Goal and Variability Modeling for Service-oriented Systems: Integrating *i** with Decision Models

P. Grünbacher[1] D. Dhungana[1] N. Seyff[2] M. Quintus[2] R. Clotet[3] X. Franch[3] L. López[3] J. Marco[3]

Christian Doppler Lab for Automated Software Eng.[1]
Johannes Kepler University
4040 Linz, Austria

Systems Engineering and Automation[2]
Johannes Kepler University
4040 Linz, Austria

Universitat Politècnica de Catalunya (UPC)[3]
Barcelona, Spain

Paul.Gruenbacher@jku.at

Abstract. Variability modeling and service-orientation are important approaches that address both the flexibility and adaptability required by stakeholders of today's software systems. Goal-oriented approaches for modeling service-oriented systems and their variability in an integrated manner are needed to address the needs of heterogeneous stakeholders and to develop and evolve these systems. In this paper we propose an approach that complements the *i** modeling framework with decision models from orthogonal variability modeling. We illustrate the approach using an example and present options for tool support.

1 Introduction

Stakeholders of today's software-intensive systems demand flexibility and adaptability to allow rapid system evolution made necessary by new and changing requirements. Variability modeling and service-orientation are promising with respect to both flexibility and adaptability. Variability modeling is an approach fostering software reuse and rapid customization of systems [8][10]. Service-orientation is visible in buzzwords such as service-oriented computing, service-oriented architectures, or service-oriented software engineering and is promoted by a number of emerging standards for service-oriented development. Recently, researchers have started to explore the integration of service-oriented systems and variability modeling. Variability modeling is increasingly seen as a mechanism to support run-time evolution and dynamism in different domains and to design, analyze, monitor, and adapt service-oriented systems [7]. At the same time the modeling framework *i** [11] is gaining popularity to model service-oriented and agent-based systems [9] and researchers are seeking new ways to enhance it with variability modeling capabilities [6].

Pursuing similar goals we have been using *i** to model a service-oriented multi-stakeholder distributed system (MSDS) in the travel domain to validate its usefulness

in this context [3]. Despite the power and expressiveness of *i** we experienced some deficiencies when modeling variability in particular when specifying the needs of heterogeneous stakeholders in the MSDS or when investigating the modeling needs of service provides and service integrators. As a result we started investigating the dependencies of goal modeling and variability modeling. In this paper we discuss an initial approach integrating orthogonal variability modeling techniques into *i**. We illustrate the approach using examples and discuss tentative tool support based on our existing work on meta-tools for variability modeling. Our approach is based on our integration framework [3] as well as our earlier work on the use of *i** [5] and variability modeling [4].

2 Modeling the Variability of Service-oriented Systems with *i**

Modeling service-oriented systems requires an understanding of stakeholder goals and goal variability. *i** is an established framework for goal modeling [11] which is goal-oriented as well as actor-oriented and supports the assignment of responsibilities to system actors to express high-level actor requirements. There are two types of models in *i**: Strategic Dependency (SD) models define actors, their relationships (e.g., specialization and composition) and how they depend on each other. Strategic Rationale (SR) models state the main goals of these actors and their decomposition using some refinement constructs. Together the SD and SR models provide a comprehensive system overview. *i** supports recording the rationale behind requirements and decomposing elicited requirements at the required level of detail. At the requirements level actors are mainly used to represent stakeholders' needs, while at the architecture levels they can be used to model services: For instance, Franch and Maiden have explored the use of *i** to model architectures using roles and agents [5]. A similar approach has been proposed by Penserini *et al.* in [9]. *i** supports traceability from high-level actor goals to concrete services in the running system and vice versa. It has been shown that high-level stakeholder goals tend to be more stable than underlying requirements and selected software solutions. Linking services to high-level stakeholder goals modeled in *i** thus increases system stability and adaptability by guiding the replacement of malfunctioning services with services also fulfilling essential stakeholder goals. It also facilitates the identification of affected stakeholders [3].

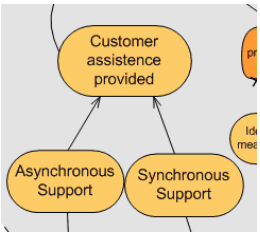

Fig 1. Modeling Goal Variability in *i**: Customer assistance can be either provided using asynchronous or synchronous support.

