Linked Data Enabled Collaboration in Large-Scale Avionics Software Development

Wei Yin, Wansheng Miao, Jiang Tuo, Xin Zhang, Min Zhang
China Aeronautical Radio Electronics Research Institute
Shanghai, P. R. China
yin_wei@careri.com, 13321971363@189.cn,
tuo_jiang@careri.com, zhang_xin@careri.com,
zhang_min@careri.com

Beijun Shen
School of Software
Shanghai Jiao Tong University
Shanghai, China
bjshen@sjtu.edu.cn

***Abstract*—Collaborative development of large-scale aviation software products across organizational boundaries adds new challenges to existing software engineering processes. In this paper, we propose a novel, linked data approach for handling the diverse software artifacts by adapting features from social network sites. The linked data is introduced into software engineering domain, which connects software specifications, design, code, commits, bugs, tasks and developers as a social network, supporting information retrieval and knowledge discovery among development teams. Our approach can construct this linked data automatically from software development data. Achieve software development traceability, linked data of avionics associate requirements with UML/SysML design, code, code changes, documents and etc., which satisfy airworthiness standard requirements DO-178B/C. It extracts information from multi-source heterogeneous software repositories, identifies the entities, and discovers their relationship. Then, we develop a linked-data-enabled integrated development platform -- SELD, to support multiple inter-team coordination and information sharing. Finally, three usage scenarios are given to illustrate the benefits of SELD.**

Keywords—linked data; knowledge management; mining software repositories; inter-team coordination; large-scale avionics software development, DO-178B/C

I. INTRODUCTION

It is significant that the software systems have been orders of magnitude more complex than ever. The functions in avionics have increased by the demand of electronic equipment and missions with the integrated modular avionics introducing. During the process of software avionics development, there are massive codes, versions, requirements, UML/SysML design models, test cases, bugs, commits, tasks, etc., that are created and maintained by multiply teams. How to establish software traceability among these elements automatically, which is required by airworthiness standard DO-178B/C? How to effectively aware information and discover knowledge from the large-scale distributed data?

To tackle with these challenges, we propose a novel linked data approach for handling the diverse software artifacts, which links the multi-source heterogeneous data at fine-grained semantic level. Our approach can construct this linked data automatically from software repositories, such as Git, UML model, test case lib, Bugzilla, and project management tool. The missing traceability links are also recovered using natural language processing (NLP) and information retrieval (IR) techniques. Using these linked data, information retrieval, change impact analysis and data mining become more effective among distributed development teams.

This paper makes the following major contributions:

1) Software engineering ontology construction. With mapping rules, we construct original ontologies from the metadata of software repositories, then fuse them to construct the whole software engineering ontology with semantic similarity.

2) Software engineering linked data (SELD) extraction. Guided by software engineering ontology, linked data are extracted from multi-source heterogeneous data, and cleaned with entity resolution technique and property elimination technique to decrease redundancy and conflict.

3) Software engineering linked data recovery. Unfortunately, some organizations have ineffective traceability practices in place, largely because of poor communication and time pressure problems. Our approach automatically records traceability links during the software development process, and learns a probabilistic topic model over artifacts by natural language processing technique and information retrieval technique. The learned model allows for potential and missing links recovery.

4) Platform and applications. We design and develop SELD, a platform that builds and shares software engineering linked data among development teams. Then we give three usage scenarios to support collaboration in large-scale avionics software development, including change impact analysis, cross-team task collaboration, expert and component finding.

The rest of this paper is organized as follows: Section II gives the related work about linked data and its applications in software engineering. In Section III, we propose a novel approach to linked data construction from multi-source heterogeneous data in software development process. In Section IV, we describe the linked-data-enabled integrated

