Goal-Driven Composition of Business Process Models

Benjamin Nagel, Christian Gerth, and Gregor Engels

s-lab - Software Quality Lab
University of Paderborn
Zukunftsmeile 1
33102 Paderborn, Germany
{bnagel,gerth,engels}@s-lab.upb.de

Abstract. Goal-driven requirements engineering is a well-known approach for the systematic elicitation and specification of strategic business goals in early phases of software engineering processes. From these goals concrete operations can be derived that are composed in terms of a business process model. Lacking consistency between goal models and derived business processes especially with respect to the dependencies between goals can result in an implementation that is not in line with the actual business objectives. Hence, constraints indicated from these dependencies need to be considered in the derivation of business process models. In previous work, we introduced the extended goal modeling language Kaos4SOA that provides comprehensive modeling capabilities for temporal and logical dependencies among goals. Further, we presented an approach to validate the consistency between goal models and business process models regarding these dependencies. Extending the previous work, this paper presents a constructive approach for the derivation of consistent business processes from goal models. We introduce an algorithm that calculates logically encapsulated business process fragments from a given goal model and describe how these fragments can be composed to a business process model that fulfills the given temporal constraints.

Keywords: Requirements engineering, goal models, business process models, business process composition.

1 Introduction

Goal-driven requirements engineering has emerged as a paradigm for the elicitation and specification of requirements in an early phase of the software lifecycle [9,19]. Goal models support the systematic definition of objectives in terms of goals that are structured hierarchically in a goal tree. In the domain of service-oriented enterprise applications these goal models are usually used to capture business goals that need to be achieved. By the iterative refinement of these goals, concrete operations are identified, that need to be performed to achieve

A.R. Lomuscio et al. (Eds.): ICSOC 2013 Workshops, LNCS 8377, pp. 16–27, 2014.

the defined goals [1,20]. These operations are used as input for the definition of business processes composing these operations to a sequence of activities.

Recent research addresses the relations between goal models and business process models by different approaches that explicitly consider the links and relationships between elements in goal models and business process models [6,8]. The explicit consideration of these links ensures completeness and traceability among both models.

However, the initial composition of operations to business processes is still an open challenge. The identified operations cannot be composed in an arbitrary way, since the different types of relationships between goals (AND-, OR-decompositions) need to be considered. In addition, domain-specific knowledge of stakeholders about dependencies between goal, e.g. the order in which goals need to be achieved, have to be considered as well.

In previous work we contributed two extensions addressing this topic. In [14] we presented an extension of KAOS goal models, termed Kaos4SOA. This approach enables the specification of temporal and logical dependencies among goals. Hereby, we enable the elicitation and modeling of the stakeholders' knowledge about dependencies among goals that need to be considered in the derivation of business processes. Further, we presented a consistency validation approach in [13]. We demonstrated the generation of formalized business process quality constraints from these goal dependencies and showed how a derived business process model can be validated against these constraints.

Extending the previous work, we introduce a constructive approach for the systematic derivation of consistent business process models from Kaos4SOA goal models. To solve the composition problem in a sufficient way we state the following requirements to our approach. First, it can not be guaranteed that all operations are constrained in a way, that they can be composed unambiguously, i.e. there is no unique valid business process model that achieves the goals and fulfills the defined quality constraints. Second, the usage of model-checking provides a high degree of automation, but the computational complexity often raises performance issues especially for large business process models with a high number of constraints to be validated. To enable an efficient usage of model-checking that guides the process designers, the number of process elements and constraints that are validated need to be reduced.

Addressing these requirements, we present a goal-driven approach that enables the systematic derivation of consistent business process models from goal models. By analyzing the logical decompositions through the goal model, business process fragments are calculated encapsulating a set of dependent operations. Applying a set of business process patterns, the operations in each fragment are composed considering the dependencies among them. Dependencies between these fragments are calculated and finally, the fragments are composed to a business process model according to the temporal dependencies by using model-checking techniques.

By realizing the presented solution our approach makes the following contributions:

1. A method for the automatic identification and clustering of business process fragments from a given goal model.
2. A pattern-based approach for the composition of operations in fragments.
3. An model-checking approach for the composition of business process models based on fragments, which are significantly smaller than the business process itself.