We are currently exploring the benefits and limitations of *i** for developing and evolving service-oriented systems. A key experience is that variability essential for modeling service-oriented systems at different levels of abstraction. Figures 1 and 2 show partial *i** examples of external and internal variability.

Our framework presented in [3] defines four different modeling layers for service-oriented systems: stakeholder needs, architecture prescription, solution architecture, and open system. Figure 2 shows a concrete modeling example on the architecture layers. For instance, the architecture prescription layer may define the actor "Travel services provider" (expressed as role in *i**) for undertaking the "Book hotel" and "Book flight" system tasks. At the lower solution architecture layer several services cover the role "Travel services provider": The services "Amadeus" and "Schubert" are modeled as *i** agents since they are real-world entities. The open system instance layer describes a running system. If the service "Amadeus" is chosen one may choose the "Spanish Amadeus Server" service hosted on a Spanish site or the "Austrian Amadeus Server" hosted on an Austrian site. Again these services are modeled as *i** agents, related to "Amadeus" by using the *instance* relationship in *i**.

Fig 2. Modelling Service Types and Services in *i**.

The example shows the capabilities of *i** for modeling service-oriented systems at different levels of abstraction. The language can be used to model high-level concerns such as stakeholder goals, architecture-level aspects, and even the configuration of the open system based on service instances [3]. The flexibility of *i** was also pointed out by other authors [1]. Traceability is a major benefit: The *contributes* relationship in *i** allows establishing traceability between stakeholder goals, service types, selected services, and service instances.

The examples, however, also show some limitations of modeling variability in *i**. The expressiveness and formality is insufficient compared to existing variability modeling approaches. There are no language constructs to capture more formal aspects required in variability models such as constraints (e.g., between services), conditions under which services become active or inactive, selector types, or cardinalities [10]. The variability modeling capabilities of *i** should therefore be enhanced.

3 Using Variability Modeling with *i**

We propose an approach based on our framework for multi-stakeholder distributed systems [3] and our earlier work on the use of *i** [5] and meta-tools for variability modeling [4]. A fundamental approach in variability modeling is to complement existing models and artifacts with variability information rather than using specific notations or languages. Our work is influenced by a decision-oriented approach proposed by Schmid and John [10] that supports orthogonal variability modeling for arbitrary artifacts independent from a specific notation. The benefits of such approaches are the flexibility gained and traceability established by using one variability mechanisms for different artifacts at the requirements, design, architecture, implementation, application, and runtime level. We propose to complement *i** with orthogonal variability modeling techniques. Such an approach requires:

- The development of a *decision model* describing the variability of the system and dependencies between variabilities [4].
- An *asset model* describing the system elements and their dependencies [4]. In the domain of service-oriented systems the elements include service types, services, and service instances together with their dependencies (e.g. a payment service might rely on a transaction service).
- The annotation of *i** models with rules referring to the decision model to model inclusion conditions for services and the dependencies among assets and decisions.
- A mechanism to prune *i** models based decisions taken at design-time or runtime to generate/update system configurations on the fly (e.g., by adding/removing/updating services).

We envisage an *i** model to hold a snapshot of a service-oriented system at a certain point in time that can be adapted based on decisions taken by stakeholders by considering all assets and their dependencies. In the product line terminology such a snapshot is based on the domain-level, ”product-line“ version of the *i** model. Obviously, beyond the *i** model, the runtime configuration requires the generation of additional information for configuration, i.e., concrete values of decision variables that inform system configuration (see Figure 3).

Decision Variable	*Question*	*Selection Type*	*Cardi-nality*	*Link to i* element*
Type of customer assistance	What kind of customer assistance do you need?	Set {Synchronous Support, Asynchro-nous Support}	1:2	Customer assistance provided
Degree of customer assistance	How many hours per day should the hotline be available?	Value [0..24]	1	Customer assistance provided
Travel Service Provider	Which is your preferred travel service?	Set (Ama-deus, Schu-bert)	1	Travel Service Provider

Fig 3. Partial decision model.

