5-15-2012

SUPPORTING ENTERPRISE TRANSFORMATION USING A UNIVERSAL MODEL ANALYSIS APPROACH

Matthias Steinhorst
University of Münster

Dominic Breuker
University of Münster

Patrick Delfmann
University of Münster

Hanns-Alexander Dietrich
University of Münster

SUPPORTING ENTERPRISE TRANSFORMATION USING A UNIVERSAL MODEL ANALYSIS APPROACH

Steinhorst, Matthias, University of Münster - ERCIS, Leonardo Campus 3, 48149 Münster, Germany, matthias.steinhorst@ercis.uni-muenster.de

Breuker, Dominic, University of Münster - ERCIS, Leonardo Campus 3, 48149 Münster, Germany, dominic.breuker@ercis.uni-muenster.de

Delfmann, Patrick, University of Münster - ERCIS, Leonardo Campus 3, 48149 Münster, Germany, patrick.delfmann@ercis.uni-muenster.de

Dietrich, Hanns-Alexander, University of Münster - ERCIS, Leonardo Campus 3, 48149 Münster, Germany, hanns-alexander.dietrich@ercis.uni-muenster.de

Abstract

Enterprise Architecture Management has been proposed to help organizations in their efforts to flexibly adapt to rapidly changing market environments. Enterprise architectures are described by means of conceptual models depicting, e.g., an enterprise's business processes, its organisational structure, or the data the enterprise needs to manage. Such models are stored in large repositories. Using these repositories to support enterprise transformation processes often requires detecting structural patterns containing particular labels within the model graphs. As an example, consider the case of mergers and acquisitions. Respective patterns could represent specific model fragments that occur frequently within the process models of the merging companies. This paper introduces an approach to analyse conceptual models at a structural and semantic level. In terms of structure, the approach is able to detect patterns within the model graphs. In terms of semantics, the approach is able to detect previously standardized model labels. Its core contribution to enterprise architecture management and transformation is two-fold. First, it is able to analyse conceptual models created in arbitrary modelling languages. Second, it supports a wide variety of pattern-based analysis tasks related to managing change in organisations. The approach is applied in a merger and acquisition scenario to demonstrate its applicability.

Keywords: Enterprise Architecture, Conceptual Model Analysis, Pattern Matching, Semantic Standardization.

1 Introduction

Contemporary enterprises are exposed to continuous change. To stay competitive in global markets, they constantly adopt new technologies (Karahanna, Straub, and Chervany, 1999), outsource parts of their business to other organizations (Ackermann et al., 2011), merge with or acquire former competitors (Napier, 1989), align their business processes with the IT systems supporting them (Lankhorst, Proper, and Jonkers, 2010), or integrate heterogeneous portfolios of application systems (Miklitz and Buxmann, 2007). Against the backdrop of constant transformations, the use of enterprise architectures has been put forth to guide "the development of the enterprise as a whole and the development of their IT portfolio in particular" (Op't Land and Proper, 2007, p. 1956). Enterprise architectures can be interpreted as "a coherent whole of principles, methods, and models that are used in the design and realisation of an enterprise's organisational structure, business processes, information systems, and infrastructure" (Lankhorst, 2005, p. 3). It provides an as-is description of an enterprise and can be used to analyse and subsequently change its structure to represent a desired to-be state (Winter and Aier, 2011; Buckl et al., 2011). Enterprise architectures are described by repositories of graph-based conceptual models depicting an enterprise's business processes, its portfolio of application systems, or the data the company needs to manage (Op't Land et al., 2009, p. 37).

To support enterprise transformation processes, these model repositories need to be analysed with respect to both model structure and model semantics. *Structurally analysing* conceptual models means identifying patterns within the models. In case of mergers and acquisitions for instance, such patterns may represent sets of activities that frequently occur in the process models of the merging companies (Yan, Dijkman, and Grefen, 2010). Identifying these activity patterns helps to develop a unified process model. Other applications of pattern detection in the context of enterprise transformation include identifying process weaknesses (Smirnov et al., 2009), managing process compliance (Weidlich et al., 2010), or complexity (Weber et al., 2011). *Semantically analysing* conceptual models means detecting particular labels used to further describe the model elements. As different analysts might interpret particular terms differently, standardizing labels is a prerequisite for such an analysis. Such labels must contain only terms and phrase structures that are understood in the same way by all model users.

This paper addresses the problem of analysing conceptual models both on a structural and semantic level. It introduces an analysis approach that incorporates mechanisms to detect structural patterns and previously standardized labels in conceptual models. The approach's core contribution to enterprise architecture management and transformation is the fact that it is universal in two manners. First, it supports a wide variety of pattern-based analysis tasks related to the management of change in organisations (see examples above). Secondly, as enterprise architectures may contain models created with different modelling languages, the approach is applicable to any graph-based conceptual models. We follow the argument of Iacob et al. (2005, p. 200) in postulating that analysis methods have to be applicable across multiple domains. The paper, however, focuses on applying our approach in the context of mergers and acquisitions using EPCs and ERMs.

The remainder of the paper is organized as follows. In Section 2, we discuss related work on conceptual model analysis. In Section 3, we introduce our universal model analysis approach. Section 4 exemplarily applies the approach in the context of mergers and acquisitions. We conclude the paper in Section 5 by summarizing our main contributions, elaborating on limitations of our approach and providing an outlook to future research.

