Process Modeling

Bill Curtis, Marc I. Kellner and Jim Over

Communications of the ACM

2007. 5. 30
Chanhee Yi
Contents

- Introduction
- Overview
- Conceptual framework
- Modeling paradigm
- Modeling issues
- Future growth
- Conclusion
Introduction (1/2)

- Process modeling
  - Brief Definition
    - Process
    - Modeling
  - Motivation of emergence
    - Problems in traditional software description
      - Ambiguous, unusable descriptions
      - Failure to update the documentation with change
    - Needs from the enterprise
      - Business process reengineering
      - Coordination technology
      - Process-driven software development
Process modeling (cont’d)

- Goals
  - Human understanding
  - Process improvement
  - Process management
  - Automated execution
  - Automated guidance
Overview

Perspectives

Process modeling

Conceptual framework

Modeling Paradigm

Formality

Granularity

Scriptiveness

Modeling Issues

Terminology

Future Directions

?
Conceptual framework (1/2)

● Terminology
  – Process element
    ● Any component of a process
  – Process step
    ● Atomic action of a process
  – Agent
    ● An actor who performs a process element
  – Role
    ● A set of process elements assigned to an actor
  – Artifact
    ● Product created or modified by process element execution
Conceptual framework (2/2)

- Perspectives in process representation
  - Functional
    - What elements are performed
  - Behavioral
    - When and how elements are performed
  - Organizational
    - Where and by whom elements are performed
  - Informational
    - Represent informational entities
Modeling paradigm (1/7)

- Representation with software languages
  - General start point for process modeling
  - Means for representing and enacting a process
  - Resource of process evolution through feedback
Modeling paradigm (2/7)

- **Programming models**
  - View of regarding a process as programming
  - REBUS
    - Support for the requirement specification

```plaintext
Procedure SingleUserModel is ...
Begin
  while not Done loop
    case CurrentApproach is
      when TopDown =>
        if NeedRoot then CreateRoot;
        elsif NeedChildren then
          SelectNodeToElaborate(CurrentNode); CreateChildren(CurrentNode);
        else SelectNodeToEdit(CurrentNode); EditNode(CurrentNode);
      when BottomUp =>
        if NeedLeaves then CreateLeaf;
        end if;
        :
        :
```
Modeling paradigm (3/7)

- **Functional models**
  - Hierarchical and Functional Software Process (HFSP)
    - Hierarchical process decomposition
    - Concurrent processes and backtracking
    - Flexible and dynamic execution mechanism
Modeling paradigm (4/7)

- **Functional models (cont’d)**

  ```
  activity
  JSP( spec | prog.out ) =>
  MakeProgTree( spec | progTree )
  EnumerateOprs( spec | oprs )
  .
  .
  ```

Diagram:

- JSP
- `spec` `prog`
- `MakeProgTree` `spec` `progTree`
- `EnumerateOprs` `spec` `oprs`
- `MakeProg` `prog` `oprs` `progTree`
- `DoProgInversion` `prog`

2007-05-30

Software Engineering Lab, KAIST
Modeling paradigm (5/7)

- **Plan-based models**
  - Artificial intelligence approach to contingency
  - GRAPPLE
    - Constraint-based planning system
    - Plan execution simulation system

OPERATOR: release
GOAL: current-release(System)
PRECONDS: built(System)
          regression-tested(System)
          performance-tested(System)
CONSTR: current-release(C)
         not-equal(C,System)
EFFECTS: DELETE current-release(C)
         ADD current-release(System)
         ADD customer-release(System)
         ADD prior-release(System,C)

2007-05-30
Modeling paradigm (6/7)

- **Petri-net models**
  - Coordination of the model’s structure of roles
  - Role interaction nets
    - Method of coordinating the routing of artifacts
    - Method of tracking progress
Quantitative models

- **System dynamics**
  - Application of feedback and control to social phenomena

\[
\frac{d}{dt} L(t) = -R(t) = -\frac{L(t)}{T}
\]

\[
L(t) = L(0) e^{-\frac{t}{T}}
\]

- **Perceived Work Accomplished (PWA)**
  - \(VF \times (\% \text{ of Resource remained}) + (1-VF) \times (\% \text{ of Work finished})\)
  - VF : Visibility Factor
Modeling issues (1/2)

- **Formality**
  - Specify Mathematical level of modeling languages
  - Considered to control flexibility of task representation

- **Granularity**
  - Consider precision with time and human
  - Considered to provide multiple level of abstraction

- **Precision**
  - Represents the degree of specifying process steps
Modeling issues (2/2)

