INF5180: Software Product- and Process Improvement in Systems Development

Part 02:

Processes and Process Modeling

(Section A)

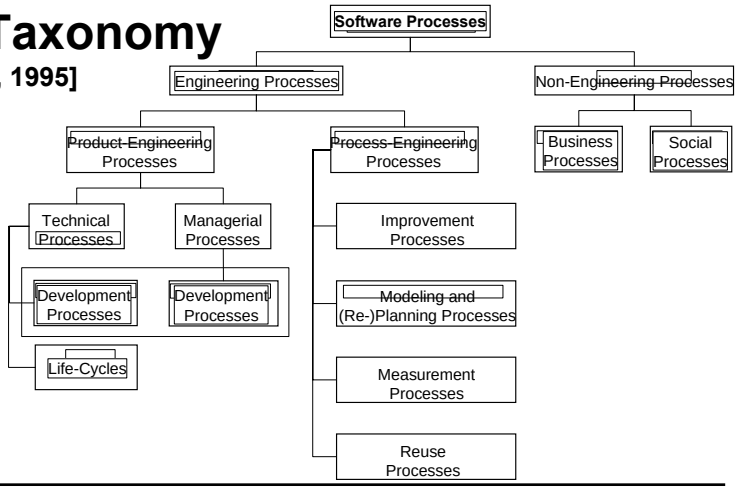
Dr. Dietmar Pfahl

email: dietmarp@simula.no

Spring 2010

A Process Taxonomy

[Rombach & Verlage, 1995]



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.

UNIVERSITETET
I OSLO

What is a (Software Development) Process?

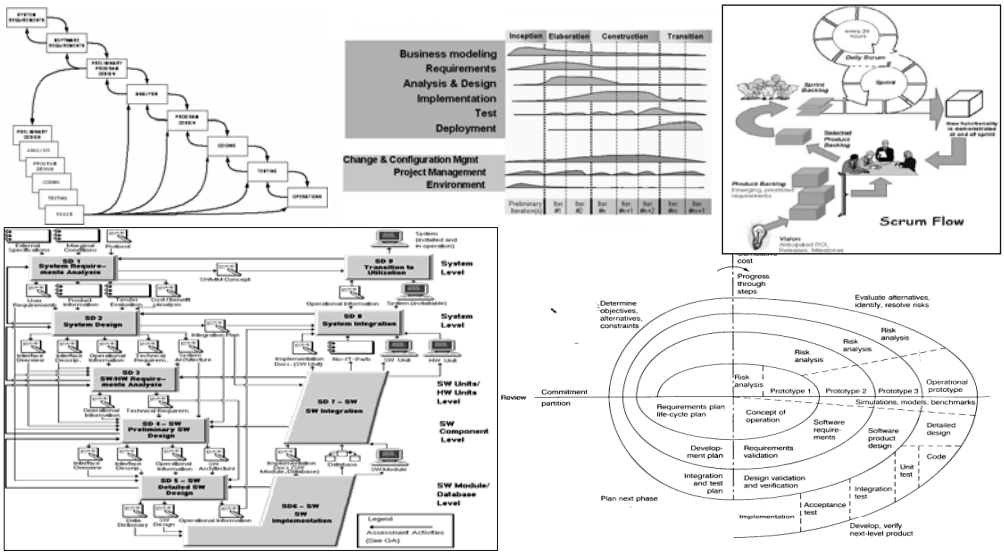
A Process …

- defines **Who** is doing **What**, **When** and **How** to reach a certain goal.
  - In software engineering the goal is to build a software product or to enhance an existing one

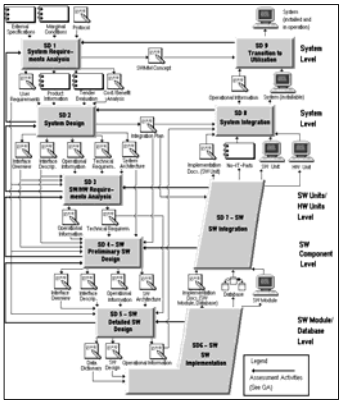
An Effective Process …

- provides guidelines for efficient development of quality software
- reduces risk and increases predictability
- promotes common vision and culture

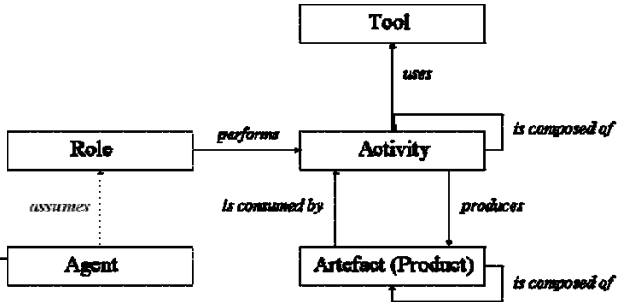
SW (Product-)Engineering Process (Model) Examples



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:



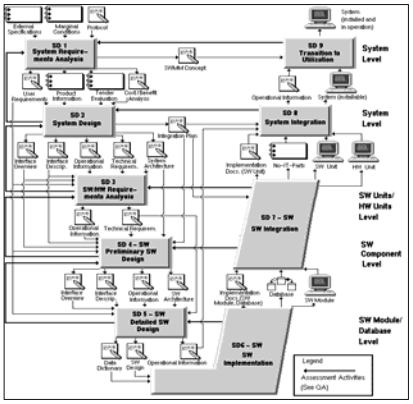
UNIVERSITETET I OSLO

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 reuse of process knowledge
- To support automatic process enactment
  - Workflow support
  - CASE tools

UNIVERSITETET I OSLO

Characterization of Process Models

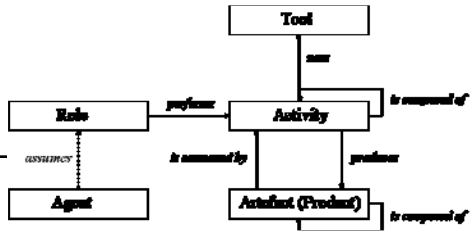


A *Process Model* defines:

- an identifiable activity or a group of activities
- a hierarchy of activities
- the control flow between activities
- the input/output products of activities
- the 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



