An Approach for Modeling and Transforming Contextually-Aware Software Engineering Workflows

Roy Oberhauser
Computer Science Dept.
Aalen University
Aalen, Germany
roy.oberhauser@htw-aalen.de

Abstract—**Software engineering environments (SEE) face challenges for providing automated guidance for human-centric software development workflows. Among the various issues is the lack of a method to model software engineering (SE) workflows, as commonly used business process modeling notation is generalized and not conducive to context-aware support for SEE, while available SE-specific process model notation, such as the Software & Systems Process Engineering Metamodel, is not executable. The solution approach in this paper simplifies and validates the modeling of SE workflows via graphical modeling capabilities that extend the Software Engineering Workflow Language (SEWL). Model-based transformation of workflow concepts to diverse workflow management systems (WfMS), as well semantic transformation of SE concepts to a contextually-aware process-centered software engineering environment - CoSEEEK, is supported. The results show the viability and practicality of such an approach to graphically model, transform, and enact SE workflows while transforming relevant SE concepts to an ontology that supports contextual guidance capabilities.**

Keywords-process-centered software engineering environments; software engineering environments; software engineering process modeling; software engineering process model transformation.

I. INTRODUCTION

In order to be generally applicable to various software (SW) development projects, most software engineering (SE) process models remain abstract and require tailoring to the specific project, team, and tool environment. Examples of SE process models include VM-XT [1] (specified for all public-sector IT development in Germany) and OpenUP [2]. Typical SE process models are documented in natural languages and are thus not easily executable in an automated form. Executable processes whose sequence can be automated and modeled in a workflow management system (WfMS) are called workflows. SE workflows can cover some sequence of activities and steps related to requirements, design, testing, etc., for instance Activity Flows in VM-XT [3] or workflows in OpenUP [4].

Despite the potential of process-aware information systems (PAIS) to provide automatic process assistance and guidance for humans, this area has lacked satisfactory assistance mechanisms and standards. While process-centered software engineering environments (PCSEEs) have attempted to address this area [5], they remain intrusive, rigid, and inflexible [6], and fail to adequately support the human, creative, and dynamics of SW development. Thus, such systems are ignored or abandoned by SW engineers.

To address this challenge for such human-centric SE processes, we created a PCSEE called the Context-aware Software Engineering Environment Event-driven Framework (CoSEEEK) [7]. Beyond SE tool sensors and other contextual knowledge, it utilizes workflows to understand the process context. That includes knowing which activities a SW engineer performed, which activity is likely currently being worked on, which activity is next, and associates these with SE concepts of project, teams, persons, roles, tools, and artifacts via an ontology and reasoner. While various facets were investigated, including collaboration [8], quality integration [9], and others, we still faced the problem of providing an easy to use way for SW engineers to model and transform SE workflows, integrating SE concepts without vendor-lockin to a specific WfMS or its tools.

Considering possible SE workflow modeling notation, the Software & Systems Process Engineering Metamodel (SPEM) [10] is aimed primarily at defining a domain-specific notation for the documentation of SE processes, and does not completely address issues related to executable SE processes so that automatic support and guidance for software engineers in operational activities can occur. On the other hand, a general workflow language notation such as the Business Process Model and Notation (BPMN) [11], while executable, lacks the inclusion and semantic meaning of various SE domain-specific concepts.

To address the executable SE workflow language gap, our team had created the text-based language SEWL [12] and had targeted the adaptive WfMS AristaFlow [13] and YAWL [14]. This paper contributes various extensions to the original workflow concepts, including: a new graphical representation for SE-specific workflows blending BPMN and SPEM notation; a graphical editor for SE workflows; details on the model-driven generation of tailored artifacts that target the ontology and heterogeneous WfMS support, specifically the common of-the-shelf (COTS) WfMS jBPM [15] and Activiti [16]; and the workflow ontology generator, which addresses the aspect of contextual-awareness support for workflows in conjunction with CoSEEEK.

The summary of the paper is as follows: the next section discusses related work; Section III describes the solution concept; followed by implementation details; Section V presents an evaluation, which is followed by a conclusion.

