2010 International Conference on Complex, Intelligent and Software Intensive Systems

Semantic Integration of Heterogeneous Data Sources for Monitoring Frequent-Release Software Projects

Stefan Biffl Wikan Danar Sunindyo Thomas Moser
Christian Doppler Laboratory for Software Engineering Integration for Flexible Automation Systems
Vienna University of Technology, Austria
{stefan.biffl, wikan.sunindyo, thomas.moser}@tuwien.ac.at

***Abstract*—Open source software teams routinely develop complex software products in frequent-release settings with rather lightweight processes and project documentation. In this context project a major challenge for data collection is how to extract the relevant project management knowledge effectively and efficiently from a wide range of software project data sources, such as artifact versions, bug reports, and discussion forums. In this paper we introduce a framework and tool support for the semantic integration of data from a variety of data sources to facilitate efficient data collection, even in projects with frequent iterations. Based on data from real-world use cases in open source projects we compare the efficiency of the proposed framework with a traditional data warehouse approach. Major result is that the proposed approach can make data collection for project monitoring about 30% - 50% more efficient, in particular, in contexts where heterogeneous data sources change during the project.**

Keywords: semantic integration, project monitoring.

I. INTRODUCTION

Open source software (OSS) projects rely on experts from various backgrounds and have gained an impressive level of stability and performance, in some areas even outperforming comparable commercial tools (e.g., tool sets of the Apache Software Foundation[1]). OSS teams routinely develop complex software products in distributed settings with rather lightweight processes and project documentation. However, there are issues that slow down the proliferation of OSS for complex projects such as insufficient awareness of changes in a project (e.g., due to time zone differences) or misunderstandings (e.g., due to cultural differences or incompatible development style). Therefore project managers and task leaders need effective and efficient data collection services as foundation for the timely overview on progress, cost, and quality of the project activities, similar to a data warehouse (DWh) in long-running business processes, for exploring and analyzing large quantities of data in order to discover meaningful patterns [2].

Unfortunately, the broad range of means for communication (e.g., e-mail, personal instant messaging, communication forums, and blogs) and coordination (e.g., version control systems, requirement management tools, and issue trackers) used in distributed development project settings has made managing such projects an increasingly difficult task. A good project manager needs to get an overview on all relevant tools used in his project, as well as of the relevant data on the status of the project work contained within these tools. The ability to correlate and assess project data in distributed project tools is vital both for estimating the current project status and also for predicting future project risks and opportunities [15, 24].

A major challenge of data collection is how to extract the relevant project management knowledge effectively and efficiently from the wide range of available software project data sources, such as artifact versions, bug reports, and discussion forums. Project participants communicate through a wide range of tools that contain knowledge on the status of tasks, artifacts, and processes. Unfortunately, these data sources exhibit semantically heterogeneous data formats and terminologies, which take significant effort to reconcile with a DWh approach. Further, to keep the overview, monitoring and evaluation processes have to be repeated regularly because of the more frequent system releases which are performed in line with user expectations for greater responsiveness and shorter cycle times. [3]. Thus a manual approach seems infeasible due to the immense amount of data. While the DWh approach has been optimized [16] significantly for data from a stable type of tools, data from heterogeneous sources still poses a major challenge.

In this paper we introduce a framework for the semantic integration of data coming from a variety of data sources and tool support to enable efficient data collection, especially in projects with frequent iterations like OSS. Major challenges for this framework are the management of incomplete and/or inconsistent data. The retrieved data should be integrated into a suitable well-defined format to ease processing and analyzing, e.g., within a DWh. Based on data from real-world use cases in OSS projects we compare the effort using the proposed framework and a traditional DWh approach in test scenarios for integrating data from OSS projects (such as Apache Cocoon[2] and Apache Tomcat[3]). Major results are that the new approach seems well suited to make data collection for project monitoring more efficient, in particular, in cases where the data sources evolve during the project.

The remainder of this paper is structured as follows: Section 2 discusses related work on DWh approaches and semantic technologies for integrating heterogeneous data sources. Section 3 motivates the research issues and intro-

[1] Apache Software Foundation – http://www.apache.org

[2] http://cocoon.apache.org

[3] http://tomcat.apache.org

978-0-7695-3967-6/10 $26.00 © 2010 IEEE
DOI 10.1109/CISIS.2010.58

duces the research approach. Section 4 describes the concepts and the design of the framework for semantic integration of heterogeneous data sources using the implemented prototype as example. Section 5 presents the results of the initial evaluation. Section 6 discusses the research results and Section 7 concludes the paper and suggests further work.

II. Related Work

This section summarizes related work about data warehousing and about semantic integration of heterogeneous data sources from software engineering environments.

A. Data Warehousing

Data warehousing (DWh) came from the need of companies to deeply analyze and better understand their business processes for decision support. The concept of DWh was found by Inmon: "*A Data Warehouse is a subject oriented, integrated, non-volatile and time-variant collection of data in support of management's decisions*" [10].

