Gap Analysis in Enterprise Architecture using Semantic Web Technologies

Philipp Diefenthaler[1,2] and Bernhard Bauer[2]
[1]*Softplant GmbH, Munich, Germany*
[2]*Institute for Software & Systems Engineering, University of Augsburg, Augsburg, Germany*

Keywords: Enterprise Modelling, Gap Analysis, Semantic Web.

Abstract: Enterprise architectures (EA) can be used for analyses in different ways and thus can support the decision making process that has to cope with an increasing number of changes, the clarification of the extent of changes and the complexity of these changes. A gap analysis is used in the context of EA to identify differences between two states of an EA. Formal models of an EA allow tool support and the visual representation of these models. This paper shows how a gap analysis can be performed using semantic web technologies on a high-level current and target state of an EA. With the results of the gap analysis and a detailed current state it is possible to show a migration path from the current state of the EA to a plan or target EA.

1 INTRODUCTION

An enterprise architecture (EA) can be used to represent the enterprise and its underlying information technology in models that can support decision making. Such EA models cover aspects from business, processes, integration, software and technology (Winter and Fischer, 2006). To cope with the inherent complexity of the elements' relationships, the number of stakeholders involved, and the change of internal and external conditions it is crucial for enterprises to use a managed approach to steer and control the redesign of the EA. In order to be able to plan the change it is necessary to have a plan basis, i.e. the current state of the EA, and to know the goal of planning activities, i.e. the target state of the EA. According to Pulkkinen and Hirvonen (2005); Pulkkinen (2006) the planning activities using an EA take place at different decision levels. These levels are named enterprise level, domain level and system level. Each of them vary in detail and levels of abstraction seem to be inevitable (Pulkkinen, 2006). The need to change and the resulting moving target (Niemann, 2006) is a challenge EA planning has to cope with and can be supported by tools. In practice there exist already several tools for EA planning. However, none of them provides the functionality described in this paper. This paper shows how the gap analysis can be performed on two high-level EA models representing the current and target state of an EA using semantic web technologies. Furthermore, it is shown how successor relationships can be added. Starting with the identified gaps, successor relationships and a detailed current EA state at hand it is possible to support the migration to a detailed target state.

The paper is structured as follows: Section 2 gives a short overview for using EA models for planning purposes and shortly introduces semantic web technologies relevant for this paper. In Section 3 related work relevant for the gap analysis is presented. The concept how to detail the target state by using the gap analysis and semantic web technologies is presented in Section 4. The paper closes in Section 5 with a summary and outlook for further research.

2 FOUNDATIONS

This section gives an introduction to the foundations of enterprise architecture models and their usage for planning purposes. Furthermore, semantic web technologies and two of their key technologies relevant for this paper are presented.

2.1 Enterprise Architecture Models

According to Buckl and Schweda (2011) enterprise architecture management (EAM) is a management cycle that consists of the phases plan, do, check and act

(Deming Cycle (Deming, 1994)). The plan phase is concerned with developing change proposals that are implemented in the do phase. Within the check phase differences between intended and actually achieved results are controlled. Based upon the results from the check phase the act phase provides input to the plan phase by supplying information for the next plan phase. Models of an enterprise, as an abstraction from reality, can support the plan phase as part of an enterprise architecture management approach (Aier and Gleichauf, 2010a; Buckl et al., 2009).

A model of an EA can be used to represent the architecture of an enterprise at different points in time (Buckl and Schweda, 2011). The current state of the architecture is a documented state at the present point in time and serves as a starting point for defining a target state. The target state represents a goal state in the future which can be used to guide the development of an enterprise architecture from the current towards a target state. The development of a target state highly depends on the enterprises' EA goals. It is influenced by business requirements, strategic goals and IT objectives like master data consolidation, improve the flexibility of IT and drive the coverage of standard platforms Hanschke (2009). Which factors and how exactly they influence the target architecture highly depends on the architecture method applied and how it is integrated into the enterprise. Between the current and target state it is possible to develop planned states which can represent alternative states, e.g. two planned states exist for the same point in time, or are successors of the current and predecessor states of the target. A gap analysis, sometimes also referred to as delta analysis, is the comparison between two different states of an enterprise architecture that is used to clarify the differences between those two states. Different states that can be compared are current to target, current to planned, planned to target and planned to planned (Buckl and Schweda, 2011).