The remainder of the paper is structured as follows. In Section 2 we introduce the foundations for our work. Our approach for the goal-driven composition of business processes is presented in Section 3. Related work is discussed in Section 4 and finally Section 5 concludes this paper.

2 Foundations

2.1 Goal Models

Recent research in goal-driven requirements engineering brought up several approaches for the elicitation and specification of goal models. For the expression of these goal models, different notations, like KAOS [5], Tropos [4] and i* [21] have been developed, that provide languages for the definition of goals and relationships among them. Due to its expressiveness and understandability KAOS has been adopted by several approaches [3,8] to specify goal models in the domain of service-oriented systems.

To illustrate the modeling capabilities of KAOS, an example is depicted in Figure 1 that defines a simplified goal model from the scenario introduced in [10]. *Fulfill book order* is the overall root goal that is decomposed to four subgoals. The AND-decomposition expresses that all subgoals need to be achieved in order to achieve the higher-level goal. These subgoals can be further decomposed as exemplary shown for the goal *Payment received*. This goal is OR-decomposed to the subsubgoals *Payment via credit card* and *Payment via money order*, which means that the payment can be received by either credit card or money order.

As illustrated by the ellipses, each leaf goal is operationalized to one or more operations. For example the goal *Book delivered* is achieved by performing the operations *Deliver to courier* and *Courier delivers to customer*. That means all operations assigned to a leaf goal need to be performed in order to achieve it sufficiently.

In previous work we extended KAOS by a concept for expressing temporal relationships between goals and a more precise definition of logical decompositions. To avoid an increasing complexity of the goal models, the temporal dependencies are expressed by goal annotations. The dependency predecessor/successor between two goals G_1 and G_2 expresses that a goal needs to be achieved before or after another goal. Temporal dependencies can only be defined between goals that are AND-decomposed through the whole hierarchy of the goal model, because it is not feasible to define a mandatory temporal dependency among alternative goals in an OR-decomposition.

An example of an order dependency for the goal model depicted in Figure 1 is the dependency between goal *Books delivered* and *Books available* that states

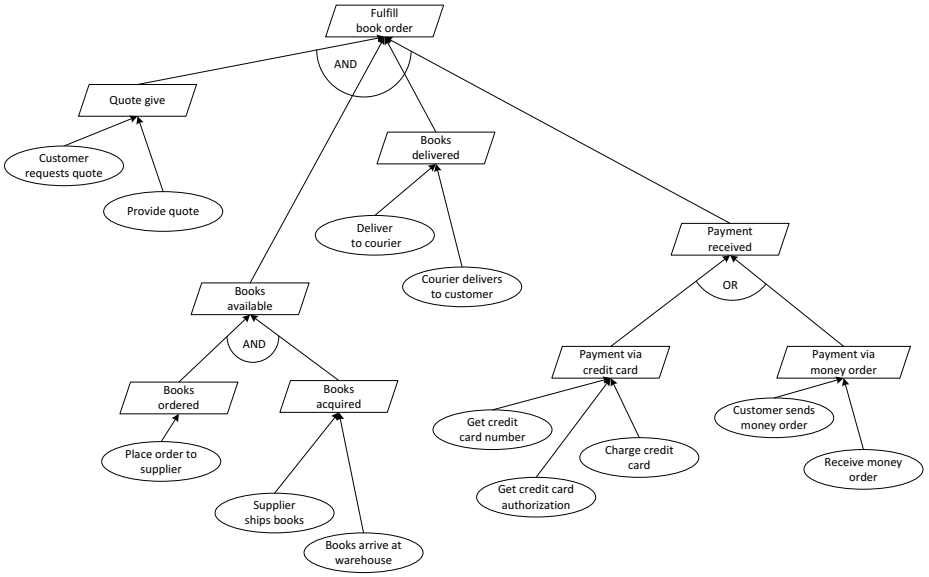

Fig. 1. Exemplary KAOS Goal Model

that the books cannot be delivered until they are available. To express this dependency the goal *Books delivered* is annotated with `Order.Predecessor` *Books available*. A temporal succeeding dependency can be specified between the goals *Payment received* and *Books delivered*. To make sure that the books are delivered after the payment has been received the following annotation can be used. The strict order dependency is expressed with the annotation `Order.Successor` *Books delivered*.