978-1-5386-0365-9/17/$31.00 ©2017 IEEE

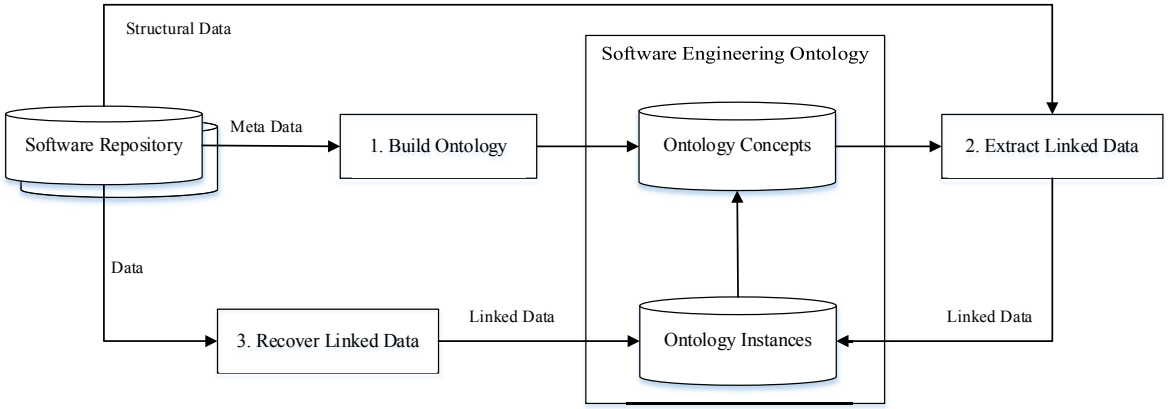

Fig. 1. Using Linked Data to Generate the Software Engineering Data Relationship

development platform and its usage scenarios. Finally, conclusions and future work are offered in Section V.

II. Related Work

With Resource Description Framework (RDF) and Web Ontology Language (OWL) and other forms, linked data turn large-scale heterogeneous disorder data into a structured data network rich in semantics, which can be understood by computer. Compared with traditional data storage system, it has strong scalability, rich semantics, and many other advantages. Accordingly, linked data begins in its applications in software engineering in recent years, in order to cope with the growing scale and complexity of software engineering development data.

Microsoft builds a social-network-based cross-team collaboration software development platform CodeBook[1]. It connects code, API, schedule, documents and personnel information. By providing a web interface to the graph of these connections, software engineers can keep track of task dependencies, discover and maintain connections with other teams, and understand the history and rationale behind the code. Kiefer et al. [2] construct an OWL-based data exchange mode between software repositories -- EvoOnt, which link source code repository, knowledge base and bug information, and query elements using iSPARQL engine. Based on EvoOnt, Iqbal et al. [3] propose a software development approach based on linked data. With Uniform Resource Identifier (URI), it extracts data from version control system, bug tracking tools and source code, turns them into RDF format to construct linked data, and then uses iSPARQL engine to query. The introduction of linked data provides a new solution for the complex data processing in software engineering.

However, most researches only include source code, version information, and defect tracking, while exclude models, documents, project plans, developer information, and other data. In order to establish a complete software engineering semantic web and discover much more useful information, this paper will present a general approach to construction of software engineering linked data, which both extracts linked data from software repositories and discovers their lost linked data automatically.

III. Linked Data Construction Approach

In order to build a software social network for collaborative aviation software development, semantic web is introduced into software engineering to construct fine-grain semantic links among multi-artifacts. We construct software engineering linked data in three steps, as shown in figure 1.

Step 1: Software engineering ontology is built firstly as a common meta-model to allow for a multi-perspective analysis across different views on software. Software engineering ontology concepts are learnt from meta-data of each software repositories. For example, a BUG concept is acquired from a bug table in a software bug tracking repository, with its field as a property of the concept. Then multiple ontologies from different software repositories are merged into a whole one.

Step 2: Guided by software engineering ontology, linked data is extracted from structural data stored in software repositories. For example, there are numerous data stored in a bug table, of which each is mapped to an instance of BUG concept with a link to a specific PERSON instance.

Step 3: Missing linked data is recovered using NLP (Natural Language Processing) and IR (Information Retrieval) technologies. In order to get linking information, we compare the similarity between different data set of data repository with three features: synonyms, verb-object phrases, and structural information.

These three steps will be described in details in the following subsections.

