Generating and Optimizing Graphical User Interfaces for Semantic Service Compositions

Eran Toch[1], Iris Reinhartz-Berger[2], Avigdor Gal[1], and Dov Dori[1]

[1] Faculty of Industrial Engineering and Management
Technion - Israel Institute of Technology
erant@tx.technion.ac.il, dori@ie.technion.ac.il, avigal@ie.technion.ac.il
[2] Department of Information Systems
University of Haifa
iris@mis.hevra.haifa.ac.il

1 Background

Semantic Web service composition is a discovery process in which a given set of requirements are fulfilled by dynamically locating and assembling semantically annotated services [5, 6]. Semantic annotation of Web services is a set of models that describe its properties (e.g., inputs, outputs, process), in a formal language such as OWL-S [2]. These models provide an unambiguous description of service properties by relating them to concepts belonging to Web ontologies. While dynamic service composition provides a flexible applications which can change according to service failures and other factors, it raises several questions regarding the way users interact with the generated applications. Specifically, it raises a challenge for *usability*, which is defined as the effectiveness, efficiency and satisfaction in which users perform tasks using a given system [1].

In traditional software development processes, the user interface is manually designed, implemented and tested in order to ensure its usability. In contrast, in dynamically composed applications, the functionality is not established during the design of the system. Therefore, the user interface cannot be designed, let alone tested, for usability. The conclusion is that the user interface should be generated dynamically as well, reflecting the temporal functionality of the application.

The field of automatic generation of user interfaces attempts to formally define the elements of user interfaces, including presentation and interaction [4]. However, they do not deal with usability optimization as they presume the models are designed with usability in mind. Therefore, this approach will not suffice for dynamic compositions, as these compositions are not optimized for usability. The contribution of our work is in suggesting a method for optimizing the usability of dynamically composed applications, using formal methods.

2 Optimization by Model Transformation

In order to address the problem of usability in dynamically-created compositions, we present **Liquid-Interface**, a framework for user-interface generation and

optimization[1]. The framework generates Web-based user interfaces by analyzing the semantic properties of compositions defined in OWL-S [2]. Liquid-Interface applies an optimization technique to improve the usability of the user-interface, and specifically the way users navigate the applciation.

Liquid-Interface derives semantic concepts from the service description, visually expressing them using interface widgets. For example, concepts that express dates are displayed using a calendar, and concepts that have a bounded set of values (e.g. countries or currencies) are displayed as combo-box lists. Other semantic characteristics are expressed using user interface elements, including cardinality, concept generalization, multi-lingual concepts, and input validity checks.

Navigation optimization modifies the process execution order of the original OWL-S [2] model according to a set of user interaction design patterns [3]. We created a taxonomy of user interaction design patterns, which are relevant to navigation, and expressing them using formal mathematical models. For example, the *Flat and Narrow Tree* design pattern defines optimal measures to link distribution between the pages. The patterns are used in order to assign a *usability score* to a configuration of the application's navigational properties. These properties include the number of links between processes, the number of fields within a process, and so fourth. The optimization process searches for a configuration with an optimal accumulative score. Heuristic methods are used in order to bound the search space. The Liquid-Interface framework exhibit an open architecture that allows new design patterns to be defined and added dynamically to the optimization process. Preliminary results prove the feasibility of our approach, and reveal interesting relations between design patterns, including patterns that contradict, or reinforce, each other.

References

1. ISO 9241-11. Ergonomic requirements for office work with visual display terminals, part 11: Guidance on usability, 1998.
2. A. Ankolekar, M. Burstein, J. R. Hobbs, O. Lassila, D. L. Martin, S. A. McIlraith, S. Narayanan, M. Paolucci, T. Payne, K. Sycara, and H. Zeng. Daml-s: Semantic markup for web services. In *Proceedings of the International Semantic Web Workshop (SWWS)*, pages 411–430, July 13 2001.
3. Jan Borchers. *A Pattern Approach to Interaction Design.* John Wiley & Sons, Inc., 2001. Foreword By-Frank Buschmann.
4. Deepali Khushraj and Ora Lassila. Ontological approach to generating personalized user interfaces for web services. In *International Semantic Web Conference*, pages 916–927, 2005.
5. Matthias Klusch. Semantic service coordination. In H. Schuldt M. Schumacher, H. Helin, editor, *CASCOM - Intelligent Service Coordination in the Semantic Web*, chapter 4. Birkhaeuser Verlag, Springer, 2008.
6. Eran Toch, Avigdor Gal, Iris Reinhartz-Berger, and Dov Dori. A semantic approach to approximate service retrieval. *ACM Trans. Inter. Tech.*, 8(1):2, 2007.

[1] The Framework can be used and downloaded at: http://dori.technion.ac.il/liquidInterface