To enable a more precise specification of the decomposition relations between goals we introduced the XOR-decomposition for the explicit distinction of dependencies from the inclusive-OR provided by the KAOS notation. Applied to our running example the OR-decomposition of goal *Payment received* is not precise enough, since the OR-decomposition between subgoals *Payment via credit card* and *Payment via money order* should be exclusive as the customer will pay either by credit card or money order and not both. The updated decomposition with the corresponding conditions is depicted in Figure 2. Our extension also facilitates the definition of conditions for inclusive-OR decompositions.

2.2 Business Process Modeling

Business process models provide visual representations for business processes by describing sequences of activities and gateways connected by edges, defining the order in which the activities are performed. These models enable a common understanding, the analysis of business processes, and also define the required composition of services. To precisely specify business process models in an

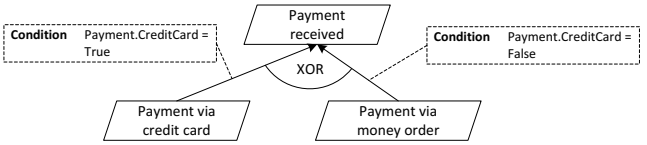

Fig. 2. Exemplary Definition of XOR-decomposition

understandable way, existing process modeling languages like BPMN [16] or UML activity diagrams [15] can be used.

In our approach, we leverage the generic business process modeling language introduced in [7]. Business process models defined in this language can be translated to BPMN. Hence, the usage of this language does not reduce the applicability of our approach. Compared to existing modeling languages this notation supports the explicit definition of business process fragments. A business process fragment encloses a set of business process model elements. These fragments are single-entry-single-exit fragments, that means they have a unique single entry node and a unique one exit node.

3 Approach

To enable the derivation of business process models from goal models, we introduce the goal-driven approach illustrated in Figure 3. As input for our approach we use a given goal model following the Kaos4SOA notation and a set of CTL constraints. These constraints are formal representations of the temporal and logical dependencies that are identified and defined by the approach presented in [13]. By an iterative refinement the used CTL constraints are expressed on the level of operations, i.e. they express constraints between operations.

In the first step, business process fragments which cluster logically related operations are identifed from the goal model. The operations in each fragment are composed using a set of business process patterns according to their logical dependencies in the goal model. By using the given CTL constraint, temporal dependencies between fragments are calculated based on the clustered operations. Using these constraints the fragments are composed to a valid business process model. The three steps of our approach are explained in the following.

3.1 Clustering of Business Process Fragments

The first step of our approach calculates business process fragments from a given goal model. To that extent, we identify operations that can be clustered in fragments. The operations in the fragments can be composed by applying a set of defined business process patterns based on their logical relationships. For this purpose, all decomposition links through the goal model need to be considered for each operation. Therefore we provide a top-down approach starting from the root goal that considers the complete hierarchy of decomposition links.

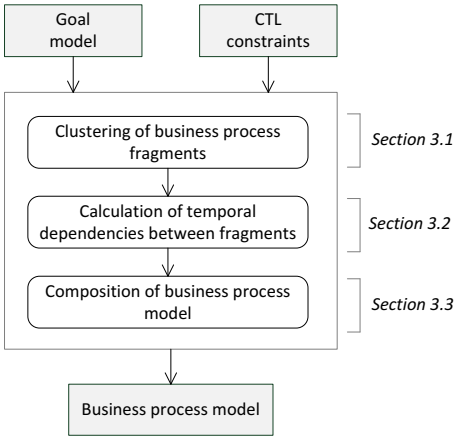

Fig. 3. Conceptual Overview of the proposed Approach

The clustering algorithm is specified in Algorithm 1 and explained in the following. Starting from the root goal, the first fragment is created representing the overall business process that is composed. Then, for each child-goal the algorithm is executed recursively. For each child, that is not a leaf-goal, the further processing depends on the type of decomposition that the goal is part of.

Following the KAOS semantics all goals and it's assigned operations are considered in the business process composition, but of course some goals may be optional, e.g. in an OR-decision. Hence, an AND-decomposition does not intend an additional logical dependency despite the fact that all goals need to be considered. That means, we are able to create logically independent fragments for each goal in an AND-decomposition. The composition of the different fragments with respect to the temporal dependencies among them is part of the following steps.