A typical DWh consists of three elements, which focus on the data staging area [12]. The operational source systems take the data from heterogeneous sources, and puts them into the data staging area, which contains three processes, namely extraction, transformation, and loading (ETL). In the extraction process, the data gets read from the source and analyzed for further manipulation. Transformation means cleansing data, e.g., correcting misspellings, dealing with domain conflicts and missing elements, and parsing the data into standard formats. It can also combine data from multiple sources, deduplicate data, or assign DWh keys. The load phase actually loads the data into the DWh. In the data presentation area, the data gets organized, stored, and made available for direct querying by users, or other analytical applications. Several access tools like ad-hoc query tools, sophisticated data mining, or modeling applications are needed to access data from the data presentation area.

In multi-dimensional queries each dimension refers to a specific criterion of interest, e.g., a period, product, or region. In this example a typical query would be: "How big is the monthly production volume of a specific product in a specific manufacturing plant?" One possibility to perform multi-dimensional queries is OLAP (On Line Analytical Processing) [13]. OLAP tools enable casual users to carry out fast, interactive, and flexible ad-hoc queries in order to create analytical evaluations without the effort of learning a specific query or programming language.

Data mining is the process of exploration and analysis of large quantities of data in order to discover meaningful patterns and rules [2]. Data mining processes are usually complicated, time and resource consuming. With increasing resources an evolution of DWh from a reporting tool to an important real-time business asset took place. Hackathorn [8] highlights the relationship of data freshness to business value and states the vitality of minimizing the time to make new information available for decision support.

With this new role of data warehousing the demand for zero-latency DWh arose. As a consequence zero-latency DWh has been a subject of intensive research in recent years [6, 16-18]. As described in [18] the "*Zero-Latency Data Warehouse is a data warehouse, which enables a complete business intelligence process to observe, understand, predict, react to, reorganize, monitor, automate and control feedback loops in the minimal latency*". In software engineering context, Järvinen [11] used tool environments like SAS Data Warehouse and MetriFlame to collect, process, and then analyze the data from software project by using GQM (Goal Question Metrics) approach. However, this thesis is more focused on measurement data from assessment of software engineering process which are more homogeneous rather than dealing with heterogeneous data of software project.

B. Semantic Integration of Heterogeneous Data Sources

Semantic integration is defined as solving problems that originate from the intent to share data across disparate and semantically heterogeneous data [9]. These problems include the matching of data definitions in ontologies or schemas, the detection of duplicate entries, the reconciliation of inconsistencies, and the modeling of complex relations in different sources [22]. Over the last years, semantic integration became increasingly crucial to a variety of information processing applications and has received much attention in the web, database, data mining and AI communities [14, 23]. One of the most important and most actively studied problems in semantic integration is establishing semantic mappings between vocabularies of different data sources [5].

Noy identified three major dimensions of the application of ontologies for supporting semantic integration: the task of finding mappings (semi-)automatically, the declarative formal representation of these mappings, and reasoning using these mappings [21]. There exist two major architectures for mapping discovery between ontologies. On the one hand, the vision is a general upper ontology which is agreed upon by developers of different applications. Two of the ontologies that are built specifically with the purpose of being formal top-level ontologies are the Suggested Upper Merged Ontology (SUMO) [19] and DOLCE [7]. On the other hand, there are approaches comprising heuristics-based or machine-learning techniques that use various characteristics of ontologies (e.g., structure, concepts, instances) to find mappings. These approaches are similar to approaches for mapping XML schemas or other structured data [1, 4].

Using ontologies to structure information repositories also entails the use of semantic indexing techniques, or adding semantic annotations to the documents themselves. If different repositories are indexed to different ontologies, then a semantically integrated information access system could deploy mappings between different ontologies and retrieve answers from multiple repositories [25]. While semantic technologies, in particular ontologies, in principle can provide significant advantages for data collection from software engineering projects, there are very few reports on empirical evaluation of the performance of semantic technologies in real-world use cases.

