Automated Model Grouping

Michael Becker
University of Leipzig
Applied Telematics / e-Business Group
Klostergrasse 3, 04109 Leipzig, Germany
michael.becker@uni-leipzig.de

Volker Gruhn*
University Duisburg-Essen
Software Engineering
Schützenbahn 70, 45127 Essen, Germany
volker.gruhn@uni-due.de

ABSTRACT
A tremendous amount of software models has been created so far. This growing number of models adds to the fact that it gets more and more difficult to organise, structure, and reuse them. Thereby new software development projects can not profit from existing knowledge. In our research we will study existing natural language processing techniques for their adaptability in the reuse of software models. We will research methods to group existing models according to their functionality.

Categories and Subject Descriptors
D.2.9 [**Software Engineering**]:

General Terms
Algorithms, Design, Experimentation, Management

Keywords
Model reusing, grouping, structuring

1. PROBLEM DESCRIPTION
Using software models it is possible to discuss the structure and behaviour of software on a more abstract level than using code. There are different types of notations that can be used in a wide area of applications. For example process models like Event Driven Process Chains (EPC) [12] or the Business Process Modeling Notation (BPMN) [8] are used to model the temporal flow of processes on a high level of abstraction. UML activity diagrams [17] or Petri Nets [18] are better suited for a more detailed description of software behaviour. In addition there are plenty of other concepts to model software.

Nowadays models gain more importance because they are used in Model Driven Engineering (MDE). The fundamental idea behind MDE is to write as few code as possible and instead describe software using models. Thereby the complexity of the software development process should be lowered. One result of the adaption of MDE is the existence of a large number of models.

This growing number of models adds to the fact that it becomes increasingly difficult to organise models in a reasonable way and to find existing models compatible with requirements. One the one hand it is more complex to group models on the other hand it is also more complex to search for models in a large model collection. This may lead to the problem that existing models and thereby existing knowledge can not be reused and instead the same problem is modelled several times. Developers will not reuse existing models if it is more time-consuming to find, understand, and integrate existing models. Therefore they need active support of their modelling tool for reusing models.

Today's modelling tools in fact provide very good modelling support. But normally they tend not to provide support for reusing existing models. Often reusing is done by copy and paste of existing model parts. This makes knowledge about existing models implicitly. Developers have to know if models exist and this knowledge is gone when developers leave an organisation. It would be better to make this knowledge explicitly so new developers can profit from it, too.

2. MOTIVATION AND OBJECTIVES
This work's basic idea is to develop methods and techniques to support the better reuse of existing software models. This should result in an acceleration and simplification of model-driven processes. During the research we will study different ways to structure models to navigate between related models as well as to find models during development time.

To reuse existing models they must be structured and documented. Structuring results explication of the implicitly knowledge about models. Documenting models results in a better understandability. So one option to find models in a large model base is to manually keyword and describe models, and eventually to group them into different categories. Using Information Retrieval methods it is possible to find models on the basis of their description or keywords.

Because this manual indexing is very labour-intensive and time-consuming good results are only expected when it is applied on a small sized collection of models. The more models are produced the more improper the descriptions

*Dissertation supervisor

will become. Additionally when there are too many models manual indexing is not possible anymore at reasonable cost. This leads to a situation where there are many models that con not be indexed and therefore can not be found whereby existing knowledge is not used. Especially organisations that use model-driven techniques for a longer time have a large collection of unstructured models that can be reused in parts or completely. They need a feasible possibility to structure these artifacts so they can be reused eventually.

The objective of our work is to automatically structure an existing collection of models by grouping models that describe similar facts based on specific characteristics of these models. We expect that the structuring of models contributes to a better model browsing - navigating through a collection of models based on unspecified requirements so developers can get inspiration from existing models. In addition we want to support similarity searching of models to improve model retrieval.

Generally these two objectives contribute to simplify the development of new software systems. Developers should be notified when they model situations similar to existing ones in a collection of models. Our methodology will be notation independent, e.g. while developing UML activity diagrams developers should be notified about similar EPCs, too. Based on these recommended developers of models can determine problems and challenges that have been overcome already. This will result in a reduced software development process complexity and increase the overall process productivity.

