Semantic Web Enabled Software Engineering, LD2SD as a Case Study

Samad Paydar
samad.paydar@stu-mail.um.ac.ir
WTLab Research Group
Ferdowsi University of Mashhad

13th June 2010
Outline

- Introduction
- Some Themes
- Case Study: LD2SD
Motivations

- Software engineering is a knowledge-intensive activity
- Semantic Technologies (e.g. RDF & Ontology) are the silver bullet for knowledge management
- Semantic technologies enables machines to capture, understand and utilize knowledge of software
- Hence they can help software engineers by automating different activities involved in software engineering
Motivations

- software is becoming more complex day-to-day
  - E.g. move from traditional software to Web applications
  - Move from Web applications to Web 2.0 applications
    - Social Web
  - Move from Web 2.0 to Web 3.0 applications
    - Semantic Web
- Semantic technologies provide machine support to tackle these complexities
Motivations

- New paradigms are evolving in software area
  - Service-oriented architecture
  - Mashup-oriented programming
  - Collaborative programming
  - Ubiquitous computing
  - Grid computing
Motivations

- These paradigms require software agents to have specific capabilities like
  - Interoperability
  - Negotiation
  - Collaboration
  - Dynamic discovery
  - Data Integration & Data Exchange
  - Reusability and extensibility of data models
  - Handling loosely-coupledness
- Semantic technologies are great for these needs
Some Themes
Some Themes

- Using SW technologies in different SDLC steps
  - Requirement engineering [1, 2]
  - Software testing [3]
  - MDSD: Automatic model creation from textual specifications using ontologies [5, 6, 7]
  - Design pattern specification, automatic detection and selection [8, 9, 10, 11, 12]
Some Themes

- Using SW technologies in different related activities
  - Configuration management [13]
  - Version control systems [14, 15]
  - Bug tracking and bug report quality assessment [16]
  - Variability management in Software Product Line Engineering [17]
  - User interface design [21]
Some Themes

- Using semantic model as data model
  - E.g. using RDF model instead of relational model [18, 19]
  - Increased flexibility and extendibility
  - Handling open-world data model
  - Inference capabilities over data
  - The main drawback is less scalability
Some Themes

- New paradigms like
  - Agent-oriented software development
  - Service-oriented architecture and semantic web services
  - Ubiquitous computing
  - …
Some Themes

- Using SW technologies to provide an integrated view over the whole artifacts involved in software system life cycle
- LD2SD is a good example [20]
Case Study: Linked Data Driven Software Development
There are different software artifacts involved in software development life cycle:

- Specifications
- Test data
- Source code
- Bug reports
- Feature requests

- Discussion forums
- Version control
- Configuration management
- Emails
- ….
software artefacts

- bug reports
- feature requests
- specification
- documentation
- version control
- test data
- developer discussion board
- configuration management (releases, etc.)
Therefore, information about a software project are stored in number of heterogeneous, closely related and interdependent datasets.

- These datasets are logically interconnected, but not physically.
- Interconnection is implicit, not explicit.
- Valuable knowledge is hidden inside these datasets.
Introduction

- A thread in the discussion forum focuses on a special module
- It leads to a feature request
- Several emails are communicated between development staff
- Modifications are made on current code
- New Java classes are added
- New unit tests
- Several people might be involved
- Documentation must be updated
- Different people are involved
Introduction

• It is required to make the links between software artifacts and people explicit
• Also to link them to data on the Web (e.g. discussion forums)
LD2SD

• LD2SD is:
  • a light-weight Semantic Web methodology for turning software artifacts into linked data
  • This explicit representation makes new scenarios possible
Finding an expert

Jim is a software project manager. He needs to find a developer in his team with a special expertise and experience.

E.g. finding a developer with experience in parser development which has been involved in the last year projects and no bug is reported for code he has written.
• Bug tracking issues not fixed in due time
  • Jim wants to know if all the issues due yesterday have been fixed and which packages are affected.
Find developer replacement

- Jim needs to find a developer to be replaced with Mary.
- He needs to analyze Mary’s expertise and latest activities:
  - Assigned bugs
  - Committed code
  - Mailing list and blog posts
- And finally he wants to find a developer whose CV matches Mary’s expertise
LD2SD methodology

- Assign URIs to all entities in software artifacts and convert to RDF representations based on the linked data principles, yielding LD2SD datasets
- Use semantic indexers, e.g. Sindice, to index the LD2SD datasets
- Use semantic pipes, e.g. DERI pipes, allowing to integrate, align and filter the LD2SD datasets
- Deliver information to end-users integrated in their preferred environments
LD2SD

