A Linked Data Platform for Mining Software Repositories

Iman Keivanloo, Christopher Forbes, Aseel Hmood, Mostafa Erfani, Christopher Neal, George Peristerakis, Juergen Rilling
Department of Computer Science and Software Engineering
Concordia University
Montreal, Canada
{i_keiv, c_forb, ahmood, c_neal, m_erfa, g_perist, rilling}@encs.concordia.ca

***Abstract*—The mining of software repositories involves the extraction of both basic and value-added information from existing software repositories. The repositories will be mined to extract facts by different stakeholders (e.g. researchers, managers) and for various purposes. To avoid unnecessary pre-processing and analysis steps, sharing and integration of both basic and value-added facts are needed. In this research, we introduce SeCold, an open and collaborative platform for sharing software datasets. SeCold provides the first online software ecosystem Linked Data platform that supports data extraction and on-the-fly inter-dataset integration from major version control, issue tracking, and quality evaluation systems. In its first release, the dataset contains about two billion facts, such as source code statements, software licenses, and code clones from 18 000 software projects. In its second release the SeCold project will contain additional facts mined from issue trackers and versioning systems. Our approach is based on the same fundamental principle as Wikipedia: researchers and tool developers share analysis results obtained from their tools by publishing them as part of the SeCold portal and therefore make them an integrated part of the global knowledge domain. The SeCold project is an official member of the Linked Data dataset cloud and is currently the eighth largest online dataset available on the Web.**

***Keywords*-Linked Data; software mining; fact sharing**

I. Introduction

Mining software repositories (MSR) is an active research area where the extraction of facts and dependencies is used to support software development, maintenance, and prediction tasks. MSR involves time-consuming pre-processing steps due to the heterogeneity and constant changes of the data and structures being mined. XML-based exchange formats have been introduced [1] to address some of these shortcomings. While they work well for smaller and stable datasets, automated integration and sharing of large distributed heterogeneous data is beyond the capability of XML and databases [2].

Moreover, automated integration of analysis results has been a major research challenge due to the lack of a common naming and format schema [2, 3]. Nowadays, research groups analyze the same data from different perspectives that result in several *related* but distinct datasets. To speed up research progress and avoid reinventing the wheel, sharing and integrating these related datasets has become a necessity [2]. However, current approaches do not allow for an automated on-the-fly data integration since facts in individual datasets use (1) random or (2) contextually dependent and (3) non-standardized identifiers [2]. This makes it almost impossible to identify similar (same) entities in different datasets and as a result requires either a significant manual or programming effort to integrate these datasets. An elegant approach to address this problem is to create stable reproducible identifiers that are unique for each fact. In this research, we introduce our SeCold Linked Data platform, that promotes the use of a Reproducible Identifiers [3] to describe facts independent from analysis tools and analysis context. We further discuss how our SeCold dataset addresses the fact sharing and integration challenges.

The remainder of this paper is organized as follows: Section 2 introduces Linked Data. Section 3 reviews related work. Our platform is presented in Sections 4 and 5. Section 6 reviews two sample usage scenarios for SeCold.

II. Background

Linked Data [4, 5] is a by-product of the Semantic Web and has been promoted to address interoperability and sharing issues for open and online datasets. It is designed to be superior to XML-based sharing. It enables both humans and machines to interpret the data for mining, searching, and analysis purposes.

Graph-based Model. While other technologies such as CSV, XML, and databases are designed inherently to represent data as table or tree, Linked Data has been designed based on a graph. A *fact* about an entity is represented using a sentence with three sections which are *subject*, *predicate*, and *object* (e.g. `Bug#1 hasAuthor Smith`). In graph theory, it can be represented using a labeled directed graph link. At the end, a global single data graph will be generated by producing such sentences (i.e. facts) and uploading them to the repository.

Extensible Data Schema. Unlike data exchange mechanisms such as CSV, XML, and web services, it does not require a predefined static schema. Instead, it uses a vocabulary set where it is defined using machine understandable language (contrary to pure XML). The vocabulary set models concepts (e.g. Bug, Commit, VariableName, and Java Class) and relations (e.g. hasAuthor) in the domain of discourse. At anytime, the model can be extended by adding new terms. Moreover, it is possible to have various revisions of the model at the same time.

Feasible and Scalable Reasoning. Contrary to the Semantic Web, Linked Data guidelines [5] do not rely on heavy reasoning and complex logic. It does not mandate

978-1-4673-1761-0/12/$31.00 © 2012 IEEE

reasoning, however transitive closure-based reasoning (e.g. sub type computation) is commonly used by this community.

