIVERSITETET I OSLO

What is a (Software Development) Process?

A Process …

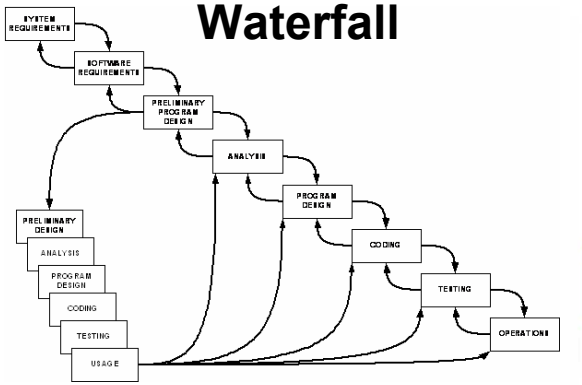
- defines **Who** is doing W… each a specific goal.
  - In software engin… oftware product or to enhance an ex…

An Effective Process …

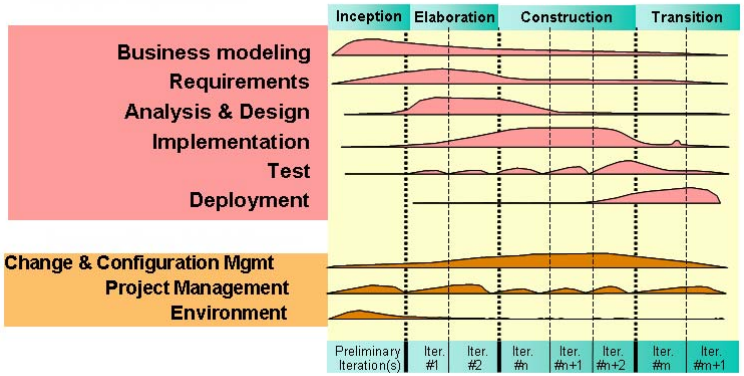
- provides guidelines for … quality software
- reduces risk and increase…
- promotes common vision and…

Software Process Examples

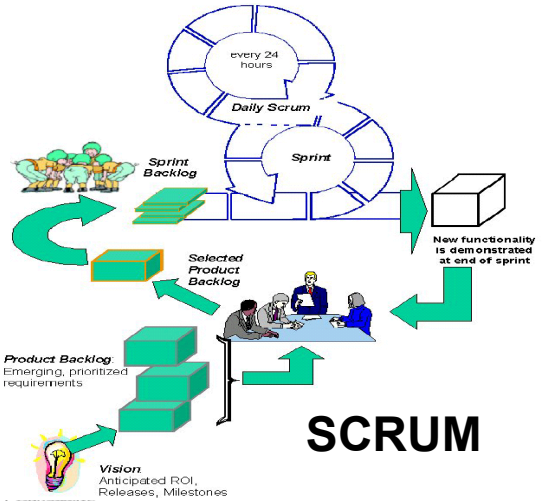
Waterfall



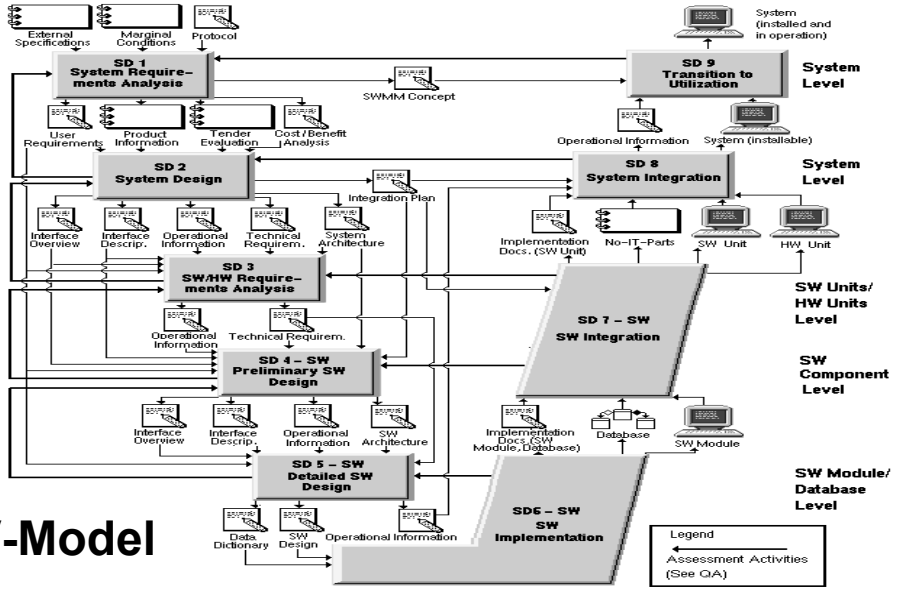
RUP



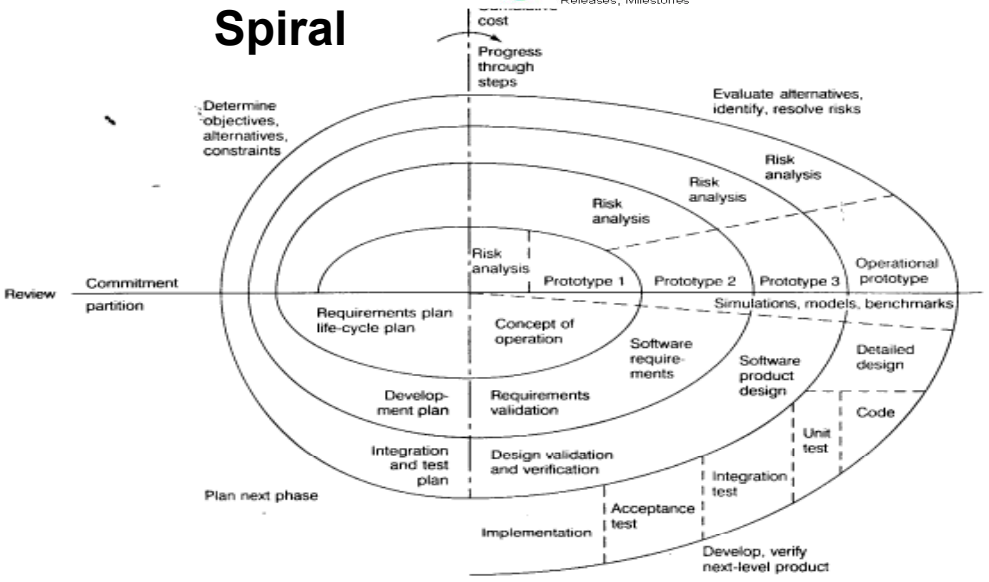
SCRUM



V-Model



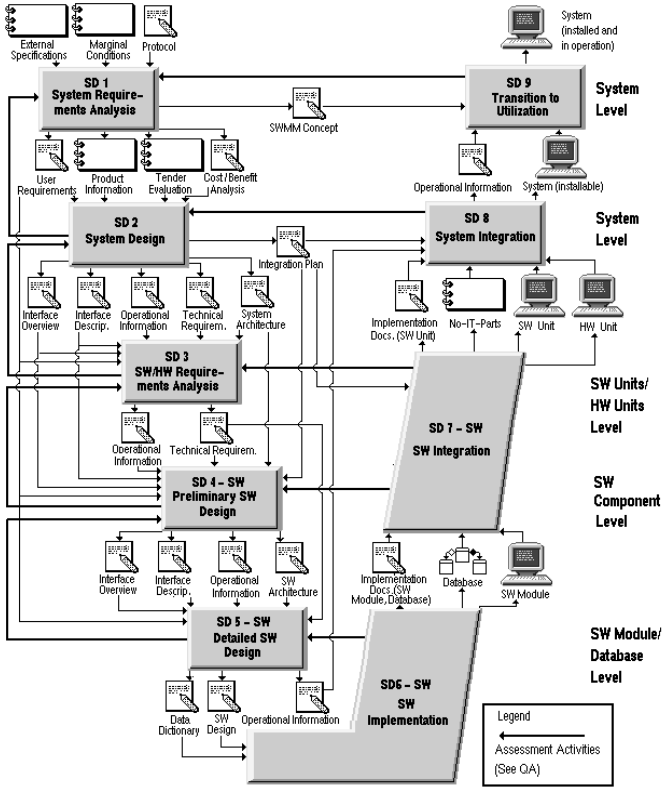
Spiral



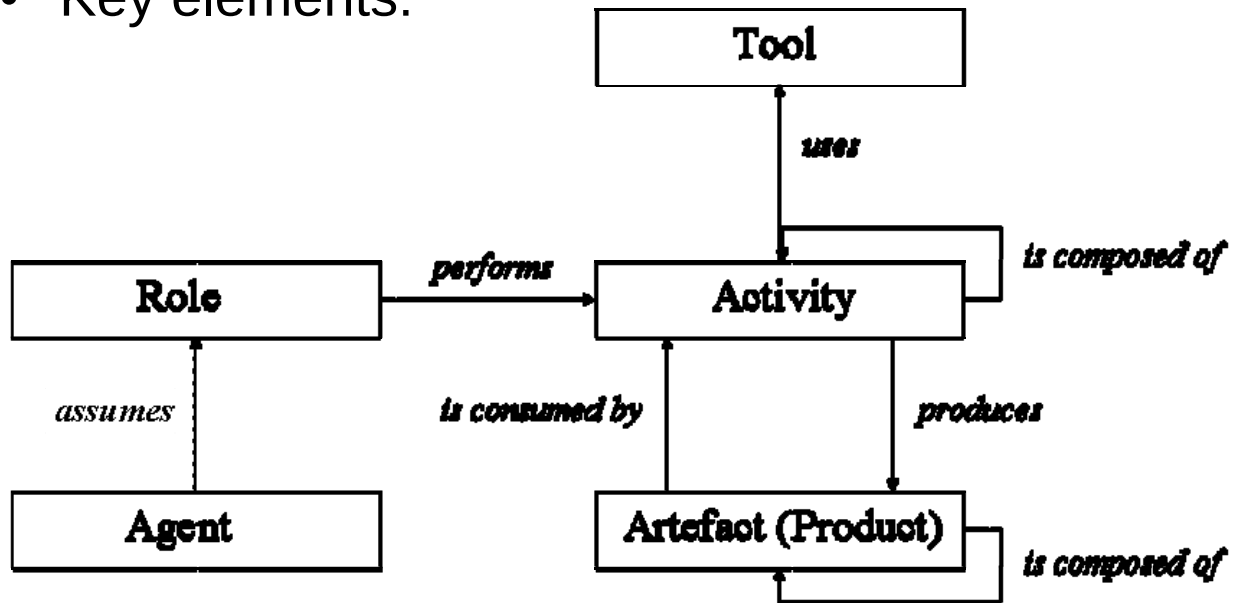
What are the Goals of Process Modeling?

- To enable effective understanding and communication
  - At one development site (developers, teams, ...)
  - Between development sites (distributed development, outsourcing, contractor-supplier relations, ...)
- To improve software development activities
  - Improving real processes requires measurement and measurement requires defined processes
  - Evolving processes
- To support project management
  - Transparency, tracking, ...
- To guide the developers
  - Incorporating new employees
- To support automatic process enactment
  - Workflow support
  - CASE tools
- To support reuse of process knowledge
  - Organsational learning

What is a (Software) Process Model?

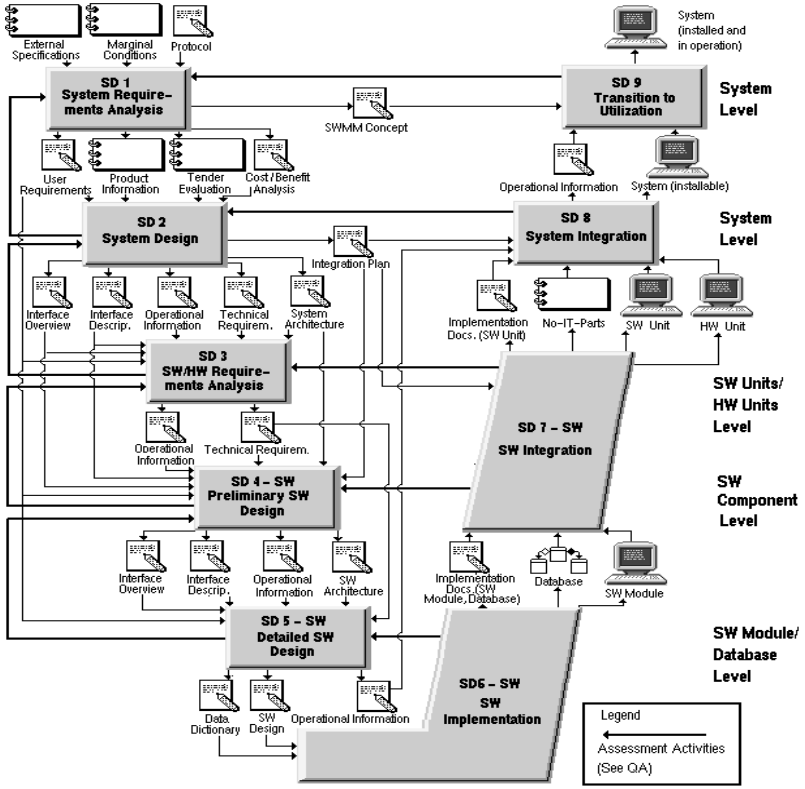


- "Software Process Model: An abstract software process description. It can be more or less formal." [Lonchamp 93]
- Key elements:



UNIVERSITETET I OSLO

Characterization of Process Models



A *Process Model* defines:

- an identifiable activity or a group of activities
- a hierarchy of activities
- the sequence/order of activities (→ control flow)
- the input/output products (artifacts) of activities (→ product flow)
- the relations between activities and techniques, methods, tools, and roles

The Role Concept

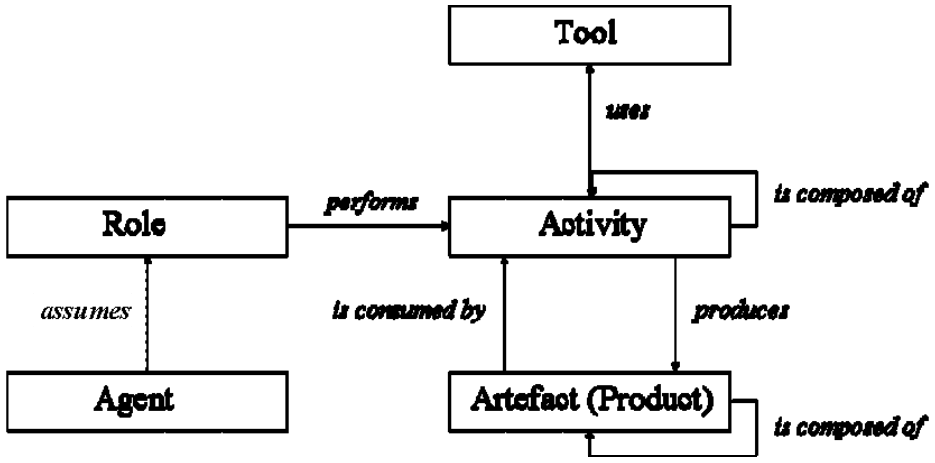
- *Role*
  - A role is in charge of one or more activities defined in one or more processes
  - A role has defined responsibilities
  - Possible relationships between agents and roles

    1 : 1

    1 : m

    n : 1

    n : m



Role Responsibilities

RASCI Matrix

Roles / Activities	Module developer	Moderator	Tester	Quality assurer
Module design	R			
Module coding	R			
Module review	S, R	S		A
Module testing			R	I

R = Responsible

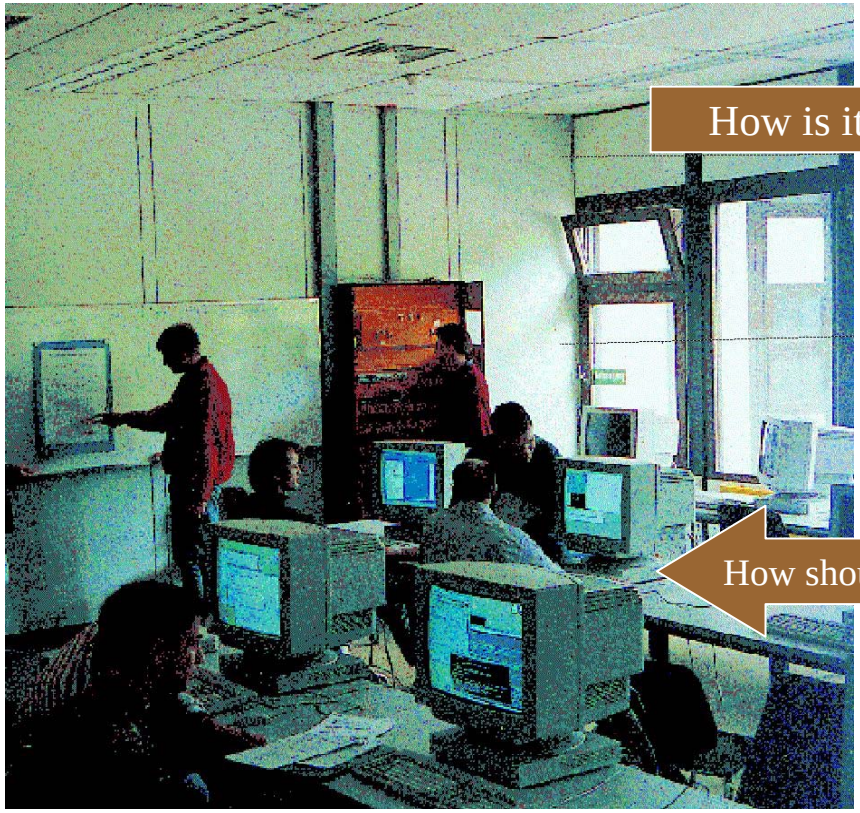
A = Approve

S = Support

C = Consult

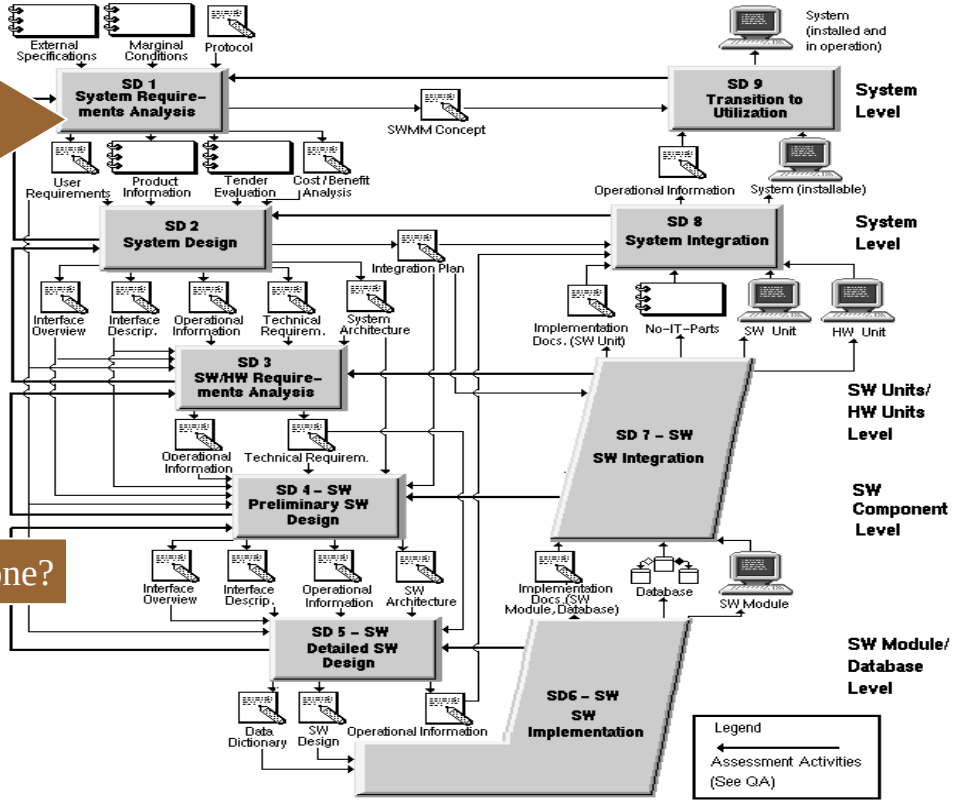
I = Inform

Descriptive vs. Prescriptive Process Models



How is it done?

How should it be done?