4 Adopting a Variability Modeling Meta-Tool

We are aiming to provide tool support for the discussed approach and have been tailoring the orthogonal variability modelling meta-tool DecisionKing to our problem context [4]. DecisionKing allows the definition of meta models for arbitrary asset types to create a customized variability modeling tool. The tailoring of a custom-built variability modeling tool in DecisionKing covers (i) the definition of a domain-specific meta-model and (ii) the development of domain-specific plug-ins:

Definition of the meta-model for service-oriented variability modeling. This step covers the identification of the relevant asset and dependency types. We identified the asset types goal, service type, service, and service instances: A *goal* of a stakeholders maps to an actor goal in *i**. Different *services types* contribute to fulfilling these goals (e.g., "Travel services provider"). Available services realizing a service type are modeled as a *service* (e.g., "Amadeus"). Finally, available runtime implementations of services can be modeled as service instances (e.g., "Spanish Amadeus Server"). We also identified two kinds of relationships between the assets: The *requires* relationship is used whenever the selection of a certain assets leads to the selection of another asset. This can be a result of logical dependencies between goals, conceptual relationships between service types, relationships between services, or functional dependencies between service instances. The *contributesTo* relationship is used to capture structural dependencies between assets of different levels. Service instances for example contribute to services. Services contribute to service types which contribute to goals. It is however also possible that goals are split up into sub-goals. Such compositional relationships between goals can also be modeled using the *contributesTo* relationship.

Development of plugins. DecisionKing's capabilities can be extended by plugging-in domain-specific functionality [4]. Using this mechanism we are developing a link between DecisionKing and the *i** modeling tools REDEPEND using an XML-based interchange language for our tool suite.

5 Open Issues

In this paper we proposed an initial approach to complement *i** with an orthogonal variability modeling technique. There are several open issues needing attention:

We need to extend the *i** language in order to include variability information. On the one hand, we need to provide complete formal semantics for the *is_a* inheritance *i** mechanism, which is currently only defined at the actor level. On the other hand, we need to provide a formal syntax for modeling taken decisions (e.g., which services are chosen) and variation points. We are considering the use of the *i** routine concept for reflection decisions taken in the model.

We also need to complete tool integration to improve traceability between *i** and variability models. For this purpose we will link the *i** meta-model and the variability meta-model and exchange information between models using an XML interchange definition language currently under definition.

References

[1] J. Castro, M. Kolp, J. Mylopoulos. "Towards Requirements-Driven Information Systems Engineering: The Tropos Project". *Information Systems*, vol. 27, 2002.

[2] L. Chung, B.A. Nixon, E. Yu, J. Mylopoulos, Non-functional requirements in software engineering, Kluwer Academic Publishers, 2000.

[3] R. Clotet, X. Franch, P. Grünbacher, L. López, J. Marco, M. Quintus, N. Seyff: "Requirements Modelling for Multi-Stakeholder Distributed Systems: Challenges and Techniques". RCIS'07: 1st IEEE Int. Conf. on Research Challenges in IS, Ouarzazate, 2007

[4] D. Dhungana, P. Grünbacher, and R. Rabiser, "DecisionKing: A Flexible and Extensible Tool for Integrated Variability Modeling.", 1st International Workshop on Variability Modelling of Software-intensive Systems, Limerick, Ireland, 2007.

[5] X. Franch, N.A.M. Maiden. "Modeling Component Dependencies to Inform their Selection". In *Proceedings 2nd International Conference on COTS-Based Software Systems* (ICCBSS), Lecture Notes on Computer Science 2580, Springer,2003.

[6] S. Liaskos, Y. Yu, E. Yu, J. Mylopoulos. "On Goal-based Variability Acquisition and Analysis". *Proc. 14th IEEE Int'l Requirements Engineering Conference* (RE'06) (Sept 11-15, 2006). IEEE Computer Society

[7] J. Peña, M.G. Hinchey, A. Ruiz-Cortés. "Multi-agent system product lines: challenges and benefits". *Communications of the ACM*, vol. 49, n. 12 (Dec. 2006), 82-84.

[8] K. Pohl, G. Böckle, and F. J. van der Linden, Software Product Line Engineering: Foundations, Principles, and Techniques: Springer, 2005.

[9] L. Penserini, A. Perini, A. Susi, J. Mylopoulos. "From Stakeholder Needs to Service Requirements". *Proceedings of the 2nd International Workshop on Service-Oriented Computing: Challenges on Engineering Requirements* (SOCCER), 2006.

[10] K. Schmid and I. John, "A Customizable Approach to Full-Life Cycle Variability Management". *Journal of the Science of Computer Programming*, Special Issue on Variability Management, vol. 53(3), pp. 259-284, 2004.

[11] E. Yu. Modeling *Strategic Relationships for Process Reengineering*, PhD Thesis, Toronto, 1995.