Software Development Methodologies

Lecturer: Raman Ramsin

Lecture 14

Process Metamodels
Process Metamodels

- Results of applying abstraction to software development processes

- Highlight the high-level features of a process or family of processes

- Can be instantiated in order to produce concrete processes

- The two most well-known object-oriented process metamodels:
  - OPEN Consortium’s OPEN Process Framework (OPF)
  - OMG’s Software and Systems Process Engineering Metamodel (SPEM 2.0)
OPEN Process Framework (OPF)

- A process metamodel defining five classes of components and guidelines for constructing customized OPEN processes
- Complemented by a component library from which process-component instances can be selected and assembled to create a specific process

For each element (represented by box), OPEN permits the user to select how many and which instances will be used. The OPF documentation provides a comprehensive list of suggestions on the best selections together with guidelines on their best organization.

[Fioresmith and Henderson-Sellers 2001]
OPF: Component Classes

- **Work Products**: any significant thing of value (document, diagram, model, class, application) developed during the project.

- **Languages**: the media used to document work products, such as natural languages, modeling languages such as UML or OML, and implementation languages such as Java, SQL, or CORBA-IDL.

- **Producers**: active entities (human or nonhuman) that develop the work products.

- **Work Units**: operations that are performed by producers when developing work products. One or more producers develop a *work product* during the execution of one or more *work units*.

- **Stages**: durations or points in time that provide a high-level organization to the work units.
OPF: Work Units

- **Activity:**
  - a major work unit consisting of a related collection of jobs that produce a set of work products
  - Coarse-grained descriptions of what needs to be done
  - Some important instances defined by OPEN are: Project Initiation, Requirements Engineering, Analysis and Model Refinement, Project Planning, and Build

- **Task:**
  - Smallest atomic unit of work
  - Small-scale jobs associated with and comprising the activities
  - Resulting in the creation, modification, or evaluation of one or more work products

- **Technique:**
  - Define how the jobs are to be done
  - Ways of doing the tasks and activities
Software Process Engineering Metamodell (SPEM 1.0)

- Similar in essence to OPF yet much simpler

- Primarily based on Rational Corporation’s *Unified Software Process Metamodell (USPM)*, which was chiefly intended as a metamodel for the RUP process

- Mainly supports the modeling of UML-based processes similar to RUP

- Unlike OPF, SPEM 1.0
  - does not include a process component library.
  - does not offer a specific procedure for instantiating a software development process using the metamodel (only well-formedness rules are provided).
SPEM 1.0: Core Structure

- Regards the core structure of a software development process as consisting of:
  - process roles
  - work products
  - activities

- Regards a software development process as
  - a collaboration of active entities (process roles)
    - aimed at performing specific operations (activities)
      - performed on a set of tangible artefacts (work products)
      - continued until the artefacts are brought to a well-defined state, and declared as complete.
SPEM 1.0: Core Structure

[OMG 2002]
SPEM 1.0: Detailed Structure

- **Work products:**
  - may be composed of other work products;
  - can be associated with a state machine.

- **Activities:**
  - can be partitioned into *disciplines* based on their common structural and functional themes;
  - may consist of atomic sub-activities called *steps*;
  - can have a *precondition* and a *goal* as constraints on its enactment;
  - may be associated with an *activity graph*, which shows the flow of steps in the activity.
SPEM 1.0: Lifecycle Definition

- SPEM incorporates definitions for
  - *Iteration*
  - *Phase*
  - *Lifecycle*

- Intended to constrain the order in which the activities are performed, and to define the lifecycle structure of the process

- Very similar to their corresponding definitions in RUP
Software and Systems Process Engineering Metamodel (SPEM 2.0)

- Adopted by OMG in December 2006, and revised in 2008
- Addresses the weaknesses of SPEM 1.0
- Provides necessary concepts for modeling, documenting, presenting, managing, interchanging, and enacting development methods and processes
  - Provides standardized representation and managed libraries of reusable method content
  - Supports systematic development, management, and growth of development processes
  - Supports deployment of method content and process needed by defining configurations of processes and method content
  - Supports enactment of process for development projects
SPEM 2.0: Conceptual Usage Framework

Standardize representation and manage libraries of reusable

**Method Content**

- Content on agile development
- Content on managing iterative development
- Guidance on serialized java beans

