Towards Versioning Sustainability Reports

Dilshodbek Kuryazov
Carl von Ossietzky University, Oldenburg

Andreas Solsbach
Carl von Ossietzky University, Oldenburg

Andreas Winter
Carl von Ossietzky University, Oldenburg

OLNSE Number 5/2013
May 2013

Oldenburg Lecture Notes in Software Engineering

Oldenburg Lecture Notes
on Software Engineering (OLNSE)
Carl von Ossietzky University Oldenburg
Department for Computer Science
Software Engineering
26111 Oldenburg, Germany

Content

Towards Versioning Sustainability Reports*

Dilshodbek Kuryazov, Andreas Solsbach, Andreas Winter

Abstract Sustainability reporting is a key issue of sustainability development in companies. Sustainability reports are documents which describe data about different performances of companies. They intend to report and analyse sustainability information to provide sustainable future. During sustainability development, reports have to be versioned to analyse histories of sustainability reports and to present changes and improvements of sustainability data. This paper introduces an approach to versioning sustainability reports by means of delta storage. The approach explains an operation-based, schema generic means to store differences between versions. Differences are represented by a sequence of executable operations which allow for obtaining older versions by applying these to the current version. The approach is applied to versioning sustainability reports within the STORM project.

* A German version of this paper was published at 5. BUIS-Tage: IT-gestütztes Ressourcen- und Energiemanagement, to appear in 2013. Springer Verlag. http://enviroinfo.eu/5-buis-tage

Dilshodbek Kuryazov
University of Oldenburg, e-mail: `kuryazov@se.uni-oldenburg.de`

Andreas Solsbach
University of Oldenburg, e-mail: `andreas.solsbach@informatik.uni-oldenburg.de`

Andreas Winter
University of Oldenburg, e-mail: `winter@se.uni-oldenburg.de`

1 Motivation[3]

Sustainability reporting allows companies to document their performance on specific sustainability issues by measuring, tracking, and monitoring companies information such as economy, environment or social impacts. Environmental sustainability information includes standardised reporting guidelines concerning the environment which includes criteria on energy, biodiversity, and emissions [4]. Meanwhile, sustainability information is shared with investors and stakeholders during sustainability development.

In order to report sustainability information, most companies take advantage of information management software systems. Sustainability reports are documents which provide stakeholders with information about different aspects of companies. Sustainability reports enable stakeholders of a company to analyse past and present sustainability status, and support to deciding on future activities. Recently, most companies are reporting their sustainability reports on the web [16]. Also, the usage of the web features provides dialogue-based sustainability reporting among stakeholders. As far as sustainability reports are changed during their life-cycle, these changes need to be stored and versioned in order to maintain, analyse and exploit histories of sustainability data.

While reporting sustainability reports, companies feel a need for software systems which let them manage and report sustainability information to support a web-based dialogue between stakeholders and reporting companies. Sustainable Online Reporting Model (STORM) [2] which is developed at the University of Oldenburg aims at dialogue-based public or private sustainability reporting which intend to engage stakeholders in sustainability development. While maintaining sustainability reports, STORM stores all required information in its relational database and obtains various reports by appropriate requests. The storing procedure used in STORM leads to ”flood of information” within the system. Another issue which appears in case of database approach is to versioning reports in a way which is easy to analyse version histories. According to data structure of sustainability reports, versioning reports and associated data in STORM is not as simple as versioning textual documents. By looking at sustainability reports and associated data, data structure of them can be viewed as sustainability model. The problem of versioning sustainability reports is a similar issue to versioning sustainability models.

For big companies, there exists a huge amount of social, economical and ecological information which has to be handled during sustainability development over years. If a database contains an enormous amount of records, an versioning approach reports promises to anticipate a number of challenges such as storing report versions and analysing and exploiting version histories. Versioning sustainability reports and their associated data is a complex challenge like versioning software models. In

[3] Andreas Solsbach is associated with ”IT-for-Green: Environment, Energy and Resource Management with BUIS 2.0” which is funded by the European Regional Development Fund (grant number W/O III 80119242). Dilshodbek Kuryazov is funded by Erasmus Mundus TARGET II. The entire work is part of the ExploIT Dynamics research perspective at Carl von Ossietzky University.

a model-driven view, sustainability data is represented in a sustainability model. Versioning these models data might be based on model versioning approaches [1], which, in contrast to text-based versioning, allows for directly addressing modeling concepts instead of sequences of characters.

