# Retrieval of User Interface Templates Based on Tasks

Jordan Janeiro[1]

Advisors: Thomas Springer[1], Simone DJ Barbosa[2], Alexander Schill[1]
[1] Technische Universität Dresden, Department of Computer Science, Institute for Systems Architecture, Nöthnitzerstr. 46, 01187 Dresden, Germany
{jordan.janeiro, thomas.springer, alexander.schill}@tu-dresden.de
[2] PUC-Rio, Computer Science Department, Rua Marquês de São Vicente, 225, Gávea, Rio de Janeiro, RJ, Brazil
simone@inf.puc-rio.br

**Abstract.** Using template structures is an interesting approach to develop user interfaces. A designer can predefine such structures, following best user interface practices, to be constantly reused. However, it may be a problem to navigate through a repository of templates to find a suitable one for a certain application. Therefore, we propose a retrieval mechanism for templates based on its supported tasks.

**Keywords:** user interface; template; information retrieval; natural language; task terminology; search engine.

## 1 Introduction

The user interfaces subject is still a complex open problem in computer science. According statistics, 50% of the source code and development time of a system are due to the user interface design [1]. Therefore, there is naturally the requirement to automate such process. Indeed, there are approaches which aim to automatically generate user interfaces, analyzing the logics of a system. However, such automatically approaches generally present usability problems [2].

A suitable approach, as an alternative for full manual or automatic user interface creation, is the use of *templates* [3]. The idea of using templates is that a designer can previously define a generic user interface for certain types of system functionalities and publish them, in the sense that designers or even non-designers can reuse the templates to easily create end user interfaces.

Generally, in software companies, such repositories may contain a large number of templates and thus, it becomes difficult for developers to find a template for a specific purpose. Therefore, the goal of this work is to support developers on finding templates based on aspects as, the user tasks the template supports and its structural description. Such aspects are specified by a natural language query and have to be matched with a formal description attached to the templates. Our approach should identify the most relevant templates which cope with the semantics of such query.

The details of our approach are illustrated by a running example; a personal web page scenario, which manages a photo album. For this scenario, we defined a template, as presented by Figure 1 (b).

## 2 Template Search Mechanism

Our approach proposes an engine which searches for templates, based on its semantic information. Figure 1 (a) presents an overview of the steps involved in the search mechanism. First, the user describes the template he searches, by specifying its semantic information in a query; the engine receives the query and analyses it to find the matching templates and, finally, the engine presents the templates, ranking them by relevance. The following subsections provide more details about the search mechanism just described.

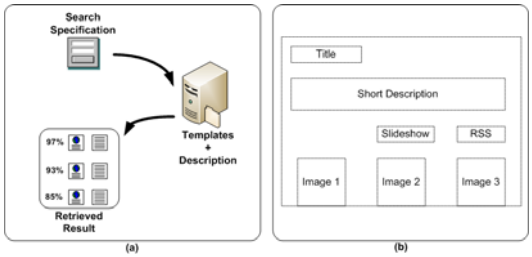


Figure 1. (a) Methodology for searching templates based on its description. (b) template for a personal photography web page.

### 2.1 Template Description Model

This work proposes a standard definition for a model which describes user interface templates, called here *template description model*. By template description we mean semantic information such as: the possible tasks a template supports, from the end user point of view; the modality in which a task is executed; the elements which compose a certain template and the device which the template targets. Such information is then attached to the templates to allow the search engine to perform semantic searches.

Our model is based on previous approaches [4][5] which aim to describe taxonomies for user interface tasks, which can be performed by end users. Based on the model we propose here, we identify that the template of our use case might perform the following tasks using the notation proposed here [5], for example: *UseInformation(slideshowHyperlink)*, *UseInformation(viewThumbnail)* and *ProvideInformation(subscribeRSSFeed)*. With such tasks, we can describe, for example, that our use case template, allows the user to follow the slideshow hyperlink, to view the slideshow of the photos which are currently in the web page; or simply to view thumbnail pictures; or to allow the user to subscribe to news (RSS feeds) associated to the web page.

