INF5120 – Model-Based System Development

Lecture #7: Method engineering, SPEM and EPF

2 March 2009
Brian Elvesæter, SINTEF ICT

Based on material developed in the ATHENA (IST-507849), COMBINE (IST-1999-20839), INTEROP (IST-508011), MODELWARE (IST-511731) and SHAPE (ICT-2007-216408) research projects, and the “An Introduction to EPF” presentation available at the EPF website http://www.eclipse.org/epf/general/An_Introduction_to_EPF.zip

Model-Driven Engineering (MDE) Framework

OMG MDA

Metamodelling

UML Profiles & DSLs

Model Transformations

Method Engineering

Reusable MDE Assets

• Method chunks
• Tools and services
• Models and metamodels
• Model transformations
• DSLs and UML profiles
• Reference examples
Outline

- Motivation – Why systems development methodology?
- Methodology
- Method Engineering (ME) and Process Framework (PF)
- Eclipse Process Framework (EPF)
- Software Process Engineering Metamodel (SPEM)
- EPF Composer
- References

Motivation – Why system development methodology?
Challenges for system developers

- Interoperability
- Increasing complexity
- Effectiveness (shorter time to market)
- Increasing quality requirements
- Understand the market needs
- Flexibility
- Technology independence
- Maintainability

Challenges in system development

- Reduce the risks of
  - Develop inappropriate systems
  - Exceed budget
  - Exceed time (too late to market)

- Want to
  - Develop maintainable systems
  - Reuse
  - Develop consistent systems
  - Modularise
  - Interoperate with existing/other systems
It is important to define the scope of a development. Otherwise a good answer may be given, but to the wrong question.

- Scope: to the delivery point?
- Or to the “end of life” for the system?
Solution

- Model-based software development methodology
  - Precise & standardized notation
  - Well-defined process
  - Well-defined deliverables
  - Well-defined responsibilities
  - Improved communication
  - Documentation
  - Knowledge maintenance
- Project management
- Tool support
- QA
- ...

Model-based system development

- A model is an abstraction
- A model is never complete
  - Focus on certain aspects
- A model is created for a purpose
  - Describe a real world phenomena
  - Describe a new phenomena (not existing, not yet existing)
Methodology

Methodology definition (1/2)

- Method
  - Systematic process, technique or mode of inquiry that is used to aid in the creation of a satisfactory software product. [Blum94]
  - Use a method to produce models
- Technique
  - A specific construct supporting a method
- Process
  - A sequence of actions leading to some result
- Method include technique and process
  - CRC method includes CRC technique and CRC process
- Methodology
  - Body of methods
  - Meant to support all software development phases
Methodology definition (2/2)

Method 1
Method 2
...
Method n

Methodological process

Underlying concepts (paradigm)
E.g. service-oriented software development

Role of the software process

The software process ties people and technology together to develop software products in a specific environment.
9 principals for modern software development

1. Architecture-centric
   - Service-oriented architecture
2. Iterative and incremental process
3. Service- and component-orientation
4. Manage a highly dynamic environment
   - Process: Iterative and incremental
   - Software system: Easy to change
5. Model-based
6. “Round-trip” engineering
7. Divide and conquer
8. Quality check
9. Configurable process

How can we describe increments, iterations, and spirals using linear calendar time?

- Givens:
  - Developer time vs. Project Manager time
  - The “Validation V”
- Techniques:
  - V’s for increments & iterations
  - V-W staging
  - Managing by milestones and reviews
Different process for different type of project

- Process depends on type of system
  - Brand new system (very rare)
  - Reengineering (old system exist)
  - Modification (fixing a major problem)
  - Adding a new module (functionality)

Problem (1): Developers do not live in linear time

- Artifacts grow fast, slow, even backwards!
- Spiral is a good word.

("Developer's time")
Problem (2): Project managers DO live in linear time

- The calendar does not show spirals.
- Time moves inexorably forward...

(“Project Manager’s time”)

How do you schedule a Gestalt Round Trip?

