Kaos4SOA - Extending KAOS Models with Temporal and Logical Dependencies

Benjamin Nagel, Christian Gerth, Jennifer Post, and Gregor Engels

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

Abstract. In requirements engineering for service-oriented systems, stakeholder objectives are described by goal models from which business process models are derived that define the required service compositions. A goal-based requirements specification should ensure completeness of the goals that need to be achieved as well as the consideration of temporal and logical dependencies between goals. Recent approaches lack the ability to elicitate and specify the stakeholders' knowledge about dependencies between goals that need to be considered in the derivation of business processes. We present Kaos4SOA, an extended goal notation for the explicit specification of goal dependencies and briefly describe how existing approaches can be extended to validate the consistency of business processes against these dependencies. The application of our approach is supported by the implemented Kaos4SOA modeling workbench.

Keywords: Requirements engineering, goal models, dependency modeling, business process models, service-oriented systems

1 Introduction

Following the paradigm of service-oriented architectures [4], today's information systems are developed by composing loosely coupled services into highly flexible service compositions. Requirements engineering for such systems differentiates between two different kinds of requirements. *Early requirements* specify stakeholder objectives on a high level of abstraction and are iteratively refined into operationalized goals termed operations [1]. Based on these operations, *late requirements* are derived that represent concrete requirements in terms of business process models.

The elicitation and analysis of early requirements using goal models has been addressed comprehensively by existing work [3, 5, 6]. Different notations, like KAOS [3] or Tropos/i* [2, 16], provide modeling languages and methodologies for the specification and refinement of goal models. The systematic derivation of business processes from goal models has been addressed by recent research [8, 11].

While providing a high degree of flexibility, these approaches do not constrain how (i.e. with respect to temporal and logical dependencies) operations shall be composed. Figure 1 provides an example. From the source goal model (G_1) three different process models (P_1, P_2, P_3) can be derived. Each of these process models considers all required operations specified in G_1, but obviously results in different execution orders of the operations. This ambiguity can lead to inconsistencies between the actual stakeholder objectives and derived business process models.

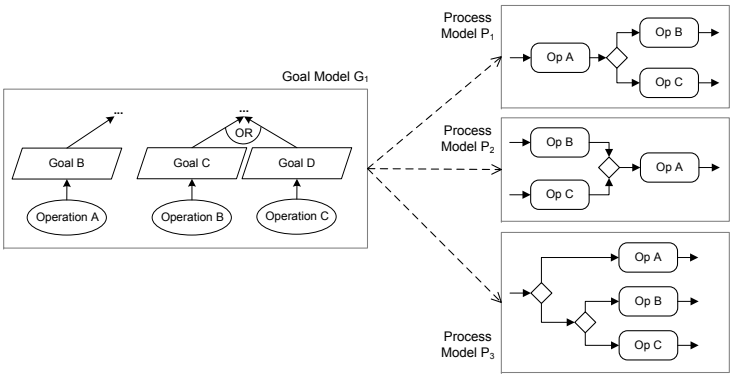

Fig. 1. Derivation of Different Process Models from a Goal Model

To address this issue, we propose to elicitate the stakeholders' knowledge about logical and temporal dependencies by specifying them explicitly in KAOS goal models. These dependencies are defined between goals on different levels of abstraction and constrain how the operations shall be composed.

The contribution of this paper consists of two parts. We introduce Kaos4SOA, an **extended KAOS goal notation** that supports the specification of temporal and logical dependencies between goals [13]. Further, we present a **tool support** in terms of a modeling workbench for the specification of Kaos4SOA goal models.

The remainder of the paper is structured as follows. In the next section the foundations of KAOS goal models are introduced. Section 3 presents the Kaos4SOA approach and discusses the extension of existing approaches to generate verifiable business process constraints from Kaos4SOA models. In Section 4, the implemented Kaos4SOA modeling workbench is described. Section 5 discusses related work and finally Section 6 concludes the paper and gives an overview about future work.