2.2 Semantic Web Technologies in a Nutshell

Semantic web technologies can be used to integrate heterogeneous data sets and formalize the underlying structure of the information to allow a machine to understand the semantics of it (Shadbolt et al., 2006). The World Wide Web Consortium (W3C) provides a set of standards to describe an ontology and to query it. Two standards are of relevance for this paper: firstly, the Web Ontology Language (OWL) (Bechhofer et al., 2004), which is capable of describing the state of an EA model and secondly, the SPARQL Query Language for RDF (SPARQL) (Prud'hommeaux and Seaborne, 2008), which allows querying these models. The Resource Description Framework (RDF) (Manola et al., 2004) is a basis for both standards, as OWL ontologies can be represented as RDF graphs and can be accessed via SPARQL. A RDF graph consists of triples of the form 'subject, predicate, object'. Every information in an ontology can be identified by a resource identifier which contains a namespace, which allows for example distinguishing between a bank in a financial context and a bank of a river. Semantic web technologies have already been applied to several different applications that range from semantic business process modelling (Lautenbacher, 2010) to diagnosis of embedded systems (Grimm et al., 2012). First implementations based upon semantic web technologies for EA management already exist from TopQuadrant with its TopBraid Composer [1] and Essential Project [2].

3 RELATED WORK

In this section related work for gap analysis is introduced. As a starting point the technical report 'On the State of the Art in Enterprise Architecture Management Literature' (Buckl and Schweda, 2011) was taken, as they consider the gap (delta) analysis as part of the different approaches. The focus in this paper is on using the gap analysis as method to detect differences between a current and a target state. Besides the listed approaches in the technical report an approach from the University of Oldenburg was identified as relevant for the purpose of this paper.

3.1 Gap Analysis - University of Oldenburg

The Institute for Information Technology of the University of Oldenburg presents a tool supported approach for performing a gap analysis on a current and ideal landscape (Postina et al., 2009; Gringel and Postina, 2010). The approach is tightly coupled to the Quasar Enterprise (Engels et al., 2008) approach, which can be used to develop service-oriented application landscapes. In order to be able to perform their gap analysis it is necessary to model the current application landscape consisting of current components, current interfaces, current operations and business objects. The ideal landscape is modelled with ideal

[1] www.topquadrant.com/docs/whitepapers/WP-BuildingSemanticEASolutions-withTopBraid.pdf

[2] http://www.enterprise-architecture.org/

components, ideal interfaces, ideal operations and domains. Based on these two models the tool is capable to generate a list of actions that would, if all were applied, result in the ideal landscape. Within the paper the suggested procedure for selecting actions is to allow an architect to select certain actions that result in a target. Furthermore, the tool is capable to provide metrics for quantitative analysis of the application landscape. Gringel and Postina state that the gap analysis needs a "detailed level of description when it comes to modelling both landscapes" ((Gringel and Postina, 2010), p. 283) and as a result the "data necessary to perform gap analysis on the entire application landscape on a detailed level considering operations is overwhelming" (Gringel and Postina (2010), p. 291). How the different actions interfere with each other is not considered and actions can only be provided if an ideal landscape with all details has been modelled.

3.2 Gap Analysis - University of St. Gallen

The EAM approach of the University of St. Gallen uses the gap analysis as a starting point to plan the transformation by identifying the differences between two states considering changed elements. Aier and Gleichauf distinguish between a macro and micro level of enterprise models, e.g. a current and target state, whereas on the micro level detailed information about successor relationships of the elements is available (Aier and Gleichauf, 2010b). This information is kept in a so called transformation model. Furthermore, information about changed relationships is part of the transformation model. Aier and Gleichauf (2010a) suggest to compare the graphs of the different architecture states to gain information about necessary changes. Afterwards, it is possible to derive six different successor relationships between the elements of the different model states and store this information in a transformation model. How the successor relationships are derived or if they are modelled manually is not described. Furthermore, changed dependencies are not considered as part of the changes.

3.3 Gap Analysis - Strategic IT Managment by Hanschke

The 'Strategic IT Management' (Hanschke, 2009) approach is intended to serve as a toolkit for EAM by providing best-practices derived from work experience. After a target state has been modelled and agreed upon the gap analysis is used to detect differences between the current and target state. The gap analysis is performed on the basis of process support maps visualizing which information systems support which business processes (x-axis) and which customer group (y-axis) the information systems are assigned to. For a more fine grained gap analysis Hanschke suggests to additionally add information about interfaces and information objects of the information systems. Afterwards, for each gap possible actions to close the gap are considered. These actions range from introducing a new information system, adding or reducing functionality of an existing information system, changing or adding interfaces to the shut down of information systems and interfaces. Based upon the results of the gap analysis and derivation of appropriate actions it is necessary to clarify dependencies between the actions, bundle the actions and create planned states as recommendations for change. As far as we were able to verify the limitations of the tool and approach it is not possible to create suggestions for a detailed target state.