UNIVERSITETET I OSLO

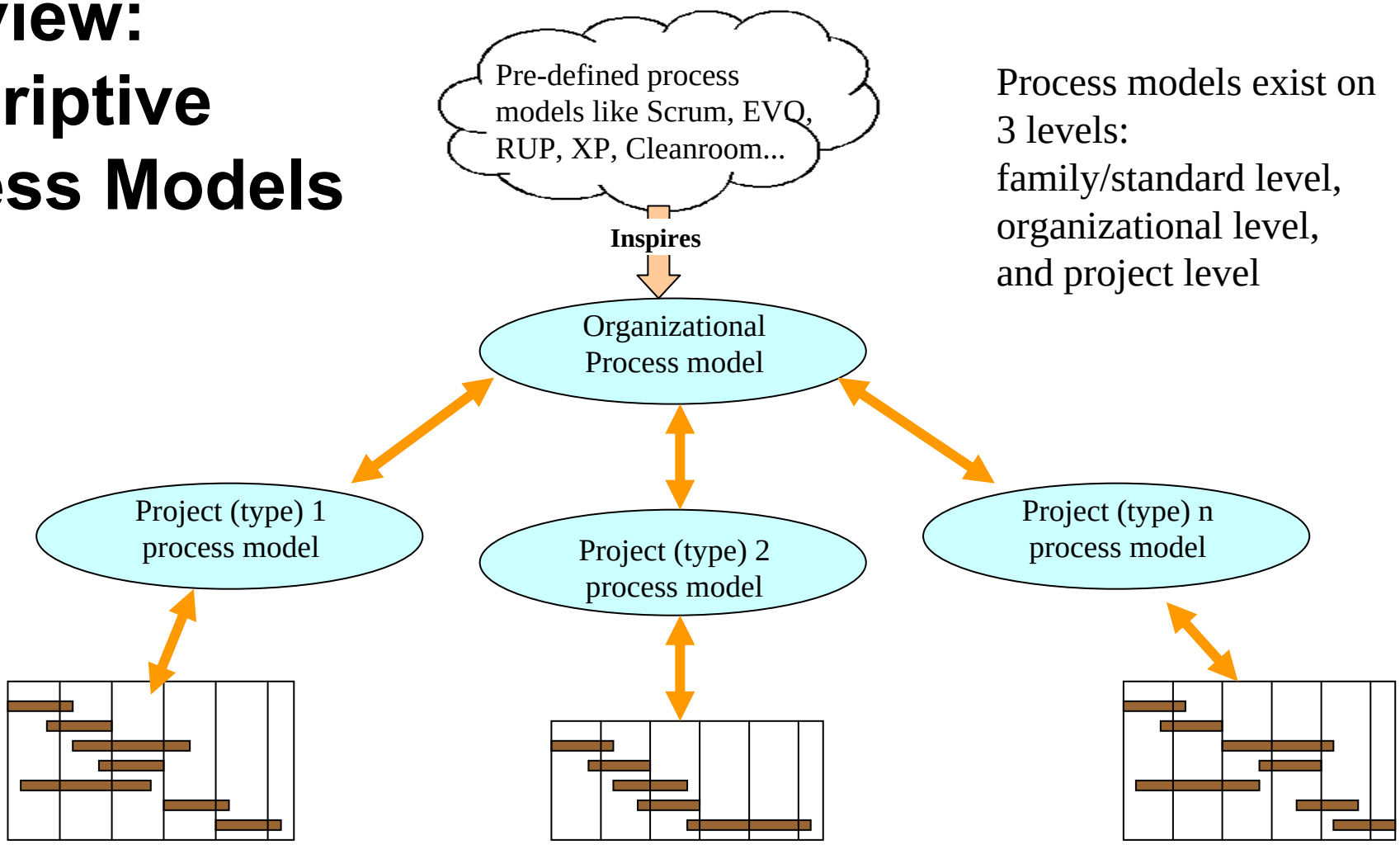
Prescriptive vs. Descriptive Process Models

- Prescriptive Models (theoretical)
  - "Ideal" Process
  - (Assumed) best practice
  - Often requires instantiation and detailing
  - Deviations from real processes are likely
  - Examples: waterfall, V-model, spiral model, incremental, iterative, evolutionary, agile process models
- Descriptive Models (empirical)
  - Accurate elicitation of actual, real processes
  - Basis for the revision of existing (prescriptive) process models based on observation and experience

Structure of Lecture 02

- Hour 1:
  - Introduction into Process Modelling
  - Prescriptive Process Models
- Hour 2:
  - Process Families/Standards
  - Descriptive Process Modelling
- Hour 3:
  - Exercises

Overview: Prescriptive Process Models



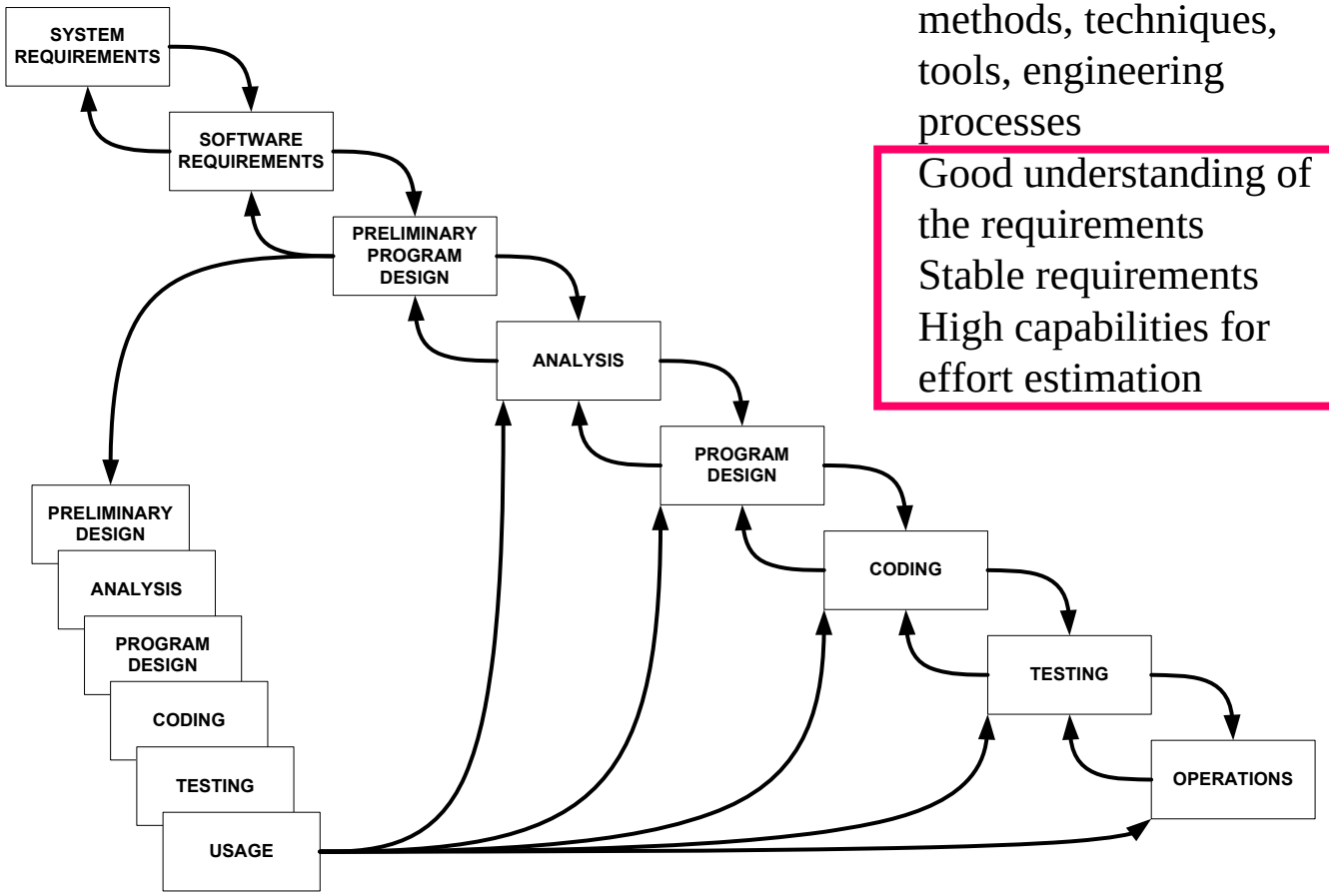
Process models exist on 3 levels: family/standard level, organizational level, and project level

Waterfall: Royce Model (1970)

Idea:

Sequential creation of products on different levels of abstraction (e.g., precede code by design, precede design by requirements) and integration in reverse direction

Strictly sequential control flow can be weakened by controlled iterations



Prerequisites:

Familiarity with application domains, methods, techniques, tools, engineering processes

Good understanding of the requirements
Stable requirements
High capabilities for effort estimation

Often Waterfall is Bad

- For many projects, the waterfall model is a poor choice
  - Late risk resolution
    - can't tell requirements or design risks exist until late in the life cycle
  - Requirements drive functional decomposition
    - exhaustive requirements make it hard to tell if the design is viable;
    - hard to identify critical requirements
  - Adversarial stakeholder relationships
    - written definitions of requirements often lead to extended (and heated) discussion of their interpretation
  - Focus on documents and reviews
    - fulfilling the letter of a contract can lead to the appearance of progress, but without real communication
  - Inflexible!

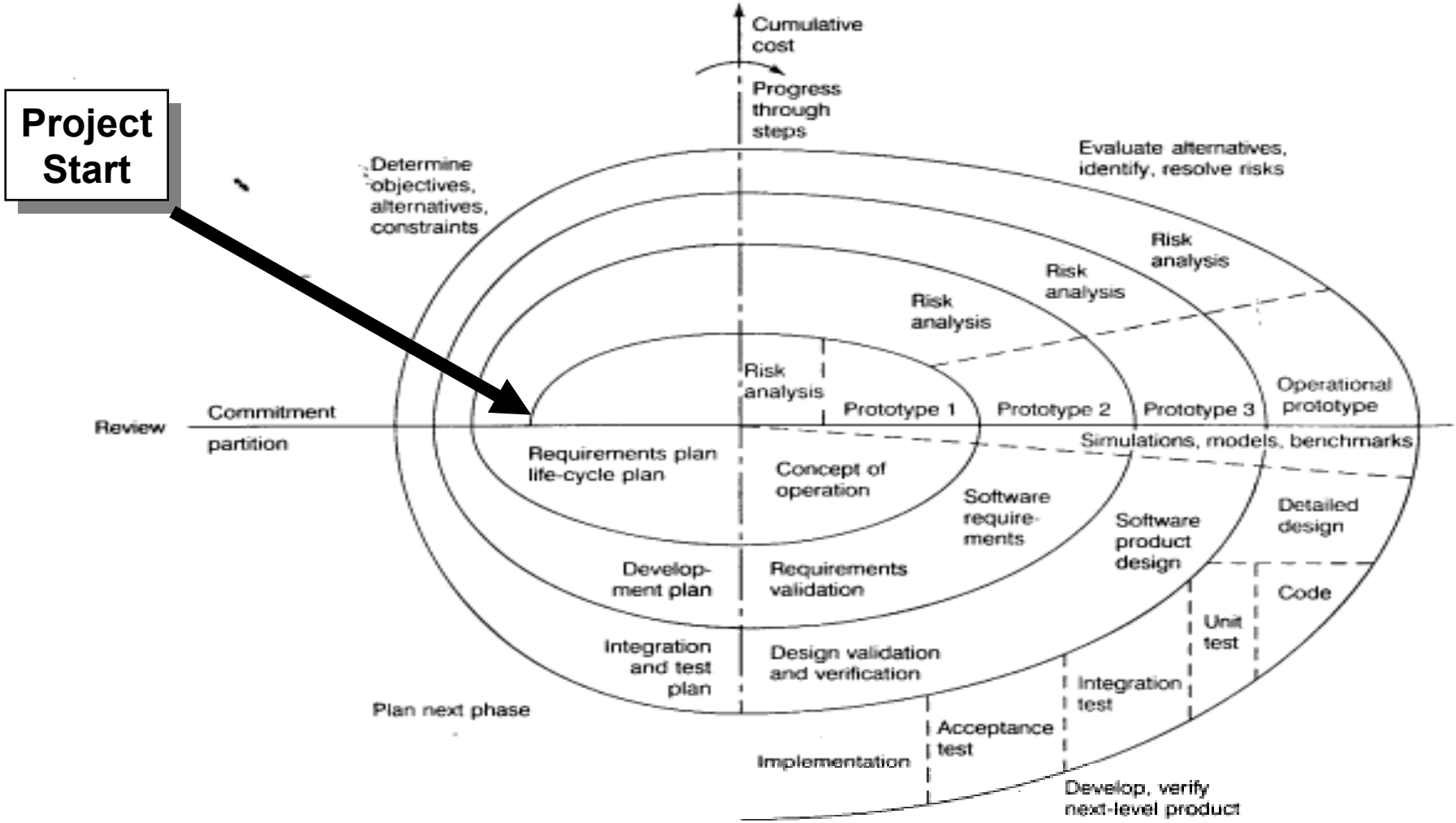
Prototyping

'*An iterative process of creating quickly and inexpensively live and working models to test out requirements and assumptions*'

(Sprague and McNurlin)

- Main types:
  - 'throw away' prototypes
  - evolutionary prototypes
- What is being prototyped?
  - human-computer interface
  - functionality

Boehm's Spiral Model



Spiral Model → Highly Iterative

- *The spiral model proposed by Boehm (1988) is an iterative model with focus on risk resolution:*
  - Determine objectives and constraints
  - Evaluate Alternatives
  - Identify risks
  - Resolve risks after assigning priorities to risks
  - Develop a series of prototypes for the identified risks starting with the highest risk
  - Use a waterfall model for each prototype development (“cycle”)
  - If a risk has successfully been resolved, evaluate the results of the “cycle” and plan the next round
  - If a certain risk cannot be resolved, terminate the project immediately

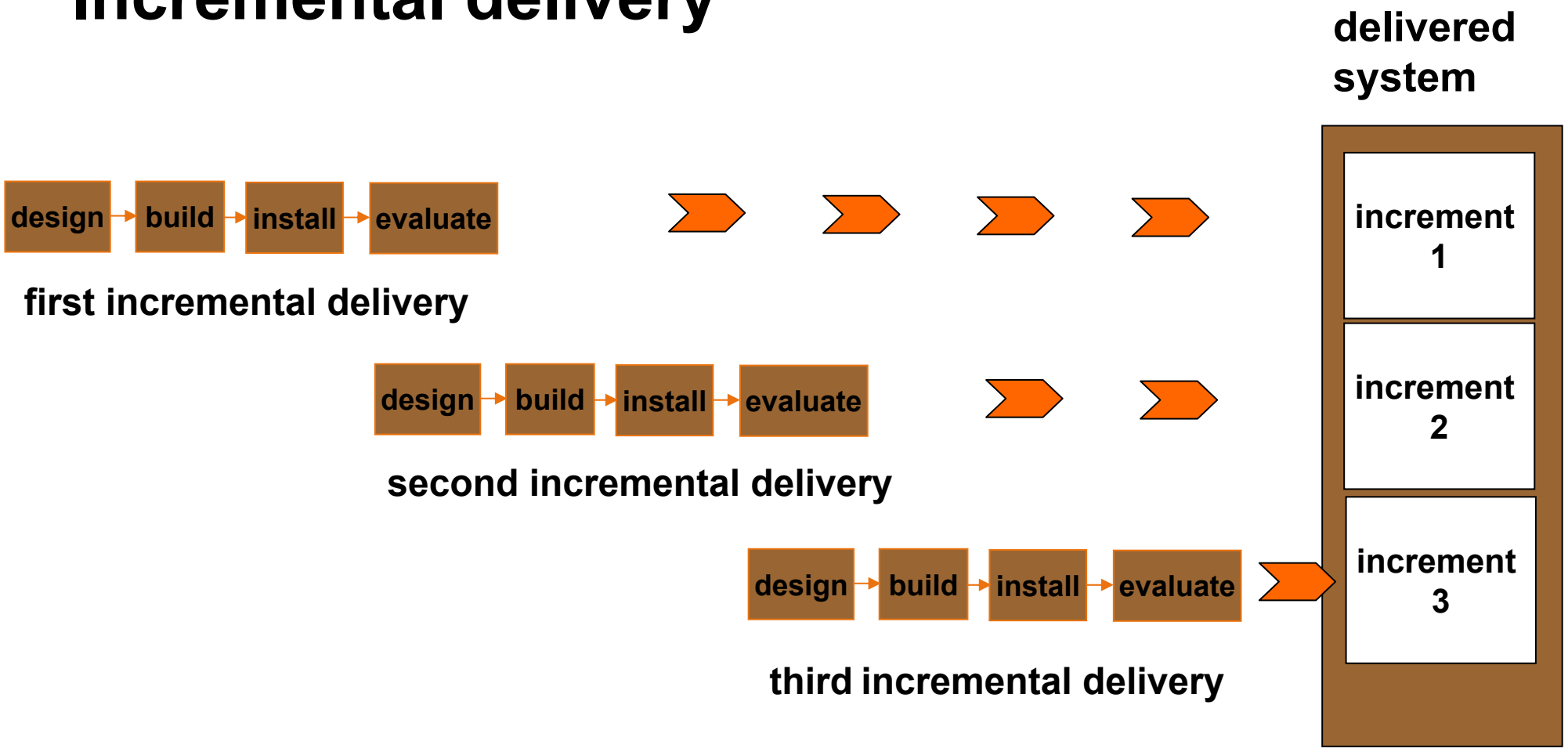
Types of Prototypes used in the Spiral Model

- **Illustrative Prototype**
  - Develop the user interface with a set of storyboards
  - Implement them on a napkin or with a user interface builder (Visual C++, ...)
  - Good for first dialog with client
- **Functional Prototype**
  - Implement and deliver an operational system with minimum functionality
  - Then add more functionality
  - Order identified by risk
- **Exploratory Prototype ("Hacking")**
  - Implement part of the system to learn more about the requirements.
  - Good for situations in which paradigm discontinuities occur

Iterative Enhancement (Incremental Delivery)

- **Origin:** **Basili und Turner, 1975**
- **Idea:**
  - Split functionality into several increments
  - Develop each increment (i.e., a product part that fulfills a subset of requirements) in a Waterfall style; integrate increment by increment into the product until delivery
  - The focus of the development of an increment might be completion of functionality or structure, but it can also be refinement and improvement
  - Strictly sequential control flow can be weakened by controlled iterations
- **Prerequisites:**
  - Structure of the problem permits incremental development

Incremental delivery



Iterative Enhancement (Incremental Delivery)

Advantages:

- Efficient learning during the project; thus, experience level can be low
- Early availability of a product, with the essential properties of the final product.
- Allows for early customer involvement and feedback
- Applicable when parts of requirements are unclear or unstable
- Supports integration testing
- Good applicability in case of fixed delivery dates (→ prioritize requirements with the customer)

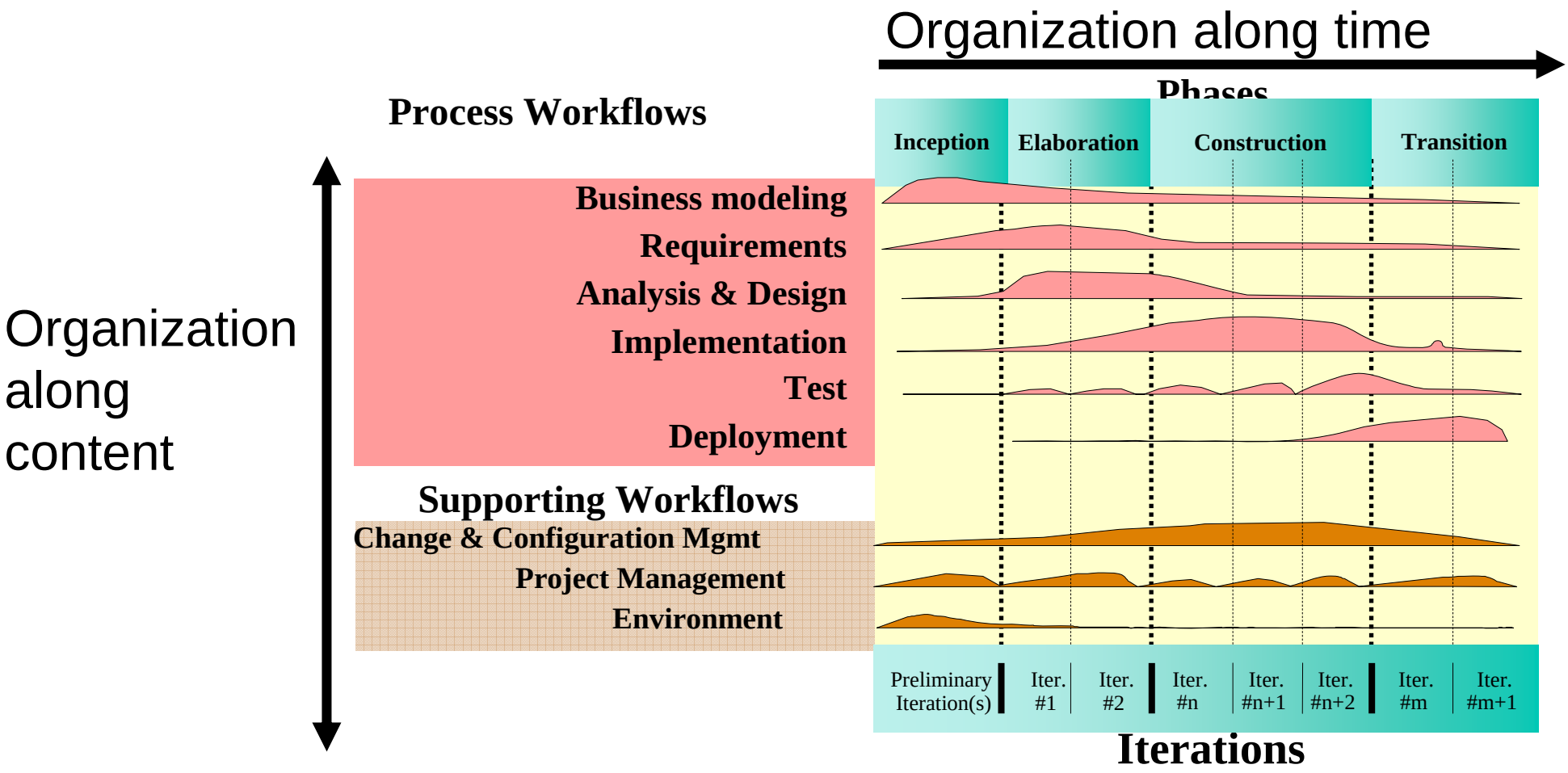
Disadvantages:

- Risk that, by ignoring specific requirements, the product will be designed in such a way that fulfilling future requirements becomes difficult/expensive
  - particularly problematic are non-functional requirements
- Comprehensive version and configuration management is necessary

Unified Process

- Family: Iterative Enhancement
- Origin:
  - **Ivar Jacobson, James Rumbaugh, Grady Booch, 1998**
- Defines process framework that is adaptable to
  - various application domains
  - different organizations
  - different competence levels
  - different project sizes
- Characteristics:
  - use case driven
  - architecture-centric
- **Provides only rudimentary instructions**
- **Refined version:**
  - **Rational Unified Process (Ph. Kruchten)**

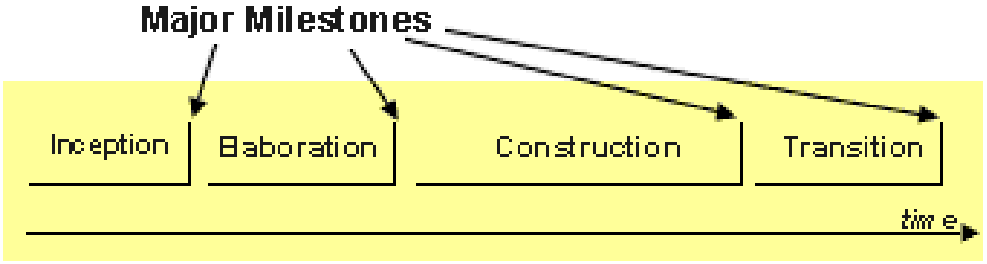
Rational Unified Process (RUP)



UNIVERSITETET I OSLO

RUP Phases and Iterations — The Time Dimension

- This is the dynamic organization of the process along time.
- The software lifecycle is broken into cycles, each cycle working on a new generation of the product. The Rational Unified Process divides one development cycle in four consecutive phases.
  - Inception phase
  - Elaboration phase
  - Construction phase
  - Transition phase



- Each phase is concluded with a well-defined *milestone*--a point in time at which certain critical decisions must be made, and therefore key goals must have been achieved.

RUP Phases – Example: Inception Phase

- During the inception phase: establish the business case for the system and delimit the project scope.
- To accomplish this you must identify all external entities with which the system will interact (actors) and define the nature of this interaction at a high-level.
- This involves identifying all use cases and describing a few significant ones. The business case includes success criteria, risk assessment, and estimate of the resources needed, and a phase plan showing dates of major milestones.
  - The outcome of the inception phase is:
  - A vision document: a general vision of the core project's requirements, key features, and main constraints.
  - An initial use-case model (10%-20% complete).
  - An initial project glossary (may optionally be partially expressed as a domain model).
  - An initial business case, which includes business context, success criteria (revenue projection, market recognition, and so on), and financial forecast.
  - An initial risk assessment.
  - A project plan, showing phases and iterations.
  - A business model, if necessary.
  - One or several prototypes.

At the end of the inception phase is the first major project milestone: the Lifecycle Objectives Milestone. The evaluation criteria for the inception phase are:

- Stakeholder concurrence on scope definition and cost/schedule estimates.

- Requirements understanding as evidenced by the fidelity of the primary use cases.

- Credibility of the cost/schedule estimates, priorities, risks, and development process.

- Depth and breadth of any architectural prototype that was developed.

- Actual expenditures versus planned expenditures.

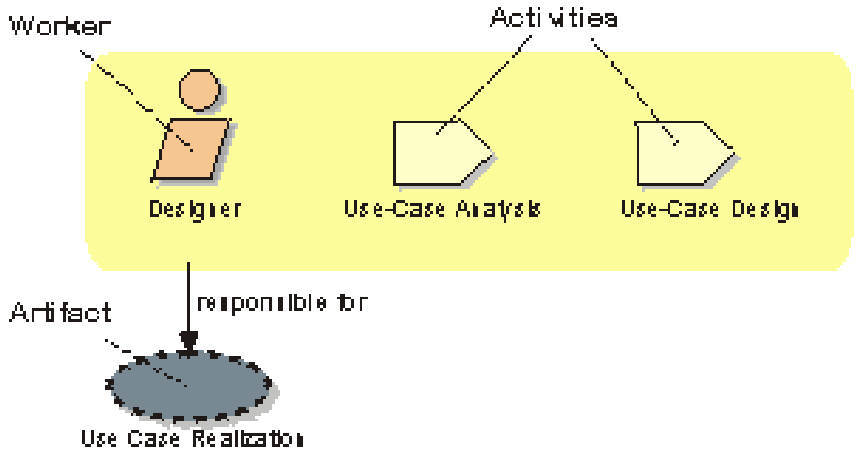
The project may be cancelled or considerably re-thought if it fails to pass this milestone.

RUP – Static Process

Static Structure of the Process

- A process describes who is doing what, how, and when.
- The RIP is represented using four primary modeling elements:
  - Workers (Roles), the "who"
  - Activities, the "how"
  - Artifacts, the "what"
  - Workflows, the "when"

Activities, Artifacts, and Workers



RUP – Activities and Artifacts

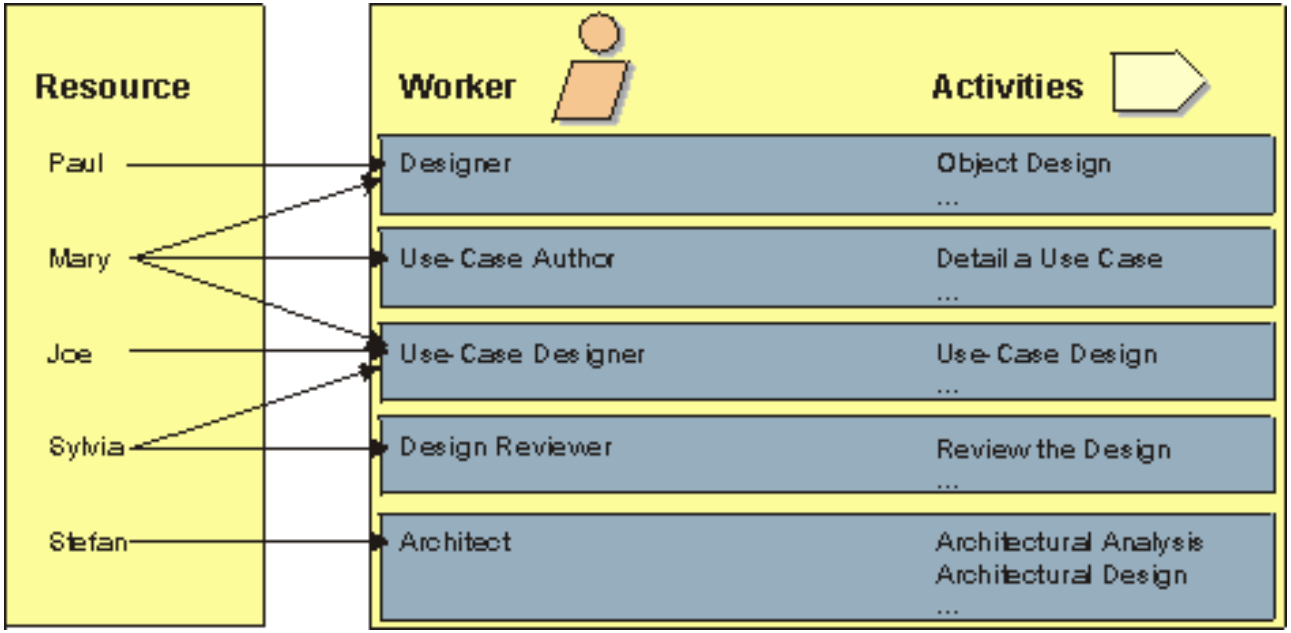
Activity

- An *activity* of a specific worker is a unit of work that an individual in that role may be asked to perform.
- The activity has a clear purpose, usually expressed in terms of creating or updating some artifacts, such as a model, a class, a plan.
- Every activity is assigned to a specific worker. The granularity of an activity is generally a few hours to a few days, it usually involves one worker, and affects one or only a small number of artifacts.
- An activity should be usable as an element of planning and progress; if it is too small, it will be neglected, and if it is too large, progress would have to be expressed in terms of an activity's parts.
- Example of activities:
  - Plan an iteration, for the Worker: Project Manager
  - Find use cases and actors, for the Worker: System Analyst
  - Review the design, for the Worker: Design Reviewer
  - Execute performance test, for the Worker: Performance Tester

Artifact

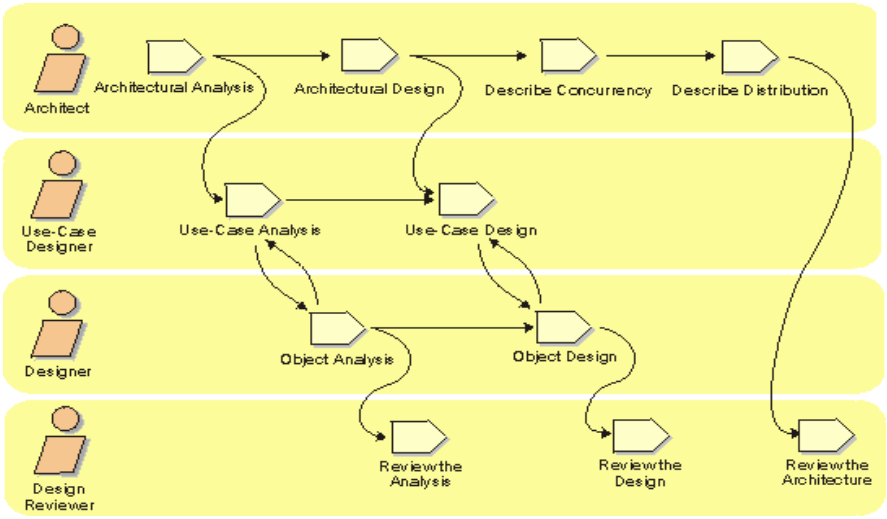
- An *artifact* is a piece of information that is produced, modified, or used by a process. Artifacts are the tangible products of the project, the things the project produces or uses while working towards the final product.
- Artifacts are used as input by workers to perform an activity, and are the result or output of such activities. In object-oriented design terms, as activities are operations on an active object (the worker), artifacts are the parameters of these activities.
- Artifacts may take various shapes or forms:
  - A model, such as the Use-Case Model or the Design Model
  - A model element, i.e. an element within a model, such as a class, a use case or a subsystem
  - A document, such as Business Case or Software Architecture Document
  - Source code
  - Executables

RUP – Resources and Workers (Roles)



- A *worker* defines the behavior and responsibilities of an individual, or a group of individuals working together as a team.
- You could regard a worker as a "hat" an individual can wear in the project.
- One individual may wear many different hats. This is an important distinction because it is natural to think of a worker as the individual or team itself, but in the Unified Process the worker is more the role defining how the individuals should carry out the work.
- The responsibilities we assign to a worker include both to perform a certain set of activities as well as being owner of a set of artifacts.

RUP Workflow – Example: Analysis & Design



Workflows

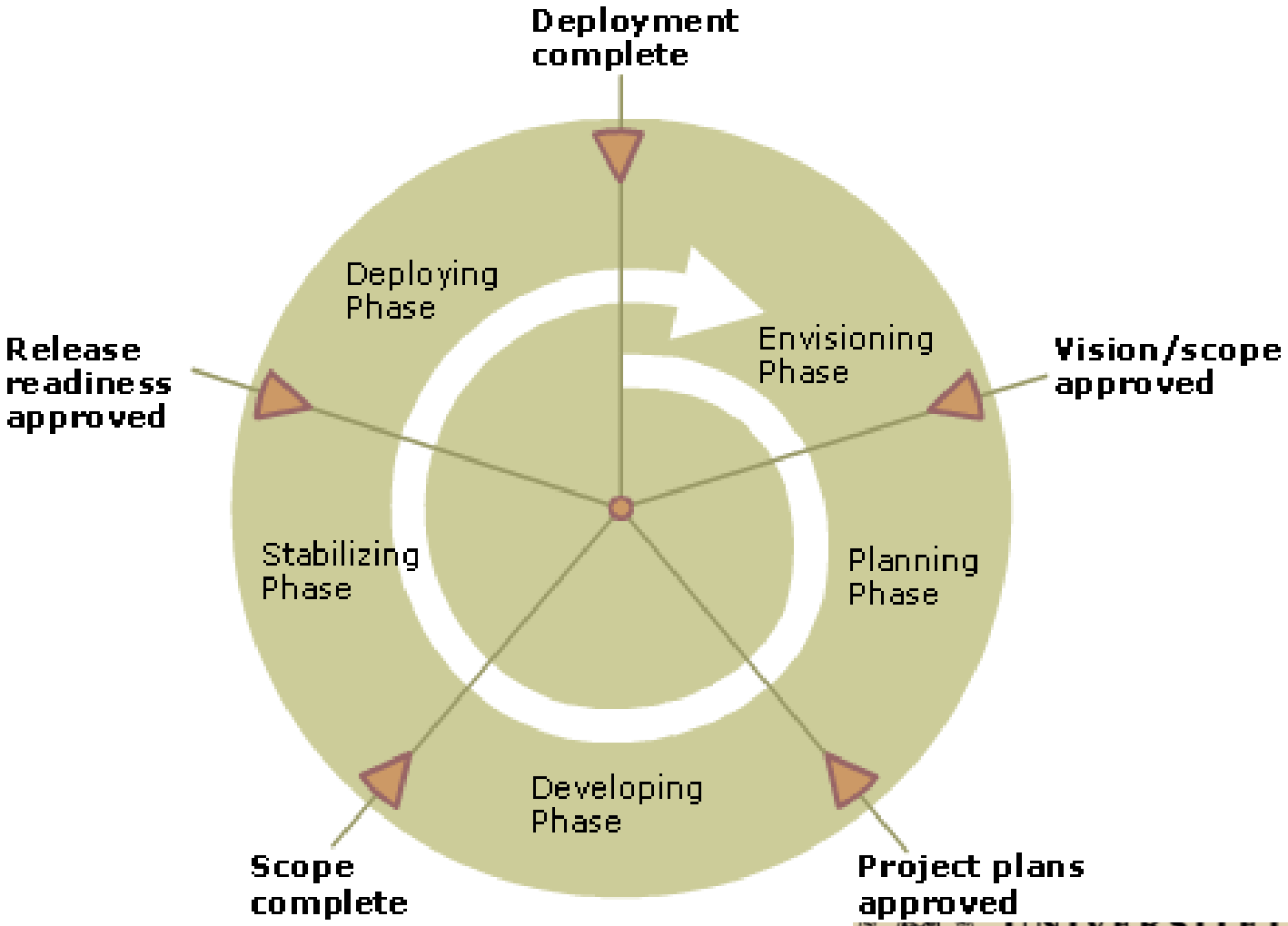
- A mere enumeration of all workers, activities and artifacts does not quite constitute a process. We need to describe meaningful sequences of activities that produce some valuable result, and to show interactions between workers.
- A *workflow* is a sequence of activities that produces a result of observable value.
- In UML terms, a workflow can be expressed as a sequence diagram, a collaboration diagram, or an activity diagram (cf. activity diagram on the left hand side).

RUP Workflow – Example: Analysis & Design

- The goal of the Analysis and Design workflow is to show how the system will be realized in the implementation phase. You want to build a system that:
  - Performs – in a specific implementation environment – the tasks and functions specified in the use-case descriptions.
  - Fulfills all its requirements.
  - Is structured to be robust (easy to change if and when its functional requirements change).
- Analysis and Design results in a design model and optionally an analysis model. The design model serves as an abstraction of the source code; that is, the design model acts as a 'blueprint' of how the source code is structured and written.
- The design model consists of design classes structured into design packages and design subsystems with well-defined interfaces, representing what will become components in the implementation. It also contains descriptions of how objects of these design classes collaborate to perform use cases.

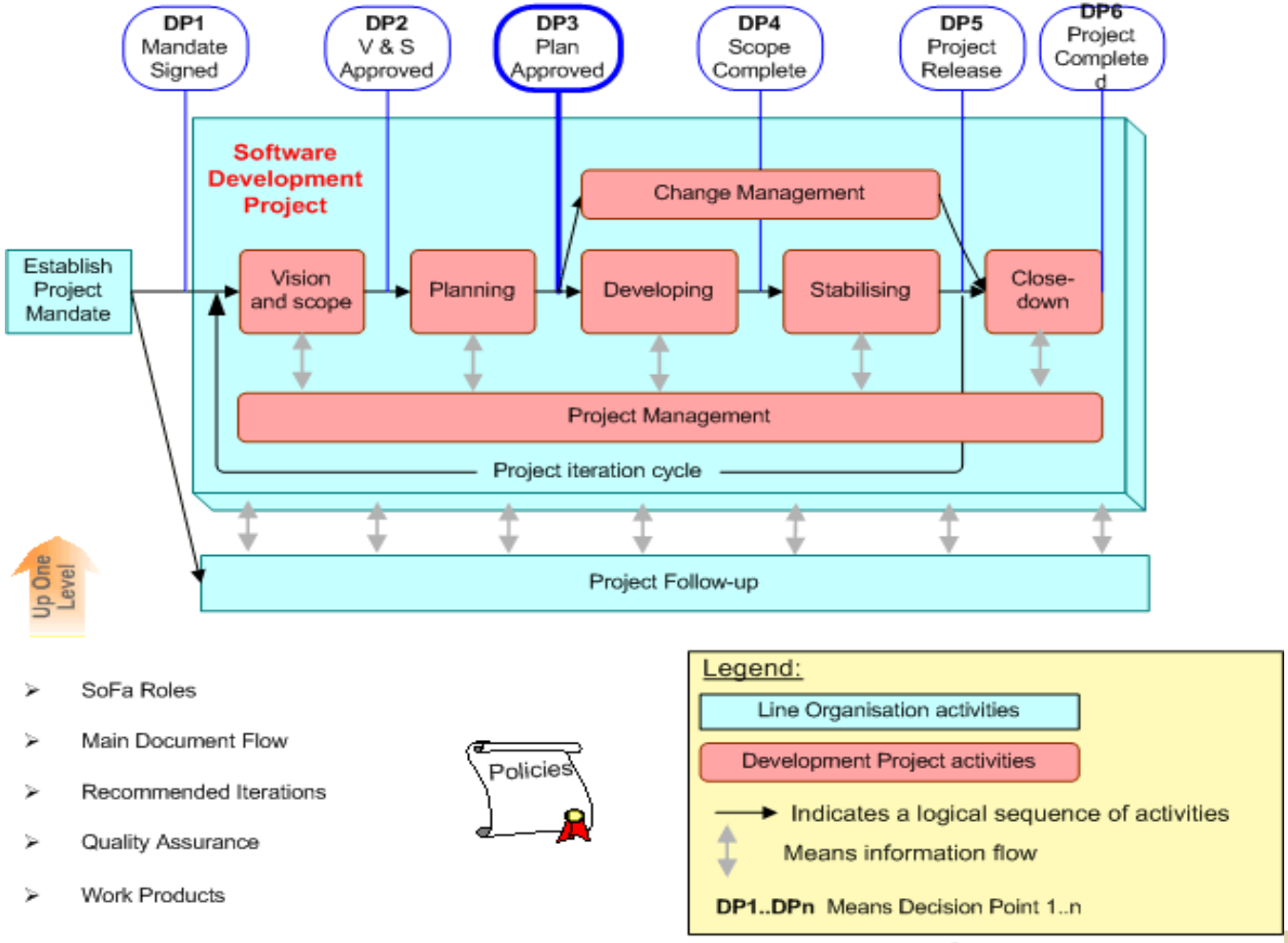
MSF (Microsoft Solution Framework)