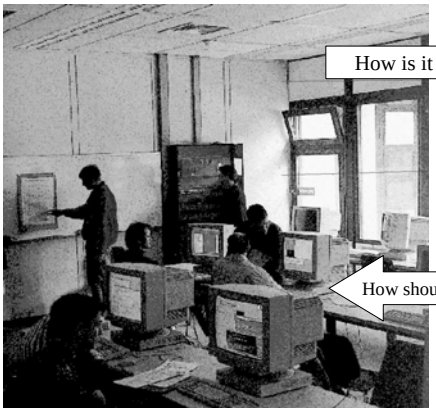
Role Responsibilities

RASCI Matrix

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

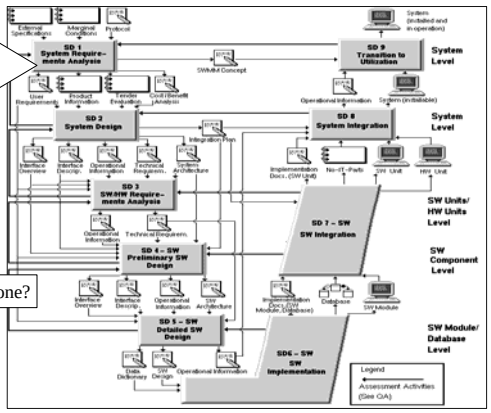
R = Responsible
A = Approve
S = Support
C = Consult
I = Inform

Descriptive vs. Prescriptive Process Models



How is it done?

How should it be done?

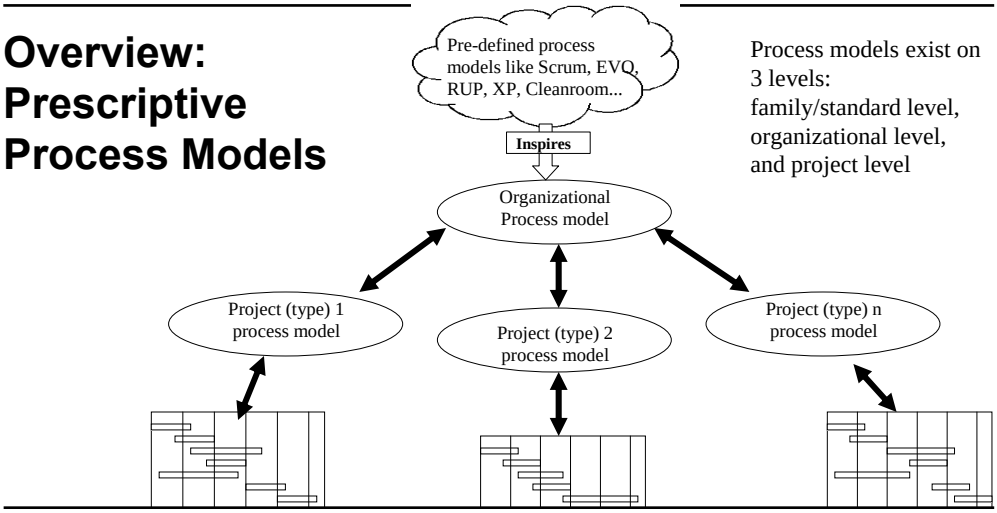


Prescriptive vs. Descriptive Process Models

- 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

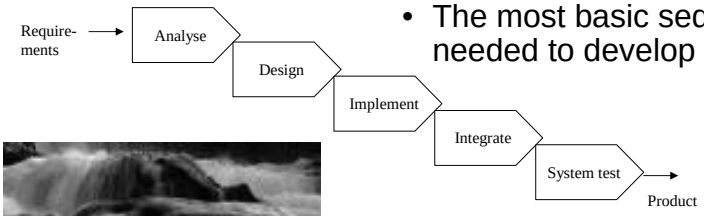
Prescriptive Process Modeling

Overview: Prescriptive Process Models



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

An Example Process: The Waterfall Model



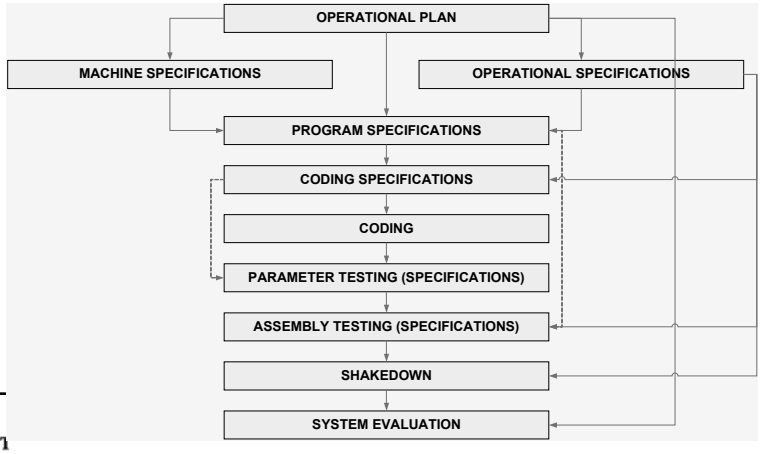
- The most basic sequence of activities needed to develop sizable software

- Well tried, well documented model
- Well suited to successive breakdown
- Is simple to manage
- Requires stability
- Requires careful analysis and planning
- Results in ”big-bang integration”

Early Waterfall Model (1956)

(Ref: Boehm, ICSE 2006 keynote)

The SAGE Software Development Process (Benington, 1956)



Waterfall: Royce Model (1970)

Idea:

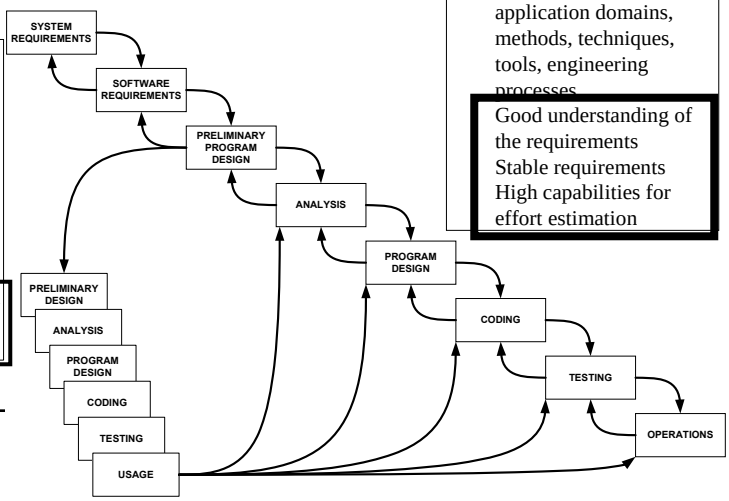
Sequential creation of products on different levels of abstraction (e.g., precede code by design, precede design by requirements) and integration in reverse direction