4 USING SEMANTIC WEB TECHNOLOGIES FOR GAP ANALYSIS

The presented example in this section is similar to the examples given in (Hanschke, 2009) as these provide detailed information on the current state and high-level information of the current and target state. The goal of the proposed approach is to deliver a more detailed target state by making suggestions to a user how a detailed target state could look like, derived from the gaps identified in the high level states and a detailed current state. The ontology editor Protégé [3] was used to model the information model and the current and target state.

4.1 Information Model in OWL

Figure 1 shows the classes and object properties, i.e. the information model, of the ontology which are relevant for modelling the current and target state on a high- as well as on a detailed level [4]. It consists of the classes Customer Group, Business Process, Information System, Interface and Information Object. Object properties are used to relate classes to each other whereas the arrow indicates the direction to determine the domain, i.e. source, and range, i.e. target, of an object property. For example an Information System

[3]http://protege.stanford.edu/overview/protege-owl.html

[4]Please note that there exists no standard graphical notation for ontologies

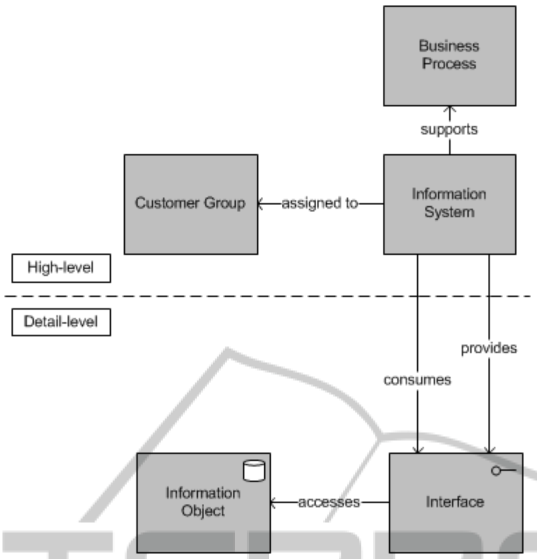

Figure 1: Information Model distinguishing High-level and Detail-level.

(class) *supports* (object property) a Business Process (class). Cardinalities of object properties are not represented in Figure 1 but all object properties are one to many, as e.g. an Information System may support several business processes or may be assigned to several customer groups. Data properties, i.e. attributes of classes, were not modelled. The information model in Figure 1 can represent the business support information systems and their assignment to customer groups in a simple way. However, there may be situations where the presented information model could not reconstruct the business support of information systems in an appropriate way. In this case it is necessary to introduce a 'BusinessSupport' class, which is connected with exactly one process and one customer group (c.f. Buckl et al., 2009). For our examples we use the simpler version, as it can be used for all examples that have to be modelled. However, the approach presented in this paper is not limited to the simpler version of the information model but is easier to describe and understand.

4.2 Modelling Current and Target State

Figure 2 shows an excerpt of a process support map representing a current and target state of an enterprise architecture model. The underlying information in the ontology for this information is modelled as follows exemplified with the information system Elcaro CRM in the current state. Elcaro CRM *supports* the business process Sales Governance and is *assigned to* Enterprise and Institutional Clients. In the target state Elcaro CRM *supports* the business process Customer Management and is *assigned to* Private, Enterprise and Institutional Clients. The instances of classes, of the current as well as the target state are based on the same classes. First, a current state was modelled as shown in (Hanschke, 2012, Figure C.2, p.7). Afterwards, a target state was modelled by reusing the model of the current state and changing it to the desired target state (Hanschke, 2012, Figure C.3, p.7). Every information system that is modelled *supports* at least one business process and is at least *assigned to* one customer group.

Results of the Modelling

The result of this phase are two ontologies:
currentState = modelled ontology of the current state of the enterprise architecture
targetState = modelled ontology of the target state of the enterprise architecture
A copy of the current state needs to be kept in order to be able to perform the gap analysis later on.

4.3 Performing a Gap Analysis

The gap analysis was performed using OWLDiff (Kremen et al., 2011). It is a plugin for Protégé, which can be used to compare and merge OWL ontologies. We used OWLDiff to compare the modelled current and target state. Two result sets *onlyCurrentState* and *onlyTargetState*, which are relevant for the proposed approach, are produced by OWLDiff. In order to be able to compare the two ontologies it is necessary that current and target state have the same namespace.