 ISBN: 978-1-61208-367-4

II. Related Work

SPEM 2.0 [17] was approved without supporting full process enactment. It proposes two possible approaches for enactment: One proposes mapping to project planning tools. However, this does not support automated adaptation to changing project contexts during project execution. The other proposal is to use the Process Behavior package to relate SPEM process elements to external behavior models using proxy classes. Both approaches lack full workflow modeling and executability at the level of BPMN.

Other work related to enactment of SPEM includes eXecutable SPEM (xSPEM) [17]. Process execution is addressed via transformation to the Business Process Execution Language (BPEL), while process validation is addressed via transformation to a Petri net in combination with a model checker. [18] maps SPEM to the UML Extended Workflow Metamodel (UML-EWM) in order to create a concretely executable workflow. [19] and [20] investigate transforming SPEM to BPMN, while [21] maps SPEM to XPDL. xSPIDER ML [22] is an extension profile of SPEM 2.0 to enable process enactment.

The novelty of our solution is that, in contrast to the above approaches, it targets a simple graphical as well as textual SE process language and notation for modeling, blending the strengths of BPMN and SPEM; it concretely generates executable workflows on different WfMS targets; and it generates an OWL-compliant ontology of SE concepts for context-aware PCSEE tooling support. This addresses prior hindrances and challenges for modeling and integrating SE workflows in SEE.

III. Solution

Figure 1 will be used to describe the solution concept. The four SE process phases shown at the top will be referenced in the solution description below. The basis of the solution concept is the SEWL workflow. A SEWL workflow is *modeled*, either with the graphical SEWL editor or a textual editor, and provided as input for the Generator. To *transform* the input, the Generator utilizes various adapters that generate appropriate workflow templates tailored for a specific WfMS while concurrently providing OWL-DL [23] output of semantic concept instances. These templates are then *deployed*. During *operations*, a Process Manager Service abstracts, via an interface, the WfMS-specific management and interaction details for CoSEEEK (thus CoSEEEK doesn't need to be a PAIS but only extend one) and the ontology is referenced internally during *operations* by CoSEEEK.

Ontologies and semantic technology are advantageous in providing a taxonomy for modeled entities and their relations, a vocabulary, and supporting logical statements about entities [24]. Automated consistency checking and interoperability between different applications and agents support reuse.

To support loose-coupling with CoSEEEK, a service-oriented event-driven architecture was used in conjunction with a tuple space [25] composed on top of a native XML database eXist [26]. A Process Manager Service manages and abstracts the peculiarities of each WfMS (such as jBPM and Activiti), interacting via events indirectly with CoSEEEK through the Space.

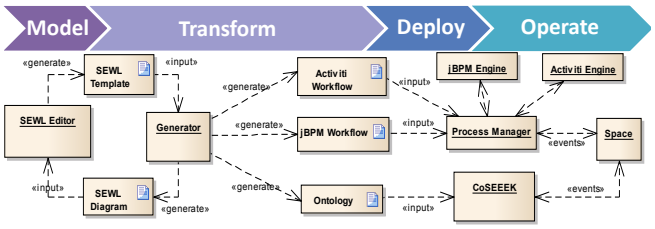

Figure 1. Solution concept.

Model. One primary principle of the approach is that SEWL workflows always remain the reference point of truth. In the process modeling phase, a graphical SEWL Editor assists the process modeler in creating the textual SEWL workflows, which maintain the essence of workflow concepts. Supplemental graphical diagram information (position, font, color, etc.) is retained in separately maintained diagram files, which are kept in sync with the SEWL workflows during the Transform phase. Manual editing of the XML-based SEWL format is thus also possible, however, the Generator will remove all non-applicable elements from the graphical diagrams. Further details are provided in the next section.

Transform. In the Transform phase, workflow templates for the WfMS are generated as shown in Figure 1. The Generator also semantically transforms SE concepts in the workflow to produce an OWL-DL compliant ontology that it utilized for process contextual awareness by CoSEEEK.

