Identifying and Modelling Complex Workflow Requirements in Web Applications

Mario Matias Urbieta[1,2], Gustavo Rossi[1,2], Silvia Gordillo[1,3], Wieland Schwinger[4], Werner Retschitzegger[4], and María José Escalona[5]

[1] LIFIA, Facultad de Informática, UNLP, La Plata, Argentina
{murbieta,gustavo,gordillo}@lifia.info.unlp.edu.ar
[2] Conicet
[3] CiCPBA
[4] Department of Cooperative Information Systems,
Johannes Kepler University Linz
{wieland.schwinger,werner.retschitzegger}@jku.at
[5] IWT2 Group. University of Seville, Spain
mjescalona@us.es

Abstract. Workflow plays a major role in nowadays business and therefore its requirement elicitation must be accurate and clear for achieving the solution closest to business's needs. Due to Web applications popularity, the Web is becoming the standard platform for implementing business workflows. In this context, Web applications and their workflows must be adapted to market demands in such a way that time and effort are minimize. As they get more popular, they must give support to different functional requirements but also they contain tangled and scattered behaviour. In this work we present a model-driven approach for modelling workflows using a Domain Specific Language for Web application requirement called WebSpec. We present an extension to WebSpec based on Pattern Specifications for modelling crosscutting workflow requirements identifying tangled and scattered behaviour and reducing inconsistencies early in the cycle.

Keywords: Requirements, Workflow, Crosscutting, Model-driven paradigm, Web application.

1 Introduction

Nowadays business must adapt to global trends in order to keep users engaged; unplanned marketing campaigns, season promotions (final season sales), crisis management, among others business requirements are examples of unexpected requirements that stress the whole applications' infrastructure.

We will focus on the problem produced by those requirements that demand business processes change according to the users' context. Depending on context variables like current day, payment method, active market campaign, accessing device, etc. the system may modify the underlying workflow model; this may imply execut-

M. Grossniklaus and M. Wimmer (Eds.): ICWE 2012 Workshops, LNCS 7703, pp. 146–157, 2012.

ing a slightly different workflow providing adaptations which support new requirements like discounts and free-shipping, or introduces new workflow steps like new forms to be filled, etc. Unfortunately, these changes may affect different application´s features. In Web Applications these changes compromise several applications' tiers (model, navigation, and interface). When the underlying workflow changes, user interfaces may, for example, introduce a new form that will demand new view controllers that orchestrates validation and navigation, and finally the business model must be modified for supporting new entities' forms and fields.

To make matter worst, when new concerns are unforeseen and unpredictable like Crisis Management[7] or Volatile requirements[8], these requirements are usually introduced in an ad-hoc way. The inadequate implementation of these changes may lead to a decay of software quality compromising application maintenance, stability, and complexity, and finally the application's budget.

In this paper we present a model-driven approach for analysing and modelling workflow changes in Web adaptations in the early stage of requirement gathering. The main contribution is a model-driven approach for dealing with base and adaptation requirements. It is based on a clear separation of concerns applied in the early phase of the software development process. The approach allows defining symmetrically both base and adaptation requirements; later these models are used for implementing test suites that assess the final application behaviour.

The rest of the paper is structured as follows: in Section 2 the problem will be motivated with simple but illustrative examples; in Section 3, we discuss some related work; in Section 4, we present some background themes; in Section 5, we introduce an extension for WebSpec that uses Pattern Specification; and in Section 6 we present our model-driven approach for modelling workflow changes in Web Application and in Section 7, a running example is presented; finally in section 8 we conclude and discuss some further work we are pursuing.