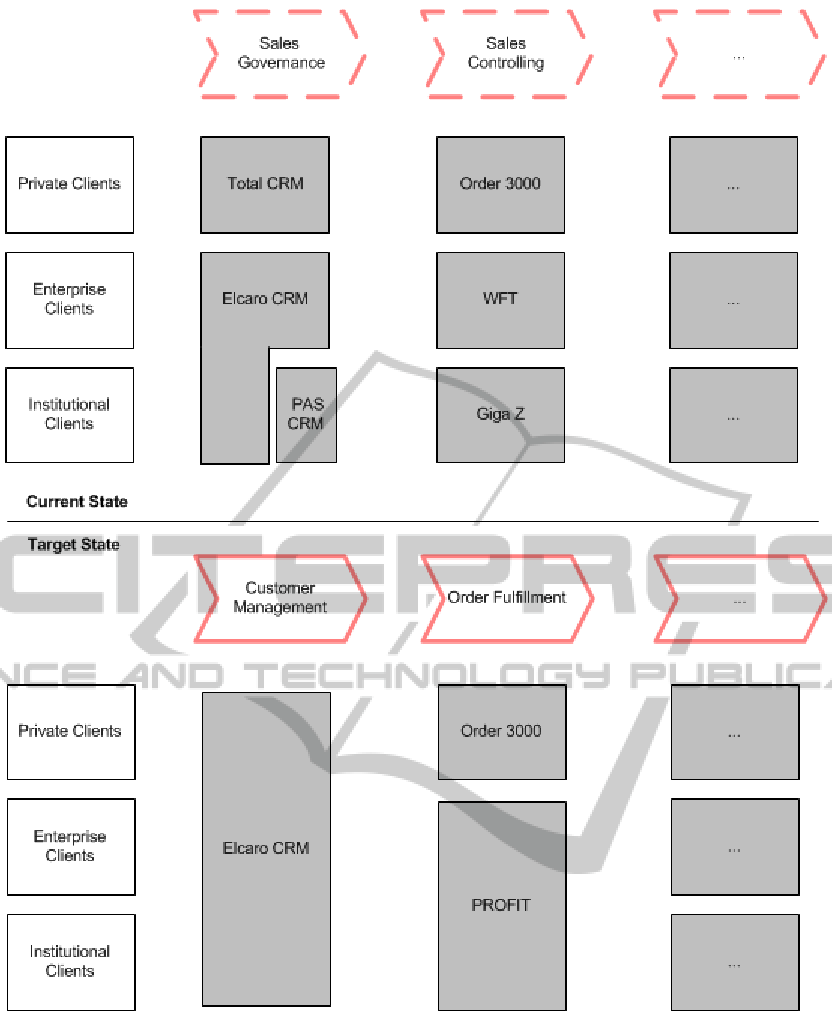

Figure 2: Excerpt of a Process Support Map for Current and Target State.

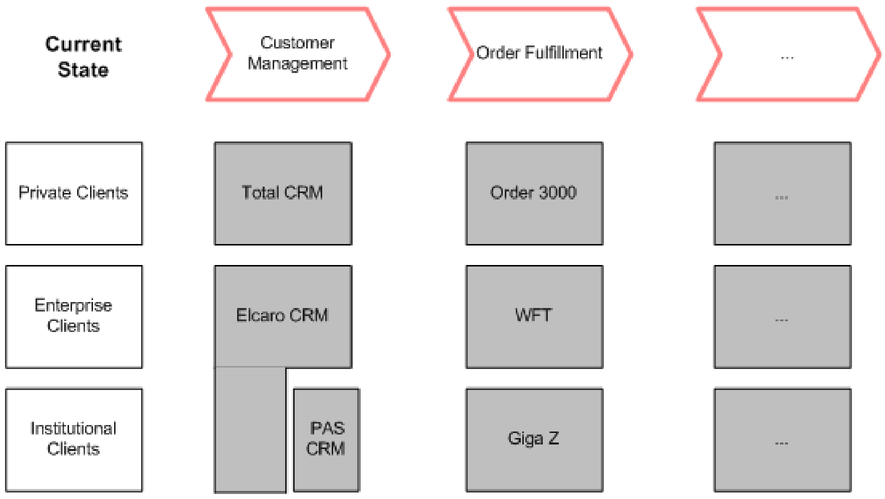

Figure 3: Current State after Localization.

Results of the Gap Analysis

onlyCurrentState is the set of classes, object properties, data properties and instances that only exist in the model of the current state.

$onlyCurrentState = \{x \mid \forall x : x \in currentState$
$\wedge x \notin targetState\}$

In contrast, *onlyTargetState* is the set of classes, object properties, data properties and assertions that only exist in the target state.

$onlyTargetState = \{x \mid \forall x : x \notin currentState$
$\wedge x \in targetState\}$

Both sets consist only of the changed instances and their changed object properties, as data properties were not modelled and the information model, in Figure 1, remained unchanged between the current and target state. In accordance to Figure 2 this means that the information system PAS CRM is part of the set *onlyCurrentState*, as it is not present in the target state. The assertion Elcaro CRM *assigned to* Private Clients belongs to the set *onlyTargetState* as this relationship is only present in the target state. Order 3000 and its object property assertions are neither part of *onlyCurrentState* nor *onlyTargetState*. The business processes between the current and target state also changes and thus Sales Governance and Sales Controlling are also part of *onlyCurrentState*. Furthermore, the business processes Customer Management and Order Fulfillment belong to *onlyTargetState*. The proposed solution in Hanschke (2009) is to create a common localization, i.e. the same customer groups and business processes as in the target state, for the information systems of the current state. This change was modelled manually as proposed by Hanschke. Figure 3 shows the information systems of the current state with the same localization as for the information systems in the target state. The gap analysis can be performed again and the changed business processes are no longer part of *onlyCurrentState* and *onlyTargetState*.