2 Related Work

This paper introduces an approach to analyse conceptual models both on a structural as well as on a semantic level. In terms of *structural analysis*, much work has been presented in the domain of Business Process Management (BPM). Here, structurally analysing process models refers to

examining the control flow of various business activities. This, for instance, serves to identify typical weaknesses or best practices that are represented as activity patterns within the models. Approaches to detect such patterns have been proposed by Smirnov et al. (2009) or van Dongen, Mendling, and van der Aalst (2006). In the context of enterprise transformation, detecting weakness/best practice patterns helps to improve an enterprise's business processes. It may also lead to a company outsourcing those parts of its business that it cannot manage efficiently. Other work addresses the growing complexity of process models by presenting approaches to identify frequently occurring activity patterns (Weber et al., 2011; Reijers, Mendling, and Dijkman, 2011). Such patterns can be represented by aggregated activities describing the process on a higher level of abstraction. This is particularly helpful to understand a company's core business activities that need to be represented in an enterprise architecture. Further work is concerned with identifying similar process models. To that end, Yan, Dijkman, and Grefen (2010) introduce an approach based on the idea that similar processes contain a large number of equal activity patterns. Possible application scenarios of this work include mergers and acquisitions. Here, business processes of two formerly separate companies need to be compared to one another to define consolidated processes of the integrated enterprise.

A structural analysis only takes into account model elements and the arrows connecting them. To allow for a meaningful analysis, the *content of model elements* has to be considered as well. In the domain of BPM, the usage of ontologies has been suggested to capture the corporate vocabulary and define semantic relationships between particular terms (Thomas and Fellmann, 2009). Such ontologies define an unambiguous, formal conceptualization of the domain knowledge which is applied to describe a given business process (Hua, Zhao, and Storey, 2010). Other approaches are concerned with identifying and subsequently standardizing particular labelling styles of process activities (Leopold, Smirnov, and Mendling, 2010). In the domain of database engineering, the use of ontologies has been put forth to match semantically similar parts of various database schemas to one another (Cruz, Antonelli, and Stroe, 2009 or Tavages et al., 2009).

In this paper, we present a model analysis approach that considers both the structure as well as the semantics of conceptual models. We introduce an approach that incorporates a mechanism to detect arbitrary structural patterns in any kind of conceptual model. Thus it is applicable for a wide variety of pattern-based analysis problems (see examples above). In terms of model semantics, our analysis approach is able to standardize model labels that can be searched for in a subsequent analysis.

3 A Universal Model Analysis Approach

Section 3 introduces our model analysis approach. It incorporates mechanisms to detect semantically standardized model element labels (Section 3.1) and structural patterns (Section 3.2). After presenting these underlying concepts a conceptual specification of the combined analysis approach (Section 3.3) as well as details on its implementation (Section 3.4) is given.

3.1 Semantic Model Analysis

To run meaningful analyses on conceptual models, it is imperative to standardize their labelling. Studies show that conceptual models differ significantly in terms of utilized phrase structures and vocabulary (Hadar and Soffer, 2006). This causes difficulties in determining the semantics of particular model elements. For instance, an activity label "Check invoice" of a given process model may or may not be semantically identical to an activity label "Bill verification". For this reason, our model analysis approach includes a mechanism to semantically standardize labels of model elements. Prior to running an analysis, this mechanism is executed on the entire model repository to be analysed.

The mechanism rests on two pillars: a corporate vocabulary and phrase structure conventions. The *corporate vocabulary* defines all terms that are allowed to be used for labelling. It represents a subset of the natural language employed in its model repository. The corporate vocabulary can be developed from scratch by domain experts or by reusing existing glossaries or thesauri. It contains only nouns,

verbs, adjectives, and adverbs, as all other word classes are independent from a particular enterprise. Other than an enumeration of valid terms, the vocabulary defines relationships between them. It specifies what words are synonyms or homonyms of what other words. For each synonym or homonym relationship the vocabulary also defines a dominant term that is to be used instead of its synonyms/homonyms. *Phrase structure conventions* define the grammatical structure of element labels. For each element type of each modelling language at least one phrase structure convention has to be defined. An activity in a BPMN model can, for instance, be labelled according to the phrase structure (<verb, imperative>, <noun singular>). Such a structure allows for labels like "Check invoice" or "Execute goods receipt". Phrase structures conventions consequently provide a template for element labels and can be arranged in any grammatically sensible order and inflexion.

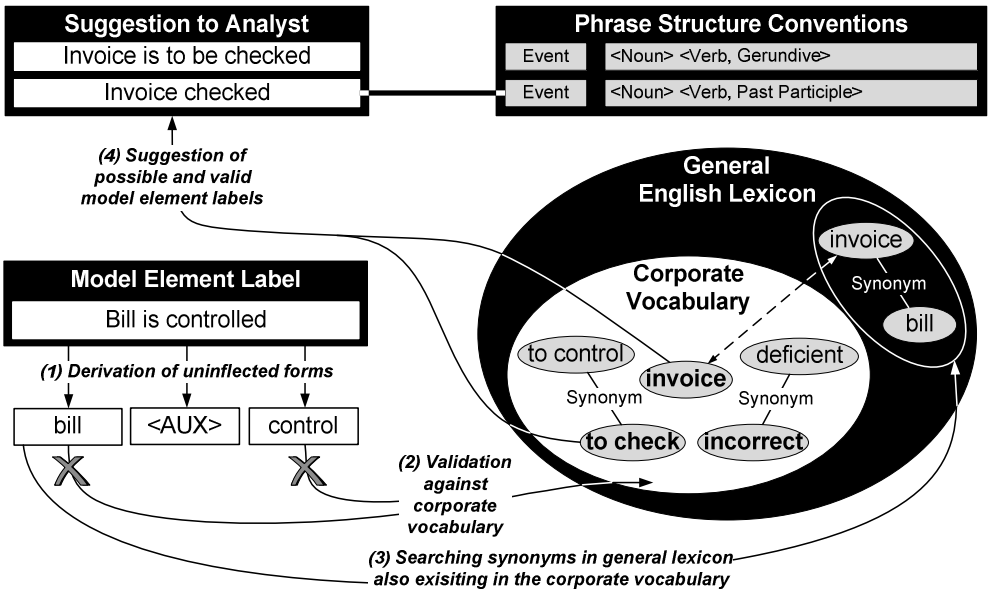

Figure 1: Process of Label Standardization using the example of EPC-Events (adapted from Delfmann et al., 2011)

