A Language-Independent Model Query Tool

Patrick Delfmann, Hanns-Alexander Dietrich, Jean-Marie Havel, and Matthias Steinhorst

European Research Center for Information Systems (ERCIS), University of Münster, Leonardo-Campus 3, 48149 Münster, Germany
{delfmann,dietrich,steinhorst}@ercis.de, j_have07@uni-muenster.de

Abstract. This paper introduces a prototype implementing a visual graph-based model query language. Querying models refers to identifying particular fragments in the model that comply with a predefined pattern query. The language takes advantage of the fact that models of any type and modelling language can conceptually be represented as a labeled graph. As a consequence the query language remains flexible and is not restricted to specific model types or languages. The language supports topologically exact as well as similar pattern matching and includes additional constraints and attributes in the matching process. In doing so, the language is applicable to many different analysis tasks. Following the design science approach we develop and demonstrate a prototype of such a language which allows for visually defining a pattern query and visually representing the results of the pattern matching process.

Keywords: Conceptual model analysis, model querying, pattern matching.

1 Introduction and Problem Statement

Querying conceptual models to detect patterns in them (i.e., model fragments complying with predefined structural and semantic properties) is the basis for many analysis tasks, such as business process weakness detection or business process compliance checking. For instance, a business rule prescribing that a credit application has to be checked twice by two different persons in a business process before the credit can be granted relates to a specific process model pattern as follows: a process model check activity has to be followed by another process model check activity over a control flow path. Both have to be related to the same document, and the activities have to be performed by different persons. By matching such a pattern against a business process model, we can determine whether or not the process is compliant. Corresponding examples can be found for several further application scenarios.

Many companies are nowadays maintaining large model collections containing models of different types and languages [1]. These model collections are used as a means of analyzing particular aspects of corporate reality [2]. Due to the variety of different analysis tasks and the complexity of these collections [3], such an analysis is becoming increasingly difficult. The research community has proposed many analysis approaches to that end. However, they are mostly restricted to a certain modelling

M.C. Tremblay et al. (Eds.): DESRIST 2014, LNCS 8463, pp. 453–457, 2014.

language or to a specific analysis scenario, even though any model can be conceptually represented as a labeled graph which could then be queried. Further, they often only provide rather basic means for querying. In particular, only very few approaches implement functionalities such as querying attributes other than the type or the label or further constraints on model paths and loops (cf. Section 2.4). Furthermore, it is mostly not possible to visualize the querying results within the model collection, which is essential for further use of the results. Instead, many query approaches only return the model containing a particular pattern occurrence. Therefore, the paper introduces a tool prototype including a novel query language supporting the described functionalities (cf. Section 2.4). It was developed following the design science paradigm [4].

2 Design of the Artifact

Scenarios. The range of analysis tasks the query language supports is manifold. Querying models is concerned with identifying predefined model fragments (the patterns) with (partly) given structural characteristics and (partly) given contents. Querying models serves different business purposes including compliance checking [5], weakness detection [6], model translation [7], detecting structural or behavioral conflicts [8], and model abstraction [9].

Features. The features the query language provides include pattern identification techniques as well as means to visually draw a pattern query and to display the results. As our querying language is implemented as a plugin for a meta-modelling tool operating on models represented as labeled graphs, it is able to support arbitrary conceptual model types and languages (F1). Further, it is able to include arbitrary attributes of nodes and edges in the matching process (F2). To identify exact matches it provides means to query isomorphic fragments consisting of elements having relations and elements having in- /output relations (F3). To be able to detect similar topological structures, it is able to search for paths and loops (F4). Moreover, these paths and loops may contain a variety of constraints such as the containment or exclusion of defined elements or sub-patterns (F5). Patterns can be combined and can contain subpatterns (F6). Attributes of nodes and edges of the pattern can be expressed as variables and put into relation in form of logic expressions (F7). Patterns can be visually drawn (F8). Finally, the result of the query can be visually displayed within the searched models (F9).

3 Significance to Research and Practice

Significance to Research. In graph theory, finding exact matches is known as the problem of subgraph isomorphism for which several algorithms have been proposed [10]. Subgraph isomorphism algorithms create a mapping in which nodes of the pattern are mapped to counterparts of the model. Isomorphic matching does not allow mapping edges to paths of arbitrary length. However, in the context of conceptual model analysis it is necessary to also identify paths or previously unknown length (cf. F4). This problem is known as subgraph homeomorphism which is computationally very complex. Therefore, we introduce a novel problem called relaxed subgraph

isomorphism. Hereby, nodes of the pattern graph are matched to exactly one equivalent node of the model. However, an edge may also be mapped to a path (cf. F4 and F5). Further, the nodes and edges of a pattern can be annotated with attributes, which are included in the checking process. An edge can be either directed, undirected or of any direction. Thus a model is represented as a labelled mixed multi-graph. Our algorithm is able to solve a pattern matching problem as described above (cf. F3 and F4).