4.4 Setting the Successor Relationships for Information Systems

In order to be able to set the successor relationships for information systems the ontology of the target state is transferred to a different namespace than the current state and a transformation ontology is created that contains the information about successor relationships (Aier and Gleichauf, 2010b). Changing the namespace of an OWL ontology can be done in Protégé. The set of information systems, business processes and customer groups are defined as follows:

$businessProcess = \{x \mid \forall x : x \text{ isA Business Process}\}$
$informationSystem =$
$\{x \mid \forall x : x \text{ isA Information System}\}$
$customerGroup = \{x \mid \forall x : x \text{ isA Customer Group}\}$

The successor relationship for information systems is defined as:

$successor\ (x,y) \equiv x,y \in informationSystem$
$\wedge x \in targetState \wedge y \in currentState$
$\wedge \exists b : b \in businessProcess \wedge x \text{ supports } b$
$\wedge y \text{ supports } b \wedge \exists c : c \in customerGroup$
$\wedge x \text{ assigned to } c \wedge y \text{ assigned to } c\}$

As the target and current state do not have the same namespace for setting the successor relationships it is necessary to include information which business process in the current state is the same as in the target state. This was modelled manually in the Protégé tool. An alternative is to relate the information systems of the target state to the customer groups and business processes of the current state. We did not use this alternative as we used a copy and transferred it to a different namespace. However, we recommend to include this information as otherwise, the queries on the models have to include the information which business process of the current state is the same business process in the target state. An automated addition of this information can be implemented, as the results from the gap analysis show that there are no changes in business processes and customer groups.

Setting the successor relationship can be done with SPARQL queries using the CONTSRUCT command, which allows to use SPARQL as a simple rule language. This task cannot be performed in Protégé. In our case the rule is that if an information system supports a certain business process and is assigned to a certain customer group the element in the target state is a successor of the information system that supports the same business process and is assigned to the same customer group.

We modelled the information of the same individuals manually. For each information system in the current state check which information system with the same localization is in the target state and create a successor relationship. With the successor relationships at hand it is possible to identify the dependency type for information systems which can be divided into *noSuccessor*, *noPredecessor*, *oneToOne*, *oneToMany*, *manyToOne*, and *manyToMany*. All information systems in onlyCurrentState that do not have a successor belong to the set *noSuccessor* whereas all information systems that belong to onlyTargetState and do not have an incoming successor relationship belong to the set *noPredecessor*. The set *oneToOne* consists of the pairs of information systems that have exactly one successor and this successor has only one predecessor. *oneToMany* is the set of information systems that have several successors in the target state whereas the set *manyToOne* is the set of information systems which have the same successor in the target state. *manyToMany* is the set of information systems which have several predecessors and successors. To which set an information system belongs to can also be determined by SPARQL queries.

A successor relationship is part of exactly one of the above sets. Please note that within the six different sets disjoint subsets exist. From Figure 2 two disjoint subsets of the *manyToOne* set could be derived. The first subset consisting of Total CRM (successor), PAS CRM (successor) and Elcaro CRM (predecessor) whereas the other consists of Giga Z (successor), WFT (successor) and PROFIT (predecessor). For the *noSuccessor* and *noPredecessor* set each information system represents a disjoint subset. In order to make suggestions the model of the current state is now detailed considering interfaces and information objects (c.f. Figure 1). With the detailed information of the current state and the successor relationships at hand it is possible to generate suggestions how a detailed target state could look like.

4.5 Creating Suggestions for a Detailed Target State

Depending on the successor set an information system belongs to different suggestions are made and a user can follow or overrule them. By following a suggestion or not the target is stepwise getting more detailed, as all sets of successor relationships are getting processed. The result is a detailed target state. At first all provided interfaces are transferred to the detailed target state. Then the consumes dependencies can be added to the detailed target state.