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

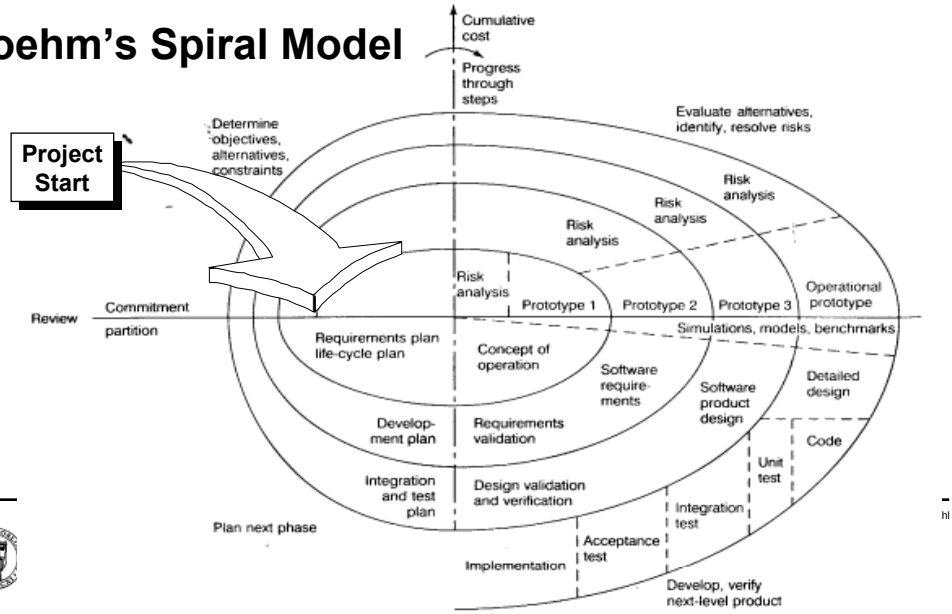


Often Waterfall is Bad

- 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!

UNIVERSITETET I OSLO

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

UNIVERSITETET I OSLO

Activities & Cycles in Boehm's Spiral Model

- Concept of Operation
- Software Requirements
- Software Product Design
- Detailed Design
- Code
- Unit Test
- Integration and Test
- Acceptance Test
- Implementation

- For each cycle go through these activities
  - Quadrant IV: Define objectives, alternatives, constraints
  - Quadrant I: Evaluate alternatives, identify and resolve risks
  - Quadrant II: Develop, verify prototype
  - Quadrant III: Plan next "cycle"
- The first 3 cycles are shown in a polar coordinate system.
  - The polar coordinates r = (l, a) of a point indicate the resource spent in the project and the type of activity

UNIVERSITETET I OSLO

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

Limitations of the Waterfall and Spiral Models

- Neither of these model deals well with frequent change
  - The (simplified) Waterfall model assumes that once you are done with a phase, all issues covered in that phase are closed and cannot be reopened
  - The Spiral model can deal with change between phases, but once inside a phase, no change is allowed
- What do you do if change is happening more frequently? (“The only constant is the change”)

Iterative Enhancement: Overview

- **Origin: Basili und Turner, 1975**
- **Idea:**
  - 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

UNIVERSITETET I OSLO

Iterative Enhancement (or Incremental Development)

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:

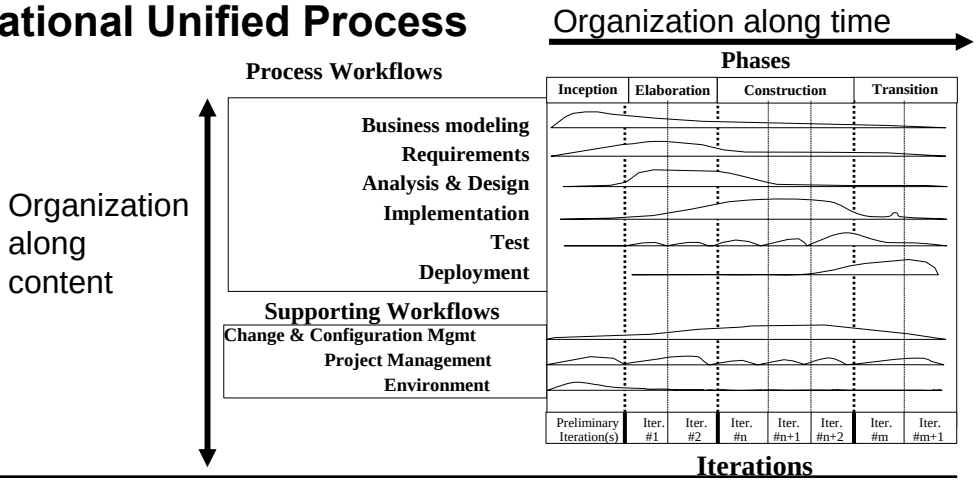
- Risk that, by ignoring specific requirements, the product will be designed in such a way that fulfilling future requirements becomes difficult/expensive
  - particularly problematic are non-functional requirements
- Comprehensive version and configuration management is necessary

UNIVERSITETET I OSLO

Unified Process

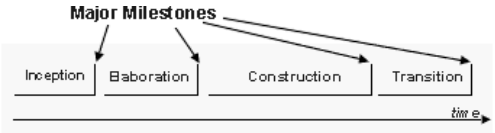
- Family: Iterative Enhancement
- Origin:
  - **Ivar Jacobson, James Rumbaugh, Grady Booch, 1998**
- Defines process framework that is adaptable to
  - various application domains
  - different organizations
  - different competence levels
  - different project sizes
- Characteristics:
  - use case driven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

Rational Unified Process



RUP 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, you 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.

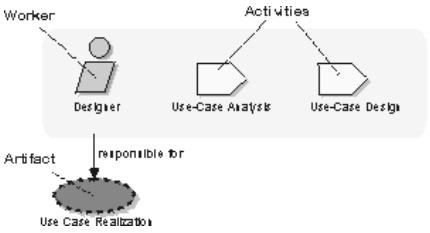
The project may be cancelled or considerably re-thought if it fails to pass this milestone.