Table I shows the mapping of concepts, whereby WUC stands for Work Unit Container and WU for Work Unit. The primary difference between jBPM and Activiti is that in Activiti loops are typically expressed via inclusive gateways, and in jBPM via exclusive gateways. E.g., any concurrent tasks in an SE workflow would be modeled with the BPMN parallelGateway, which activates all branches simultaneously and, when merging, waits for all branches to complete. Most WfMS support such basic features.

TABLE I. Mapping of SE and Workflow Concepts

SEWL	Activiti	jBPM	Ontology
Phase	Service Task + inclusiveGateway	Service Task + exclusiveGateway	WUC + WU
Activity	Service Task	Service Task	WUC + WU
Iteration	Service Task + inclusiveGateway	Service Task + exclusiveGateway	WUC + WU
Task	Service Task	Service Task	WU
Sequence	-	-	-
Parallel	parallelGateway	parallelGateway	-
Loop	inclusiveGateway	exclusiveGateway	-
XOR	exclusiveGateway	exclusiveGateway	-
Roles	-	-	Role Template
Artefacts	-	-	Artefact Template
Variables	-	-	Workflow Variables Template

To address and abstract the integration, communication, and coordination details of the specific WfMS with the

Space, each Activity or Task is represented as a Service Task and during generation wrapped with code that supports the tracking or triggering of the start and finish of an activity or task via event listening and generation.

Deploy. In the process deployment phase, workflows in the WfMS-specific format are deployed into their respective engine and the workflow ontology integrated into CoSEEEK.

Operate. In the process operational phase, the indirect interaction between a Service Task in a WfMS and CoSEEEK via the Space and Process Manager is shown in Figure 2.

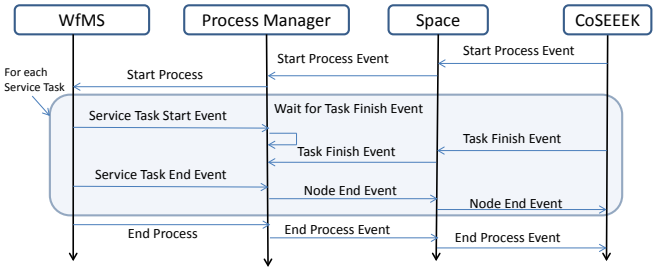

Figure 2. Primary runtime component interaction.

Events (e.g., Task ID 79 start) are written to the Space, and any component can register for events using the Space. As an aside, because all event history is kept in the Space, CoSEEEK components coming online after an absence can determine the context or catch up on any missed events.

IV. IMPLEMENTATION

Details on the implementation will now be discussed. The Graphical Modeling Framework (GMF) and the Eclipse Modeling Framework (EMF), which includes ecore, were utilized. Figure 3 shows a simplified metamodel snippet for implementing the model-driven approach.

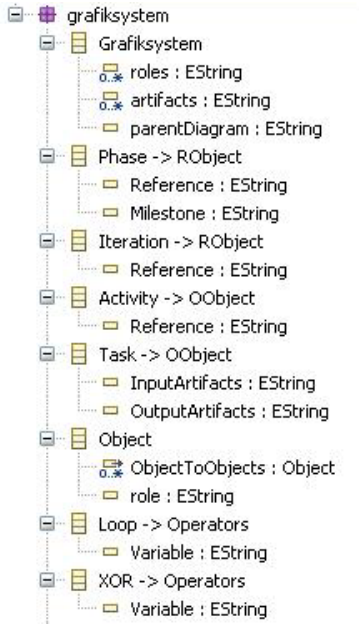

Figure 3. Highly simplified metamodel used in ecore.

Transformation Adapters. Because the transformations are XML-centric, the transformation adapters were coded primarily in Scala. Unique IDs were be generated for every element transformed and its target transformed element. This permits a clear mapping association, useful also for logging. The ontology adapter uses the Jena framework for programmatic ontology access [27] to generate the ontology instances for phases, activities, roles, artefacts, etc.