In our approach the goals are used as temporary elements in the business process model that are refined to subgoals and finally replaced by the operations fulfilling these goals. To compose the goals and operations according to their logical relations, we leverage the business process patterns proposed in [18]. An overview of the patterns used in our algorithm is given in Figure 4. Goals in an OR-decomposition are composed by applying an inclusive-OR gateway (P3). In the case of an conditional OR-decomposition the defined conditions are added to the OR-gateway and pattern P4 is applied. The goals in an XOR-decomposition are composed by an exclusive OR gateway using pattern P5.

Algorithm 1. Cluster Business Process Fragments

```
function ClusterFragments(Goal goal, Fragment frag)
    if goal.isRootGoal() then
        processModel = createProcessModel(goal.name)
        for all childGoal in goal.getChildGoals() do
            ClusterFragments(childGoal, processModel)
    else if goal.isLeafGoal() then
        for all operation in goal.getOperations() do
            newFrag.addElement(operation)
            if goal.getOperations().count() ≥ 2 then
                if operation canBeParallelizedWithOperationIn(newFrag) then
                    composition = applyProcessPattern(P2)
                else
                    composition = applyProcessPattern(P1)
                replace(goal, composition)
            else
                replace(goal,operation)
    else
        if goal.isPartOfORDecomposition then
            applyProcessFragment(P3)
        else if goal.isPartOfCondORDecomposition then
            applyProcessFragment(P4)
        else if goal.isPartOfXORDecomposition then
            applyProcessPattern(P5)
        else                                          ▷ AND-decomposition
            newFrag = createFragment(goal.name)
            frag.addElement(newFrag)
        for all childGoal in goal.childGoals do
            ClusterFragments(childGoal, newFrag)
```

Following the algorithm each goal in the model is refined until a leaf goal is reached. Finally, each goal is replaced by its operations. If a goal is operationalized by exactly one operation, it is replaced by it. More than one operation means that all operations need to be performed to achieve the stated goal. In this case it is checked if the execution of the operations can be parallized. Depending on that, the pattern P2 (parallel execution possible) or P1 (no parallel execution possible) is applied to compose the operations. The order in which the operations need to be performed in a sequence (P1) is decided manually by the business process designer.

An exemplary execution of the algorithm for an excerpt of the running example is shown in Figure 5, which uses the running example introduced in Section 2.1 with the XOR decomposition depicted in Figure 2. Following the decomposition links in the goal model, a new fragment *Payment received* is created and added to the process *Fulfill book order*. The two subgoals are added to the fragment by applying pattern P5, adding an exclusive OR. The temporary goal construct is then replaced by its operations. In this example the goal *Payment*

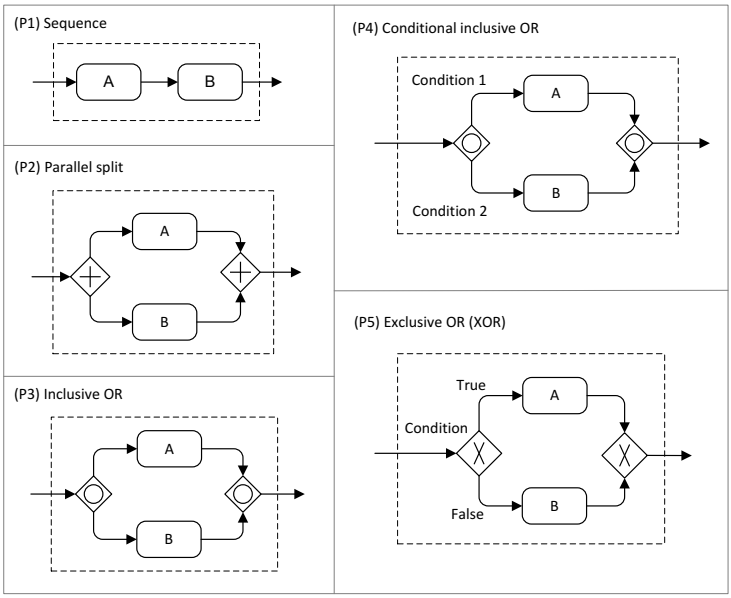

Fig. 4. Business Process Patterns (based on [18])

via credit card is replaced by three operations composed as a sequence (pattern P1).