RUP – Static Process

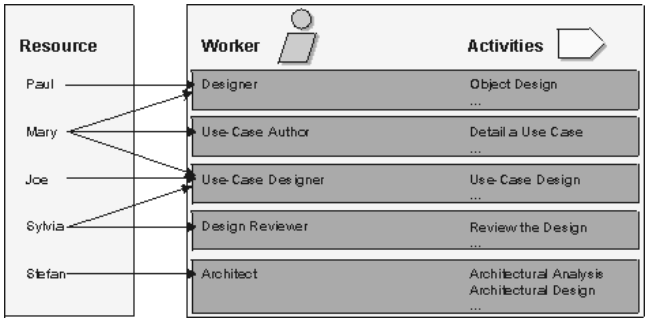
Static Structure of the Process

- A process describes who is doing what, how, and when.
- The Rational Unified Process 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 – 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 – 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

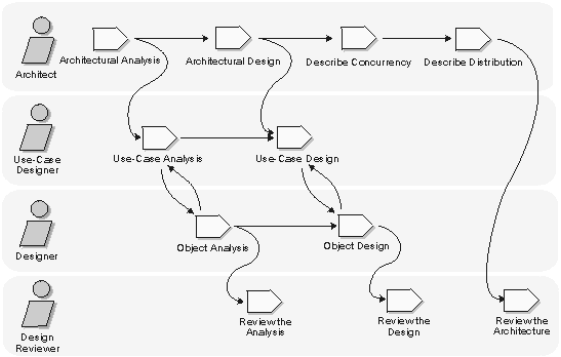
Artifact

- 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 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 right 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. The next figure shows part of a sample design model for the recycling-machine system in the use-case model shown in the previous figure.

RUP Tools

http://www-01.ibm.com/software/rational/sw-bycategory/subcategory/SW720.html

Rational Method Composer

- A flexible process platform that includes the Rational Unified Process (Learn about)

Rational Portfolio Manager

- Aligns priorities, projects, and people (Learn about)

Rational SoDA

- Automates software project documentation throughout the entire life cycle (Learn about)

Rational Suite

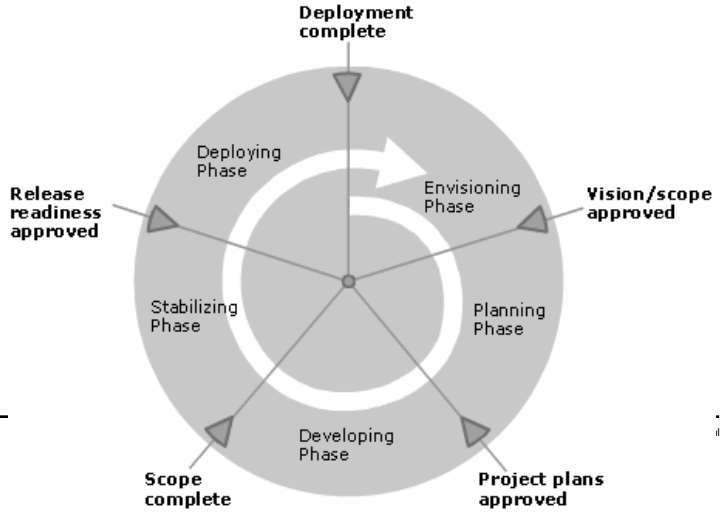
- Provides a complete, integrated lifecycle solution of best practices, tools, and services (Learn about)

Rational Team Unifying Platform

- Provides common access to development assets, requirements, and process guidance (Learn about)

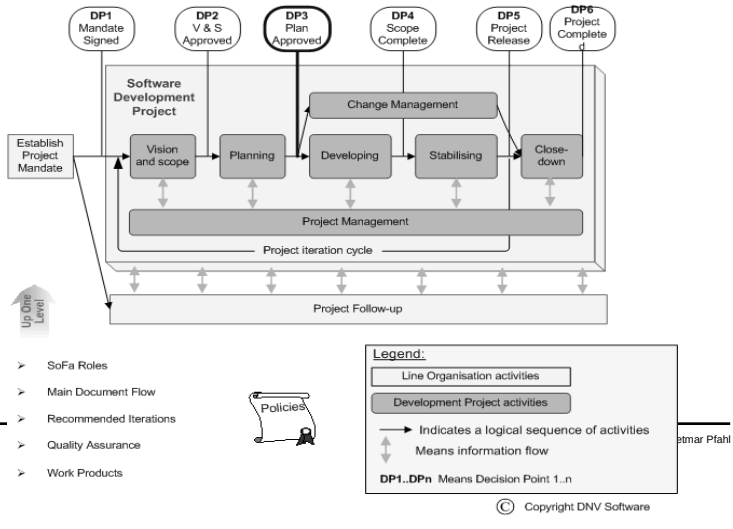
MSF (Microsoft Solution Framework)

For details pls. refer to the White Paper in p02 of the Reading Materials



MSF-Inspired Process Model (at DNV)

For sub-contractor management processes and agile development processes using the Norwegian PS2000 process standard pls. refer to the reports in p02 of the reading materials



Process Families/Standards

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

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 and Organisational Life Cycle Processes.


UNIVERSITETET I OSLO

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


UNIVERSITETET I OSLO

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

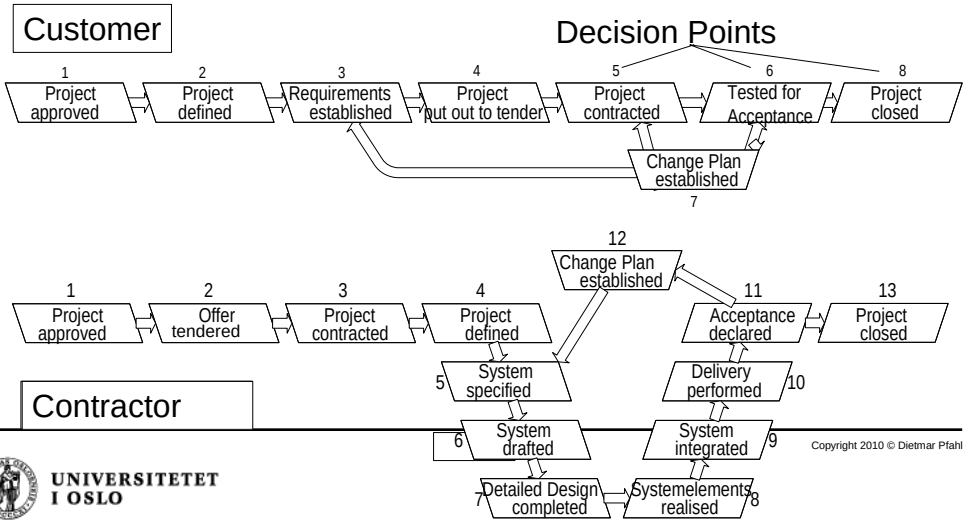

UNIVERSITETET I OSLO

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.


