Liquid-Interface: Automatically Generating and Optimizing User-Interfaces for Dynamic Compositions

Eran Toch*, Iris Reinhartz-Berger†, and Dov Dori*
*Faculty of Industrial Engineering and Management
Technion - Israel Institute of Technology
erant@tx.technion.ac.il, dori@ie.technion.ac.il
†Department of Information Systems
University of Haifa
iris@mis.hevra.haifa.ac.il

I. Introduction

Dynamic composition is a new way for creating software applications. Rather than manual coding the new application, the application is generated automatically by reusing existing software services according to the user's requirements [3], [7]. The method has several advantages: it answers an instantaneous request of the user, and the application is flexible: the application can change instantly, reacting to changes in the underlying services (e.g. failures, price change, quality of service etc). While dynamic composition promises an exciting vision for software development, it raises several questions regarding the way users interact with the generated application. 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 derived from the requirements and desired functionality of the application model. It can be carefully designed and tested in order to insure its usability. In contrast, in dynamically composed applications, the functionality is not set during the design of the system. Therefore, it 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 temporary functionality of the application.

The field of automatic generation of user-interfaces attempts to formally define the elements of user-interface, including presentation and interaction, and use the formal model in order to generate user-interfaces [5], [4], [6]. While model-based user-interfaces provide the foundations for automatic generation of user-interfaces, they do not deal with usability optimization as they presume the models are already usable. However, this approach will not suffice for dynamic compositions, as these compositions are not optimized for usability.

II. Objectives

In this work, we provide a model of user-interface generation and optimization for dynamically-composed applications. Our framework, named *Liquid-Interface* automatically generates form-based user-interface from dynamic compositions. The output of the generation process is a *mockup*: a visual presentation of a design that approximates what the final application will look and behave. It does not, however, capable of executing the functionality of the application.

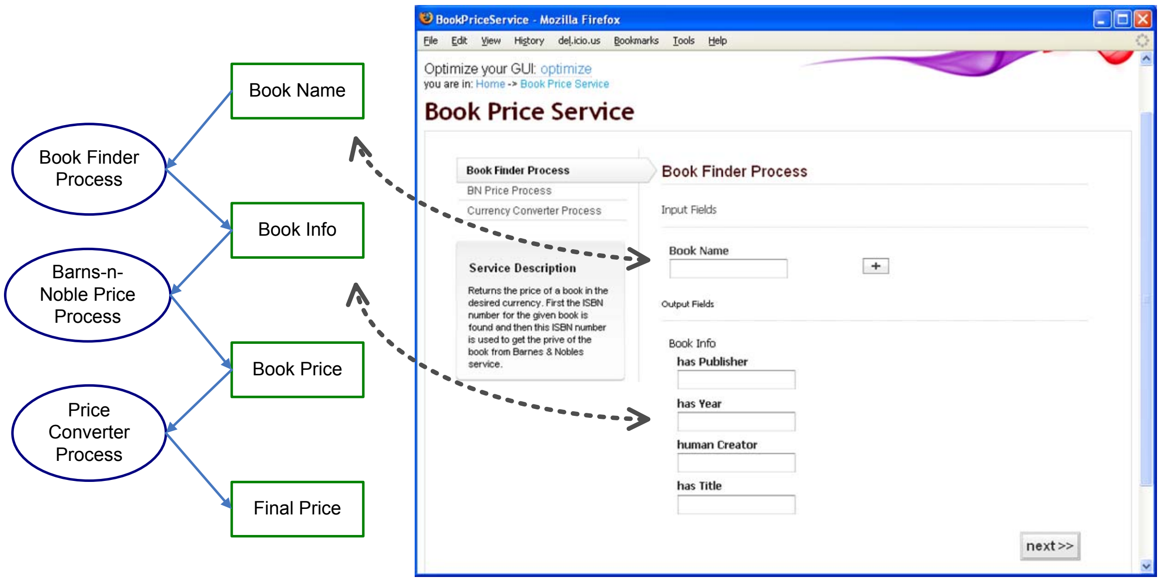
The usability of the user-interface is enhanced through two dimensions:

- Optimizing semantics: The user-interface is brought nearer to the user's concepts and vocabulary by providing additional information and explanations taken from ontologies which are related to the application.
- Optimizing navigation: modifying the navigation of the application with the intention of making it more efficient, secure and manageable.

