# Integrating Software Engineering Data Using Semantic Web Technologies

Yuan-Fang Li
Clayton School of IT
Monash University
Melbourne, Australia
yuanfang.li@monash.edu

Hongyu Zhang
School of Software
Tsinghua University
Beijing 100084, China
hongyu@tsinghua.edu.cn

## ABSTRACT

A plethora of software engineering data have been produced by different organizations and tools over time. These data may come from different sources, and are often disparate and distributed. The integration of these data may open up the possibility of conducting systemic, holistic study of software projects in ways previously unexplored. Semantic Web technologies have been used successfully in a wide array of domains such as health care and life sciences as a platform for information integration and knowledge management. The success is largely due to the open and extensible nature of ontology languages as well as growing tool support. We believe that Semantic Web technologies represent an ideal platform for the integration of software engineering data in a *semantic* repository. By querying and analyzing such a repository, researchers and practitioners can better understand and control software engineering activities and processes. In this paper, we describe how we apply Semantic Web techniques to integrate object-oriented software engineering data from different sources. We also show how the integrated data can help us answer complex queries about large-scale software projects through a case study on the Eclipse system.

## Categories and Subject Descriptors

D.2.7 [**Software Engineering**]: Distribution, Maintenance, and Enhancement—*Restructuring, reverse engineering, and reengineering*

## General Terms

Design, Experimentation

## Keywords

Software engineering data, Semantic Web, data integration

## 1. INTRODUCTION

Over the years of software practice, a plethora of structured or semi-structured software engineering data have been produced by different organizations and tools. These data include project information, source code, changes, bug reports, metrics, etc. Many organizations are now maintaining large software repositories and are willing to mine or even share their software engineering data to facilitate the exchange of results and to improve their current practices by learning from others.

The construction of software repositories is further facilitated by the open-source software (OSS) movement, which has gained momentum in the past decade. A significant and growing number of open-source projects have been created, resulting in increasing amount of software engineering data from a variety of sources.

The large amount of heterogeneous software engineering data brings both opportunities and challenges for researchers and practitioners. On one hand, the abundance of software engineering data enables people to extract useful information from the data and to better manage software engineering activities. In fact, this is a vision of the mining software repository (MSR) community. On the other hand, there is a great variability in the software engineering data. These data may come from different sources, for different purposes, in different formats, language, etc. The data are often disparate and distributed. Such disparity makes it difficult to perform tasks such as mining and query. The integration of these data may open up the possibility of conducting systemic, holistic study of software projects in ways previously unexplored.

After a decade of development, Semantic Web technologies have made tremendous progress with a stack of open standards (such as RDF and OWL) and growing tool support. We believe that Semantic Web technique can provide invaluable insight into the body of software engineering research by translating these data into a single, coherent and logical representation in RDF and making it publicly available. Moreover, such a knowledge source alleviates the current data integration hurdles and help perform large-scale empirical studies.

In this paper, we propose to build such an open, semantic repository for open-source, object-oriented software. We develop domain ontologies that cover the essential concepts about software. Such ontologies will be used as the schema for the integration of data from different sources into the semantic repository. We also propose a general data integration approach and demonstrate how a semantic repository can be constructed and used to answer complex queries.


*MSR*'11, May 21–22, 2011, Waikiki, Honolulu, HI, USA

## 2. RELATED WORK

Over the years of software practice, a huge amount of software engineering data has been accumulated. Recently researchers began to explore methods for mining these repositories (e.g., the MSR conference series) in order to extract useful information that can be used to better understand software engineering activities such as software maintenance and evolution. However, collecting and integrating data from various sources is a non-trivial task, which is often done in an ad-hoc manner that can be very time consuming and error-prone.

Many methods have been proposed to integrate software engineering data. For example, CDIF (CASE Data Interchange Format) was proposed as an industrial standard for exchanging information about software models and code in plain text [1]. XMI (XML Metadata Interchange) is an OMG standard for exchanging information including UML models in XML format. FAMIX (FAMOOS Information Exchange Model [2]) developed in the FAMOOS project at the University of Berne supports the exchange of software engineering data based on CDIF and XMI. FAMIX provides an intermediate representation of object-oriented entities (such as class, method, and attribute) and relations (such as inheritance and access). It can be used as a portable format among a variety of tools. However, the CDIF standard that FAMIX is based on lacks support from both industrial and research communities, and the XMI format is too verbose for human to read. These limitations make FAMIX less extensible. Also, there is a lack of methods and tools that can support integration and queries over the FAMIX data.

The Graph Exchange Language (GXL) [3] is also designed to exchange software representations among various reverse engineering tools. GXL represents software as a graph-based abstraction and then uses XML to encode the graph. Being based on a non-standard representation format, the extensibility of GXL is also limited.