2 Motivating Example

We motivate our research with an example in the e-commerce domain. In the checkout process for buying selected items, the user must follow a simple workflow presenting several steps such as choosing the wrapping configuration (regular or special for birthday), selecting the shipping address, and the payment method, etc. Suppose an unforeseen event such as a catastrophe that leads to a donation campaign. We may require the introduction of a new donation step in the workflow, where users can choose between different pre-set amounts of money to donate. This change will require at least a set of modifications:

- Implement a page that holds a donation form with its corresponding fields;
- The corresponding step must be placed in the workflow and the workflow must be modified to be coherent;
- New data needs to be stored and therefore we need to add persistence machinery for these data;
- Navigation models demands modification to let users navigate to their donations for example.

In this case the set of changes must be present only when the catastrophe campaign is active otherwise they make no sense. In the mid-term we have an adaptation requirement (the existing of a catastrophe and the donation campaign) which lead to a "context-aware" workflow behaviour.

Additionally, the impact in the application of the adaptation may not be simple; that is, the introduction of this adaptation may cross other workflows such as ticket booking for a recital, product pre-order, etc. Therefore, the way in which the adaptation requirements are modelled is critical to assure that they correctly implemented.

To make matter worst, the incoming of new context-aware requirements that crosscut several workflows make the situation more complex since different business domains are compromised by the same set of events.

3 Related Work

Adams [1] et al. presents the soviet "Activity Theory" as a driver for a more flexible and better directed workflow support. A subset of the main theory´s principles highlights the need of context awareness in each possible workflow action execution. The authors propose a set of criteria as requirements of Workflow Management Systems (WfMSs). One criteria "adaptation by reflection" promotes flexible, dynamic and evolving workflows. In this case, systems must record derivations (exceptional flows in the workflow definition) capturing its reasons and its resolution that later can become part of the next workflow instantiation. Although this attempt will help to implement awareness workflows, it works reactively from exception instead of being a proactive solution. As exceptions are captured in real-time, the solution recorded is ad-hoc and isn´t neither modelled nor optimized by domain experts. This work was assessed with the implementation of a WfMS so called YAWL [2] that allows implementing dynamic workflows. The platform defines Worklet as a reusable unit of work. Each time a workflow derivation event is detected it is either possible to choose an already defined worklet or define a new one.

AO4BPEL [3] is an aspect-oriented extension to BPEL that allows describing workflow´s crosscutting behaviour. The extension comprises a language that is used to declare aspects and an execution engine that is responsible of weaving core workflows with workflow aspects. The language introduces constructors for pointcut, jointpoint and advice concepts. It is noteworthy that the extension supports process-level aspects being activated in all workflow instances and instance-level being activated on certain instance of the workflow. AO4BPEL is a powerful tool for describing aspects in Business Process models but aspects are taken into account later (in the design phase) where crosscutting can not be identified and checked with stakeholders.

We are not aware about any approach that allows identifying workflows and specifying its aspects in the requirement gathering phase in such a way that the whole application behaviour is described allowing assessing its behaviour first with the user and later by automatic testing.

4 Background

In this section we introduce some base work which we have used in our approach, namely WebSpec for modelling workflow requirements and Pattern Specifications for specifying the binding of requirements belonging to different concerns.

4.1 WebSpec

WebSpec [9] is a visual language; its main artefact for specifying requirements is the WebSpec diagram which can contain *interactions*, *navigations* and *rich behaviors*.

A WebSpec diagram defines a set of scenarios that the web application must satisfy. An *interaction* (denoted with a rounded rectangle) represents a point where the user can interact with the application by using its interface objects (widgets). *Interactions* have a name (unique per diagram) and may have widgets such as labels, list boxes, etc. In WebSpec, a *transition (*either *navigation* or *rich behavior)* is graphically represented with arrows between *interactions* while its name, precondition and triggering actions are displayed as labels over them. In particular, its name appears with a prefix of the character '#', the precondition between {} and the actions in the following lines.

The scenarios specified by a WebSpec diagram are obtained by traversing the diagram using the depth-first search algorithm. The algorithm starts from a set of special nodes called "starting" nodes (*interactions* bordered with dashed lines) and following the edges (*transitions*) of the graph (diagram).