Being Online. Each entity in the dataset has an online URL. It mandates that the URLs must be dereferencable. That is, anybody on the Web should be able to access facts related to the target entity using its URL[1] via HTTP.

Accessible to Humans. Using the assigned URL to the entity and a web browser, a researcher must be able to see related facts in human-readable format which is HTML.

Accessible to Software. Using the assigned URL to the entity and a HTTP library, a software application must be able to access and fetch related facts in any syntax such as HTML, XML, plain text, etc. The preferred format must be mentioned in the HTTP request header[2].

Being Queryable. Anybody must be able to query the online repository using its Web browser or software[3] to see or download facts matching to the query criteria.

Being Integrated. Linked Data is not only about *intra* but also *inter* dataset integration. Since each entity has its unique online URL, it is possible to have inter-dataset facts.

III. Related Work

Although Linked Data was introduced in recent years, it has already been accepted in diverse domains such as health care [6] and mathematics [7]. More than one hundred official datasets have been published online (linkeddata.org). The Official Linking Open Data Cloud (LOD) graph (Fig. 2) [8] shows these datasets and their connections as of September 2011. Government data, bioinformatics, news, scientific publications and general purpose knowledgebase (e.g. Wikipedia and OpenCyc) constitute major themes within the Linked Data cloud. One of the largest and most popular projects in the cloud is DBpedia which is an NLP-based research project to convert Wikipedia unstructured data into a structured Linked Data format. This allows users to query over Wikipedia data via DBpedia to answer questions like *Find musicians who live in Kingston, Canada*.

Existing efforts for sharing software engineering and research related datasets have focused on sharing source code facts through relational databases [9] or XML-based exchange formats. The PROMISE project provides a website to share research results related to MSR. However, all of these approaches focus on fact sharing and not *inter-dataset integration*.

IV. Our Platform

In this research we introduce a Linked Data based infrastructure for the MSR community that allows for a seamless sharing and integration of research data online. The objective is to provide the same paradigm shift to the MSR community as Wikipedia did to the traditional encyclopedia domain. Knowledge becomes a common, sharable, and integrated item, which is maintained and enriched as part of a community effort. This form of community-based knowledge sharing also addresses some of the MSR challenges discussed by Ahmed Hassan [10]. Our Linked Data approach has been motivated by the success representational approaches in other research domains (e.g. Bioinformatics), where many of these re-usable and integrateable information clouds exist. However, there are challenges unique to the MSR domain, which we discussed earlier in [3].

Extensible Data Models. Modeling for MSR is challenging due to the heterogeneity of software artifacts that refers to fairly similar concepts i.e. issue tracker artifacts such as Bugzilla, Issuezilla, and Jira and version control artifact such as SVN, CVS, and Git. The challenge was to identify similarities and differences among these artifacts then define an abstract representation that captures the main concepts and relationships to model underlying knowledge using an ontological approach that supports extendibility.

In our approach, we ensure tool independence **(**e.g. SVN, CVS, Git) of the data models by representing facts as Linked Data for source code, bug report, commit, code clone, licenses, etc. All models are made available online and take advantage of Linked Data features, such as extensibility and schema versioning. Some of the design challenges are discussed in [3]. A partial view of our ontology family and its connections are shown in Fig. 1. The ontologies are further enriched with traceability links that integrate related concepts. For example, the contributor in METON with the committer in the version control system ontology (VERON) and contributor who resolved an issue (ISSUEON) by modifying some lines of code (SOCON) within the project with specific quality scores (QUAMON) are all aligned.

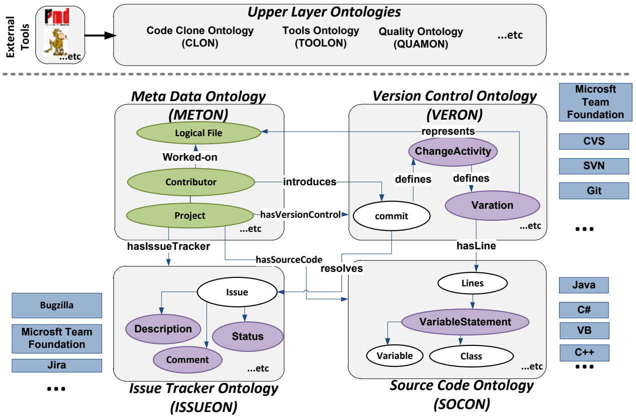


Figure 1. SeCold Ontology Family for IssueTracker, VersionControl, SourceCode, Quality, etc. Stable versions of models are available online at http://secold.org/ontology including guideline documents for end-users

Automated integration. The need for on-the-fly data integration and alignment with no synchronization of corresponding URLs between extracted datasets is addressed by introducing the idea of Reproducible Identifiers [3]. As part of our approach, we have defined several URL generation schemas covering all software ecosystem entities (i.e. digitally captured entities).