A. Build Software Engineering Ontology

We propose a rule-based mapping method to build the corresponding ontology from metadata of software repository. It extracts metadata from relational database, including table names, column names, primary keys, foreign keys, and integrity constraints, and then analyzes primary keys, foreign keys, and other information. Relational mapping rules are adopted to create new concepts, concept levels, concept properties, and concept relationships. Part of the built software engineering ontology is shown in Figure 2.

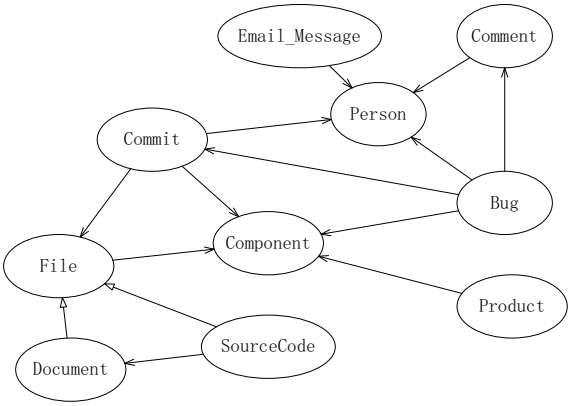

Fig. 2. Part of the built software engineering ontology

Multiple ontologies from different data sources are merged into a complete and unified software engineering domain ontology. Using existing ontology merging method [4,5], we calculate similarities between concepts and merge identical concepts.

1) Name similarity. Concept extracted from relational database usually takes the table name as its name. In traditional software development, name usually reflects the meaning of the concept. Thus, similar names can indicate similar concepts. We analyze the name similarity with respect to string length and minimum edit distance.

$$Sim_{name}(G_1, G_2) = \frac{\min(|c_1|, |c_2|) - edit(c_1, c_2)}{\min(|c_1|, |c_2|)} \quad (1)$$

where c1 and c2 denote the name string of the concept G1 and G2, and the function edit(c1, c2) denotes a minimum edit distance between two strings.

2) Property similarity. Similar concepts usually have similar property sets. Thus we analyze the property similarity with respect to the concept property sets using

$$Sim_{property}(G_1, G_2) = \frac{R_1 \cap R_2}{R_1 \cup R_2} \quad (2)$$

where R1 and R2 denote the property sets of concept G1 and G2.

3) Structure similarity. Concepts that share similar parent concept or sub-concept may be similar; concepts that share similar relationships with other concepts may be similar. Thus we analyze the structure similarity with respect to the similarity of parent concept and sub-concept using

$$Sim_{struct}(G_1, G_2) = Sim_{fset}(G_1, G_2) + Sim_{sset}(G_1, G_2) \quad (3)$$

where Simfset(G1, G2) and Simsset(G1, G2) denote the similarity of parent concept and sub-concept of concept G1 and G2, which are calculated by Formula (1) and Formula (2).

These concepts are identified as the same and merged into one, if the similarity value exceeds a threshold which can be set through experiments.

B. Extract Linked Data

Linked data and their properties and relations are extracted from software repository. Then, as concept instances, they are mapped to the concepts in software engineering ontology. The main mapping between ontology and relational schema in database is shown in Table I.

Linked data of software engineering consists of software elements (i.e. concept instances) and their relations (i.e. object properties of elements). Fig. 3 illustrates a sample of extracted linked data of software engineering in black.

TABLE I. MAPPING BETWEEN ONTOLOGY AND RELATIONAL SCHEMA

Ontology		Relational Database		
Element	**Label**	**Element**	**Label**	**Instance Identification**
concept	class	table	table	primary key
property	data_property	column	col	/
relationship	object_property	column	col	foreign key