JUnit user guidance

Content on J2EE

Configuration mgmt guidelines

Develop and manage **Processes**
for performing projects

Lessons learnt from previous project and iteration

Process assets patterns

Corporate guidelines on compliance

Standard or reference processes

Project plan templates

**Configure** a cohesive process framework customized for my project needs

Create project plan templates for **Enactment** of process in the context of my project

[OMG 2008]
SPEM 2.0: Separation of Method Content from Development Process (1)

[OMG 2008]
SPERM 2.0: Separation of Method Content from Development Process (2)

[OMG 2008]

Department of Computer Engineering
**SPEM 2.0: Separation of Method Content from Development Process (3)**

- Method Content Element "Task Definition" referenced in more than one Process.
- Underlying technical concept to support reuse and smart customization: "Task Use".

Individual customization of a "Task Use" by selecting steps, providing additional documentation, etc.

[OMG 2008]
SPEM 2.0: Method Content - Elements

- Roles are responsible for work products
  - Each work product is the responsibility of a single role
- Process roles perform tasks
  - Each task is only performed by a single role
- Work products used as inputs to tasks and outputs from tasks
- “Somebody does something that changes something”
SPEM 2.0: Method Content - Guidance

- Can be associated with any process model element to provide more detailed information about the element to the practitioner
- Can standalone – does not have to be associated
- Most often associated with activities and work products
- SPEM comes with a set of built-in guidance types:
  - Checklist
  - Template
  - Example
  - Tool mentor
  - Guideline
SPEM 2.0: Process Components

- Allow the user to treat the actual definition of the work that produces the outputs as a "black box."
- Allow different styles or techniques of doing work to be replaced with others.

[OMG 2008]

