SemTrace: Semantic Requirements Tracing Using Explicit Requirement Knowledge

Thomas Moser and Stefan Biffl

Christian Doppler Laboratory SE-Flex
Vienna University of Technology
Favoritenstrasse 9-11/188, 1040 Vienna, Austria
+43 (1) 58801 188051
{thomas.moser, stefan.biffl}@tuwien.ac.at

Abstract. In the software engineering (SE) domain the EU challenge on semantic systems and services motivates better support of knowledge creation as well as better access of SE stakeholders to the knowledge they need to perform their activities. Application Lifecycle Management (ALM) is the coordination of development lifecycle activities by process automation, management of relationships between artifacts, and reporting on development progress. ALM focuses on the integration of knowledge between the tools that support SE roles and thus seems particularly well suited to leverage benefits from advanced semantic technologies and services that overcome limitations from semantic gaps in today's heterogeneous SE platforms. In this paper, we present a semantic requirements tracing approach (SemTrace), which makes the implicit interdependencies between requirements and other artifacts explicit. We evaluate the proposed approach in a change impact analysis use case. Initial results show that the SemTrace approach allows for a flexibly identification of semantically connected artifacts in SE projects..

Keywords: Requirement Tracing, Requirement Knowledge, Soft-Links.

1 Introduction

The effective and efficient creation of high-quality software products is a major goal of software development organizations where teams coming from business and technical domains collaborate to produce knowledge and artifacts. A major challenge in distributed software development is the integration of the knowledge scattered over processes, tools, and people to facilitate effective and efficient coordination of activities along the development lifecycle. Application Lifecycle Management (ALM) is defined as "*the coordination of development life-cycle activities, including requirements, modeling, development, build, and testing*" [14].

Software engineering organizations can radically improve their project effectiveness and efficiency if artifacts can be exchanged and understood both by the human roles involved and machines, similar to advanced semantic technologies

and services that facilitate the creation and usage of machine-readable semantic annotation of information. The EU research FP7 motivates a key strategic research challenge[1] in the area of semantic systems/technology development to deal with the "*growing load and diversity of information and content*" which particularly applies to SE projects where semantic support for data analysis and reasoning would improve data and knowledge exchange among distributed project teams and promises more effective collaboration.

Current ALM approaches aim at solving the technical challenges of accessing data in different tools to provide a common (relational) data model for process automation, tracing of relationships between artifacts, reporting and progress analysis beyond the data available in a single tool. ALM has been successful in the context of semantically homogeneous platforms, e.g., IBM Jazz[2] or Collabnet[3], but faces major obstacles due to semantic gaps in the integration and evolution of heterogeneous platforms, tool sets, and organizational contexts. Typical semantic gaps are multitudes of proprietary tools with their own proprietary notations, data that is scattered over multiple repositories (e.g., SVN, Mailing Lists, Bug Tracking Systems), and the fact that often just syntactic matching is used for detecting equalities between artifacts. These semantic gaps cannot be overcome efficiently with the limited syntactic approaches of current ALM approaches.

Semantic tracing builds on the semantic connections in the ALM data repository for more effective, efficient, and robust tracing of artifacts along the life cycle (e.g., to find all artifacts that help implement or test a certain requirement for change impact analysis) than traditional ALM approaches that do not use semantic knowledge. Effective tracing is demanded by SE standards (e.g., SEI's CMM-I[4]) and a key enabler for management, design, and quality assurance activities that have to understand the relationships between artifacts along the life cycle, but currently not sufficiently well supported. Research target is to use the ontology as a means for semantic tracing in order to derive dependencies between work products (requirements, test cases, source code) from knowledge and explicit user input.

In this paper we present a semantic requirements tracing approach (SemTrace), which makes the implicit interdependencies between requirements and other artifacts explicit. In the discussed use case, a change impact analysis is done for a changing requirement to find out which issues and developers are affected by the change request. This information can be used to mark all dependent artifacts for review and to contact all involved developers automatically. Furthermore it allows better estimates for the costs of the changes. Major results of the evaluation are that the change impact analysis could be defined without prior knowledge regarding the relations between the used engineering concepts, besides the fact that such a relation exists. Furthermore, the possibility to de-

[1] http://cordis.europa.eu/fp7/ict/content-knowledge/fp7_en.html
[2] http://www.jazz.net (IBM Jazz collaboration Platform)
[3] http://www.collabnet.net (Collabnet ALM platform)
[4] http://www.sei.cmu.edu/cmmi

fine and use semantic meta information to model relations between engineering concepts is a very flexible mechanism, which can also be used for other advanced applications, like for example automatically linking commits to related issues.