Kim et al. proposed TA-RE as an exchange language for mining software repositories [5]. It is envisioned that TA-RE can support the representation and integration of software change, transaction and project data mined from SCM (software configuration management) systems. However, to our best knowledge, the goal of TA-RE has not been realized.

Recently researchers started to apply Semantic Web techniques to software engineering data. A number of ontologies and vocabularies have been defined in the software engineering domain. The Bug And Enhancement Tracking LanguagE (Baetle) ontology[1] covers bug-related information in defect databases such as Bugzilla and Jira. The Project Vocabulary[2] and Description of a Project (DOAP) ontology[3] both define vocabularies to describe (software) projects. In [6], Witte et al. proposed an ontology-based approach to software maintenance analysis, through the integration of source code and documentation using ontologies. Kiefer et al. defined a software evolution ontology (EvoOnt) and performed analysis on software change between versions and code smell detection [4]. In [7], the authors also chose to use ontology to facilitate software maintenance.

In this paper, we propose ontologies for modeling object-oriented systems and present our Semantic Web based methods for the representation, integration and query of data associated with such systems.

[1] `http://code.google.com/p/baetle/`.
[2] `http://hyperdata.org/xmlns/project/`.
[3] `http://trac.usefulinc.com/doap`.

## 3. INTEGRATION OF SOFTWARE ENGINEERING DATA

There are a large amount of open data available for open-source software projects. These data represent an invaluable information source for software engineering research. However, these data are usually stored in individual *silos* and cannot be easily integrated. A main reason is that these datasets are usually collected from different sources and saved in different formats. In other words, the lack of an open, commonly-agreed schema greatly hinders the integration of software engineering data.

Figure 1 gives an overview our Semantic Web based approach to software engineering data integration. The major steps are as follows:

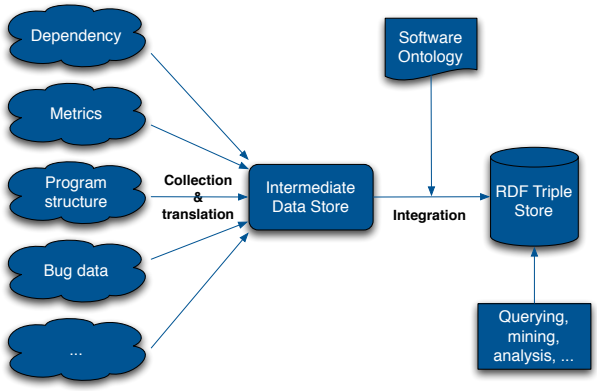


**Figure 1: The high-level processing steps for the integration of software engineering data.**

**Ontology definition.** An ontology serves as an open semantic model that defines concepts and attributes for and relationships among these concepts in a domain. Any semantics enabled knowledge representation and information integration approach must define one or more core domain ontologies. The software engineering domain contains a large number of concepts and relations and it is impractical to represent them all in one ontology. In our work, we take a modular approach and define more closely-related concepts in one ontology. Modularity also simplifies ontology maintenance and improves ontology reuse. The resultant ontology can be easily used/reused through the import mechanism defined in OWL.

**Data collection & translation.** A number of parsers read input files from different sources and translate them into the data structure defined by ontologies. Each software project is treated as an independent entity. Hence, a software project is contained in an RDF named graph (a set of RDF triples identified by a URI, Uniform Resource Identifier). This practice gives scopes to software projects and makes maintenance of RDF data of individual projects easy.

**Data integration.** We use an RDF generator that traverses the above data structure and generates RDF triples and deposits them into the RDF triple store of choice. Much like a database, an RDF triple store stores RDF graphs, which are sets of RDF triples, in persistent storage. Unlike

databases, RDF triple stores do not require or enforce predefined schemas for RDF graphs. However, if an RDF graph specify some OWL or RDFS ontology as its model, the triple store is able to take advantage of the definitions in the ontology during query answering. A number of high-performance triple stores are capable of storing large amounts of RDF triples efficiently. In this paper, we use the Sesame triple store[4] but the choice is irrelevant to the overall approach.

Note that the open nature of Semantic Web based approach facilitates extensibility and data integration. In database and SQL based approaches, database schemas must be defined upfront and they remain relatively stable throughout the life of the application. Schema change usually implies data migration, which can be a tedious and error-prone process. Moreover, new information cannot be easily added as it usually involves modification of existing schemas. On the other hand, RDF-based approaches such as the one presented in this work does not mandate the use of a schema (ontologies). The absence of the compulsory link between RDF data and ontologies makes it easy to add new information. In addition, ontologies, which define domain concepts and their relations, can be overlaid on top of the RDF triple store to provide extra reasoning capabilities for analysis tasks.