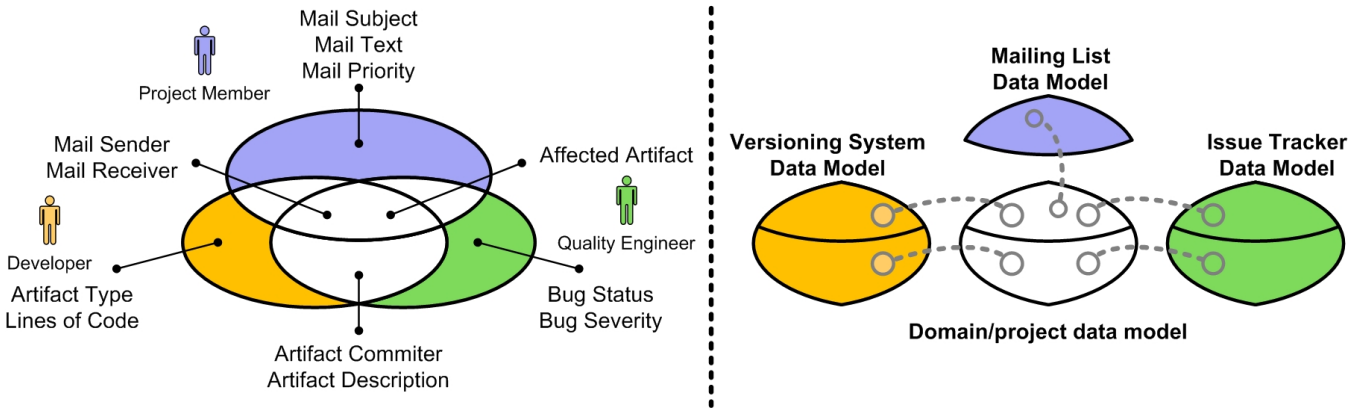

Figure 1: Overview of data models and exemplary data to be integrated.

III. RESEARCH ISSUES

As distributed development has become increasingly popular in commercial and OSS software development projects, project management scope needs to expand from a local to a global view. With this metamorphosis, conventional techniques as well as tools for data collection and analysis need to adapt or evolve. Important management decisions such as decisions regarding product quality (e.g., measured by defect density in artifacts or the average time needed to fix major defects) or decisions regarding the development team (e.g., identification and preservation of core developers) are typically based on data originating from a range of tools.

Currently, data collection is based on queries from a wide range of sources such as mailing lists, version control systems, and issue trackers. This approach has become very time-consuming. In addition, this data has to be available with little delay to support quickly reacting to various internal such as changes in the development team as well as external condition such as new releases of new software libraries used in a project. The faster relevant data can be retrieved, the more agile project steering can become. As project circumstances can change quickly or releases if new software versions are performed periodically, data collection has to be as well repeated with the same frequency, which is infeasible without proper tool support. Furthermore, once the data is retrieved, the process of analyzing and evaluating this data is even more difficult, if the data is collected from inhomogeneous sources with a variety of different, often incompatible data formats. Another issue is the data quality. Invalid, malformed, and irrelevant data elements further complicate the evaluation process, often making the outcome unsuitable for decision support.

In this paper, we propose a semantic framework that enables automated collection and integration of data originating from a set of heterogeneous tools used in software development.Figure 1 illustrates three heterogeneous data models of tools used during a typical software development process (versioning system, mailing list, issue tracker). The data models of these tools contain both elements which are only used in the context of a specific tool, as well as elements which are also used in the context of other tools. In order to integrate the data models respectively the tools, these so called "common concepts" need to be identified. As a next step, the concepts of the local tool data models that are similar or equal to a specific common concept, are mapped to this common concept, as shown on the right hand side of Figure 1. To be of use for decision support, data integration has to be carried out efficiently to provide quasi-instant availability of the data, despite the fact that these tasks are very complex. The integrated and validated data can then be used as basis for basic of project data analysis and data improvement such as aggregation of data. Based on the data, also more advanced project management methods such as quality prediction (5) [27-29], in-time notification of relevant stakeholders [26], or decision support for project managers can be applied.

From this approach we derive the following research issues.

RI-1 Comparison of a traditional data collection process to a semantically-enabled data collection process. Currently, the collection and integration of data originating from a set of heterogeneous tools is a mainly manual task. As summarized in the related work, there exists tool support for the loading processes of a DWh, however, these tools are often only useable for specific applications and therefore hard to use for more generic processes without major adaptations. The proposed framework supports the collection and integration process by providing automated process steps, such as time-triggered collection or automated checks of data consistency and integrity. While we expect the proposed framework to make the data collection and validation process steps significantly more efficient, we also see reasonable effort investment in setting up the framework in a given context. Thus empirical evaluation is necessary to assess by when a breakeven point is likely to be achieved.

RI-2: Integration of additional data sources. Current tools used in a distributed software engineering environment are not fixed, but frequently change over time or according to new project requirements. In order to support such

