Ontology-Driven Unified Governance in Software Engineering: The PoolParty Case Study

Monika Solanki[1(✉)], Christian Mader[2], Helmut Nagy[3], Margot Mückstein[3], Mahek Hanfi[3], Robert David[3], and Andreas Koller[3]

[1] Department of Computer Science, University of Oxford, Oxford, UK
monika.solanki@cs.ox.ac.uk
[2] Fraunhofer IAIS, Bonn, Germany
[3] Semantic Web Company, Vienna, Austria

Abstract. Collaborative software engineering environments have transformed the nature of workflows typically undertaken during the design of software artifacts. However, they do not provide the mechanism needed to integrate software requirements and implementation issues for unified governance in the engineering process. In this paper we present an ontology-driven approach that exploits the Design Intent Ontology (DIO) for aligning requirements specification with the issues raised during software development and software maintenance. Our methodology has been applied in an industrial setting for the PoolParty Thesaurus server. We integrate the requirements specified and issues raised by PoolParty customers and developers, and provide a graph search powered, unified governance dashboard implementation over the annotated and integrated datasets. Our evaluation shows an impressive 50% increase in efficiency when searching over datasets semantically annotated with DIO as compared to searching over Confluence and JIRA.

1 Introduction

In today's dynamic, agile and collaborative environments, software design and development has metamorphosed into a complex social activity, involving teams of software architects, developers, testers and maintainers. In response to this need, several collaborative and social development frameworks for software engineering have been implemented and are in wide use [2].

In collaborative software development, given a set of requirements, typically, several iterations, deliberations and informal discussions are undertaken among the design team members, before a final consensus can be reached, on the features that are to be included in the concrete realisation of the artifact. The requirements and discussions, if at all documented, are recorded as unstructured text, that makes their search and retrieval a cumbersome process. Further, during and after implementation and deployment of the software, several issues may typically arise and get recorded either as part of the maintenance process or in response to new requirements. One critical shortcoming of collaborative

E. Blomqvist et al. (Eds.): ESWC 2017, Part II, LNCS 10250, pp. 109–124, 2017.
DOI: 10.1007/978-3-319-58451-5_8

environments in wide use today such as Atlassian Confluence[1], JIRA[2] and Github[3] is that they provide generic fields such as "issue" and "comment" which encapsulate all discussion types. This makes it extremely difficult to retrieve information relevant to a specific aspect of a requirement. Further, they also do not provide the interfaces needed to associate design requirements with implementation issues in a structured and systematic way. Integrating the requirements with the issues which should be a core functional capability of most collaborative design environments, is not yet well supported.

In this paper we present an ontology-driven approach that captures the knowledge emerging during software design, development, implementation and maintenance and exploits it for *unified governance* of the engineering process. The term "unified governance" was defined in the ALIGNED[4] project and denotes capturing knowledge of the software development lifecycle from various sources and expressing it using a common ontology so that it can be governed, i.e., queried and new knowledge inferred, in a unified way. We exploit DIO (Design Intent Ontology)[5], a content ontology design pattern, that provides a generic mechanism for formally describing the intents or the rationales that emerge or are generated during the processes that underlie modern design decision phases. Our approach conforms to one of the main criteria for systems that record design rationale and decisions, mainly that it should be least disruptive to the designers and the actual design process itself [3,5].

We evaluate our approach within the settings of an industry-driven use case from the Semantic Web Company (SWC henceforth)[6], where customers and developers of the PoolParty Thesaurus Server (PPT henceforth) document their requirements in a tailored version of Confluence and file bug reports or raise design issues using JIRA. By aligning and creating merged repositories of requirements, customer feedback, bug reports, project documentation and mining of web resources we are able to exploit the integrated knowledge to develop a semantic search mechanism for unified governance.

The remainder of this paper is structured as follows: Sect. 2 discusses related work. Section 3 describes our use case scenario. Section 4 presents the requirements that need to be addressed by a unified governance framework. Section 5 describes our knowledge mining and representation approach. Section 6 presents the architecture and implementation. Section 7 outlines our evaluation strategy and finally Sect. 8 presents conclusions.

2 Related Work

Extensive research [4] has shown that capturing design intents is a very difficult problem. Surveys [8,11] on the capture of design rationale and barriers to their

[1] https://www.atlassian.com/software/confluence.
[2] https://www.atlassian.com/software/jira.
[3] https://github.com/.
[4] http://aligned-project.eu/.
[5] https://w3id.org/dio.
[6] https://www.semantic-web.at/.

uptake have also been reported. IBIS [9], one of the most widely adopted argumentation models, is a network-oriented model based on capturing the issues, positions and arguments underlying a design deliberation.

Representing design intents or design rationales as ontologies have been explored for various specialised domains such as software engineering [6], ontology engineering (OE) [13], product engineering [14] and aerospace engineering [10]. However there is no generic, domain-independent design intent capture model available as a design pattern that can be specialised for any design rationale capture scenario.

Based on IBIS, one of the first attempts to provide an ontological representation of design rationale was by Medeiros et al. [6]. The authors proposed the Kuaba ontology for the representation of model-based software design. A major limitation of this approach is that the reuse of the design rationale is strongly dependent on the formal model used for artifact design, thereby introducing encoding bias [7].