The result of the presented algorithm is a frame for a business process model that encapsulates all required operations clustered in business process fragments. To complete the business process model, the fragments need to be composed. For this purpose, we first calculate temporal dependencies between these fragments (Section 3.2) and provide a composition approach based on model-checking (Section 3.3).

3.2 Calculation of Temporal Dependencies between Fragments

To enable the composition of the clustered fragments temporal dependencies between operations contained in the fragments need to be considered. As discussed in Section 2.1 temporal dependencies can only be defined between goals in AND-decompositions. Following Algorithm 1 the goals in AND-decompositions are encapsulated in different fragment, which means that temporal dependencies are always stated between operations in different business process fragments.

Algorithm 2 provides a precise definition of the proposed calculation approach. The algorithm iterates through all stated temporal constraints. Each constraint is defined by expressing temporal relations between two or more operations. To derive constraints for fragments, each operation in the constraint is replaced by the business process fragment it is assigned to.

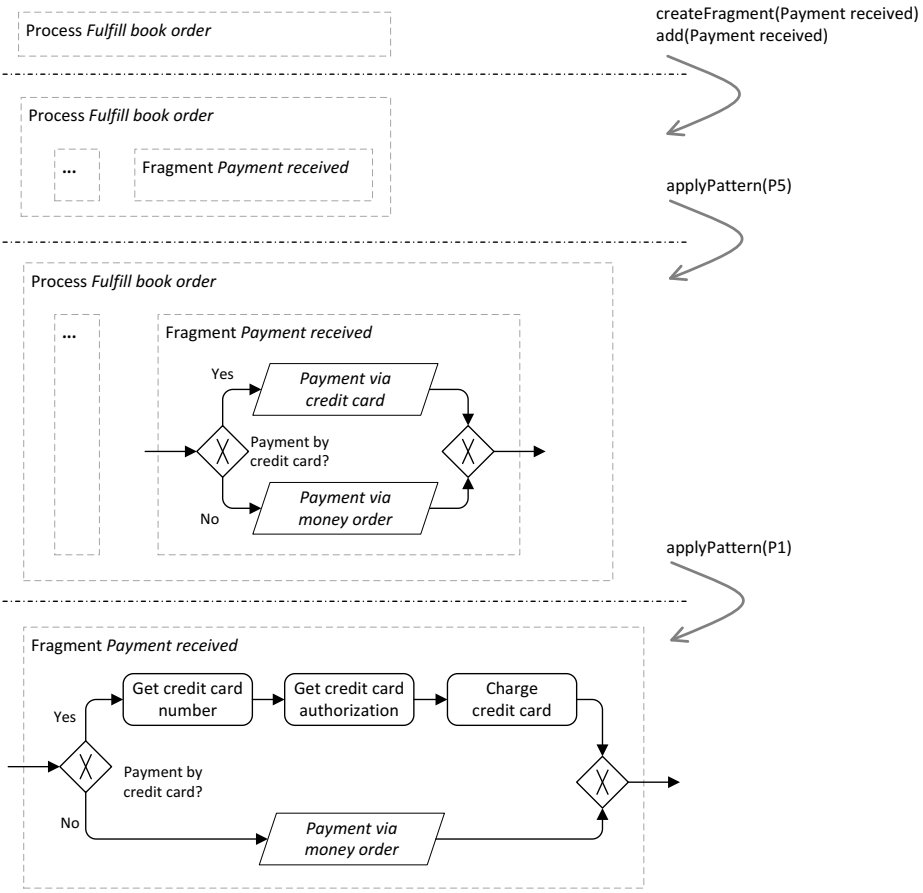

Fig. 5. Exemplary Execution of Composition Algorithm

Algorithm 2. Calculate Temporal Dependencies between Fragments

```
function CalculateFragDependencies(TempConstraints tempConstraints)
    for all tempConstraint in tempConstraints do
        operations = tempConstraint.getElements()
        for all operation in operations do
            tempConstraint.replaceoperation, operation.getFragment()
```

As a result, the algorithm provides a set of CTL constraints that define temporal dependencies among the clustered business process fragments. For example, the temporal succeeding dependency between two fragments F_1 and F_2 is expressed in CTL as follows: $AG(F_1 \rightarrow AF(F_2))$. In the next step, the identified fragments and the dependencies among them are used to compose a valid business process model that fulfills the given constraints.