The remainder of this paper is structured as follows: Section 2 summarizes related work on requirements tracing and on semantic systems integration. Section 3 identifies the research issues and presents the use case, while section 4 presents the SemTrace approach. Finally, section 5 discusses the results of the initial evaluation, concludes the paper and identifies further work.

2 Related Work

This section summarizes related work on requirements tracing and on semantic systems integration.

2.1 Requirements Tracing

Traceability is demanded by multiple software engineering standards like CMMI [12] and provides SE project participants with explicit knowledge about relationships/dependencies between artifacts existing in the project. Usually, project participants have to capture these dependencies manually, e.g. in matrices [10] and current ALM platforms support this manual capture of dependencies. Unfortunately, there is the challenge of significant effort to manually provide full traceability of all requirements to all other artifacts [5]. Thus, automated approaches have been developed to capture dependencies based on syntactical identity (e.g. keyword-matching as in information retrieval approaches), e.g. [1][2][4]. These automated approaches do not capture dependencies completely, because they cannot capture dependencies between semantically related artifacts without syntactic identity. Incomplete capture of dependencies has negative effects (higher efforts) on important SE activities like change impact analysis and consistency checking [7].

We have recently proposed innovative approaches for "value-based requirements tracing" [8], and tool support for tracing across tool borders (by integrating a requirements management tool and the Eclipse IDE) [7]. In these initial pilot studies we provided the project participants with semantic clues, which improved their tracing efficiency significantly (between 60% and 85% savings of effort). While the semantic clues were provided with extra effort by the research team in the pilot studies, the SemTrace approach bridges the semantic gap mentioned above and provides access to semantic knowledge across tool borders as part of its concept. As the explicit knowledge is usable both for humans and machines, SemTrace provides better foundations for automated tracing approaches, makes tracing cognitive less challenging and takes considerably less effort.

2.2 Semantic Systems Integration

A major problem that occurs when different systems have to be integrated is data heterogeneity. According to Cruz et al. [3] the mismatch in data representation

can be classified into three categories: syntactic, schematic and semantic. While technical integration focuses more on syntactical and to some extent schematic issues and on providing a common message format to make communication between different systems possible, semantic integration focuses on the meaning of the data exchanged between those systems. This includes both schematic and semantic factors. Effective and efficient cooperation between different tools or systems is only possible if the semantics of source and target system are compatible [13]. Yet this compatibility is not given in most organizations, because applications are not developed with interoperability in mind and because new systems are added as a result of mergers or acquisitions [6]. Semantic integration tries to solve the problem of semantic heterogeneity by providing an intermediate layer that automatically transforms data between the involved systems.

The Engineering Knowledge Base (EKB) is a semantic integration approach for tool and data integration in the engineering domain proposed by Moser [11]. The three main features of an EKB are: 1) data integration using mappings between different engineering concepts; 2) transformations between different engineering concepts utilizing these mappings; and 3) advanced applications building upon these foundations. The EKB framework was developed for engineering tool integration. By providing an effective and efficient semantic integration layer it simplifies the process of engineering. Especially tasks that span different domains, where experts with different technical background have to cooperate can be performed with less effort if all tools are integrated semantically.

3 Research Issues & Use Case

As engineering processes become more agile and customers are more closely integrated into the development process the focus of requirement engineering changes. The precise definition of the requirements up-front becomes less important than the consistent management of the requirements. Part of this consistent management is the traceability from a requirement to dependent engineering artifacts and vice versa. There are four types of traces [9]:

Forward from Requirements Traces from requirements to dependent engineering artifacts. These traces are necessary to evaluate which changes have to be performed if a requirement is updated, so they are used for change impact analysis.