As long as software systems gather sustainability reports causing a huge amount of data, an adequate versioning approach for sustainability reports yield various advantages in reporting and analyzing systems. Appropriate techniques are required to storing reports and their associated data which on the one hand only consume view memory and on the other hand provide efficient analyses of report version history. In case of STORM, reports are stored in a database based on schema which is, at some point, inconvenient to provoke versions of them and analyse version histories. Taking these issues into consideration, this paper intends to apply the meta-model generic versioning approach introduced in [1] to STORM sustainability models. The meta-model generic and operation-based approach is applicable to any domain by referring the appropriate meta-model within that domain. In case of sustainability reporting, a data schema of sustainability reports is considered as a domain meta-model, here. The main objectives of the approach is to provide new concept to versioning sustainability reports dealing with a data schema.

The approach aims at versioning reports by only storing differences between report versions. From the meta model for the given domain a set of standardized operations, which describe single model changes, is derived. Model difference are represented as sequences of applications of these operations. The STORM sustainability report data schema is taken as a concrete application.

The paper is structured as follows: Section 2 gives a brief overview of the STORM project and the schema of database. Section 3 motivates and presents the proposed approach following an example. This section also includes references to related works on version control systems. Finally, a conclusion and an agenda for future works is given in Section 4.

2 Versioning in STORM

The interactive sustainability report management system STORM (Sustainable Online Reporting Model) [2] aims at preparing a variety of reports, publishing them on the web, involves stakeholders and persuade investors. Hence, it serves as a sustainability report template for various companies and IT-environments. STORM intends to continuously seek and maintain a dialogue with all relevant stakeholders, including customers, consumers, suppliers, employees, and shareholders.

The data schema of STORM is based on the Global Reporting Initiative (GRI) reporting framework (G3) [4]. GRI is a worldwide standard including sustainability reporting.

As a continuous example, the following tables represent a small example for a sustainability report. This example follows a simplified structure used in STORM.

Figure 1 depicts the data of a sustainability report with an extra version including a number of articles (Fig. 2).

No.	name	status	date	version	articles[a]
1	Management approach to environmental responsibility	in process	11.11.2012	2	1,2
2	Management approach to environmental responsibility	new	01.11.2012	1	3,4

[a] Article numbers are taken from articles table

Fig. 1 Sustainability Report (example)

Articles represent several indicators (Fig. 3). Two versions of one sustainability report are published in this list. While changing the report from the first to the second version, the status is changed from *new* to *in process*, date and article numbers are also changed simultaneously.

Articles in figure 2 summarize the key *energy* and *water* aspects of a company. An article also includes a number of additional fields, like ”report”, ”title”, ”status”, ”posted date”, ”version”, and a number of ”indicators” which refer to the table of indicators.

No.	title	text	status	date	version	indicators[bc]
1	Energy	Energy consumption is saved by using new energy sources	in process	11.11.2012	2	1
2	Water	Reduce water consumption achieved by optimization	in process	15.10.2012	2	3
3	Energy	Energy consumption is saved by using new energy sources	new	01.11.2012	1	2
4	Water	Reduce water consumption achieved by optimization	new	05.10.2012	1	4

[b] Indicator numbers are taken from indicators table
[c] Each article may consist of several indicators

Fig. 2 Articles in Report (example)

In this article list, two articles are published in two versions modifying ”status”, ”date”, and ”indicator”.

These indicators offer transparency in many respects. Their values and the evolution of these values help companies to identify potential improvements, steer programs, monitor target achievement, and inform the public about their performance and progress in compact form. Figure 3 represents exemplary indicators. These values represent environmental performance indicators of a company for the current reporting period e.g. one year, one month or specific period of time accordingly.

No.	indicator	name	value	date	version
1	EN3	Direct energy consumption in thousand megawatt hours	7,921	11.11.2012	2
2	EN8	Total water withdrawal by source in thousand cubic meters	2,220	15.10.2012	2
3	EN3	Direct energy consumption in thousand megawatt hours	8,688	01.11.2012	1
4	EN8	Total water withdrawal by source in thousand cubic meters	2,440	05.10.2012	1

Fig. 3 Indicators in Article (example)

In these reports, only some fields or some slices of data are changed, but other information still remains unchanged. For instance, EN3 indicator is posted two times in this list. Only status, date, and version fields are changed, but others are unchanged. If there are many versions of exactly the same report, article, or indicator with small changes in each version, it causes repeating (duplicating) the same information. Moreover, in the case of databases, there might be created empty cells with NULL values. According to the database approach, fields are typed and each field is given a range of memory even it is empty.

A *report* consists of a number of *articles*. Consequently, an article includes several *indicators*. Sustainability reports and associated data, as depicted in Fig.1–3, conform to the data schema in figure 4. To enable versioning of all relevant data, ”date” and ”version” are associated to each data type (cf. the *VersionObject* class in most right side of the figure).

