Composition of Semantic Process Fragments to Domain-Related Process Families

Claudia Reuter

Fraunhofer Institute for Software and Systems Engineering, Emil-Figge-Str. 91,
44227 Dortmund, Germany
claudia.reuter@isst.fraunhofer.de

Abstract. Efficient and effective process management is considered as key success factor for competitiveness of enterprises in an increasingly complex and closely connected environment. Today, there exists a plentitude of IT-tools that support modeling, execution, monitoring, and even flexible change of processes. Though, most process management solutions offer possibilities for reusing workflow components, development of new process models is still a cost and time consuming task. Either common process knowledge is scattered among a growing amount of process models or it is divided into unspecific components, the interrelations of which are difficult to manage. This problem becomes even worse considering potential variations of workflows. In the SPOT project, we adapted the feature modeling approach in order to represent enterprise-specific process knowledge in the form of process families. Process families consist of semantically enriched process fragments and enable the composition of business processes that conform to domain-related rules and regulations.

Keywords: Process management, semantic process fragment, compositional process modeling, feature modeling, process families.

1 Introduction

The introduction and successful deployment of business process management (BPM) and optimization techniques represents a major challenge in an increasing number of domains. E.g., with regard to cost pressure and quality requirements, healthcare providers and logistics experts strive for solutions that enable efficient management of processes. However, especially in highly dynamic markets process standardization approaches quickly reach their limits: In the healthcare domain, medical treatment processes have to focus on patients as individuals with individual needs; logistic experts have to tailor procurement and delivery processes to the specific requirements of their customers in order to stay competitive. Therefore, BPM solutions must provide the means to efficiently customize process models to the individual demands of a single case and to flexibly create new workflows based on existing process knowledge.

Although, most BPM tools already offer possibilities for reusing existing process and application components, they still lack a formal concept to develop and

P. van Bommel et al. (Eds.): PoEM 2010, LNBIP 68, pp. 61–75, 2010.

continuously maintain common process knowledge within an institution. Usually, the process knowledge is distributed among a plentitude of processes, sub processes, and application components. However, these fragments only represent the building blocks of process knowledge; equally important are the specification of conditions under which components can be selected and the management of common information about their dependencies and interrelations. In the context of the SPOT project, we analyzed requirements of the domains healthcare and logistics with regard to process management. Facing high flexibility demands, we developed a new approach towards creation and maintenance of domain-related process knowledge in the form of process families. This concept provides the basis for our solution of compositional process modeling, which makes it possible to efficiently derive new business processes according to common process knowledge; our approach not only contributes to the efficiency of process modeling, but ensures that all business processes conform to general rules and regulations of an enterprise.

In this paper, we will introduce Semantic Process Fragments (SPF), which represent basic components for the creation of domain-oriented process knowledge. After a discussion of related work in section 2, we briefly introduce the SPOT project and its novel approach towards compositional process modeling in section 3. Then, we present SPF type graphs as models that enable aggregation and semantic annotation of process fragments to domain-related process families (section 4). Following a presentation of fundamental concepts, we provide a formal semantics for SPF type graphs based on widely-approved feature modeling constructs. This includes a well-specified set of correctness criteria that ensure the soundness of the model. In section 5, we give an outlook on how SPF type graphs facilitate efficient development of standard and case-specific business processes using scenarios from healthcare and logistics domains as example. The paper ends with a short summary of our work (section 6).

2 Related Work

In recent years, the concept of managing potential process variations within the same model became known under the label “process configuration”. In [1], the authors describe a methodology for defining variant-rich processes for the e-business and automotive domain. Thereby, they use variation points in order to adapt a basic process model to the specific requirements of the actual case. This idea was picked up and further developed in context of the Provop approach [2]. Provop even enables the dynamic configuration of processes at runtime based on a well-defined context model. Another approach towards process configuration is proposed by Gottschalk et al., who introduce configurable workflows as common multiple of all process variations combined with a standard configuration [3]. The application of this configuration results in the basic process model representing the standard way of procedure; modifications of the standard configuration lead to variations of the process model. So far, the process modeling languages YAWL [3] and EPC [4] have been extended to support the new concepts. Although, these solutions make it possible to flexibly adapt standard business processes to specific needs, the basic process always remains the core component, from which variations can be derived. Frequently, the basic process

corresponds to a standard way of procedure with regard to a goal, as e.g. the treatment of a specific diagnosis. However, related work in the context of process configuration does not address methods towards management of domain-related process knowledge, which makes it possible to flexibly create different types of process models by composing reusable process fragments.