Backward to Requirements Traces from engineering artifacts to requirements. These are used to make sure that for every part of the developed system a requirement exists and no superfluous work is done.

Forward to Requirements Traces from high level project descriptions or design documents to derived requirements. If stakeholders change high level system goals these traces can be used to determine all affected requirements and finally all affected engineering artifacts.

Backward from Requirements Traces from requirements to high level project descriptions and design documents. These traces are important to eval-

uate the quality of requirements, as the business needs leading to the respective requirements can be identified.

It is possible to derive backward to requirements traces from forward from requirements traces and vice versa if the tracing system has full information about all trace links. This is done by automatically establishing links in both directions if a link is created in one direction. Therefore many projects define trace links only in one direction, relying on the possibility to derive links in the other direction if they are needed. Likewise forward to requirements can be derived from backward from requirements and vice versa. These use cases especially focuses on the backward to and forward from requirements traces. The advantages of managing the implicit dependencies between requirements and engineering artifacts explicitly are on the one hand better decision support in every phase of development, because crucial information can be found directly and on the other hand easier change management, as affected artifacts can be identified with the help of trace links. Some process quality standards like CMMI demand the establishment of a structured requirement management and tracing approach.

Despite all these advantages only few engineering projects use requirement tracing. The reason is the huge effort needed to capture and manage requirements and trace links to engineering artifacts. Although there are specialized tools for requirement tracing their major drawback is the missing integration with other development tools. Furthermore, when requirement tracing is done with a specialized tool the trace links have to be established in a dedicated work step. This is undesirable, because it has to be either done after the developer finished his work or parallel to the work. The first approach is impractical, because the effort for trace generation is considerably higher if tracing is done in an extra step after the development task is finished. The second approach reduces the time consumption for trace generation, but might break the workflow of a developer as he has to constantly switch tools during development. Integrated requirement tracing is an alternative, but is hard to accomplish in a heterogeneous environment. Therefore requirement tracing is often done ad-hoc, hindering the precise measurement of costs or benefits [8].

In the "Change Impact Analysis for Requirement Changes" use case the integrated environment with semantic integration support proposed in this paper is used to extract requirement traces from available information. In many software engineering projects tracing information is captured in a structured and well defined, but informal way, which can be understood by humans, but which is not usable for automatic processing. By defining the semantics of tool data models including the tracing information it is possible to automate parts of a change impact analysis.

Figure 1 gives an overview about this use case. If a stakeholder files a change request (1) the requirement engineer has to identify the affected requirements using a requirement management tool (2). Then he can conduct a change impact analysis with the help of the interaction and workflow component (3). The change impact analysis is done for the affected requirements using the virtual common

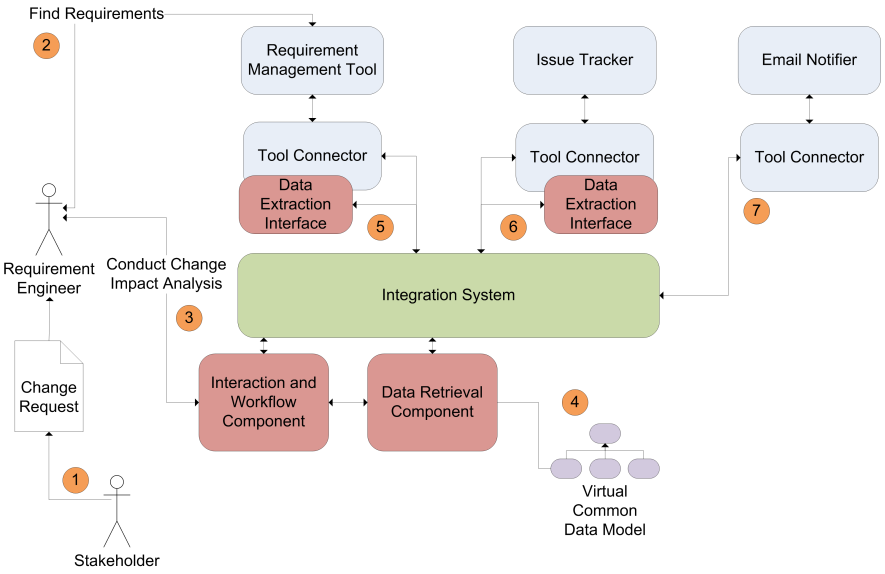

Fig. 1. Overview of the "Change Impact Analysis for Requirement Changes" use case.

