Coupling ontology and software development processes - a rendez-vous approach

2nd Workshop on Ontologies and Metamodels in Software and Data Engineering (WOMSDE '07)
Joao Pessoa/Brazil, Oct. 17th, 2007

Andrej Bachmann, Wolfgang Hesse, Aaron Ruß (Univ. Marburg)

- What does OBSE mean?
- Goals of our project
- Foundations of OBSE: KCPM and EOS
- The OBSE process – combining software & ontology development
- The rendez-vous concept, import and export bridges
- OBSE tool architecture
- Outlook: From MDD to ODD?
What does OBSE mean?

Definition 1:

An ontology is an explicit specification of a conceptualisation (T. Gruber [Gru 95]).

• Conceptualisations are not only needed for robots and agents but also for human designers – e.g. of database or application domains of software projects.

• Software developers normally derive the needed domain knowledge from requirements – sometimes from models of previous projects.

• Analysis and design for software projects might very well profit from existing ontologies covering their application domain.

Definition 2:

Ontology-based Software Engineering (OBSE) is a new approach for integrating ontologies in the SE process. Ontologies are used as reservoirs of domain knowledge for starting software projects and – vice versa – can gather domain knowledge from concluded projects.
Sources of domain knowledge in the SE process

- Domain knowledge base (ontology)
- Project requirements [NL]
- Project knowledge base

- Import
- Extract
- Transform & develop
- Implement

- Model (e.g. [UML])
- Code (e.g. [Java])
Main questions

(1) What is the appropriate linguistic level (form / language) for expressing ontologies in the SE context?

(2) How can we define a joint process model – combining Software (project) & ontology development cycles?

Our answers (at a glimpse)

ad (1): For software projects in their early phases, semi-formal ontologies are most useful – e.g. in the form of glossaries.

ad (2): Both kinds of processes should follow an evolutionary approach – i.e. both software systems and ontologies (co-) evolve in multi-cyclic, incremental processes which can be linked by a rendez-vous mechanism.
Goals of the project

• To develop and present a *new approach* to OBSE,
• to define uniform *software development & ontology development* processes for co-evolution,
• to incorporate *glossaries* in the processes using the *KCP Methodology*¹,
• to use an *evolutionary, multi-cyclic process model* (the Marburg based *EOS²* approach) appropriate for process combination via *rendez-vous* bridges
• to build *tools* supporting the OBSE process(es) and its bridges.

Foundations for OBSE:

¹ *KCPM: Klagenfurt Conceptual Predesign Methodology*
² *EOS: Evolutionary, Object-oriented Software development*
Combining the life cycles: The OBSE approach

Ontology life cycle

Domain knowledge (NL)

extract

Ontology (glossary form)

transform

Ontology (UML form)

Ontology (OL form)

ontology life cycle

Software project life cycle

Project requirements (NL)

extract

Project KB (glossary form)

exchange knowledge

Ontology (glossary form)

convert

System model (UML form)

build

System version (PL form)

revise

Ontology (UML form)

Ontology (OL form)
**Problem:**

- End users and developers normally do not "speak the same language:"
- Users prefer natural language but NL-formulated requirements are often ambiguous and incomplete.
- Developers' conceptual models are not well understood by end users.

**KCPM: Klagenfurt Conceptual Predesign Model (1)**
**Problem:**
- End users and developers normally do not “speak the same language:
- Users prefer natural language but NL-formulated requirements are often ambiguous and incomplete.
- Developers' conceptual models are not well understood by end users.

**Solution:**
- Add a new intermediate step: *conceptual predesign*
- KCPM has only few orthogonal and intuitively understandable modeling concepts.
- KCPM is based on glossaries and mainly used for requirements elicitation, analysis and as a basis for validation.
Use of **KCPM** for managing domain ontologies

- Domain ontologies may be represented and managed by means of **glossaries**.
- They consist of
  - **thing types**,  
  - **connection types**,  
  - **operation / cooperation types**,  
  - **conditions**,  
  - ...  
- Forms of presentation:
  - as **table**,  
  - **graphical (as network)**  
  - **UML-like**