4.5.1 Suggestions for Provided Interfaces

1. *noSuccessor* set: for each provided interface in the current state check if it is consumed by an information system that is part of the target state or the consuming information system has a successor relationship.
   (a) If there are any information systems it is necessary to check if they still can work properly without consuming the interface.
   (b) Otherwise, no information from the current state is added to the target state.
2. *noPredecessor* set: it is not possible to suggest a detail for the target state as there exists no detail in the current state. A manual addition of provided interfaces and their information objects in the target state is necessary.
3. *oneToMany* set:
   (a) If the predecessor is part of *onlyCurrentState* all provided interfaces of the predecessor, including their information objects, are suggested to be provided by one of the successor information systems.
   (b) Otherwise, all provided interfaces and information objects of the predecessor are suggested to be provided by one of the successor information systems or the remaining part of the predecessor in the target state.
4. *manyToOne* set:
   (a) If the successor is part of *onlyTargetState* it is suggested to provide each interface of its successors, but only one per information object.
   (b) Otherwise, it is suggested that the successor provides the interfaces already provided in the current state, i. e. by itself, and provide all interfaces of the other predecessors, but only one per information object.
5. *manyToMany* set: All provided interfaces are suggested to be provided by one of the successors. If more than one predecessor provides an interface with the same information object the suggestion is to provide only one interface in the target state with such an information object. Further suggestions were not identified as this type represents a complex type of restructuring. Nevertheless, the user should be supported with information about information systems changing business support and assigned customer groups. Furthermore, information which information systems belong to *onlyCurrentState* and *onlyTargetState* needs to be presented to the user.
6. *oneToOne* set: all interfaces, including their information objects, provided by the predecessor are suggested to be provided by the successor.
7. Furthermore, the user can model additional provided interfaces or let suggested interfaces be provided by an information system that is not a successor of the information system that provided it in the current state.
8. For each interface information is stored if it is the successor of one or more interfaces in the current state. This is necessary to allow a sound migration planning (Aier and Gleichauf, 2010a).

As a result all provided interfaces have been modelled in the target state including their information objects. Furthermore, the information about successor relationships of the interfaces is available.