As an example of WebSpec's concepts we present in Fig. 1 the specification for the user story: "As a customer, I would like to search products by name and see their details" in an e-commerce application. *Home* represents the starting point of the specification and it contains 2 widgets: *searchField* text field and *search* button (see [9] for further details).

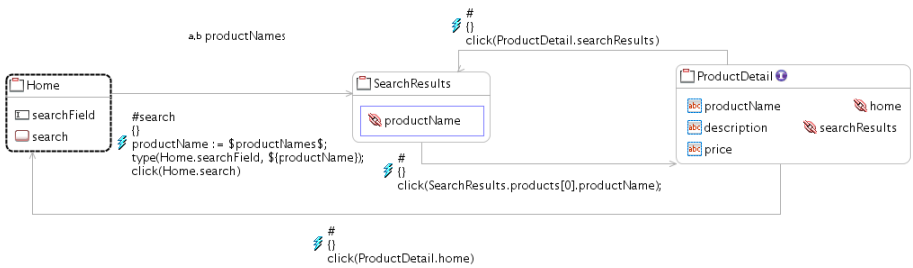

Fig. 1. WebSpec diagram of the *Search by name* scenario

WebSpec has a supporting tool [16] with features that allows, in the early phases of requirement gathering, realizing simulation of application interaction against mock interfaces and generating independent Web tests for testing the final development result.

4.2 Pattern Specification

Pattern Specifications (PSs) [6] is a tool for formalizing the reuse of models. Originally the notation for PSs was presented using the Unified Modelling Language (UML) as a base but in this work we will instead use the concept in the WebSpec realm. A Pattern Specification describes a pattern of structure defined over the roles which participants of the pattern play. Role names are preceded by a vertical bar ("|"). A PS can be instantiated by assigning concrete elements to play these roles.

5 Crosscutting Behaviour Modelling Using Pattern Specification

WebSpec provides a powerful language for describing user's interaction of Web application as it was introduced in previous section. Nonetheless it lacks a means for portraying generalization of interaction patterns; for example, common patterns required in determined workflows' points (tasks or transitions) that stop the workflow execution up to the manager authorizes to continue, or landmarks-like behaviour where a given subworkflow can be accessed from steps belonging to a main workflow. This restriction increase size and complexity of diagrams, and effort to document the requirement. So, we propose the use of Pattern Specifications (PS) where, in our case, a role is a specialization of a WebSpec *Interaction* restricted by additional properties that any *Interaction* fulfilling the role must possess. A model conforms to a PS if its model elements that play the roles of the PS, satisfy the properties defined by the roles.

In Figure 2, a requirement that generalizes an interaction pattern defines two roles: |*sourceInteraction* and |*targetInteraction*. The |*sourceInteraction* role (notice that role's name starts with "|") demands a widget of type Label called *mandatoryWidget* that must be present in the *Interaction* that conforms the role, and defines a new widget of type TextField called *introducedWidget* that will be part of conforming *Interaction*. The |*targetInteraction* role is analogous to the previous role; it demands a widget of type Combobox called *mandatoryWidget* to be part of the interaction that matches the role. Finally, when both roles are bound in a given diagram, a new interaction is introduced with the corresponding transitions called *IntroducedInteraction* as it is defined in Figure 2.

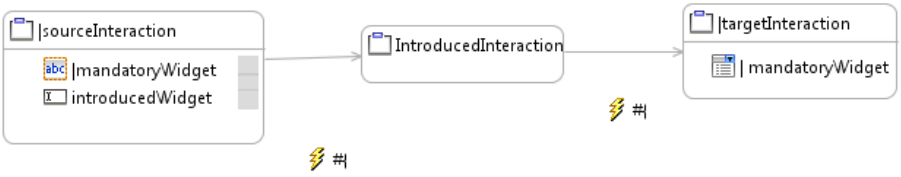

Fig. 2. Introducing interactions and elements in a Workflow requirement

In Figure 2, PS was used for introducing a new *Interaction*. Alternatively, it can be used for defining constraints over a diagram that may lead to an overriding of existing definitions. E.g. Navigations preconditions and actions may be introduced by PS in order to enrich the scenario for making consistent a set of changes. This kind of situations is usually present in adaptive requirements where some behaviour is intended to be replaced by other.