Having defined the corporate vocabulary as well as phrase structure conventions, element labels can be semantically standardized. Figure 1 illustrates the respective standardization procedure using the example of an event-driven process chain (EPC). An event can be labelled according to two phrase structure conventions (as depicted in the upper right corner of the figure). Given a particular label, the algorithm first calls a linguistic parser that determines the uninflected forms of all the terms contained in that label (1). In a second step, the uninflected terms are validated against the corporate vocabulary (2). In the example depicted in Figure 1, the verb "to control" is indeed contained in the corporate vocabulary but is dominated by its synonym "to check" (dominant synonym is highlighted in bold). The noun "bill" on the other hand is not contained in the corporate vocabulary. If that is so, the algorithm consults a general lexicon of the English language (3). In this case, it identifies the term "invoice", which is contained in the vocabulary, as a synonym of "bill". Having determined appropriate terms, the algorithm now constructs phrase structures according to the predefined conventions (4). As two conventions are allowed for events, two phrase structures are constructed: "Invoice is to be checked" and "Invoice checked". As the algorithm cannot decide which structure is appropriate in the given context, a manual input of an analyst is required, who chooses among the returned suggestions. Manual input is furthermore required if the standardization algorithm fails at any step during its execution. This can happen, if (a) a synonym for a particular term cannot be determined, because the term is unknown to the general lexicon, (b) no synonyms can be identified for an invalid term, or (c) synonyms not contained in the corporate vocabulary are found. In all of these cases, the analyst has to decide whether the respective term is added to the vocabulary.

This mechanism to standardize model labels is based on the assumption that both the corporate vocabulary and phrase structure conventions are a priori artefacts. Constructing an enterprise-wide vocabulary poses a significant challenge. Different people may understand particular terms differently. In order for the presented standardization mechanism to work, a group of domain experts has to define all business objects, activities, IT systems, organizational units, etc. that are to be represented within the model repository of an enterprise. Based on these definitions, appropriate terms best describing these artefacts can be determined. In doing so, it is assured that a process activity labelled "Produce X" actually produces the business object X. A more detailed discussion on the challenges of defining and maintaining a corporate vocabulary can be found in Moriarty (2001). In terms of phrase structure conventions, the mechanism assumes that conventions are defined for each element type of each modelling language used to create conceptual models. If business processes are modelled as EPCs, appropriate phrase structure conventions have to be defined for functions, events, and all other element types contained in the EPC meta-model. If the model repository also contains data models created as ERMs, additional phrase structures have to be defined for entity types, relationship types, etc.

3.2 Structural Model Analysis

The structural analysis of conceptual models requires the identification of patterns within model graphs. This serves a variety of different purposes ranging from model similarity search to identifying improvement potential (Cf. Section 2). Our model analysis approach incorporates a pattern matching mechanism that allows finding arbitrary structural patterns within models of any modelling language. Being based on set-theory, the approach represents any conceptual model in terms of two basic sets. These are the set O of its objects and the set R of its relationships. The set E of elements is defined as the union of O and R. Set-altering functions and operators are defined that perform particular operations on these basic sets. To be able to determine arbitrary patterns, we define four classes of functions. First, we have to be able to recognize elements belonging to a particular element type of the modelling language:

- *ElementsOfType(X,a)* returns all elements of the input set X that belong to a particular type a. The respective elements are put into one output set.

Furthermore, we need to identify all elements that have (a particular number of) ingoing or outgoing relations (of a particular type):

- *ElementsWith{In|Out}Relations(X,Z)* return all elements of X and their {ingoing | outgoing} relationships defined in Z. These functions each return a set of sets. Each inner set contains an element of X and all its relationships of Z.
- *ElementsWith{In|Out}RelationsOfType(X,Z,c)* return all elements of X and their {ingoing | outgoing} relationships of Z that are of type c. Again, these functions return a set of sets with each inner set containing one element of X as well as its {ingoing | outgoing} relationships of Z that belong to type c.
- *ElementsWithNumberOf{In|Out}Relations(X,Z,n)* return all elements of X that have a predefined number n of relationships of Z. All occurrences of that pattern are represented in one inner set of the returned set of sets.
- *ElementsWithNumberOf{In|Out}RelationsOfType(X,Z,n,c)* are a combination of the two latter groups of functions. They return elements having a predefined number n of relationships of Z that are of type c. These functions return a set of sets.

In addition, we want to be able to find particular elements, their immediate neighbours, and the relationship between them:

- *ElementsDirectlyRelatedInclRelations(X_1,X_2)* and *DirectSuccessorsInclRelations(X_1,X_2)* return all elements of X_1, their neighbouring elements of X_2, as well as the relationships between the respective elements. ElementsDirectlyRelatedInclRelations(X_1,X_2) only works on undirected graphs whereas DirectSuccessorsInclRelations(X_1,X_2) only works on directed graphs.

Lastly, to be able to find structures representing element paths of arbitrary length, we included the following functions in the pattern matching mechanism:

- *{Directed}Paths(X_1,X_n)* return all {directed} paths between all elements of X_1 and all elements of X_n. One inner set of the resulting set of sets contains one path from one element of X_1 to one element of X_n.
- *{Directed}Paths{Not}ContainingElements(X_1,X_n,X_c)* return all paths from all elements of X_1 to all elements of X_n that either contain at least one or no element of X_c.