For details refer to the related White Paper in the Reading Materials



MSF-Inspired Process Model (at DNV)

For sub-contractor management processes and agile development processes using the Norwegian PS2000 process standard refer to the related reports in the reading materials



Structure of Lecture 02

- Hour 1:
  - Introduction into Process Modelling
  - Prescriptive Process Models
- Hour 2:
  - Process Families/Standards
  - Descriptive Process Modelling
- Hour 3:
  - Exercises

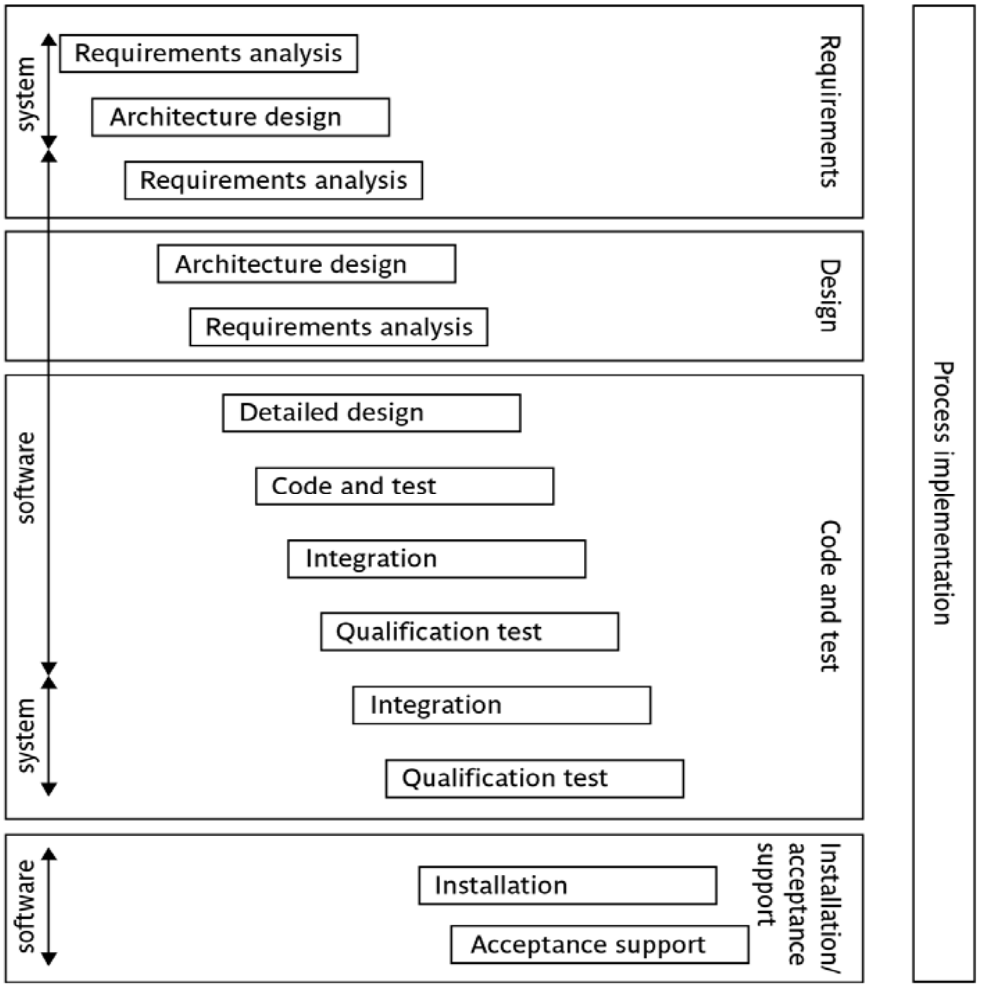
ISO 12207: Standard for Information Technology-Software Life Cycle Processes

- This standard officially replaced MIL-STD-498 for the development of DoD software systems in August 1998
- This standard defines a comprehensive set of processes that cover the entire life-cycle of a software system – from the time a concept is made to the retirement of the software
- The standard defines a set of processes, which are in turn defined in terms of activities. The activities are broken down into a set of tasks.
- The processes are defined in three broad categories:
  - Primary Life Cycle Processes
  - Supporting Life Cycle Processes
  - Organisational Life Cycle Processes

ISO 12207 Processes

- **Primary life cycle processes:**
  - Acquisition process
  - Supply process
  - Development process
  - Operation process
  - Maintenance process
- **Supporting life cycle processes:**
  - Audit process
  - Configuration Management
  - Joint review process
  - Documentation process
  - Quality assurance process
  - Problem solving process
  - Verification process
  - Validation process
- **Organisational processes:**
  - Management process
  - Infrastructure process
  - Improvement process
  - Training process

The software development life-cycle (ISO 12207)



UNIVERSITETET I OSLO

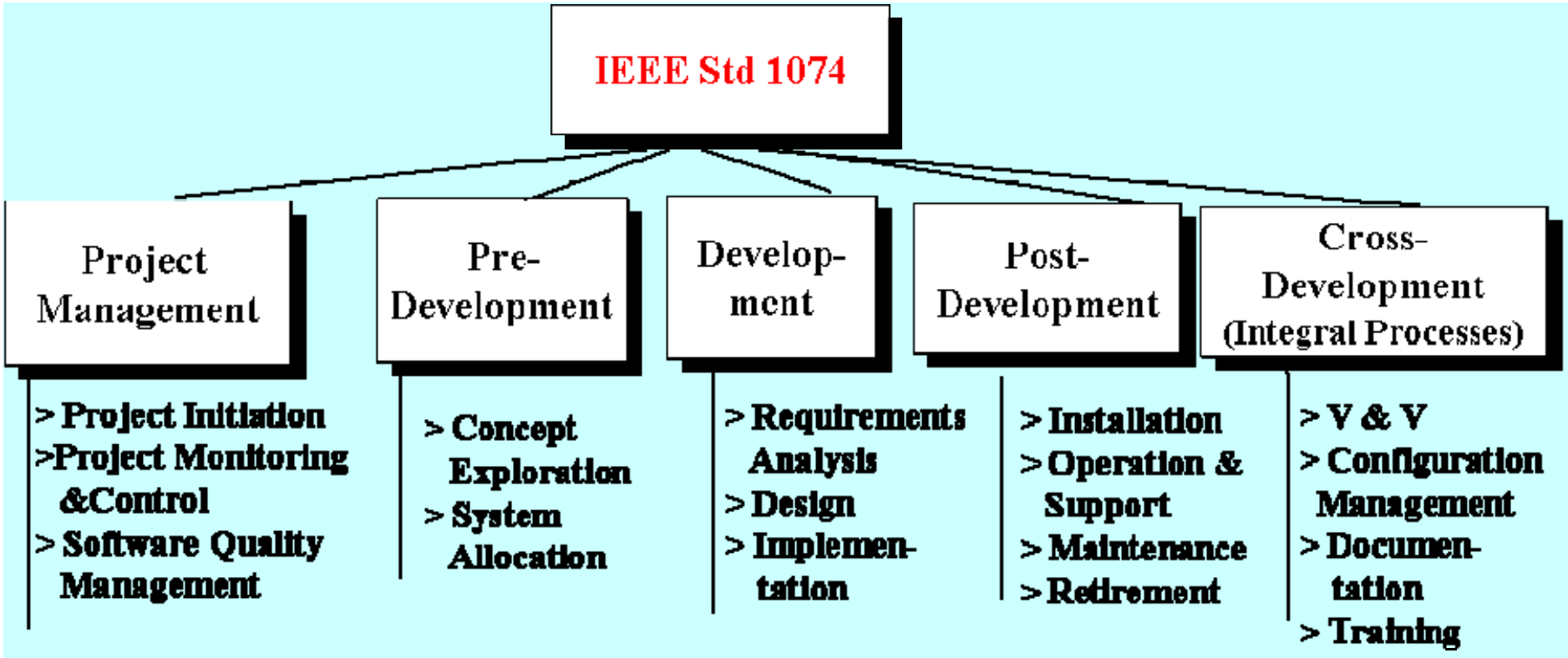
DOD Standard 2167A

- Required by the Department of Defense for all software contractors in the 1980-90s
- Waterfall-based model with the software development activities:
  - System Requirements Analysis/Design
  - Software Requirements Analysis
  - Preliminary Design and Detailed Design
  - Coding and CSU testing (CSU = Computer Software Unit)
  - CSC Integration and Testing (CSC = Computer Software Component, can be decomposed into CSC's and CSU's)
  - CSCI Testing (CSCI = Computer Software Configuration Item)
  - System integration and Testing

IEEE Std 1074

- Institutional standard (‘least common denominator’) published in 1997
- Process description comparable with V-Modell® XT (on a high level), but no statements about products, roles
- Offers only little guidance for developers

IEEE Std 1074: Standard for Software Lifecycle



V-Modell® XT (XT = Extreme Tailoring)

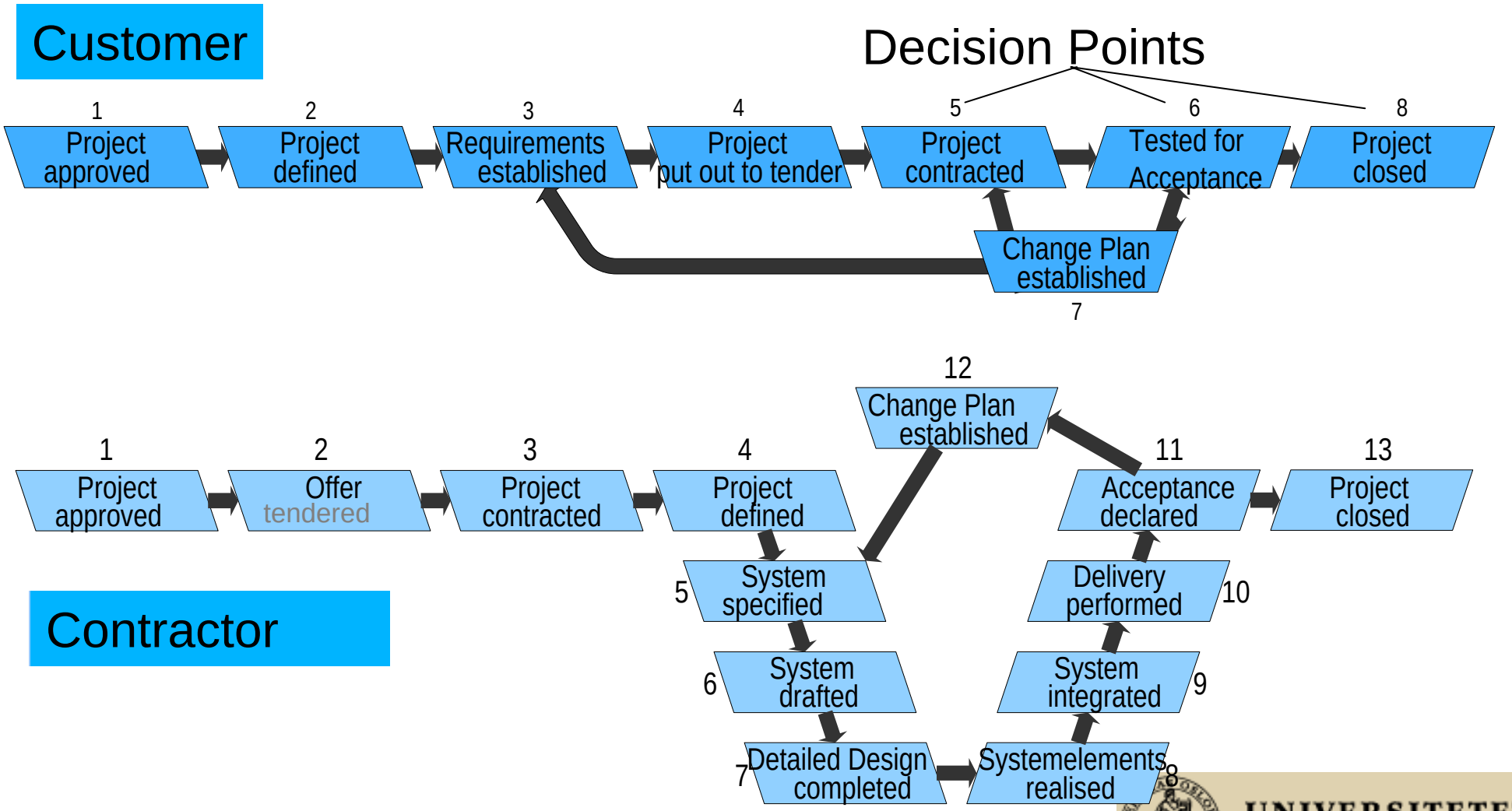
- Published in January 2005
- Predecessor: V-Model (1997) for military authorities in Germany
- Structured in a modular way
- Mandatory for IT projects in public and military domains in Germany
- Goals:
  - Enhance support for adaptability, scaleability, changeability, and expandability of V-Model 97
  - Consider state of the art and adapt to current regulations and standards
  - Expand application range considering the complete system lifecycle of development projects
  - Introduce a process of organizational process improvement

Somewhat Comparable to the role of PS 2000 in Norway

V-Model XT – Purpose and Scope

- The V-Model XT is a guideline for the Planning and Management of IT Development Projects.
- Scope of the V-Model are:
  - Improvement of Planning and Tracking of IT Development Projects,
  - Minimization of Project Risks,
  - Improvement and Quality Assurance,
  - Improvement of Communication between Project Stakeholders,
  - Containment of Total Costs over the Project and System Life Cycle.
- The V-Model supports different Project Execution Strategies and the Concept of Decision Points.
- The V-Model can be tailored according to the specific conditions and needs of an ICT Project
- The V-Model addresses the Customer and the Contractor.

Customer vs. Contractor View

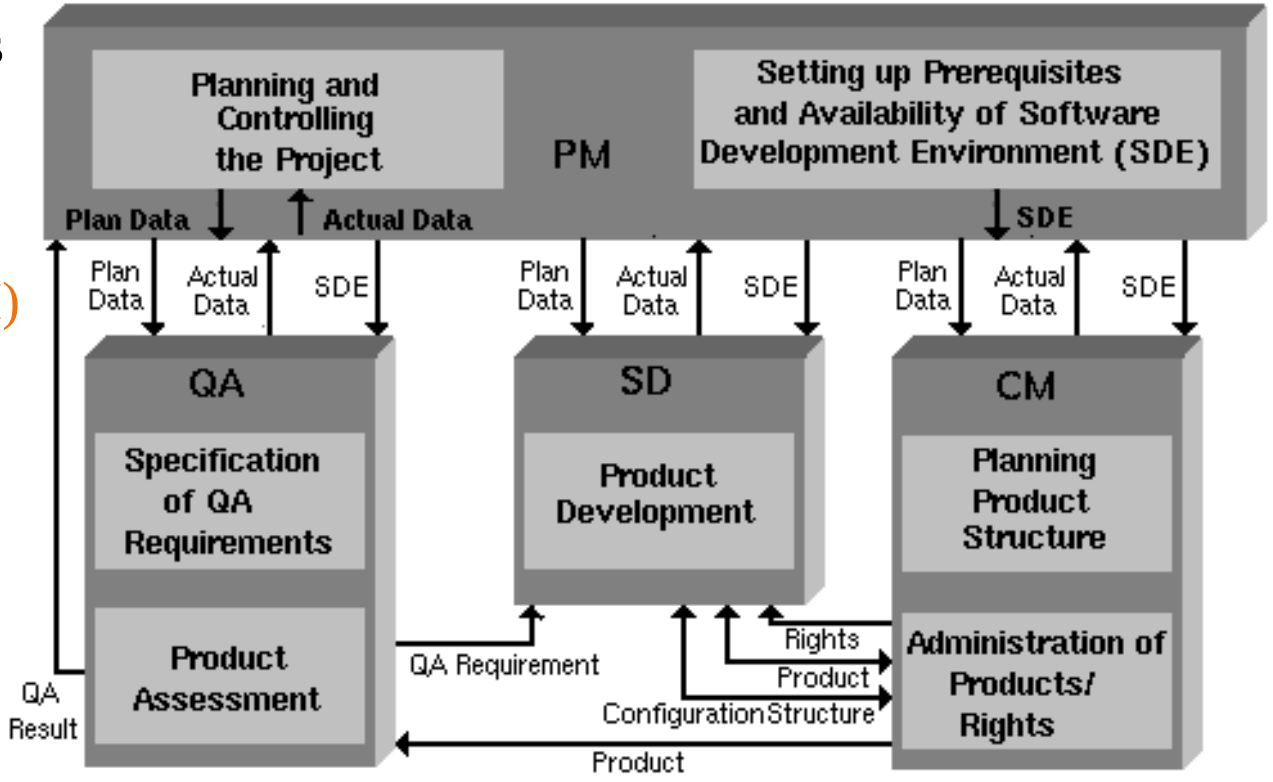


UNIVERSITETET I OSLO

German V-Model: The Big Picture

The German V-Model comprises four sub-models:

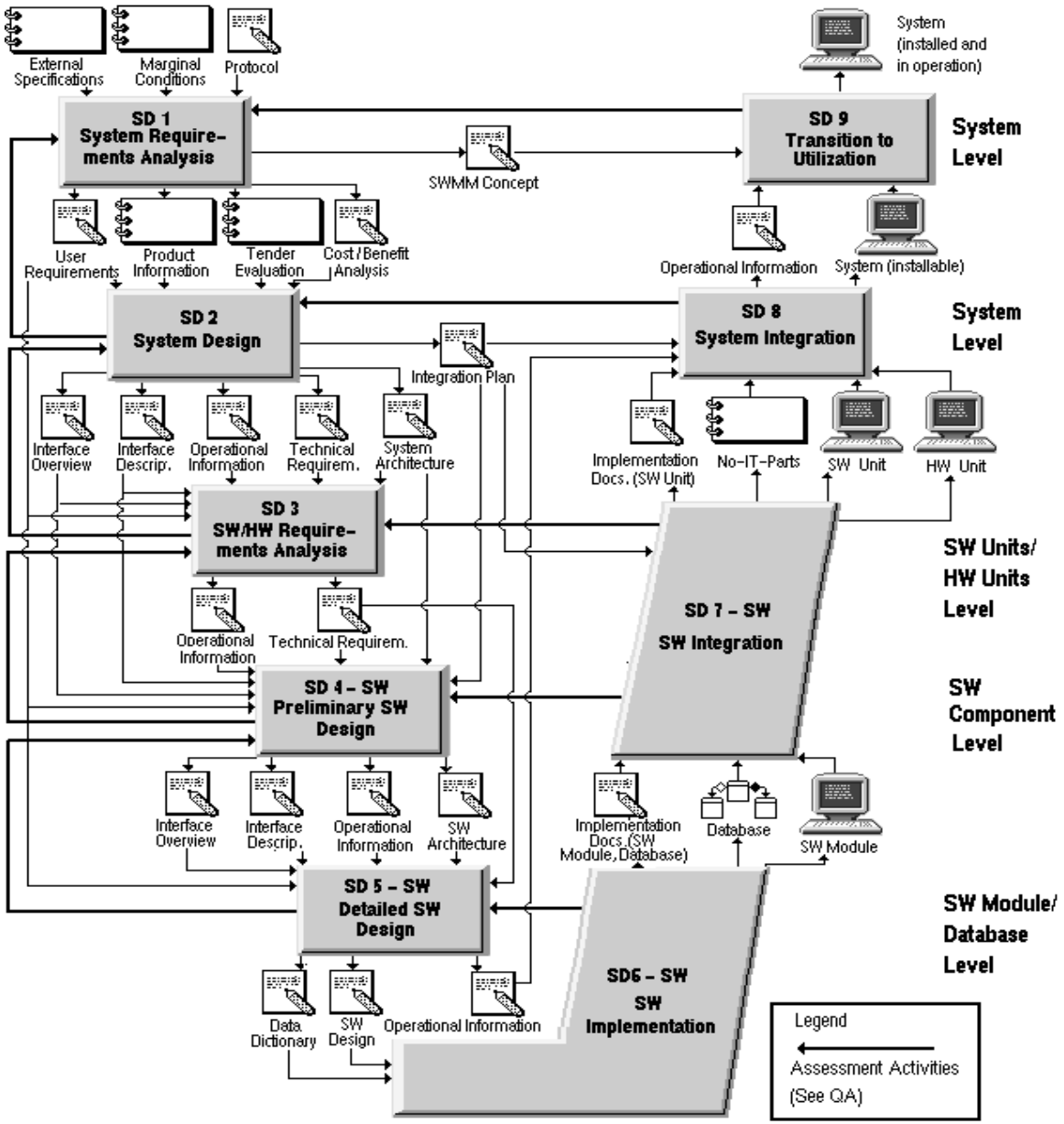
- System Development (SD)
- Quality Assurance (QA)
- Configuration Management (CM)
- Project Management (PM)



