Chapter 2

Modeling the Process and Life Cycle

Shari L. Pfleeger
Joanne M. Atlee

4th Edition

Contents

2.1 The Meaning of Process
2.2 Software Process Models
2.3 Tools and Techniques for Process Modeling
2.4 Practical Process Modeling
2.5 Information System Example
2.6 Real Time Example
2.7 What this Chapter Means for You
Chapter 2 Objectives

• What we mean by a “process”
• Software development products, processes, and resources
• Several models of the software development process
• Tools and techniques for process modeling

2.1 The Meaning of Process

• A **process**: a series of steps involving activities, constraints, and resources that produce an intended output of some kind

• A process involves a set of tools and techniques
2.1 The Meaning of Process

Process Characteristics

• Prescribes all major process activities
• Input:
  – Uses resources (e.g., customer input, specifications),
  – subject to set of constraints (such as schedule, platform reqts)
• Output: Produces intermediate and final products (e.g., models)
• Structure:
  – May be composed of subprocesses
  – with hierarchy or links
• Properties:
  – Each process activity has entry and exit criteria
  – Activities are organized in sequence, so timing is clear
  – Each process guiding principles, including goals of each activity
  – Constraints may apply to an activity, resource or product

2.1 The Meaning of Process

The Importance of Processes

• Impose consistency and structure on a set of activities

• Guide us to understand, control, examine, and improve the activities

• Enable us to capture our experiences and pass them along
2.2 Software Process Models

Reasons for Modeling a Process

- To form a common understanding
- To find inconsistencies, redundancies, omissions
- To find and evaluate appropriate activities for reaching process goals
- To tailor a general process for a particular situation in which it will be used

Software Life Cycle

- When a process involves building a software, the process may be referred to as software life cycle
  - Requirements analysis and definition
  - System (architecture) design
  - Program (detailed/procedural) design
  - Writing programs (coding/implementation)
  - Testing: unit, integration, system
  - System delivery (deployment)
  - Maintenance
2.2 Software Process Models

Software Development Process Models

- Waterfall model
- V model
- Prototyping model
- Operational specification
- Transformational model
- Phased development: increments and iteration
- Spiral model
- Agile methods

Waterfall Model

- One of the first process development models proposed
- Works for well understood problems with minimal or no changes in the requirements
- Simple and easy to explain to customers
- It presents
  - a very high-level view of the development process
  - sequence of process activities
- Each major phase is marked by milestones and deliverables (artifacts)
2.2 Software Process Models
Waterfall Model (continued)

- There is no iteration in waterfall model
- Most software developments apply a great many iterations
2.2 Software Process Models

Sidebar 2.1 Drawbacks of The Waterfall Model

• Provides no guidance how to handle changes to products and activities during development (assumes requirements can be frozen)
• Views software development as manufacturing process rather than as creative process
• There is no iterative activities that lead to creating a final product
• Long wait before a final product

2.2 Software Process Models

Waterfall Model with Prototype

• Different types of prototype:
  – Requirements: User interface shell to illustrate observable behavior (i.e., used to elicit reqts info)
  – Design: A prototype can be a partially developed product (black box design details and/or components)

• Prototyping helps
  – users understand what the system will be like (user interface prototype)
  – developers assess alternative design strategies (design prototype)

• Prototyping is useful for verification and validation
2.2 Software Process Models

Waterfall Model with Prototype (continued)

- Waterfall model with prototyping

2.2 Software Process Models

V Model (continued)
2.2 Software Process Models

V Model

• A variation of the waterfall model
• Uses unit testing to verify procedural design
• Uses integration testing to verify architectural (system) design
• Uses acceptance testing to validate the requirements
• If problems are found during verification and validation, the left side of the V can be re-executed before testing on the right side is re-enacted
2.2 Software Process Models

Prototyping Model

- Allows repeated investigation of the requirements or design
- Reduces risk and uncertainty in the development

Operational Specification Model

- Requirements are executed (examined) and their implication evaluated early in the development process
- Functionality and the design are allowed to be merged
2.2 Software Process Models
Transformational Model

- Fewer major development steps
- Applies a series of transformations
  - Change data representation
  - Select algorithms
  - Optimize
  - Compile
- Relies on formalism
- Requires formal specification (to allow transformations)

[Diagram of transformational model]
2.2 Software Process Models
Phased Development: Increments and Iterations

- Shorter cycle time
- System delivered in pieces
  - enables customers to have some functionality while rest is being developed
- Allows two systems functioning in parallel
  - the production system (release n): currently being used
  - the development system (release n+1): the next version
2.2 Software Process Models
Phased Development: Increments and Iterations (continued)