For all paths-functions the pattern matching approach includes versions that determine only the shortest or longest paths as well as loops. As its theoretical basis is set theory, the approach furthermore incorporates the set operators *Union*, *Intersect*, and *Complement* that perform the standard set operations on two sets of elements. A *Join*-operator unifies two input sets if they have at least one element in common. Analogously to the *Intersect*- and *Complement*-Operators, the approach offers versions working on sets of sets (*InnerIntersect* and *InnerComplement*). The *SelfUnion* operator turns a set of sets into a single set, whereas the *SelfIntersect* operator performs an intersection on all inner sets resulting in one single set that holds all elements contained in every inner set. These set-altering functions and operators allow for building up arbitrary pattern definitions recursively. Result sets of one particular function/operator call serve as input for another function/operator. Such pattern definitions can be run on a repository of conceptual models to identify all pattern occurrences within the models of the repository. A detailed formal specification of all functions and operators can be found in Delfmann et al. (2010).

3.3 Conceptual Specification of the Combined Analysis Approach

In the following, the two approaches to semantically standardize element labels as well as to detect structural patterns are combined into an integrated model analysis approach. Its overall procedure is subdivided into two main steps, which are depicted in Figure 2 (black-shaded elements are derived from the pattern matching approach, grey-shaded elements are derived from the semantic standardization approach, and non-shaded elements are new). Provided that, first, the model base to be analysed is semantically standardized as described in Section 3.1 and, second, structural patterns are defined as explained in Section 3.2, an analysis can be specified (Cf. Analysis Definition in Figure 2). That analysis is then applied to a set of models in order to generate a report as a result (Cf. Report Generation in Figure 2).

An *analysis* is composed of one or more *sub-analyses*. The *scope* of a (sub-) analysis defines whether it is run on one set (simple analysis) or two sets of models (comparative analysis). The model set contains all models that are to be analysed. The set can hold an entire model repository or just parts of it (e.g. all process models). In case of a comparative analysis, the results obtained from both model sets are compared to one another. In case they are equivalent in terms of the comparison type, they are returned as a result pair marked as “equivalent”. Consider the example of two data structures contained in one pattern equivalence class. If one of these structures is found in a model of the first set, while the other pattern is found in a model of the second set, both pattern occurrences are returned and marked as equivalent.

Each sub-analysis is assigned a *search criterion* describing properties of the expected analysis results (e.g., “find all receipt structures containing the term invoice”). A search criterion can be *atomic* or *composed*. An atomic search criterion can either be a single *structural pattern* that is to be searched in the entire model set, a set of such patterns that are considered equivalent (*pattern equivalence class*), or an *element type* of a particular modelling language. In these cases, all occurrences of the respective structures within the model set(s) are returned as analysis results. The pattern equivalence class allows for finding two or more structural patterns in one analysis run. As far as semantic search criteria are concerned, occurrences of predefined *phrase structures*, *word classes*, or *words* within particular model elements can be searched for.

A composed search criterion consists of a combination of search criteria connected by logical operators. The search criteria to be combined are specified by the *criterion structure*. Two different search criteria can be combined using the *operators* AND, OR, XOR, and NOT. In doing so,

arbitrarily complex search criteria can be constructed, as a sub-criterion can itself be further composed. For instance, we are able to define an analysis that identifies all occurrences of a particular pattern in combination with all occurrences of a given phrase structure while excluding all occurrences of a specific word. A *criterion restriction* is used to refine a search criterion. Here, a search criterion – either an atomic or a complex one – serves as a constraint for an atomic search criterion restricting the resulting set of model fragments. Therefore, the restricting search criterion is directly assigned to the atomic search criterion to be restricted. This allows for specifying an analysis that, for instance, returns all pattern occurrences containing a particular phrase structure. In this example, the search criterion containing the pattern definition is further constrained, so that only those pattern occurrences are returned that also contain the predefined phrase structure.

The attribute *output type* defines the granularity of the analysis results to be displayed. Consider the example of an analysis to search for structural pattern occurrences. By defining patterns as output type, the complete pattern occurrences are included in the report. By setting the value to "element", the report is straightened to visualize occurrences of single model elements contained in the returned pattern occurrences. By defining "phrase syntax" as output type only phrase syntax occurrences contained in the returned pattern occurrences are visualized, and so on. To avoid empty reports, the output type has to be defined with respect to the specified search criteria.

