Modelling adaptations requirements in Web workflows

M. Urbieta
LIFIA, Facultad de Informática,
UNLP, La Plata, Argentina
murbieta@lifia.info.unlp.edu.ar

G. Rossi
LIFIA, Facultad de Informática,
UNLP, La Plata, Argentina
Conicet
gustavo@lifia.info.unlp.edu.ar

S. Gordillo
LIFIA, Facultad de Informática,
UNLP, La Plata, Argentina
CiCPBA
gordillo@lifia.info.unlp.edu.ar

W. Retschitzegger
Department of Cooperative
Information Systems,
Johannes Kepler University Linz
werner.retschitzegger@jku.at

W. Schwinger
Department of Cooperative
Information Systems,
Johannes Kepler University Linz
wieland.schwinger@jku.ac.at

E. Robles Luna
LIFIA, Facultad de Informática,
UNLP, La Plata, Argentina
esteban.robles@lifia.info.unlp.edu.ar

ABSTRACT
Workflows play a major role in nowadays business and therefore its requirement elicitation must be accurate and clear for achieving the closest solution 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 to minimize development effort. In this work we present a model-driven approach for specifying Web workflows adaptations using a Domain Specific Language for Web application requirement called WebSpec. We present an extension to WebSpec based on Pattern Specifications for dealing with crosscutting workflow requirements by identifying tangled and scattered behaviour and reducing inconsistencies early in the requirement gathering phase. Using simple but illustrative examples we show the expressive power of the approach.

Categories and Subject Descriptors
D.2.1 [**Requirements/Specifications**]
H.4.1 [**Information System Applications**]: Office Automation - *Workflow Management*.
H.5.3 [**Information Interfaces and Presentation**]: Group and Organization Interfaces - *Web-based interaction*

General Terms
Algorithms, Documentation, Performance, Design, Languages, Theory, Verification.

Keywords
Requirements, adaptation, Model-driven paradigm, Web.

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[10], among others business requirements are examples of unexpected requirements that stress the whole applications' infrastructure.

iiWAS2012, 3-5 December, 2012, Bali, Indonesia.

In this paper we focus on the problems posed by those requirements that demand business processes to change according to the users' context. Depending on context variables like current date, payment method, active market campaign, accessing device, etc. the system may modify the underlying workflow model; this may imply executing a slightly different workflow version which support new requirements like discounts and free-shipping, or introduces new workflow steps like new forms to be filled, etc. 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 controller that orchestrates form's validation and workflow's navigation, and finally the business model must be modified for supporting new form's entities and fields.

As a motivation we use the checkout process in an e-commerce site; in order to buy some items, the user must follow a simple workflow comprising several steps such as selecting a product, choosing the wrapping configuration (regular or special for birthday), selecting the shipping address, and defining the payment method, etc. Suppose an unforeseen event such as a catastrophe happens that leads to a donation campaign. We may require the introduction of a new donation step in the purchase workflow, where users can choose between different pre-set amounts of money to donate. This change will require at least a set of modifications:

(i) implement a page that holds a donation form with its corresponding fields;

(ii) the corresponding step must be placed in the workflow and the workflow must be modified to be coherent;

(iii) new data needs to be stored and therefore we need to add persistence machinery for these data; and

(iv) navigation functionality must be upgraded 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 of the adaptation in the application 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 (AR from now on) are modelled is critical to assure that they correctly implemented.

To make matter worse, 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 workflow requirements. In the background of an Austrian highway control agency (ASFINAG), an agent (user) follows a workflow that specifies how to react to different situation that needs special treatment such as the existence of roadwork, traffic jam or fog on road. In this context, the workflow lets she decide whether to suggest an alternative path based on highway's status that can be taken in the nearest exit. On the other side, in an Austrian train control agency (LINZAG), a different workflow takes care of any train accident in order to inform passengers to walk in direction to closest location where they will be quickly assisted. Things complicates when both independent workflows face a critical situation at the same time but none of them is aware of the existence of the other workflow and how to proceed taking into account the whole context-situation. In Figure 1, a sequence of events corresponding to different transportation systems is depicted. In this example, in a foggy day (point 6) where there is a roadwork (point 1), the agent in charge of the highway management suggests to take an alternative path (point 3). If there is a train accident over suggested road, we face a deadlock because the highway agent suggests drivers to take an alternative path but it is blocked by a train accident. From the other viewpoint, train's passengers receive an inaccurate suggestion that recommends walking in direction of the jammed highway being impossible to receive a quick assistance as expected. In this case, the lack of context awareness support doesn't allow systems to provide a solution to a problem that is its reason for existing.