2 KAOS Goal Models

In this section, we introduce the KAOS modeling approach for specifying goal models [3]. KAOS provides a systematic approach to specify and refine goal models. Hereby, abstract high-level goals are iteratively decomposed to subgoals that are operationalized to operations, representing concrete functional requirements. These operations are used as input for the composition of process models.

An example for a KAOS goal model is illustrated in Figure 2. In order to achieve the main goal *Customer order processed*, four subgoals need to be fulfilled as specified by the AND-decomposition. The subgoals *Customer data recorded* and *Payment received* are further decomposed into subsubgoals that provide alternatives to achieve the subgoals (OR-decompositions).

The decomposition of *Customer data recorded* is a special type of decomposition. The circle connecting the goals *Account login data entered* and *Customer profile loaded* indicates an AND-decomposition which itself is part of an OR-decomposition. This means that the goal *Customer data recorded* is achieved by *New Customer account created* or by *Account login data entered* AND *Customer profile loaded*.

Using operationalization [1, 14], operations are identified that need to be executed in order to achieve the related goals. The operations are defined by input and output parameter as well as different conditions. The conditions are differentiated into pre- and postconditions in the domain and additional conditions. Two examples for the detailed definition of operations are depicted in Table 1.

Operation	Create account
Input	c:Customer, d:CustomerData
Output	a:CustomerAccount
DomPre	c.status = newCustomer
DomPost	c.status = registeredCustomer
ReqPre	c.Account = ""
ReqPost	c.Account = a

Operation	Enter credit card number
Input	c:Customer, cc:CreditCard
Output	c:Customer, cc:CreditCard
DomPre	cf.sent = false
DomPost	cf.sent = true
ReqPre	c.hasCard(cc) = true
ReqPost	cc.number = c.enteredNumber

Table 1. Exemplary Definition of Operations

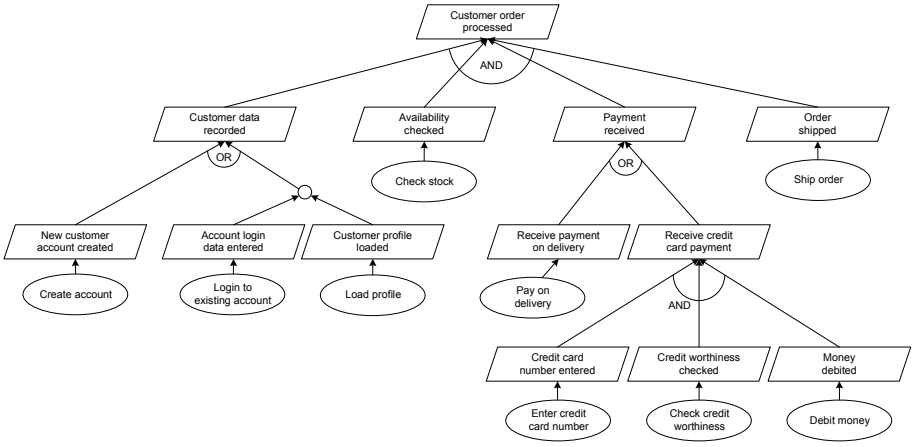

Fig. 2. *Customer Order Processing* Goal Model

In addition to the AND-/OR-refinement relationship between goals, KAOS also provides the ability to express conflicts between goals. For this purpose, the

ConflictsWith attribute [3] is used. The conflict relationship defines a dependency between two goals G_1 and G_2 defining that G_1, G_2 can not be achieved together. In our exemplary goal model the *ConflictsWith* relationship can be used to express that new customers are not allowed to pay on delivery by defining a conflict relation between the goals *New Customer account created* and *Receive payment on delivery*.

3 Kaos4SOA

In Section 1 we motivated the need for the elicitation and specification of goal dependencies to enable their consideration in the derivation of business processes. We propose Kaos4SOA, an extension of KAOS to express these dependencies in a goal model. In addition, we briefly describe the requirements for the automatic generation of verifiable constraints.

3.1 Goal Dependency Modeling

As described, KAOS goal models provide the ability to express relationships between goals (e.g. AND/OR-relation) and pre- and postconditions for atomic operations. We have analyzed existing research presented in [9, 12, 15] and identified two extensions for a more precise definition of temporal and logical dependencies between goals.