While Provop and other configurable workflow models comply with the procedural process modeling concept, Declare represents a declarative process modeling language [5]. In this way, the order of execution of process components is no longer determined by the control flow but by specifying constraint relations between process activities. The objective of Declare is the prevention of over-specification, which often leads to very complex workflow models that are difficult to maintain. However, the BPM system Declare supports the flexible arrangement of process activities at runtime; whereas the management of domain-related process knowledge as basis for deriving standard and case-specific process models is not addressed. Although, usage of declarative languages contributes to process flexibility at runtime, they make it difficult to specify standard workflows and reduce the predictability of business processes, which is the main argument to establish BPM solutions in the first place. What is still missing is a method for capturing domain-related process knowledge that enables the efficient creation of business processes in compliance with general practices and regulations of an enterprise.

In [6], the authors also identify the need for a model comprising the entire scope of alternative options and interdependencies between business processes. The paper focus on the introduction of generic ERP modeling steps, which are independent of specific modeling languages and aim at developing such a common model. Furthermore, the approach is evaluated by using OPM (Object Process Methodology) [7] as an example. However, the solution does not define correctness criteria that dictate the way in which the overall model has to be created and ensure that only semantically correct processes can be derived from the model.

3 The SPOT Approach

During the SPOT Project (Service-based and Process-oriented Orchestration Technology)[1] we analyzed requirements from the healthcare and logistics domain on BPM solutions. According to our results, both domains are characterized by very dynamic processes that have to be flexibly and efficiently customized to the case-specific demands. However, due to complexity of current BPM solutions that rely heavily on technology, domain specialists like physicians or logistics managers lack the technical skills, which are necessary to efficiently create and change process models. Thus, they mostly depend on IT-specialists, which possess the technical skills for the development of process models, but have no knowledge how the business processes really work in practice. The SPOT approach stems from the conclusion that domain experts will only be able to modify processes if they can create the initial workflow models by themselves. Consequently, SPOT aims at changing the way, in which process models are developed. Figure 1 shows how the SPOT approach contrasts with the traditional approach towards process modeling.

[1] See http://www.spot.fraunhofer.de/ for details on the SPOT project.

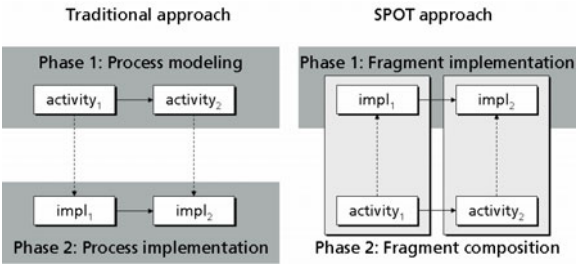

Fig. 1. SPOT approach towards compositional process modeling

Traditionally, the process modeling phase is followed by the implementation phase, which comprises the development of application components and their assignment to process activities. These two procedures must be repeated for every single business process. In SPOT, process development begins with the identification and implementation of process fragments. A process fragment is an executable component that consists of a standardized interface specification and a concrete implementation, as e.g. a web service, a program routine, or a human task interface. This work is usually done by IT-specialists, who can concentrate on their core competencies and do not have to comprehend complex business processes completely. Whereas hospitals and logistic companies often do not have detailed process descriptions at disposal, their service offerings are clearly defined, e.g. in the form of service catalogues, which can be used to identify process fragments. After that, domain experts can overtake the responsibility for the composition of the available process fragments to meaningful business processes. In this way, the two development phases, fragment implementation and fragment composition, better comply with the different competencies of IT-specialists and business professionals. Furthermore, it is possible to create a whole set of workflow models based on the same process fragments.

In order to realize this approach towards compositional process modeling, basically we need a concept that facilitates management of process knowledge in the form of reusable components and determines the way, in which these fragments can be selected and composed into process models.

4 Using Semantic Process Fragments to Build Domain-Related Process Families

In the context of development of software products, the necessity of modeling variant-rich processes is well recognized. A product family represents a set of components which address the demands of a specific market segment, as e.g. the automobile industry. The creation of product families is based on a thorough identification of solution components for a domain by considering possible variations of the resulting product. A widely-approved methodology for the development of product families is the feature modeling approach. This technique was introduced by Kang et al. [8] and since then has been refined in a number of publications [9-15]. In order to facilitate

capturing of domain-related process knowledge, we adopted and extended the feature modeling approach. Thereby, each feature represents a concept that is important within the context of at least one business process. With regard to the healthcare domain, a feature can be a radiological examination or a drug therapy. By way of feature modeling, it is possible to abstract from single business processes and to concentrate on work sections and service offerings of enterprises. Then, services that play a role in several processes are classified in a bottom up manner. In contrast to traditional process modeling, dependencies and relations between features are only specified if they adhere to common rules of an institution and thus, comply with practical process knowledge.

Although, feature models are a well-established concept for modeling variability in product lines, most solution approaches still lack formal semantics [16]. E.g., existing approaches [12-15] do not provide criteria that ensure the correct usage of logical operators or the validity of combinations between operators and constraints. Therefore, we chose the feature modeling constructs that are suited for our goal and formally specified SPF type graphs and respective correctness criteria in order to appropriately design process families.