Significance to Practice. Our approach's sophisticated functionalities, its language independency, and its applicability to a variety of analysis scenarios make it interesting for practitioners. Most importantly, it provides a convenient means of drawing a pattern query instead of writing formal expressions which is error prone [11] or unfeasible for non-experts. Finally, the result of the analysis can be visually displayed and browsed.

4 Demonstration

The pattern matching process starts with defining a query (cf. Figure 1). The query is drawn visually (cf. F8). A pattern query consists of nodes and edges, which can be further specified. An element's type can be specified according to the used modelling language (cf. icon at the bottom-right corner of a node). Further, a caption match expression for the label can be specified which is set to a wildcard by default.

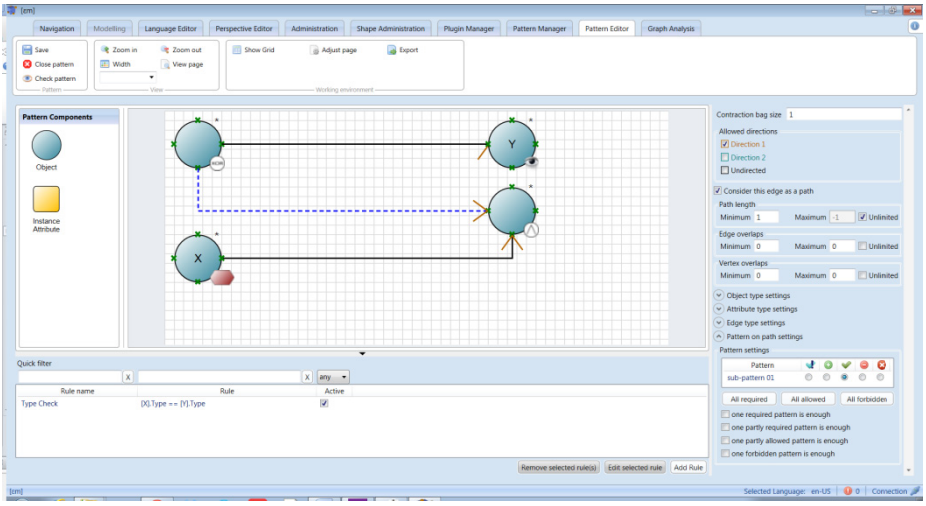

Fig. 1. Pattern editor

The allowed directions of an edge can be set to either be undirected, directed or of any direction. It is also possible to provide a caption match expression as well as to define the type of the edge. Further attributes can be attached to nodes and edges. Such a pattern query consisting of nodes, plain edges, and attributes thus allows for searching for exact pattern matches.

In order to define topologically similar structures, an edge can be specified to be considered as a path (edge-path mapping). The available edge settings then allow defining a min/max path length, edge overlaps and vertex overlaps. A vertex overlap could, for example, represent a loop starting and ending in an XOR-split which may thus be visited multiple times. An edge overlap can represent a loop in which a certain path is possibly passed multiple times. It is important to notice that both notions of overlaps are different. Further, it is possible to define edge and vertex overlaps globally which allows to define whether two different paths are allowed to cross each other at a certain node (global vertex overlap) or within a certain path (global edge overlap).