UNIVERSITETET I OSLO

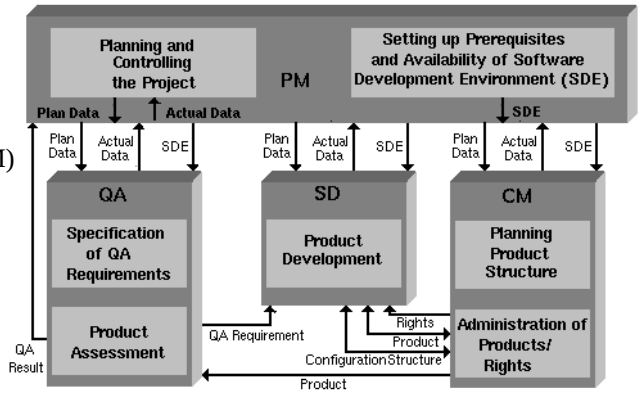
Customer vs. Contractor View



UNIVERSITETET I OSLO

V-Model: The Big Picture

The V-Model comprises
four sub-models:
System Development (SD)
Quality Assurance (QA)
Configuration Management (CM)
Project Management (PM)

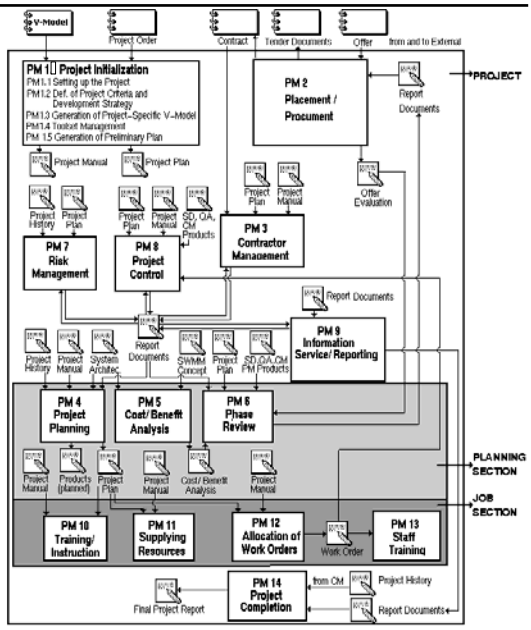


UNIVERSITETET I OSLO

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

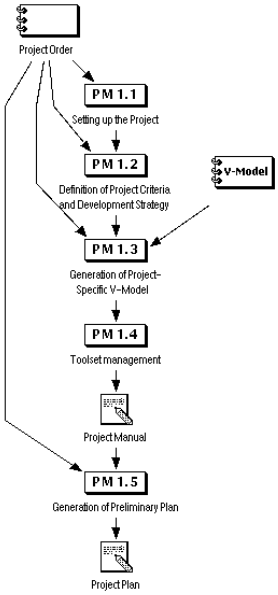
Activity Types:

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



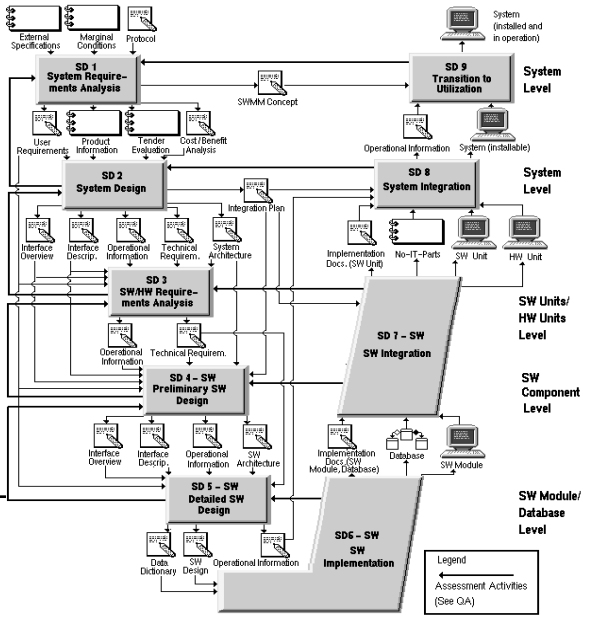
V-Model: Project Management (PM) Sub-Model

PM1: Project Initialization



"Handling" (Refinement)

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



UNIVERSITETET I OSLO

INF5180 – Spring 2009 Part 02: Processes and Process Modeling

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

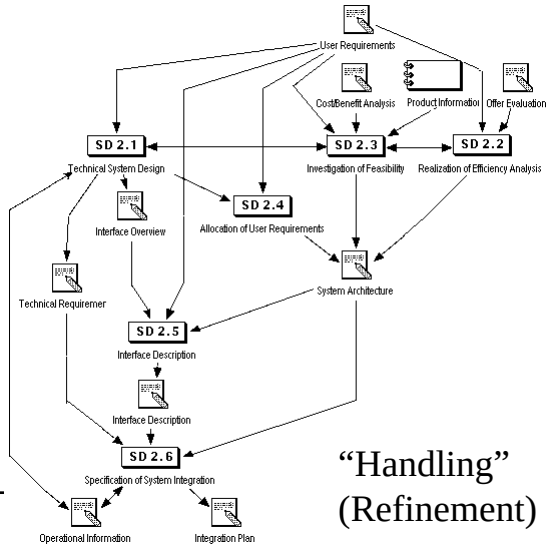
SD2: System Design

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 ECTD 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/		

UNIVERSITETET I OSLO

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

SD2: System Design



"Handling" (Refinement)