Versioning is a major challenge in software engineering and mining. Preserving and presenting fine-grained revisions

[1] e.g. http://aseg.cs.concordia.ca/secold/resource/project/creativecomputing

[2] http://secold.org/online-access

[3] http://secold.org/query-endpoint

of pieces of data is the most challenging part of our research compared to other Linked Data projects. As an example, consider a Java expression such as `foo==1`, implemented in one of the revisions of a specific class. In order to avoid an ambiguous or invalid dataset, the URL must be unique for a particular statement and its specific implementation revision. Therefore, for each revision of a class, we require a different (unique) URL to achieve maximum data perpetuation. Using this approach, we are able to have a URL for each fact in the software ecosystem (e.g. a unique URL for the specific revision of the call graph link in commit#100 of the foo.java class).

V. SeCold – Source code ECOsystem Linked Data

Our SeCold platform (secold.org) provides the largest and first publicly available online Linked Data source code dataset to software engineering researchers and practitioners. The dataset is a repository of source code entities (e.g. AST data, tokens, lines, method blocks, and files), with each of these entities having their own *dereferenceable* URL. SeCold extracts this heterogonous data from several information sources (e.g. online source code, issue tracker, versioning control) and analysis modules. The analysis modules are responsible for extracting relevant explicit (e.g. line number, line content) and implicit (e.g. similarity relation between lines of code) facts.

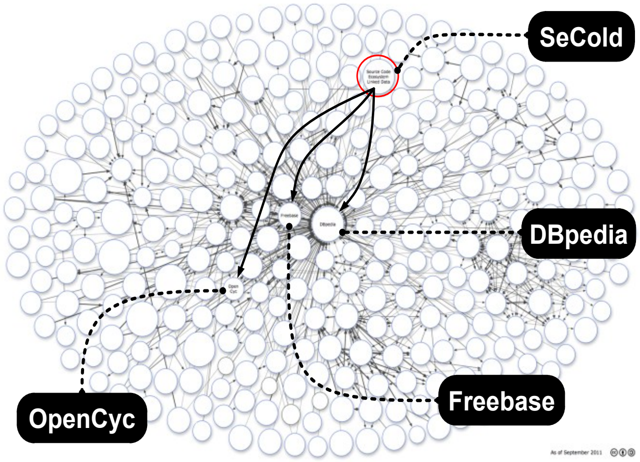


Figure 2. SeCold within LOD graph and its connections to datasets [8]

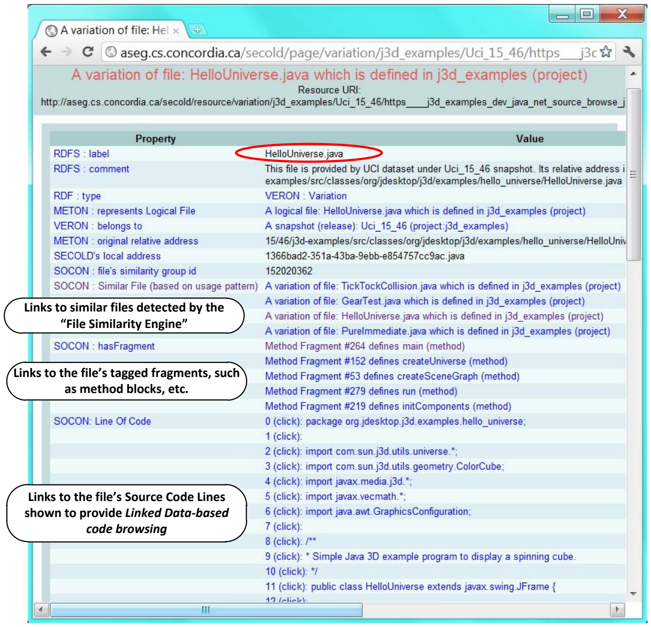
A key feature of SeCold is its integration approach, which provides on-the-fly automatic dataset merging with no need for synchronization. Therefore, the output facts from all analysis modules can be made available as a single integrated dataset. SeCold is accessible in five forms: (1) online HTML (for humans), (2) online RDF/XML (for code search tools), (3) dataset dump files (for research purposes), (4) public query endpoint (for structural queries), and (5) public API search (for free-form and similarity search). SeCold has been included in the LOD Cloud as of September 2011 (Fig. 2).