Figure 4 shows the code generated for the jBPM Service Task, while Figure 5 shows that for Activiti.

```
<task id='2'name='RequestChange'tns:taskName='SEWL Task'>
 <extensionElements>
 <tns:onEntry-script
scriptFormat='http://www.java.com/java'>
  <script>StartEventListener listener = new
StartEventListener();
     kcontext=listener.writeNodeStart(kcontext);</script>
 </tns:onEntry-script>
<tns:onExit-script
scriptFormat='http://www.java.com/java'>
  <script>EndEventListener listener = new
EndEventListener();
           listener.writeNodeEnd(kcontext);</script>
  </tns:onExit-script>
 </extensionElements>
</task>
```

Figure 4. Listing of generated jBPM Service Task.

```
<serviceTask id='RequestChange' name='Request Change'
activity:class='Service'>
 <extensionElements>
  <activity:executionListener event='start'
class='StartEventListener'/>
  <activity:executionListener event='end'
class='EndEventListener'/>
 </extensionElements>
</serviceTask>
```

Figure 5. Listing of generated Activiti Service Task.

The generated OWL output was loaded into Protégé and is shown for a work unit activity in Figure 6. Because the entire XML is very verbose, it is not shown.

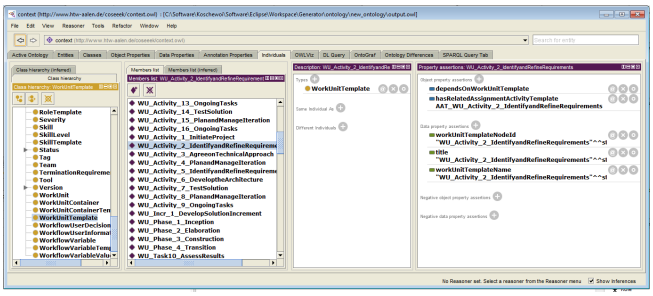

Figure 6: Generated OWL Ontology for CoSEEEK.

SEWL Editor. The textual language supports the specification of SE elements a process may have. Multi-lingual support for referencing the same SE concept instance was implemented, supporting global software development (GSD) processes and their documentation.

The graphical notation is extensible and can be adapted or "skinned" with icons to suit the preferences of the user, which can minimize notation confrontations between different user "tribes", e.g., BPMN purists or SPEM purists.

 ISBN: 978-1-61208-367-4

In order to get the "best of both worlds", the SEWL Editor currently applied a mix of graphical notation as follows:

- SPEM icons for all SE concepts (e.g., phase, activity, iteration, task, role, artefact),
- BPMN icons for process notation, e.g., events, gateways, and connections.

An OpenUP Inception phase workflow in the SEWL Editor is shown in its graphical notation (Figure 7) followed by its textual notation (Figure 8).

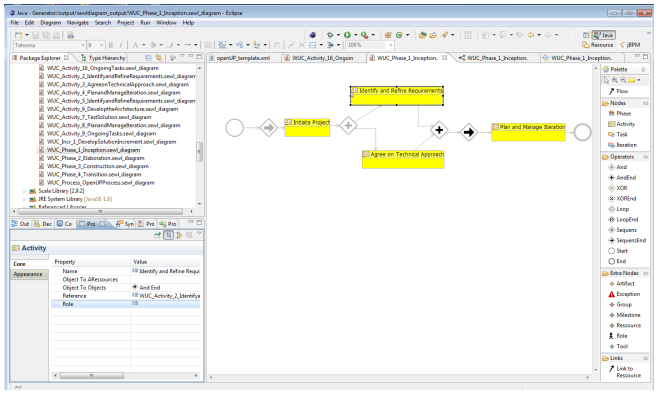

Figure 7. SEWL Editor showing OpenUP Inception Phase diagram.

```
<process base="default_process.xml" xmlns=...>
  <resources>
    <roles>
      <role id="1" name="Analyst" />
      <role id="2" name="Project Manager" />
...
  <elements>
    <element name="phase" base="container">
      <structure>
        <attribute name="repeatable">true</attribute>
      <rules>
        <contains element="activity" />
        <contains element="iteration" />
...
  <artifacts>
    <types/>
    <instances>
      <artifact type="Artifact">Project Plan</artifact>
...
  <tools/>
  <element type="sequence" name="OpenUP Process"
resource="6">
    <element type="phase" name="Inception"
milestone="Lifecycle Objectives">
      <element type="sequence">
        <element type="activity" name="Initiate Project">
          <element type="task" name="Develop Technical
Vision" resource="1">
            <output>
              <parameter name="vision"
tailoring="true">Vision</parameter>
              <parameter name="glossary"
tailoring="true">Glossary</parameter>
...
        <element type="parallel">
          <element type="activity" name="Identify and
Refine Requirements">
            <element type="sequence" resource="1">
...
                <element type="activity" name="Agree on
Technical Approach" resource="4">
...
      <element type="activity" name="Plan and Manage
Iteration" resource="2">
        <element type="sequence">
          <output>...
          </output>
```