data model (4). As a first step data is retrieved from the requirement management tool (5) and by using the semantic information in the virtual common data model all dependent issues are identified. So traces of the type "forward from requirements" are used to navigate from the requirements to dependent issues. For this purpose the links need not actually be available in this direction, as the integration system can use traces of the type "backward to requirements" to derive the trace links in the other direction. In this use case the link between the trace link between issue and requirement is defined by a reference to the addressed requirement in the issue description. Such informal or semi-formal forms of trace information are found in most engineering projects and can only be used if semantic information about the meaning of these links is available. The affected issues are retrieved from the integrated issue tracker (6) and separated into three groups:

Open Issues These issues have to be marked for review. They are likely to change as the underlying requirement has changed. After the issue is redefined according to the new requirements and its estimates updated the change impact can be evaluated.

In Progress Issues, which are currently in progress are the most critical group. All development team members, who are assigned to this issue have to be notified about the requirement change. They have to update the issue's definition and estimates, before they can continue with their work. Furthermore they need to contact the requirement engineer undertaking the change impact analysis and provide their opinion about the amount of work necessary to perform the

changes. The integration of the affected team members into the change impact analysis is critical for its quality and correctness

Resolved and Closed Issues All issues, which are already finished have to be reopened and marked for review, as the underlying requirement has changed. The developers, which are assigned to these issues are notified that an already finished piece of work has to be reevaluated. Furthermore, they have to provide their estimates for the amount of work necessary to perform the requested changes.

To notify the affected team members again semantic information has to be used. The issues contain information about the assignee, which has to be extracted and mapped to a member of the development team. Then another mapping between the developer and its contact information has to be used to determine the recipient of the notification. An integrated notification component, like an email connector (7) sends the notifications. Finally the requirement engineer, which started the change impact analysis process is presented with a report containing all the information, which could be gathered automatically and a list of all persons, who have to provide manual feedback.

This semi-automatic form of change impact analysis for requirement changes provides a high quality result without the need to manually search and evaluate the trace links between requirements, issues and developers. Furthermore, it provides the possibility to inform all affected team members during the process and to automatically perform necessary project management steps in the issue tracker, like reopening already finished issues.

The goal of the "Change Impact Analysis for Requirement Changes" use case is to show that the proposed EKB based semantic integration solution for (software+) engineering can be facilitated to perform a difficult and complex software engineering task, like requirement tracing. Advanced applications like change impact analysis can be built based upon the semantic integration framework, which help to generate better estimates. This use case is designed to underline the importance of efficient cooperation between different team members to perform complex tasks, like the reevaluation of parts of the system after a requirement change and the crucial role of effective tool support during this process.

4 The SemTrace Approach

In the described use case, the semantic integration infrastructure is used to conduct requirement tracing, an advanced software engineering task. A semi-automatic change impact analysis for requirement changes is performed using trace links between requirements, issues and developers. Team members, which are affected by the requirement change are notified and project management tasks like issue updates are performed automatically. Figure 2 shows a possible setup for this use case. A requirement management tool, like Rationale RequisitePro[5] is connected to the integration system, as well as an issue tracker

[5] http://www-01.ibm.com/software/awdtools/reqpro

like Trac[6]. In addition a connector for email notifications and a tool for the management of developer contact and identity information has to be available.