An argumentation ontology has been proposed as part of the DILIGENT [13] framework for the collaborative building of ontologies. The basic conceptualisation in the ontology has been derived from IBIS and extended to include argumentation-specific entities, particularly those suitable for ontology engineering. The critical limitations of this ontology are that the issues are not interlinked to the requirements.

The fundamental difference between the various semantic and non-semantic models for representing design rationale and DIO, lies in the granularity of the representation. IBIS-based models provide a single conceptualisation for many aspects of a design intent, while DIO provides a finer level of representation, thereby making the specification more expressive, without compromising computability. As an example, typically the terms, "Idea", "Position" and "Solution" have been used to represent the solution to a "Question" or an "Issue", while the term "Argument" has been used for representing the rationales for and against a proposed solution. On the other hand, DIO incorporates the mutually disjoint concepts of "Argument" and "Justification" to represent the distinct rationales as well as the mutually disjoint concepts of "AlternativeSolution" and "MandatedSolution" to represent all solutions and accepted solutions respectively.

Table 1 compares the conceptual abstractions defined in DIO with those in some of the approaches discussed above.

Wikidsmart by zAgile[7] claims to provide "semantic enablement" to Atlassian Confluence by capturing the data in a semantic repository, using "a set of ontologies and metamodels specific to a domain of interest". It is, however, unclear what ontologies are used by the framework's semantic repository to model the information contained in Confluence. Our work follows a similar approach to Wikidsmart in the way that we also extract information from existing Confluence and JIRA installations and express it using semantic technologies. However, we contribute and publish reusable OWL-based ontologies for modeling this

[7] http://www.zagile.com.

Table 1. A comparison of the ontological approaches for capturing design intents

	DIO	Kuaba	DILIGENT	RaDEX	ISAA
Available as a design pattern	Yes	No	No	No	No
Generic across domains	Yes	No	No	No	No
Formal representation	Yes	Yes	No	No	No
Ontological serialisation	Yes	No	No	No	No
Explicit provenance metadata	Yes	Limited to agents	Limited to agents	No	No
Traceability wrt. design requirements	Yes	No	No	No	Yes
Traceability wrt. design artifact	Yes	Yes	No	No	Yes
Capturing intent	Yes	No	No	No	Yes
Assumption specification	Yes	No	No	No	
Constraint specification	Yes	No	No	No	No
Heuristics specification	Yes	No	No	No	No
Temporal specification	Yes	Partial	No	No	No

knowledge (DIO and DIOPP), ready to be processed by a wide range of tools of the Linked Data technology stack.

3 Motivating Scenario

The motivation for our work is the current setup at SWC, where Atlassian Confluence is used to support requirements engineering and JIRA is used by team members and SWC customers for issue and change tracking, organising ideas from team members as well as collecting them from customers. Following the agile methodology of software development, the data is recorded in Confluence under headings such as "Requirements", "Goal","User Story", "Epic" and "Stakeholders". Additional fields such as "Precondition", "Detailed description", "Acceptance criteria & Test scenario" are included to provide further context to the requirements. A single field, "Comment", captures the opinions/discussion carried out by human agents.

SWC collects the requirements for each version of PPT in the designated Confluence space. Requirements are then linked to pages containing epics and user stories. Most of these pages are structured based on standard templates defined by SWC. The outputs from these template-based pages are largely document-centric and require extensive human intervention to synthesise and synchronise them with PoolParty development tasks.

By using ontologies to annotate and provide metadata to the content extracted from Confluence and JIRA, SWC would be able to create merged repositories of requirements, customer feedback, bug reports and project documentation thereby consolidating PoolParty experiences, customer ideas and market needs in order to integrate them into products. This is a key factor for successful development of SWC products and for raising customer satisfaction and enterprise agility. The integrated information would enable the mining of intents that lead to the development in PoolParty. Questions asked by customers will flow faster into the requirements engineering system. The process will help to generate concise reports on distributed business objects and entities relevant for the development processes, and to coordinate the data management and development workflows required to deliver new versions of the evolving PoolParty product.

4 Requirements for Unified Governance

Based on the challenges that need to be addressed to enable the mapping of design intents for unified governance, where intents are derived from requirements in Confluence and issues in JIRA, we identified the following requirements:

- **Enabling Knowledge reuse**: As new agents — humans and software get involved during various phases of design, new processes are introduced. The key motivation behind recording the intents underlying decisions is that the knowledge can be used to inform future agents and processes of the discussions that have happened so far.
- **Shared semantics via annotations**: The framework must facilitate a common understanding and uniform interpretation of the design intents among the participants of the decision making process. The unified representation format could act as a *lingua franca* between various tools and resources that assist in the design and implementation process.
- **Flexibility**: The ontology conceptualisation should be flexible enough to be mapped to current requirement specification environments and primitives. It should facilitate the integration of interfaces for capturing design requirements and the design issues.
- **Traceability**: It must allow forward and backward traceability of final designs as well as initial design requirements from design intents and issues.