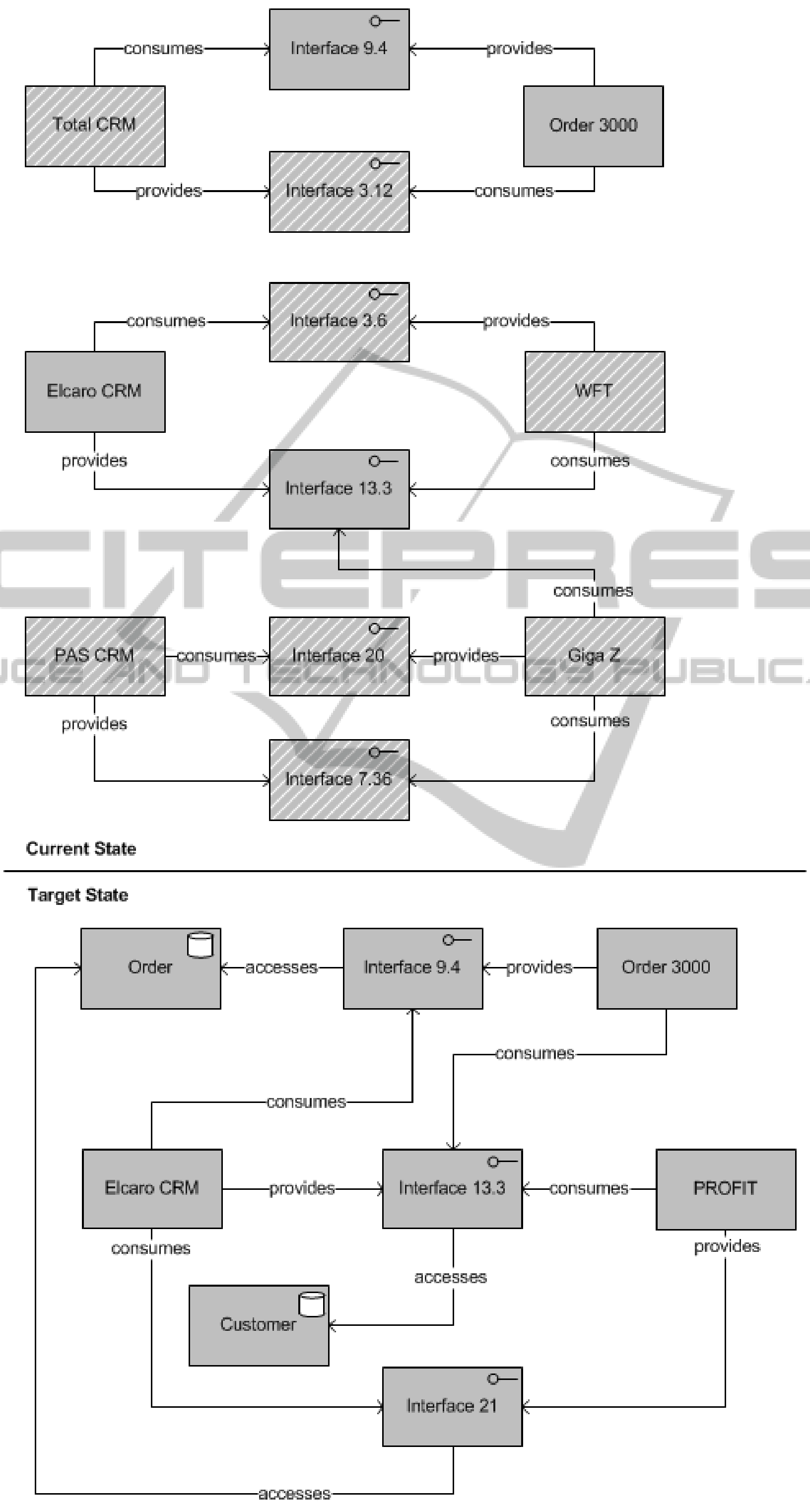

Figure 4: Excerpt of detailed Current and an exemplary Target State.

4.5.2 Suggestions for Consumed Interfaces

1. *manyToMany* set: all consumed interfaces of predecessors are suggested to be consumed by at least one successor. The user can choose if more than one successor consumes the interface of a predecessor.
2. *oneToOne* set: all interfaces consumed by the predecessor are suggested to be consumed by the successor.
3. *manyToOne* set: consumed interfaces of the predecessors are suggested to be also consumed in the target state.
4. *oneToMany* set:
   (a) If the predecessor is part of *onlyCurrentState* all consumed interfaces of the predecessor are suggested to be consumed by one of the successor information systems.
   (b) Otherwise, all consumed interfaces of the predecessor are suggested to be consumed by one of the successor information systems or the remaining part of the predecessor in the target state.
5. *noPredecessor* set: which interfaces are consumed by the information system need to be modelled manually as no information from the current state is available.
6. *noSuccessor* set: as the information system does not exist in the target state no information about consumed interfaces needs to be added to the target state.
7. Furthermore, the user can model additionally consumed interfaces for every information system.

4.5.3 Results of the Guided Refinement

The result is a detailed target state including provided and consumed interfaces with related information objects. Figure 4 shows an excerpt of a detailed current and target state. The dashed boxes in the current state indicate that the information systems belong to the set *onlyCurrentState* and their provided interfaces are suggested to be removed. Consistency checks can be performed on the target state with SPARQL queries to check whether interfaces exist which are provided but no longer consumed by any information system. Afterwards, the namespace of the modelled detailed target state is changed to the namespace of the detailed current state and the gap analysis can be performed again. The user gets the detailed gaps between current and target state.

With the results of the gap analysis and a detailed current state it is possible to assist a user in modelling a detailed target state by making suggestions how to detail it based on the current state. The variety of suggestions that can be provided is limited to the information model. For example, technical information about the interfaces can be added to allow more sophisticated suggestions, like to prefer web service technology for interfaces of information systems that have to be build. Furthermore, the presented approach should be evaluated in a real world setting with enterprises that use business support maps for planning purposes. The creation of the OWL ontologies and the SPARQL queries is also a task that needs expert knowledge of semantic web technologies. Nevertheless, the proposed approach shows the ability of semantic web technologies to assist in the planning process without being limited to a certain methodology or information model regarding the gap analysis. It was also presented how the creation of successor relationships between information systems of the current and target state can be added automatically, using business support maps. Another advantage of the proposed solution is the possibility to suggest solutions for the target state only having defined a high level target state.

5 SUMMARY

It was shown that semantic web technologies are capable to perform the gap analysis on a current and target state on a high level as well as on a detail level. Furthermore, the approach proposed how suggestions for a user can be generated from the current state to assist in the modelling of a detailed target state in detail. Metrics were not taken into account in the proposed approach. The further elaboration of metrics and their relation to the information model needs to be considered in order to allow a quantitative analysis of the current and target state. Furthermore, Protégé requires knowledge of semantic web technologies and is not ready to use for enterprise architects. Expert knowledge is necessary for the OWL ontology creation, maintenance as well as for the SPARQL queries. To implement the presented approach in an EA tool, which is ready for production, is also accompanied with an effort. Nevertheless, the semantic web technologies offer the capability to perform the gap analysis and can be leveraged for planning support. Future work should also address the capability of semantic web technologies for automated documentation of EA models.

REFERENCES

Aier, S. and Gleichauf, B. (2010a). Application of enterprise models for enginnering enterprise transformation. *Enterprise Modelling and Information Systems Architectures*, 5(1):56–72.

Aier, S. and Gleichauf, B. (2010b). Towards a systematic approach for capturing dynamic transformation in enterprise models. In Sprague, R. H., editor, *Proceedings of the 43rd Annual Hawaii International Conference on System Sciences*. IEEE Computer Society.