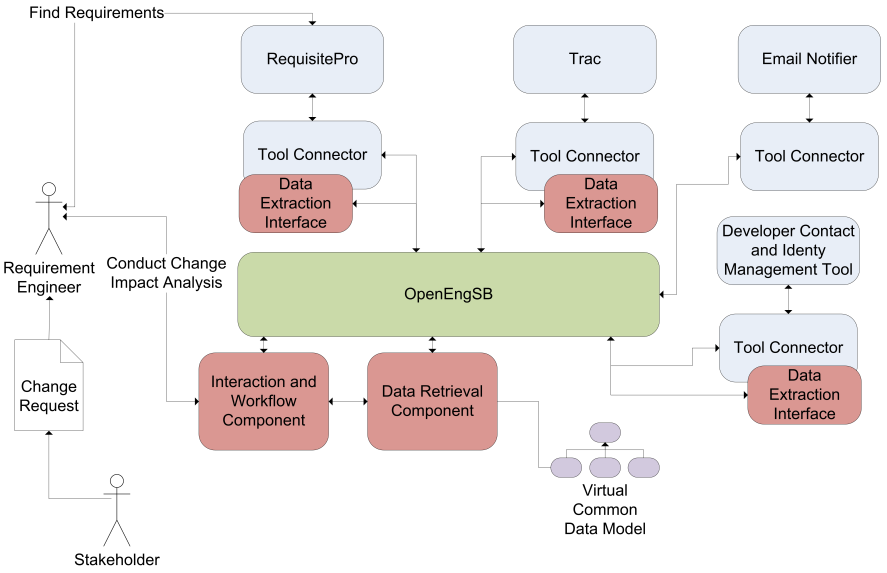

Fig. 2. Possible setup for the "Change Impact Analysis for Requirement Changes" use case.

In this use case trace links between issues and requirements are established using informal semantic information in the issue description. Each issue, which is related to a requirement has to include a reference to the respective requirement using the format "#requirement(¡requirementId¿)", with the respective requirement identifier in its description. Issues can either be related to no requirement, to a single requirement or to multiple requirements. This informal semantic information is used by the integration system to establish trace links. Other forms of trace links between requirements and issues are possible, like for example explicit mapping tables, but for the sake of simplicity only this simple form of direct references is used. Links between the issue and developer are established with the help of the assignee attribute of the issue concept. Figure 3 shows the references between these three concepts with actual and automatically derived backward links. Although they are not explicitly modeled the backward links can be navigated like normal traces and are automatically managed by the integration system.

The change impact analysis process is triggered by an requirement change request of a stakeholder of the (software+) engineering project. The requirement engineer analyses the change request and identifies affected requirements. These

[6] http://trac.edgewall.org

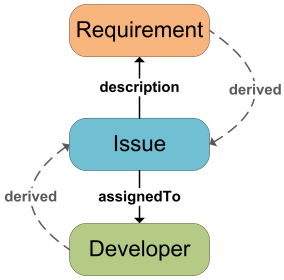

Fig. 3. Trace links between requirement, issue and developer concept.

requirements are used as input for the change impact analysis. In a first step all related issues are identified. Depending on their current status they are simply marked for review, or reopened and marked for review. All affected developers are notified about the changes and asked for their estimates. Finally a report is generated, which contains information about affected issues and developers and who will need to contact the requirement engineer for a re-evaluation of the requirement estimate.

Using the infrastructure provided by the proposed EKB based semantic integration solution the "Change Impact Analysis for Requirement Changes" use case can be realized with a simple configuration step during development of the tool connectors and by using the workflow and interaction component of the technical integration system in combination with the features provided by the EKB. The definition of the relation between two different engineering concepts based on informal semantic information is modeled in the virtual common data model. For this purpose the soft reference mechanism of the EKB is used.

Relations between different concepts, which are represented by semantic meta information, like for example trace links are modeled as soft references. The goal of soft references is to provide a possibility to establish links between two different concepts based on informal semantic information. In contrast to hard references soft references are not directly included in the data model in the form of a specific reference attribute. This means that they are part of the semantic meta information and not included in the common data schema.

To establish a soft reference the following steps are necessary: The *target concept* has to define a key attribute. This key attribute is used to identify the target instance of the reference. The data extraction infrastructure also uses the key attributes to load specific data items. The source concept has to define the target of the soft reference and the attribute, which contains the soft reference. In addition the mechanism for the extraction of the actual reference from the content of this attribute has to be defined. The extraction process is adaptable to make it possible to handle different forms of semantic meta information. In the prototype a regular expressions based solution is implemented, which makes it possible to extract the reference from an arbitrary textual source. A regular expression based soft reference definition contains the target concept and the

regular expression for the key extraction. To establish a concrete reference the content of the respective attribute of the source concept has to include the reference, which has to be extractable using the mechanism explained in the previous step.

