

Generating and Optimizing Graphical User Interfaces for Semantic Service Composition

1 Background

Semantic Web service composition is a discovery process in which a given set of requirements are fulfilled by locating and assembling semantically annotated services. Semantic annotation of Web services is based on expanding the description of Web services with formal semantic models, such as OWL-S [2]. These models provide an unambiguous description of service properties by relating them to concepts belonging to *Web ontologies* [3].

Dynamic composition is a new way for creating software applications. Rather than manual coding the new application, the application is generated automatically, at least partially, by reusing existing software services according to the user's requirements [6, 8]. The method has several advantages: a) it answers an instantaneous request of the user; b) the application is flexible, changing according to failures, price change, 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].

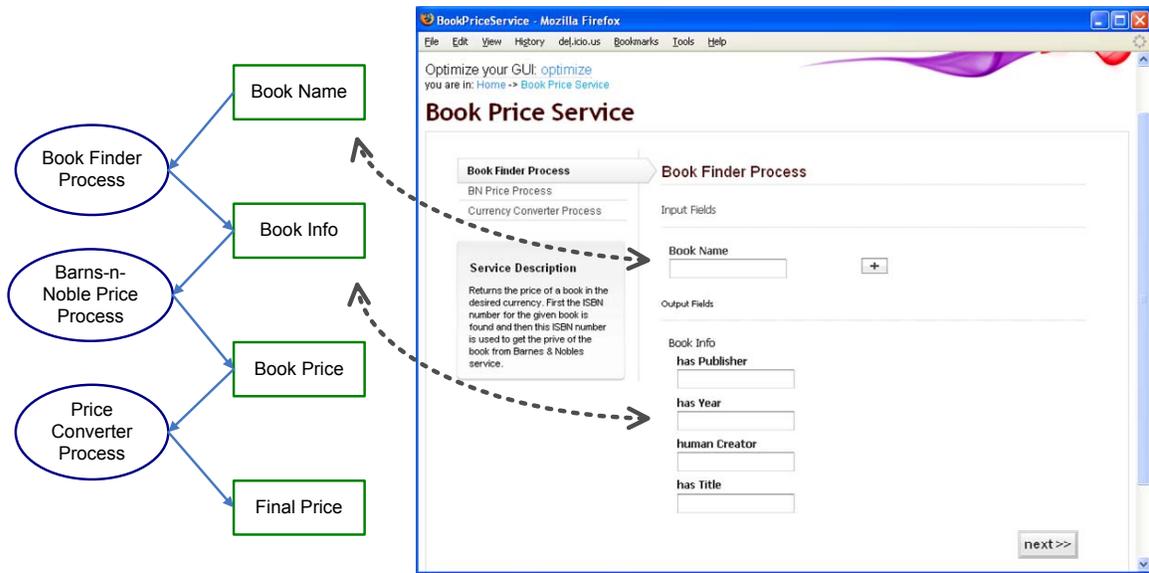
In traditional software development processes, the user interface is derived from the requirements and desired functionality of the application model. It can be carefully designed and tested in order to ensure its usability. In contrast, in dynamically composed applications, the functionality is not set 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 temporary 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, and using the formal model in order to generate user interfaces [7, 5]. 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.

In this demonstration, 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 Framework can be used online at:

<http://dori.technion.ac.il/liquidInterface>. The code is distributed under open-source license, and can be downloaded from

<http://projects.semwebcentral.org/projects/liquidinterface/>. The composition is



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

(b) The Output: a Generated Web-Form

created by submitting queries in a natural language to a service matcher. The output of the generation process is a *prototype*: a visual presentation of a design that approximates what the final application will look and behave. The prototype then goes through a process of optimizations, in which the model is transformed according to a set of design patterns that reflect the usability of the application.

2 The Liquid-Interface Framework

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 2 (b) contains a screenshot of the form generated from the first process of the model

in Figure 2 (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 as parallel, the user would be able to interact with each of the processes using independent tabs.

3 Optimization by Model Transformation

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.

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 calendar, 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 [4] 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 Liquid-Interface framework includes an open architecture that allows new design patterns to be defined and added dynamically to the optimization process, in a given order.

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, including patterns that contradict (or enforce) each other.

4 Demonstration Walkthrough

To illustrate the framework, we will consider a set of business processes used by a corporation to handle its book-buying services. A set of screenshots describing the demonstrations can be found at:

<http://dori.technion.ac.il/liquidInterface/demo>.

The demonstration would include the following steps:

1. The demonstration starts with an online assembling of a new composition, which implements the book-buying process. The composition is created by Opossum [8] - a search engine that composes services according to natural language queries.
2. Liquid-Interface creates a prototype, which is an interactive user interface that reflects the newly-created composition. The demonstration highlight visual representations of semantic properties, such as compound concepts, subset relations between concepts, instance enumerations, and cardinality constraints.
3. The optimization of the prototype's navigation is presented. The initial prototype exhibit an unusable user interface, due to the complexity of the concepts related to the composition. By activating several optimization patterns, the usability of the prototype is improved, in a visible way.

References

1. Iso 9241-11. ergonomic requirements for office work with visual display terminals (vdots) 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. T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, 284(5):34–43, 2001.
4. Jan Borchers. *A Pattern Approach to Interaction Design*. John Wiley & Sons, Inc., 2001. Foreword By-Frank Buschmann.
5. Deepali Khushraj and Ora Lassila. Ontological approach to generating personalized user interfaces for web services. In *International Semantic Web Conference*, pages 916–927, 2005.
6. 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.
7. A. R. Puerta and J. Eisenstein. Towards a general computational framework for model-based interface development systems. *Knowledge-Based Systems*, 12:433–442, 1999.
8. 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.