Unfortunately, workflows are not modelled to support this kind of situations; adaptations arrive once the workflow has been released.

When new concerns are unforeseen and unpredictable like Crisis Management or Volatile requirements[11], 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 extend our previous work[18] presenting 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 AR. 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 AR; later these models are used for implementing test suites that assess the final application behaviour. This work complements [18] with some novel contributions: a set of improvements on the WebSpec language, a supporting tool for Pattern Specification based models and a detailed description of the weaving process.

The rest of the paper is structured as follows: in Section 2, we discuss some related work; in Section 3, we present some background themes; in Section 4 we present our model-driven approach for modelling workflow changes in Web Application; in Section 5, we introduce an extension for WebSpec that uses Pattern Specification and stereotypes; in Section 6, we present a catalogue of common adaptations present in Web workflows; in Section 7, we present a brief description of current supporting tool; and finally in Section 8 we conclude and discuss some further work we are pursuing.

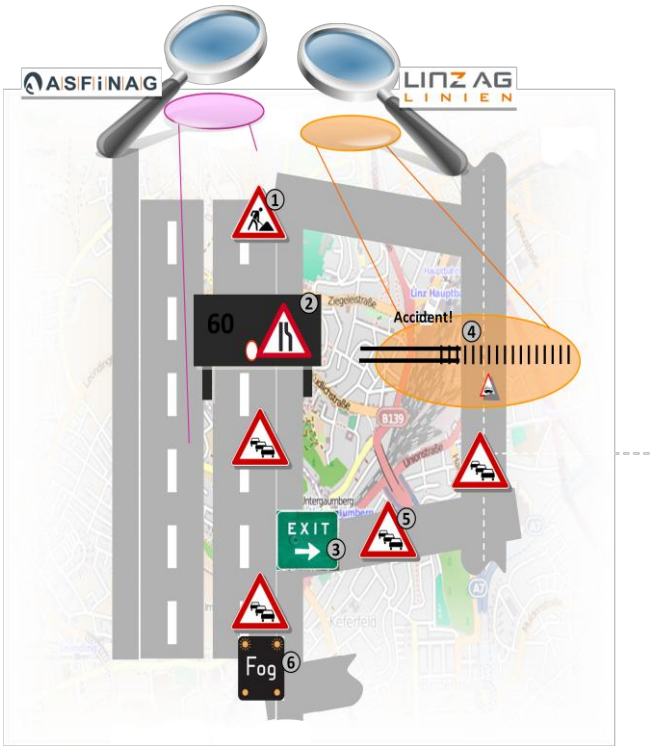

Figure 1. Workflows' deadlock schema

2. 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 their reasons and resolution that later can become part of the next workflow instantiation. Although this attempt will help to implement awareness in 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 provides an 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, joinpoint 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)

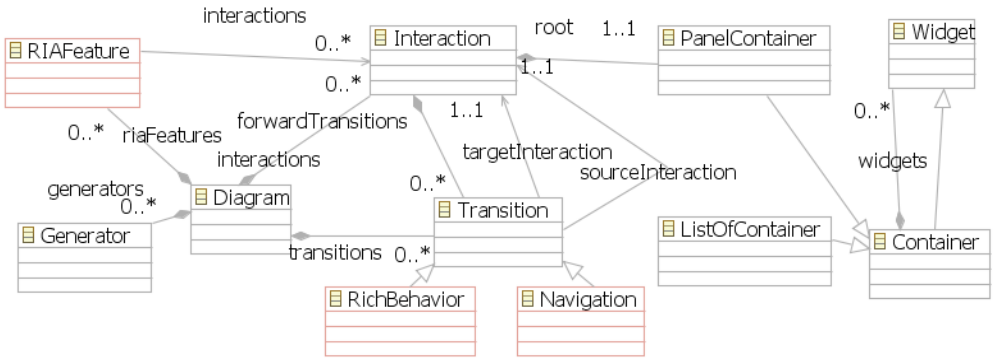

Figure 2. Simplified WebSpec meta-model

where crosscutting cannot be identified and checked with stakeholders.

We are not aware about any approach that allows identifying workflows and specifying their 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.

3. 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.

We will adopt a workflow's definition presented in [19] 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.

3.1 WebSpec

WebSpec [12] is a visual domain specific language for representing Web applications requirements; 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. Figure 2 shows a simplified WebSpec metamodel. 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).

In Figure 3, 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 (see [12] for further details).

WebSpec has a supporting tool [20] 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.

3.2 Pattern specification

Pattern Specifications (PSs) [8] 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.