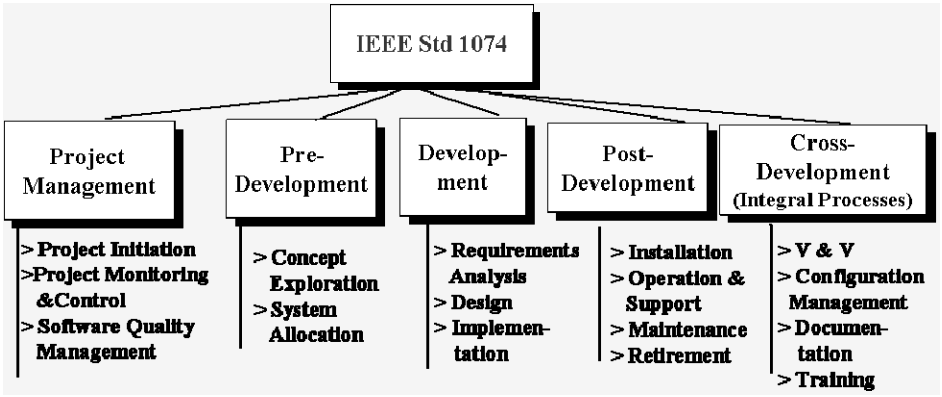
UNIVERSITETET I OSLO

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

UNIVERSITETET I OSLO

IEEE Std 1074: Standard for Software Lifecycle



Descriptive Process Modeling

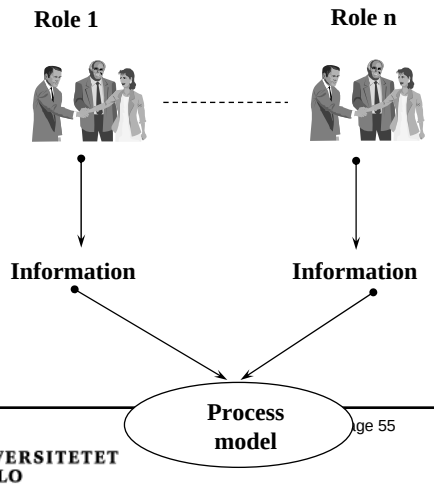
Goals of Descriptive Process Modeling

- 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

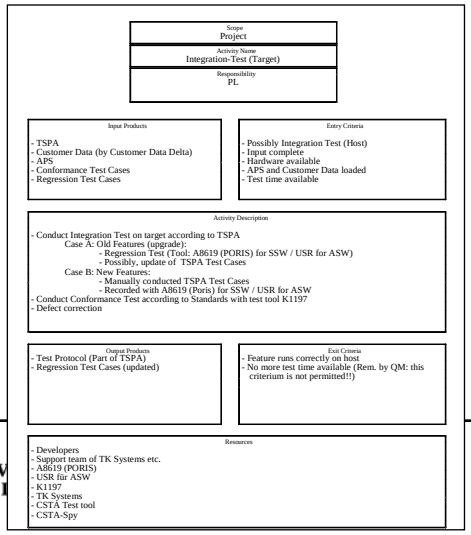


How to do it?

- 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

UNIVERSITETET I OSLO

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

UNIV
I OSL

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

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

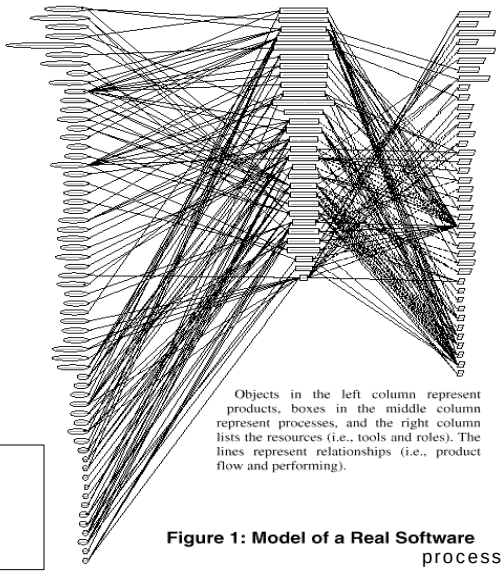
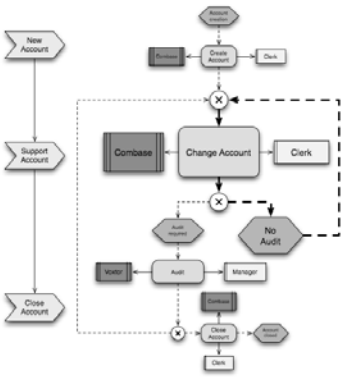


Figure 1: Model of a Real Software process

Modeling Languages (suitable for PM)

- 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.
- 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.
- Unified Modeling Language (UML) is a general modeling language to describe software both structurally and behaviorally. It has a graphical notation and allow for extension with a Profile (UML).