Our first release consists of two billion triples extracted from 18 000 open source Java projects crawled from the Internet, covering major open source repositories, like Google Code[4] and Sourceforge[5]. As a member of the LOD cloud [13], our SeCold project is connected to DBpedia, Open Cyc, and Freebase (Fig. 2) via shared concepts such as software licenses and project names.

VI. Potential Use Cases

A. First Use Case – Linked Data-Based Fact Browsing

As part of the SeCold environment, we have deployed a public Web Server for SeCold to handle HTTP requests from the research community. The Web Server conforms to Linked Data publication standards [5]. For each type of requested resource, it shows the retrieved triples in the following order: (1) a set of basic facts, (2) a list of value-added facts, and (3) some other related facts. One example is shown in Fig. 3 which demonstrates how SeCold facilitates both browsing of source code (i.e. *Linked Data-Based Source Code Browsing*) and retrieval of related facts (e.g. similar source code) using the Linked Data.



aseg.cs.concordia.ca/secold/page/variation/j3d_examples/Uci_15_46/https___j3c

A variation of file: HelloUniverse.java which is defined in j3d_examples (project)

Resource URI:
http://aseg.cs.concordia.ca/secold/resource/variation/j3d_examples/Uci_15_46/https___j3d_examples_dev_java_net_source_browse_j

Property	Value
RDFS : label	HelloUniverse.java
RDFS : comment	This file is provided by UCI dataset under Uci_15_46 snapshot. Its relative address i examples/src/classes/org/jdesktop/j3d/examples/hello_universe/HelloUniverse.java
RDF : type	VERON : Variation
METON : represents Logical File	A logical file: HelloUniverse.java which is defined in j3d_examples (project)
VERON : belongs to	A snapshot (release): Uci_15_46 (project:j3d_examples)
METON : original relative address	15/46/j3d-examples/src/classes/org/jdesktop/j3d/examples/hello_universe/HelloUniv
SECOLD's local address	1366bad2-351a-43ba-9ebb-e854757cc9ac.java
SOCON : file's similarity group id	152020362
SOCON : Similar File (based on usage pattern)	A variation of file: TickTockCollision.java which is defined in j3d_examples (project) A variation of file: GearTest.java which is defined in j3d_examples (project) A variation of file: HelloUniverse.java which is defined in j3d_examples (project) A variation of file: PureImmediate.java which is defined in j3d_examples (project)
SOCON : hasFragment	Method Fragment #264 defines main (method) Method Fragment #152 defines createUniverse (method) Method Fragment #53 defines createSceneGraph (method) Method Fragment #279 defines run (method) Method Fragment #219 defines initComponents (method)

SOCON: Line Of Code	0 (click): package org.jdesktop.j3d.examples.hello_universe;
	1 (click):
	2 (click): import com.sun.j3d.utils.universe.*;
	3 (click): import com.sun.j3d.utils.geometry.ColorCube;
	4 (click): import javax.media.j3d.*;
	5 (click): import javax.vecmath.*;
	6 (click): import java.awt.GraphicsConfiguration;
	7 (click):
	8 (click): /**
	9 (click): * Simple Java 3D example program to display a spinning cube.
	10 (click): */
	11 (click): public class HelloUniverse extends javax.swing.JFrame {
	12 (click):

Links to similar files detected by the "File Similarity Engine"

Links to the file's tagged fragments, such as method blocks, etc.

Links to the file's Source Code Lines shown to provide *Linked Data-based code browsing*

Figure 3. A Java class resource with its dereferenceable source code URLs and other extracted facts, such as similar files, etc. This approach provides Linked Data-based Fact Browsing.

