Chapter 15
Semantic User Interfaces

Károly Tilly
Oracle, Hungary

Zoltán Porkoláb
Eötvös Lóránd University, Hungary

ABSTRACT

Semantic User Interfaces (SUIs), are sets of interrelated, static, domain specific documents having layout and content, whose interpretation is defined through semantic decoration. SUIs are declarative in nature. They allow program composition by the user herself at the user interface level. The operation of SUI based applications follow a service oriented approach. SUI elements referenced in user requests are automatically mapped to reusable service provider components, whose contracts are specified in domain ontologies. This assures semantic separation of user interface components from elements of the underlying application system infrastructure, which allows full separation of concerns during system development; real, application independent, reusable components; user editable applications and generic learnability. This article presents the architecture and components of a SUI framework, basic elements of SUI documents and relevant properties of domain ontologies for SUI documents. The basics of representation and operation of SUI applications are explained through a motivating example.

INTRODUCTION

Creating and editing static documents is one of the most common computer skills. The required abilities, entering and editing text, importing figures, defining layout and decorating pieces of contents seem to be learnable by any literate person. If we take a look at the user interface of an application system, we just see sets of text and figures representing menus, toolbars, controls and pieces of content. There is seemingly nothing there, which could not have been arranged by a teenage student, who knows, how to edit static web pages using a fancy HTML editor.

In fact teenage students are currently not able to create user interfaces. Let us see, why. User interfaces contain pieces of contents with specific semantic meaning known by the user.
This knowledge is the user’s mental model of the application. On the other hand, semantic meaning is encoded by implementors into program components and data structures according to information gained about the application domain during system analysis and design. Human computer interaction is basically a mapping between the user’s mental model and the internal representations of domain knowledge incorporated in the application system. This mapping is nowadays implemented by hardcoded, application specific associations between user interface- and application level components using event handlers and relatively simple protocols. We call this feature strong semantic coupling between user interfaces and underlying application layers. It results that it is not enough for a teenage student to understand user interface level concepts and components, but she should also be familiar with the methods of creating and activating application components for executing user requests.

We argue that strong semantic coupling is one of the main reasons, which currently holds back domain experts from creating applications by themselves. If we could semantically separate the user interface from underlying components, it would lead to significant advantages, among others full separation of concerns between stakeholders during system development; real, application independent, reusable components; user editable applications and generic learnability.

To characterize semantic separation, we distinguish it first from modular and spatial separation.

Modular separation is related to application design by clearly isolating user interface functionalities from pieces of application logic at the implementation source code level. This kind of separation is successfully implemented in state of the art GUI libraries, generally based on a form of the Model-View-Controller pattern.

Spatial separation is mainly related to application deployment. It allows user interface components and application logic components to be executed on physically different devices. Spatial separation is fundamental in web based environments, where user interface components are rendered by web browsers, while the execution layer is embodied in application logic and database tier components activated through application servers. Spatial separation supports platform- and device independence as well.

Modular and spatial separation do not prevent strong semantic coupling, however, so the user interface and the underlying application logic is still designed, implemented and operated as a single, monolithic unit.

Semantic separation is about explicitly separating domain concepts of the user’s mental model from user interface components and application components. The set of separated domain concepts allow a dynamic mapping between elements of user interface documents and underlying application components. In the following it is supposed that domain concepts are organized and stored in domain ontologies.

The power of semantic separation is demonstrated by hyperlinks in static HTML pages. From our current point of view a hyperlink offers a service with a predefined, simple contract, which says that the associated target is navigated, whenever the user clicks the hyperlink object on the screen. This contract is fixed in HTML/HTTP standards, and implemented by web browsers and HTTP servers. Anybody can assemble websites from static HTML pages, because it is enough to know the contents of the document, and the hyperlink contract, but it is not necessary to know anything about the underlying application environment (in this case the HTTP server).

To generalize the above described nice properties of hyperlinks, we introduce Semantic User Interfaces (SUIs) and SUI Documents (SUIDs). A SUI is a user interface, which implements semantic separation. SUIDs are static, user editable documents for describing SUIs. SUIDs can contain references to arbitrary services, whose contracts are specified in domain ontologies. The purpose of SUIDs is to empower domain experts to create