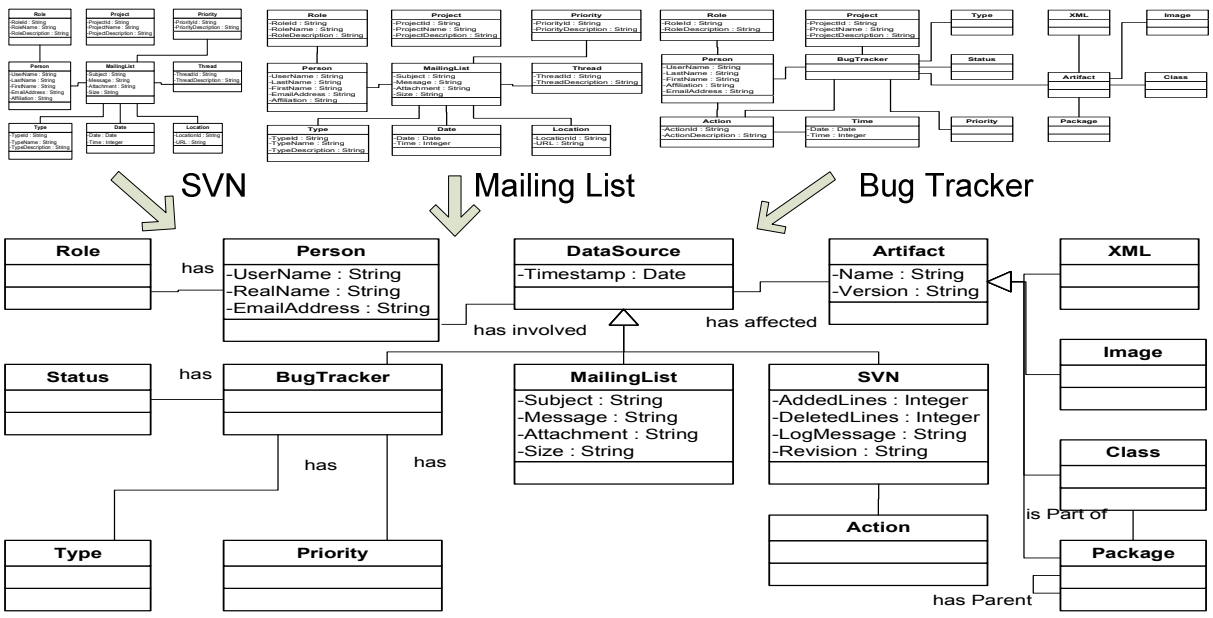

Figure 2: Overview of the data model of the data from various data sources

changes of data sources, the proposed approach needs to provide extensibility regarding both the underlying process as well as the designed data model. To assess the cost and benefit of this extensibility, the effort needed for the inclusion of another data source needs to be measured as well as the relationship of the extension effort to the number and types of already integrated data sources.

IV. Semantic Integration of Data Sources

This section describes the concepts and the design of the framework prototype for semantic integration of heterogeneous data sources from software engineering (SE) environments.

The aim of the prototype was to design and to implement a semantically-enabled DWh to store the data and artifacts retrieved from the heterogeneous SE repositories. This includes the interfaces and retrieval mechanisms for a set of different types of repositories, the semantic model supporting the DWh as well as a set of predefined data consistency and completeness checks. This semantically-enabled DWh then can be used as a basis for advanced data analysis and support for decision making.

Before designing a tool prototype one has to be aware of its various requirements. Besides the task-specific intrinsic requirements, there are some common requirements, which apply to most software tools. With one of its application purposes in assisting project leaders, by handling time-consuming tasks, it is a logical consequence that the tool has to be easy and efficient to use. Concerning today's tight schedules in software development, furthermore the tool has to be quick and easy to set up and maintain. As the tool is intended for heterogeneous distributed networks, one natural requirement is platform independence. The tool should seamlessly integrate into existing parts of the infrastructure.

Since there are already components available (e.g., frameworks for ontology access or data retrieval), which can handle small parts of the proposed workflow, the integration of these well known and tested components would be advisable.

A. Data Model of the Prototype

Figure 2 shows how the different data models from various data sources are integrated into a common data model using ontology. Using UML standard notation, data models from SVN, mailing list and bug tracker are captured, analyzed and then merged into the common data model for the prototype. This data model can integrate related classes from different data sources, such that we can reduce the duplication between data models. We can also discover the relationship of different data sources, for example the person who commit in the SVN is actually the same person who fixed the bug in the bug tracker. We use *Protégé*[4] ontology editor in designing ontology for these data models.

B. Implementation of the Prototype

According to the requirements discussed above, the tool was developed using the Java programming language, with the advantages of platform independence, a big community and therefore ample OSS libraries. For example, the access to the ontology was realized using the ontology processing features of the *Jena*[5] framework. The *Jena* framework provides, amongst other things, an OWL API for programmatic access to OWL ontologies using Java. As a starting point the *Jena* framework provides a tool called "*schemagen*" which creates a java class file, containing an instance of the ontology model as well as

[4] http://protege.stanford.edu
[5] http://jena.sourceforge.net

the elements of the input ontology as static fields. This allows easy access of programs to the ontology and its vocabulary.

A basic element of the Jena Framework is the *OntModel* class. An existing ontology model can be loaded into an *OntModel*, which provides suitable features to modify the model and persist the model to a file. In our case, the ontology is only modified by adding individuals as well as setting the individual's properties. This can be done using the *createIndividual* method of the *OntModel* class, as well as the *addProperty* method of the *Individual* class, which is another basic element within the *Jena* Framework. To retrieve existing individuals the method *listInstances* of the *OntClass* class can be used, which returns an iterator over the correlative individuals.

Once the ontology model has been made accessible, the next step was to provide a convenient way to configure the tool. This was done using "*Commons Configuration*"[6], which enables Java applications to read configuration data from a variety of sources. At first general options are declared, as the locations where to find the input ontology or where to write the populated ontology in the file system. For the SVN data source the URL of the repository is configured as well as the revision where the import procedure should start at. The mailing list and bug tracker data sources are configured similarly. In code, once the configuration source is loaded, the several elements can easily be accessed considering their XML structure.

