Web-based Tool Integration: A Web Augmentation Approach

Oscar Díaz[1], Josune De Sosa[2], Cristóbal Arellano[1], and Salvador Trujillo[2]

[1] ONEKIN Group, University of the Basque Country (UPV/EHU), Spain
{oscar.diaz,cristobal.arellano}@ehu.es
[2] IKERLAN Research Centre, Mondragon, Spain
{jdesosa, strujillo}@ikerlan.es

Abstract. Desktop tools are steadily being turned into web applications. Tool integration then becomes a question of website integration. This work uses Web Augmentation techniques for this purpose. An integration layer is deployed on top of the existing Web-based tools that augments the rendering of those tools for the integration experience. Layers are specified through a statechart-like DSL and transformed into JavaScript.

1 Introduction

Tool integration is a matter of reducing "*accidental complexity*" due to the different semantics brought by each tool. Tools might differ on the data format, user-interface conventions, use of common functions, the process flow, etc [3]. Tool integration can be achieved on three different levels: the data source level, the business logic level, and the user interface (UI) level [4]. UI integration has two significant benefits: *(i)* existing applications' UIs can be reused, and *(ii)* users already familiar with existing UIs do not have to learn how to work with new ones. So far, UI integration has been investigated at the component level where a bright new integration application is constructed from existing components [1]. However, tools are not components but full-fledged web applications. Portlets can fit this scenario but they impose a heavy footprint on both the tool provider and the tool consumer (i.e. the tool integrator) [2]. This certainly hinders the openness and self-serviceness of the solution.

This work's research question is whether the use of Web Augmentation (WA) techniques can provide an alternative balance between expressiveness and self-serviceness. Rather than sophisticated and expressive solutions such as those of portlets or UI components, WA introduces a lightweight solution based on the front-end. WA is to the Web what *Augmented Reality* is to the physical world: layering relevant content/layout/navigation over the existing Web to customize the user experience. This is achieved through *JavaScript* using browser weavers (e.g. *Greasemonkey*). However, WA is hindered by being programming intensive and prone to malware. As a result, we resort to Domain-Specific Languages (DSLs) as a way to abstract away from the implementation details, ease user participation, and promote openness. We introduce *CORSET*, a DSL for Web-based tool integration based on process flows.

2 The Running Example

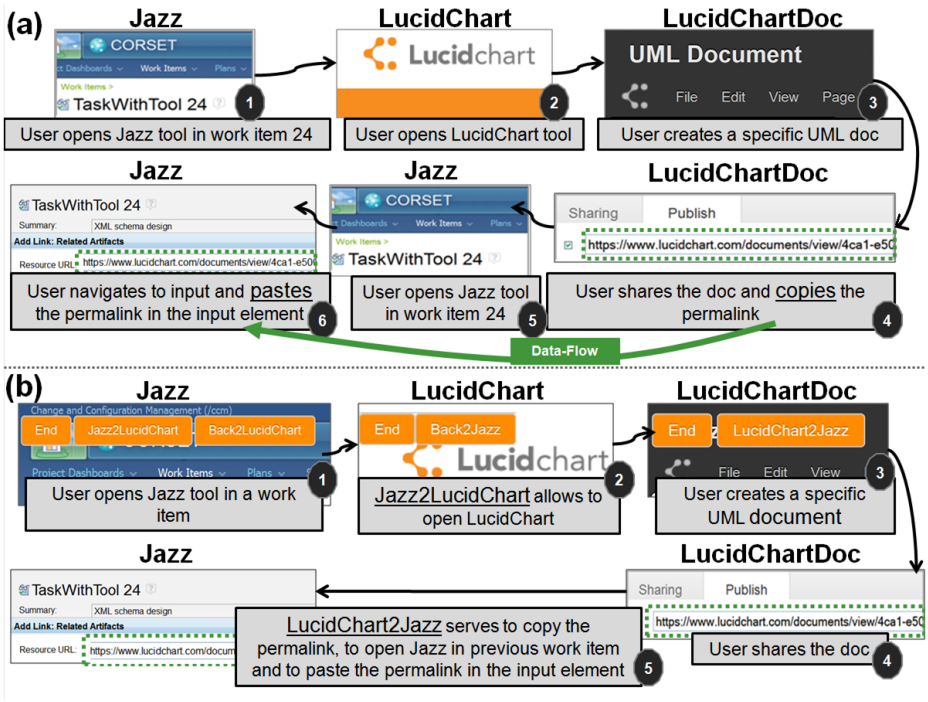

Fig. 1. Integrating the websites of *Jazz* and *LucidChart*. (a) The process expands along the two websites. (b) A CORSET layer is interspersed to support this integration.

The tools to be integrated include: *Jazz Rational Team Concert*[3] (hereafter, just *Jazz*) to be used for the management of the software development lifecycle, and *LucidChart*[4], a tool for model design. Figure 1(a) outlines the process flow between *Jazz* and *LucidChart*. First, the user opens *Jazz* to check the workload. Task *24* has been assigned: *"design the xml schema"*. This task involves the design of a UML class diagram. This requires to move to a different tool: *LucidChart*. Once created, the UML diagram is assigned a permalink. This permalink is to be shared with the rest of the *Jazz* community. To this end, the user goes back to task *24* in *Jazz*. Finally, the user copy&paste the permalink as an artefact associated to the *Jazz* task. In this scenario, the integration functionality is in the user's head: no support is given to sustain neither the control flow (e.g. when to navigate from *Jazz* to *LucidChart*, and vice versa) nor the data flow (e.g. the diagram permalink that flows between the websites). This is the very