# 4. A CASE STUDY ON ECLIPSE DATA INTEGRATION & QUERY

To illustrate our approach, in this section, we present our work on integrating some of the software engineering data from the Eclipse system. Eclipse is a widely-used open-source software development environment. We have collected real data of the Eclipse 3.0 project, which is a large-scale project containing more than 10,000 files and 1,300K lines of code. We demonstrate the integration of three different but related data sets: object-oriented language data, program dependency data and software metrics data. Note that although in this paper we only describe three forms of software engineering data, new information can be easily added and integrated.

## 4.1 Ontology Definitions & Data Collection

We first define ontologies for modeling object-oriented language data, program dependency data and metrics data[5].

### *Object-oriented language data*

Object-oriented languages have many unique concepts such as class and inheritance. Without loss of generality, we will focus the discussion on the Java language as it is widely used and there is an abundance of open-source Java software available. We model the object-oriented language elements (such as classes, interfaces, methods, attributes, visibility, inheritance, etc) as OWL classes, predicates and restrictions.

We use MOOSE to collect object-oriented language data. MOOSE is a platform that supports reverse engineering and software visualization. MOOSE uses a language independent meta-model FAMIX [2] as its internal representation, and exports data in a generic exchange format called MSE.

### *Program dependency data*

As software systems become more and more modularized, dependency modeling becomes important as it is the foundation for tasks such as program understanding and change-impact analysis. Dependency means that the functioning of one element requires the presence of other elements. We define two OWL classes, `Dependable` and `Dependent` and OWL predicates as abstract concepts to represent dependency-related entities such as classes, methods, invocations and return types.

We extract program dependency data produced by the *Dependency Finder* tool, which operates on Java bytecode and extracts dependency graphs. It organizes information in a hierarchical way, records the inter-dependencies among package, class, feature (e.g., method, attribute), and outputs the data in XML format.

### *Software metric data*

Software metrics are an important means for measuring the complexity and quality of a software system. Essentially, each metric has a name, value and associated scale. We define a `Metric` class in OWL and a number of properties to represent its attributes.

We collect complexity metrics data produced by the *Understand for Java* tool. These data includes object-oriented metric data (such as WMC, DIT, etc.) as well as program complexity data (such as lines of code, cyclomatic complexity metrics, etc.). The tool outputs the metric data for each class in comma-delimited plain text format.

## 4.2 Data Integration

Having collected the above data about Eclipse 3.0, we developed a program to automatically convert different datasets into RDF triples and store them in a native (on-disk, persistent) Sesame triple store. Table 1 shows brief statistics of the data collected. In total, more than eight million RDF triples were generated.

**Table 1: Statistics of Integrated Data for Eclipse 3.0**

| Type | Number |
|---|---|
| Package | 1,090 |
| Class | 16,983 |
| Method | 146,356 |
| Constructor | 10,482 |
| Attribute | 54,167 |
| Formal parameter | 111,866 |
| Local variable | 27,071 |
| Containment | 584,835 |
| Inheritance | 139,921 |
| Method invocation | 314,154 |
| Dependency | 401,367 |
| Metrics | 2,119,680 |
| **RDF triples** | **8,762,073** |

## 4.3 Querying the Semantic Repository

Data integration is only the first step in the systemic understanding and analysis of software systems. Having the RDF data available, SPARQL queries[6] can be issued over the integrated dataset to help us better understand the system. Syntactically similar to SQL, SPARQL provides a number of ways to specify constraints on query variables (such as basic triple patterns, logical operations, optional pattern matching, constraints on data values, etc.) to query information in RDF triple stores.

In this section, we use the Eclipse RDF dataset integrated in the previous section to demonstrate the kinds of queries that can be answered through SPARQL. The prefixes `rdf`,

[4] `http://www.openrdf.org/`.

[5] The full ontology is available at `http://csse.monash.edu.au/~yli/OOModel/`.

[6] `http://www.w3.org/TR/rdf-sparql-query/`

oom and dii each represent a different naemspace: rdf represents the namespace for the RDF vocabulary, oom represents the ontology we defined previously and dii represents the namespace of the instance RDF data in the triple store.

**Query** 1. Find all classes that use the public attribute x defined in class org.eclipse.swt.graphics.Point.

```
SELECT DISTINCT ?class
WHERE {
  ?method rdf:type oom:Method .
  ?method oom:hasDependable
      dii:org.eclipse.swt.graphics.Point.x .
  ?class oom:contains ?method .
  ?class rdf:type oom:Class .
}
```

This simple query finds all classes that have a method that uses the specified variable x defined in class org.eclipse.swt.graphics.Point. As it turns out, only the class org.eclipse.swt.tools.internal.JNIGeneratorAppUI uses this variable. Such type of query can help program understanding and maintenance activities.