4.1 Specification of Semantic Process Fragments

Process fragments consist of a standardized interface definition and a concrete implementation. They provide the basis for the development of business processes using a process modeling language, as e.g. BPEL [17], BPMN 2.0 [18], or Workflow Nets [19]. The assignment of application components to fragment specifications is the precondition for the composition of executable process models. The interface definition of process fragments, which are distinguished by their unique identifier, comprises a name, a description, a version number, a role of the responsible agent, as well as a reference to an application component and its respective method. Furthermore, it determines which data objects are passed to the application and which objects are returned to the BPM system responsible for the process execution. IT-specialists who assign implementations to interface specifications have to ensure, that the method parameters of the implementation conform to the definitions provided by the interface. After the realization of process fragments, it is possible to set them into relation by creating a process family as *SPF type graph*. SPF type graphs are acyclic, directed graphs. The following figure illustrates their basic structure.

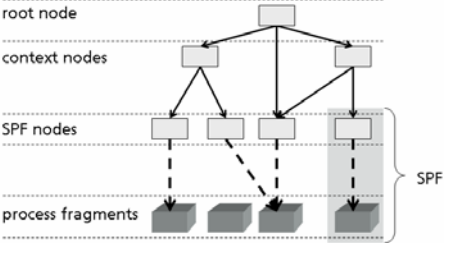

Fig. 2. Basic structure of SPF type graphs

SPF type graphs consist of exactly one root node and an optional set of further nodes. Nodes without child nodes are called *SPF nodes*. Nodes that are neither root nodes nor SPF nodes are called *context nodes*. SPF type graphs assign semantics to process fragments. In this way, they cause the creation of *Semantic Process Fragments (SPF)*, which provide the necessary information about their selection and composition options. As figure 2 shows, an SPF is a process fragment that is assigned to the SPF node of an SPF type graph. In this way, SPF type graphs have to fulfil an important condition that does not hold for feature models: They must provide a function that assigns process fragments to SPF nodes.

4.2 Basic Characteristics of SPF Type Graphs

Principally, SPF type graphs provide the hierarchical and semantic context, which facilitates selection and composition of process fragments to specific process models and ensures that processes conform to the general rules of a domain. In order to illustrate the basic concepts of SPF type graphs, the next figure presents their meta model design as UML class diagram.

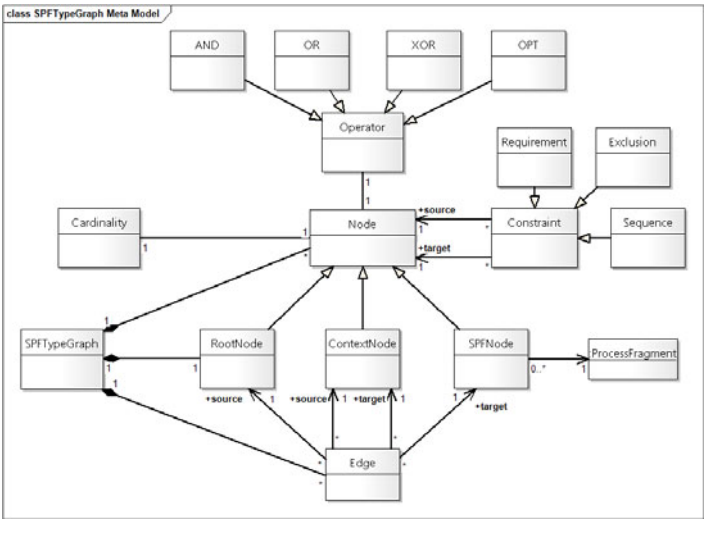

Fig. 3. Meta model of SPF type graphs as UML class diagram

In SPF type graphs, root node, context nodes, and SPF nodes are connected via directed edges. SPF nodes are the only elements that possess a relation to exactly one process fragment; though, the same fragment can be assigned to different SPF nodes. The semantic context of process fragments results from the arrangement of nodes within the graph. As we can see, objects of class type *Node* are related to a logical operator; this operator determines the amount of child nodes that can be selected for a given superior node. We distinguish the following *Operator* class types:

- AND: If a node with AND operator is chosen, all its child nodes have to be selected as well and are therefore mandatory.
- OR: Nodes with OR operator require the selection of at least one successor node.
- XOR: XOR operators only allow for choosing exactly one child node.
- OPT: OPT operators make it possible to select an arbitrary number of successor nodes or none at all. This allows for integrating optional nodes within the graph.