To access the subversion repository, the "*SVNKit*"[7] code library was used. *SVNKit* is an OSS toolkit for Java and provides an API to access and manipulate subversion repositories online as well as local working copies. To retrieve the data from the mailing list archives, the "*mstor*"[8] library was used. *Mstor* is a local store provider enabling access to email messages in *mbox* format, stored in the local file system.

Here some simple heuristics are applied: it is assumed, that the username of a person is the part of the email address before the '@' character. Further, it is assumed that every person uses the same username in all of the tools associated with the project, hence our data sources. By applying these assumptions, it is possible to identify individuals already included in the ontology as the identical *Person*. Therefore, these individuals do not need to be added again, but supplemental properties can be added, as for example the *hasRealName* and *hasEmailAddress* properties not known when fetching from the SVN repository. Another heuristic is applied, indicating that a person is a core member of the project if the persons email address includes the project name after the '@' character. For example persons with email addresses containing "@apache.org" are considered to be core members. Therefore the property *hasRole* is added with the value *CoreMember* as well as with the value *MailingListParticipant*, as obviously an email was sent to the mailing list.

Next, the message body is being processed. An individual of the type *MailingList* is created and added to the ontology, along with the corresponding properties *hasSubject*, *hasMessage*, *hasTimestamp*, *hasSize*, *hasInvolvedPerson*, *hasAttachment* and *hasAffectedArtifact*. The subject, timestamp and size can be read directly from the mail message. The involved person is already known, as processed in the previous step. The email message, attachments and affected artifacts need more attention. The email message may contain plain text or other elements. Therefore, the content type of the email body has to be examined. If the content type is "text", the *hasMessage* property is added with the corresponding message text. If the content type is "multipart", every part of the message is queried and processed. These parts can either be text or contain attachments. In the first case this text is added as *hasMessage* property. Otherwise the attachment names are added as *hasAttachment* property.

V. EVALUATION

This section develops the design of the evaluation and reports results of the evaluation to address the research issues of this work. During the development process the prototype was carefully evaluated using several Apache projects as target for the retrieval process. The step using the Apache projects as setup for the evaluation was chosen, because former work has been done regarding these projects. Therefore it was possible to compare the results of the automated data retrieval and merging, with data previously obtained. Two projects were used during the evaluation procedure: *Apache Tomcat* and *Apache Cocoon*.

Regarding the research issues, on the one hand side, the focus of the evaluation was the comparison of the effort needed for successfully collecting and integrating data originating from semantically heterogeneous data sources, while on the other hand side, the effort needed for the inclusion of an additional data source in the overall collection and integration process was also determined for both traditional integration into a DWh and semantic integration using an ontology. The main evaluation criteria were the delay of newly available data before becoming available for analysis, the effort needed for collecting and integrating the data, and finally the quality of the integrated data regarding consistency and integrity.

The following sources were included in the data retrieval process: SVN repositories, developer mailing list archives, and bug tracker data. Querying these data sources of the two test projects could be successfully performed. After retrieval the resulting ontologies were carefully examined to rule out any possible errors related to the retrieval as well as the integration procedure. During the development process, special emphasis was laid on

[6] http://commons.apache.org/configuration
[7] http://svnkit.com
[8] http://mstor.sourceforge.net

providing easy usability. The tool has to be configured by inserting, into a configuration file in XML format, the online locations of the several data sources as well as the desired range of data (e.g., range of SVN revisions or time span of mail conversations) to be included in the project evaluation. Once the configuration is completed, the tool can easily be executed on the command prompt, without the need of specifying any further parameters. Of course automatic periodic execution is easily possible, e.g., by setting up a scheduled task. Furthermore, incomplete or inconsistent data sets are excluded automatically.

The integration processes of heterogeneous data from different sources using traditional and ontology-based approaches are pictured in Figure 3. The integration processes consist of three steps: data collection, knowledge representation and data quality assurance. In the data collection step for traditional approach, the software developers will get the data of the project by downloading the SVN repository, e-mails from mailing list and bug report by using tools, and then put the information into the databases. While in the ontology-based approach, the software project development data will be collected by using data fetcher tool that has been explained previously.

The knowledge representation step of the traditional approach consists of several tasks: First, *normalizing the data* to make them more subject-oriented. For example, one revision data from SVN may consist of several actions concerning different modules. We create separate entities for these modules, so we can access and analyze them separately for further purposes. Second, *identifying and creating relationships* between entities originating from different data sources which have similarities. For example the author entities from SVN could correlate to the sender of the mailing-list. Third, *cleansing the data*, e.g., by completing the missing data, adding keys to entities or adding more information to entities. Fourth, *integrating the data format*, e.g., the format of the date and time should be the same to make comparisons between different data easier. Fifth, *integrating the tool data*, e.g., by using database format rather than CSV format.