3. APPROACH

Our planned approach is structured in four distinct steps which we will specify in the following sections. The first step is to identify feasible characteristics as a basis for grouping and calculating models. Having these characteristics in mind existing classification methods will be studied with the focus on adaptability on software models. The used methods will be evaluated and compared with each other. To allow a practical use of the proposed methodologies we will show how reuse of models based on similarity can be integrated in software development processes.

3.1 Identification of model characteristics

To group models in a feasible way it is required to know what the main characteristics are that distinct one model from another. So as a first step we will identify and extract these characteristics. We want to achieve an automatic grouping so we will study how to automatically extract or deduce these characteristics.

Based on the specification of the modelling notation different characteristics have to be considered. In an UML model the colour of elements is no specific feature. But this is no general rule; e.g. one can think of a feature model in software product lines where red elements are mandatory and green elements are optional. Having this in mind the characteristics depend on the definition of notational elements of a modelling language.

In this first draft we will concentrate on the labels of notation elements and study different characteristics in the course of our work. We will next present the three features text, structure and semantics as well as possibilities to extract them. These features are all based on the labels of elements and are generic enough to be used on a wide variety of modelling notations.

3.1.1 Text

The first and most obvious characteristic of models is their text. With text in the area of modelling we mean e.g. element labels or notes. The simplest methodology is to use the persistent representation of models and group them according to the complete text of this representation. Notations like UML provide a standardised way to save models in the so called XMI format [16]. The text will be used as the basis for grouping models. Using this approach we expect a distinction of different types of notations. For example UML activity diagrams will be grouped in another class than EPCs.

This leads to a first structure in unstructured collection of models. But as stated above our objective is to structure models based on their meaning and not on their type. Differences in the meaning of models will be undiscovered due to the amount of control characters in the persistent storage representation. Therefore we will only extract representative parts, e.g. the label of elements. By knowing the layout of the persistent storage (usually an XML dialect) it is possible to extract the necessary elements. Using these labels it is possible to create thematically aligned groups.

Two common problems when using text to group different types of documents are synonyms - one concept that can be expressed by different words - and homonyms - one word that can mean different concepts. Due to synonyms models that describe the same concept may not be found because of different labels. In contrast, homonyms may lead to the grouping of models with different concepts because of equal labels. These obstacles may be solved by using an ontology of words like WordNet [15]; an approach that is used in literature, too.

3.1.2 Structure

The proposed text-based approach is a first possibility to group models according to the concepts they describe. But it is very generic and may lead to wrong results when applied to different modelling notations. This is due to the fact that a label in an UML activity diagram may have another meaning than the same label in the function of an EPC. So as a second step we will take the structure of notations into account. Most if not all of today's notations have a formal syntax that can be used to identify different structural elements of the notations.

When comparing only models of one notation it is relatively easy to include syntactic relations in the comparison. To calculate a similarity it is only necessary to compare labels of one type with each other. Using UML diagrams one would compare class labels with each others whereas in an EPC collection labels of functions may be compared. But due to the fact that our objective is to create a notation independent methodology structural elements of different notations have to be compared with each other.

The amount of notations that have to be compared adds to the complexity of the problem. Comparing two notations requires the creation of relations between the different elements of this notation, e.g. UML activities in an activity diagram may be mapped to functions in EPCs. Every new notation introduces new mappings. Therefore it is feasible to use an intermediate language representing different aspects of a modelling notation. In the area of business logic the AMABULO approach may be an alternative. AMABULO introduces different mappings of business logic modelling notations [3]. This can be used as a starting point for different mappings.

After creating the mappings of the notations into the intermediate language the similarity of models can be calculated by the similarity of their respective intermediate representation. This allows a more accurate grouping because the structure of the models is considered, too.