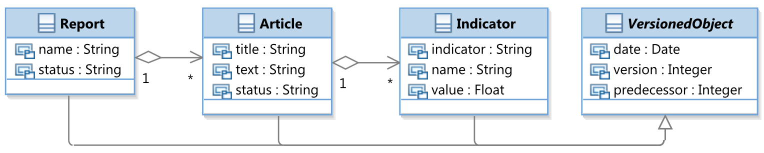

Fig. 4 A schema for sustainability reporting

The simplified schema in figure 4 presents only an excerpt data schema of the sustainability report within STORM. The complete STORM schema conforms the GRI C3 standard.

When there is a huge amount of sustainability reports and if all the reports are registered in a database, the database gets bigger and bigger. In addition, all data is integrated into a single database. Hence, sustainability reporting is a long term employment which constantly gathers information, registers information, and maintains them. When a database has vast amount of records with relationships among them, preparing sustainability reports require a lot of queries with many statements which are complicated and time consuming. Therefore, a tool support has to be efficient in time and space.

For current STORM, collecting all data in an identical database is a naive approach. Usually database is on-line and more users and different applications have access to the information. Only failure of database management system may cause additional risks with confidentiality, privacy and security which extremely important to companies.

Analysis of version histories is a key aspect of sustainability development. Due to report history analysis intentions, sustainability reports have to versioned and stored in separated document which is easy to access and obtain required information. Also, it is easy to analyse version histories by visualizing and browsing changes which have been done to sustainability reports and associated data over time.

According to the schema in figure 4 and exemplary reports in this section, sustainability reports and associated data can't be viewed as plain text documents. Since data structure of sustainability reports and associated data is more complex with constraints and behaviour, they can't be versioned, managed, and maintained by text-based version control systems [14]. Otherwise, it will loose efficiency of associated data concepts. In order to carry all associated data, a more consistent approach to versioning sustainability reports is required which satisfies all the requirements of associated data concepts. Thus, the new approach only has to store changes of modeling concepts in report versions and associated data, has to avoid redundancies, and has to provide efficient access to sustainability data versions and their evolution.

3 Versioning reports by Delta storage

According to requirements for versioning sustainability reports, mentioned in section 2, this section introduces a new approach to versioning sustainability reports by differences i.e. Deltas (Δ-storage) [1]. The approach is meta-model generic which operates with meta-model of domain instance-model. Instance model conforms to meta-model like sustainability reports conform to a data schema. According to discussions on previous section, concepts of sustainability reports correspond to software modeling concepts, consequently, data schema of sustainability reports fits to meta-model notations. So, the meta-model generic approach is applied to versioning sustainability reports. Applicability of the approach improves exploitation of it.

This schema (like the one given in Fig. 4) determines the notation to represent modeling deltas. So, the approach is schema-generic i.e it takes the data schema of sustainability reports (cf. figure 4) and generates a sequence of operations which represents a minimal set of possible activities to change report versions. The operations intend to represent only differences instead of making several copies of useless information.

There, two types of deltas are distinguished: *Forward delta* which leads from older version to newer version and *Backward delta* which leads from newer version to older version. Forward deltas represent versions according to their formation. Since retrieving the most current version in a forward delta representation requires to apply all deltas between the first and the current version, usually backward deltas are used. Thus, in backward delta representations, the current version is stored as delta according the empty model, such that the current version is accessible most efficiently. In the following, backward delta is taken into consideration, but a forward delta representation can be applied analogously.

Currently, most version control approaches provide managing and maintaining versions of plain text documents (for instance, *diff*, which is applied in RCS [13], CVS [14], SCCS [15], Git [12] and subversion [11]). Model differences are only viewed in a text-based way, but do not refer to modeling concepts. Text-based version control systems do not allow for directly representing and analyzing changes of modeling concepts, they only refer to more fine grained changes of character sequences loosing information on the documents abstract structure. Hence, further work has been done on versioning models directly. Cicchetti et al. [5] introduced a meta-model-independent approach in order to represent differences between consecutive versions of a software model. The approach is implemented by model transformations like ATL [8]. Alanen and Porres also explain operation based means to calculate differences, merge, and calculate the union of models [7]. A graph based approach to calculate differences between models is provided by SiDiff [9] which detects differences by representing models as graphs and using graph search algorithms to identify differences. SiDiff algorithm is applicable to various type of software models. But if sustainability models are represented by graphs, the graph approach is pretty slow and time consuming because of a vast amount of data. So, none of these approaches is applied to versioning sustainability reports. These approaches are mostly meta-model dependent which is not generic and can not be applied to independent domains.