The ‘Validation V’: Not waterfall, but a fact of life

Req'ts
Design
Code

Validate req'ts
Validate logic
Validate syntax

How can we use this ‘fact of life’?
#1 24-month V is “waterfall”
#6 4-month Vs are incremental, effective

Vs fit within Vs for increments / iterations, and to manage visibility
Every V is a major milestone for the Project Manager’s calendar

The V-W model lets project managers put increments / iterations on the calendar

- What this model does NOT do:
  - Does not make project managers smarter
  - Does not improve your first estimate
  - Does not guarantee project delivered on time

- What this model DOES do:
  - Gives a vocabulary to what we know is effective, but hard to describe.
  - Lets project managers SCHEDULE iterations
  - Let different people develop differently
Methodology gives the Who, What, When of key interactions between people

- **Quality**: Precision, Accuracy, Tolerance
- **Regression tests**, Object model, Project plan, Use cases
- **Planning**: Staging, Testing
- **Activities**: Workshops, Use cases, CRC cards
- **Techniques**: Use cases, CRC cards
- **Standards**: Java Project, 3 month increments, UML, JEE
- **Tools**: Project manager, Documenter, Designer, Tester
- **Personality**: Skills
- **Teams**: Roles

Project Lifecycle

- Roles: Designer/Programmer, Writer, Tester, UI Expert, Lead Designer, Business Expert, Expert User, Project Manager, Project Sponsor, Trainer, Secretary, Contractor, Night Watchman, Janitor
- Activities: Rest and recreation, Vacations and basic business, Technical education, Time tracking, Project development
Lighter methodologies are more effective, but have limits

Some popular methodologies

- UP (RUP)
  - (Unified Process, Rational unified process)
- KOBRA
- Catalysis
- Select Perspective
- OOram
  - Object Oriented Role Analysis and Modelling
- Lightweight methodologies
  - XP
  - Adaptive Software Development (ASD)
  - SCRUM
  - Crystal Clear
Method Engineering (ME) and Process Framework (PF)

From the engineering perspective, a method is made up of a set of product models and a set of corresponding process models.

A product model represents the concepts that are used in the method, relationships between these concepts as well as constraints that they have to satisfy.

A process model represents the way to accomplish the development of the corresponding product.

Method engineering process

I. Reengineering of methods into method chunks

II. Assembly-based Situational Method Construction

Method chunk

A method chunk is an autonomous and coherent part of a method supporting the realisation of some specific system development or management activity. Such a modular view of methods favours their adaptation, configuration and extension. Moreover, this view permits to reuse chunks of a given method in the construction of new ones.
The responsibility of the method engineer is to identify and orchestrate the activities needed in the MDD system development process.

Based on the organisation’s specific needs, the method engineer selects the different process elements, from different process frameworks, and defines an appropriate system development process for an organisation.

The method engineer must ensure completeness of the defined system development process, for example that roles are coherent with roles existing within the organisation and so on.
CIM-PIM-PSM method components with models/tools

Objecteering SOA (OSOA)

Method components

<table>
<thead>
<tr>
<th>Information</th>
<th>Service</th>
<th>Process</th>
<th>Rules</th>
<th>Events</th>
<th>Organization</th>
<th>Goals</th>
<th>NFA</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIM OSOA-x</td>
<td>OSOA-x</td>
<td>OSOA-x</td>
<td>OSOA-x</td>
<td>OSOA-x</td>
<td>OSOA-x</td>
<td>OSOA-x</td>
<td>OSOA-x</td>
</tr>
<tr>
<td>PIM OSOA-x</td>
<td>OSOA-x</td>
<td>OSOA-x</td>
<td>OSOA-x</td>
<td>OSOA-x</td>
<td>OSOA-x</td>
<td>OSOA-x</td>
<td>OSOA-x</td>
</tr>
<tr>
<td>(Cl.diag)</td>
<td>(SoaML)</td>
<td>(BPMN,</td>
<td>Activ.d)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PIM Objecteering tool</td>
<td>Objecteering tool</td>
<td>Objecteering tool</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Java, XML</td>
<td>Java, WSDL</td>
<td>Java, BPEL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
SHAPE Methodology Framework

Methodology tool architecture
Custom Methodology Creation

1. Define overall engineering process
   - Discipline level (process of disciplines)
   - Tool support:
     - Suggest possible order
     - Validation (wrt references processes)

2. Refine engineering process
   - Define tasks for each discipline
   - Tool support:
     - Suggest candidate tasks

3. Select actual Methods
   - Chose methods for each task
   - Tool support:
     - Suggest candidates suggestion & validation
     - Validation (wrt notation / result / engin. process)

Representation in EPF
Eclipse Process Framework (EPF)
http://www.eclipse.org/epf/

The EPF Project: Overview

- EPF is an Open Source project within the Eclipse Foundation

The goals of EPF are to provide:
- An extensible framework and tooling for authoring, configuring and publishing processes
- Exemplary processes - first delivered is OpenUP

- EPF Project initiated in January 2006.