Department of Computer Engineering
SPEM 2.0: Process Patterns

<table>
<thead>
<tr>
<th>Presentation Name</th>
<th>Index</th>
<th>Model Info</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Early Project Phase</td>
<td>0</td>
<td></td>
<td>Delivery Process</td>
</tr>
<tr>
<td>Plan an Iteration</td>
<td>1</td>
<td></td>
<td>Phase</td>
</tr>
<tr>
<td>Design Work</td>
<td>1</td>
<td></td>
<td>Activity</td>
</tr>
<tr>
<td>Create a Scenario</td>
<td>2</td>
<td>extends 'Plan an Iteration, msf-agile'</td>
<td>Activity</td>
</tr>
<tr>
<td>Create Solution Architecture</td>
<td>3</td>
<td>extends 'Design Work, msf-agile'</td>
<td>Activity</td>
</tr>
<tr>
<td>Partition the System</td>
<td>4</td>
<td></td>
<td>Task Descriptor</td>
</tr>
<tr>
<td>Determine Interfaces</td>
<td>5</td>
<td></td>
<td>Task Descriptor</td>
</tr>
<tr>
<td>Develop Threat Model</td>
<td>6</td>
<td></td>
<td>Task Descriptor</td>
</tr>
<tr>
<td>Develop Performance Model</td>
<td>7</td>
<td></td>
<td>Task Descriptor</td>
</tr>
<tr>
<td>Create Architectural Prototype</td>
<td>8</td>
<td></td>
<td>Task Descriptor</td>
</tr>
<tr>
<td>Create Infrastructure Architecture</td>
<td>9</td>
<td></td>
<td>Task Descriptor</td>
</tr>
<tr>
<td>Intermediate Project Phase</td>
<td>10</td>
<td></td>
<td>Phase</td>
</tr>
<tr>
<td>Plan an Iteration</td>
<td>11</td>
<td>extends 'Plan an Iteration, msf-agile'</td>
<td>Activity</td>
</tr>
<tr>
<td>Design Work</td>
<td>12</td>
<td>extends 'Design Work, msf-agile'</td>
<td>Activity</td>
</tr>
<tr>
<td>Create a Scenario</td>
<td>13</td>
<td>extends 'Create a Scenario, msf-agile'</td>
<td>Activity</td>
</tr>
<tr>
<td>Create Solution Architecture</td>
<td>14</td>
<td>extends 'Create Solution Architecture, msf-agile'</td>
<td>Activity</td>
</tr>
<tr>
<td>Partition the System</td>
<td>15</td>
<td></td>
<td>Task Descriptor</td>
</tr>
<tr>
<td>Determine Interfaces</td>
<td>16</td>
<td></td>
<td>Task Descriptor</td>
</tr>
<tr>
<td>Develop Threat Model</td>
<td>17</td>
<td></td>
<td>Task Descriptor</td>
</tr>
<tr>
<td>Develop Performance Model</td>
<td>18</td>
<td></td>
<td>Task Descriptor</td>
</tr>
<tr>
<td>Create Architectural Prototype</td>
<td>19</td>
<td></td>
<td>Task Descriptor</td>
</tr>
<tr>
<td>Create Infrastructure Architecture</td>
<td>20</td>
<td></td>
<td>Task Descriptor</td>
</tr>
<tr>
<td>Late Project Phase</td>
<td>21</td>
<td></td>
<td>Phase</td>
</tr>
<tr>
<td>Plan an Iteration</td>
<td>22</td>
<td></td>
<td>Activity</td>
</tr>
<tr>
<td>Design Work</td>
<td>23</td>
<td>extends 'Plan an Iteration, msf-agile'</td>
<td>Activity</td>
</tr>
<tr>
<td>Create a Scenario</td>
<td>24</td>
<td>extends 'Design Work, msf-agile'</td>
<td>Activity</td>
</tr>
<tr>
<td>Create Solution Architecture</td>
<td>25</td>
<td>extends 'Create a Scenario, msf-agile'</td>
<td>Activity</td>
</tr>
<tr>
<td>Partition the System</td>
<td>26</td>
<td></td>
<td>Task Descriptor</td>
</tr>
<tr>
<td>Determine Interfaces</td>
<td>27</td>
<td></td>
<td>Task Descriptor</td>
</tr>
<tr>
<td>Develop Threat Model</td>
<td>28</td>
<td></td>
<td>Task Descriptor</td>
</tr>
<tr>
<td>Develop Performance Model</td>
<td>29</td>
<td></td>
<td>Task Descriptor</td>
</tr>
<tr>
<td>Create Architectural Prototype</td>
<td>30</td>
<td></td>
<td>Task Descriptor</td>
</tr>
<tr>
<td>Create Infrastructure Architecture</td>
<td>31</td>
<td></td>
<td>Task Descriptor</td>
</tr>
<tr>
<td>Late Project Phase</td>
<td>32</td>
<td></td>
<td>Phase</td>
</tr>
<tr>
<td>Plan an Iteration</td>
<td>33</td>
<td>extends 'Plan an Iteration, msf-agile'</td>
<td>Activity</td>
</tr>
<tr>
<td>Design Work</td>
<td>34</td>
<td>extends 'Design Work, msf-agile'</td>
<td>Activity</td>
</tr>
<tr>
<td>Create a Scenario</td>
<td>35</td>
<td>extends 'Create a Scenario, msf-agile'</td>
<td>Activity</td>
</tr>
<tr>
<td>Create Solution Architecture</td>
<td>36</td>
<td>extends 'Create Solution Architecture, msf-agile'</td>
<td>Activity</td>
</tr>
<tr>
<td>Partition the System</td>
<td>37</td>
<td></td>
<td>Task Descriptor</td>
</tr>
<tr>
<td>Determine Interfaces</td>
<td>38</td>
<td></td>
<td>Task Descriptor</td>
</tr>
<tr>
<td>Develop Threat Model</td>
<td>39</td>
<td></td>
<td>Task Descriptor</td>
</tr>
<tr>
<td>Develop Performance Model</td>
<td>40</td>
<td></td>
<td>Task Descriptor</td>
</tr>
<tr>
<td>Create Architectural Prototype</td>
<td>41</td>
<td></td>
<td>Task Descriptor</td>
</tr>
<tr>
<td>Create Infrastructure Architecture</td>
<td>42</td>
<td></td>
<td>Task Descriptor</td>
</tr>
</tbody>
</table>

Revisiting design work based on same underlying pattern

Dynamic linking of patterns increases maintainability

Changes in patterns require zero updates

[OMG 2008]
SPEM 2.0: Modeling Enactable Processes

Role, Tasks, and Work Products

Activity Diagram

Work Breakdown Structures

[OMG 2008]
References