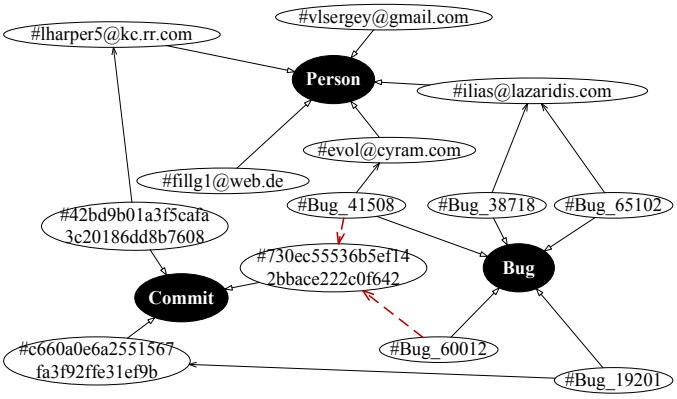

Fig. 3. A sample of extracted and recovered linked data of software engineering

C. Recover Missing Linked Data

Unfortunately, many organizations have ineffective traceability practices in place, largely because of poor communication and timing-pressure problems. Therefore, some important links are missed or lost in software repository, such as requirement-code links. We propose a generic method to recover these links between two software elements, combining three features:

· **Synonyms**. We use the IR technique to extract terms from software artifacts, i.e., software documents and source code. Although these terms are not identical, they are often synonyms when two software elements are linked correctly.

· **Verb-object phrase**s. In software artifacts, most sentences have verb-object phrases, which convey the essential meanings of sentences.

· **Structural Information**. There is lots of structural information potentially contained in source code. For example, source code related to the same software element always has traceability links.

Following is a procedure of recovering missing links:

1) Extract verb-object phrases from software artifacts, including software document and code comments, through syntax and semantic analysis;

2) Use WordNet to search for synonyms of a given term, and merge these synonyms;

3) Represent software artifacts using a Vector Space Model (VSM);

4) Set VSM parameters and similarity threshold through annotated training data;

5) Calculate text similarity between software artifacts as the cosine of the angle between the corresponding vectors, and find initial links;

6) Update text similarity using structural information of source code, such as inheritance, call relationship.

As a result, recovered links will complement the software engineering linked data. As illustrated in Fig. 3, there are two recovered links between a bug instance and a commit instance, highlighted in red.

IV. Platform Development and Applications

We design and develop SELD, a platform that builds and shares software engineering linked data among development teams.

A. Platform Features

SELD focuses, in its current implementation, on linking the data among software tools. In our organization, several software tools like IBM DOORS, Rhapsody, Ansys SCADE Display, QT, Testbed and SVN are used to develop the avionics software, following model-based method. Each tool creates and maintains its own data or model. SELD bridges these information islands through linked data, to integrate those tools, and thus build a large-scale avionics software development integrated platform.

SELD platform offers the linked data features of extraction, construction, profiling, sharing, searching, browsing, query and data mining. See Table II for a detailed description.

TABLE II. MAIN FEATURES IN THE SELD PLATFORM

Feature	Description
Data extraction	Extract data from software tools and repositories.
Linked data construction	Link the multi-source heterogeneous data at fine-grained semantic level.
Profile	Profiles with information about developers and models.
Data sharing	Download of the actual model artifact.
Data searching and browsing	Search and browse linked data.
Linked Data query	SPARQL query over linked data.
Defect prediction	Not implemented.
API recommendation	Not implemented.
Process pattern mining	Not implemented.

The constructed linked data supports the artifact traceability in software life cycle, required by DO 178B/C standard [6,7], as shown in Fig. 4. It has to be considered the traceability of HLR (High-Level requirements), LLR (Low-Level Requirements), software architecture, source codes and outputs of the integration process at Chapter 6. Those are the objectives in Table A-3(6), A-4(6), and A-5(5), which are the traceable design results.

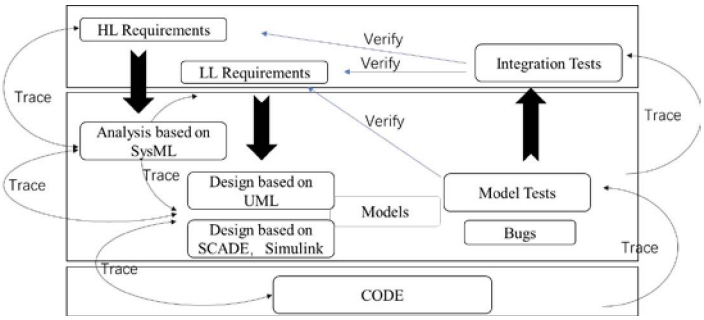

Fig. 4. Artifact traceability in software life cycle

B. Platform Architecture

The architecture of SELD are shown in Fig. 5. It can support multiple development stages of avionics software, integrate requirement tools, design tools, implementation tools, simulation tools, testing tools, version control tools and others, and standardize the development process. Through this linked-data-enabled integrated development platform, teams can share information and knowledge, and collaborate effectively. And also, we can conduct data analysis and mining on the linked data to find insights, after accumulating a certain amount of data in near future.