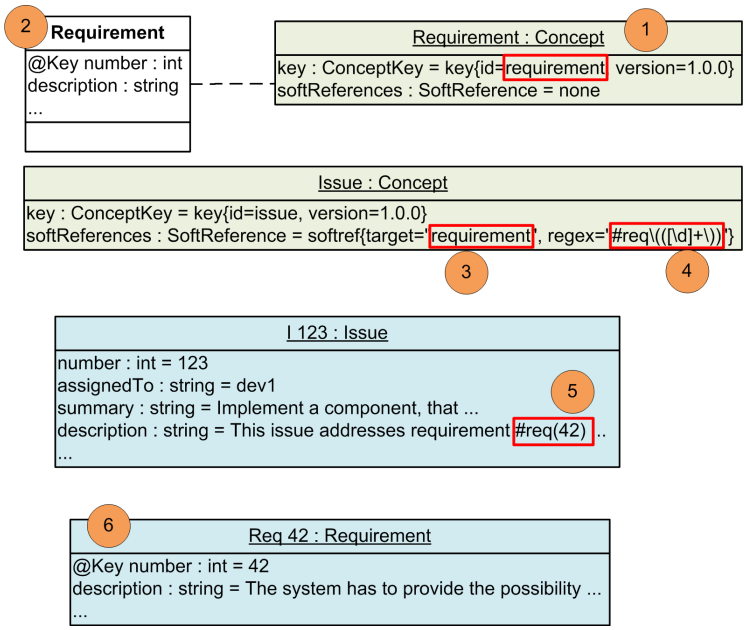

Fig. 4. Overview about the soft reference definition process.

Figure 4 gives an overview about the different steps necessary for the definition of a soft reference between an issue and a requirement. First the requirement concept has to define an identifier (1) and the requirement type, which is the attached data type of the requirement concept has to define a key attribute (2). In this example the requirement number attribute is used as key. Then the soft reference between the issue concept and the requirement concept can be defined. In the example shown in Figure 3 this is done with the help of a regular expression based soft reference. Besides the identifier for the target concept (3), also the regular expression for the extraction of the reference key has to be defined (4). Note that when only the identifier of the target concept is defined and the version is omitted then the version of the source concept is also used for the target concept. If an issue has a description which contains a reference (5) then the semantic integration infrastructure can de-reference this link and load the respective requirement (6).

In this use case the description attribute of an issue includes soft references to requirements and the attribute assignedTo is a soft reference to a developer. Listing 1 shows the definition of the soft references using an annotation based

Listing 1. Definition of soft references from issue to requirement and developer concepts.

```
@ReferenceId(targetConceptId = "developer",
targetConceptVersion = "1.0.0",
regexp = ".+")
private String assignee;

@ReferenceId(targetConceptID = "requirement",
targetConceptVersion = "1.0.0",
regexp = "\#requirement\\(([^\\)]+)\\)")
private String description;
```

concept definition mechanism. As regular expression based soft references are used, the definition includes beside target concept also the regular expression for the extraction of the actual references. The configuration effort for this use case is reduced to the definition of the attribute containing the reference and a mechanism for extracting the actual reference from the field content. The EKB based semantic integration solution supports the usage of other solutions for the extraction of the references. Even very complex scenarios, where an knowledge system is used to find references between elements is possible, but a regular expression based approach is sufficient for this use case.

Building upon this configuration the workflow for the change impact analysis is implemented using the soft reference mechanism of the EKB. Based on the relation between requirements, issues and developers defined as soft references and stored in the virtual common data model, the EKB can derive actual trace links between instances of these concepts. So the requirement engineer defining and using this process does not need to know how requirements are actually linked to issues or developers. This knowledge is contained in the definition of the soft references, which are performed parallel to the introduction of the guideline for semantic meta information they originate from. The guideline in this use case is that an issue has to reference related requirements in its description. The requirement engineer can query the system for related issues and developers and use this information to conduct a high quality change impact analysis and to inform all affected team members.