In general, changes of models can be reduces to only three basic operations. Each modeling construct can be *added* or *deleted*; and each attribute value of a modeling construct can be *changed* (if they are initialised by default values). Thus, a generic approach relying on the appropriate meta model (only) requires to provide these operations for each modeling concept and each attribute, which also includes roles in associations.

In order to exemplify the approach, the data schema representing the structure of sustainability reports in figure 4 is taken as an example. Interfaces which summarise all operations required for representing model changes are depicted in figure 5. These interfaces are derived automatically from the schema in figure 4. During maintaining or evolving sustainability reports the three basic operations *add*, *delete* have to be applied to reports, articles, and indicators and the *change* operation has to be applied to attributes like `article.text` or `indicator.value`.

Each interface consists of a number of operations which derived from attributes of the schema by applying three atomic operations *add*, *delete*, and *change* to each notation of the data schema. Classes such as report, article, and indicators can be created or deleted while attributes of these classes can only be changed. In these interfaces, all the possible operations can be seen which can represent differences (changes) between versions of reports, articles, and indicators.

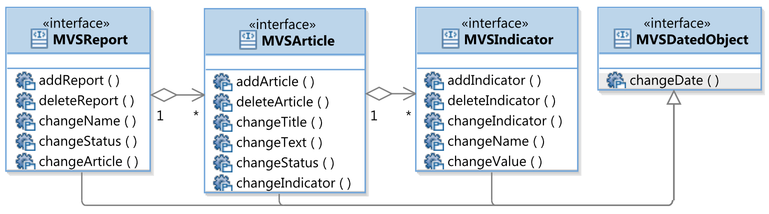

Fig. 5 An interface for sustainability reporting

According to the interfaces in figure 5, all the changes between report versions can be represented by a set of operations. In case of backward (forward) deltas, the last (first) version is stored as difference to the empty model. During maintaining sustainability reports, the changes are stored in a Delta document. As represented here, delta documents include a sequence of operations which carry changed information in sustainability reports. This also is an executable set of operations i.e. if these operations are applied to current version of a report, it results previous version of that report.

Due to implementation purposes, the operation-based model versioning approach is implemented by using the model transformation approach VIATRA [10]. Delta operations are represented by an executable model transformation rules. In general, these sequence of operations can be implemented by other transformation approaches, too. Also, it is being implemented by ATL model transformation approach [1] as a bachelor thesis.

In the ongoing example, the approach is applied to the specific schema within the concrete application sketched in Fig. 1–3. But, in the same way, the approach is applicable to other types of schemas or meta-models and operations can be derived in order to represent differences. Below, we are going to apply the approach to sustainability reports and associated data which presented in the last section.

The two versions of the sustainability report sketched in section 2 are represented by sequences of operations. In the *first version*, following modifications are done (with respect to the empty model):

- A report (here the 2nd version) is created in reports list (Fig. 1) with parameters like report name, report status, report date, and article numbers
  - *rp1 = addReport(”Management approach to environmental responsibility”, ”in process”,”11.11.2012”,”1,2”);*
- Two articles are created in the articles list (Fig. 2) with parameters article title, article text, article status, article date, and indicator numbers

– *ar1 = addArticle("Energy","Energy consumption is saved by using new energy sources","in process","11.11.2012","1");*
– *ar2 = addArticle("Water","Reduce water consumption achieved by optimization","in process","15.10.2012","3");*

- Two indicators are created in indicators list (Fig. 3) with parameters like indicator code, indicator name, indicator value, and posted date
  – *in1 = addIndicator("EN3","Direct energy consumption in thousand megawatt hours","7,921","11.11.2012");*
  – *in2 = addIndicator("EN8","Total water withdrawal by source in thousand cubic meters","2,220","15.10.2012");*

In the *first version*, the change operations represent those transformations which are required to derive the previous (here first) version from its successor (here second version) :

- The existing *report* is changed from the second version to the first version including status, date, and a number of articles (Fig. 1). Here, *rp1* is a variable of the report:
  – *rp1.changeStatus("new");*
  – *rp1.changeDate("01.11.2012");*
  – *rp1.changeArticle("3,4");*
- *Articles* are changed from the second version to the first version including status, date, and indicators. Figure 2 contains two articles with one extra version for each of them. *ar1* and *ar2* are variables, consequently:
  – *ar1.changeStatus("new");*
  – *ar1.changeDate("01.11.2012");*
  – *ar1.changeIndicator("2");*
  – *ar2.changeStatus("new");*
  – *ar2.changeDate("05.10.2012");*
  – *ar2.changeIndicator("4");*
- *Indicators* are changed from the second version to the first version including values and dates, Figure 3 contains two indicators with one extra version for each of them. Here *in1* and *in2* are variables:
  – *in1.changeValue("8,688");*
  – *in1.changeDate("01.11.2012");*
  – *in2.changeValue("2,440");*
  – *in2.changeDate("05.10.2012");*