However, with logical operators it is only possible to express relations between successors of the same superior node. Hence, another option to influence the composition of process fragments is given by constraints, which connect two nodes along the horizontal axis. Using constraints, we can specify common rules according to practical process knowledge that should evaluate to true for any business process based on the same SPF type graph. Stemmed from our requirement analysis of healthcare and logistic related processes, we distinguish between the following types of constraints:

- Requirement: The expression "node A requires node B" means that every process model that comprises process fragments associated with node A must at least contain the fragments of B. Furthermore, it must be ensured that the execution of the process fragments of B finishes before the fragments of A are initiated.
- Sequence: The sequence constraint does not impact the selection of process fragments, but it determines their execution order within the process model. The expression "node A precedes node B" means that the process fragments of A must be performed before the fragments of B can be initiated.
- Exclusion: Process fragments associated with nodes that are connected via an exclusion constraint, may not be part of the same process model in the context of the least common sub graph.

The next figure shows how the different types of constraints are visualized in SPF type graphs.

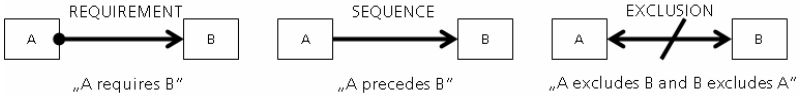

Fig. 4. Representation of possible constraint relations between nodes of SPF type graphs

Finally, it is possible to influence the structure of process models by declaring node cardinalities. In SPF type graphs, the cardinality corresponds to an integer value and indicates the maximum number of times a node can be selected. Multiple instances of nodes are called clone nodes and correspond to loop constructs within traditional process modelling languages.

4.3 Formal Semantics of SPF Type Graphs and Their Correctness Criteria

After an introduction to their basic concepts, we will now give a set-based, formal definition for SPF type graphs and their correctness criteria. An SPF type graph is a tuple STG = (V, E, V_{SPF}, r, C, VO, VC, VP):

- V is a finite nonempty set of unique node labels, including root, SPF nodes, and context nodes.
- $E \subseteq V \times V$ is a finite set of directed edges connecting two nodes with each other. The expression $(v \in V, u \in V) \in E$ indicates the existence of a directed edge leading from v to u. In this respect, we define two abbreviated notations:

$$v \rightarrow u \Leftrightarrow (v, u) \in E \text{ (u is direct successor of v)} .$$

$$v \rightarrow^* u \Leftrightarrow v \rightarrow u \vee \exists z \in V: v \rightarrow z \wedge z \rightarrow^* u \text{ (path from v to u)} .$$

- $V_{SPF} = \{v \in V \mid \nexists u \in V: v \rightarrow u\}$ is the set of all SPF nodes.
- $r \in V$ is the root node of the SPF type graph.
- $C \subseteq V \times V \times CT$ is a set of constraints that establish constraint relations between nodes, with CT = {REQUIREMENT, SEQUENCE, EXCLUSION} being the set of possible constraint types. E.g. the expression $(v \in V, u \in V, \text{SEQUENCE}) \in C$ means that there is a constraint relation from v to u of type SEQUENCE.
- VO: $V \rightarrow$ {AND, OR, XOR, OPT} is a function that assigns a logical operator to each node. The operator defines the rules under which the child nodes of the current node have to be selected.
- VC: $V \rightarrow \mathbb{N} \setminus 0$ is a function that assigns a cardinality to each node. The cardinality determines the upper limit of the number of clone nodes.
- VP: $V_{SPF} \rightarrow P$ is a function that assigns a process fragment to each SPF node, with P being the set of all available process fragments.

This formal specification states what types of elements may appear in SPF type graphs and in which way they can be interconnected. Based on this formalism, it is possible to define a set of correctness criteria that determine the structure of SPF type graphs. For each SPF type graph STG = (V, E, V_{SPF}, r, C, VO, VC, VP) the following conditions must hold:

Root Conditions. Each SPF type graph must have exactly one root node, which does not possess any superior nodes. If an SPF type graph solely consists of a root node, the root node simultaneously represents an SPF node, which must be related to a process fragment. The cardinality of the root node is always one.

$$r \in V \text{ (existence of the root node)} .$$

$$\nexists v \in V: v \rightarrow r \text{ (no superior node)} .$$

$$VC(r) = 1 \text{ (no clone nodes)} .$$

$$|V| = 1 \Leftrightarrow r \in V_{SPF} \text{ (root as SPF node)} .$$

SPF Node Condition. Besides the fact, that SPF nodes do not have any child nodes, each SPF node must refer to exactly one process fragment within P.

$$\forall v \in V_{SPF} \; \exists ! \, p \in P: VP(v) = p .$$

Reachability Condition. There must be a path of directed edges from the root node to any other node within the SPF type graph, whether it corresponds to a context or an SPF node. Though, it is possible that a given node can be reached using different paths. In this way, the semantic context of clone nodes may also differ, depending on its route of selection.

$$\forall v \in V: v \neq r \Rightarrow \exists\ u \in V: u \rightarrow v\ .$$

Acyclic Graph Condition. With regard to their set of hierarchic, directed edges, SPF type graphs have to be acyclic.