First, it is desirable to consider the order in which goals need to be achieved. We argue that even on the abstract level of goals, a requirements engineer is already able to decide about temporal dependencies between goals. In addition, we aim for a more precise definition of the logical OR-refinement. In the existing KAOS notation the OR-statement is interpreted as an inclusive-OR, i.e. OR(G_1 , G_2) $\longrightarrow$ $G_1 \vee G_2 \vee (G_1 \wedge G_2)$. Currently, it is not possible to define conditions to specify when G_1 or G_2 need to be achieved. For this purpose we introduce annotation capabilities for branches to define conditional branches. The proposed extensions of KAOS to address the required dependencies are described in the following.

Order Constraints

KAOS goal models do not provide the ability to express dependencies among goals regarding the order in which the goals need to be achieved. For this purpose, we extend the annotation capabilities of KAOS with the additional `Order` element. Two different types of orders are expressible by the proposed extension.

Loose Order describes that a goal G_2 needs to be achieved some time later than goal G_1. It does not matter, if other goals are achieved meanwhile. For expressing the loose order dependency the attributes `Order.Predecessor` and `Order.Successor` are introduced. Figure 3 (a) shows a loose order example: *Before the ordered goods are shipped, the goods' availability has to be checked.*

Strict Order expresses that the achievement of goal G_1 is directly followed by the achievement of goal G_2 without other goals achieved in-between. This kind of sequential execution is described by the annotations `Order.StrictPredecessor` and `Order.StrictSuccessor`. As an example the strict order *The credit worthiness needs to be checked directly before the credit card is debited* is shown in Figure 3 (b).

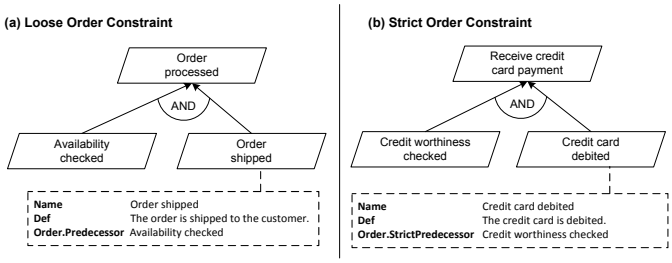

Fig. 3. Goal Attributes for Order Constraints

Conditional Branch

In some cases not only the achievement of single goals is important to be constrained by a condition, instead the achievement of a whole set of goals depends on a condition. Often this can be seen for alternatives, which are all suitable to achieve a desired result, but not each goal sequence is appropriate in each case. An example for conditional branches is depicted in Figure 4. The goal *Customer data recorded* is decomposed by an OR-refinement. Which alternative subgoal shall be chosen under which condition, can be differentiated by the new attribute condition which is defined for both links. In this example, the conditional branch is used to define the following policies.

> *If the customer is ordering for the first time (is a new customer), he needs to create a new account for recording the customer data.*
>
> *If the customer already has an account, he needs to login AND the customer profile needs to be loaded to record the customer data.*

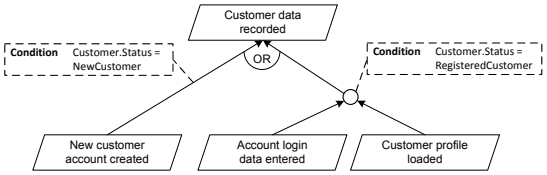

Fig. 4. Goal Attributes for Conditional Branches

3.2 Towards the Goal-driven Generation of Business Process Constraints

To preserve the consistency between goal models and business process models it is required to evaluate that the derived business processes do not violate the

constraints indicated by the temporal and logical dependencies in the Kaos4SOA model.

Kaos4SOA facilitates additional goal annotations expressing the dependencies, but does not provide formal constraints. To evaluate the consistency of a business process regarding these dependencies, formalized constraints need to be generated from the Kaos4SOA goal model. Since the dependencies are defined among goals on different levels of abstraction they need to refined iteratively to the operational level. From the refined dependencies, business process quality constraints are generated that express the defined dependencies by concrete, verifiable constraints. [7] provides a language to express constraints for business process models, but do not provide a method for deriving them from goal models. To apply these constraints existing approaches [8, 11] for the derivation of business processes need to be extended by model checking techniques for evaluating the business process quality constraints.