As shown here, all the modifications are represented by operations (cf. Fig. 5), derived from the original meta model representing the documents structure. Here, a preceding version only represents changes to the next version, the unchanged information maintains as it was. It lets to store only differences instead of making multiple useless copies of the same data. Also, it is easy to use later when it is needed. Delta carries reports and associated data with all constraints regarding the data schema. Therefore, version histories of sustainability reports about any point or whole process of sustainability development can be analysed by visualizing and browsing changes.

4 Conclusion and Outlook

In this paper, a meta-model generic, operation-based model versioning approach was applied to versioning sustainability reports in STORM project.

The approach introduces new means to versioning sustainability reports by delta storage to differences between consecutive versions of a report instead of making redundant copies of complete reports. The approach also provides to analyse histories of sustainability reports. Following the meta-model based versioning approach, the evolution of each modeling construct is accessible directly and independently. By analysing histories and changes on sustainability reports, stakeholders of a company will have a chance to manifest the evolution of sustainability data, manage risks, estimate company's future, and draw outline regarding different performances.

The here presented approach was also applied to versioning UML activity models [1]. It is implemented by model transformation approach [8] as a bachelor thesis. For now, the approach is stated as a prototype but it is intended to completely embedded the approach to STORM, in order to establish versioning of sustainability reports.

References

1. Kuryazov, Dilshodbek; Jelschen, Jan; Winter, Andreas: Describing Modeling Delta By Model Transformation, Gesellschaft fr Informatik, In: Softwaretechnik Trends (Issue on International Workshop on Comparison and Versioning of Software Models (CVSM 2012)), (2012).
2. Solsbach A., Spke D., Wagner vom Berg B., Marx Gómez J.: Sustainable Online Reporting Model – a Web Based Sustainability Reporting Software. In: ITEE 2011 - Proceedings of the 5th International Symposium on Information Technologies in Environmental Engineering, Poznan, Poland (2010).
3. Grady Booch, Ivar Jacobson, Jim Rumbaugh: OMG Unified Modeling Language Specification, Version 1.3, First Edition (2000).
4. Sustainability Reporting Guidelines. Copyright 2000-2011 GRI. Version 3.1, https://www.globalreporting.org/resourcelibrary/G3.1-Sustainability-Reporting-Guidelines.pdf
5. Cicchetti, Antonio; Ruscio, Davide Di; Pierantonio, Alfonso.: A Metamodel Independent Approach to Difference Representation. In: Journal of Object Technology 6:9. p. 165-185. (2007).
6. Herrmannsdrfer, Markus: Operation-Based Versioning of Metamodels With COPE. In: Proc. 2009 ICSE Workshop on Comparison and Versioning of Software Models, IEEE, p. 49-54 (2009).
7. M. Alanen and I. Porres: Difference and union of models. In: Proc. 6th Int. Conf. on the UML, Springer, volume 2863 of LNCS, p. 2-17 (2003).

8. F. Jouault and I. Kurtev: Transforming Models with ATL. Springer-Verlag, volume 3844 of LNCS, In: MoDELS Satellite Events, p. 128138 (2005).
9. Pit Pietsch: The SiDiff Framework. Technical report. University of Siegen, Germany (2009).
10. D. Varro, A. Balogh: The model transformation language of the VIATRA2 framework. Science of Computer Programming. Budapest, Hungary. p. 214-234 (2007).
11. Ben Collins-Sussman, Brian W. Fitzpatrick, C. Michael Pilato.: Version Control with Subversion. OReilly Media. (2004).
12. Jon Loeliger. Version Control with Git: Powerful Tools and Techniques for Collaborative Software Development. O'Reilly Media (2009).
13. Walter F. Tichy: RSC - a system for version control. Software – Practice Experience, Volume 15, Issue 7, (1985).
14. Tom Mens and Serge Demeyer: Concurrent Versions Systems. Springer (2008).
15. Alan L. Glasser: The evolution of a Source Code Control System. ACM SIGMETRICS Performance Evaluation Review, Volume 7, Issue 3-4 (1978).
16. Isenmann, R., Marx Gómez, J.: Internetbasierte Nachhaltigkeitsberichterstattung: Mageschneiderte Stakeholder-Kommunikation mit IT. Berlin: Schmidt (2008).