$$\forall v, u \in V: v \rightarrow^* u \Rightarrow \neg\ (u \rightarrow^* v)\ .$$

Constraint Conditions. The definition of constraint relations is forbidden for nodes that are connected via hierarchical edges.

$$\forall v, u \in V, ct \in CT: (v, u, ct) \in C \Rightarrow \neg\ (v \rightarrow^* u \vee u \rightarrow^* v)\ .$$

More specifically, two constraints must not refer to the same scope regarding the semantic context of an SPF type graph. The scope always consists of the nodes that are connected via the constraint as well as their associated sub graphs. A constraint relation $(v, u, ct) \in C$ means that it is not possible to define further constraints between the sub graphs of v and u respectively. In order to be able to formally specify this condition, first we have to introduce an auxiliary function minNode: $2^V \rightarrow V$ that serves the purpose of identifying the least common multiple node of a given set of nodes. Let $Z \subseteq V$ be a non-empty set of nodes of an SPF type graph; then minNode can be defined in the following way:

$$\text{minNode}(Z) = \{v \in V |\ v \rightarrow^* Z \setminus v \wedge \neg\ (\exists\ u \in V: u \rightarrow^* Z \wedge v \rightarrow^* u)\}\ .$$

According to this definition, it is possible that the least common superior node is part of Z itself. Now, we can define the correctness criterion that ensures that the scope of a constraint relation does not overlap with the scope of another constraint:

$$\forall v, u \in V, ct_1 \in CT: (v, u, ct_1) \in C \Rightarrow \nexists x, y \in V, ct_2 \in CT: (x, y, ct_2) \in C \setminus (v, u, ct_1) \wedge ((v \rightarrow^* x \vee u \rightarrow^* x \vee x = v \vee x = u) \wedge ((v \rightarrow^* y \vee u \rightarrow^* y \vee y = v \vee y = u) \wedge (\text{minNode}(x, y) \rightarrow^* v \vee \text{minNode}(x, y) \rightarrow^* u)\ .$$

The following figure presents an example that shows the importance of this condition.

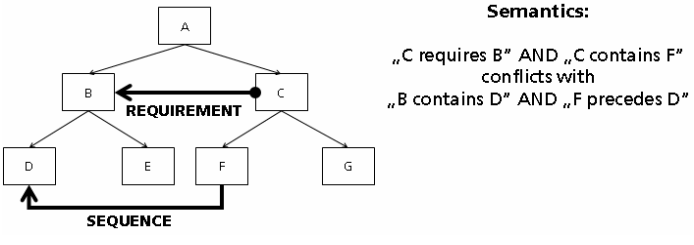

Fig. 5. Example of a conflict between constraints within the same scope

In the example above, the sequence constraint between the nodes D and F violates the requirement condition between C and B, because it is part of the same scope. This criterion also prohibits the specification of constraints, where source and target are identical as well as loop constraints.

The next condition refers to the transitivity of constraint relations. If node v requires node u and u requires z, v requires z as well; please note, that the same is true for sequence constraints.

$$\exists\, v, u, z \in V\text{: } (v, u, \text{REQUIREMENT}), (u, z, \text{REQUIREMENT}) \in C \Rightarrow \exists\ (v, z, \text{REQUIREMENT}) \in C\ .$$

Finally, it is not allowed to define requirement and exclusion constraints in such a way, that it is not possible to fulfill the conditions:

$$\exists\, v, u, z \in V\text{: } (v, u, \text{REQUIREMENT}), (v, z, \text{REQUIREMENT}) \in C \Rightarrow \nexists\, (u, z, \text{EXCLUSION}) \in C \wedge \nexists\, (z, u, \text{EXCLUSION}) \in C\ .$$

Conditions with Respect to Combinations of Logical Operators and Constraints. Besides correctness criteria that only relate to the specification of constraints within SPF type graphs, we also have to deal with problems arising from invalid combinations of logical operators and constraints. Regarding child nodes of a parent with XOR operator, it is not allowed to specify constraints at all.

$$\forall\, v, u \in V\ \exists\, z \in V\text{: } z \rightarrow^* v \wedge z \rightarrow^* u \wedge VO(z) = XOR \Rightarrow \neg(v, u, ct \in CT) \in C\ .$$

Furthermore, if two nodes are connected via exclusion constraint, it must be ensured that there is at least one option that avoids the selection of both nodes**.**

$$\forall\, v, u \in V\text{: } (v, u, \text{EXCLUSION}) \in C \Rightarrow \exists\, z \in Z\text{: } VO(z) = OPT \vee (VO(z) \neq \text{AND} \wedge (\exists\ x \in V \setminus Z\text{: } z \rightarrow x)) \text{ with } Z = \{z \in V \mid z = \text{minNode}(v, u) \vee ((z \rightarrow^* v \vee z \rightarrow^* u) \wedge \text{minNode}(v, u) \rightarrow^* z)\}\ .$$

According to this formula, Z comprises all nodes that correspond to any node that is located between v or u and their least common superior node or represents the least common superior node itself. Either one of these nodes is associated with the logical operator OPT or it must contain a child node as alternative option to v or u.