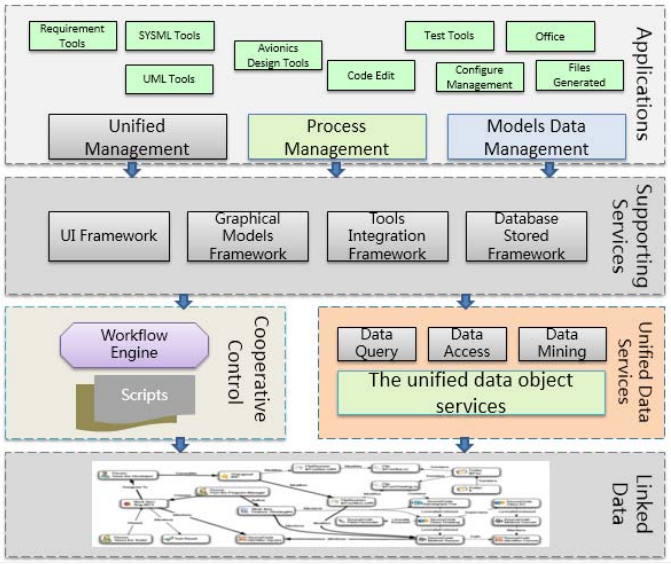

Fig. 5. Architecture of SELD platform

Figure 6 shows a screenshot of the SELD, where requirements from DOORS are linked with design elements in models from SCADE.

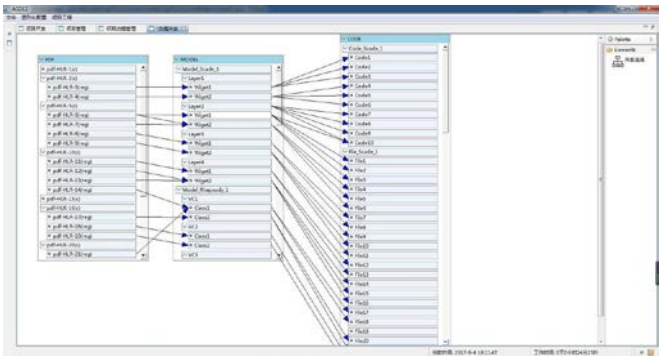

Fig. 6. Screenshot of SELD platform

C. Platform Application Results

We have applied an avionics project (display system) in SELD platform, which are lack of connections to other software engineering process assets. In this project, the relationship between requirements and requirement models are realized by DOORS with Rhapsody, and the relationship between requirements and design models are realized by DOORS with SCADE. Besides this project, simulated data are also used to test SELD.

In COTS IBM products, there are connection gateways among DOORS and Rhapsody or others. If it's just that the two tools are connected, then it may spend for at least 10 minutes or more, when we look for the links between elements in the requirement model and those in the design model.

Through the SELD platform, the one-on-one tools gap can be solved during the software development lifecycle. The links among requirements, comments, models and codes from the avionics project and simulated data have been established.

TABLE III. SEARCHING TIME OF RELATIONSHIP

	An avionics project		**Simulated data**	
Relations among requirements-models-codes (items)	5000	10000	20000	32949
Search Time of an item (seconds)	0.538	1.046	2.135	3.544

It is limited by performance problems when importing from the DOORS database to Gateway with number of objects, size of texts, and amount of links per objects etc., which is not a problem in SELD. As we pick one relationship from large amounts of them, the search result can be rapidly presented highlight, as shown in Table II. Furthermore, it is a global project view to trace the software engineering process repository.