In Figure 3, we show a generalization of a Web application requirements that provides the option for donate. This introduces a banner between two roles describing the donation goal and allows traversing towards a donation form. This requirement can be instantiated in Figure 1 example where |*stepOne* role is bound with the *Home* interaction and |stepTwo with the *SearchResult* interaction.

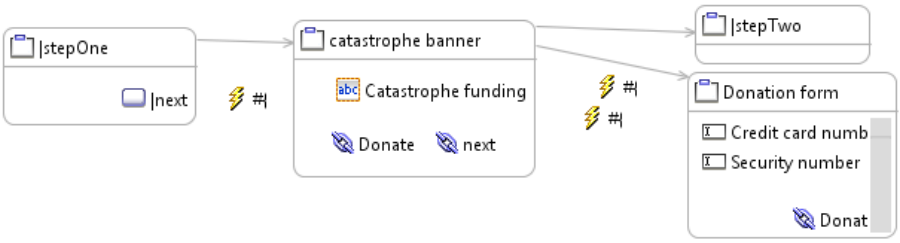

Fig. 3. Donation requirement model using PS

5.1 Yet Another AOSD Visual Language?

Although there are several AOSD (Aspect-Oriented Software Development) formal and visual languages already defined for almost any model of a Web application (conceptual, navigational, and interface models), none of them covers requirement gathering phase and indeed these are focused on describing just functional features closer to the conceptual model [17].

Tackling crosscutting workflow behaviour in the early requirement analysis phase allows identifying crosscutting behaviours in the system, and context variables that rules adaptation behaviour. The use of WebSpec with PS, will help to separate matter of interest in (WebSpec) requirement diagrams and thus in the whole System Requirement Specification (SRS) documents.

In this case, the extension provided for WebSpec using PS not only allows defining high level reusable requirements for Web Applications; it also helps to derive the set of tests that will be used for validating the final result of the application design and implementation.

6 Our Approach in a Nutshell

Next, we will present our approach to identify, design and implement adaptive requirements in Web Workflows. The approach is based on the idea that any adaptive

requirement must be treated as first–class; as a consequence we consider these requirements as belonging to separate concern[1] [11] allowing us to isolate, model and later compose both core application workflows with adaptive requirements. In this aspect we focus on Web workflow requirements, specifically in analysis and modelling aspects. Their impact in different application tiers has been already presented in[14,12].

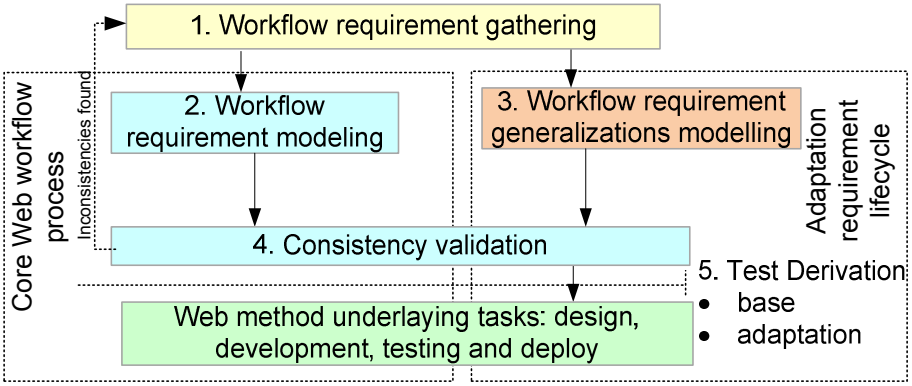

Fig. 4. Overall schema for workflow requirement modelling

The approach comprises a set of steps that are depicted in Figure 4 and described below:

Step 1: **Workflow requirement gathering**. Using well-known requirement elicitation techniques such as meetings, surveys, Joint Application Development (JAD), etc. a Software Requirement Specification (usually in natural language) is produced. In the case of an agile underlying development process, a briefer description is usually produced with user stories [4].