5 Compositional Process Modeling in Healthcare and Logistics

In this section, we would like to clarify the contribution of SPF type graphs to compositional process modeling by way of examples from the healthcare and logistics domain.

5.1 Treatment Processes for Diagnostics of Spinal Diseases

About three quarters of the adult population will experience back pain during their lifetime. The medical treatment can be a variable process depending on the severity level of a spinal disease and the individual situation of a patient. However, process

management solutions that facilitate the coordination of the activities of the participating healthcare providers can considerably contribute to the success of the treatment. Assuming that a hospital has already begun to maintain process knowledge in the form of SPF type graphs. Given the process family for diagnostic procedures, as illustrated in figure 6, it is possible to compose new process models for the diagnostics of spinal diseases based on semantic process fragments.

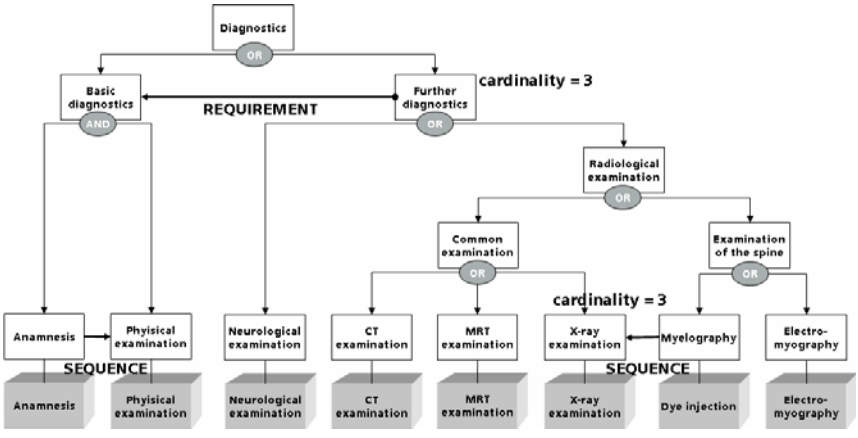

Fig. 6. Example of an SPF Type graph for diagnostic procedures

According to this graph, diagnostics can be distinguished in basic and further diagnostics. While basic diagnostics schedules the anamnesis and physical examination, further diagnostics is related to neurological and radiological examinations. Though, considering the requirement constraint, further examinations must not take place as long as the basic diagnostics is not completed. With respect to radiological examinations, we divide common examinations from examinations of the spine. Common examinations, such as X-ray, MRT, and CT, can be performed in the context of many diagnoses; whereby myelography and electromyography are used to diagnose the source of back pain. The myelography involves the injection of contrast medium into the spine, followed by X-ray examinations. As the node cardinalities show, it is possible to repeat X-ray examinations or even the whole complex of further diagnostics up to three times.

Now, new processes derived from the SPF type graph can either outline the standard way of procedure for patients that suffer from back pain or they can deal with individual treatment cases. The next figure visualizes the standard process for diagnostics of spinal diseases according to the SPF type graph.

The order of execution of the process activities is determined by the constraint relations. Due to a sequence constraint, the physical examination always follows the anamnesis. As there is no constraint between MRT and X-ray, the examinations can be performed in arbitrary order. Note that a constraint between two context nodes also adheres to the nodes of their associated sub graphs.

In standard treatment cases, physicians perform an X-ray and an MRT after completion of anamnesis and physical examination. Though, assume that we have to deal with a patient that is suspected to suffer from a recurrent lumbar disc hernia;