3.1.3 Semantics

A more advanced approach is to group models according to their operational semantics. Because many modelling notations at least provide semiformal semantics this can be used to derive the behaviour of the modelled concepts. Two main questions arise in this context: What is the behaviour of static diagrams, e.g. a class diagram in UML? How can the semantics of different notations be represented so they can be compared? One approach is to use an intermediate language like it was introduced in the previous section. But this is a research area for itself so considering the semantics may be only possible when using homogeneous modelling notations.

3.1.4 Automatic feature deduction

It may eventually be possible to automatically extract different characteristics of notations. Thereby characteristics that are directly related to the notation can be used. They can for example be extracted using machine learning techniques. We expect to introduce new modelling notations into the process of grouping without having to create new extraction rules by this. Thereby our methodology will be generic enough to be used in different application areas.

Concluding the extraction step we want to state that one must think about the indexing and representation of models when a model repository is used. In the simplest form there is no further work necessary and models are saved as they are. Contrariwise, it is possible to use different semantic indexing techniques like e.g. Latent Semantic Indexing [4]. Thereby the repository can be conceptualised and eventually provide better search performance.

3.2 Analysing existing methodologies

After extracting different model characteristics we want to group existing models according to them. Depending on the application area different methods can be used. Among others it is necessary to check if models should be grouped unambiguously, meaning that one model should be allocated to exactly one group. On the other hand a fuzzy grouping where one model can be assigned to more than one group may be appropriate, too. This can be achieved by calculating likelihoods for the assignment of a model to every group.

Grouping is based on the assumption that relevant models are more similar to each other than irrelevant models, proved in [10].

In general we identified the following four steps of grouping that have to be applied:

Formatting models Existing models have to be formatted so they can be processed by the grouping algorithms. According to the used characteristics this step may contain different sub-steps. If the whole text of a model should be used, the complete content of the persistent storage can be handed to the grouping. Otherwise, extraction of characteristics like element labels or element colours may be necessary.

Selecting a similarity measure Before grouping it is required to decide how similarities and differences between models can be calculated. Depending on the type of characteristics different approaches are possible. For example similarity of element labels can be computed using the Levenshtein distance presented in [14]. In addition there are plenty of other measures from different application areas like image comparison or text mining. In the area of model similarity there are some measures already, too. They are presented in section 5.

Calculating a similarity matrix After selecting a similarity measure the models are grouped according to this measure. Therefore a matrix is created containing the computed model similarity.

Calculating groups The resulting groups are calculated on the basis of the similarity matrix. For this purpose, a threshold is chosen and models with a similarity above this threshold are grouped together. In the case of fuzzy grouping this threshold may be a percentage value. In general the threshold is defined experimental.

3.3 Evaluation of methods

To be able to benchmark the quality of methods the results have to be evaluated and compared with each other. To achieve this we will apply different methods on different types of model collections. Among others we will use LDraw models (for Lego CAD programs), 3D models from Google Earth, and the SAP reference model. Further information about our evaluation strategies are given in section 4.

3.4 Software development process integration

After giving a theoretical insight in our grouping approach we want to realise reuse of models based on grouping in real software projects. This is achieved by integrating the search for existing models as a central task in early phases of a software development process. For developers to accept this activity it has to be integrated seamless. Therefore an optimal tool support is required.

As a last step we want do develop a software that manages an existing model collection and helps developers by suggesting them existing models. This is supported by existing

model repositories. Using the software should enable developers on the one hand to search for models on the other hand they should be informed about existing similar models during design time. A possible seamless integration is an auto-suggest option for model like modern IDEs provide for code-based development.

4. EVALUATION STRATEGIES

To show that our proposed grouping approach works efficiently we will evaluate it. Therefore as stated above we will apply the grouping on different types of model collections. As a first step we will only group models according to the same notation. This will give a first insight about general characteristics of our grouping methods. To measure the quality of the thematic grouping existing models will be manually grouped into different classes and this manual grouping will be compared to the automatically received results. Obstacles of this approach are possible subjectivity and ambiguity of manual grouping. Different experts may group models in different categories. We want to overcome these obstacles by using the SAP reference model that contains a large amount of models. Differences in subjective groupings will become statistically insignificant. Results of this first run can be used as a baseline for following runs where different modelling notations will be used.