4 Implementation

The applicability of our approach depends on the ability to create and edit complex goal models and the dependencies among the goals in an efficient way. To provide a tool support for our modeling approach we implemented the *Kaos4SOA Workbench*. It provides modeling capabilities for standard KAOS modeling elements as well as for the extended Kaos4SOA goal dependencies.

The Kaos4SOA Workbench is realized as a plugin for the Eclipse development environment [1]. A metamodel for the extended KAOS notation is defined by EMF Ecore models using the Eclipse modeling framework (EMF). The concrete syntax for the extended goal models is modeled by the Graphical Modeling Framework (GMF). Based on the defined GMF models a graphical editor is generated, that enables the creation and editing of Kaos4SOA models. A screenshot of the modeling UI is depicted in Figure 5.

5 Related Work

The i* framework [16] is a goal-oriented requirements engineering approach from the domain of agent-oriented software. As well as KAOS it is designed for the early phase of requirements analysis, which aims to analyse and model stakeholder interests. In i* different kinds of relationships can be defined. Dependency links are used to define relations between actors. For the specification of relations between goals i* distinguishes *Task-Decomposition-Links*, *Means-End-Links* and *Contribution Links*. Hereby, the decomposition of goals to tasks, traceability between goals and tasks and the contribution of tasks to soft goals can be defined.

Tropos [2, 6, 10] is a software development framework which spans all phases of software development from early requirements specification to implementation. Tropos extends the relationships provided by *i** by providing differentiated AND-/OR- Decompositions of goals into subgoals. Furthermore, Tropos

[1] http://www.eclipse.org/

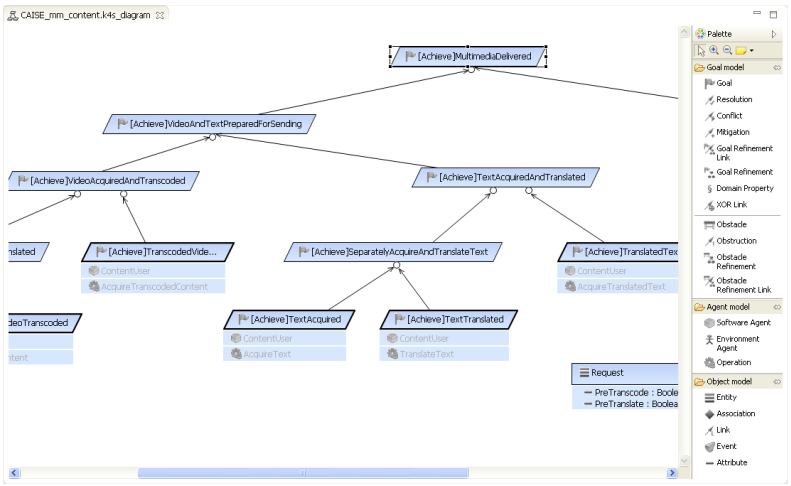

Fig. 5. Screenshot of the Kaos4SOA Workbench

allows the application of *Contribution Links* that enable to identify *Goals*, that contribute positively or negatively to the fulfillment of other *Goals*.

In summary, existing goal oriented approaches provide concepts for expressing and modeling dependencies among goals. Nonetheless, they lack the ability to define concrete logical dependencies for branches and do not support any kind of temporal dependencies.

In [1] a framework for the requirements operationalization from goal models is presented. This approach describes the iterative operationalization and formalization of goals for preserving completeness and correctness with respect to possible goal violations but does not consider dependencies among goals.

6 Conclusion and Future Work

This paper presents Kaos4SOA, an extension of the well-known KAOS approach to elicitate the stakeholders' knowledge about dependencies between goals. Based on an analysis of existing research we identified required dependencies and the shortcomings of KAOS for expressing these dependencies. We introduce and define new notation elements for logical and temporal dependencies. Finally, we present a tool support that supports the creation of extended KAOS models in a graphical modeling workbench.