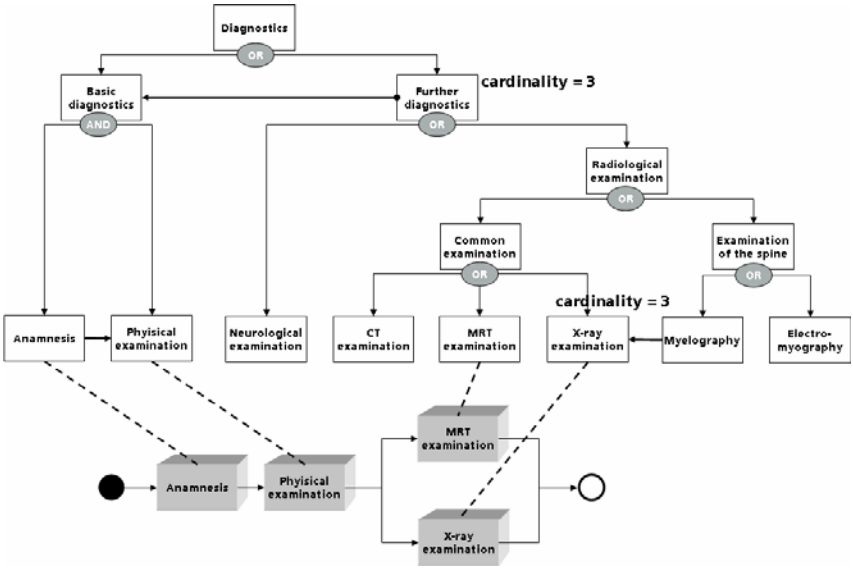

Fig. 7. Basic process for the diagnostics of spinal diseases

therefore, a myelography is indicated to distinguish a relapse from a local post-operative scar formation. A myelography is an x-ray scan that is performed after dye has been injected into the spinal fluid. Consequently, the diagnostic procedures for this patient must differ from the standard process; figure 8 illustrates the individual workflow model.

Fig. 8. Case-specific process for the diagnostic of spinal diseases

In this way, SPF type graphs provide the basis for deriving standard processes, which are executed for the majority of patients, as well as individual processes depending on the requirements of specific treatment cases.

5.2 Logistic Process for the Transportation of Goods

Goods have to be transported from their place of production to the various locations of the customers. Generally, logistic processes can vary according to the type of transfer and the transportation mode as illustrated in the following SPF type graph.

As the figure shows, there are different modes of transportation, which primarily depend on source, destination, costs, and safety. Potential carriers are air-freight, railroad, waterway, and motor-freight carriers. In some cases it is possible to directly transfer the goods from origin to destination; thereby, the process only comprises

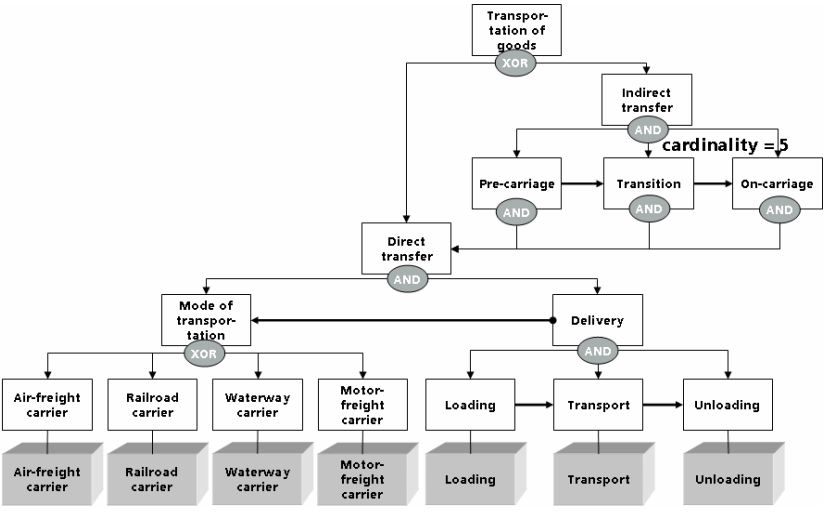

Fig. 9. Example of an SPF type graph for the transportation of goods

selection and arrangement of the transportation mode as well as the loading, transport, and unloading of the goods. Though, in case of air-freight, railroad, and waterway carrier, mostly it is also necessary to organize the pre- and on-carriage. The drive of the lorry from the loader to the terminal of departure is considered as pre-carriage; on-carriage covers the delivery of the freight from the terminal of destination to the receiving costumer. Therefore, one or more transitions occur between pre- and on-carriage. The SPF type graph defines a maximum number of five transitions.

Now, a logistics manager composes for her new customer a transport process; as mode of transportation she is going to choose air-freight carrier. In order to organize the transport to and from the airport terminals, she decides on pre- and on-carriage via motor-freight carrier. The next figure shows the resulting process model.

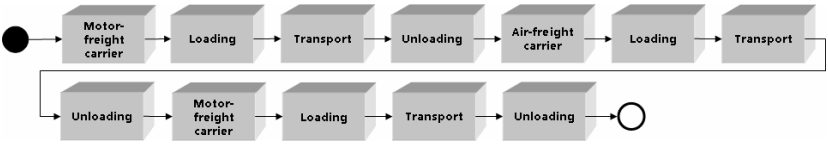

Fig. 10. Case-specific process for the indirect transfer of goods

With our approach of compositional process modeling, the SPF type graph makes it possible to efficiently tailor logistics processes to the specific needs of a customer.