- EPF is NOT:
  - Only applicable for Eclipse Java development.
  - Intended to create the "perfect process"
The EPF Project: Two Audiences

- Process Authors and Coaches (Process Management Team)
  - Tooling for creating and publishing processes
  - Foundational process for starting point
  - Libraries of additional content that can be plugged-in

- Process Consumers (Project Team)
  - Published website of process content for simple browsing
  - Guidance in the form of checklists, concepts, guidelines
  - Browse the content adapted to your experience level

What development teams are facing today

- No common language or terminology between processes – redundancy and inconsistencies.
- Knowledge cannot easily be customized for different projects or new best practices
- No central community or communication framework to facilitate convergence of best practices across domains.

A better approach

Standardize representation and manage libraries of reusable method content

Develop and manage processes for performing projects

Cohesive Web site customized for my project needs

Project plan templates and optional process instrumentation, relevant for the context of my project

Project goals

- Provide an extensible framework and exemplary tools and content for software process engineering
  - Extensible framework
    - Metamodel based on OMG SPEM
    - Core extensible process tooling framework
  - Exemplary and extensible tools
    - Method and Process authoring
    - Library management and content extensibility
    - Configuring and publishing
  - Exemplary and extensible process content
    - Range of software development and management processes supporting
      - iterative, agile, and incremental development
      - applicable to a broad set of development platforms and applications
Some tools and services

- **Method Authoring**
  - Best practices can be captured as a set of reusable method building blocks as defined in the meta-model: roles, work products, tasks, and guidance, such as templates, guidelines, examples, and check lists.
  - A rich-text editor allows you to document method elements, and graphical views present diagrams showing relevant relationships.
  - Reuse is facilitated by allowing you to create a method element as a derivative of another method element through various inheritance-type of relationships.

- **Process Authoring**
  - Reusable process building blocks can be organized into processes along a lifecycle dimension by defining e.g. Work Breakdown Structures (WBSs), and when in the lifecycle to produce what work products in which state.
  - The tool allows you to construct reusable chunks of processes through so called capability patterns.
  - A capability pattern may for example define how to define, design, implement and test a scenario or a user story, and this pattern can now be reused in a variety of processes.

- **Library Management and Content Extensibility**
  - An XMI-based library enables persistence and flexible configuration management as well as content interchange for distributed client-server implementations.
  - Method and process content can be packaged into content plug-ins and content packages allowing simple distribution, management and extensibility of content.

- **Configuring and Publishing**
  - A process configuration can be created by selecting a set of content plug-ins and content packages.
  - Optionally, an exemplary process configuration can be used as a starting point, and content plug-ins and content packages added or removed from this exemplary configuration.
Software Engineering Process Metamodel (SPEM)

SPEM

- Software Process Engineering Metamodel
- Metamodel and UML profile to describe software engineering processes
  - Identifies the typical concepts of a process (process, phase, role, model, etc.)
  - Defines them using UML extensions (stereotypes applied to various elements: class, use cases, operations, etc.)
  - Assigns a characteristic icon to each new item.
- Although the title implies Software Processes, any process can be represented using SPEM.
- EPF uses SPEM.
Method Library

- Method Library
  - All Method Elements are stored in a Method Library
- Method Plug-in
  - A Method Plug-in represents a physical container for Method Packages and Process Packages. It defines a largest granularity level for the modularization and organization of method content and processes.
- Method Configuration
  - a logical subset of a Method Library
- Delivery Process
  - a complete and integrated approach for performing a specific type of project.

Method Content & Process

- Method Content (Who, What, Why, How)
  - Highly re-useable information
  - Definition of Roles, Tasks, Work Products and associated relationships
  - Includes Guidance and Categories
  - No timing information

- Process (When)
  - End-End sequence of Phases, Iterations, Activities and Milestones that define the development lifecycle.
  - Defines When tasks are performed via Activity Diagrams and/or Work Breakdown Structures
Method Content: Role

- Roles define a set of related skills, competencies and responsibilities.
- Roles are not individuals.
- Individuals on the development team may play multiple roles.
- Roles Perform Tasks.
- Roles are Responsible for Work Products.