Step 2: **Workflow requirement modelling**. Web application requirements are formalized using a requirement Domain Specific Language (DSL). This formalization is essential during the requirement gathering process with stakeholders. By means of using a requirement DSL, the tasks such as tests derivation and scenarios simulations can be automated easily. In this work, we selected WebSpec as requirement DSL.

Step 3: **Workflow requirement generalizations modelling**. Base Workflow changes are modelled using the Pattern Specification extension for the requirement DSL; in this paper we exemplify with the WebSpec extension.

Step 4: **Consistency validation**. Syntactic and semantic analysis is performed over requirements. By means of an algebraic comparison of models, candidate

[1] In software engineering a concern represents a matter of interest that groups a coherent set of requirements.

structural and navigational conflicts are detected. On the other hand, candidate conflicts are analyzed and semantic equivalences are detected. For each candidate conflict, both the new requirement and the compromised requirement are translated from a high abstraction level (the requirements DSL) to a minimal form, using an atomic constructor in order to detect semantic differences. Semantic equivalences between requirements are detected for warning requirement analysts. For more information see [13].
In the case of adaptation requirements, a previous weaving is performed among both kind of requirements obtaining instantiated PS.

Step 5: **Test derivation**. In this step, both traditional WebSpec diagram and WebSpec PS extension are processed for producing tests that allow validating the final Web Application. This also allows assessing the set of requirement with users by using simulations in the early stages of UI mocking. Later the same tests are used in the testing phase of the software development process.

In the following section we present a simple but illustrative example for modeling workflow requirements. First, a simple workflow for checking out products in an e-commerce Web application is modelled using WebSpec. On the other hand, a simple requirement that introduces context awareness in the workflow is designed using PS.

6.1 Requirement Gathering (Step 1)

We use as a running example the development of an e-commerce site. In Figure 5, user stories [4] derived from gathered requirements are shown. There are three user stories: "Checkout process" (US1), "Reduced checkout process from smartphone" (US2), and "Ordering a product" (US3). The first, on the left-hand side, defines a basic workflow for checking out selected products in a straightforward way where issues such as product wrapping, delivery and payment method must be covered. In the middle, it is required that the delivery configuration step in the workflow must be removed and in its place the current location is used for setting up the shipping address. Finally, on the right-hand side, a user story defines another view point of the checkout process defined by a different stakeholder.

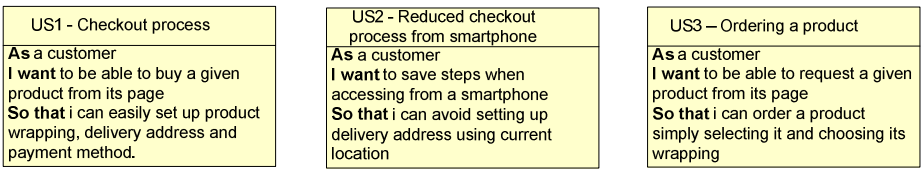

Fig. 5. Application's user stories

6.2 Workflow Requirement Modelling (Step 2)

For this step we will adopt a workflow's definition presented in [15] where a workflow has as a main objective to deal with a case. A workflow has a set of elements that allows achieving the objective: a state and a set of interconnected task where each

one can have conditions that enable its execution. From this definition, we claim that WebSpec can help modelling Workflows requirement from a user interaction perspective. User stories define the case that motivates workflow design with WebSpec. WebSpec *interaction* are used for presenting available tasks and state information, meanwhile *transition* are used for describing workflow conditions and state changes. Therefore workflows are described in a WebSpec scenario that comprises a set of (WebSpec) *interactions* and *transition*. Each *Interaction* describes the expected workflow's input and output using widgets (Labels, Radio Button, etcs.) meanwhile *transitions* represents actions that application must perform with its corresponding guard.