Figure 8. Example OpenUP SEWL workflow snippets (end-tags omitted).

To retain graphical data of the layout of nodes and edges, XMI [28] was used. See Figure 12 for an example.

An exemplary subset of the included constraints used to validate the model is listed here, i.e., audit rules. These were implemented in Java to allow usage outside of the GMF.

- Verify phase/activity element has an output and a submodel
- Verify end element has no output
- Verify task does not target iteration/activity/phase
- Verify Loop has LoopEnd, Sequence has SequenceEnd, XOR has XOREnd, And has AndEnd.

V. EVALUATION

The evaluation configuration consisted of an Intel Core 2 Duo CPU 2.26 GHz, 3 GB RAM, Windows XP Pro SP3, JDK 1.6.0-31, Scala 2.91, Activiti 5.8, jBPM 5.2, Eclipse EMT (Helios) SR2. Measurements used `System.nano()`.

Feasibility. As to supporting a broad modeling spectrum, the Eclipse Process Framework (EPF) was used as a reference for modeling Scrum and OpenUP as was an industry partner's internal SE development process. These models were successfully modeled and transformed. Although the complete OpenUP process was modeled, only portions of the Inception Phase are shown below due to space constraints.

Based on the Editor input, the generator was executed and the following output was generated. Since there was no mechanism in the jBPM editor at the time to automatically arrange the elements, all elements are by default placed at the upper left. Thus, Figure 9 was rearranged by hand.

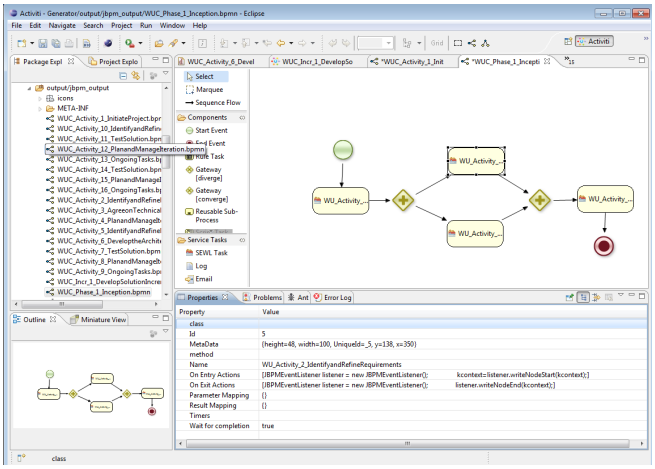

Figure 9. jBPM generated output (arranged later by hand).

A snippet of the corresponding generated output for jBPM is shown in Figure 10 and for Activiti in Figure 11.

Performance. To determine the generator performance, an OpenUP process consisting of a sequence of five nodes was used as the input to the Editor, measuring the performance of each of the generators. For each round, a loop of 1000 generations was averaged. The results are presented in Table II.

 ISBN: 978-1-61208-367-4

TABLE II. GENERATOR PERFORMANCE (IN MILLISEC)

Round	SEWL template	Ontology	Activiti	jBPM	SEWL-Diagram
1	10.3	6701.6	25.2	28.8	65.3
2	10.5	6847.2	24.7	26.6	64.4
3	10.4	6976.9	22.1	27.7	66.4
4	10.1	6901.2	24	25.9	64.2
5	10.4	6945.8	23.2	28.6	65
Avg.	10.3	6917.8	23.8	27.5	65.1

The performance of the generators is satisfactory for typical SE process transformation, except that the verbose OWL generation consumes significant time and should be considered for future optimization.