In the ontology-based approach, the knowledge representation is done by designing and implementing an initial ontology that captures all data models and requirements in advance. The designer defines the classes, attributes and relations of ontology that will be populated automatically by using tool during the data collection step. In the ontology, the designer should also define restrictions, rules and axioms that are used for data quality assurance to check the syntax and the constraint of the data. For checking the logic and semantic between the data, the domain expert uses reasoning.

The domain expert in the traditional approach uses manual checks to assure the data quality, i.e., he performs several checks, like checking the relationships between entities, checking whether there is no missing data, checking whether the formats of all data are correct and follow the standard, checking the validity of the data, and checking the data constraints, etc.

Compared to previous retrieval attempts on collecting and integrating data from the mentioned project data sources, which has to be carried out mainly manually, the time saving was significant. Also the previously conducted manual integration of the data was now performed automatically, resulting in a homogeneous, error-free data set. The automated data collection approach provides a data fetcher tool in advance. To build that tool, it needed analysis, design, implementation, and testing steps before the tool can be released and used by the user to collect the data. Inappropriate tool development could lead to the wrong data collection.

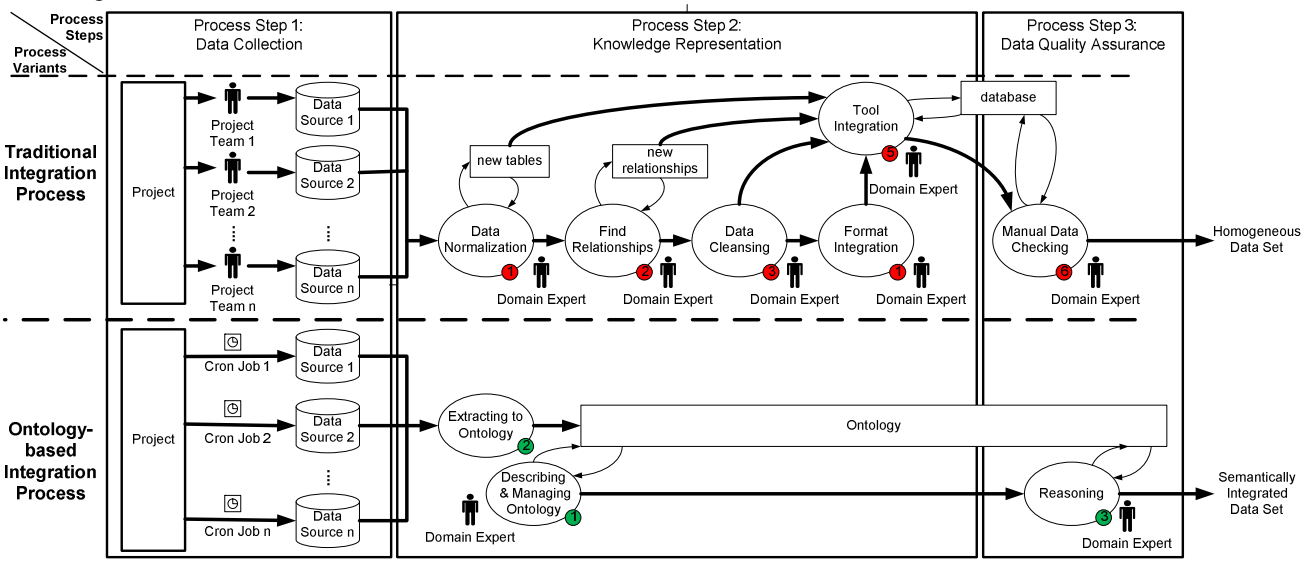

Figure 3: Comparison of traditional and semantically-enabled integration processes.

From our measurements we estimate, that the application of the proposed tool saves about 30% - 50% effort regarding data collection and validation in the study context. As a basis for this estimation, we compare the number of steps needed to do the traditional approach versus using ontology-based approach to do the data integration (3 steps compared to 6 steps). Of course, the actual savings depend strongly on the experience of the person carrying out the manual data retrieval. Further, it has to be considered that the actual time savings depend on many factors that need to be explored for achieving more reliable estimates. In contrast to manual data retrieval, the proposed framework has to be set up only once at the beginning of the project-evaluation by configuring the various data sources as well as the execution interval. From this moment on the tool does its job fully automated, so that none of the project members has to devote precious time to data collection that could be better invested into the actual project work.

Considering that during automatic retrieval using the proposed tool no human interaction is needed, and therefore freeing additional manpower, the overall time saving is much higher. In summary the proposed tool automates the full data collection and integration process, providing project managers with in-time information on their project, allowing to immediately conducting an evaluation of all desired parameters.

VI. DISCUSSION

This section discusses the described framework and its prototypic implementation, as well as the initial results of the evaluation with regard to the research issues identified in section 3.

RI-1 Comparison of traditional and semantically-enabled data collection processes. To succeed in building a tool that is capable of mastering all described requirements, one has to carefully choose the right technology to handle the described required tasks. The approach using an ontology is obvious, since, in contrast to a database, an ontology is capable of a proper knowledge representation based on well-defined semantics. While a database only supports integrity checks on a structural level, conducting integrity checks on a semantic level is an intrinsic part of an ontology. Furthermore an ontology provides extensive reasoning capabilities, which means the possibility to use a priori hidden knowledge by deducting new facts out of known ones. By providing an explicit specification of the stored data's intended meaning, instead of the sole data itself, an ontology allows sophisticated querying. This is important to address the issue of being able to provide project managers with a proper tool for decision support.

