Dynamic Requirements Specification for Adaptable and Open Service Systems

Ivan J. Jureta, Stéphane Faulkner, Philippe Thiran
Information Management Research Unit (IMRU), University of Namur, Belgium
iju@info.fundp.ac.be, {sfaulkne, pthiran}@fundp.ac.be

Abstract

The Dynamic Requirements Adaptation Method (DRAM) is suggested to assist existing RE methodologies in updating requirements specifications at runtime for adaptable and open service-oriented systems. Updates are needed because an adaptable and open system continually changes how and to what extent initial requirements are achieved.

1. Problem

In *service-orientation*, individual services are "self-describing, open components that support rapid, low-cost composition of distributed applications" [5]. An *open* service-oriented system may involve a large pool of distinct and competing services orignating from various service providers. To be *adaptable*, such a system coordinates service provision by dynamically selecting the participating services according to multiple quality criteria, so that the users continually receive optimal results. The proposal outlined in the remainder resulted from the difficulties we encountered in engineering requirements for an adaptable and open service-oriented system (AOSS), called TravelWeb here, which allows users to search for and book flights, trains, hotels, rental cars, or any combination thereof through a web interface. Services which perform search and booking originate from the various service providers that either represent the various airlines and other companies, so that TravelWeb aggregates and provides an interface to the user when moving through the offerings of the various providers. Each provider can decide what options to offer to the user: e.g., in addition to the basics, such as booking a seat on an airplane, some airlines may ask for seating, entertainment, and food preferences, while others may further personalize the offering through additional options. We present the architecture and algorithms for TravelWeb elsewhere [4]. Following any established RE methodology (e.g., [2, 1]), a requirements specification for TravelWeb would be constructed by moving from abstract stakeholder expectations towards a detailed specification of the entire system's behavior. This is not feasible because: (i) services that may participate are unknown at development time; (ii) an adaptable system is usually needed when the environment is unpredictable—we cannot anticipate all possible operating conditions, and environment and system behaviors; (iii) services that will participate in TravelWeb are not developed by same service providers. We argue that, when performing the RE of AOSS: (a) the initial specification ought to be updated at runtime; (b) the initial specification need not be extensive; and, (c) a separation is to be made in the RE for AOSS according to the various concerns of the developers of the services and of the AOSS.

The extended version of this paper[1] explains what parts of the initial specification to update and how to update them at runtime using the Dynamic Requirements Adaptation Method (DRAM). Here, we outline DRAM.

2 Outline of DRAM

DRAM is not a standalone RE methodology; it integrates concepts and techniques for defining mappings (called justified correspondences in DRAM) between fragments of the requirements specification produced by an existing RE methodology and service requests. Correspondence ensures that the stakeholders' expectations are translated into constraints and quality parameters understood by the AOSS. Correspondence between requests and requirements allows the requirements specification to be updated to reflect runtime changes in the system due to adaptability and openness. A *justified correspondence* $\phi \triangleq \psi$ exists between a fragment ϕ of a usual requirements specification and a service request (involving service process descriptions, quality parameters, and/on user preferences) ψ iff there is a justification $\langle P, \phi \triangleq \psi \rangle$, i.e., $\phi \triangleq \psi$ iff $\exists \langle P, \phi \triangleq \psi \rangle$. A *justification* is constructed by applying a justification process (see, [3] and the extended version of the present paper). Justified correspondences together with the requirements and service requests constitute a dynamic requirements specification. A justified correspondence is used as an update rule: when-