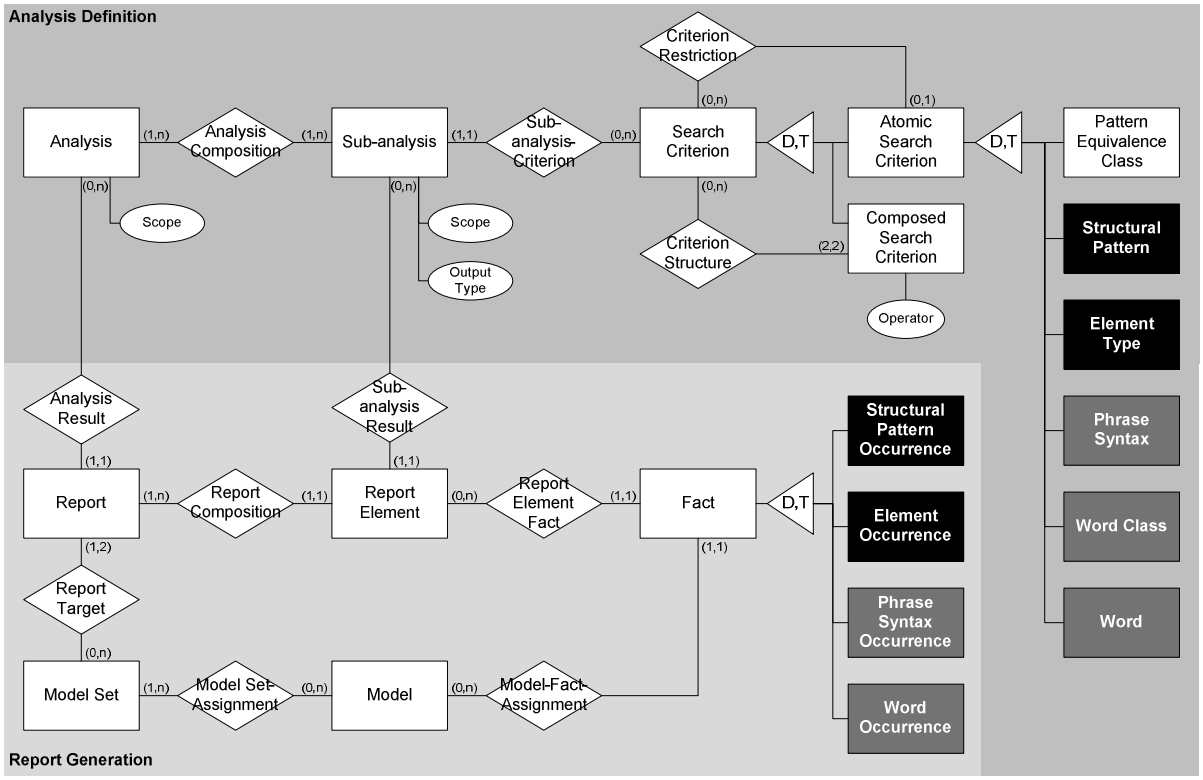

Figure 2: Conceptual Specification of the Combined Analysis Approach (adapted from: Herwig et al., 2011)

The results of an analysis are visualized in a *report*. Depending on the scope of an analysis, a report is targeted at either one *model set* for a simple analysis or two model sets for a comparative analysis. Each model set consists of one or more *models* to be analysed. A report is composed of one or more *report elements*, each of them resulting from one sub-analysis. A report element represents a single row in a report and shows the particular *facts* returned as search results from the corresponding sub-analysis. A fact defines a particular match of the sub-analysis' search criterion, which is depicted in the report. With respect to the possible output types of a sub-analysis, a fact can be a *structural pattern occurrence*, an *element occurrence*, a *phrase syntax occurrence*, or a *word occurrence*.

3.4 Implementation of the Combined Analysis Approach

In Figure 3, implementation details of the combined analysis approach are given in pseudo-code. The method *RunAnalysis* in the upper part of the figure takes two model sets and an analysis specification as input, while returning a report as output. For each sub-analysis contained in the analysis, the scope is determined (lines 4 and 8). In case it is set to *simple*, a search run is executed for each model contained in the first model set returning a list of facts (line 6) that is added to the report (line 7). For a simple analysis the input parameter representing the second model set is set to null. In case the scope is set to comparative, search runs are executed for all models contained in both model sets (lines 9 to 14). Lastly, the identified facts are added to the final report (line 15) returned to the analyst (line 16).

Analysis Execution

```
01: Report RunAnalysis(List<Model> firstModelSet, List<Model> secondModelSet, Analysis analysis){
02:    Report report = new Report();
03:    foreach (subanalysis in analysis.Subanalyses){
04:       if(subanalysis.Scope == Simple){
05:          foreach(model in firstModelSet){
06:             List<Fact> facts = Execute(subanalysis.SearchCriterion, model, null);
07:             report.Add(new ReportElement(facts);}}
08:       else if(subanalysis.Scope == Comparative){
09:          List<Fact> factsOfFirstModelSet = new List<Fact>();
10:          foreach(model1 in firstModelSet){
11:             factsOfFirstModelSet.addList(Execute(subanalysis.SearchCriterion, model1, null));}
12:          List<Fact> factsOfSecoundModelSet = new List<Fact>();
13:          foreach(model2 in secoundModelSet){
14:             factsOfSecoundModelSet.addList(Execute(subanalysis.SearchCriterion, model2, null));}
15:          report.Add(new ReportElement(factsOfFirstModelSet, factsOfSecoundModelSet));}}
16:    return report;}
```

Search Criterion Execution

```
01: List<Fact> Execute(SearchCriterion sc, Model m, List<Fact> facts){
02:    List<Fact> output;
03:    if(sc is AtomicSearchCriterion){
04:       output = ExecuteAtomic(sc, m, facts);
05:       if(sc.HasRestriction){
06:          output = Execute(sc.Restriction, null, output);}}
07:    else if(sc is ComposedSearchCriterion){
08:          List<Fact> facts1 = Execute(sc.FirstSearchCriterion, m, facts);
09:          List<Fact> facts2 = Execute(sc.SecondSearchCriterion, m, facts);
10:          output = MergeResults(facts1, facts2, sc.Operator);}
11:    return output;}
```

Figure 3: Implementation of the Combined Analysis Approach

To determine all facts contained in a particular model, the *Execute* method is called (lines 6, 11, and 14). This method is explained in more detail in the lower part of Figure 3. It is called with a particular search criterion, a model that is to be searched, as well as a list of intermediate facts. First, the method determines if the given search criterion is atomic (line 3) or composed (line 7). In case of the former the method *ExecuteAtomic* is called which determines all facts in the given model corresponding to an atomic search criterion (line 4). As explained in Section 3.3, this can for instance be an occurrence of a structural pattern or of a predefined phrase structure contained in a model element. In a next step, the *Execute* method determines if the search criterion is restricted (line 5). If that is so, the output list is overwritten with a recursive call to *Execute* (line 6). This call takes the previously determined output as fact parameter, whereas the model parameter is set to null. In doing so, the following call to the *ExecuteAtomic* method is also given this input. *ExecuteAtomic* is built in such a way that if its second parameter is set to null and its third parameter contains a list of preliminary facts, a search is run on this list instead of the model. This allows for refining the list of facts to include only those fact occurrences that correspond to the restriction criterion.