5 The Knowledge Representation Framework

A high level representation of the architecture implemented for integrating the requirements and the issues is illustrated in Fig. 1. The core components of the

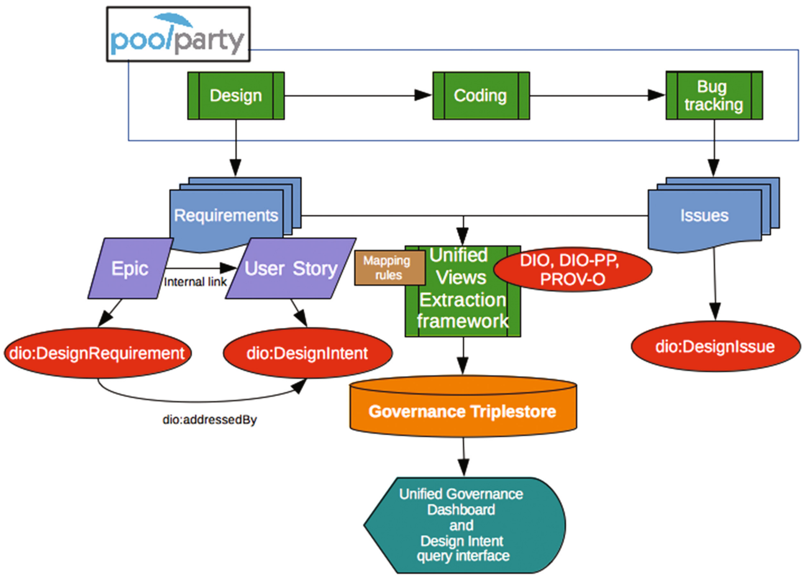


Fig. 1. The knowledge representation and extraction architecture

architecture are the ontologies, the Unified Views extraction pipeline and the unified governance dashboard that provides visual analytics for the results of searches over the integrated datasets.

The DIO content design pattern provides a minimalistic abstraction and defines conceptual, generic entities for the modelling of semantically enriched knowledge required to capture the intents or rationale behind the design of an artifact. DIO is a domain agnostic pattern and most domain specific design rationale models will specialise from it.

Figure 2 illustrates the graphical representation of DIO. It depicts the entities defined for the pattern and their relationships with entities from PROV-O[8].

The axiomatisation of a `DesignIntentArtifact` can be represented in OWL using the Description Logic [1] notation as,

$$\begin{aligned} \texttt{DesignIntentArtifact} \sqsubseteq & ((\exists \texttt{wasAttributedTo.DesignIntent}) \\ & \sqcap (= 1 \texttt{wasAttributedTo.Agent}) \sqcap (\geq 1 \texttt{description}) \\ & \sqcap (= 1 \texttt{version}) \sqcap (= 1 \texttt{generatedAtTime})) \end{aligned}$$

[8] http://www.w3.org/ns/prov-o.

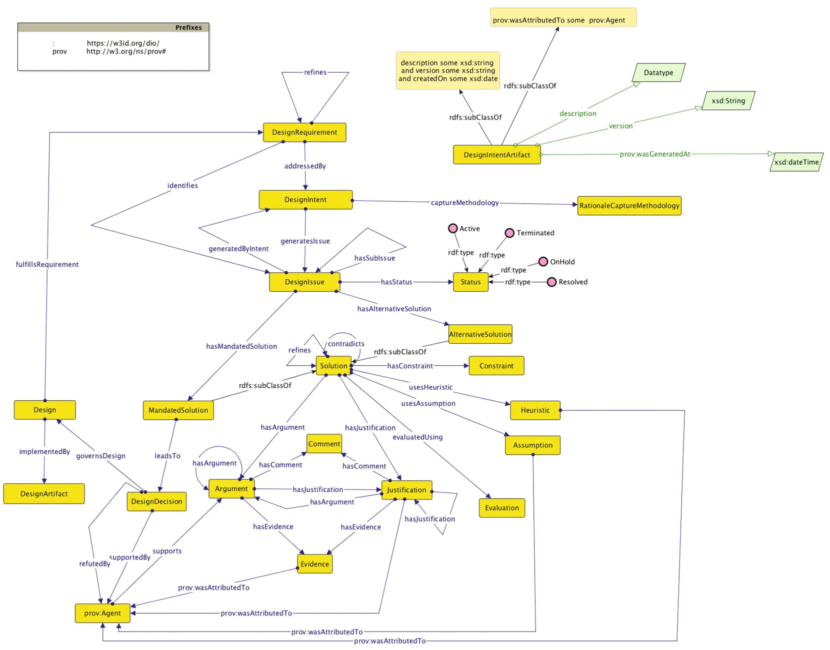


Fig. 2. Graphical representation of DIO

The axiomatisation for a `DesignIssue` captures the intuition that a design issue can have several alternative solution, but just one mandated solution.

$$\begin{aligned} \texttt{DesignIssue} \sqsubseteq (&\texttt{DesignIntentArtifact} \\ &\sqcap(\forall \texttt{hasAlternativeSolution.AlternativeSolution}) \\ &\sqcap(\forall \texttt{hasMandatedSolution.MandatedSolution}) \\ &\sqcap(= 1 \texttt{hasMandatedSolution.MandatedSolution})) \end{aligned}$$

For further details, the interested reader is referred to [12].