[3] https://jazz.net/projects/rational-team-concert/
[4] http://www.lucidchart.com/

purpose of *CORSET*. Figure1(b) introduces a *CORSET* layer to sustain the sample scenario. The *Jazz* website is augmented with three additional buttons: *Jazz2LucidChart*, *Back2LucidChart* and *End*. The former serves to seamless navigate to *LucidChart* the very first time (creating a bright new UML diagram) whereas *Back2LucidChart* handles posterior visits. Likewise, the *LucidChart* website now exhibits two new buttons. *Back2Jazz* serves to navigate back to the departing state at *Jazz*. Unlike the previous case, this navigation is contextual in the sense that navigation is parameterized by the permalink of the current *LucidChart* artefact. It is worth noticing that at any moment the user can move away from these two tools, and browses other web applications. At any moment, users can finalize the process by clicking the *End* button.

3 CORSET

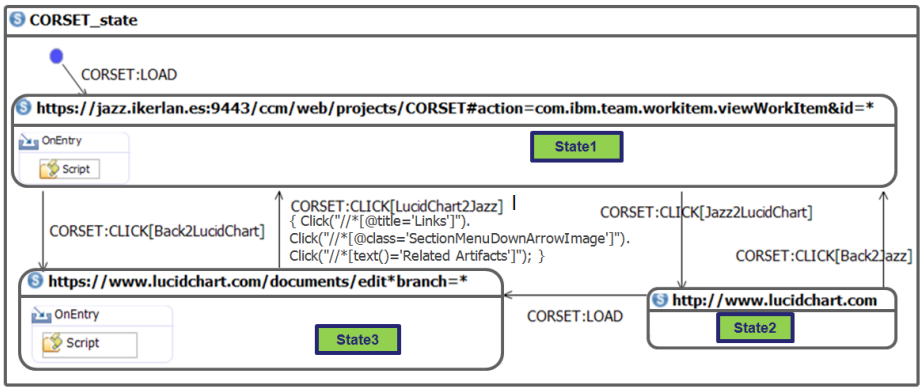

Fig. 2. *CORSET* expression for the running scenario.

CORSET uses statecharts to describe the integration scenario. Figure 2 shows the *CORSET* expression for the running scenario. This diagram is transformed into a JavaScript program. A process-based UI-centric approach to tool integration entails a control flow, a data flow and the UI augmentations.

Control Flow. It is captured through statecharts: a state is characterized by a URL pattern that identifies the set of pages that participate in accomplishing a given task; a transition normally implies moving between websites. For our running example, we have three simple states, one for *Jazz* (***https://jazz.ikerlan.es:9443/ccm/web/projects/CORSET...*** see Figure 2) and two for *LucidChart* (***https://www.lucidchart.com/documents/edit...*** and ***www.lucidchart.com***). A transition has an *event*, a *condition* and a *target* state. *Events* denote abstractions of happenings which are meaningful for the integration purpose. They are signalled by *CORSET* and abstracted from low-level DOM events. So far, two event types are considered: *CLICK*, that denotes

pushing a *CORSET* button, and *LOAD,* that is an abstraction of the DOM load event. This event is risen by *CORSET* when a tool loads an UI that matches a state of the *CORSET* at hand. Finally, an action denotes a *CORSET Script*. These scripts describe actions to be enacted in the target tool as a result of this transition. For instance, moving from *State3* in *LucidChart* to *State1* in *Jazz* requires the previous "internal navigation" of three clicks till the right UI is reached (see Figure 2 where the "Click" action mimics user clicks)

Dataflow. Some transitions might be turned into contextual links, i.e. links that carry data from the source to the target. Broadly, we have to mimic copy&paste as conducted by the user. Hence, CORSET offers a high-level "copy" and "paste" operation, and uses state variables as the clipboard. This is part of the CORSET script (not shown in the figure). For instance, the script *Copy("XPath expression").Into($stateVariable)* keeps in the state variable the output of evaluating the XPath.

UI Integration Augmentation. A *CORSET* expression also has a rendering counterpart: *the buttons*. Buttons are automatically generated from transitions. Button placement is based on the assumption that the place that holds some data of interests (the data being extracted) coincides with the place where the control flow should be governed. If no such data exists, buttons are inserted in the upper left of the window. This heuristic permits the *CORSET* engine to automatically generate the UI.

4 Conclusions

We investigate Web Augmentation techniques for tool integration. By using a DSL, we strive to shelter users from JavaScript and describe the integration declaratively as statecharts. A must follow-on is to conduct usability studies among the tool users to assess whether *CORSET* expressiveness fulfils their requirements.

References

1. Daniel, F., Soi, S., Tranquillini, S., Casati, F., Heng, C., Yan, L.: From People to Services to UI: Distributed Orchestration of User Interfaces. In: 8th International Conference on Business Process Management, BPM'10. pp. 310–326 (2010)
2. Díaz, O., Rodriguez, J.: Portlets as Web Components: an Introduction. Journal of Universal Computer Sciences (JUCS) 10(4), 454–472 (Apr 2004)
3. Thomas, I., Nejmeh, B.: Definitions of Tool Integration for Environments. IEEE Software 9(2), 29–35 (Mar 1992), http://dx.doi.org/10.1109/52.120599
4. Westermann, U., Jain, R.: Toward a Common Event Model for Multimedia Applications. IEEE MultiMedia 14(1), 19–29 (2007)