### 2.2 Query Formulation

Describing the semantics of a template in any kind of query language may imply the addition of unnecessary complexity for the template adopter, such as learning a new language and obtaining enough experience with it to describe precisely the goal of a query. Therefore, we chose to use a natural language, in this case English, as the query language for the search mechanism. For example, for our use case example we can formulate a query like, "*a user interface which presents a photo album as thumbnails. The user can: click on the thumbnails and view full images, the user can view the photo album as a slideshow and the user can subscribe to the RSS feeds service*". We assume that the template adopters intuitively use the task terms in the query which can be further extracted and mapped to the tasks compiled by our template description model.

### 2.3 Search Matching

If in one hand the use of English as a natural language provides flexibility on specifying a query for templates, in the other hand it introduces the complexity of identifying its goals. As we allow the user to formulate a query freely using the English language, our major challenge is to analyze such a query to identify which terms match the terminology, specified by our model. For this purpose, our approach uses the unstructured information management (UIM) framework[1], a system which performs the recognition of named terms (structured data) from a text (unstructured data) by information extraction techniques. The framework achieves its goal by integrating a structured data source, which contains a terminology, and then identifying these terms in an unstructured data (text). In our case, the structured data refers to the template description model terminology and the unstructured data refers to the query that the user formulates in natural language.

As the user formulates a query using natural language, we must consider that the query may not specify exactly the terminology foreseen in our model, due to the possibility to formulate it with synonyms of our terminology, or even formulate the purpose of a query in a different way. To support the solution of this problem, we intend to use existing terminology databases, such as *WordNet*[2] or *SAPTerm*[3], which aim to describe concepts and its relationship. We can use such database in combination with the UIM framework to identify the goal of a query formulated with other terms, which may be synonyms.

## 3 Related Work

Many approaches in literature present proposals to define task taxonomies, based on the interaction between end users and systems [4][5][6]. Although these studies

[1] Unstructured Information Management Architecture, http://uima-framework.sourceforge.net
[2] WordNet, http://wordnet.princeton.edu/
[3] SAPTerm, http://help.sap.com/saphelp_nw04/helpdata/en

describe a set of tasks a certain user interface supports, they do not propose any practical kind of integration of taxonomies that formally describe user interfaces.

With similar concepts presented in this work, the Web Semantic concept [7] proposes the attachment of additional information (semantic) to web services to allow its discovery, in a further step. There are some existing semantic service matchmakers which perform such discovery processes based on the attached semantic, by non-logic, logic or hybrid matching techniques [8]. In our approach, the semantic we attach to the templates represent tasks and our search engine represents the semantic service matchmakers, which searches the templates based on the attached tasks.

## 4 Conclusion

This paper presents an approach which retrieves templates, annotated with the template description model, through natural language queries. For achieving this goal, we define a model, mainly based on task terminologies, which aims to describe the tasks a template supports, by the end user point of view. We annotate the templates using such a model to perform searches of templates based on tasks. We use natural language as a query language, to allow the user to easily formulate queries to find the templates. To support such an approach, we adopt the UIM framework for processing the queries and identify the tasks defined within it. In combination with the UIM framework, we use a terminology database to recognize the synonyms of the tasks.

## References


1. Myers, B., Rosson, M.: Survey on User Interface Programming. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 195-202. ACM Press, New York (1992)
2. Spillner, J., Feldmann, M., Braun, I., Springer, T., Schill, A.: Ad-Hoc Usage of Web Services with Dynvoker. In: Towards a Service-Based Internet. LNCS, vol. 5377, pp.208-219. Springer, Heidelberg (2008)
3. Nielsen, J.: User Interface Directions for the Web . In: Communications of the ACM, pp. 65-72. ACM Press, New York (1999)
4. Zhou, M., Feiner, S.: Visual Task Characterization for Automated Visual Discourse Synthesis . In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 392-399. ACM Press, New York (1998)
5. Byrne, M., John, B., Wehrle, N., CrowThe, D.: Tangled Web We Wove: a Taskonomy of WWW Use. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 544-551. ACM Press, New York (1999)
6. Shneiderman, B.: The Eyes Have It: a Task by Data Type Taxonomy for Information Visualizations. In: IEEE Symposium on Visual Languages, pp. 336-343. IEEE Press, New York (1996)
7. Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web, Scientific American, vol. 284, pp. 34-43. (2001)
8. Klusch, M.: Semantic Web Service Coordination, In: CASCOM: Intelligent Service Coordination in the Semantic Web, pp.59-104. Springer, Heidelberg (2008)