- **Scriptiveness**
  - involves prescriptive modeling
    - Planned modeling
  - Involves descriptive modeling
    - Evolutionary modeling
  - Involves proscriptive modeling
    - Law-governed modeling

- **Fitness**
  - Represents the degree of agent’s fulfillment to steps
  - Used to analyze prescriptive models
Future directions (1/6)

- **Multi-paradigm representations**
  - Emphasis on diversity and suitability of a modeling
  - STATEMATE

---

**DEV_MODULE_TESTS**

- **description**: Task of planning and developing unit tests for a module.
- **executed throughout**: DEVELOP_TESTS
- **carried out by**: QA
- **attributes**:
  - **NAME**: VALUE:
  - USES_FACILITY: DVLPMT_SUITE
  - HAND_CARRIED_2
- **Contains**: CODE_LISTINGS: TESTPlans

---

**Interconnections Textual Information**

2007-05-30

Software Engineering Lab, KAIST
Future directions (2/6)

- Multi-paradigm representations (cont’d)

[State chart diagram]

Behavioral Perspective - State chart

Organizational Perspective - Module Chart
Future directions (3/6)

- **Process improvement**
  - Effects of a well-defined process
    - Results of documentation and person training
    - Foundation for quality and productivity
  - Capability Maturity Model (CMM)
    - The way for organization to improve schedules and budget
    - Identification to the current maturity of process
    - Improvement on top of documents
Future directions (4/6)

- Process improvement (cont’d)

- Level 1: Initial
  - Inconsistent management
  - Repeatability

- Level 2: Repeatable
  - Project management
  - Common engineering processes

- Level 3: Defined
  - Process management
  - Quantitative understanding and control

- Level 4: Quantitatively Managed
  - Capability management
  - Continuous improvement practices

- Level 5: Optimizing
  - Change management

2007-05-30
Software Engineering Lab, KAIST
Future directions (5/6)

- Process-based software development
  - Automated execution support and guidance
    - Tools that support the entire process
    - Tool integration with environment
    - Design for work coordination
Future directions (6/6)

- Process-based software development (cont’d)
  - MARVEL
    - Rule-based approach
    - Backward-chaining and forward-chaining
    - Emphasis on functional and behavioral perspectives

```
strategy doc_tools_rules
Imports none;
Exports all;
Objectbase
DOC_HEAD :: superclass TOOL:
  doc_head : string = specify_header;
End
End_objectbase

rules

  doc_head[?d : DOC_TYPES] :
suchthat :
  { DOC_HEAD doc_head ?d }
  (?d.head_created = HeadFileExists);
  (?d.head_created = HeadFileNonExists);
  .
  .
```

2007-05-30
Conclusion

● Guidance to process modeling
  – Concepts and perspectives in representation
  – Five paradigms as an approach
  – Future directions with field growth
<table>
<thead>
<tr>
<th>Perspectives</th>
<th>Functional</th>
<th>Behavioral</th>
<th>Organizational</th>
<th>Informational</th>
</tr>
</thead>
<tbody>
<tr>
<td>Procedural programming languages</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>System analysis and design</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>AI languages and approaches</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>Events and triggers</td>
<td>O</td>
<td></td>
<td></td>
<td>O</td>
</tr>
<tr>
<td>State transition and petri-nets</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>Control flow</td>
<td></td>
<td></td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>Functional languages</td>
<td>O</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Formal languages</td>
<td>O</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data modeling</td>
<td></td>
<td></td>
<td></td>
<td>O</td>
</tr>
<tr>
<td>Object modeling</td>
<td>O</td>
<td></td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>Precedence networks</td>
<td>O</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

2007-05-30

Software Engineering Lab, KAIST

24/16
Ada process programming language based on aspen

- Relations
  - Program unit that provides for the storage of data
- Triggers
  - Tasks that represent concurrent threads of control
- Predicates
  - Named boolean expressions over relations
STATEMATE

- Multiple language paradigms
  - State transitions with events and triggers
  - System analysis and design diagrams
  - Data-modeling

- Common denominator schema
  - Loose incorporation between various paradigms
  - Object-Oriented and entry-relationship schema
Common denominator schema

- Input Representations
  - one or more of various modeling approaches
  - used for development of process

- Translation / transformation to CDR: automated
  - direct input
  - translation

- Common Denominator Representation (CDR)
  - union of vital process information

- Translation / transformation from CDR: automated or semi-automated

- Output Representations
  - intended for presentation
  - one or more of various modeling approaches
  - may be documentation style
  - use for further process engineering
Petri-net

- Models of concurrency, non-determinism and control flow
- Modeling framework for discrete event dynamically systems

Sequential Execution

Concurrency

Synchronization