The future work of our research is twofold. First we aim to extend our approach with the generation of verifiable business process quality constraints, that can be used to evaluate the consistency between goals and derived business processes. Our future work also includes the extension of the Kaos4SOA Workbench with a model checker which allows the integrated, automated verification of business process models against defined constraints.

References

1. D. Alrajeh, J. Kramer, A. Russo, and S. Uchitel. Learning operational requirements from goal models. In *Proceedings of the 31st International Conference on Software Engineering*, ICSE '09, pages 265–275. IEEE Computer Society, 2009.
2. P. Bresciani, P. Giorgini, F. Giunchiglia, J. Mylopoulos, and A. Perini. TROPOS: An agent-oriented software development methodology. *Autonomous Agents and Multi-Agent Systems*, 2004.
3. A. Dardenne, A. van Lamsweerde, and S. Fickas. Goal-directed requirements acquisition. In *Selected Papers of the Sixth International Workshop on Software Specification and Design*, pages 3–50. Elsevier Science Publishers B. V., 1993.
4. T. Erl. *Service-Oriented Architecture: Concepts, Technology, and Design*. Prentice Hall PTR, 2005.
5. A. Fuxman, L. Liu, J. Mylopoulos, M. Pistore, M. Roveri, and P. Traverso. Specifying and analyzing early requirements in tropos. *Requirements Engineering*, 9(2):132–150, 2004.
6. P. Giorgini, M. Kolp, J. Mylopoulos, and M. Pistore. The tropos methodology: An overview. In *Methodologies and Software Engineering for Agent Systems*. Kluwer Academic Press, 2003.
7. L. Khaluf, C. Gerth, and G. Engels. Pattern-based modeling and formalizing of business process quality constraints. In *Proceedings of the 23rd Int. Conf. on Advanced Information Systems Engineering*, CAiSE'11, pages 521–535, Berlin, Heidelberg, 2011. Springer-Verlag.
8. A. Lo and E. Yu. From business models to service-oriented design: A reference catalog approach. In C. Parent, K.-D. Schewe, V. C. Storey, and B. Thalheim, editors, *Conceptual Modeling - ER 2007*, volume 4801, pages 87–101. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.
9. R. Lu, S. Sadiq, G. Governatori, and X. Yang. Defining Adaptation Constraints for Business Process Variants. In *Business Information Systems*, volume 21 of *Lecture Notes in Business Information Processing*, pages 145–156. Springer Berlin Heidelberg, 2009.
10. J. Mylopoulos and J. Castro. Tropos: A framework for requirements-driven software development. In S. Brinkkemper, E. Lindencrona, and A. Sølvberg, editors, *Information Systems Engineering: State of the Art and Research Themes*. Springer, 2000.
11. B. Pernici. Methodologies for design of service-based systems. In *Intentional Perspectives on Information Systems Engineering*, pages 307–318. Springer, Berlin, Heidelberg, 2010.
12. M. Pesic, H. Schonenberg, and W. M. P. van der Aalst. Declare: Full support for loosely-structured processes. In *Proceedings of the 11th IEEE Int. Enterprise Distributed Object Computing Conf.*, page 287. IEEE Computer Society, 2007.
13. J. Post. Goal-driven refinement of quality constraints for adaptive business processes. Master's thesis, University of Paderborn, 2012.
14. A. Van Lamsweerde. *Requirements Engineering: From System Goals to UML Models to Software Specifications*. John Wiley, 2009.
15. J. Wainer and F. de Lima Bezerra. Constraint-based flexible workflows. In J. Favela and D. Decouchant, editors, *CRIWG*, volume 2806 of *Lecture Notes in Computer Science*, pages 151–158. Springer, 2003.
16. E. S.-K. Yu. Towards Modeling and Reasoning Support for Early-Phase Requirements Engineering. In *Proceedings of the 3rd IEEE International Symposium on Requirements Engineering*, pages 226–235. IEEE Computer Society, 1997.