- LD2SD datasets can be linked to LOD datasets such as Dbpedia and Revyu
- It enables the reuse of existing information in the software development process
LD2SD Implementation

- “Sindice software project” as the reference software project
- A list of candidate software artifacts
Data layer

- RDFication and Interlinking

<table>
<thead>
<tr>
<th>Software Artefacts</th>
<th>RDFiser</th>
<th>Vocabulary</th>
</tr>
</thead>
<tbody>
<tr>
<td>JIRA Bug Tracker</td>
<td>D2RQ</td>
<td>BAETLE</td>
</tr>
<tr>
<td>JavaSource code</td>
<td>Java2RDF</td>
<td>LD2SD</td>
</tr>
<tr>
<td>Subversion</td>
<td>BAETLE RDFiser</td>
<td>BAETLE</td>
</tr>
<tr>
<td>Developer’s Calendar</td>
<td>iCalendar to RDF</td>
<td>iCalendar</td>
</tr>
<tr>
<td>Developer’s Profile</td>
<td>System specific</td>
<td>FOAF</td>
</tr>
<tr>
<td>Developer Blog</td>
<td>SIOC exporter</td>
<td>SIOC</td>
</tr>
<tr>
<td>Project Mailing Lists</td>
<td>SIMILE RDFiser</td>
<td>SIMILE Email2RDF</td>
</tr>
<tr>
<td>Test Cases</td>
<td>TestCase2RDF</td>
<td>LD2SD</td>
</tr>
</tbody>
</table>
package org.sindice.projects.wp;

import java.awt.Component;
import java.io.IOFile;
import java.lang.String;

public class HelperJava{
    public run(){}
    public getDocument()
}

Data layer
Data layer

Revision: 275
Type: committing
Modified: http://sindice.com/wp/
Added: http://sindice.com/wp/Splitter.java
Author: oanure
Data layer

svn: HelperThread.java

1d2sd: 275

baetle: modified

baetle: added

baetle: author

1d2sd: oanure

svn: Splitter.java

1d2sd: committing

1d2sd: getDocument

1d2sd: hasMethod

1d2sd: getDocument

1d2sd: hasMethod

1d2sd: run

1d2sd: hasMethod

1d2sd: getClass

1d2sd: run

java

HelperThread.java

Splitter.java

JavaClass

modification

committing

run
DERI pipes are used to build RDF-based mashups. They allow to fetch documents from different sources, merge them and operate on them.

4 steps:
1. Fetch the RDF representation of the artifacts using the RDF Fetch operator
2. Merge the datasets using a Simple Mix operator
3. Query the resulting, integrated dataset with SPARQL
4. Apply XQuery in order to sort and format the data from the previous step

The output of the implemented pipe is then accessible via an URI.
Integration Layer

- Integration Layer
Interaction Layer

- Handles the interaction between the integrated data and the end-users such as developers
- Semantic Widgets are used
LD2SD Plug-in

- A plug-in is implemented for Eclipse IDE
- Enables developers to find related information about software artifacts without leaving their development environment
LD2SD Plug-in
Evaluation

- 12 participants with 1-5 years development experience
- Were asked to carry out a set of tasks in two ways: Manual Approach, and Plug-in Approach
  - Identify all blog posts that mention a specific Java class
  - Identify all bugs that have been fixed by modifying a specific Java class
  - Identify all developers that are working on a Java package
  - Identify all blog posts that mentions a specific Java class
  - Identify all bugs that belong to a specific Java package
Evaluation Results

<table>
<thead>
<tr>
<th>Question</th>
<th>Yes</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>Is the tool useful to discover related information?</td>
<td>12</td>
<td>0</td>
</tr>
<tr>
<td>Does our approach added value compared to the usual exploration of related information?</td>
<td>12</td>
<td>0</td>
</tr>
<tr>
<td>Is the design and layout of the tool suited enough for usage?</td>
<td>9</td>
<td>3</td>
</tr>
<tr>
<td>Does the integration of software artefacts as an Eclipse plug-in offer an advantage?</td>
<td>10</td>
<td>2</td>
</tr>
</tbody>
</table>

**Evaluation**

![Bar chart showing time in seconds for tasks T1 to T5.](chart.png)
Conclusion

• Introduced linked data approach in software development paradigm
• The idea is to make implicit links between software artifacts explicit and expose them using RDF
• Provide valuable information to end users by aggregating information from different interconnected software artifacts