If the search criterion is composed (line 7), two lists of facts are calculated each representing the set of facts identified in the model according to the two sub-criteria. This is again achieved by recursively calling the *Execute* method with the respective input parameters (lines 8 and 9). In each case, the first parameter represents the respective sub-criterion. The resulting output is determined by merging the

two fact lists according to the operator type specified in the composed search criterion (line 10). In case the operator type is set to AND, the *MergeResults* method, for instance, determines the unified set of all fact occurrences found in the model.

4 Application

To demonstrate the applicability of our model analysis approach in the context of enterprise transformation, we prototypically implemented it as a plugin for a meta-modelling tool which was available from a previous research project (Delfmann and Knackstedt, 2007). This plugin contains environments to specify structural patterns, manage the corporate vocabulary as well as phrase structures, and define analyses. We decided to provide application examples in the area of mergers and acquisitions. A major challenge in merging two or more companies is integrating the respective IT landscapes (Miklitz and Buxmann, 2007). For the purpose of this application example, assume that we have two companies. For each of these companies, we modelled the three business processes "campaign execution", "order processing", and "request processing" as EPCs. Furthermore, we created two Entity Relationship Models representing the data models of a CRM application supporting these three business processes. In total, we therefore ran our analysis on two model repositories that each consists of three EPC models and one ERM. A first step toward IT integration in merger scenarios is identifying the different application systems that support a particular business activity in all involved companies (Keller, 2004).

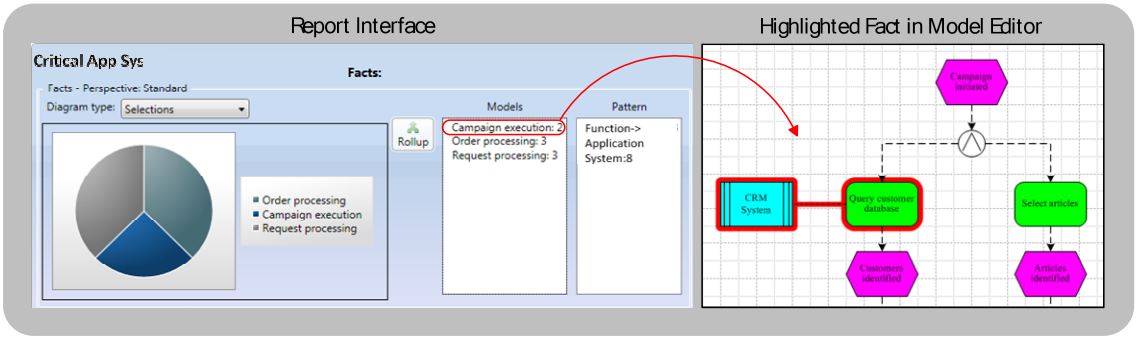

Figure 4. Report for a Simple Analysis generated from first model repository (adapted from Herwig et al., 2011)

To identify such structures, we define a simple analysis with the output type *pattern*. The report will show us all model structures that match the predefined search criteria. For the example of application systems supporting specific business activities we therefore defined a structural pattern that identifies all functions directly related to an application system. According to the matching mechanism outlined in Section 3.2, such a pattern can be defined as follows:

```
ElementsDirectlyRelatedInclRelations(ElementsOfType(O,Function),
ElementsOfType(O,ApplicationSystem))
```

The calls to *ElementsOfType* determine all objects that are functions and application systems respectively. The two resulting sets serve as input for the call to *ElementsDirectlyRelatedInclRelations* returning all functions that are directly connected to an application system. As far as the linguistic features are concerned, we define the words "customer" and "CRM" to be part of the vocabulary for this domain. Consequently, we try to identify activities that involve customers and that are supported by CRM systems. Given this pattern and these terms, we can construct the search criterion that allows for searching occurrences of the pattern. The search is further restricted to include only those pattern occurrences containing either the word "customer" or "CRM" at least once.

Running this analysis on one of the model repositories introduced above delivers the report depicted in Figure 4. The left part of the figure contains the report interface displaying the results of the analysis. Here, the business process "campaign execution" contains two occurrences of the search criterion. The

processes “order processing” and “request processing” each contain three occurrences. By clicking on a particular model, the model editor of the meta-modelling tool opens and all fact occurrences contained in the model are highlighted (right part of Figure 4). In the example, all functions directly connected to an application system containing the terms “customer” or “CRM” are displayed.

This analysis consequently allows for identifying particular application systems supporting business processes of the two companies. It helps business analysts to identify which processes are supported by which application systems. Such an analysis consequently represents a first step toward consolidating and subsequently integrating different IT portfolios. This requires not only integrating business processes but also the underlying databases (Batini, Grega, and Maurion, 2010). One possibility to achieve this is to merge the respective data schemas. Such an integration scenario can also be supported by our analysis approach, as it allows for comparing models in order to reveal similarities. Identifying similarities between data models can be seen as a first step toward integrating them. In the case of our two companies, a comparative analysis could be used to find equivalent structures in the conceptual schemas of the CRM systems identified before.