UNIVERSITETET I OSLO

German V-Model

System Development (SD) Sub-Model

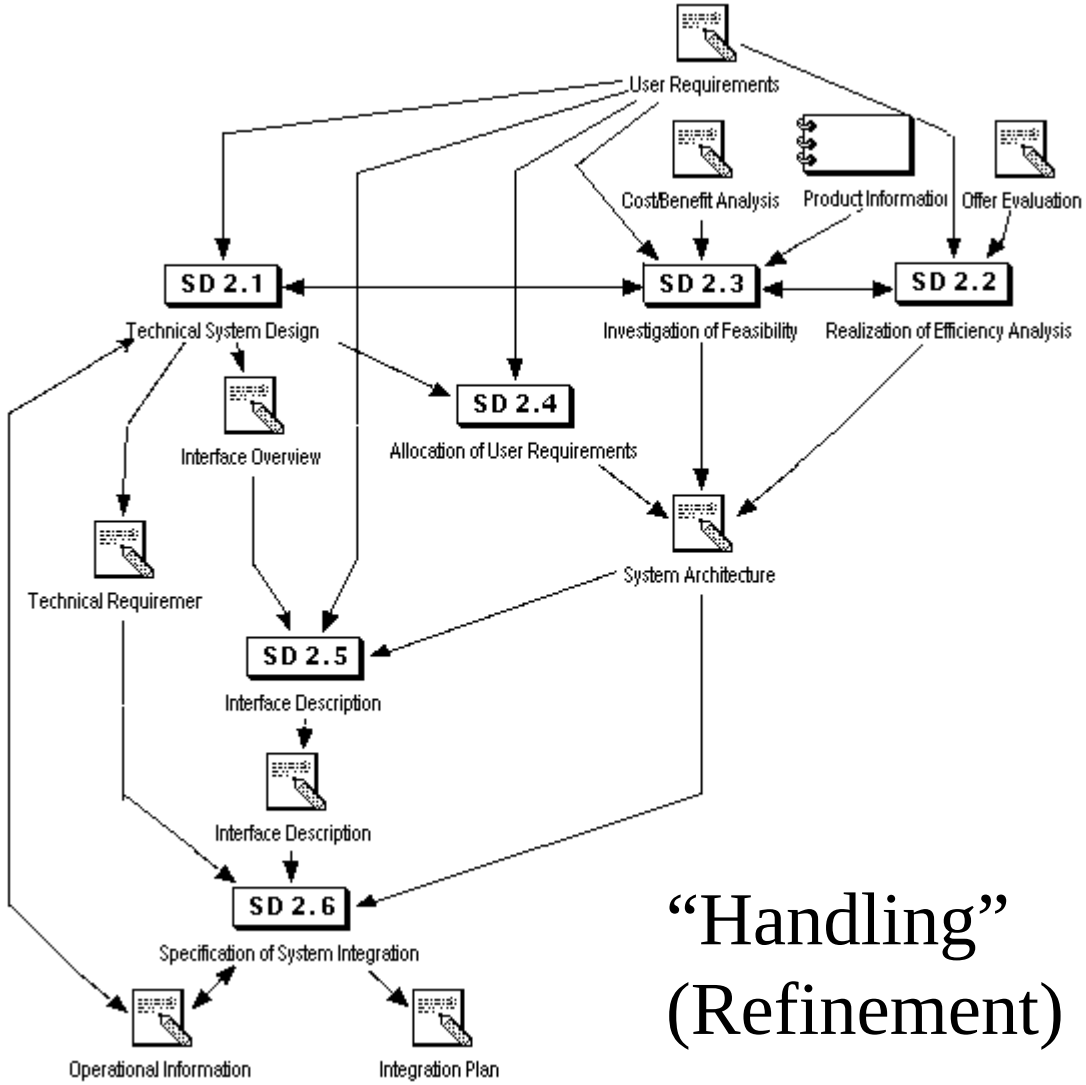


UNIVERSITETET I OSLO

German V-Model:

System Development (SD) Sub-Model

SD2:
System
Design



"Handling"
(Refinement)

V-Model: System Development (SD) Sub-Model

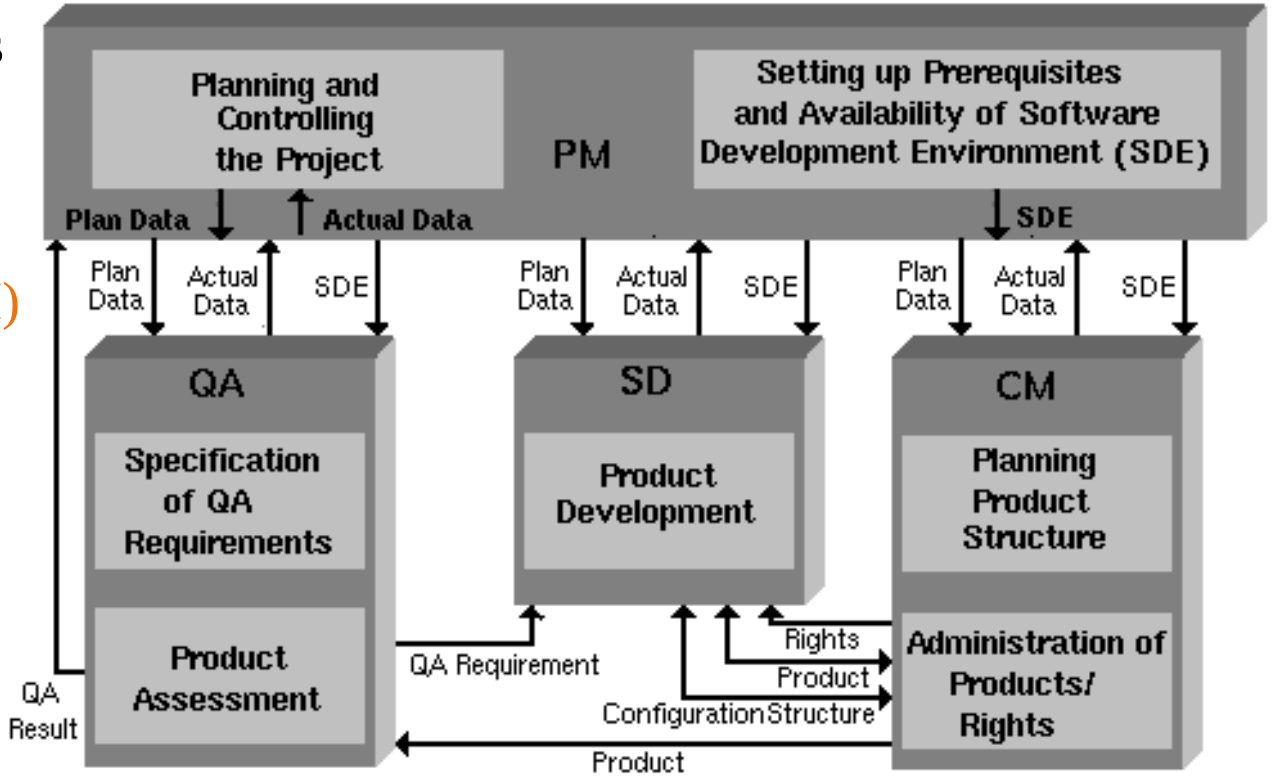
SD2: System Design

From		Product		to		Methods	Tool Req.	Ext. Norms
Activity	State	Chapter	Title	Activity	State			
External	-	All	Product Information	-	-			
PM2	accepted	All	Offer Evaluation (1)	-	-			
PM5	accepted	All	Cost/Benefit Analysis	-	-			
SD1	accepted	Existing	User Requirements	-	-			
-	-	Existing	System Architecture	SD1 (2), SD3, SD4-SW, PM4, PM5	submitted	BA, COM, CRC, ER, FCTD, IAM, PRODIAG, SIMU, SSM		
-	-	Existing	Technical Requirements	SD3	being proc.	COM, CRC, PRODIAG, SSM		
-	-	Existing	Operational Information: User Manual, Diagnosis Manual, Operator Manual, Other Application Information	SD3	being proc.			
-	-	Existing	Interface Overview	SD4-SW, CM4	being proc.	COM, DFM, SSM		
-	-	Existing	Interface Description	SD3, SD4-SW, CM4	being proc.	ACC, COM, DVER, FS, IAM, SSM, STMO		
-	-	Existing	Integration Plan	SD4-SW, QA3	being proc.	BAR, DTAB, NPT/		

German V-Model: The Big Picture

The German V-Model comprises four sub-models:

- System Development (SD)
- Quality Assurance (QA)
- Configuration Management (CM)
- Project Management (PM)

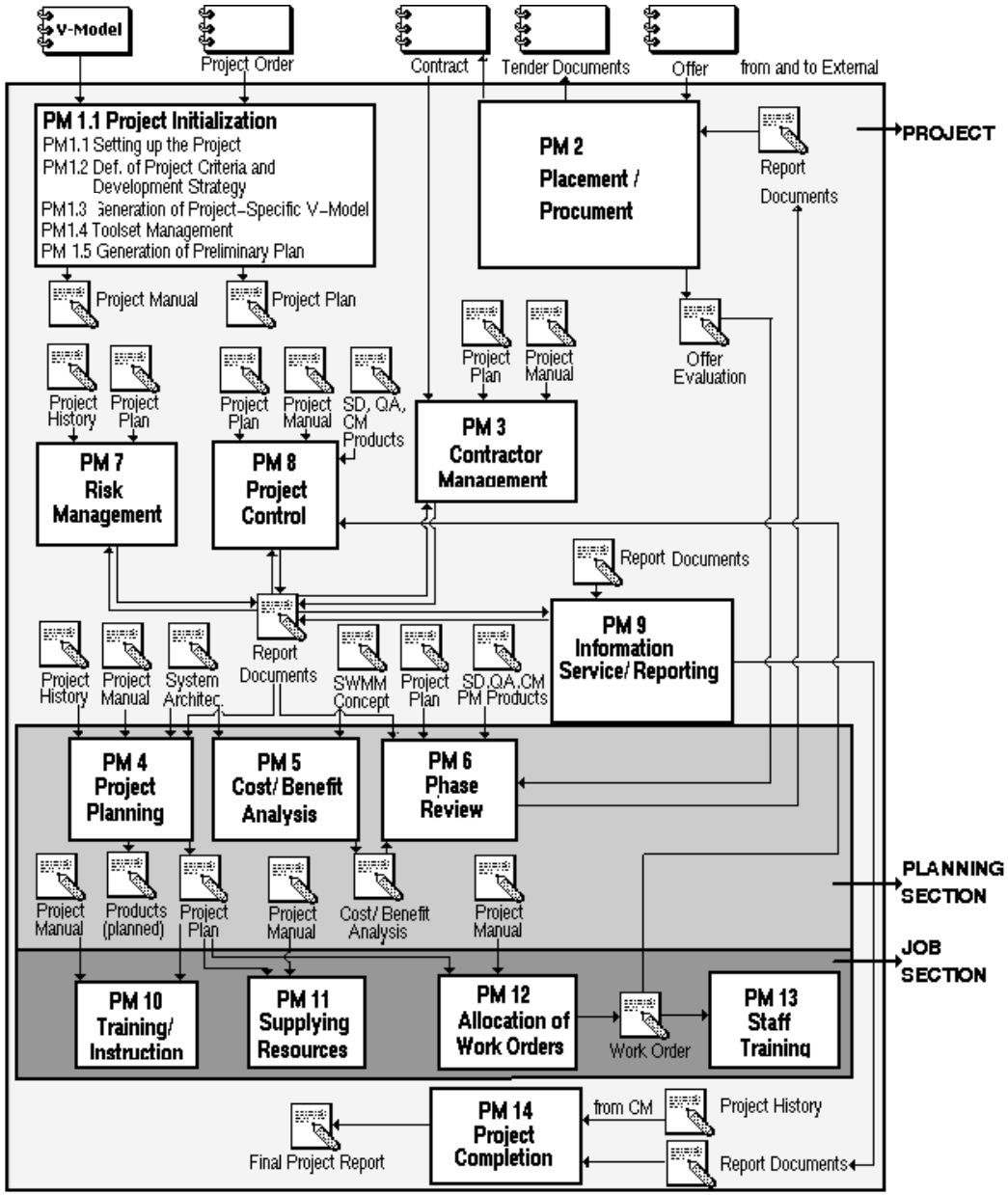


UNIVERSITETET I OSLO

V-Model: Project Management (PM) Sub-Model

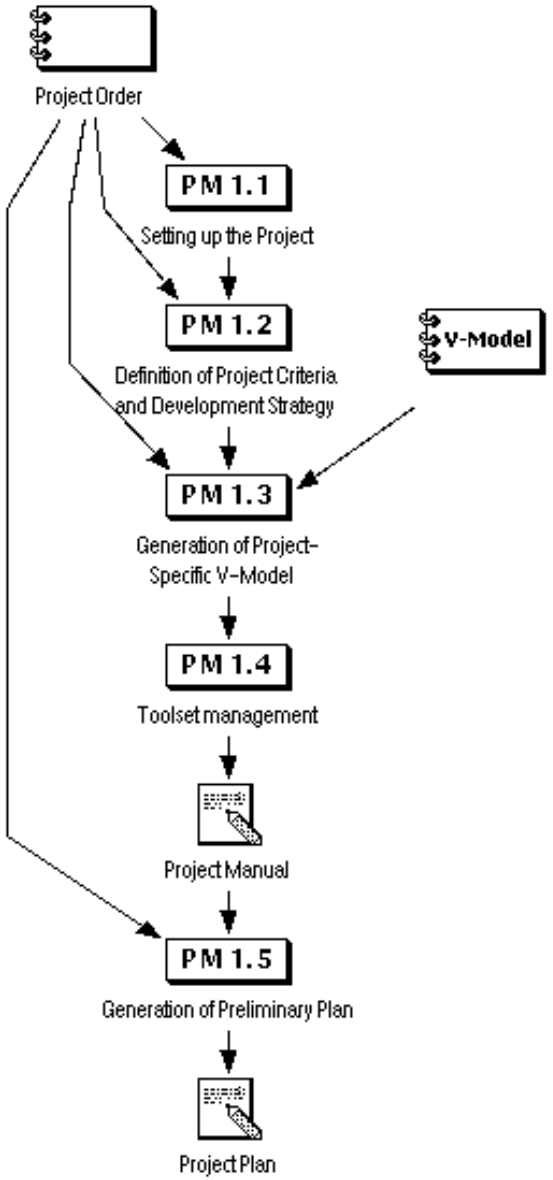
Activity Types:

- Management-related
  - Initialization/Finalization
  - Periodically Required
- Placement/Procurement-related
- Planning-related
- Resource-related



V-Model: Project Management (PM) Sub-Model

PM1:
Project
Initialization



"Handling"
(Refinement)

Structure of Lecture 02

- Hour 1:
  - Introduction into Process Modelling
  - Prescriptive Process Models
- Hour 2:
  - Process Families/Standards
  - Descriptive Process Modelling
- Hour 3:
  - Exercises

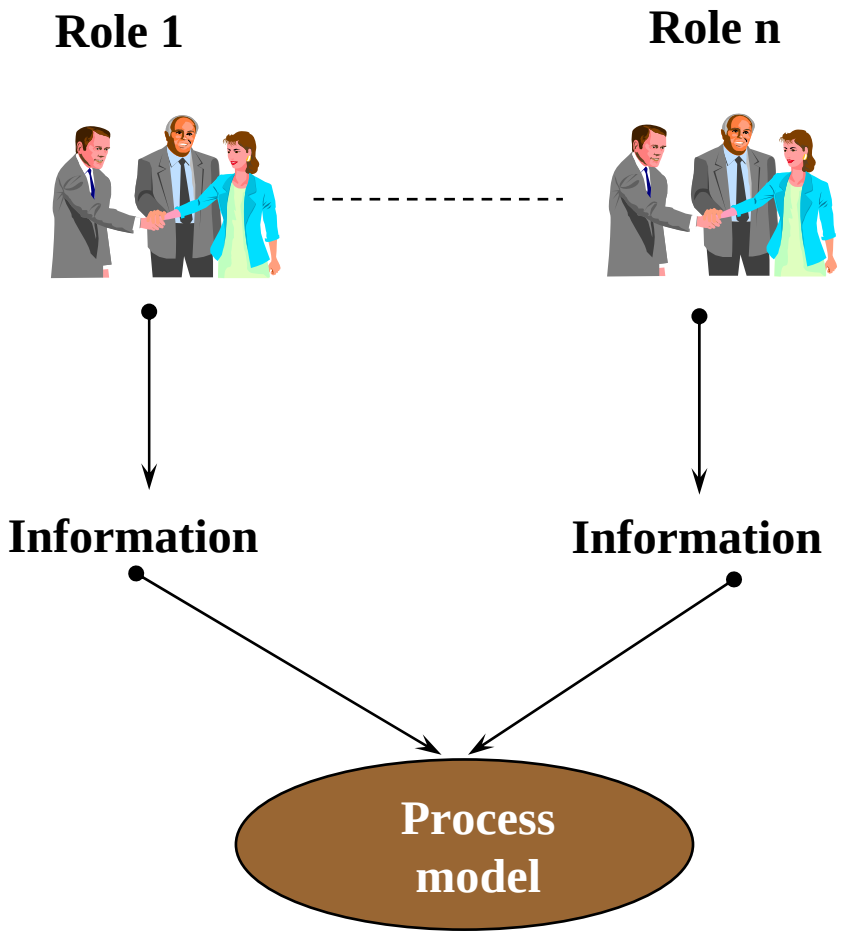
Goals of Descriptive Process Modeling

- Understand the process
  - Explicit documentation
  - Analyses (consistency, completeness, complexity)
- Communicate (about) the process
  - Find agreement in case of conflicting opinions
  - Propagation of ‘Best Practices’
- Support measurement
  - Describe, who can measure what and when
  - Collect quantitative information about processes, products and resources
- Manage the process (and products)
  - Define goals (target values) and control the adherence to these goals.

Steps of Descriptive Process Modeling

1. Formulate goals and scope of the task
2. Choose a conceptual schema (meta-model)
3. Choose a process modeling language / notation
4. Select or adapt tools
5. **“Elicitation”**
6. Create process model
7. Analyze process model
8. Analyze process

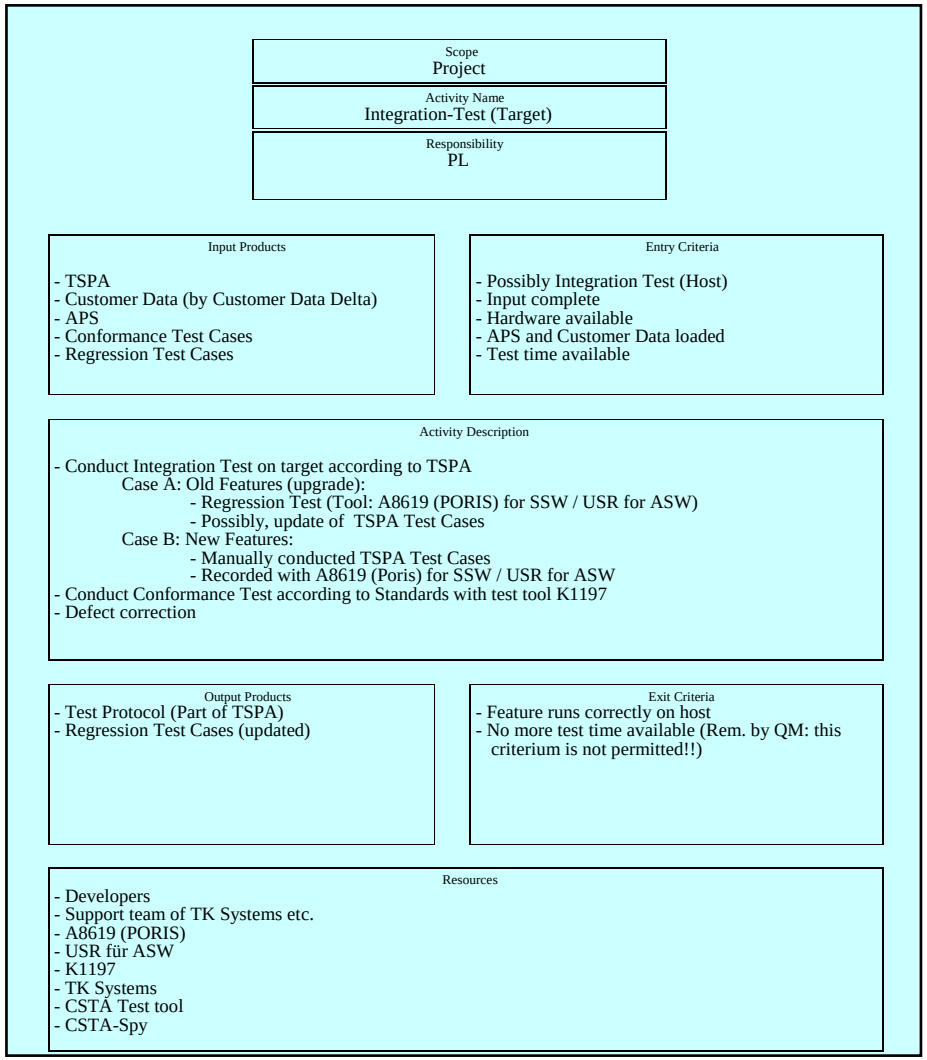
Process Elicitation



How to do it?