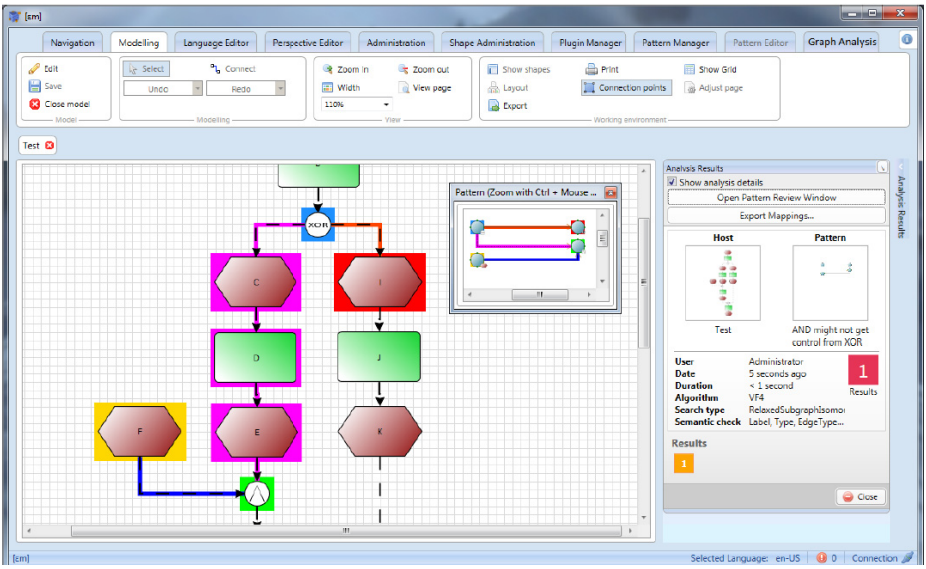

Fig. 2. Pattern match visualization

The user can define the set of elements that must or may not be contained on the path. The same holds for attribute types which can be allowed, required or forbidden. Individual edge types can also be allowed, required or forbidden on the path. Patterns can be combined by specifying that another pattern must be or is not allowed to be contained on a path – either completely or in parts (cf. F6). As for the other settings (cf. first paragraph of this section), the existence of the sub-pattern can be configured to be allowed or to be (partly) required or forbidden. Global rules allow for specifying constraints on the value of particular attributes (cf. F7). In doing so, it is possible to compare attributes of the nodes and edges (e.g., type, label, NoOfIncomingEdges). Global rules can be defined using logic expressions (e.g., "[X].Caption == [Y].Caption").

Once the pattern query is visually defined, the analysis can be performed. The user is prompted to select the models which are to be queried. The results are displayed together with analysis statistics. The results are highlighted in the models and can be browsed (cf. F9, Figure 2).

5 Contributions, Limitations, and Outlook

In this paper, we developed a modelling language independent model query tool which is able to support multiple analysis scenarios by providing sophisticated means of defining pattern queries visually. We hence contribute to the range of existing approaches, because the functionalities provided by our prototype are only partly given in research by now. A limitation of our approach is that it only supports querying structural properties and does not allow for identifying conflicts within the execution semantics of a model. Future work will evaluate the application potential of the query language in real world model analysis scenarios.

References

1. Davies, I., Green, P., Rosemann, M., Indulska, M., Gallo, S.: How do practitioners use conceptual modeling in practice? Data & Knowledge Engineering 58, 358–380 (2006)
2. Dijkman, R.M., La Rosa, M., Reijers, H.A.: Managing Large Collections of Business Process Models - Current Techniques and Challenges. Computers in Industry 63, 91–97 (2012)
3. Dijkman, R.M., Dumas, M., van Dongen, B., Käärik, R., Mendling, J.: Similarity of business process models: Metrics and evaluation. Information Systems 36, 498–516 (2011)
4. Peffers, K., Tuunanen, T., Rothenberger, M.A., Chatterjee, S.: A design science research methodology for information systems research. Journal of Management Information Systems 24, 45–77 (2007)
5. Knuplesch, D., Ly, L.T., Rinderle-Ma, S., Pfeifer, H., Dadam, P.: On Enabling Data-Aware Compliance Checking of Business Process Models. In: Parsons, J., Saeki, M., Shoval, P., Woo, C., Wand, Y. (eds.) ER 2010. LNCS, vol. 6412, pp. 332–346. Springer, Heidelberg (2010)
6. Becker, J., Bergener, P., Räckers, M., Weiß, B., Winkelmann, A.: Pattern-Based Semi-Automatic Analysis of Weaknesses in Semantic Business Process Models in the Banking Sector. In: Proc. of the European Conference on Information Systems (ECIS 2010), Pretoria, South Africa (2010)
7. Ouyang, C., Dumas, M., ter Hofstede, A.H.M., van der Aalst, W.M.P.: Pattern-Based Translation of BPMN Process Models to BPEL Web Services. International Journal of Web Services Research 5, 42–62 (2008)
8. Mendling, J., Verbeek, H.M.W., van Dongen, B.F., van der Aalst, W.M.P., Neumann, G.: Detection and prediction of errors in EPCs of the SAP reference model. Data & Knowledge Engineering 64, 312–329 (2008)
9. Polyvyanyy, A., Smirnov, S., Weske, M.: Business Process Model Abstraction. In: Brocke, J., Rosemann, M. (eds.) Handbook on Business Process Management 1, pp. 149–166. Springer, Heidelberg (2010)
10. Foggia, P., Sansone, C., Vento, M.: A Performance Comparison of Five Algorithms for Graph Isomorphism. In: Proceedings of the 3rd IAPR TC-15 Workshop on Graph-based Representations in Pattern Recognition, pp. 188–199 (2001)
11. Becker, J., Bergener, P., Delfmann, P., Weiß, B.: Modeling and Checking Business Process Compliance Rules in the Financial Sector. In: Galletta, D.F., Liang, T.-P. (eds.) Proc. of the International Conference on Information Systems (ICIS 2011). Association for Information Systems (2011)