Currently the proposed tool has been successfully tested with *Bugzilla*[9], *SVN*[10] and mailing list archives in *mbox* format. Supporting this data sources was chosen, since they are widely used. Furthermore, they are free and standardized, so that they can be used in any project without having to pay license fees or signing special contracts, making them a good choice for open source projects as well as for commercial ones. Moreover, due to the wide use, there are many free-to-use code libraries for data access available, avoiding the effort of implementing all the desired features from the scratch.

Despite the ease of use of the proposed tool there are some basic requirements a project leader has to consider before project setup, when planning to apply the tool. Of course, to be able to automatically retrieve data from the various project data sources (like the version control system, the bug tracker, and the mailing lists) these sources have to be implemented in a way that allows accessing their content directly. By carefully choosing the systems for versioning, bug tracking, and mailing lists, additional time-consuming modifications and/or feature-implementations to the proposed tool can be avoided. For example, if the used bug tracker only allows downloading of bug reports in a specific data format, not known to the tool's underlying parsing mechanisms, transferring the data into the ontology will fail. Of course implementing support for new mechanisms to the tool is possible, but time-consuming.

RI-2: Integration of additional data sources. During the design process of the tool special care was taken to retain the possibility of integrating support for additional data sources. Despite the fact that implementing support for additional data sources (as mentioned earlier) is time-consuming, providing this possibility is important, since it allows the integration of the proposed tool into already existing environments. During the integration process, when merging data retrieved from the various data sources into the ontology to successfully carry out the combination, the routines have to be able to recognize relations between the various entries. For example, a person sending emails to the mailing lists has to be recognized as the same individual when committing an artifact to the version control system. This task can be very difficult to achieve. Therefore, simple heuristics have been applied. If a project's structure is not accordant to this predefined heuristics, the matching procedure cannot be satisfactorily performed. In our example this would be the inconsistent use of usernames among the different systems. To avoid this problem in a project either consistent usernames should be used, or a suitable identifier has to be provided. However, this would cause the need for adapting the heuristics or implementations of new matching mechanisms.

In the current prototypic implementation, retrieving the data of a project results in a single corresponding ontology. A topic of discussion is the implementation of the possibility to integrate data from two or more different projects into the same ontology. This could enable for the analysis of possible synergy effects between different projects as well as combined statistics. Of course, the corresponding project leaders would have to evaluate, whether this step makes sense for their particular projects.

[9] http://www.bugzilla.org
[10] http://subversion.tigris.org

VII. Conclusion and Further Work

OSS teams routinely develop complex software products in distributed settings and with rather lightweight processes and project documentation. In this context project managers and task leaders need data collection services as foundation for the timely overview on progress, cost, and quality of the project activities, similar to a data warehouse for analyzing business processes. However, a major challenge of data collection is to extract the relevant project management knowledge effectively and efficiently from semantically heterogeneous software project data sources, which can take significant effort to reconcile.

In this paper we introduced a novel framework for the semantic integration of data from a variety of data sources and tool support to allow the efficient data collection, even in projects with frequent iterations. The retrieval process was accomplished using existing tools for accessing the data sources. Further, the retrieved data was merged using simple heuristics as well as integrated into an ontology following the ontology design guidelines in [20] for efficient integration and further processing of the data.

Based on real-world use cases in two OSS projects we compared the proposed framework with a traditional DWh approach. Major result is that the new approach seems well suited to make data collection for project monitoring 30% - 50% more efficient, in particular, if the data sources evolve during the project. For SE environments, in which tool sets used and their associated data sources change, the proposed approach seems particularly well suited to support project monitoring and analysis.

Further work. Once configured properly, the framework handled retrieval and merging of project data automatically. A next step the methods for easier usage of the provided data have to be provided. We plan to make the integrated and validated data available to improve the value of existing types of project monitoring cockpits by supporting queries such as cost, effort, and defect prediction.

Acknowledgments

The authors would like to thank Dindin Wahyudin and Raphael Zaki for helping with the design and implementation of the framework. This work has been supported by the Christian Doppler Forschungsgesellschaft and the BMWFJ, Austria. In addition, this work has been partially funded by the Vienna University of Technology, in the *Complex Systems Design & Engineering Lab*.

References

[1] S. Bergamaschi, S. Castano and M. Vincini, "Semantic integration of semistructured and structured data sources," SIGMOD Rec., vol. 28, 1999, pp. 54-59.

[2] M. Berry and G. Linoff, Data Mining Techniques For Marketing, Sales, and Customer Support. John Wiley & Sons, 1997.

[3] A.W. Brown and G. Booch, "Reusing Open-Source Software and Practices: The Impact of Open-Source on Commercial Vendors," Proc. 7th Intl Conf on Software Reuse: Methods, Techniques, and Tools, Springer, 2002.