- Structured interview, 1-2 Hours
- 2 interviewers
- Separated by roles
  - no large groups
  - clear focus
  - manageable process models
  - no mutual interaction (horizontal and vertical hierarchic relations)
- Perform interviews one after another, however not more than 3 interviews per day

Example Form for Structured Interview



Scope
Project

Activity Name
Integration-Test (Target)

Responsibility
PL

Input Products

- TSPA
- Customer Data (by Customer Data Delta)
- APS
- Conformance Test Cases
- Regression Test Cases

Entry Criteria

- Possibly Integration Test (Host)
- Input complete
- Hardware available
- APS and Customer Data loaded
- Test time available

Activity Description

- Conduct Integration Test on target according to TSPA
  - Case A: Old Features (upgrade):
    - Regression Test (Tool: A8619 (PORIS) for SSW / USR for ASW)
    - Possibly, update of TSPA Test Cases
  - Case B: New Features:
    - Manually conducted TSPA Test Cases
    - Recorded with A8619 (Poris) for SSW / USR for ASW
- Conduct Conformance Test according to Standards with test tool K1197
- Defect correction

Output Products

- Test Protocol (Part of TSPA)
- Regression Test Cases (updated)

Exit Criteria

- Feature runs correctly on host
- No more test time available (Rem. by QM: this criterium is not permitted!!)

Resources

- Developers
- Support team of TK Systems etc.
- A8619 (PORIS)
- USR für ASW
- K1197
- TK Systems
- CSTA Test tool
- CSTA-Spy

Project, process name, role

Input products and entry conditions

Description of the process/activity

Output products and exit conditions

Resources

Rules for Process Elicitation (1/3)

- Obtain information about
  - the organization
  - the software domain
- Analyze existing documents and products
- Observe the relation between developers and quality assurance
- Ask whether an ongoing or upcoming organizational restructuring impacts the process
- Make sure that the interview partner is selected according to your instructions / guidelines
- Begin the interviews with a quality manager or project manager

Rules for Process Elicitation (2/3)

- Opening of Interview
  - Summary
  - Explain goal and purpose
  - Stress confidentiality
  - General questions about the process, and existence of variants
- Main part of Interview
  - Behave neutral
  - At first ask about the products
  - Then ask about processes
  - What are typical (known) deviations from the prescribed processes?
  - Which other roles participate in the processes? (Cross-Check)
  - Always be precise
  - Try to identify process variants

Rules for Process Elicitation (3/3)

- Closing of Interview
  - Explain future steps
  - Agree on time for the review
  - Thank your interview partner

- Ask questions even when a noticed ambiguity seems to be small, often big problems are hidden behind it
- Don't try to solve all ambiguities and conflicts (during the interview) – but follow-up on observed inconsistencies afterwards
- After the interview: give a quick feedback to the interview-partner about what you did with his/her information

Example: Process Analysis

- The number of products is higher (approx. twice as high) than the number of processes.
- The complexity of product flow interfaces of processes is relatively high (most of the processes access more then a dozen of products).
- Most of processes are undertaken by several roles (partly over five roles).
- Most of roles are involved in execution of more then a third of the whole process.

30 Processes
66 Products
42 Resources

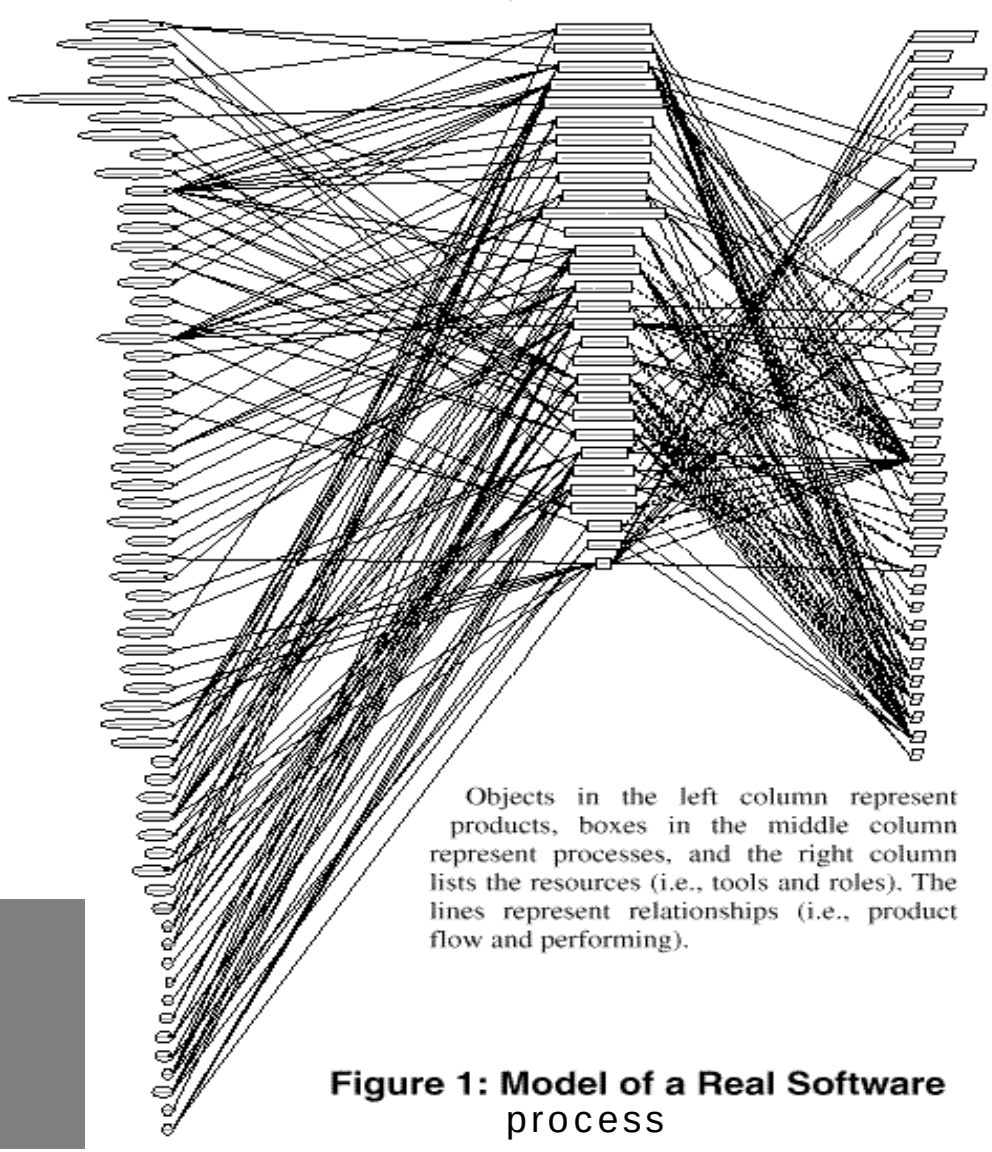


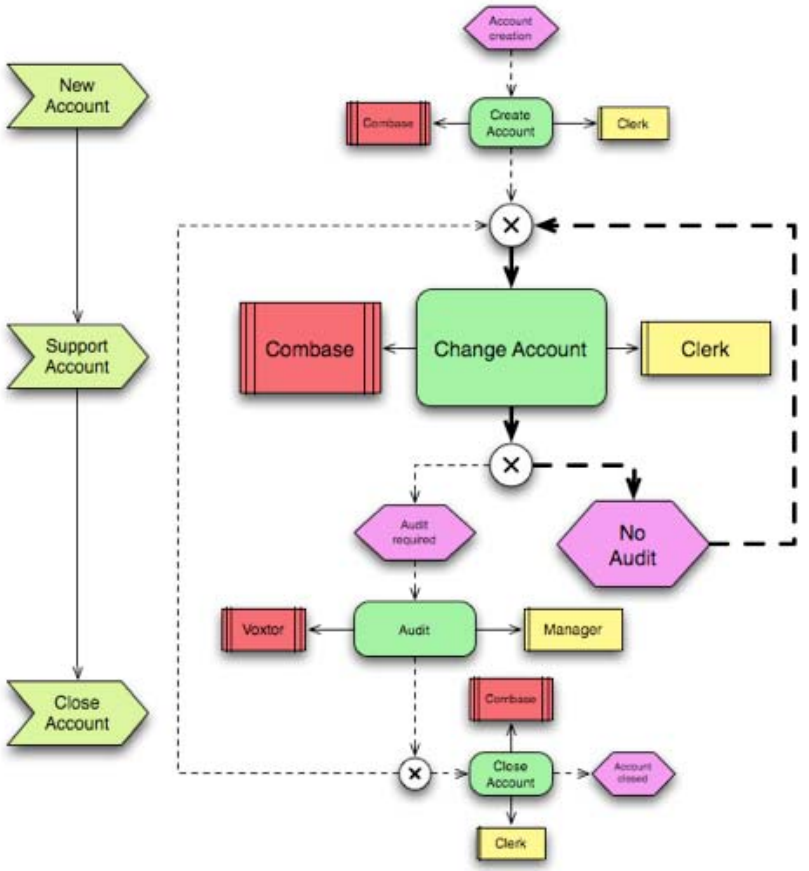
Figure 1: Model of a Real Software process

Modeling Languages (suitable for PM)

- Flowchart is a schematic representation of an algorithm or a stepwise process,
- IDEF is a family of modeling languages, the most notable of which include IDEF0 for functional modeling, IDEF1X for information modeling, and IDEF5 for modeling ontologies.
- Business Process Modeling Notation (BPMN, and the XML form BPML) is an example of a Process Modeling language.
- Extended Enterprise Modeling Language (EEML) is commonly used for business process modeling across a number of layers.
- Unified Modeling Language (UML) is a general modeling language to describe software both structurally and behaviorally. It has a graphical notation and allows for extension with a Profile (UML).

Process Modeling Tools

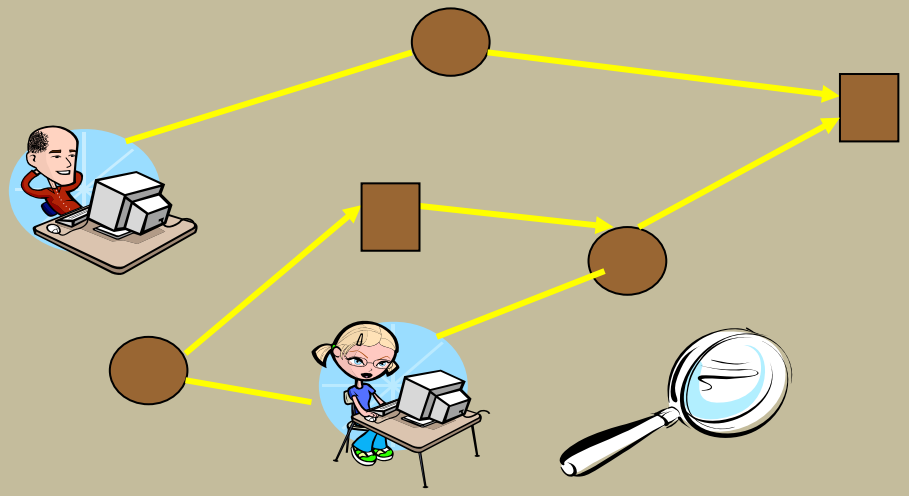
- Commercial tools not dedicated to process modeling
  - E.g., UML tools, ABC Flowcharter, Microsoft Visio, Statemate
- Workflow Management Systems
  - E.g., ARIS Toolset (event-driven process chains, EPC)
- Research prototypes
  - E.g., Spearmint

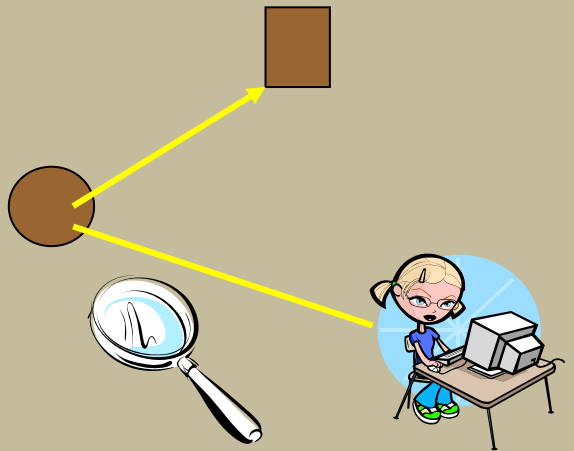


UNIVERSITETET I OSLO

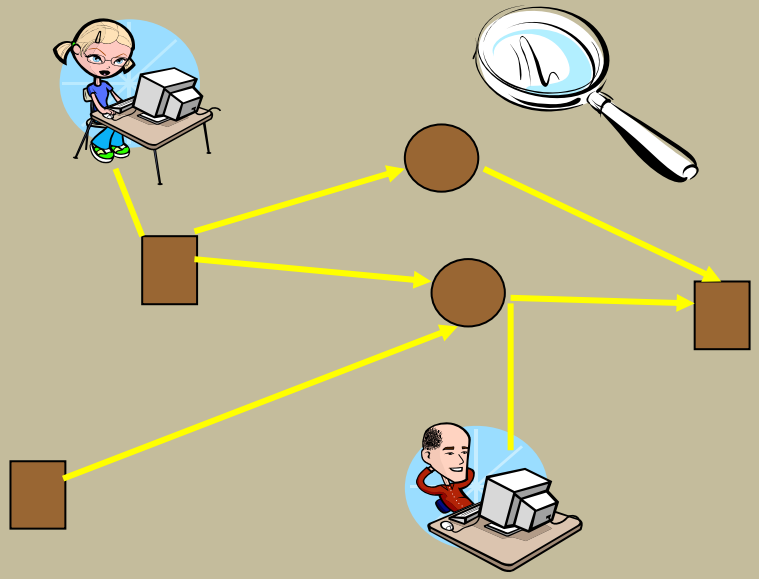
Example SPM Tool Spearmint™

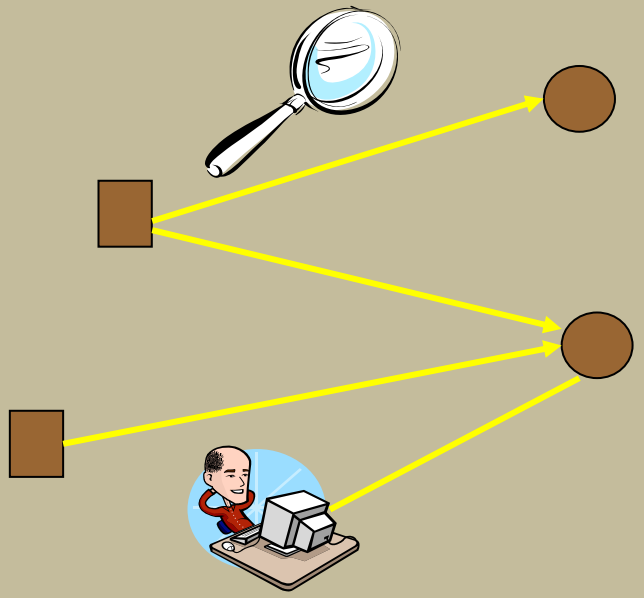
- SPEARMINT™ – ***S***oftware ***P***rocess ***E***licitation, ***A***nalysis, ***R***eview, and ***M***anagement in an in***T***egrated Environment
- Assists a process engineer in creating and maintaining complex process models.
- Allows for efficient modeling of different views of the process model
- Generates EPG (Electronic Process Guide)