[1] Available at: http://www.jureta.com/papers/DRAM-RE07.pdf

1090-705X/07 $25.00 © 2007 IEEE
DOI 10.1109/RE.2007.28

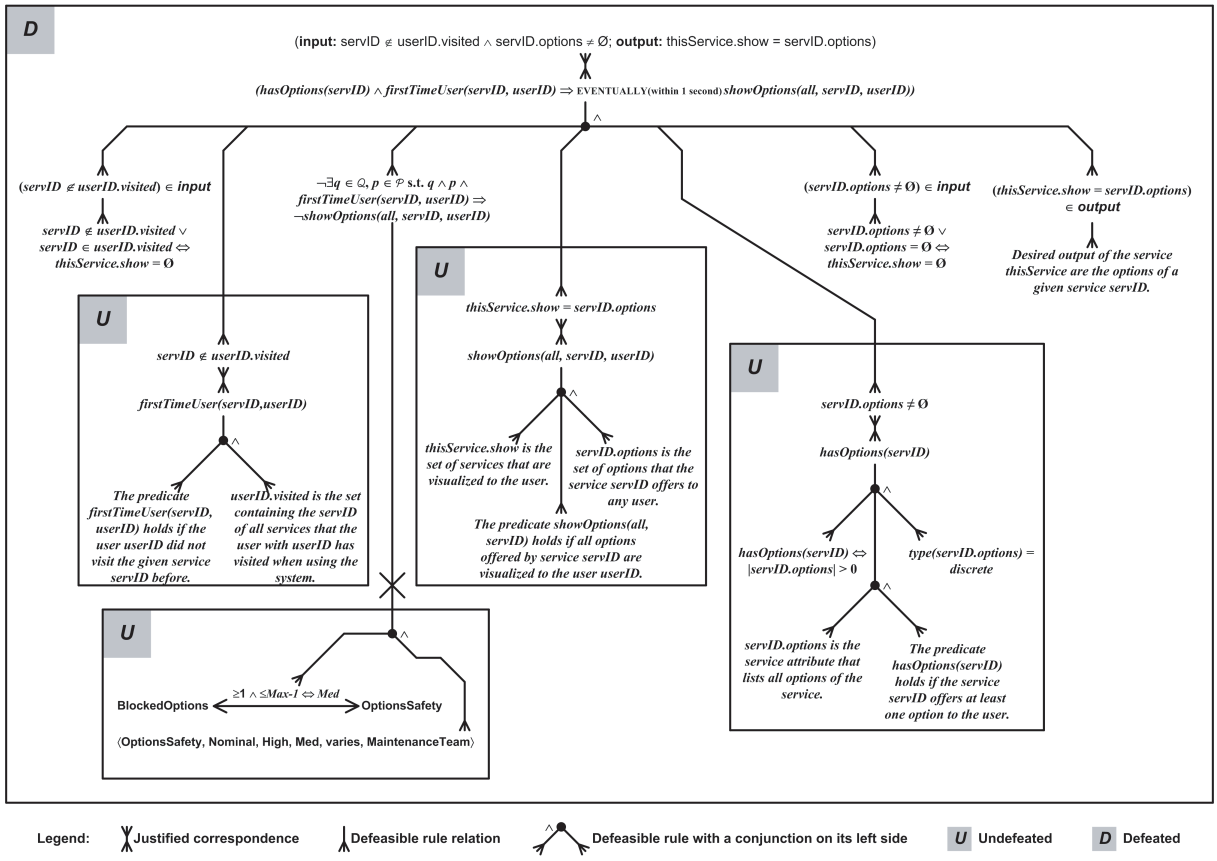

Figure 1: Example output of the construction of a justified correspondence in DRAM.

ever we encounter information appearing in a correspondence, we check if the correspondence is maintained; if not, we update requirements or requests to maintain the correspondence. DRAM uses the following process to build the dynamic requirements specification:
(1) Starting from the requirements, select a fragment thereof that has not been converted into a service request fragment.
(2) Determine the service request information that can be extracted from the given requirements fragment.
(3) Write down the obtained service request information, along with arguments and justifications used in mapping the requirements fragment to the request fragment. Each justified correspondence obtained by performing the step (2) above is written down as an update rule.
(4) Verify that the new arguments used in the justified correspondence do not defeat available justifications; revise the old justifications if needed.

As an example (detailed in the extended version), consider the correspondence used in TravelWeb, and shown in the top of Fig.1. There, we justified the correspondence using various information from requirements and requests, along with informal accounts to simplify presentation. Observe that the correspondence is defeated (labeled "D"), that is, is not justified because we have evidence (given in the branches, below the correspondence) that the correspondence does not hold. To make it justified, we would need to defeat the undefeated argument (bottom-right of the figure) by providing evidence to the contrary of that argument.

Automation of the various techniques in DRAM is the focus of current work, namely through patters of justifications for justified correspondences. For further details on DRAM and the illustrating examples, the reader is referred to the extended version of this paper. It is available online, the url is cited on the previous page.

References

[1] J. Castro, M. Kolp, J. Mylopoulos. Towards requirements-driven information systems engineering: the Tropos project. *Information Systems*, 27(6), 2002.

[2] A. Dardenne, A. van Lamsweerde, S. Fickas. Goal-directed requirements acquisition. *Sci. Comp. Progr.*, 20, 1993.

[3] I. J. Jureta, S. Faulkner, P.-Y. Schobbens. Justifying Goal Models. *Proc. Int. Conf. Req. Eng. (RE'06)*, 2006.

[4] I. J. Jureta, S. Faulkner, Y. Achbany, M. Saerens. Dynamic Web Service Composition within a Service-Oriented Architecture. *Proc. Int. Conf. Web Services (ICWS'07)*. To appear.

[5] M. P. Papazoglou, D. Georgakopoulos. Service-Oriented Computing. *Comm. ACM*, 46(10), 2003.