Method Content: Work Product

- Work Products (in most cases) represent the tangible things used, modified or produced by a Task.
- Roles use Work Products to perform tasks and produce Work Products in the course of performing tasks.
- Work Products are the responsibility of a Role.
- There are three types of work products:
  - Artifact: typically a configuration managed item
  - Deliverable: required customer/stakeholder deliverable
  - Outcome: "intangible" result of a task such as an installed server or tool.
Method Content: Task

- A Task defines an assignable unit of work (usually a few hours to a few days in length).
- Tasks are performed by Roles (one primary, and optionally additional supporting roles).
- Tasks have a clear purpose, and provide step-by-step descriptions of the work that needs to be done to achieve the goal.
- Tasks modify or produce Work Products.
- Tasks do not define when they are performed in the lifecycle.

Method Content: Guidance

- Guidance may be associate with Roles, Tasks, and Work Products.
- Different types of Guidance depending upon purpose.
- Use Guidance for detailed methodology and supporting information. This will simplify tailoring.
  - For example, Tasks should tell you “what” needs to be done, Guidelines provide detailed “how to”.

Types of Guidance:
- Checklist
- Concept
- Example
- Guideline
- Estimate
- Considerations
- Practice
- Report
- Reusable Asset
- Roadmap
- Supporting Material
- Template
- Term Definition
- Tool Mentor
- Whitepaper
Process: Activities and roles (1/2)

- **Activity**: It's the main subclass of WorkDefinition. An activity describes any part of work executed or assisted by a ProcessRole like tasks, operations, and actions. An activity may consist of atomic elements called steps.

- **Process Role**: The process role is the performer of activities. Also defines responsibilities over specific WorkProducts, and defines the roles that perform and assist in specific activities.

- **Process Performer**: The process performer is the performer of higher-level aggregate WorkDefinitions that cannot be associated with individual ProcessRoles. ProcessPerformer represents abstractly the “whole process” or one of its components, and is used to own WorkDefinitions that do not have a more specific owner.

- **Relationships**: Finally we need to relate the Activities with the Process Roles and Process Performers for doing so we will use the UML relationships perform and assist.

Process: Activities and roles (2/2)

- **Example**

  ![Diagram](image-url)
Process: Capability Patterns (1/2)

- Capability Patterns define the sequence of related Tasks, performed to achieve a greater purpose.
- Task can be specialized for the given context (ex. suppress steps, work products)

Process: Capability Patterns (2/2)

- Capability Patterns may be nested and viewed graphically
- An Activity is an instance of a Capability Pattern.
Process: Delivery Process

- Defined using Work Breakdown Structures and/or Activity Diagrams.
- Defines end-end full-lifecycle process.
- May include Iterations, Phases, Milestones (types of Activities).
- This is just one example, any other lifecycle can be defined.

EPF Composer
EPF Composer

- EPF Composer is a tool platform for process engineers, project leads, project and program managers who are responsible for maintaining and implementing processes for development organizations or individual projects.

- Aims to:
  - Provide for development practitioners a knowledge base of intellectual capital that allows them to browse, manage and deploy content.
  - Provide process engineering capabilities by supporting process engineers and project managers in selecting, tailoring, and rapidly assembling processes for their concrete development process.

EPF Composer

- EPF Composer is built upon the Eclipse platform.
- Supports many of the Eclipse plug-ins:
  - For example, Mylar.
- Different Views present specific information:
  - For example, Library view shows plug-ins and their content.
- Perspectives group related views to support a workflow.
- Standard Perspectives are:
  - Authoring: for editing method content.
  - Browsing: for previewing published elements.
EPF Composer Authoring Perspective

Library View

Task Editor (form based)

Configuration View

Authoring Perspective

Form based plain text or…

...Rich text editors

EPF Composer Authoring Perspective
EPF Composer Browsing Perspective

Configuration: Package Selection
- Select sub-set of method library for publishing to HTML or exporting to MS Project or XML
Configuration: View Definition

- Categories group related elements
- Views defined by selecting Categories

References
References (1)


References (2)

Rational: The Rational Development process
Paul Allen, Stuart Frost, Component-Based Development for Enterprise Systems, Applying the SELECT Perspective, SIGS Book and Multimedia 1998