One approach to measure our quality are cluster quality measurements. They allow to calculate the homogeneity of clusters, where a higher homogeneity relates to a higher similarity of elements in one cluster than of elements in different clusters. Furthermore, it is possible to check the grouping stability by using different similarity measures.

To check our retrieval we can use the known information retrieval measures recall, precision, and F-score as the harmonic mean of recall and precision [20, 13]. In our case recall is the fraction of received relevant models to all relevant models and precision is the fraction of received relevant models to all retrieved models. In typical information retrieval applications both measures should be as high as possible meaning that as many models as possible should be retrieved and as many retrieved models as possible should be relevant.

In our model retrieval scenario we expect recall is not the deciding factor. This can be explained by the effort that is necessary to understand retrieved models. It does not make sense to retrieve dozens of models because developers can not understand all these models in reasonable time. So our objective is to maximise precision rather than having a high recall.

5. RELATED WORK

There is already some work done in the field of similarity calculation between models of one notation type. In the following we will give a brief overview about similarity calculation of UML models and process models. After that we will discuss works that concentrate on automatic model suggestion.

According to Gomes et al. in [9] reuse on design level is more important than reuse on code level. They present an approach for reusing UML models based on Case Based Reasoning (CBR, [1]). Knowledge is saved in so called cases whereas a case describes a specific situation. In their work a case represents a software design consisting of UML class diagrams and is made up of a unique identifier, a package containing all elements of the class diagram, and the name of the file containing the case. WordNet is used as a common ontology to categorise software objects according to synsets. A class diagram is added to a synset based on it's labels. Similarity of different diagrams is calculated by the similarity of their respective synsets.

Most works dealing with similarity of process models have a levelled approach in common. These levels are syntax, semantics, and context. On the syntactic level only element labels are taken into account, on the semantic level the meaning of words is considered, too. This is in general achieved by using WordNet. The context of an element is made up of the elements that are connected (either direct or indirect) to the respective element.

Dijkman et al. present in [5] three similarity for process model measures based on so-called Causal Footprints [6]. Syntactic similarity is calculated by the edit distance between labels. Semantic similarity is calculated by the equivalence of labels with a higher weight for exact matches than for synonyms. As stated above on the contextual level they take the context of elements into account. Similarity between two elements is based on the similarity of their incoming and outgoing elements. They validate their approach by applying it to EPC models but they state that their approach is generic enough to be used on all kinds of process models.

A related approach is used in [11]. Jung et al. calculate the similarity of process models by using two vector representations. The first vector contains the likelihoods that activities are executed in a process and the second one is a distance vector creating implicit activity connections. This second vector is similar to the Causal Footprints. Similarity between two process models is calculated by the cosine measure and grouping of processes is done using a hierarchical agglomerative clustering.

Semantic Business Process Models (SBMPs) are used in [7]. SBMPs are a special kind of business processes based of ontologies known from semantic web. In their work Ehrig et al. map Petri Nets to OWL DL (Web Ontology Language Description Logic) and calculate the similarity by using syntactic, linguistic, and structural similarities. This approach is used in [2] where existing processes that may be used in the current context are suggested during design time based on the similarity measures. The authors assure that only valid elements can be inserted so no deadlocks or synchronisation errors will occur.

Smirnov et al. present the principle of action patterns in [19]. These action patterns are closely related to the semantic content of a model but abstract enough so they can be applied in different domains. They are used to suggest additional process activities during design time. Therefore two types of action patterns are presented. The first type are based on cooccurrences and represent activities that occur together in a statistical significant amount of times. The

second type are behavioural patterns that allow suggestions about missing activities. The patterns are calculated by analysing the activity labels of business processes.

6. EXPECTED CONTRIBUTION

Our proposed research will help in creating new software engineering approaches and in improving the development process. We want to give new research research impulses to ease the development of software and to enable developers to concentrate on the core functionality of new software systems. The methods and tools we will create during the research will be presented to the research community and improved according to the feedback.