VI. CONCLUSION AND FUTURE WORK

A solution approach for an easy to use graphical modeling capability for executable SE workflows that can execute on COTS WfMS, while retaining SE semantic information in a separate OWL file for contextually aware PCSEEs, was described conceptually, implemented, and evaluated. The results are indicative that model-based support for transforming SE workflows to common WfMS is both feasible and practical.

Future work includes case studies with industry partners in live settings. Also, bidirectional workflow transformation support between SEWL and an engine-specific workflow format would allow editing in the workflow editor of choice. This entails providing reverse transformation support for engine-specific workflow templates, enabling engine-specific usage of features and editing capabilities via workflow engine-specific editors. For instance, changes made to jBPM and Activiti workflows could be automatically reflected in a SEWL template.

ACKNOWLEDGMENT

The author thanks and acknowledges Vitali Koschewoi for his work on the implementation and diagrams and Gregor Grambow for his assistance with CoSEEEK-related adaptations.

REFERENCES

[1] S. Biffl, D. Winkler, R. Höhn, and H. Wetzel, "Software process improvement in Europe: potential of the new V-modell XT and research issues," Software Process: Improvement and Practice, 11(3), 2006, pp. 229-238.

[2] P. Kroll and B. MacIsaac, Agility and Discipline Made Easy: Practices from OpenUP and RUP. Pearson Education, 2006.

[3] http://v-modell.iabg.de/v-modell-xt-html-english/

[4] http://epf.eclipse.org/wikis/openup/

[5] V. Gruhn, "Process-Centered Software Engineering Environments: A Brief History and Future Challenges," Annals of Software Engineering, 14(1-4), 2002, pp. 363-382.

[6] A. Fuggetta, "Software process: a roadmap," Proc. Conf. on the Future of Software Eng., ACM, May 2000, pp. 25-34.

[7] R. Oberhauser, "Leveraging Semantic Web Computing for Context-Aware Software Engineering Environments," Semantic Web, Gang Wu (ed.), In-Tech, Austria, 2010.

[8] G. Grambow, R. Oberhauser, and M. Reichert, "Enabling Automatic Process-aware Collaboration Support in Software Engineering Projects," Software and Data Technologies (Editors: J. Cordeiro, M. Virvou, B. Shishkov), CCIS 303, Springer Verlag, ISBN 978-3-642-29577-5, 2012, pp. 73-88.

[9] G. Grambow, R. Oberhauser, and M. Reichert, "Contextually Injecting Quality Measures into Software Engineering Processes," the International Journal On Advances in Software, ISSN 1942-2628, vol. 4, no. 1 & 2, 2011, pp. 76-99.

[10] Object Management Group, "Software & Systems Process Engineering Metamodel Specification (SPEM) Version 2.0," Object Management Group, 2008.

[11] Object Management Group, "Business Process Model and Notation (BPMN) Version 2.0," 2011.

[12] G. Grambow, R. Oberhauser, and M. Reichert, "Towards a Workflow Language for Software Engineering," Proc. of the The Tenth IASTED Int'l Conf. on Software Engineering (SE 2011), ISBN 978-0-88986-880-9, ACTA Press, 2011.

[13] P. Dadam et al., "From ADEPT to AristaFlow BPM suite: a research vision has become reality," in Business process management workshops, Springer, Jan. 2010, pp. 529-531.

[14] W. Van Der Aalst and A. Ter Hofstede, "YAWL: yet another workflow language," Information systems, 30(4), 2005, pp. 245-275.

[15] M. Salatino and E. Aliverti, jBPM5 Developer Guide, ISBN 1849516448, Packt Publishing, 2012.

[16] T. Rademakers, "Activiti in Action: Executable business processes in BPMN 2.0," Manning Publications Co., 2012.

[17] R. Bendraou, B. Combemale, X. Crégut, and M. Gervais, "Definition of an Executable SPEM 2.0," In Proc. APSEC 2007, IEEE, 2007, pp. 390-397.

[18] N. Debnath, D. Riesco, M. Cota, J. Garcia Perez-Schofield, and D. Uva, "Supporting the SPEM with a UML Extended Workflow Metamodel," Proc. IEEE Conf. on Computer Systems and Applications, AICCSA, 2006, pp. 1151-1154.