B. Second Use Case – License Violation Mining

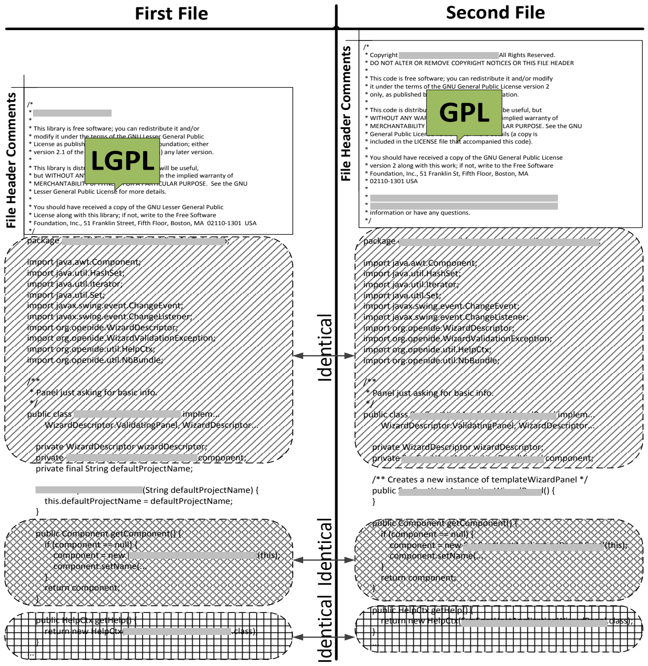
In this section, we describe how the knowledge modeled in the SeCold project can be applied towards the MSR community. The following use case illustrates how SeCold can support the detection of source code license violations across projects in the open source community.

In the software community, it is quite a common practice to assign a license to each source code file and released

[4] http://code.google.com/hosting/

[5] www.sourceforge.net

software. Usually license information is included as plain text at the beginning of a source code file. Over the years, a large number of licenses, mostly for Open Source software development, have been devised. The major challenge resulting from these different licenses is that many of them are incompatible with each other, resulting in license violations, such as when people are combining and reusing software fragments with different licenses in their own systems (e.g. [11]). Fig. 4 shows a concrete example for a potential license violation which is detected using a query on SeCold.



```
/*
 * [redacted]
 *
 * This library is free software; you can redistribute it and/or
 * modify it under the terms of the GNU Lesser General Public
 * License as publish                    oundation; either
 * version 2.1 of the                    ) any later version.
 *
 * This library is distr                 will be useful,
 * but WITHOUT AN                        n the implied warranty of
 * MERCHANTABILIT                        CULAR PURPOSE.  See the GNU
 * Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public
 * License along with this library; if not, write to the Free Software
 * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301  USA
 */
package [redacted];

import java.awt.Component;
import java.util.HashSet;
import java.util.Iterator;
import java.util.Set;
import javax.swing.event.ChangeEvent;
import javax.swing.event.ChangeListener;
import org.openide.WizardDescriptor;
import org.openide.WizardValidationException;
import org.openide.util.HelpCtx;
import org.openide.util.NbBundle;

/**
 * Panel just asking for basic info.
 */
public class [redacted] implem...
    WizardDescriptor.ValidatingPanel, WizardDescriptor...

  private WizardDescriptor wizardDescriptor;
  private [redacted] component;
  private final String defaultProjectName;

  [redacted](String defaultProjectName) {
    this.defaultProjectName = defaultProjectName;
  }

  public Component getComponent() {
    if (component == null) {
      component = new [redacted](this);
      component.setName(...
    }
    return component;
  }

  public HelpCtx getHelp() {
    return new HelpCtx([redacted].class);
  }
...
```

```
/*
 * Copyright [redacted] All Rights Reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER
 *
 * This code is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License version 2
 * only, as published b                    ation.
 *
 * This code is distribu                   be useful, but
 * WITHOUT ANY WAR                         implied warranty of
 * MERCHANTABILITY                         LAR PURPOSE. See the GNU
 * General Public Licen                    ils (a copy is
 * included in the LICENSE file that accompanied this code).
 *
 * You should have received a copy of the GNU General Public License
 * version 2 along with this work; if not, write to the Free Software
 * Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA
 * 02110-1301 USA
 *
 * [redacted]
 *
 * information or have any questions.
 */
package [redacted]

import java.awt.Component;
import java.util.HashSet;
import java.util.Iterator;
import java.util.Set;
import javax.swing.event.ChangeEvent;
import javax.swing.event.ChangeListener;
import org.openide.WizardDescriptor;
import org.openide.WizardValidationException;
import org.openide.util.HelpCtx;
import org.openide.util.NbBundle;

/**
 * Panel just asking for basic info.
 */
public class [redacted] implem...
    WizardDescriptor.ValidatingPanel, WizardDescriptor...

  private WizardDescriptor wizardDescriptor;
  private [redacted] component;

  /** Creates a new instance of templateWizardPanel */
  public [redacted]() {
  }

  public Component getComponent() {
    if (component == null) {
      component = new [redacted](this);
      component.setName(...
    }
    return component;
  }

  public HelpCtx getHelp() {
    return new HelpCtx([redacted].class);
  }
...
```