As DIO is a general purpose design intent ontology, it does not capture attributes specific to PoolParty development. To bridge this gap, we define DIO-PP[9] - an extension to DIO that is specifically aimed at capturing conceptualisations from Confluence and JIRA. Besides DIO-PP, we also define bespoke mappings to DIO entities, for the data recorded during requirements capture.

5.1 Mapping Epic to Design Requirements

The SWC Confluence environment consist of two main parts: The Epic and the User story.

[9] https://w3id.org/diopp.

The Epic captures the high level description of the requirements and the stakeholders affected or involved in addressing the requirements. An Epic consists of: *goal* (the high level explanation of the requirement), *requirements*, the set of user stories and *stakeholders*. The mapping[10] from the informal requirement specification in Epic to DIO is done as follows: Every *Epic*, corresponds to a `dio:DesignRequirement`.

$$Epic \rightarrow \texttt{dio} : \texttt{DesignRequirement}$$

The epic's *title* is mapped as a `dc:title` for the `dio:DesignRequirement`

$$\begin{aligned} Epic/title \rightarrow (&\texttt{dio} : \texttt{DesignRequirement} \\ &\texttt{dc} : \texttt{title}\ \texttt{xsd} : \texttt{String}) \end{aligned}$$

The epic's *goal* is mapped as a `dc:description` for the `dio:DesignRequirement`

$$\begin{aligned} Epic/goal \rightarrow (&\texttt{dio} : \texttt{DesignRequirement} \\ &\texttt{dc} : \texttt{description}\ \texttt{xsd} : \texttt{String}) \end{aligned}$$

5.2 Mapping User Story to Design Intent

Each Epic includes a set of requirements, each of which are represented as a "User story" in Confluence. Each user story captures the design intent. Every *User story*, corresponds to a `dio:DesignIntent`.

$$User\ story \rightarrow \texttt{dio} : \texttt{DesignIntent}$$

The user story's *title* is mapped as a `dc:title` for the `dio:DesignIntent`

$$\begin{aligned} User\ story/title \rightarrow (&\texttt{dio} : \texttt{DesignIntent} \\ &\texttt{dc} : \texttt{title}\ \texttt{xsd} : \texttt{String}) \end{aligned}$$

The user story's *description* is mapped as a `dc:description` for the `dio:DesignIntent`

$$\begin{aligned} User\ story/description \rightarrow (&\texttt{dio} : \texttt{DesignIntent} \\ &\texttt{dc} : \texttt{description}\ \texttt{xsd} : \texttt{String}) \end{aligned}$$

The use story may include a link to a JIRA issue, if the issue relates to certain aspects of the requirement. This is captured using the predicate `dio:generatesIssue`

$$\begin{aligned} User\ story/jira\ issue \rightarrow (&\texttt{dio} : \texttt{DesignIntent} \\ &\texttt{dio} : \texttt{generatesIssue}\ \texttt{dio} : \texttt{DesignIssue}) \end{aligned}$$

[10] indicated by "→".

The user story includes part of the solution, that addresses the requirement and is the most significant part of the mapping. The user story captures the following elements that contribute to an `AlternativeSolution` and `MandatedSolution` for a `DesignIssue` in DIO: *Preconditions*, *Detailed Description*, *Affected Components*, *Acceptance criteria and test scenarios*, *Variations*, *Comments*. The DIO-PP ontology conceptualises these elements.

$$\begin{aligned}\texttt{diopp}:\texttt{Precondition} &\sqsubseteq \texttt{dio}:\texttt{Assumption}\\ \texttt{diopp}:\texttt{AffectedComponent} &\sqsubseteq \texttt{dio}:\texttt{Heuristic}\\ \texttt{diopp}:\texttt{AcceptanceCriteria} &\sqsubseteq \texttt{dio}:\texttt{Evaluation}\\ \texttt{dio}:\texttt{Comment} &\sqsubseteq \texttt{dio}:\texttt{Argument}\\ &\sqcup \texttt{dio}:\texttt{Justification}\end{aligned}$$

The user story's *Detailed Description* element is mapped as a `dc:description` for the `dio:AlternativeSolution`

$$\begin{aligned}&\mathit{User\ story/detailed\ description} \rightarrow\\ (&\texttt{dio}:\texttt{AlternativeSolution}\ \texttt{dc}:\texttt{description}\\ &\texttt{xsd}:\texttt{String})\end{aligned}$$

5.3 Mapping JIRA Design Issues

JIRA is used as the platform by PoolParty customers to highlight design issues related to the software and request new features. Typical properties that need to be ascribed to a design issue documented in JIRA are the issue type, `IssueType`, its status, `Status`, its priority, `PriorityType`, its resolution status, `ResolutionType` and the various agents that are associated with the issue: `Watcher`,`Assignee` and `Reporter`. The DIO-PP ontology defines these entities. The axiomatisation of a `DesignIssue` for PPT can be represented as,

$$\begin{aligned}\texttt{DesignIssue} \sqsubseteq ((&= 1\,\texttt{isOfIssueType.IssueType})\sqcap\\ (&= 1\,\texttt{hasVersionType.VersionType})\sqcap\\ (&= 1\,\texttt{hasPriority.PriorityType})\sqcap\\ (&= 1\,\texttt{hasStatus.IssueStatusType})\sqcap\\ (&= 1\,\texttt{hasResolution.IssueResolution})\sqcap\\ (&= 1\,\texttt{hasReporter.Agent})\sqcap\\ (&= 1\,\texttt{hasAssignee.Agent})\sqcap\\ (&\geq 1\,\texttt{hasAffectedComponent.Component})\sqcap\\ (&\geq 1\,\texttt{hasWatcher.Agent})\sqcap\\ (&\geq 1\,\texttt{hasEnvironment.}\top)\sqcap\\ (&\leq 1\,\texttt{updateDate}))\end{aligned}$$