In Figure 6, the checkout process in a Web application is depicted as a set of interactions where the user is able to select a product for start setting out its purchase (interaction *Products*); next she is able to choose whether a simple or gift wrap should be used; next, delivery information must be introduced such as address and city; and finally the list of current orders is shown.

Fig. 6. WebSpec scenario for Checkout process based on US1

6.3 Workflow Requirement Generalizations Modelling (Step 3)

So far, we have modelled workflows in Web application using WebSpec. Sometimes there are requirements, such as US2 – "Reduced checkout process from smartphone", that introduce enhancements over main workflows like adaptations or temporal changes. In order to model this kind of requirements, we will use the proposed extension of WebSpec that introduces PS concepts for generalizing behaviours.

In Figure 7, User Story 2 (US2) is modelled overriding the default navigation presented in Figure 6 where delivery information specification (Delivery *interaction*) is by-passed, and, instead, Order Status *interaction* is exhibited after selecting Packaging configuration. This "by passing" is achieved defining a transition that goes from "Packing" interaction to "Order status" interaction. As the specification is abstract, it defines the "|Packaging" role that later binds to Packaging interaction and "|Order status" that later binds to Order status *interaction* overriding the transition identified with #next originally defined in Figure 6.

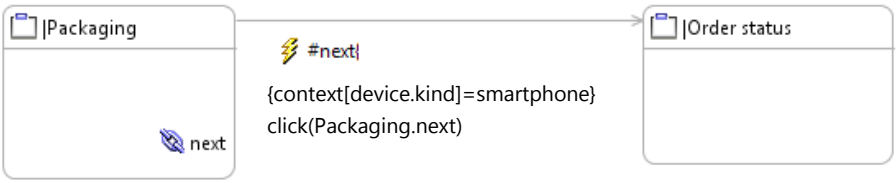

Fig. 7. WebSpec diagram for "Reduced checkout process for smartphones" user story

The semantic result of this adaptation can be seen in Figure 8 where Delivery interaction is not part of the workflow any more.

For example, the requirement modeled in Figure 3, donation requirement can also be introduced in the checkout by binding "lstepOne" role with "Packaging" and "lstepTwol with Order status. After weaving diagrams, it is possible to donate to catastrophe help´s funding after performing the checkout process. For the sake of space, we are not showing the weaving result.

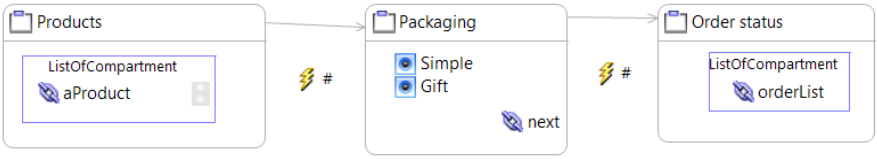

Fig. 8. Resulting WebSpec diagram after composing requirements

Although roles in this example only bind to one *interaction,* it is possible to have situations were a role may be bound to several interactions when having scattered behaviour.

6.4 Consistency Validation (Step 4)

Conflicts between requirements may arise when two (or more) stakeholders have a different point of view for a given workflow requirement. These situations present themselves as structural or navigational inconsistencies. The former type corresponds to a difference in the data belonging to a business concept meanwhile the latter defines a difference in the way interaction occurs. For more information see [13].

User story US3 proposes a slightly different workflow with respect to the one presented in Figure 6 corresponding to US1. The proposed workflow differs from the one in US1 in the way it is navigated and the data handled.