[19] D. Riesco, G. Montejano, N. Debnath, and M. Cota, "Formalizing the Management Automation with Workflow of Software Development Process Based on the SPEM Activities View," Proc. 6th Int'l Conf. on information Technology: New Generations, 2009, pp. 131-136.

[20] M. Perez Cota, D. Riesco, I. Lee, N. Debnath, and G. Montejano, "Transformations from SPEM work sequences to BPMN sequence flows for the automation of software development process," J. Comp. Methods in Sci. and Eng. 10, 1-2S1, (September 2010), 2010, pp. 61-72.

[21] Y. Feng, L. Mingshu, and W. Zhigang, "SPEM2XPDL: Towards SPEM Model Enactment," Proc. of SERP, 2006, pp. 240-245.

[22] C. Portela et al. "xSPIDER ML: Proposal of a Software Processes Enactment Language Compliant with SPEM 2.0," J. of SW Eng. & Applications, 5(6), 2012, pp. 375-384.

[23] D. McGuinness and F. Van Harmelen, "OWL web ontology language overview," W3C recommendation, 2004.

[24] D. Gasevic, D. Djuric, and V. Devedzic, Model driven architecture and ontology development. Springer, 2006.

[25] D. Gelernter, "Generative communication in Linda," ACM Transactions on Programming Languages and Systems, 7(1), 1985, pp. 80-112.

[26] W. Meier, "eXist: An open source native XML database. Web," Web-Services, and Database Systems, LNCS, 2593, 2009, pp. 169-183.

[27] B. McBride, "Jena: a semantic web toolkit," Internet Computing, Nov. 2002, pp. 55-59.

[28] Object Management Group, "MOF 2 XMI Mapping Version 2.4," 2010.

 ISBN: 978-1-61208-367-4

APPENDIX

```
<process processType='Private' isExecutable='true'
id='WUC_Phase_1_Inception' name='Inception'>
 <extensionElements>
  <tns:import name='coseeek.workflow.process
.jbpm.extension.JBPMEventListener'/>
 </extensionElements>
  <startEvent id='_1' name='Start'></startEvent>
...
   <parallelGateway id='_3'
gatewayDirection='Diverging' />
   <parallelGateway id='_4'
gatewayDirection='Converging' />
  <task id='_5'
name='WU_Activity_2_IdentifyandRefineRequirements'
tns:taskName='SEWL Task' >
  <extensionElements>
  <tns:onEntry-script
scriptFormat='http://www.java.com/java'>
  <script>JBPMEventListener listener =
      new JBPMEventListener();
   kcontext=listener.writeNodeStart(kcontext);</script>
  </tns:onEntry-script>
  <tns:onExit-script
     scriptFormat='http://www.java.com/java'>
    <script>JBPMEventListener listener = new
JBPMEventListener();
        listener.writeNodeEnd(kcontext);</script>
    </tns:onExit-script>
  </extensionElements>
  <ioSpecification>
    <inputSet/>
    <outputSet/>
  </ioSpecification>
  </task>
<task id='_6' name='WU_Activity_3_AgreeonTechnicalApp...
```

Figure 10. Example jBPM workflow Snippet.

```
<process id='WUC_Phase_1_Inception' name='Inception'>
 <extensionElements>
    <activiti:executionListener event='start'
class='coseeek.workflow.process.activiti.extension.Proc
essStartEndListener'></activiti:executionListener>
   <activiti:executionListener event='end'
class='coseeek.workflow.process.activiti.extension.Proc
essStartEndListener'></activiti:executionListener>
 </extensionElements>
  <startEvent id='startevent1'
name='Start'></startEvent>
  <endEvent   id='endevent1'  name='End'></endEvent>
    <serviceTask id='WU_Activity_1_InitiateProject'
name='Initiate Project'
activiti:class='coseeek.workflow.process.activiti.exten
sion.DummyService '>
    <extensionElements>
     <activiti:executionListener event='start'
class='coseeek.workflow.process.activiti.extension.Even
tListener'></activiti:executionListener>
     <activiti:executionListener event='end'
class='coseeek.workflow.process.activiti.extension.Even
tListener'></activiti:executionListener>
    </extensionElements>
    </serviceTask>
   <parallelGateway id='parallelGatewayFork1' />
      <serviceTask
id='WU_Activity_2_IdentifyandRefineRequirements'
name='Identify and Refine Requirements'
activiti:class='coseeek.workflow.process.activiti.exten
sion.DummyService'>
      <extensionElements>
       <activiti:executionListener event='start'
class='coseeek.workflow.process.activiti.extension.Even
tListener'></activiti:executionListener>
       <activiti:executionListener event='end'
class='coseeek.workflow.process.activiti.extension.Even
tListener'></activiti:executionListener>
```