6 Data Extraction Workflow

As highlighted in Sect. 3, SWC collects the requirements and issues for each version of PPT in the designated Confluence space and JIRA respectively. This data needs to be extracted and semantically annotated with DIO and DIO-PP. To perform the data extraction and conversion process, we provide three contributions: (1) an extraction tool which connects to the Confluence and JIRA instances for PPT and converts the contained data into RDF. (2) a Data Processing Unit (DPU) for UnifiedViews[11] which wraps the extraction tool so that it can be included into custom data processing pipelines. (3) a UnifiedViews pipeline, which encompasses data extraction, data annotation using a PPT thesaurus and loading the annotated data into a remote Virtuosos triple store.

6.1 Extraction Tool

The extraction tool connects to the Confluence and JIRA REST APIs and retrieves (meta-)data that should be converted into the RDF output format using DIO-PP. While data from JIRA can be retrieved in structured JSON format, certain information we need to formalize must be retrieved by parsing HTML pages from the Confluence installation. The tool consists of two independent extraction components - one extracts data from Confluence, the other one from JIRA. The Confluence extractor iterates through all pages in a hierarchy below a page named "Requirements PP" in the PPT development space, retrieves the JSON metadata (e.g., history information or creation date) of these pages and parses the HTML content. This data is then used to build an in-memory object model which is serialized as RDF data. Extraction and conversion of JIRA data works in a similar way, except that no HTML parsing needs to be performed because the API provides the information in JSON format.

6.2 Integration into a Processing Pipeline

As the extraction tool produces an RDF document, we can leverage the potential of Linked Data to enrich the extracted data with information coming from other sources. Specifically, we enrich data from Confluence and JIRA with annotation (tagging) information coming from an enterprise thesaurus, allowing to, e.g., improve the search functionality. For the Unified Governance use case we show how to add tagging information by providing a pipeline for the UnifiedViews tool, which is illustrated in Fig. 3. The pipeline consists of seven DPUs: the extraction tool (red box) retrieves data from Confluence and JIRA in RDF format. It is then filtered (blue box below) so that it only contains text literals which serve as an input for annotating. This annotation stage (box below filtering step) connects to the PoolParty Extractor annotation API and returns a RDF graph containing the identified annotations for each resource. Afterwards, the original

[11] An Extract-Transform-Load (ETL) tool focused on processing RDF data, http://www.unifiedviews.eu/.

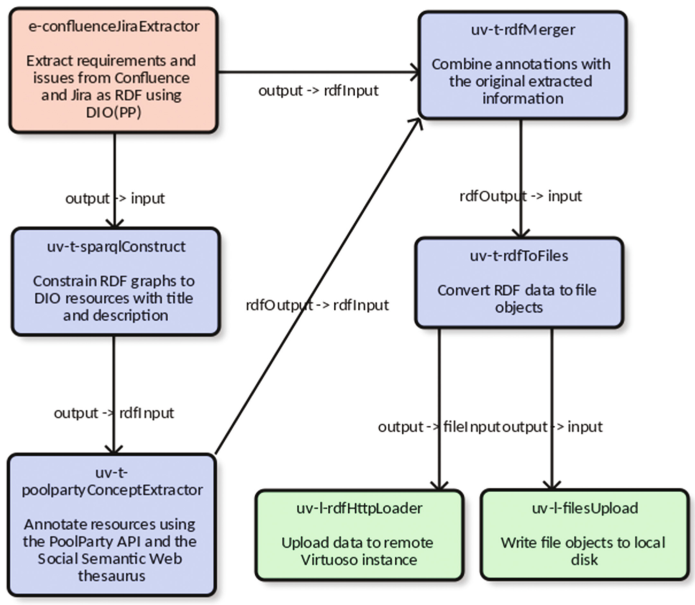

e-confluenceJiraExtractor
Extract requirements and issues from Confluence and Jira as RDF using DIO(PP)
output -> rdfInput
uv-t-rdfMerger
Combine annotations with the original extracted information
output -> input
uv-t-sparqlConstruct
Constrain RDF graphs to DIO resources with title and description
rdfOutput -> rdfInput
rdfOutput -> input
uv-t-rdfToFiles
Convert RDF data to file objects
output -> rdfInput
uv-t-poolpartyConceptExtractor
Annotate resources using the PoolParty API and the Social Semantic Web thesaurus
output -> fileInput
output -> input
uv-l-rdfHttpLoader
Upload data to remote Virtuoso instance
uv-l-filesUpload
Write file objects to local disk

Fig. 3. UnifiedViews extraction pipeline

extracted data is combined with the filtered annotated data and both written to the local file system as well as uploaded to a Virtuoso server.

We ran the extraction process on a subset of the requirements (5394) and issues (184534). This resulted in 215401 triples. The extraction is now run at regular intervals with the resulting triples added to the triple store.