- **Incremental development**: starts with small functional subsystem and adds functionality with each new release
- **Iterative development**: starts with full system, then changes functionality of each subsystem with each new release
2.2 Software Process Models
Phased Development: Increments and Iterations
(continued)

• Phased development is desirable for several reasons
  – Training can begin early, even though some functions are missing
  – Markets can be created early for functionality that has never before been offered
  – Frequent releases allow developers to fix unanticipated problems globally and quickly
  – The development team can focus on different areas of expertise with different releases

2.2 Software Process Models
Spiral Model

• Suggested by Barry Boehm (1988)
• Combines development activities with risk management to minimize and control risks
• The model is presented as a spiral in which each iteration is represented by a circuit around four major activities
  – Plan
  – Determine goals, alternatives and constraints
  – Evaluate alternatives and risks
  – Develop and test
2.2 Software Process Models

Spiral Model (continued)

Agile Methods

- Emphasis on flexibility in producing software quickly and capably
- **Agile manifesto**
  - Value individuals and interactions over process and tools
  - Prefer to invest time in producing working software rather than in producing comprehensive documentation
  - Focus on customer collaboration rather than contract negotiation
  - Concentrate on responding to change rather than on creating a plan and then following it
2.2 Software Process Models
Agile Methods: Examples of Agile Process

- Extreme programming (XP)
- Crystal: a collection of approaches based on the notion that every project needs a unique set of policies and conventions
- Scrum: 30–day iterations; multiple self-organizing teams; daily “scrum” coordination
- Adaptive software development (ASD)

2.2 Software Process Models
Agile Methods: Extreme Programming

- Emphasis on four characteristics of agility
  - **Communication:** continual interchange between customers and developers
  - **Simplicity:** select the simplest design or implementation
  - **Courage:** commitment to delivering functionality early and often
  - **Feedback:** loops built into the various activities during the development process
2.2 Software Process Models
Agile Methods: Twelve Facets of XP

- The planning game (customer defines value)
- Small release
- Metaphor (common vision, common names)
- Simple design
- Writing tests first
- Refactoring
- Pair programming
- Collective ownership
- Continuous integration (small increments)
- Sustainable pace (40 hours/week)
- On-site customer
- Coding standard

Sidebar 2.2 When Extreme is Too Extreme?

- Extreme programming's practices are interdependent
  - A vulnerability if one of them is modified
- Requirements expressed as a set of test cases must be passed by the software
  - System passes the tests but is not what the customer is paying for
- Refactoring issue
  - Difficult to rework a system without degrading its architecture
2.2 Software Process Models
Sidebar 2.3 Collections of Process Models

- Development process is a problem-solving activity
- Curtis, Krasner, and Iscoe (1988) performed a field study to determine which problem-solving factors to capture in process model
- The results suggest a layered behavioral model as supplement to the traditional model
- Process model should not only describe series of tasks, but also should detail factors that contribute to a project's inherent uncertainty and risk

2.7 What this Chapter Means for You

- Process development involves activities, resources, and product
- Process model includes organizational, functional, behavioral and other perspectives
- A process model is useful for guiding team behavior, coordination and collaboration
2.3 Tools and Techniques for Process Modeling

- Notation depends on what we want to capture in the model
- The two major notation categories
  - Static model: depicts the process
  - Dynamic model: enacts the process
2.3 Tools and Techniques for Process Modeling

Static Modeling: Lai Notation

- Element of a process are viewed in terms of seven types
  - Activity
  - Sequence
  - Process model
  - Resource
  - Control
  - Policy
  - Organization
- Several templates, such as an Artifact Definition Template
2.3 Tools and Techniques for Process Modeling
Static Modeling: Lai Notation (continued)

• The process of starting a car

2.3 Tools and Techniques for Process Modeling
Static Modeling: Lai Notation (continued)

• Transition diagram illustrates the transition for a car
2.3 Tools and Techniques for Process Modeling

Dynamic Modeling

• Enables enaction of process to see what happens to resources and artifacts as activities occur
• Simulate alternatives and make changes to improve the process
• Example: systems dynamics model

2.3 Tools and Techniques for Process Modeling

Dynamic Modeling: System Dynamics

• Introduced by Forrester in the 1950's
• Abdel–Hamid and Madnick applied it to software development
• One way to understand system dynamics is by exploring how software development process affects productivity
2.3 Tools and Techniques for Process Modeling
Dynamic Modeling: System Dynamics (continued)