By grouping existing models knowledge in organisations will be unfolded. Thereby new software projects can benefit from this searchable knowledge base. Besides this we can vision model markets, that can be used by developers during their daily work. Model providers will have the ability to take advantage of a global marketplace and developers can select models according to their specific problems from a wide variety of offers.

7. REFERENCES

[1] A. Aamodt and E. Plaza. Case-based reasoning: foundational issues, methodological variations, and system approaches. *AI Commun.*, 7(1):39–59, 1994.

[2] S. Betz, S. Klink, A. Koschmider, and A. Oberweis. Automatic user support for business process modeling. In *Proceedings of the Workshop on Semantics for Business Process Management*, pages 1–12, 2006.

[3] T. Brückmann and V. Gruhn. Amabulo - a model architecture for business logic. *Engineering of Computer-Based Systems, IEEE International Conference on the*, 0:445–452, 2008.

[4] S. Deerwester, S. Dumais, G. Furnas, T. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391–407, 1990.

[5] R. Dijkman, M. Dumas, B. van Dongen, R. Käärik, and J. Mendling. Similarity of business process models: metrics and evaluation. Technical report, Eindhoven : Technische Universiteit Eindhoven, 2009, 2009.

[6] B. F. v. Dongen, J. Mendling, and W. M. P. v. d. Aalst. Structural patterns for soundness of business process models. In *EDOC '06: Proceedings of the 10th IEEE International Enterprise Distributed Object Computing Conference*, pages 116–128, Washington, DC, USA, 2006. IEEE Computer Society.

[7] M. Ehrig, A. Koschmider, and A. Oberweis. Measuring similarity between semantic business process models. In *APCCM '07: Proceedings of the fourth Asia-Pacific conference on Comceptual modelling*, pages 71–80, Darlinghurst, Australia, Australia, 2007. Australian Computer Society, Inc.

[8] P. Fettke. Business process modeling notation. *Wirtschaftsinformatik*, 50:504–507, 2008.

[9] P. Gomes, F. C. Pereira, P. Paiva, N. Seco, P. Carreiro, J. L. Ferreira, and C. Bento. Using wordnet for case-based retrieval of uml models. *AI Commun.*, 17(1):13–23, 2004.

[10] K. S. Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28(1):11–21, 1972.

[11] J.-Y. Jung, J. Bae, and L. Liu. Hierarchical clustering of business process models. *International Journal of Innovative Computing, Information and Control*, 5(12):1349–4198, December 2009.

[12] G. Keller, M. Nüttgens, and A. W. Scheer. Semantische Prozeßmodellierung auf der grundlage ereignisgesteuerter prozeßketten (epk). Technical Report 89, Universität des Saarlandes, Germany, Saarbrücken, Germany, January 1992.

[13] B. Larsen and C. Aone. Fast and effective text mining using linear-time document clustering. In *KDD '99: Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 16–22, New York, NY, USA, 1999. ACM.

[14] V. I. Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, 10:707, 1966.

[15] G. A. Miller. Wordnet: a lexical database for english. *Commun. ACM*, 38(11):39–41, 1995.

[16] OMG. Mof 2.0/xmi mapping, version 2.1.1. Technical report, OMG, December 2007.

[17] B. Paech. On the role of activity diagrams in uml - a user task centered development process for uml. In *UML'98: Selected papers from the First International Workshop on The Unified Modeling Language*, pages 267–277, London, UK, 1999. Springer-Verlag.

[18] J. L. Peterson. Petri nets. *ACM Comput. Surv.*, 9(3):223–252, 1977.

[19] S. Smirnov, M. Weidlich, J. Mendling, and M. Weske. Action patterns in business process models. In *ICSOC-ServiceWave '09: Proceedings of the 7th International Joint Conference on Service-Oriented Computing*, pages 115–129, Berlin, Heidelberg, 2009. Springer-Verlag.

[20] Y. Zhao and G. Karypis. Evaluation of hierarchical clustering algorithms for document datasets. In *CIKM '02: Proceedings of the eleventh international conference on Information and knowledge management*, pages 515–524, New York, NY, USA, 2002. ACM.