3.3 Composition of Business Process Model

Depending on the specification in the goal model, the number of constraints itself as well as the number of constrained fragments can vary. That means, not all business process fragments do have temporal relations with other fragments. As a consequence, in some cases only parts of the business process model can be composed automatically based on the given constraints. For all unconstrained fragments our approach favors the manual composition rather than automatically choose an arbitrary position.

Therefore, the composition of the business process model in our approach comprises two steps. In the first step, a valid composition of the constrained business fragments is calculated automatically. Second, the unconstrained fragment are integrated manually into the business process model.

For the constructive composition of a business process based on a set of constraints the possible combinations need to be validated. Details for the definition of possible compositions and their verification can be found, e.g. in [17]. The advantage of our approach is that not all combinations of all available operations need to be considered. By using the clustered business process fragments, the number of elements that need to composed and as consequence the number of combinations that need to be verified can be reduced significantly.

After a valid composition has been identified, the unconstrained fragments need to integrated into the business process model as well. We consider this a completely manual step based on the domain knowledge of the business analyst.

4 Related Work

The derivation of operationalized requirements and architectural models has been addressed by recent research [12,20] which does not specificly address the composition of business process models. In [22] a pattern-based approach is presented that supports the derivation of component diagrams from goal models. While this work focuses on structural aspects the derivation of business process models is not considered in terms of a concrete algorithm for the calculation of a process composition.

For the domain of adaptive, service-oriented system the work in [2] introduces an approach for the automated service composition by matching pre- and postconditions of operations from goal models. This approach requires an exact matching of these conditions to provide a complete composition.

In [11] an approach for the derivation of business process models from goal models is proposed. By presenting a defined procedure this work provides methodical guidance for the identification of services and their compositions, but does not provide any kind of automated composition capabilities. Based on a qualitative preference analysis, the framework presented in [17] automatically calculates service compositions. Compared to our approach this framework does not consider temporal constraints for the composition and does not address the problem of checking constraints for evolving business process models. In contrast, our approach explicitly considers the efficient validation against constraints. The improvement of the

efficiency is achieved by applying the concept of business process fragments. By defining the constraints on the level of these fragments, the number of required validations can be reduced significantly.

5 Conclusion and Future Work

In this paper, we presented an approach for the guided composition of business process models in a goal-driven way. We proposed an algorithm that identifies and clusters related operations to business process fragments and we describe how these fragments can be composed to a business process model by using model-checking techniques. In summary, we provide an approach that provides a high degree of automation but also considers involvement of domain experts and business analysts during the composition.

As future work, we aim for a tool implementation of the presented approach. Based on the existing workbench presented in [13,14] the composition algorithm will be implemented by integrating a model checker (e.g. NuSMV[1]). Using this tool support we will perform comprehensive case studies to evaluate the efficiency and applicability of our approach. A main aspect in the evaluation will be the investigation of the actual improvement of the model-checking scalability by using the business fragments.

References

1. Alrajeh, D., Kramer, J., Russo, A., Uchitel, S.: Learning operational requirements from goal models. In: Proc. of the 31st Int. Conf. on Software Engineering, ICSE 2009, pp. 265–275. IEEE Computer Society (2009)
2. Baresi, L., Pasquale, L.: Adaptive Goals for Self-Adaptive Service Compositions. In: 2010 IEEE International Conference on Web Services (ICWS), pp. 353–360. IEEE (2010)
3. Baresi, L., Pasquale, L.: Adaptation Goals for Adaptive Service-oriented Architectures. In: Relating Software Requirements and Architecture, pp. 161–181. Springer, Heidelberg (2011)
4. Bresciani, P., Giorgini, P., Giunchiglia, F., Mylopoulos, J., Perini, A.: TROPOS: An agent-oriented software development methodology. Autonomous Agents and Multi-Agent Systems (2004)
5. Dardenne, A., Fickas, S., van Lamsweerde, A.: Goal-directed concept acquisition in requirements elicitation. In: Proceedings of the 6th International Workshop on Software Specification and Design, IWSSD 1991, pp. 14–21. IEEE Computer Society Press (1991)
6. Dubois, E., Petit, M., Yu, E.: From Early to Late Formal Requirements: A Process-Control Case Study. In: Proc. of the 9th Int. Workshop on Software Specification and Design, p. 34. IEEE Computer Society (1998)
7. Gerth, C., Küster, J.M., Engels, G.: Language-Independent Change Management of Process Models. In: Schürr, A., Selic, B. (eds.) MODELS 2009. LNCS, vol. 5795, pp. 152–166. Springer, Heidelberg (2009)