Figure 11. Activiti XML Snippet.

```
<graphicsystem:Graphicsystem
xmi:id='WUC_Phase_1_Inception'
parentDiagram='WUC_Process_OpenUPProcess.sewl_diagram'
>
     <newObjects xmi:type='graphicsystem:Start'
xmi:id='startevent1' ObjectToObjects='sequenceStart1'
/>
  <newObjects xmi:type='graphicsystem:Sequenz'
xmi:id='sequenceStart1'
ObjectToObjects='WU_Activity_1_InitiateProject' />
   <newObjects xmi:type='graphicsystem:Activity'
xmi:id='WU_Activity_1_InitiateProject' Name='Initiate
Project'
Reference='WUC_Activity_1_InitiateProject.sewl_diagram'
ObjectToObjects='parallelGatewayStart1' />
...
 </graphicsystem:Graphicsystem>
 <notation:Diagram xmi:id='id_WUC_Phase_1_Inception'
type='SEWL' element='WUC_Phase_1_Inception'
name='Inception.sewl_diagram' measurementUnit='Pixel'>
    <children xmi:type='notation:Shape'
xmi:id='shape_startevent1' type='2043'
element='startevent1'>
...
   </children>
     <children xmi:type='notation:Node'
xmi:id='shape_WU_Activity_1_InitiateProject'
type='2034' element='WU_Activity_1_InitiateProject'>
     <children xmi:type='notation:DecorationNode'
xmi:id='4e841147-2f14-445a-b0b4-30e714be504e'
type='5039'/>
     <children xmi:type='notation:BasicCompartment'
xmi:id='0b62527e-b592-4e3d-a367-541f17843fb9'
type='7011'/>
     <styles xmi:type='notation:DescriptionStyle'
xmi:id='1b9fea72-5856-4be5-9203-1ef5cc58d000'/>
     <styles xmi:type='notation:FontStyle'
xmi:id='3051a516-b9f4-42c6-9698-8072fbe9a301'/>
     <styles xmi:type='notation:LineStyle'
xmi:id='7ea4d238-14fc-4068-a4ce-ed6bb08820af'/>
     <layoutConstraint xmi:type='notation:Bounds'
xmi:id='11135191-6e30-4c7a-a803-dfd437a058bc' x='440'
y='185' />
    </children>
...
 <styles xmi:type='notation:DiagramStyle'
xmi:id='_avAfkaznEeGl_a7M295XCw'/>
  <edges xmi:type='notation:Connector' xmi:id='flow23'
type='4020' source='shape_startevent1'
target='shape_sequenceStart1'>
   <styles xmi:type='notation:FontStyle'
xmi:id='8712763c-8e17-4285-948b-0b78f41f90af' />
     <element xsi:nil='true' />
     <bendpoints xmi:type='notation:RelativeBendpoints'
xmi:id='71805553-c9c1-46ff-8d13-56c6a3ab24fc'
points='[20, 0, -125, 10]$[130, -14, -15, -4]'/>
     <sourceAnchor xmi:type='notation:IdentityAnchor'
xmi:id='63f1b22c-d2fd-408e-9b8a-99044df18ce6' id='EAST'
/>
  <targetAnchor xmi:type='notation:IdentityAnchor'
xmi:id='0fd5db1f-daac-468a-a457-2dcf6bf1ee43'   />
   </edges>
```

Figure 12. Example SEWL diagram XMI code snippet.

 ISBN: 978-1-61208-367-4