6 Conclusion

The necessity to specify detailed business processes on a technical level represents a major obstacle for using BPM systems. BPM repositories for management of sub

workflows and application components still lack possibilities to formally define the context of their usage and their interrelations with each other. In order to enable our approach towards compositional process modeling, we formally defined SPF type graphs as models to represent domain-oriented process families. SPF type graphs are a specific kind of feature model that allow the management of the common process knowledge within an enterprise. In contrast to other solutions in the area of process configuration, SPF type graphs do not correspond to basic processes, which contain variable elements that can be customized to the actual needs of a case. As we have shown by example of processes from the healthcare and logistic domains, it is rather possible to derive standard as well as case-specific workflows based on the same SPF type graph. Furthermore, our methodology makes process modeling easier, because IT-specialists can concentrate on the task of identifying and developing process fragments according to the service offerings of an enterprise; then, the composition of fragments to business processes can be done by domain experts. In our future work, we will formally specify criteria that ensure that fragment compositions conform to the definitions of an SPF type graph. Moreover, we will determine the rules that control the transformation of declarative process knowledge into specific process models using languages that follow the procedural modeling paradigm.

References

1. Bayer, J., Buhl, W., Giese, C., Lehner, T., Ocampo, A., Puhlmann, F., Richter, E., Schnieders, A., Weiland, J., Weske, M.: Process Family Engineering, Modeling variant-rich processes. Fraunhofer IESE Report No. 126.06/E, Version 1.0 (2005)
2. Hallerbach, A., Bauer, T., Reichert, M.: Issues in Modeling Process Variants with Provop. In: Ardagna, D., Mecella, M., Yang, J. (eds.) Business Process Management Workshops. Lecture Notes in Business Information Processing, vol. 17, pp. 56–67. Springer, Heidelberg (2009)
3. Gottschalk, F., van der Aalst, W.M.P., Jansen-Vullers, M.H., La Rosa, M.: Configurable Workflow Models. International Journal of Cooperative Information Systems 17(2), 223–225 (2008)
4. Rosemann, M., van der Aalst, W.M.P.: A Configurable Reference Modeling Language. Information Systems 32, 1–12 (2007)
5. Aalst, W.M.P.: Constraint-Based Workflow Models: Change Made Easy. In: Curbera, F., Leymann, F., Weske, M. (eds.) OTM 2007, Part I. LNCS, vol. 4803, pp. 77–94. Springer, Heidelberg (2007)
6. Soffer, P., Golany, B., Dori, D.: ERP modeling: a comprehensive approach. Information Systems 28, 673–690 (2003)
7. Dori, D.: Object Process Methodology – a Holistic Systems Paradigm. Springer, Heidelberg (2002)
8. Kang, K., Cohen, S., Hess, J., Nowak, W., Peterson, S.: Feature-oriented domain analysis (FODA) feasibility study. Technical Report CMU/SEI-90-TR-21, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, USA (1990)
9. Kang, K., Kim, S., Lee, J., Kim, K.: FORM: A Feature-Oriented Reuse Method. Annals of Software Engineering 5, 143–168 (1998)
10. van Deursen, A., Klint, P.: Domain-Specific Language Design Requires Feature Descriptions. Journal of Computing and Information Technology 10(1), 1–17 (2002)

11. van Gurp, J., Bosch, J., Svahnberg, M.: On the Notion of Variability in Software Product Lines. In: Proceeding of the Working IEEE/IFIP Conference on Software Architecture (WICSA), pp. 45–55. IEEE Computer Society Press, Washington (2001)
12. Czarnecki, K., Helsen, S., Eisenecker, U.W.: Formalizing cardinality-based feature models and their specialization. Software Process: Improvement and Practice 10(1), 7–29 (2005)
13. Mannion, M.: Using First-Order Logic for Product Line Model Validation. In: Chastek, G.J. (ed.) SPLC 2002. LNCS, vol. 2379, pp. 176–187. Springer, Heidelberg (2002)
14. Schobbens, P.-Y., Heymans, P., Trigaux, J.-C.: Feature Diagrams: A Survey and a Formal Semantics. In: Proceedings of the 14th IEEE International Requirements Engineering Conference (RE 2006), pp. 136–145 (2006)
15. Sun, J., Zhang, H., Li, Y.F., Wang, H.: Formal Semantics and Verification of Feature Modeling. In: Proceedings of the 10th IEEE International Conference on Engineering of Complex Systems (ICECCS 2005), pp. 303–312 (2002)
16. Riebisch, M., Streitferdt, D., Pashov, I.: Modeling variability for object-oriented product lines – workshop report. In: Buschmann, F., Buchmann, A., Cilia, M.A. (eds.) ECCV-WS 2003. LNCS, vol. 3013, pp. 165–178. Springer, Heidelberg (2004)
17. OASIS: Web Services Business Process Execution Language Version 2.0. OASIS Standard (April 2007), http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html (last access of May 12, 2010)
18. OMG: Business Process Model and Notation (BPMN) FTF Beta 1 Version 2.0 (August 2009), http://www.omg.org/spec/BPMN/2.0/Beta1/PD (last access of May 12, 2010)
19. van der Aalst, W.M.P.: The Application of Petri Nets to Workflow Management. The Journal of Circuits, Systems and Computers 8(1), 21–66 (1998)