7 Evaluation

In order to assess the benefits of using semantically annotated and integrated requirements engineering datasets for unified governance, we carried out an evaluation that compared the time taken in retrieving requirements and issues from Confluence and JIRA against the time taken when using the graph-search powered unified governance dashboard. Our evaluation involved six participants (members of the consulting team at SWC) with a background in software engineering. All the recruited personnel had reasonable experience in the use of Confluence, JIRA and ontology interpretation.

Confluence and JIRA provide a keyword-driven search facility to retrieve information specific to requirements and issues respectively. The results are returned as natural text. The unified governance dashboard is based on the entities captured in DIO, whose design is governed by competency questions [12], that cover the spectrum of requirements engineering knowledge which should be recorded. In order to measure the accuracy, ease and efficiency of retrieving structured and semantically enriched requirements engineering datasets against the information retrieved through Confluence and JIRA search, our evaluation[12] consisted of two main tasks:

– Task 1: Searching Confluence and JIRA to retrieve requirements and answer the corresponding competency questions.
– Task 2: Using the unified governance dashboard to answer the same competency questions.

Six competency questions were designed using inputs from a wide variety of use cases routinely encountered by the customers and developers of PPT. These questions were chosen after consultation with social scientists, developers and consultants of PPT. In Table 2 we present four[13] of these questions, along with the use case and the mapped construct from the DIO and DIOPP ontology which were used to annotate the highlighted entities in the question. Questions 1–3 are specifically targeted towards retrieving information annotated with DIO and DIOPP while question 4 is based on doing a full text search within both the systems.

Task 1 involved the following steps:

– Reading and interpreting the competency questions.
– Searching Confluence and JIRA.
– Retrieving information leading to answers to the questions.
– Documenting the search strategy used in retrieving the answers and analysing the results.

Task 2 involved the following steps:

– Reading and interpreting the competency questions.
– Using the facet browsing and full text search features of the unified governance dashboard to retrieve potential answers.
– Documenting the search strategy used in retrieving the answers and analysing the results.

The volunteers were divided into two groups, A and B. Group A executed task 1, followed by task 2, while for Group B, it was the other way around. All volunteers were provided with Turtle serialisation of the DIO ontology a week before the evaluation. On the day of the evaluation they were provided with the six competency questions, access to Confluence and JIRA and access to the unified governance dashboard. The screenshots of the dashboard interfaces are presented in Fig. 4. In order to maintain consistency across volunteers, they were advised to answer the questions in the prescribed order and the evaluation was undertaken under supervision. For each of the tasks and for each question, the following observations were recorded: (1) The time taken to answer each question in task 1 and 2. This was the total time taken for searching and analysing the results. (2) The answers for each question in task 1 and 2. (3) The strategy used to answer the question.

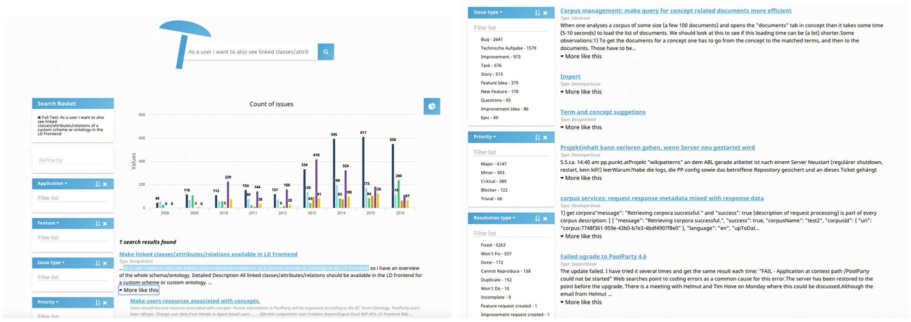
[12] All evaluation data including participants strategies, timings and the resulting timing analysis graphs has been made available at http://goo.gl/Khlaaf.

[13] Due to space constraint, we report only four here. All six questions can be found in the participant links in the Google doc containing links to the evaluation data.

Table 2. Competency questions used for evaluation

Question	Use case	Mapped DIO/DIOPP concept
List all **stories** that have affected version PoolParty 5.3.1 that were requested by a customer.	It often happens that customers request specific features that should be added or improved in PPT	User story → **dio:DesignIntent**
Which feature caused most **bug reports** in release 5.5.0 (affected Version 5.5.0)?	This goes towards release profiling: finding out which features are weak points of a certain release	Bug reports → **dio:DesignIssue**
What has changed in the last **three years** (2014, 2015, 2016) when it comes to bugs and tasks in the PoolParty support project?	To determine if we are getting better at serving our customers, it is useful to know how the interaction of the customers with our ticketing, system develops over time.	Bug reporting date → **prov:wasGeneratedAt**
Which PoolParty version (starting from PP 2.8 and including bug fix, releases in the respective **minor** or **major** release (e.g. count 5.3.1 for,5.3)) was the one affected with the most bugs that **were classified** as,"blocker", and how would you interpret this?	In order to know if we get better in our development and support, processes, it is useful to know if we are encountering more or less, critical issues than before.	Bug status→ **dio:IssueStatusType**, Bug type → **dio:IssueType**