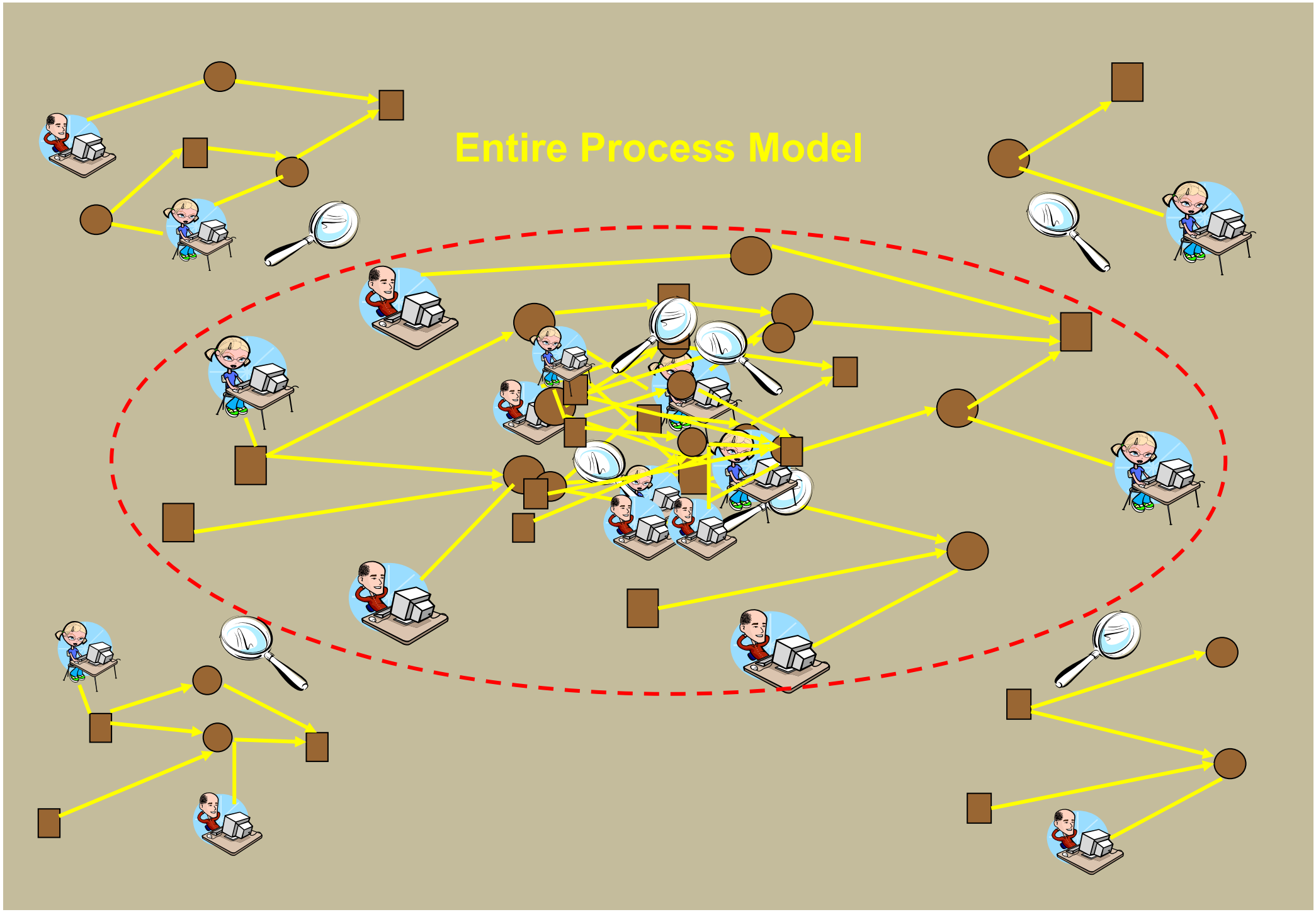




Views

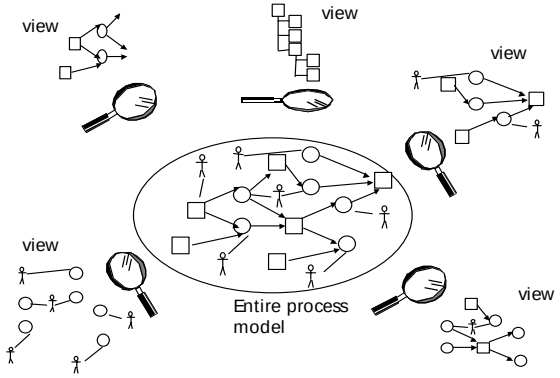





Entire Process Model

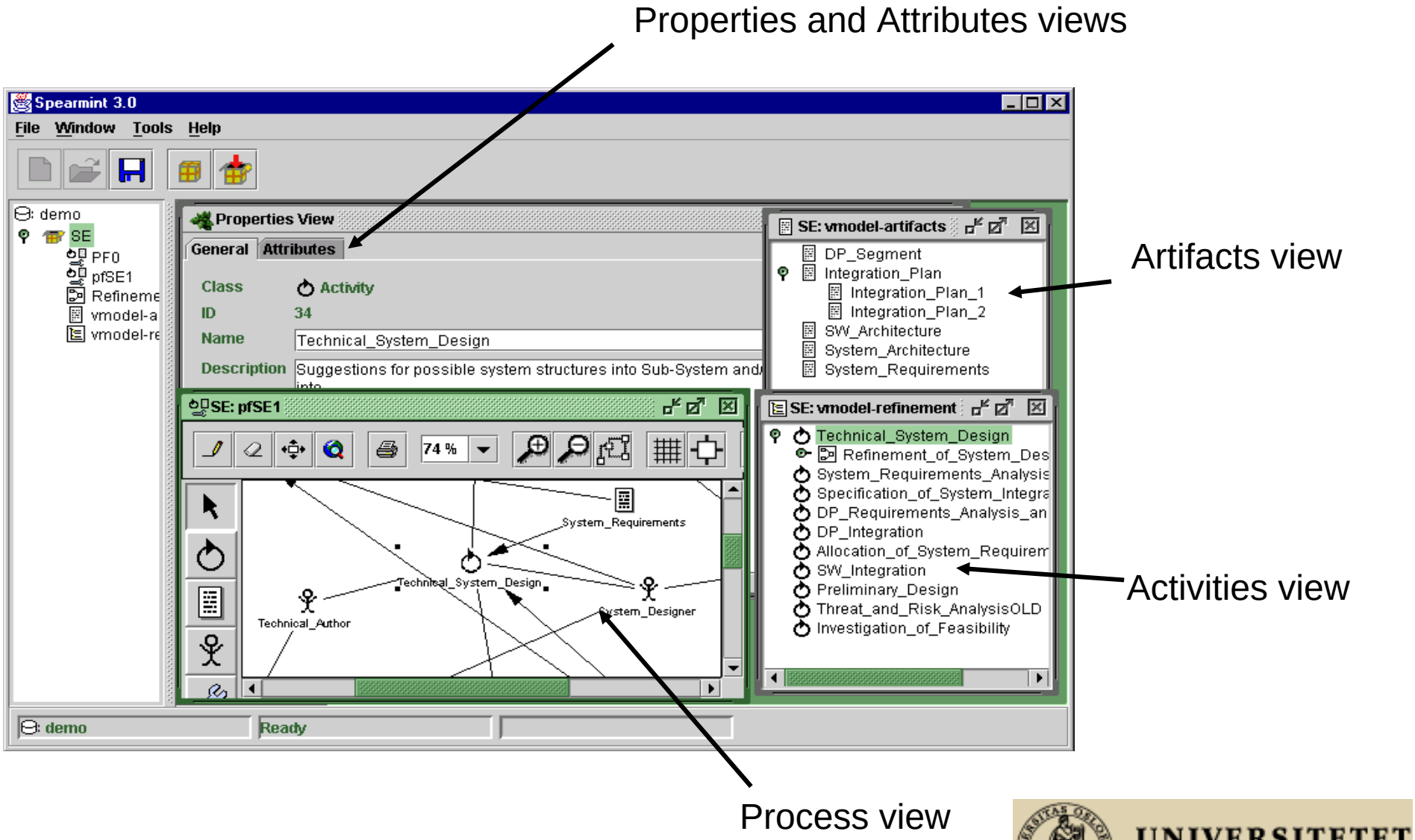
Views

Spearmint supports efficient modeling by supporting different views



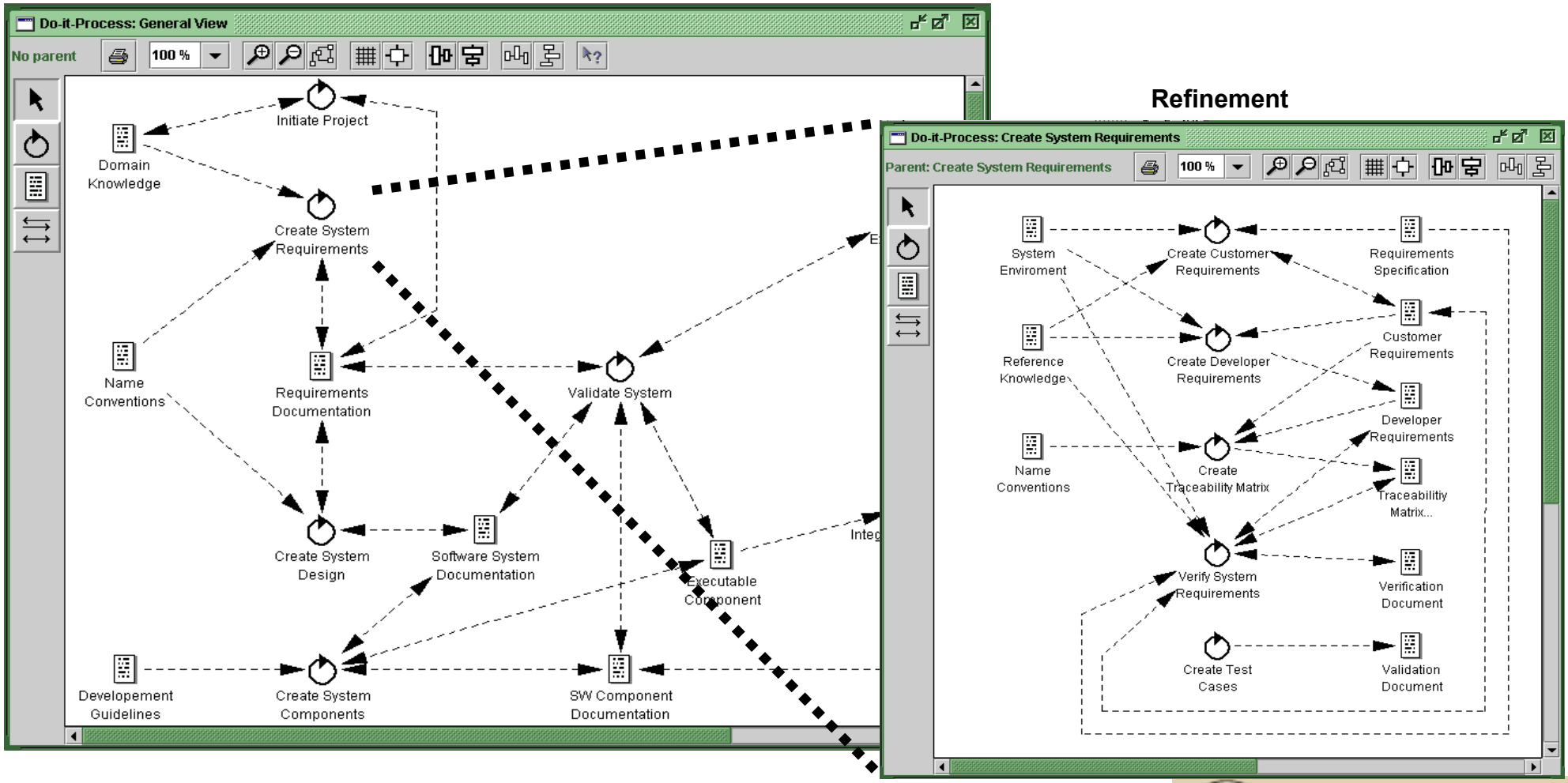
- A view is a part of the process model
  - Spearmint describes not the whole process, but only parts of it in pre-defined and user-defined views.
- A view highlights certain aspects
  - Working with views reduces the complexity of the process model.
  - Only those aspects of a model are contained, which are relevant for specific tasks.
- SPEARMINT checks consistency of all views
  - Process elements in a certain view always reference to the whole process model.

Views

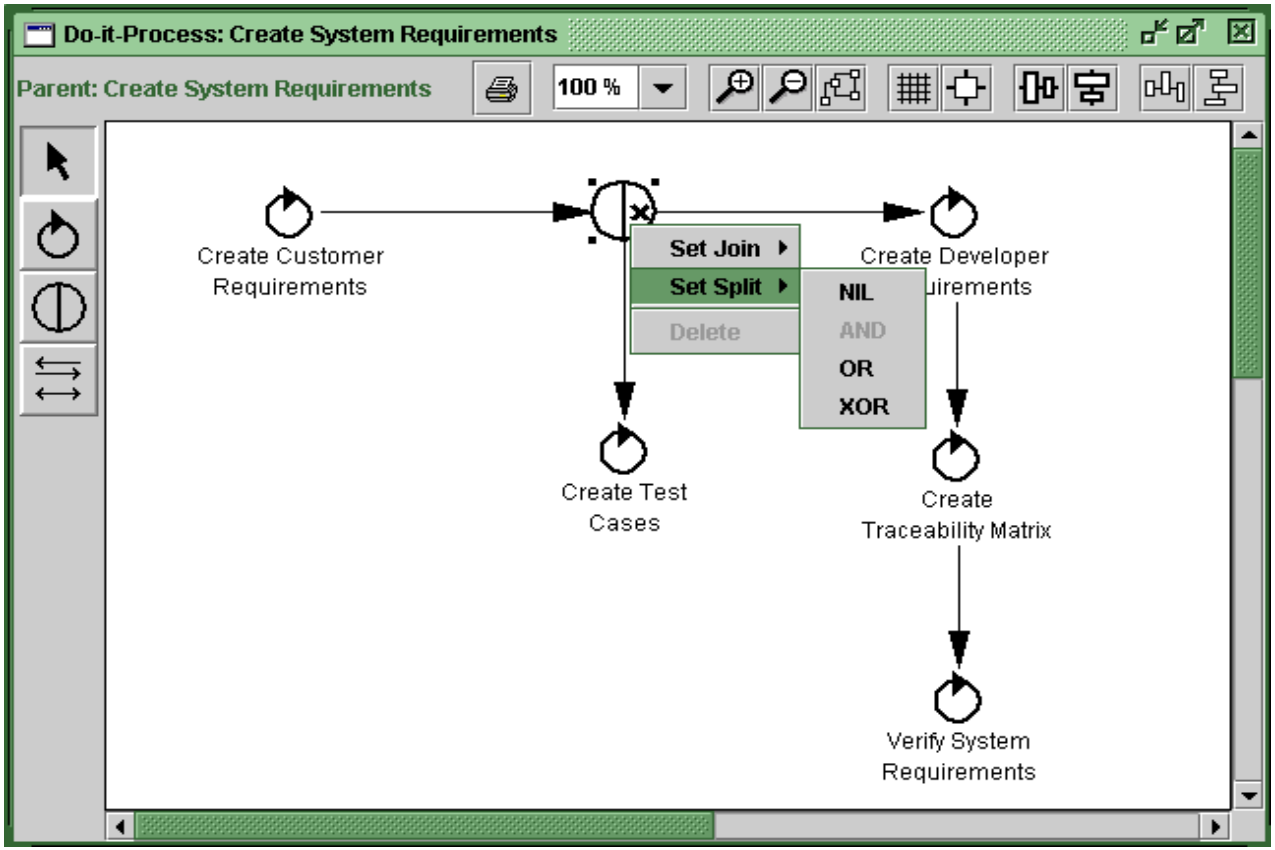


UNIVERSITETET I OSLO

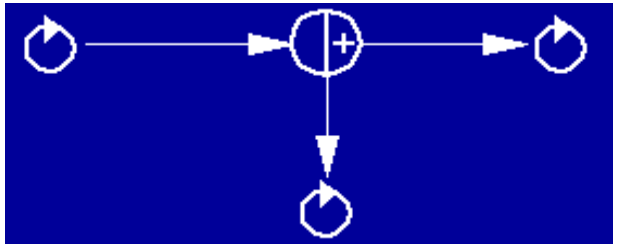
Product-Flow View

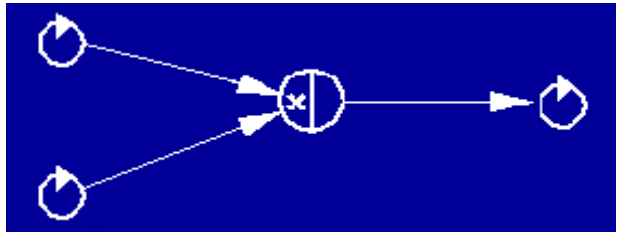


Control-Flow View

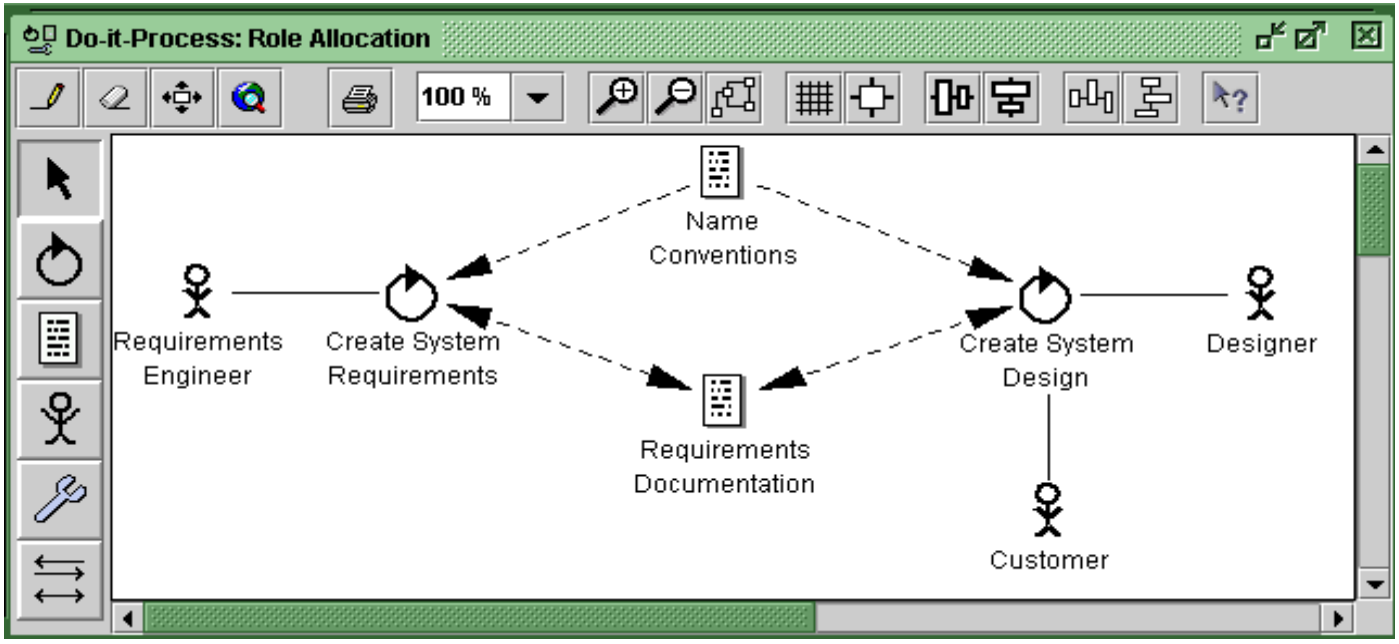


Join/Split symbols:
AND (x), OR (+), XOR(⊕)
nil (empty).

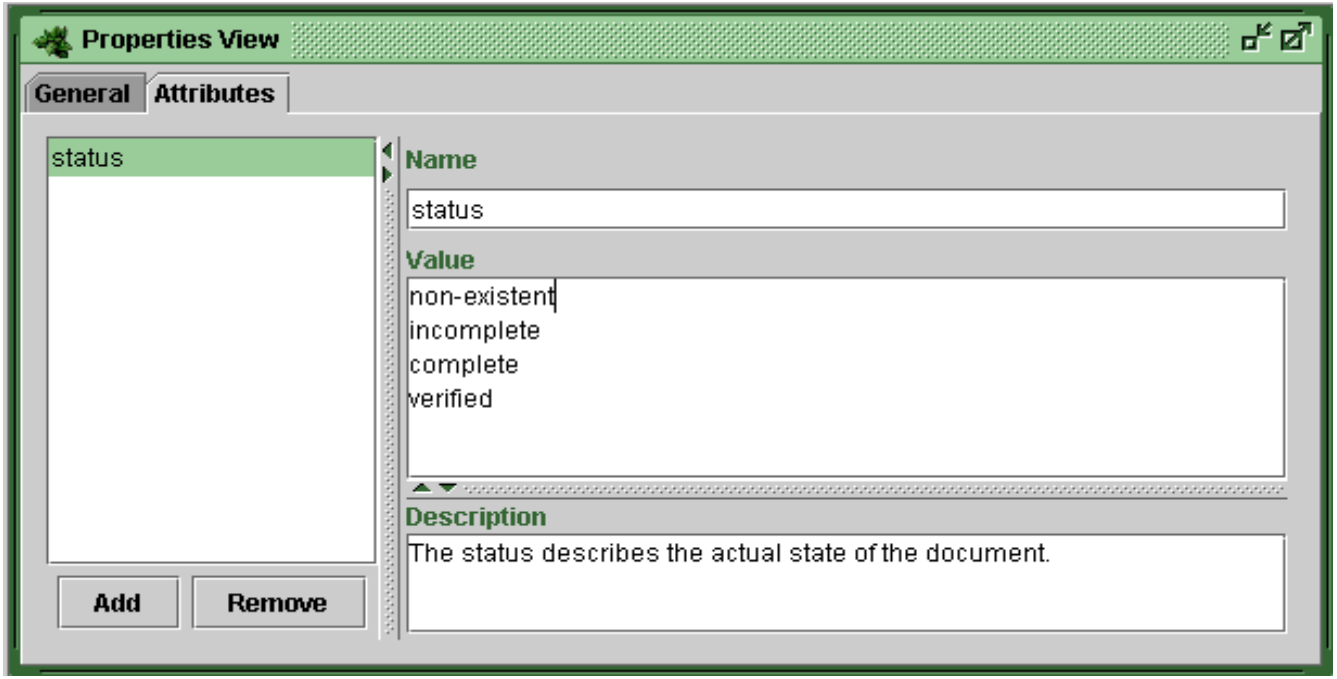




Process View

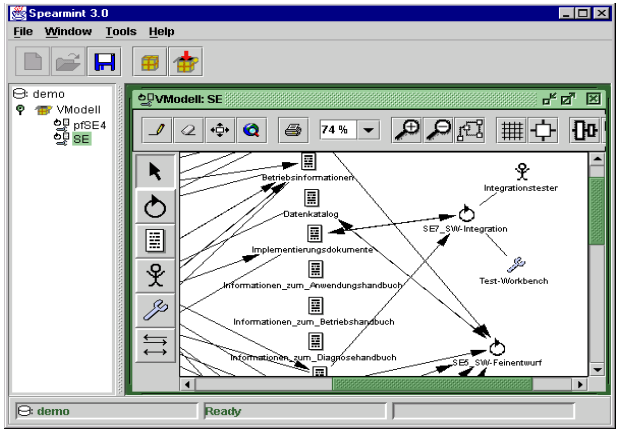


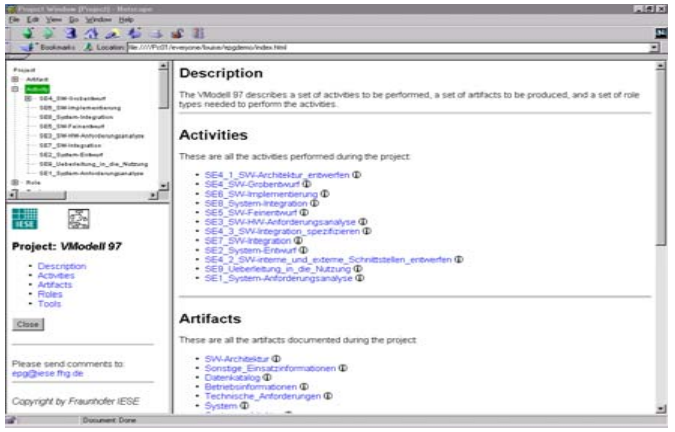
Attributes View



Generation of Hypertext View (EPG)