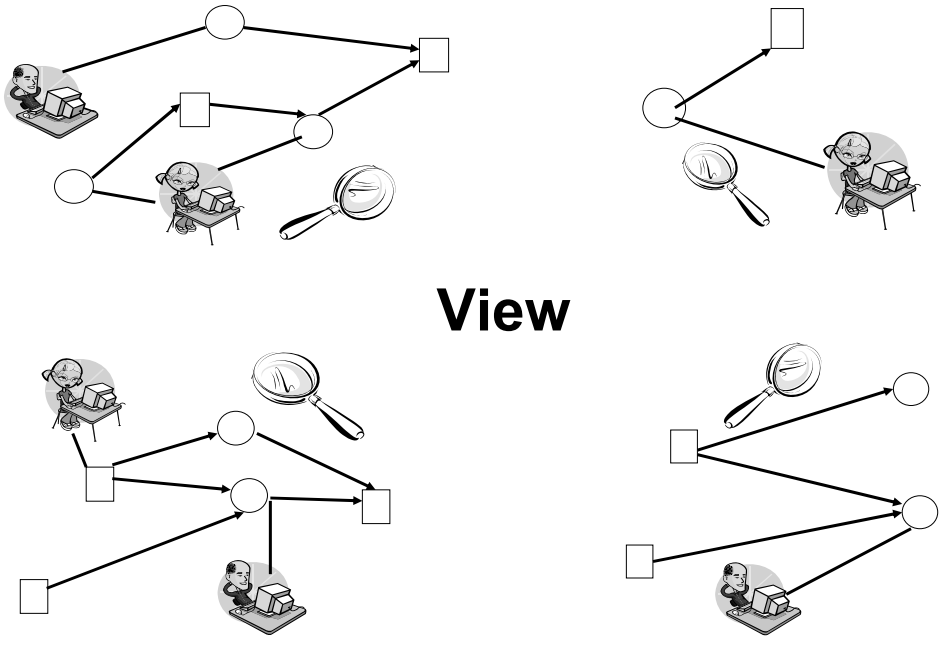
Process Modeling Tools

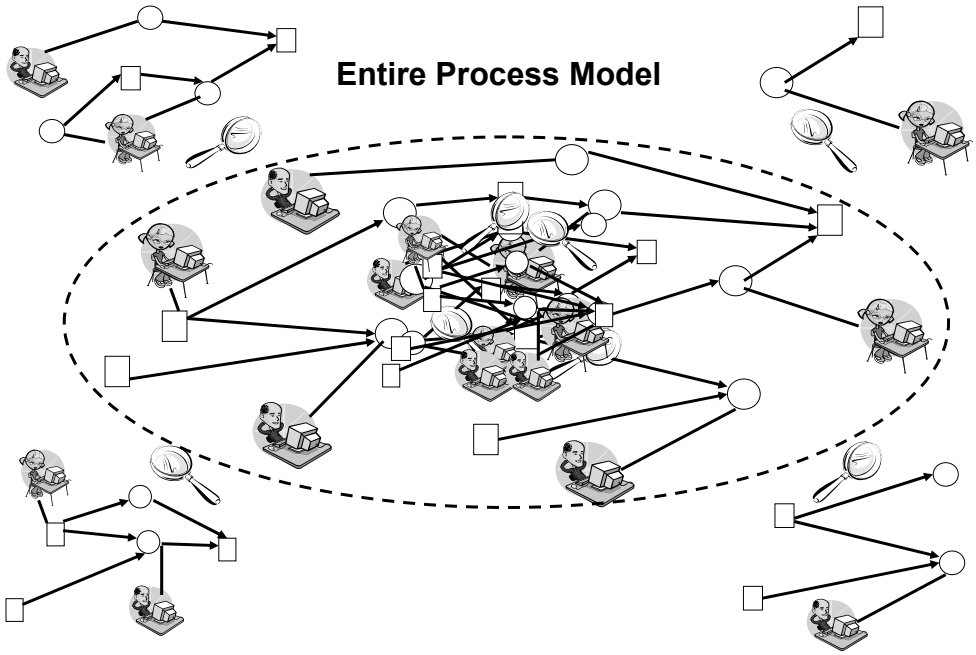


- Commercial tools not dedicated to process modeling
  - E.g., UML tools, ABC Flowcharter, Microsoft Visio, Statemate
- Workflow Management Systems
  - E.g., ARIS Toolset (event-driven process chains, EPC)
- Research prototypes
  - E.g., Spearmint

Example SPM Tool Spearmint™

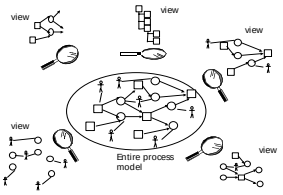
- SPEARMINT™ – ***S***oftware ***P***rocess ***E***licitation, ***A***nalysis, ***R***eview, and ***M***anagement in an in***T***egrated Environment
- Assists a process engineer in creating and maintaining complex process models.
- Allows for efficient modeling of different views of the process model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





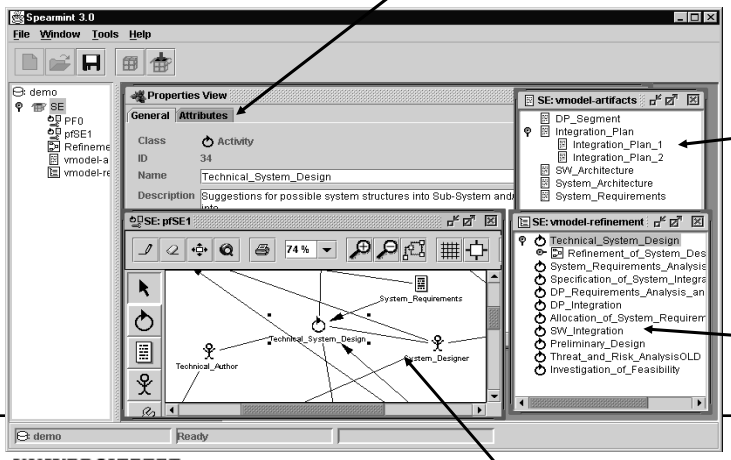
Views

Spearmint supports efficient modeling by supporting different views



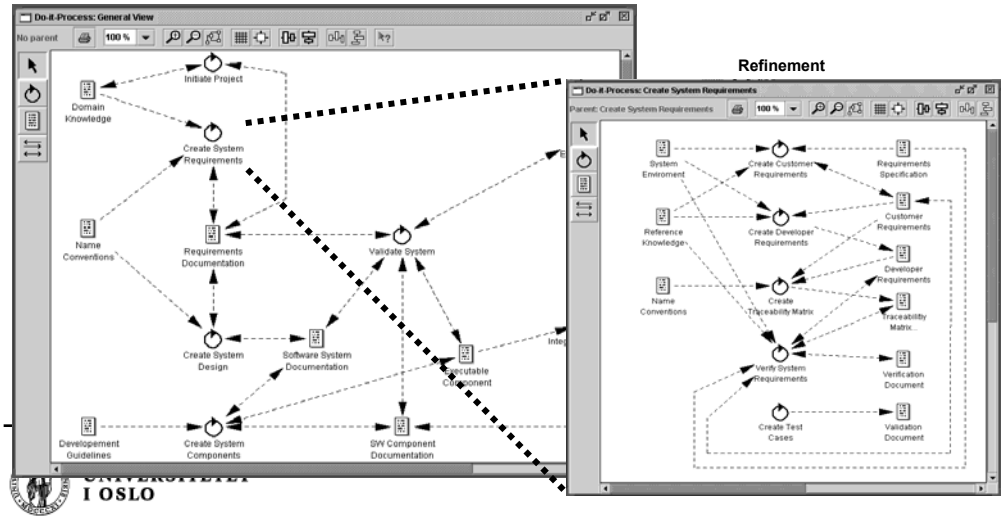
- A view is a part of the process model
  - Spearmint describes not the whole process, but only parts of it in pre-defined and user-defined views.
- A view highlights certain aspects
  - Working with views reduces the complexity of the process model.
  - Only those aspects of a model are contained, which 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