In Figure 9 a navigational conflict and a structural conflict are highlighted with an ellipse. The navigational conflict is present since it is possible to browse from the Product *interaction* towards Packaging and Delivery *interactions* defined in S1 and S3 respectively. On the other hand, the structural conflict occurs in a contradiction in the way in which the City and Country widgets are defined in the Delivery interaction; in US1 they are expected as Labels but in US3 they are expected as Combobox widgets.

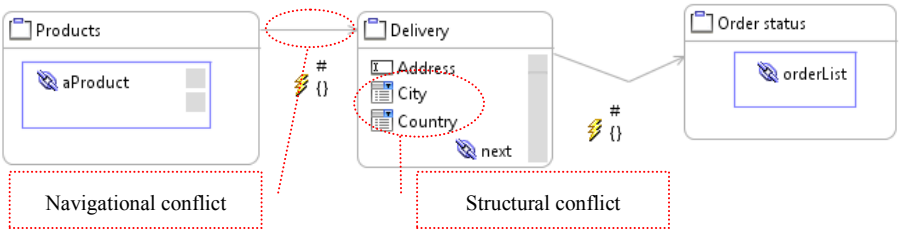

Fig. 9. WebSpec scenario for Checkout process based on US3

6.5 Test Derivation (Step 5)

Once all scenarios were described, design and further development tasks can start. The information gathered so far allows generating both core workflow and workflow's adaptation tests. That is, main workflow's tests are derived for checking navigation, and inputs/outputs from each user interaction step in the workflow. Complementary, specific tests are derived for those WebSpec diagrams that bind any adaptations (WebSpec diagrams that generalize behaviour using PS) where these validate the behaviour corresponding to the base workflow woven with adaptation requirement models.

In Code 1, we can see the result of the automatic test generation feature of WebSpec that checks the workflow of Figure 6. Besides, Code 2 shows a test case that checks the described mobile adaptation (see Figure 7 for adaptation design and Figure 8 for the resultant Workflow). Both tests uses the Selenium [10] engine for executing actions, assessments, and navigation automatically like a user would do.

```
public void
testCheckoutWflow(){
sel.click("id=aProduct");
sel.waitForPageToLoad("30000");
sel.select("id=Simple", "1");
sel.click("id=next");
sel.waitForPageToLoad("30000");
sel.type("id=Address", "..");
sel.click("id=next");
sel.waitForPageToLoad("30000");
}
```

Code 1. Checkout workflow test case

```
public void testMobCheckoutWflow(){
//context configuration
configureContextForMobileDevice()
sel.click("id=aProduct");
sel.waitForPageToLoad("30000");
sel.select("id=Simple", "1");
sel.click("id=next");
sel.waitForPageToLoad("30000");
//removed by "Reduced checkout
// process for smartphones" req.
 }
```

Code 2. Reduced Checkout workflow for mobile access test case

7 Conclusions and Future Works

In this work we have presented a novel approach for modeling Workflows in Web applications for both traditional requirements as well as crosscutting one. By using WebSpec diagrams, workflows were modeled as a set of *interactions* representing their steps and *transitions* for defining interactions' connections. In this work, a PS extension for WebSpec, allowing easily specify crosscutting workflow's behavior, was introduced. On the other hand, the approach allows modeling requirements associated to Inter-Organization Workflows [5] that, as we are aware, do not have supporting tools.

We are now implementing some extensions that allow using this approach over WebSpec tool. WebSpec diagram composition is may be the most important extension to be implemented since it must enable composing diagrams based on PS with base WebSpec diagrams. Next, the tool should reason over the set of diagram producing a semantic view (used internally) for generating tests that checks the workflows including the adaptation behaviour specified with PS.

Finally, UML class diagrams and business process models can be sketched from WebSpec diagrams by reasoning over them. Heuristics must be studied in order to produce accurate design models. Obtained UML and business process modes can be used also for producing prototype applications.