Process-engineer



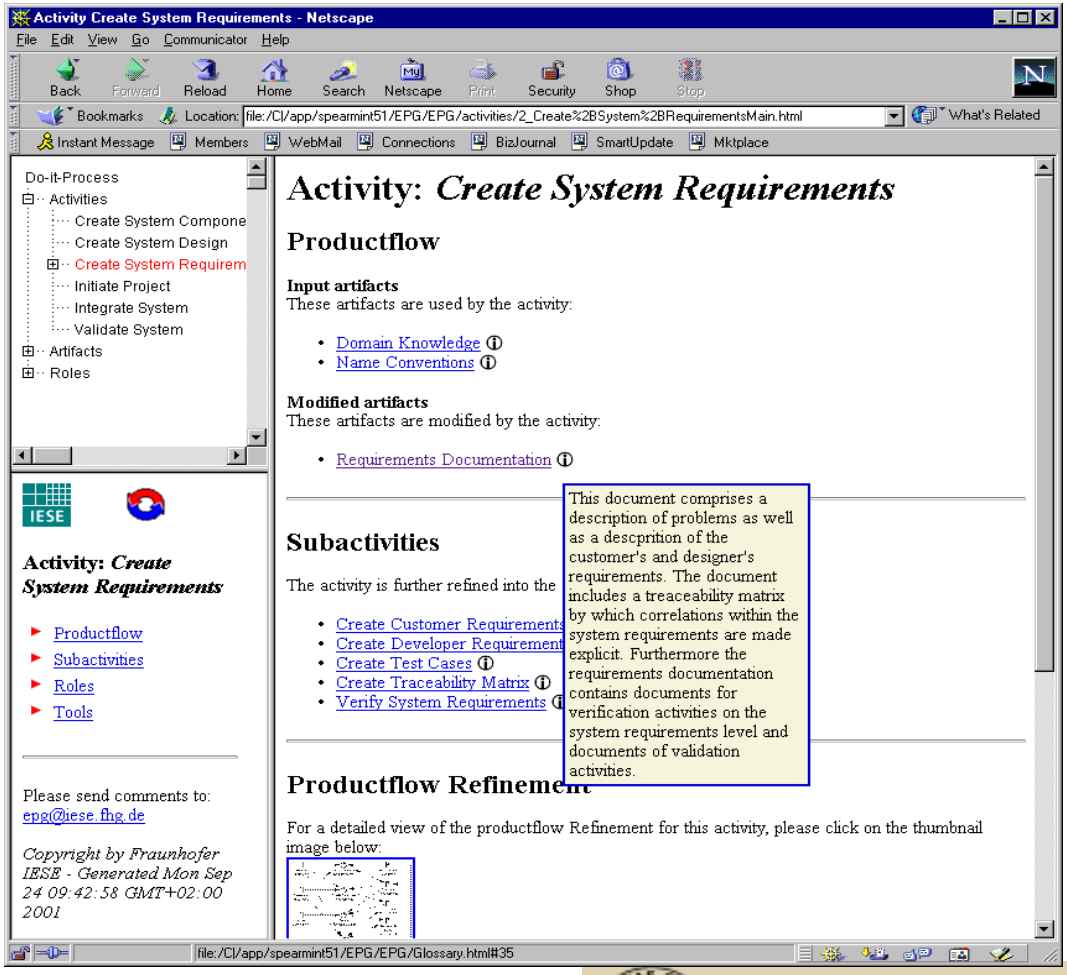


Process-user
(e.g. developer)

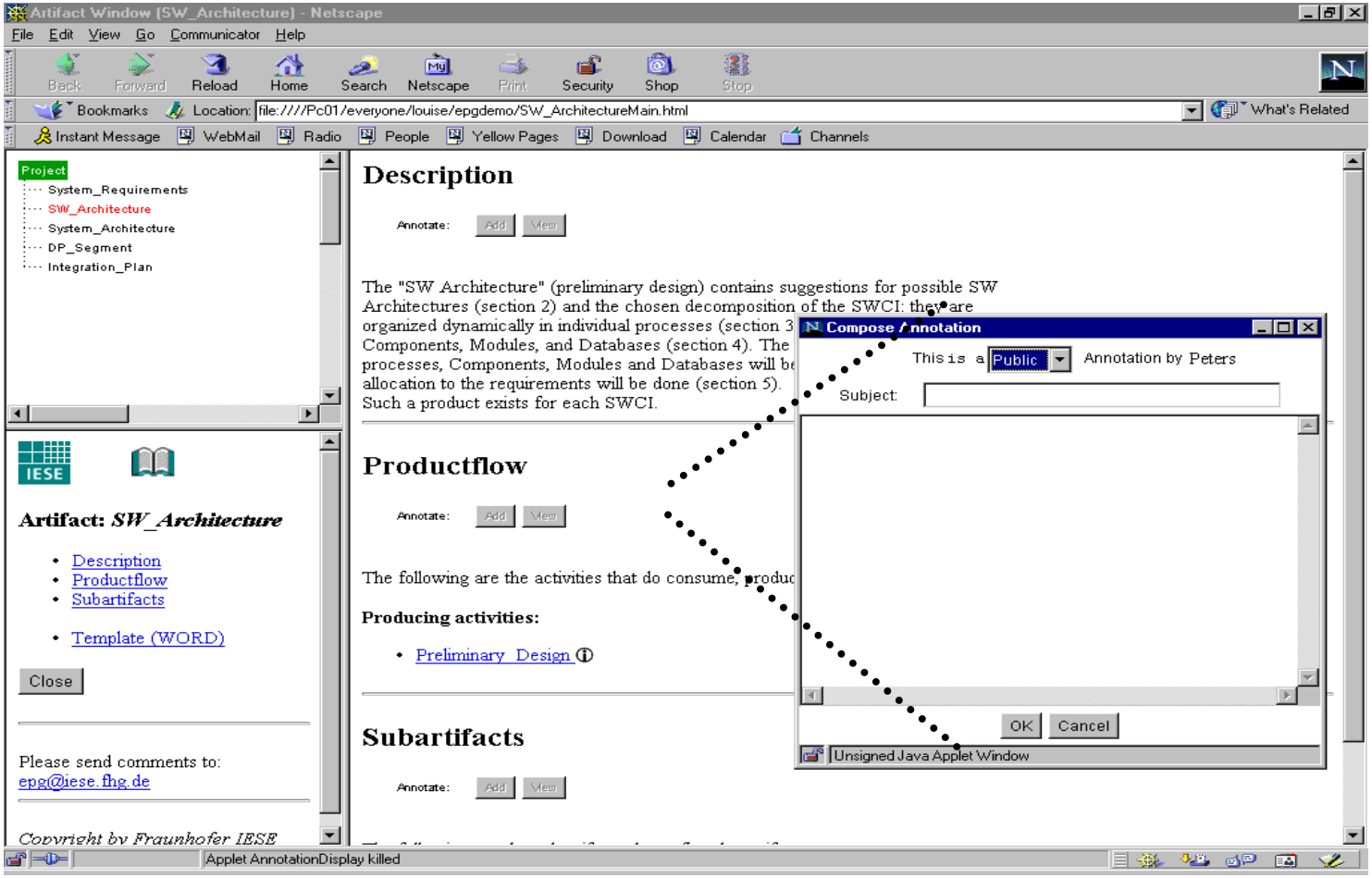
Feedback, e.g. via
annotations

UNIVERSITETET I OSLO

Hypertext-View (EPG)



Annotations



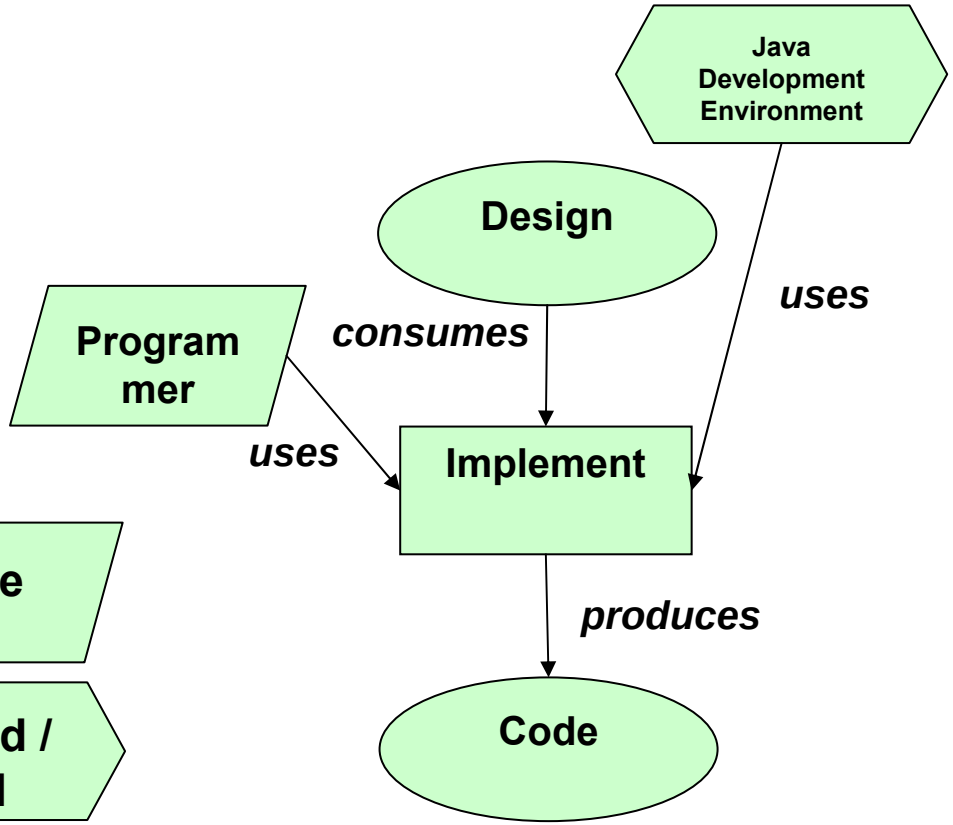
Consistency Checking

- Process models should be complete and correct representations of reality
- Consistency checking has been partly automated in SPEARMINT™
- Methodological prerequisites:
  - Process meta-model
  - Consistency rules

Structure of Lecture 02

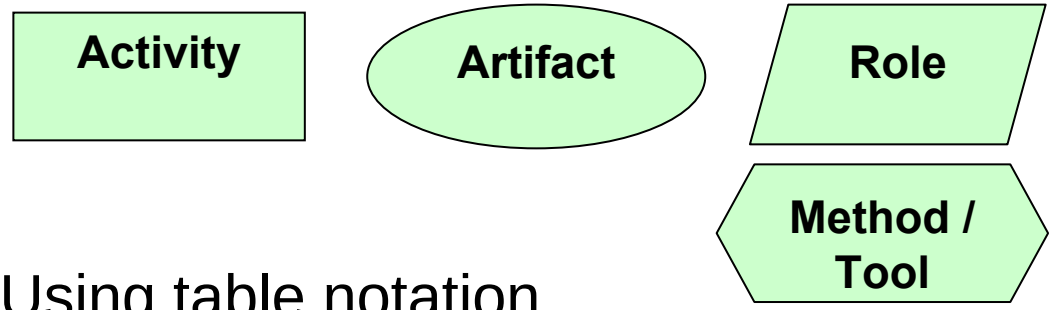
- Hour 1:
  - Introduction into Process Modelling
  - Prescriptive Process Models
- Hour 2:
  - Process Families/Standards
  - Descriptive Process Modelling
- Hour 3:
  - Exercises

Exercise 1



Process Model representations:

- Using product-flow notation



- Using table notation

Activity Name	Input Artifact	Output Artifact	Roles	Methods / Tools
…				

Exercise 1 (cont'd)

- Model the following process:

  "*Based on input from Marketing and from Customers, the Product Owner sets up the product backlog. The Product Owner is also in charge of planning sprints. He/she does this based on a prioritization of the user stories contained in the product backlog, and on effort estimates for each user story received from the Team. The Team does their effort estimates based on a refinement of user stories into tasks. Once a sprint has been defined, the Team develops the software related to a sprint. The Team does this by working on the previously identified tasks. To monitor their work, a burn-down chart is maintained. The burn-down chart shows how much of a task has been completed and how much effort is still to be used. During the development of a sprint, the Scrum Master supports the Team by helping them overcome obstacles and by guiding them through the agile methodology. Once a sprint is complete, a sprint review meeting will be performed. Everybody is invited to attend this meeting.*"

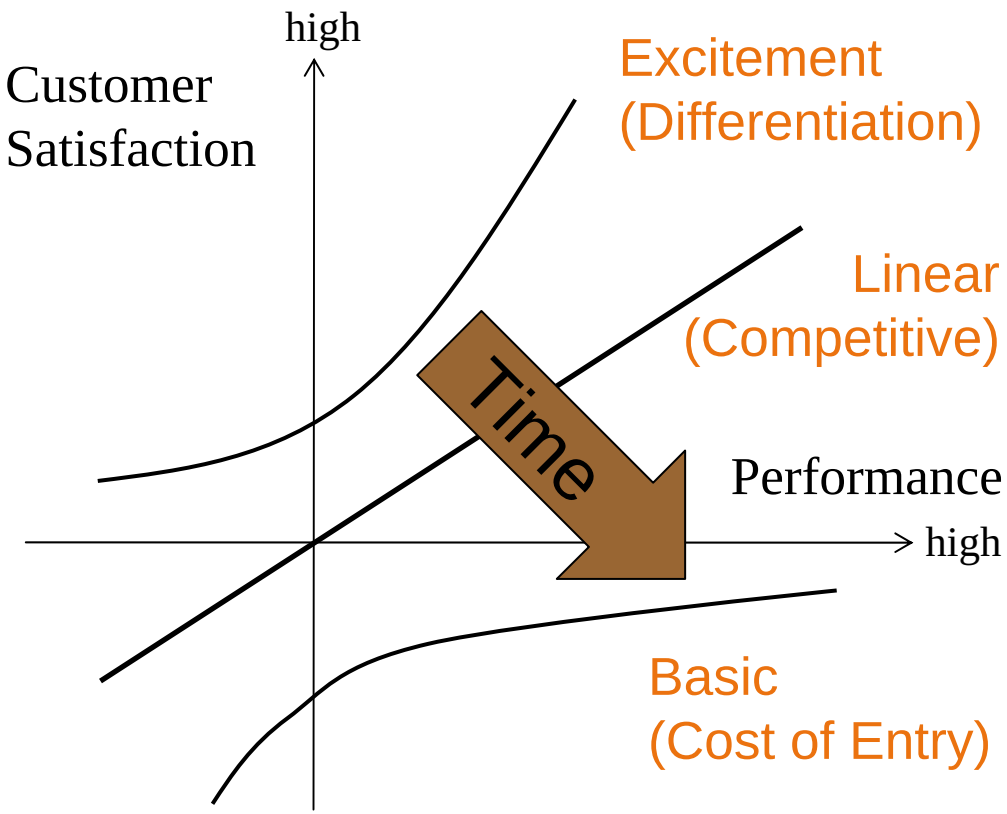
Exercise 2

- Work in pairs
- Task 1:
  - Decide who will be the “process performer” (role P) and who will be the process modeller (role M)
  - P think about a process (related to software development) and explains it to M.
  - M models the process (as in Exercise 1)
- Task 2:
  - Take turns (i.e., switch roles) and repeat task 1.
- Task 3:
  - Show your process models to someone else (not in your pair) and let that person explain the process to P.

Exercise 3 – Homework

- Task:
  - Model the process of surveying “Customer Satisfaction” using the Kano-Model
  - Specify activities, artifacts, roles, tools/techniques/methods
  - Use either the graphical or the table notation

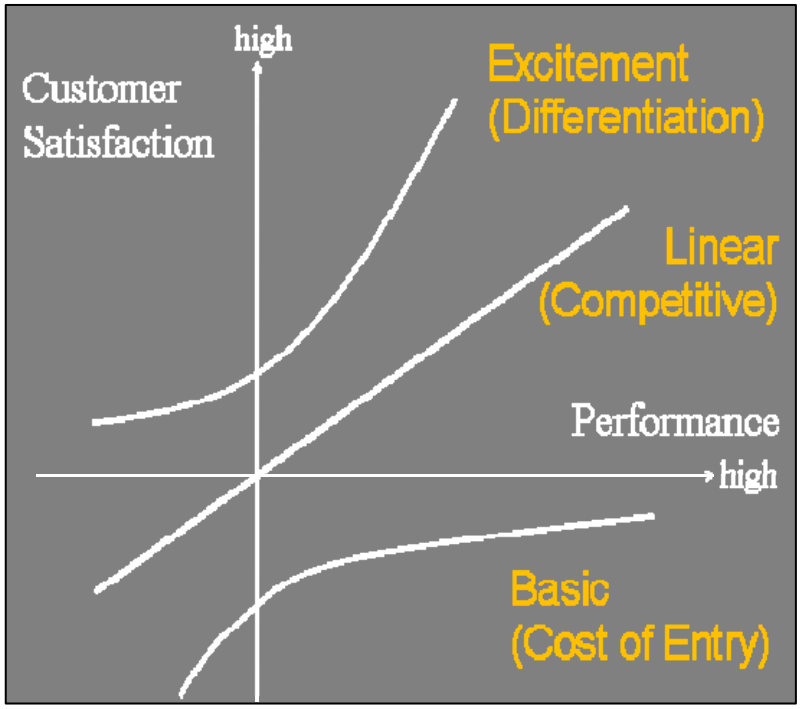
The Kano-Model



Five dimensions of quality:

- ”Basic quality” – satisfies basic “must-have” needs which probably do not even need to be specified.
- ”Competitive quality” - satisfies expressed needs (usually in requirement specification).
- ”Excitement quality” - satisfies latent needs, needs which are there but which the user hasn’t expressed and/or is himself/herself aware of
- ”Indifference quality” - needs which are covered but which user is indifferent to
- ”Reverse quality” - qualities which the customer do not want

The Kano-Model – Surveying Users



- To assess whether a feature is basic, linear, or exciting we can:
  - Sometimes guess
  - Survey a small set of users (20-30)
- We ask two questions:
  - A functional question:

    How do you feel if a feature is present?
  - A dysfunctional question:

    How do you feel if that feature is absent?

Functional and Dysfunctional Forms

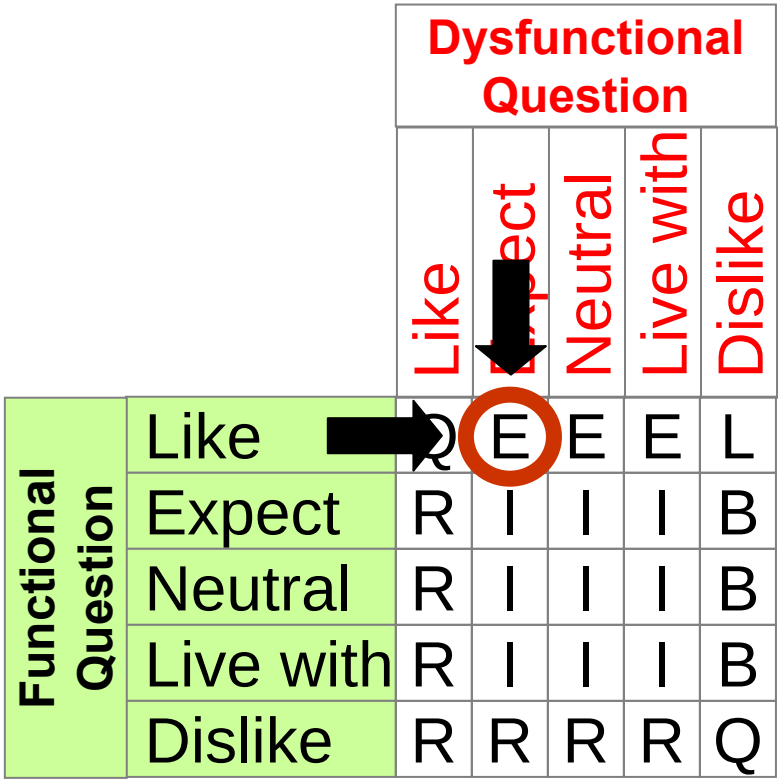
Functional form of question →

If your editor includes a voice recognition function, how do you feel?	
	I like it that way
	I expect it to be that way
	I am neutral
	I can live with it that way
	I dislike it that way

Dysfunctional form of question →

If your editor does not include a voice recognition function, how do you feel?	
	I like it that way
	I expect it to be that way
	I am neutral
	I can live with it that way
	I dislike it that way

Categorizing an Answer Pair



B: Basic (Mandatory)

L: Linear

E: Excitement

R: Reverse

I: Indifferent

Q: Questionable

Next Lecture

- Topic: Processes and Process Modeling (Section B)

- For you to do:
  - Do the homework
  - Continue thinking about your project (topic & presentation)