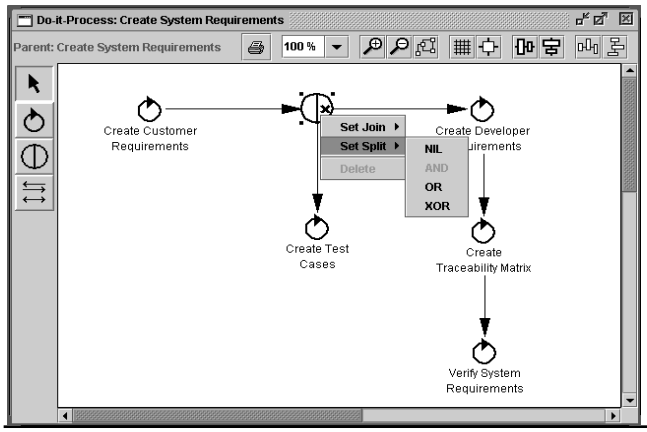
UNIVERSITETET I OSLO

Product-Flow View

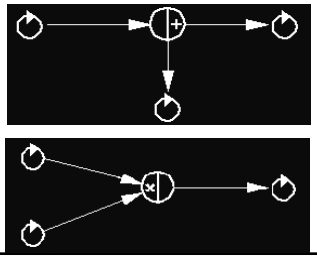


UNIVERSITETET I OSLO

Control-Flow View

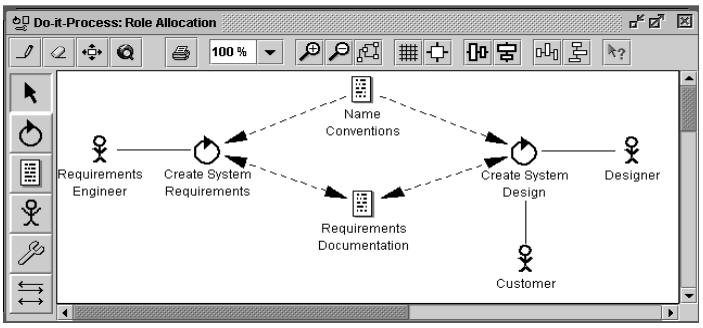


Join/Split symbols:
AND (x), OR (+), XOR(⊕)
nil (empty).



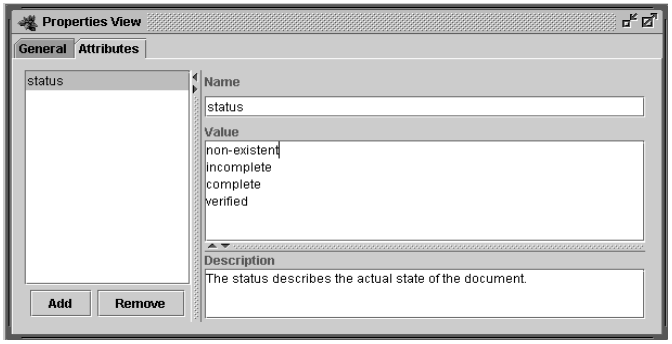
UNIVERSITETET I OSLO

Process View

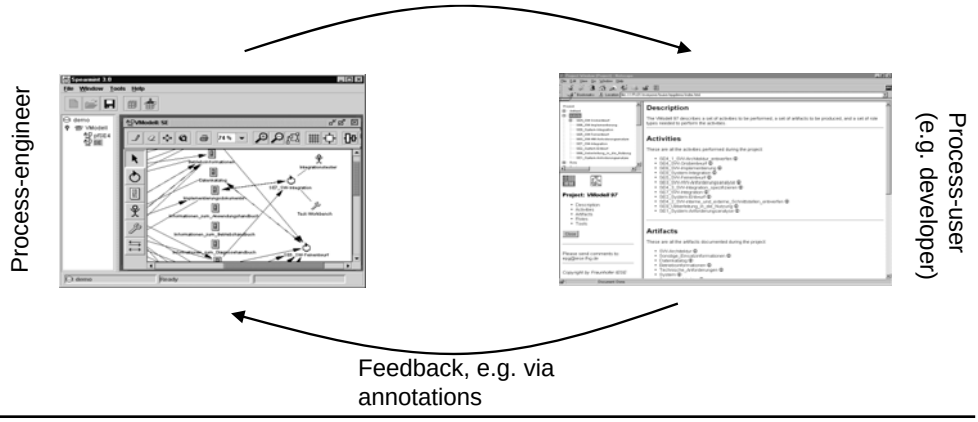


UNIVERSITETET I OSLO

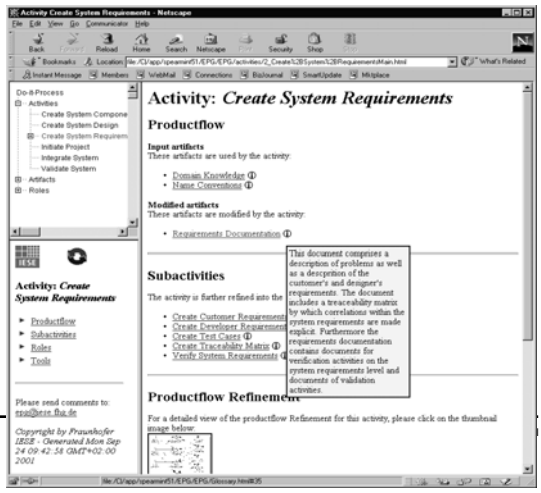
Attributes View



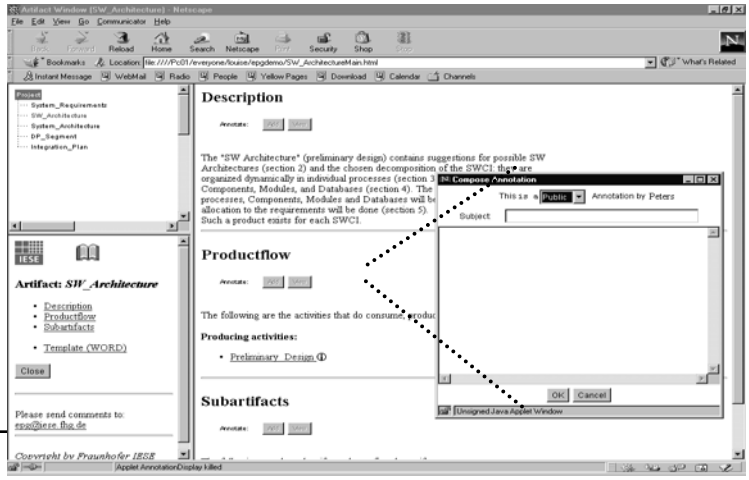
Generation of Hypertext View (EPG)



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