D. Usage Scenarios

We illustrate the benefits of adopting features from SELD platform to support collaboration in large-scale avionics software development, by following three usage scenarios.

1) Change impact analysis

During software development and maintenance, software artifacts are often changed for adapting the improvements of solutions or changes in requirements, environments, and resources. One change of a software artifact usually directly or indirectly impacts on the others, and thus the ripple effects (i.e., a sequence of follow-up changes) will exist.

We propose a novel multi-perspective change impact analysis approach to assessing the change impact on the whole software system, using software engineering linked data in SELD. First, we predict change impact propagation in one step, i.e., direct impact between two elements. And for this, we extract dependency features between these elements from the linked data, to calculate the impact degree. Then, a random walk algorithm is designed to direct change impact propagation in multiple steps, i.e., across multiple heterogeneous elements.

2) Cross-team task collaboration

When teams collaborate as part of a large project, a member of one team will often assign a task to a member of another team. Tracking the status of tasks assigned across teams is frustrating. The task can be delayed due to poor communication or differing priorities because no one advocates for it. SELD can help by increasing transparency between teams. Once the task has been assigned to a member of the other team, the designated person can befriend the assignee to watch his progress on the task and to see what other responsibilities are competing with the task.

3) Expert and component finding

In this scenario, a team decides to begin working on a new technology or a new component, and wants to know if any expert at the company or partners is working on something similar. Using the linked data, SELD can suggest projects from other teams that are similar to this project, and recommend experts with similar technology or similar development experience. Analyses of similarities between specifications and between bodies of code can produce affinities between code bases owned by separate teams. This will help the team decide whether to reuse the other team's technology and components, or collaborate with the other team to build the technology and components in a way that is compatible with both teams' visions.

V. CONCLUSION

We have motivated and introduced linked data driven software development in this paper. We propose a novel, linked data approach for connecting software specifications, design, code, commits, bugs, tasks and developers as a social network, to support collaborative development of large-scale aviation software. Then, we develop a linked-data-enabled integrated development platform SELD, and three usage scenarios are given to illustrate its benefits.

We plan to implement further SELD features to develop more applications, such as defect prediction, API recommendation, and Process pattern mining. We aim to overcome a shortcoming of the current implementation, as it is not 100% linked data conforming; in certain places we use URNs rather than HTTP URIs. Additionally we will integrate

more software tools to involve new data sources to yield higher-quality and also more links.

ACKNOWLEDGEMENT

This research is supported by National Natural Science Foundation of China (Grant No. 61472242).

REFERENCES

[1] Begal A, Deline R. 2009. Codebook: Social networking over code, In Proceedings of International Conference on Software Engineering, 263-266.

[2] Kiefer C, Bbrunstein A, Tappolet J. 2007. Mining software repositories with iSPARQL and a software evolution ontology, In Proceedings of the 29th International Conference on Software Engineering Workshops. IEEE Computer Society, Washington, DC, USA. 10.

[3] Iqbal A, Ureche O E, Hausenblas M, et al., 2009. LD2SD: Linked data driven software development. In Proceedings of the 21st International Conference on Software Engineering and Knowledge Engineering. Piscataway: IEEE, 63-69.

[4] Jain P, Hitzler P, Sheth A, et al. 2010. Ontology alignment for linked open data. In Proceedings of the Semantic Web – ISWC, Springer Berlin Heidelberg, 402-417.

[5] Suchanek F M, Abiteboul S, Senellart P. 2011. PARIS: Probabilistic alignment of relations, instances, and schema. PVLDB (Proceedings of the VLDB Endowment), 5(3):157-168.

[6] RTCA DO‐178B. 1992. Software considerations in Airborne Systems and Equipment Certification.

[7] RTCA DO-178C. 2011. Software considerations in Airborne Systems and Equipment Certification.

[8] Leanna Rierson, 2013. Developing Safety-Critical Software-A practical Guide for Aviation Software and DO-178C Compliance. CRC Press.

[9] El-Roby, A., & Aboulnaga, A. 2016. ALEX:Automatic Link Exploration in Linked Data. IEEE, International Conference on Data Engineering. 1839-1853.