[4] I.R. Cruz, X. Huiyong and H. Feihong, "An ontology-based framework for XML semantic integration," Proc. Intl. Database Engineering and Applications Symp., IEEE, 2004, pp. 217-226.

[5] A. Doan, N.F. Noy and A.Y. Halevy, "Introduction to the special issue on semantic integration," SIGMOD Rec., vol. 33, 2004, pp. 11-13.

[6] I. Foster and R.L. Grossman, "Data integration in a bandwidth-rich world," Communications of the ACM, vol. 46, 2003, pp. 50-57.

[7] A. Gangemi, N. Guarino, C. Masolo & A. Oltramari, "Sweetening WordNet with DOLCE," AI Magazine, vol. 24, 2003, pp. 13-24.

[8] R. Hackathorn, "Current practices in active data warehousing," Bolder Technology, 2002.

[9] A. Halevy, "Why your data won't mix," Queue, vol. 3, 2005, pp. 50-58.

[10] W.H. Inmon, Building the data warehouse. Wiley, 2005.

[11] J. Järvinen, "Measurement based continuous assessment of software engineering processes," University of Oulu, Finland, 2000, p. 99.

[12] R. Kimball and M. Ross, The Data Warehouse Toolkit - The Complete Guide to Dimensional Modeling, vol.Second. New York: John Wiley & Sons, Inc, 2002.

[13] M. Lusti, Data warehousing und data mining: eine Einführung in entscheidungsunterstützende Systeme. Springer, 2002.

[14] A. Maedche and S. Staab, "Ontology learning for the semantic web," IEEE Intelligent systems, vol. 16, 2001, pp. 72-79.

[15] A. Mockus, R.T. Fielding and J.D. Herbsleb, "Two case studies of open source software development: Apache and Mozilla," ACM Trans. Softw. Eng. Methodol., vol. 11, 2002, pp. 309-346.

[16] T.M. Nguyen, A.M. Tjoa, G. Kickinger and P. Brezany, "Towards service collaboration model in grid-based zero latency data stream warehouse (GZLDSWH)," Proc. IEEE Intl Conf on Services Computing, 2004, pp. 357-365.

[17] T. Nguyen, J. Schiefer and A.M. Tjoa, "Sense & response service architecture: an approach towards a real-time business intelligence solution and its use for a fraud detection application," Proc. 8th Intl Wsh on Data warehousing and OLAP, ACM, 2005, pp. 77-86.

[18] T.M. Nguyen and A.M. Tjoa, "Zero-latency data warehousing (ZLDWH): the state-of-the-art and experimental implementation approaches," Proc. Intl Conf on Research, Innovation and Vision for the Future, 2006, pp. 167-176.

[19] I. Niles and A. Pease, "Towards a standard upper ontology," Proc. 2nd Intl Conf on Formal Ontology in Information Systems, ACM, 2001, pp. 2-9.

[20] N.F. Noy and D.L. McGuinness, 2001, Ontology Development 101: A guide to creating your first ontology.

[21] N.F. Noy, "Semantic integration: a survey of ontology-based approaches," SIGMOD Rec., vol. 33, 2004, pp. 65-70.

[22] N.F. Noy, A.H. Doan and A.Y. Halevy, "Semantic Integration," AI Magazine, vol. 26, 2005, pp. 7-10.

[23] D.E. O'Leary, "Using AI in Knowledge Management: Knowledge Bases and Ontologies," IEEE Intelligent systems, vol. 13, 1998, pp. 34-39.

[24] J. Thai, B. Pekilis, A. Lau and R. Seviora, "Aspect-oriented implementation of software health indicators," Proc. 8th Asia-Pacific Software Engineering Conf, 2001, pp. 97-104.

[25] M. Uschold and M. Gruninger, "Ontologies and semantics for seamless connectivity," SIGMOD Rec., vol. 33, 2004, pp. 58-64.

[26] D. Wahyudin, M. Heindl, B. Eckhard, A. Schatten and S. Biffl, "In-time role-specific notification as formal means to balance agile practices in global software development settings," Proc. CEE-SET 2007, pp. 197 - 211.

[27] D. Wahyudin, K. Mustofa, A. Schatten, A. Tjoa and S. Biffl, "Monitoring "Health" Status of Open Source Web Engineering Projects," International Journal of Web Information Systems, vol. 1/2, 2007b, pp. 116 - 139.

[28] D. Wahyudin, A. Schatten, D. Winkler, A. Tjoa and S. Biffl, "Defect Prediction using Combined Product and Project Metrics: A Case Study from the Open Source "Apache" MyFaces Project Family," Proc. 34th Euromicro Conf on Software Engineering & Advanced Applications, IEEE, 2008, pp. 207 - 215.

[29] D. Wahyudin, R. Ramler and S. Biffl, "A Framework for Defect Prediction in Specific Software Project Contexts," Proc. CEE-SET 2008, Springer, 2008.