Figure 4. Example of a potential license violation detect by SeCold

Using traditional mining approaches, several tools will be required to extract facts from the source code (e.g. similarity, license facts etc.) and then merge them into a single repository, as part of a preliminary analysis step. Two major pieces of information are (1) the license of each file and (2) the type of source code similarity (e.g. code clone type-2). Given our data integration approach in SeCold, both information sources are readily available and integrated at different granularity levels, accessible through one common portal. We provide SPARQL query templates to illustrate how SeCold can reduce the need for developing analysis or extraction tools and to encourage the adoption of Linked Data approach by the MSR community at large.

VII. Results and Contributions

In our research project, we provided a platform based on our formal approach [3] for publishing datasets via Linked Data for the MSR community. Both the data models (http://secold.org/ontology) and our Reproducible Identifiers [3] (http://secold.org/api) are publicly available which are part of our contributions. Furthermore, we created and populated SeCold (*secold.org*), the first online Linked Data platform for our research community. SeCold contains several types of data extracted from 18 000 projects, such as source code data (e.g. code content, statements), code clone data (e.g. approximate and exact clones), project licenses, and partially issue tracker and versioning control facts. While these datasets are created and uploaded independently, the facts have been automatically integrated using our Reproducible Identifiers [3]. SeCold currently contains more than two billion facts, which makes it one of the largest LOD datasets on the Web (in Fig. 2 the diameter of the circle indicates the relative size of the dataset). Given this dataset, it is now possible to detect, for example, source code fragments that are cloned among projects with incompatible licenses. Queries like this take advantage of the common representation of these facts, which traditionally would have to be extracted, integrated, and aligned manually (i.e. source code, clones, licenses, evolvability quality), to be able to complete such analysis. Note that SeCold is able to provide similar type of data (e.g. code clones) provided by several sources of information (i.e. different clone detection tools) in one place while keeping them distinguishable from each other for querying and analysis. Since SeCold is crawling, extracting, and representing facts from different resources as Linked Data, the quality (e.g. accuracy) of the data itself depends on the information sources.

References

[1] S. Tichelaar, S. Ducasse, and S. Demeyer, "FAMIX and XMI," Seventh Working Conference on Reverse Engineering, 2000.

[2] M. Würsch, G. Reif, S. Demeyer, and H. C. Gall, "Fostering synergies: how semantic web technology could influence software repositories," ICSE Workshop on Search-driven Development: Users, Infrastructure, Tools and Evaluation, 2010.

[3] I. Keivanloo, C. Forbes, and J. Rilling, "Towards sharing source code facts using linked data," ICSE Workshop on Search-driven Development: Users, Infrastructure, Tools and Evaluation, 2011.

[4] C. Bizer, T. Heath, and T. Berners-Lee, "Linked Data—The Story So Far. Special Issue on Linked Data," International Journal on Semantic Web and Information Systems (IJSWIS), 2009.

[5] T. Berners Lee, Linked Data, http://www.w3.org/DesignIssues/LinkedData.html. Last visited Dec. 2011.

[6] A. Jentzsch et al., "Enabling tailored therapeutics with linked data," in Proceedings of 2nd Workshop Linked Data on the Web, 2009.

[7] C. David, M. Kohlhase, C. Lange, F. Rabe, N. Zhiltsov, and V. Zholudev, "Publishing Math Lecture Notes as Linked Data," Lec. Notes in Comp. Science, 2010.

[8] R. Cyganiak and A. Jentzsch, "Linking Open Data cloud diagram," http://lod-cloud.net/. Last visited Jan. 2012.

[9] S. Bajracharya, J. Ossher, and C. Lopes, "Sourcerer: An internet-scale software repository," ICSE Workshop on Search-Driven Development-Users, Infrastructure, Tools and Evaluation, 2009.

[10] A.E Hassan, "The road ahead for Mining Software Repositories," Frontiers of Software Maintenance (FoSM), 2008.

[11] A. Hemel, K. Kalleberg, and R. Vermaas, "Finding software license violations through binary code clone detection," International Working Conference on Mining Software Repositories, 2011.