**Query** 2. Find subclasses of org.eclipse.jdt.internal.compiler.ASTVisitor that will be affected if method traverse() in class org.eclipse.jdt.internal.compiler.ast.ASTNode is changed.

```
SELECT DISTINCT ?class
WHERE {
  dii:org.eclipse.jdt.internal.compiler.ASTVisitor
      oom:hasSubclass ?class .
  ?class oom:contains ?method .
  ?method oom:hasDependable ?method1 .
  filter ( regex(str(?method1),
     "org.eclipse.jdt.internal.compiler.ast.ASTNode.\
         traverse%28*")) .
}
```

This query requires the program structure data as well as the dependency data. By utilizing the integrated semantic repository, four classes are returned: three of them are in the package org.eclipse.jdt.internal.formatter and one in the package org.eclipse.jdt.internal.codeassist.complete. As shown here, the query points to the exact candidates that will be affected by the proposed change.

**Query** 3. Find the top 10 Eclipse 3.0 classes that are larger than 500 LOC (lines of code) and have WMC (weighted methods per class) larger than 10, ordered by descending LOC value.

```
SELECT DISTINCT ?class ?v ?v1
WHERE {
   ?class rdf:type oom:Class .
   ?class oom:hasMetric ?m . ?m oom:hasName "LOC" .
   ?m oom:hasValue ?v . FILTER (?v > 500)
   ?class oom:hasMetric ?n . ?n oom:hasName "WMC" .
   ?n oom:hasValue ?v1 . FILTER (?v1 > 10)
}
ORDER BY DESC (?v)
LIMIT 10
```

This query makes use of two similar blocks of filtering expressions to express constraints on the metric names and values. The query returns the top 10 classes with the largest LOC values (which are also larger than 500) and WMC larger than 10. The results show that these complex classes are mostly from the packages org.eclipse.swt and org.eclipse.jdt.internal. This type of queries helps identify problematic code and potential candidates for refactoring.

The above queries demonstrate the benefits of the integration approach we propose. By constructing an integrated dataset that includes a large amount of data about different facets of a software project, sophisticated queries can be formulated to help analyze the system under study. Although the query examples given above can be also achieved using conventional database and SQL based methods, semantic web based approach is more flexible to changes/additions in data schemas. As the integration approach is designed to be extensible, more types of data can be added with ease. As a result, the repository becomes an increasingly important information source for analysis as more data is deposited.

# 5. CONCLUSION

A huge amount of software engineering data have been accumulated over the years of software engineering practice. These data are often from different sources, in different formats and of different focus. Effective representation and integration of these data can pave the way for more powerful analysis, mining and reasoning.

Semantic Web technologies have been successfully applied in a number of domains to provide a solution to data integration and knowledge management challenges. We believe large-scale integration of software engineering data can provide opportunities for sophisticated analyses both within a single project and across projects. In this paper, we propose to apply Semantic Web techniques to representing and integrating software engineering data. Such an approach is, we believe, extensible, flexible and scalable.

Clearly the data integration and query examples described in this paper only involve a small number of data types. In future, we will perform a large-scale evaluation of the proposed Semantic Web based approach, over a variety of software engineering data (such as defects, changes, developers, etc) across multiple projects.

## Acknowledgement

This research is supported by the NSFC grant 61073006 and the Tsinghua research project 2010THZ0.

# 6. REFERENCES

[1] CDIF Technical Committee. CDIF framework for modeling and extensibility. Technical Report EIA/IS-107, Electronic Industries Association,1994.

[2] S. Demeyer, S. Tichelaar, and S. Ducasse. FAMIX 2.1 - the FAMOOS Information Exchange Model. Technical report, University of Bern, 2001.

[3] R. C. Holt and A. Winter. A Short Introduction to the GXL Software Exchange Format. In *WCRE '00*, pages 299-301, 2000.

[4] C. Kiefer, A. Bernstein, and J. Tappolet. Mining software repositories with iSPARQL and a software evolution ontology. In *MSR '07*, 2007.

[5] S. Kim, T. Zimmermann, M. Kim, A. Hassan, A. Mockus, T. Girba, M. Pinzger, E. J. Whitehead, Jr., and A. Zeller. TA-RE: an exchange language for mining software repositories. In *MSR '06*, pages 22–25, 2006.

[6] R. Witte, Y. Zhang, and J. Rilling. Empowering Software Maintainers with Semantic Web Technologies. In *ESWC'07*, pages 37–52, 2007. Springer-Verlag.

[7] D. Hyland-Wood, D. Carrington and S. Kaplan, A Semantic Web Approach to Software Maintenance. In *EKAW'06* , poster, 2006.