5 Discussion & Conclusion

As the setup to use informal semantic references between different engineering concepts is minimal, the proposed semantic integration solution is an effective and efficient solution for advanced applications like requirement tracing. Using the EKB, the usage of relations between concepts is decoupled from the actual form of the relation. This means that if the way requirements are linked to issues is changed, the change impact analysis process does not need to be adapted. The

requirement engineer can perform the change impact analysis without knowledge about the issue domain and without tedious manual research which issues and developers are affected. This means that the system is also usable for stakeholders, which are not part of the development team, or not experts for all tools and tool domains involved in the development process. No additional effort is necessary for trace link generation, as the semantic meta information, which is already available due to specific project guidelines is used to trace from requirements to issues and from issues to developers.

Stakeholders, who are not part of the actual development team, usually do not know about team internal guidelines, like the way requirements are referenced in issue descriptions. Yet they need to perform high level quality assurance and project management tasks, like the change impact analysis described in this use case. Using the proposed EKB based semantic integration solution, they can perform advanced applications using informal relations between different engineering concepts, without the need to know how the different concepts are actually connected. The robustness of the whole system is influenced positively by the fact that the semantic relations between concepts are defined directly in the respective domains and can be updated every time the underlying guidelines, a tool or a tool domain changes. The proposed EKB based semantic integration framework needs some additional setup, before soft references between different engineering concepts can be used. Basically two steps have to be performed to make references between concepts possible:

Definition of the soft reference The source concept has to define which attribute contains a soft reference and a mechanism for extracting the actual reference from the attribute value. The prototypic implementation of the EKB supports a regular expression based mechanism for soft reference definition. Listing 1 shows how this definition can be performed with the help of an annotation based concept specification mechanism. Other types of soft references can be implemented and integrated into the EKB easily using a plug-in style architecture.

Definition of a key attribute at the target concept The target concept of the soft reference needs to specify a key attribute. This has two specific implications. On the one hand the value of this key attribute is the actual reference value and used like a foreign key in a relational database. On the other hand the tool connector of the target concept needs to support queries for a specific element based on this key attribute.

These two steps are usually performed during integration of an engineering tool into the integration system. Domain experts and integration experts work together to identify key attributes and relations between engineering concepts. Furthermore, project guidelines, which are already in place and define how for example issues have to be described when they are created, give hints where semantic meta information is available and can be used to link between engineering concepts. Once the domain expert and the integration expert have an overview about potential references between different concepts the actual definition of the references can be done in a very short time. The whole process including review of project guidelines and other available meta information usually can be per-

formed in one to two developer workdays for both domain expert and integration expert. If the project is smaller and each member has a good overview about the semantic meta information less effort is necessary for this task.

Using the proposed semantic integration framework the definition of the change impact analysis does not include any details about the relation between requirement and issue, besides the fact that such a relation exists. This means that a requirement engineer designing or using the change impact analysis workflow does not need to know any details about the relation between requirement and issue concept and its actual representation. The requirement expert can concentrate on the task of change impact analysis, which is decoupled from the actual linking mechanism between different engineering concepts. The semantic linking information between engineering concepts is defined directly in the virtual common data model and can be used by any advanced application. This means that this knowledge is not duplicated throughout the system and can be changed or updated easily. As a result the team is flexible and can decide to change specific guidelines or standards for semantic meta information, if they feel the necessity to do so. Evolution of the system is possible without the fear of breaking high level process definitions. Therefore tasks like tool exchange or the introduction of new engineering tools are much easier to accomplish. The explicit nature of the reference definition in the virtual common data model is also a very useful documentation of semantic dependencies between different engineering concepts and can be analyzed by integration experts or domain experts to better understand the current setup of the system and to find potential inconsistencies or other problems.

To perform this use case in a technical only integration framework, it is necessary to use the semantic information in the change impact analysis process definition. This means that in the process implementation the semantic relation between issues and requirements and issues and developers is explicitly used. The drawback of such a solution is that the actual process of using the semantic information has to be duplicated in every advanced application which needs to facilitate the trace link between requirements and issues. In addition it is very hard to perform changes of the layout of the semantic meta information, as there is no single point of change, but all workflow definitions have to be reviewed and possibly updated.