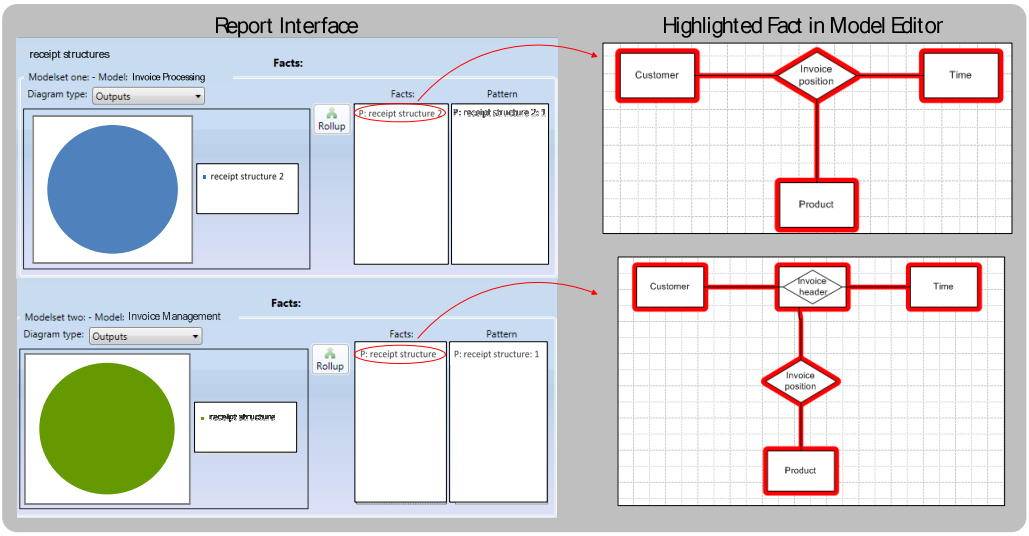

Figure 5. Report for Comparative Analysis generated from both model repositories

We consequently define a pattern equivalence class containing two patterns both describing similar aspects. The first one describes a receipt structure consisting of three entity types, one relational entity type and one relationship type. This structure implies that an invoice consists of a header containing general information about the invoice (order date, name of customer, etc.) and position data describing what products were ordered in what quantities. Such a receipt structure is, e.g., implemented in the SAP ERP system (Hefner and Dittmar, 2001, p. 107). The second pattern describes a similar structure, consisting of three entity types connected by a ternary relationship type. This pattern equivalence class is used as the analysis criterion. Furthermore, we restrict the analysis results to those pattern occurrences that contain the term “invoice”. This time, the analysis was run on both model repositories. As a result, the report contains all structures matching one of the patterns and containing the term “invoice” (Cf. Figure 5). The report separately lists all facts found in both model repositories. In particular, one model represents an invoice structure as a ternary relationship type; the other one utilizes the SAP construct.

5 Contributions, Limitations, and Outlook

In this paper, we presented an approach to analyse conceptual models both on a structural and a semantic level. In terms of structure, the approach incorporates a mechanism to detect patterns in conceptual models created with arbitrary modelling languages. In terms of semantics, the approach

contains a mechanism to detect previously standardized element labels. The core contribution of this paper to the management of enterprise architectures and transformation processes is two-fold. First, the presented analysis approach is universal in the sense that it supports a wide range of analysis tasks involving pattern matching in the context of enterprise transformation (Cf. Section 2 for more details). Second, the presented analysis approach is universal in the sense that it can be applied to conceptual models developed in arbitrary modelling languages.

The analysis approach is based on the assumption that three artefacts are a priori known. In terms of model semantics, the approach assumes that a *corporate vocabulary* and *phrase structure conventions* are defined prior to running an analysis. A group of domain experts has to clearly define all the terms that are part of that enterprise-wide vocabulary. Furthermore, for each element type of each modelling language in use at least one phrase structure convention has to be defined. If process models are created using the EPC notation, phrase structure conventions for functions, events, and all additional element types have to be specified. In terms of model structure, the approach is based on a *predefined set of patterns* that can be searched for in the model repository. Before an analysis can be specified and run, considerable time and effort consequently has to be put into creating these artefacts.

Future research will focus on applying it in other areas related to the management of enterprise transformation to discover further application potential. Long term research will focus on determining acceptance factors of our model analysis approach. Moreover, we intend to combine the corporate vocabulary with semantic technologies like ontologies to allow for the specification of more detailed linguistic relationships.

References

Ackermann, T., Miede, A., Buxmann, P., and Steinmetz, R. (2011). Taxonomy of Technological IT Outsourcing Risks. Support for Risk Identification and Quantification. In Proceedings of the 19th European Conference on Information Systems (ECIS 2011), Helsinki.

Batini, C., Grega, S., and Maurino, A. (2010). Optimal Enterprise Data Architecture using Publish and Subscribe. In Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing, p. 541, Chicago.

Buckl, S., Buschle, M., Johnson, P., Matthes, F., and Schweda, C.M. (2011). A Meta-Language for Enterprise Architecture Analysis. In 16th International Conference on Exploring Modeling Methods for Systems Analysis and Design, London.

Cruz, I. F., Antonelli, F. P., and Stroe, C. (2009). AgreementMaker: Efficient Matching for Large Real-World Schemas and Ontologies. Very Large Data Bases Endowment 2 (2), 1586-1589.

Delfmann, P., Herwig, S., Lis, L., and Becker, J. (2011). Supporting Conceptual Model Analysis Using Semantic Standardization and Structural Pattern Matching. In Semantic Technologies for Business and Information Systems Engineering: Concepts and Applications (Smolnik, S., Teuteberg, F., Thomas, O. Eds.), p. 125. Hershey, Pennsylvania.