[1] `http://nusmv.fbk.eu/`

8. Koliadis, G., Ghose, A.: Relating Business Process Models to Goal-Oriented Requirements Models in KAOS. In: Hoffmann, A., Kang, B.-H., Richards, D., Tsumoto, S. (eds.) PKAW 2006. LNCS (LNAI), vol. 4303, pp. 25–39. Springer, Heidelberg (2006)
9. Lapouchnian, A.: Goal-Oriented Requirements Engineering: An Overview of the Current Research. Requirements Engineering 8(3), 32 (2005)
10. Liaskos, S., McIlraith, S., Sohrabi, S., Mylopoulos, J.: Integrating preferences into goal models for requirements engineering. In: 2010 18th IEEE International Requirements Engineering Conference (RE), pp. 135–144 (2010)
11. Lo, A., Yu, E.: From Business Models to Service-Oriented Design: A Reference Catalog Approach. In: Parent, C., Schewe, K.-D., Storey, V.C., Thalheim, B. (eds.) ER 2007. LNCS, vol. 4801, pp. 87–101. Springer, Heidelberg (2007)
12. Martínez, A., Pastor, Ó., Mylopoulos, J., Giorgini, P.: From Early to Late Requirements: A Goal-Based Approach. In: Kolp, M., Henderson-Sellers, B., Mouratidis, H., Garcia, A., Ghose, A.K., Bresciani, P. (eds.) AOIS 2006. LNCS (LNAI), vol. 4898, pp. 123–142. Springer, Heidelberg (2008)
13. Nagel, B., Gerth, C., Post, J., Engels, G.: Ensuring Consistency among Business Goals and Business Process Models. In: Proceedings of 16th IEEE International Enterprise Distributed Object Computing Conference (EDOC), pp. 17–26 (2013)
14. Nagel, B., Gerth, C., Post, J., Engels, G.: Kaos4SOA - Extending KAOS Models with Temporal and Logical Dependencies. In: Proceedings of the CAiSE 2013 Forum at the 25th International Conference on Advanced Information Systems Engineering (CAiSE), pp. 9–16 (2013)
15. OMG. OMG Unified Modeling Language (OMG UML) Superstructure (2010)
16. OMG. Business Process Model and Notation (BPMN) (2011)
17. Oster, Z.J., Ali, S.A., Santhanam, G.R., Basu, S., Roop, P.S.: A service composition framework based on goal-oriented requirements engineering, model checking, and qualitative preference analysis. In: Liu, C., Ludwig, H., Toumani, F., Yu, Q. (eds.) ICSOC 2012. LNCS, vol. 7636, pp. 283–297. Springer, Heidelberg (2012)
18. Russell, N., Hofstede, A.H.M.T., Mulyar, N.: Workflow ControlFlow patterns: A revised view. Technical report (2006)
19. van Lamsweerde, A.: Goal-oriented requirements engineering: a guided tour. In: Proceedings of the Fifth IEEE International Symposium on Requirements Engineering, pp. 249–262 (2001)
20. van Lamsweerde, A.: From System Goals to Software Architecture. In: Bernardo, M., Inverardi, P. (eds.) SFM 2003. LNCS, vol. 2804, pp. 25–43. Springer, Heidelberg (2003)
21. Yu, E.S.-K.: Towards Modeling and Reasoning Support for Early-Phase Requirements Engineering. In: Proc. of the 3rd IEEE Int. Symposium on Requirements Engineering, pp. 226–235. IEEE Computer Society (1997)
22. Yu, Y., Lapouchnian, A., Liaskos, S., Mylopoulos, J., Leite, J.: From Goals to High-Variability Software Design. In: Foundations of Intelligent Systems, pp. 1–16 (2008)