Compared to a solution based on a technical-only integration system the definition of workflows in the proposed EKB based semantic integration system is much more portable and reusable across different projects. Due to the fact that only the existence of a link between two concepts regardless of the actual form of the link is needed to build advanced applications, workflows can be reused in other projects, where relations based on informal semantic information are represented differently. Therefore it is possible to use a common set of workflow definitions for typical project and quality management tasks recurring in all kinds of different projects operating with the same engineering concepts. These differences show that the proposed semantic integration solution is much better suited for advanced software engineering applications like requirement tracing

than a technical-only integration solution. The possibility to define and use semantic meta information to model relations between engineering concepts is a very flexible mechanism, which can also be used for other advanced applications, like for example automatically linking commits to related issues. The main advantage of this solution is the simplicity of the setup and the single point of change making the overall system easier to change and to maintain. The main drawback of the EKB based semantic integration solution is the additional configuration and setup effort during development and implementation of the tool connectors. But if advanced applications, like requirement tracing for have to be performed regularly, the additional effort during system setup pays off.

Further work will include a large scale evaluation in an industrial context. In addition, a new set of more complex soft-links between engineering concepts will be used to further evaluate the automated derivation of tracing information. Finally, we are working on efficient tool-support for a more user-friendly definition of the soft-links.

Acknowledgments. This work has been supported by the Christian Doppler Forschungsgesellschaft and the BMWFJ, Austria. Furthermore, the authors want to thank Michael Handler for his prototypic implementation of the use case.

References

1. Antoniol, G., Canfora, G., Casazza, G., De Lucia, A., Merlo, E.: Recovering traceability links between code and documentation. Software Engineering, IEEE Transactions on 28(10), 970–983 (2002)
2. Cleland-Huang, J., Zemont, G., Lukasik, W.: A heterogeneous solution for improving the return on investment of requirements traceability. In: Proceedings of the Requirements Engineering Conference, 12th IEEE International. pp. 230–239. IEEE Computer Society, Washington, DC, USA (2004), http://portal.acm.org/citation.cfm?id=1018443.1022110
3. Cruz, I.F., Xiao, H., Hsu, F.: An ontology-based framework for xml semantic integration. In: Proceedings of the International Database Engineering and Applications Symposium. pp. 217–226. IEEE Computer Society, Washington, DC, USA (2004), http://portal.acm.org/citation.cfm?id=1018432.1021520
4. Egyed, A.: A scenario-driven approach to trace dependency analysis. Software Engineering, IEEE Transactions on 29(2), 116–132 (2003)
5. Gotel, O., Finkelstein, C.: An analysis of the requirements traceability problem. In: Requirements Engineering, 1994., Proceedings of the First International Conference on. pp. 94 –101 (apr 1994)
6. Halevy, A.Y.: Why your data won't mix: Semantic heterogeneity. Queue 3(8), 50–58 (2005), article
7. Heindl, M.: Managing Dependencies in Complex Global Software Development Projects. Ph.D. thesis, Vienna University of Technology, Faculty of Informatics (2008)
8. Heindl, M., Biffl, S.: A case study on value-based requirements tracing. In: Proceedings of the 10th European software engineering conference held jointly with

13th ACM SIGSOFT international symposium on Foundations of software engineering. pp. 60–69. ESEC/FSE-13, ACM, New York, NY, USA (2005), http://doi.acm.org/10.1145/1081706.1081717
9. Jarke, M.: Requirements tracing. Commun. ACM 41(12), 32–36 (1998), article
10. Kaindl, H.: The missing link in requirements engineering. ACM SIGSOFT Software Engineering Notes 18(2), 30–39 (1993)
11. Moser, T.: Semantic Integration of Engineering Environments Using an Engineering Knowledge Base. Ph.D. thesis, Vienna University of Technology, Faculty of Informatics (2010), phdthesis
12. Paulk, M.C., Curtis, B., Chrissis, M.B., Weber, C.V.: Capability maturity model, version 1.1. Software, IEEE 10(4), 18–27 (2002)
13. Rosenthal, A., Seligman, L., Renner, S.: From semantic integration to semantics management: case studies and a way forward. SIGMOD Rec. 33(4), 44–50 (2004), article
14. Schwaber, C.: The changing face of application life-cycle management. Forrester Research (2006)