Acknowledgment. This work has been funded by the österreichische Agentur für internationale Mobilität und Kooperation in Bildung, Wissenschaft und Forschung (OeAD) under grant AR 21/2011.

References

1. Adams, M., Edmond, D., ter Hofstede, A.H.M.: The Application of Activity Theory to Dynamic Workflow Adaptation Issues. In: PACIS 2003 Proceedings. Paper 113 (2003)
2. Adams, M., ter Hofstede, A.H.M., Edmond, D., van der Aalst, W.M.P.: Worklets: A Service-Oriented Implementation of Dynamic Flexibility in Workflows. In: Meersman, R., Tari, Z. (eds.) OTM 2006. LNCS, vol. 4275, pp. 291–308. Springer, Heidelberg (2006)
3. Charfi, A.: Aspect-Oriented Workfow Languages: AO4BPEL and Applications. Phd thesis, Fachbereich Informatik, der Technischen Universit at Darmstadt (2007), http://d-nb.info/985111321
4. Cohn, M.: Succeeding with Agile: Software Development Using Scrum, 1st edn. Addison-Wesley Professional (2009)
5. Divitini, M., Hanachi, C., Sibertin-Blanc, C.: Inter-organizational workflows for enterprise coordination. In: Coordination of Internet Agents, pp. 369–398. Springer, London (2001)
6. France, R., Kim, D., Ghosh, S., Song, E.: A UML-Based Pattern Specification Technique. IEEE Transactions on Software Engineering 30(3) (2004)
7. Luecke, R.: Crisis Management: Master the Skills to Prevent Disasters. Harvard Business Press Books (2004) ISBN: 978-1591394372
8. Moreira, A., Araújo, J., Whittle, J.: Modeling Volatile Concerns as Aspects. In: Martinez, F.H., Pohl, K. (eds.) CAiSE 2006. LNCS, vol. 4001, pp. 544–558. Springer, Heidelberg (2006)
9. Luna, E.R., Garrigós, I., Grigera, J., Winckler, M.: Capture and Evolution of Web Requirements Using WebSpec. In: Benatallah, B., Casati, F., Kappel, G., Rossi, G. (eds.) ICWE 2010. LNCS, vol. 6189, pp. 173–188. Springer, Heidelberg (2010)
10. Selenium, http://seleniumhq.org/
11. Sutton, S., Rouvellou, I.: Modeling of Software Concerns in Cosmos. In: Proc. of ACM Conf. AOSD 2002, pp. 127–133. ACM Press (2002)
12. Rossi, G., Urbieta, M., Ginzburg, J.: Modular and Systematic Interface Design for Rich Internet Applications. In: Murugesan, S. (ed.) Handbook of Research on Web 2.0, 3.0, and X.0: Technologies, Business, and Social Applications, pp. 59–74 (2010)
13. Urbieta, M., Escalona, M.J., Luna, E.R., Rossi, G.: Detecting Conflicts and Inconsistencies in Web Application Requirements. In: Harth, A., Koch, N. (eds.) ICWE 2011 Workshops. LNCS, vol. 7059, pp. 278–288. Springer, Heidelberg (2012)
14. Urbieta, M., Rossi, G., Distante, M., Ginzburg, J.: Modeling, Deploying, and Controlling Volatile Functionalities in Web Applications. International Journal of Software Engineering and Knowledge Engineering (IJSEKE) 22(1), 129–155 (2012)
15. van der Aalst, W.M.P., van Hee, K.: Workflow Management Models, Methods, and Systems. The MIT Press (2004) ISBN: 978-0262720465
16. WebSpec Language, http://code.google.com/p/webspec-language/
17. Wimmer, M., Schauerhuber, A., Kappel, G., Retschitzegger, W., Schwinger, W., Kapsammer, E.: A survey on UML-based aspect-oriented design modeling. ACM Comput. Surv. (CSUR) 43(4), 28 (2011)