Figure 5(a – d) illustrates the time taken to answer each of the six questions by four of the participants across the two tasks, while Fig. 5(d) illustrates the average time taken, after discarding the extreme outliers for each participant. Our evaluation shows that the overall time taken to search the integrated datasets annotated with DIO and DIOPP and answer the six competency questions across all participants is 50% less when compared to the time taken to search Confluence and JIRA for the same questions. This is a significant and impressive saving in resources for the SWC, where consultants and developers need to manage a large number of customer requirements and issues on a regular basis. The evaluation also highlighted the weakness of the full text search implementation used in the unified governance dashboard, where for atleast three of the participants, to search answers to questions 5 and 6 took significantly longer as compared to using the full text search provided by Confluence and JIRA. As we had asked the participants to record their search strategies and use of the interface, we were also able to identify pitfalls and bottlenecks in the current design of the unified governance dashboard as an additional evaluation result.



As a user i want to also see linked classes/attrit

Search Basket

✖ Full Text: As a user i want to also see linked classes/attributes/relations of a custom scheme or ontology in the LD Frontend

Refine by

Application ▾ — Filter list

Feature ▾ — Filter list

Issue type ▾ — Filter list

Priority ▾ — Filter list

1 search results found

Make linked classes/attributes/relations available in LD Frontend
Type: DesignIntent
...*As a user i want to also see linked classes/attributes/relations of a custom scheme or ontology in the LD Frontend* so i have an overview of the whole schema/ontology. Detailed Description All linked classes/attributes/relations should be available in the LD frontend for a custom scheme or custom ontology. ...
▴ More like this

Make users resources associated with concepts.
Users should become resource associated with concepts. Person information in PoolParty will be organized according to the DC Terms Ontology. PoolParty users have rdf:type. Change user data from literals to Agent-based users. „ ... Affected components User Creation Import/Export Excel RDF APIs LD Frontend Wiki ...

Issue type ▾
Filter list
- Bug - 2641
- Technische Aufgabe - 1579
- Improvement - 973
- Task - 676
- Story - 515
- Feature Idea - 379
- New Feature - 170
- Questions - 93
- Improvement Idea - 86
- Epic - 49

Priority ▾
Filter list
- Major - 6147
- Minor - 503
- Critical - 389
- Blocker - 122
- Trivial - 66

Resolution type ▾
Filter list
- Fixed - 5263
- Won't Fix - 557
- Done - 172
- Cannot Reproduce - 158
- Duplicate - 152
- Won't Do - 10
- Incomplete - 9
- Feature request created - 1
- Improvement request created - 1

Corpus management: make query for concept related documents more efficient
Type: IdeaIssue
When one analyses a corpus of some size (a few 100 documents) and opens the "documents" tab in concept then it takes some time (5-10 seconds) to load the list of documents. We should look at this to see if this loading time can be (a lot) shorter.Some observations:1) To get the documents for a concept one has to go from the concept to the matched terms, and then to the documents. Those have to be...
▾ More like this

Import
Type: DeveloperIssue
▾ More like this

Term and concept suggetions
Type: DesignIntent
▾ More like this

Projektinhalt kann verloren gehen, wenn Server neu gestartet wird
Type: DeveloperIssue
5.5.ca. 14:40 am pp.punkt.atProjekt "wikipatterns" an dem ABL gerade arbeitet ist nach einem Server Neustart (regulärer shutdown, restart, kein kill!) leerWarum?habe die logs, die PP config sowie das betroffene Repository gesichert und an dieses Ticket gehängt
▾ More like this

corpus services: request response metadata mixed with response data
Type: DeveloperIssue
1) get corpora"message": "Retrieving corpora successful." and "success": true (description of request processing) is part of every corpus description: [{ "message": "Retrieving corpora successful.", "success": true, "corpusName": "test2", "corpusId": { "uri": "corpus:7748f361-959e-43b0-b7e3-4bdf4907f8e0" }, "language": "en", "upToDat...
▾ More like this

Failed ugrade to PoolParty 4.6
Type: SupportIssue
The update failed. I have tried it several times and get the same result each time: "FAIL - Application at context path /PoolParty could not be started" Web searches point to coding errors as a common cause for this error.The server has been restored to the point before the upgrade. There is a meeting with Helmut and Tim Hove on Monday where this could be discussed.Although the email from Helmut ...
▾ More like this