Delfmann, P., Herwig, S., Lis, L., Stein, A., Tent, K., and Becker, J. (2010). Pattern Specification and Matching in Conceptual Models. A Generic Approach Based on Set Operations. Enterprise Modelling and Information Systems Architectures, 5 (3), 24-43.

Delfmann, P. and Knackstedt, R. (2007). Towards Tool Support for Information Model Variant Management. A Design Science Research Approach. In Proceedings of the 15th European Conference on Information Systems (ECIS 2007), p. 2098, St. Gallen, Switerland.

Hadar, I. and Soffer, P. (2006). Variations in Conceptual Modeling: Classification and Ontological Analysis. Journal of the Association for Information Systems 7 (8), 568-592.

Hefner, S. and Dittmar, M. (2001). SAP R/3 Finanzwesen. Addison-Wesley, Munich.

Herwig, L., Lis, L., Steinhorst, M., Becker, J., and Delfmann, P. (2011). A Generic Multi-purpose Conceptual Model Analysis Approach. Conceptual Specification and Application. In Proceedings of the 4th International Workshop on Enterprise Modelling and Information Systems Architectures, p. 201, Hamburg, Germany.

Hua Z., Zhao J. L., and Storey, V. C. (2010). Exploring a Domain Ontology Based Approach to Business Process Design. In Proceedings of the 31st International Conference on Information Systems (ICIS 2010), Saint Louis.

Iacob, M.-E., Jonkers, H., van der Torre, L., and de Boer, F.S., Bonsangue, M., Stam, A.W. (2005). Architecture Analysis. In Enterprise Architecture at Work. Modelling, Communication, and Analysis (Lankhorst, M. Ed.), p. 191. Springer, Berlin.

Karahanna, E., Straub, D.W., and Chervany, N.L. (1999). Information technology adoption across time: a cross-sectional comparison of pre-adoption and post-adoption beliefs. MIS Quarterly 23 (2), 183-213.

Keller, W. (2004). Managing Application Portfolios in Merger Situations. In Beiträge der 34. Jahrestagung der Gesellschaft für Informatik e. V. (GI).

Lankhorst, M.M. , Proper, H.A. and Jonkers, H. (2010). The Anatomy of the ArchiMate Language. International Journal of Information System Modeling and Design, 1(1), 1-32.

Lankhorst, M. (2005). Introduction to Enterprise Architecture. In Enterprise Architecture as Work. Modelling, Communication, and Analysis (Lankhorst, M. Ed.), p. 1. Springer, Berlin.

Leopold, H., Smirnov, S., and Mendling, J. (2010). Refactoring of Activity Labels in Business Process Models. In Proceedings of the 15th International Conference on Applications of Natural Language to Information Systems, p. 268, Cardiff.

Miklitz, T. and Buxmann, P. (2007). IT Standardization and Integration in Mergers and Acqusitions: A Decision Model for the Selection of Application Systems. In Proceedings of the 15th European Conference on Information Systems (ECIS 2007), p. 1041, St. Gallen, Switzerland.

Moriarty, T (2001). The Importance of Names. The Data Administrator Newsletter 15.

Napier, N. K. (1983). Mergers and Acquisitions, Human Resource Issues and Outcomes. A review and suggested Typology. In Journal of Management Studies 26 (3), 271-289.

Op't Land, M., Proper, E., Waage, M., Cloo, J., and Steghuis, C. (2009). Enterprise Architecture. Creating Value by Informed Governance. Springer, Berlin.

Op't Land, M and Proper, E. (2007). Impact of Principles on Enterprise Engineering. In Proceedings of the 15th European Conference on Information Systems (ECIS 2007), p. 1965, St. Gallen.

Reijers, H. A., Mendling, J., and Dijkman, R. (2011): Human and automatic modularizations of process models to enhance their comprehension. Information Systems Journal 36 (5), 881-897.

Smirnov, S., Weidlich, M., Mendling, J., and Weske, M. (2009). Action Patterns in Business Process Models. In Service-Oriented Computing (Baresi, L., Chi, C. H., Suzuki, J. Eds.), p. 115. Springer, Berlin.

Tavages, D. B., de Paiva, O. A., Braga, J. L., and Filho, J. L. (2009). Analysis Procedure for Validation of Domain Class Diagrams Based on Ontological Analysis. Proceedings of the 28th International Conference on Conceptual Modeling (ER 2009), p. 159, Gramado.

Thomas, O. and Fellmann, M. (2009). Semantic Process Modeling – Design and Implementation of an Ontology-based Representation of Business Processes. Business and Information Systems Engineering 1 (6), 438-451.

van Dongen, B. F., Mendling, J., and van der Aalst, W. M. P. (2006). Structural Patterns for Soundness of Business Process Models. In Proceedings of the 10th IEEE International Enterprise Distributed Object Computing Conference , p.116, Hong Kong.

Weber, B., Reichert, M., Mendling, J., and Reijers, H. A. (2011). Refactoring large process model repositories. Computers in Industry 62 (5), 467-486.

Weidlich, M., Polyvyanyy, A., Desai, N., and Mendling, J. (2010). Process Compliance Measurement based on Behavioural Profiles. In Proceedings of the 22nd International Conference on Advanced Information Systems Engineering (CAiSE 2010), p. 499, Hammamet.

Winter, R. and Aier, S. (2011). How are Enterprise Architecture Design Principles Used? In Proceedings of the 15th IEEE International EDOC Conference Workshops, p. 314, Helsinki.

Yan, Z., Dijkman, R., and Grefen, P. (2010). Fast business process similarity search with feature-based similarity estimation. In Proceedings of the 10th Conference On the Move to Meaningful Internet Systems, p. 60, Crete.