- Pictorial presentation of factors affecting productivity
- Arrows indicate how changes in one factor change another

A system dynamic model containing four major areas affecting productivity
2.3 Tools and Techniques for Process Modeling

Sidebar 2.4 Process Programming

• A program to describe and enact the process
  – Eliminate uncertainty
  – Basis of an automated environment to produce software
• Does not capture inherent variability of underlying development process
  – Implementation environment, skill, experience, understanding the customer needs
• Provides only sequence of tasks
• Gives no warning of impending problems

2.4 Practical Process Modeling

Marvel Case Studies

• Uses Marvel process language (MPL)
• Three constructs: classes, rules, tool envelopes
• Three-part process description
  – rule-based specification of process behavior
  – object-oriented definition of model’s information process
  – set of envelopes to interface between Marvel and external software tools
2.4 Practical Process Modeling
Marvel Case Studies (continued)

- Involved two AT&T networks
  - network carried phone calls
  - signaling network responsible for routing calls and balancing the network load
- Marvel was used to describe the signaling fault resolution

![Signaling Fault Resolution Process Diagram]
2.4 Practical Process Modeling

Example of Marvel Commands

```
TICKET: superclass ENTITY
    status : (initial, open, referred_out, referral_clone,
               closed, fixed) = initial;
diagnostics : (terminal, non_terminal, none) = none;
level : integer;
description : text;
referred_to : link WORKCENTER;
referrals : set_of link TICKET;
process : link PROC_INST;
end

diagnose [?t: TICKET];
    (exists PROC_INST ?p suchthat (linkto [?t:process ?p]))
    :
    (and (?t:status = open)(?t:diagnostics = none))
    {TICKET_UTIL diagnose ?t:Name}
    (and (?t:diagnostics = terminal)
      (?p:next_task = refer_to_WC3)
      (?p:process = referral)
    {TICKET_UTIL diagnose ?t:Name}
    (and (?t:diagnostics = non_terminal)
      (?p:next_task = diagnosis)
      (?p:process = diagnosis)
    {TICKET_UTIL diagnose ?t:Name}
    (and (?t:diagnostics = none)
      (?p:next_task = refer_to_WC2)
    {TICKET_UTIL diagnose ?t:Name}
```

Desirable Properties of Process Modeling Tools and Techniques

- Facilitates human understanding and communication
- Supports process improvement
- Supports process management
- Provides automated guidance in performing the process
- Supports automated process execution
2.5. Information System Example
Piccadilly Television Advertising System

- Needs a system that is easily maintained and changed
- Requirements may change
  - Waterfall model is not applicable
- User interface prototyping is an advantage
- There is uncertainty in regulation and business constraints
  - Need to manage risks
- Spiral model is the most appropriate

2.5. Information System Example
Piccadilly System (continued)

- Risk can be viewed in terms of two facets
  - Probability: the likelihood a particular problem may occur
  - Severity: the impact it will have on the system
- To manage risk, it needs to include characterization of risks in the process model
  - Risk is an artifact that needs to be described
### 2.5. Information System Example

Lai Artifact Table for Piccadilly System

<table>
<thead>
<tr>
<th>Name</th>
<th>Risk (problemX)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Synopsis</td>
<td>This is the artifact that represents the risk that problem X will occur and have a negative effect on some aspect of the development process.</td>
</tr>
<tr>
<td>Complexity type</td>
<td>Composite</td>
</tr>
<tr>
<td>Risk-state list</td>
<td>(risk.x, user defined)</td>
</tr>
</tbody>
</table>

| low | low | The probability that problem X will occur. |
|----|----|----------------|---|
| high-medium | high | The severity of the impact should problem X occur on the project. |
| high-medium | high | The severity of the impact should problem X occur on the project. |
| high | high | The severity of the impact should problem X occur on the project. |

---

### 2.6 Real Time Example

Ariane–5 Software

- Involved reuse of software from Ariane–4
- The reuse process model
  - Identify resuable subprocesses, describe them and place them in a library
  - Examine the requirements for the new software and the reusable components from library and produce revised set of requirements
  - Use the revised requirements to design the software
  - Evaluate all reused design components to certify the correctness and consistency
  - Build or change the software
2.6 Real Time Example
Ariane–5 Software (continued)

- Reuse process model presentation

2.7 What this Chapter Means for You

- Process development involves activities, resources, and product
- Process model includes organizational, functional, behavioral and other perspectives
- A process model is useful for guiding team behavior, coordination and collaboration