Fig. 4. Unified governance dashboard with full text search and faceted browsing

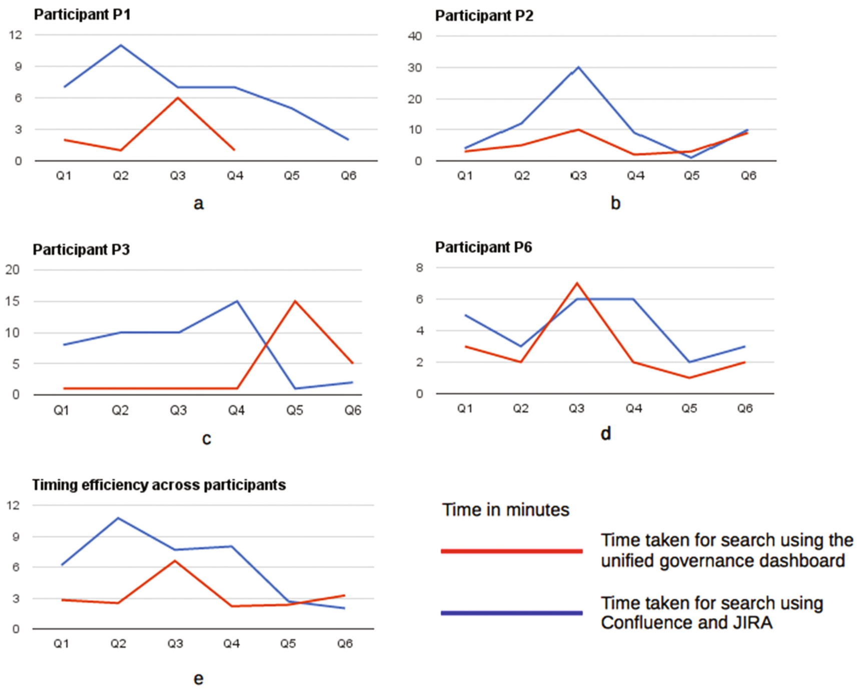


Fig. 5. Timings recorded by four participants and the average time efficiency for all participants and all questions.

8 Conclusions

In this paper, we have proposed a framework that exploits the DIO ontology for integrating the requirements and issues arising during software design, implementation and maintenance. We have evaluated our approach on an industrial software engineering case study for the PoolParty Thesaurus server development and maintenance. We have shown how a minimalistic extension of DIO can be utilised to associate and extract software design rationales with requirements recorded in Confluence and issues submitted in JIRA for PPT. Our evaluation shows an impressive 50% reduction in the time taken to search semantically annotated and integrated datasets when compared to the search provided by Confluence and JIRA. The evaluation also revealed the limitation of the full text search feature currently implemented in the unified governance dashboard. Improving this will be the focus of our immediate future work. We have presented an end-to-end solution that provides significant improvements in productivity, agility and efficiency in software development environments.

In future we aim to build on the results emerging from periodic and detailed empirical evaluation carried out on the semantic search interface for queries with various levels of complexity and implement an ontology-driven, design-intent powered, self learning system for software engineering environments.

Acknowledgments. We are grateful to Prof. Jeremy Gibbons (University of Oxford) for his insightful comments on early drafts of the paper. The research presented in this paper has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 644055, the ALIGNED project (www.aligned-project.eu).

References

1. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press, New York (2003)
2. Bani-Salameh, H., Jeffery, C.: Collaborative and social development environments: a literature review. Int. J. Comput. Appl. Technol. **49**(2), 89–103 (2014)
3. Bracewell, R., Wallace, K., Moss, M., Knott, D.: Capturing design rationale. Comput. Aided Des. **41**(3), 173–186 (2009). Computer Support for Conceptual Design
4. Burge, J.E., Carroll, J.M., McCall, R., Mistrk, I.: Rationale-Based Software Engineering, 1st edn. Springer, Heidelberg (2008)
5. Conklin, E.J., Yakemovic, K.C.B.: A process-oriented approach to design rationale. Hum.-Comput. Interact. **6**(3), 357–391 (1991)
6. Medeiros, A.P., Schwabe, D., Feijó, B.: Kuaba Ontology: Design Rationale Representation and Reuse in Model-Based Designs. In: Delcambre, L., Kop, C., Mayr, H.C., Mylopoulos, J., Pastor, O. (eds.) ER 2005. LNCS, vol. 3716, pp. 241–255. Springer, Heidelberg (2005). doi:10.1007/11568322_16
7. Gruber, T.R.: Toward principles for the design of ontologies used for knowledge sharing. Int. J. Hum. Comput. Stud. **43**(5–6), 907–928 (1995)
8. Horner, J., Atwood, M.E.: Design rationale: the rationale and the barriers. In: Proceedings of the 4th Nordic Conference on Human-Computer Interaction: Changing Roles, NordiCHI 2006, pp. 341–350. ACM (2006)
9. Kunz, W., Rittel, H.W.J., Messrs, W., Dehlinger, H., Mann, T., Protzen, J.J.: Issues as elements of information systems. Technical report (1970)
10. Kuofie, E.J.: Radex: a rationale-based ontology for aerospace design explanation (2010)
11. Lee, J.: Design rationale systems: understanding the issues. IEEE Expert **12**(3), 78–85 (1997)

12. Solanki, M.: DIO: a pattern for capturing the intents underlying designs. In: Proceedings of the 6th Workshop on Ontology and Semantic Web Patterns (WOP 2015), vol. 1461. CEUR-WS.org (2015)
13. Tempich, C., Pinto, H.S., Sure, Y., Staab, S.: An argumentation ontology for DIstributed, loosely-controlled and evolvInG engineering processes of oNTologies (DILIGENT). In: Gómez-Pérez, A., Euzenat, J. (eds.) ESWC 2005. LNCS, vol. 3532, pp. 241–256. Springer, Heidelberg (2005). doi:10.1007/11431053_17
14. Zhang, Y., Luo, X., Li, J., Buis, J.J.: A semantic representation model for design rationale of products. Adv. Eng. Inform. **27**(1), 13–26 (2013)