III. Method

The input to the generation process is a model of dynamically-composed application, written in OWL-S [2], which is a widespread language used to define dynamic compositions using rich semantic models. The parts of the OWL-S model which are relevant to this research are the process model, which defines the execution order of the processes, and the process specification, which defines the input and output parameters of processes using ontological concepts. In order to exemplify our approach we use a simple composition, depicted in Figure (a), describing a book buying application. The application is composed of three sub-processes, represented by the ellipses and ordered as a sequence. The square objects represent input and output parameters of the processes.

The user-interface generation process creates a Web-form for each sub-process, generating the form's fields from the input and output parameters. Figure (b) contains a screenshot of such a form, generated for the process model in Figure (a). The navigation between the forms is based on the execution order of the sub-processes. For instance, if the processes are ordered in a sequential form, then the user would be able to navigate between the forms through a wizard-like fashion, using *next* and *back* buttons. If the sup-processes are ordered



Service Description

Returns the price of a book in the desired currency. First the ISBN number for the given book is found and then this ISBN number is used to get the prive of the book from Barnes & Nobles service.

(a) The Input: an OWL-S Process Model

(b) The Output: a Generated Web-Form

as parallel, the user would be able to interact with each of the processes using independent windows.

The semantic optimization process is based on analyzing semantic concepts, which are part of the OWL-S process specification. In OWL-S, each input and output parameter is mapped to a concept that formally defines its essence. In order to provide richer semantics to the users, these concepts are expressed using interface widgets. For example, as the *Book Info* concept contains several properties, such as *title*, *publisher* and *creator*, these concepts are displayed as additional fields, presented in the context of the parent field. In Figure (a) and (b), dotted lines depict concept expressions. The type of the user-interface widget is adjusted to the semantic type of the concept. For example, concepts that express dates are displayed using a calender, and concepts that have a bounded set of values (e.g. countries or currencies) are displayed as 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 model according to a set of *user-interaction design patterns*. As measures for evaluating the quality of user-interface navigation are rather vague, we created a taxonomy of user-interaction design patterns, selecting patterns which are relevant to navigation. For example, the *Flat and Narrow Tree* design pattern defines optimal measures to link distribution between the pages. Each of the selected patterns were modeled as functions that assign a *navigational score* to a configuration of the application's navigational properties, such as the number of links between pages and the number of fields within a page. The functions were then used in order to maximize the overall navigational score of the application, using non-linear optimization methods.

IV. Conclusion

In this short paper we presented a model of user-interface generation and optimization for dynamically-composed applications. Preliminary results include a proof-of-concept implementation that contains full semantic optimization and partial navigation optimization. We had tested the implementation with several different compositions from various sources and observed an improvement in the overall usability of the application. The preliminary results also reveal interesting relations between design patterns used for optimizing navigation, including patterns that contradict (or enforce) each other. Investigating these relations, as well as experimentally evaluating the general model, are planned for future research.

References

[1] Iso 9241-11. ergonomic requirements for office work with visual display terminals (vdts) 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] S. Dustdar and W. Schreiner. A survey on web services composition. *International Journal of Web and Grid Services*, 1(1):1–30, 2005.
[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] A. R. Puerta and J. Eisenstein. Towards a general computational framework for model-based interface development systems. *Knowledge-Based Systems*, 12:433–442, 1999.
[6] Josef Spillner, Iris Braun, and Alexander Schill. Flexible human service interfaces. In *ICEIS (5)*, pages 79–85, 2007.
[7] 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.