Bechhofer, S., van Harmelen, F., Hendler, J., Horrocks, I., McGuiness, D., Patel-Schneider, P. F., and Stein, L. A. (2004). *OWL Web Ontology Language Reference*. World Wide Web Consortium. Retrived November 5, 2012, from http://www.w3.org/TR/owl-ref/.

Buckl, S., Ernst, A. M., Matthes, F., and Schweda, C. M. (2009). An information model capturing the managed evolution of application landscapes. *Journal of Enterprise Architecture*, 5(1):12–26.

Buckl, S. and Schweda, C. M. (2011). *On the State-of-the-Art in Enterprise Architecture Management Literature*. Technical Report, Technische Universität München, Chair for Software Engineering of Business Information Systems.

Deming, W. E. (1994). *Out of the crisis: Quality, productivity and competitive position*. Cambridge University Press, Cambridge, 19 edition.

Engels, G., Hess, A., Humm, B., Juwig, O., Lohmann, M., and Richter, J.-P. (2008). *Quasar Enterprise: Anwendungslandschaften serviceorientiert gestalten*. Dpunkt-Verlag, Heidelberg, 1 edition.

Grimm, S., Watzke, M., Hubauer, T., and Cescolini, F. (2012). Embedded el + reasoning on programmable logic controllers. In Hutchison, D., Kanade, T., Kittler, J., Kleinberg, J. M., Mattern, F., Mitchell, J. C., Naor, M., Nierstrasz, O., Pandu Rangan, C., Steffen, B., Sudan, M., Terzopoulos, D., Tygar, D., Vardi, M. Y., Weikum, G., Cudré-Mauroux, P., Heflin, J., Sirin, E., Tudorache, T., Euzenat, J., Hauswirth, M., Parreira, J. X., Hendler, J., Schreiber, G., Bernstein, A., and Blomqvist, E., editors, *The Semantic Web – ISWC 2012*, Lecture Notes in Computer Science, pages 66–81. Springer Berlin Heidelberg, Berlin and Heidelberg.

Gringel, P. and Postina, M. (2010). I-pattern for gap analysis. In Engels, G., Luckey, M., Pretschner, A., and Reussner, R., editors, *Software engineering 2010*, Lecture Notes in Informatics, pages 281–292. Gesellschaft für Informatik, Bonn.

Hanschke, I. (2009). *Strategisches Management der IT-Landschaft: Ein praktischer Leitfaden für das Enterprise Architecture Management*. Hanser, München, 1. edition.

Hanschke, I. (2012). C planungs-muster: Download-anhang zum buch strategisches management der it-landschaft. Retrieved December 5, 2012, from http://files.hanser.de/hanser/docs/20100621_21621165557-63_ HanschkeDownloadAnh%C3%A4nge_final.zip.

Kremen, P., Smid, M., and Kouba, Z. (2011). Owldiff: A practical tool for comparison and merge of owl ontologies. In *22nd International Workshop on Database and Expert Systems Applications*, pages 229–233. IEEE Computer Society.

Lautenbacher, F. (2010). *Semantic business process modeling: Principles, design support and realization*. Shaker, Aachen.

Manola, F., Miller, E., and McBride, B. (2004). *RDF Primer*. Retrived November 5, 2012, from http://www.w3.org/TR/rdf-primer/.

Niemann, K. D. (2006). *From enterprise architecture to IT governance: Elements of effective IT management*. Vieweg, Wiesbaden.

Postina, M., Sechyn, I., and Steffens, U. (2009). Gap analysis of application landscapes. In *2009 13th Enterprise Distributed Object Computing Conference Workshops*, pages 274–281. IEEE Computer Society.

Prud'hommeaux, E. and Seaborne, A. (2008). *SPARQL Query Language for RDF*. Retrived November 5, 2012, from http://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115/.

Pulkkinen, M. (2006). Systemic management of architectural decisions in enterprise architecture planning. four dimensions and three abstraction levels. In *Proceedings of the 39th Annual Hawaii International Conference on System Sciences (HICSS'06)*, page 179a. IEEE Computer Society.

Pulkkinen, M. and Hirvonen, A. (2005). Ea planning, development and management process for agile enterprise development. In *Proceedings of the 38th Annual Hawaii International Conference on System Sciences*, page 223c. IEEE Computer Society.

Shadbolt, N., Berners-Lee, T., and Hall, W. (2006). The semantic web revisited. *IEEE Intelligent Systems*, 21(3):96–101.

Winter, R. and Fischer, R. (2006). Essential layers, artifacts, and dependencies of enterprise architecture. In *2006 10th IEEE International Enterprise Distributed Object Computing Conference Workshops (EDOCW'06)*, page 30. IEEE Computer Society.