<table>
<thead>
<tr>
<th>Project: P5</th>
<th>UoD: publishing companies</th>
<th>UoD-area: publishing</th>
</tr>
</thead>
<tbody>
<tr>
<td>c-id#</td>
<td>name</td>
<td>connection-type</td>
</tr>
<tr>
<td>C001</td>
<td>author/book</td>
<td>book _writing</td>
</tr>
<tr>
<td>C001-2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C002</td>
<td>identification</td>
<td></td>
</tr>
<tr>
<td>C002-1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C003</td>
<td>contract</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**KCPM** network view on an ontology
EOS: *Evolutionary, Object-oriented Software development*

- Every component has its own process (= development cycle)
- All cycles are of similar shape, together they form a "fractal" structure
The *rendez-vous* principle: Ontology and Software life cycles intertwined.
Bridges

- Central goal of the envisaged OBSE tool: to build **bridges** between the development paths.

- **Main bridges** are:
  - **Import** bridges, e.g. between *domain ontology OU phase* and *software project SA phase*.
  - **Export** bridges, e.g. between *software project SO phase* and *domain ontology OA phase* (ontology revision cycle).
Import Bridge Overview

1. Define project requirements
2. Natural Language Requirements Analysis (NIBA)
   - Run NTS parser
   - Interpret syntax tree
3. Create KCP model

- Domain ontology repository
- KCPM domain ontology glossary

OBSE Rendezvous Import Bridge

OBSE Rendezvous Export Bridge
- Design and implementation activities
  - Glossary elements available for export
  - Project ends
Import Bridge In Detail

domain ontology repository

<<structured>>

OBSE Rendezvous Import Bridge

NIBA

retrieve domain ontology component (OC)

OC glossary local copy (OCGLC)

KCPM project glossary (PG)

compare PG element with elements in the OCGLC

show matching pattern (for PG element) in OCGLC to user

select and transfer elements from the OCGLC for project use

adapt PG with imported elements

enriched KCPM project glossary

postedit PG

[no more elements to compare]

[completed predesign glossary]

[design and implementation activities]

[final element to compare]
The OBSE tool

- supports OBSE work
- main target group: SW developers who work on projects with ontology support
- offers bridges to support ontology editors:
  - Import: from ontology to OBSE (analysis) tool
  - Export: from OBSE tool to ontology editor
- offers conversion functions between KCP glossaries and UML class diagrams
- provides an interface to schema integration functions
The OBSE tool prototype

- builds on an eclipse-based implementation of **RCP (Rich Client Platform)**
- offers a *uniform interface* (e.g., for static-/dynamic editors)
- enables switching between different *views* (tables / graphics) using MVC techniques
- offers *synchronisation* mechanisms and transaction handling
- is *meta model-based*: database structure, editors and transformations (e.g. between KCPM and UML) are derived from KCPM meta model
An architecture for the OBSE tool

RCP: Rich Client Platform

EMF: Eclipse Modelling Framework

GMF: Graphical Modelling Framework

EPF: Eclipse Process Framework
The future: From MDD via OBSE to ODD*?

- CIM's are (not) yet well defined in the MDA/MDD world.
- Glossaries might be well used as CIM's.
- MDD idea of continuous model transformations may be extended to the early phases by *Glossary-to-UML* (and vice versa) *transformations*.
Summary

• **Ontology-based Software Engineering (OBSE)** is a new approach for combining *ontology* and *software project* development processes.

• Software projects acquire *domain knowledge* not only from requirements but also from ontologies (= condensed knowledge from previous work).

• Vice versa, *ontologies* may profit from domain knowledge gained in (concluded) projects.

• Knowledge bases are *glossaries* – as defined by the *KCP method*.

• Development cycles are *uniform* and follow the *EOS process model*.

• Knowledge is transferred following the *rendez-vous principle* using *import* and *export bridges* between the two kinds of cycles.

• Bridge functions are part of the *OBSE tool* – now operational on the prototype level.
Please remember:

3rd AIS SIGSAND-Europe symposium on IS analysis and design

in Marburg/Lahn

June 12-13, 2008

sponsored by GI / EMISA

Deadline for papers: January 15th, 2008
References


References