4. WORKFLOW REQUIREMENTS MODELLING

Next, we present our approach to identify, and design adaptive requirements in Web Workflows. The approach is based on the idea that any adaptive requirement must be treated as a first–class citizen; we consider these requirements as belonging to a separate

Figure 3. WebSpec scenario for Checkout process

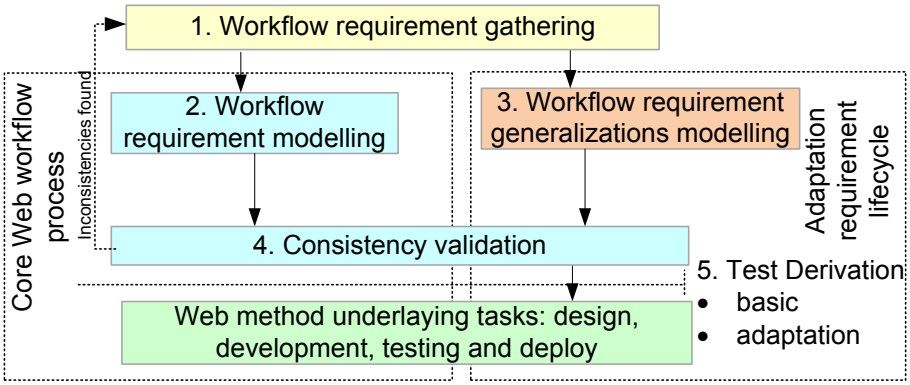

Figure 4. Overall schema for Web workflow requirement modelling

concern[1] [14] allowing us to isolate, model and later compose both core application workflow and adaptive requirements. To make this presentation thorough we first describe the general approach to model Web workflow requirements using WebSpec. The approach comprises a set of steps that are depicted in Figure 4 and described next. Step 3 below deals specifically with introducing AR:

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. Using a requirement DSL, tasks such as tests derivation and scenarios simulations can be automated easily. In this work, we selected WebSpec as the requirement DSL.

Step 3: *Workflow requirement generalizations modelling*. Base Workflow changes (e.g. adaptations) 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 structural and navigational conflicts are detected. These 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 on this process see [16]. In order to check consistency of AR, base requirements are composed with AR (following Pattern Specification semantics) giving as result a complete model that is validated. Since a given AR can be generic and so it can have several points of instantiation into the base diagram, the validation consistency procedure will only use specified binding configuration between base and adaptation model's elements.

Step 5: *Test derivation*. In this step, tests for the composition of the traditional WebSpec diagram and the WebSpec PS extension are generated producing tests that allow validating the final Web Application. Generated code is based on Selenium tool [13] allowing automating Web browsing task based on WebSpec requirements. This also allows assessing the set of requirements with users by using simulations in the early stages of UI mocking. The same tests are used later in the testing phase of the software development process. When deriving tests for AR, the same binding configurations used in step 4 are taken into account. That is, the test derivation process uses an internal model where the base model was only enhanced on those points that specify a binding configuration.

For more information about the overall approach see [18].

5. CROSSCUTTING BEHAVIOUR IN WEB WORKFLOWS

In this section, we introduce our WebSpec extension for modelling Web workflow adaptations and illustrate the main concepts with an example.

5.1 Modelling adaptations using WebSpec

WebSpec provides a powerful language for describing user's interaction of a Web application. 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 until the manager authorizes to continue, or landmarks-like behaviour where a given sub-workflow can be accessed from steps belonging to a main workflow. This restriction increases the size and complexity of diagrams, and the effort to document the requirement. So, we propose the use of Pattern Specifications 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 each one of model elements

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

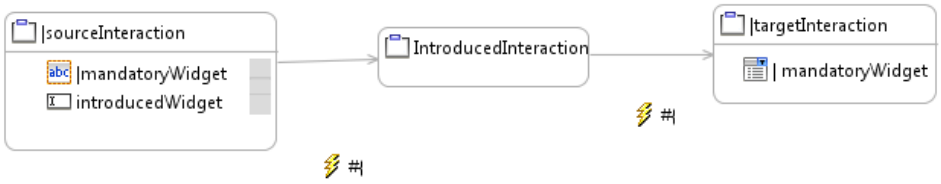

Figure 5. Introducing interactions and elements in a Workflow requirement

that plays the roles of the PS satisfies required properties defined by the roles.

In Figure 5, a requirement that generalizes an interaction pattern defines two roles: */sourceInteraction* and */targetInteraction*. The */sourceInteraction* role (notice that the 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 the 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 5. Notice that a Role can be defined based on transitions for enhancing preconditions and actions of core transitions. 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 order to improve expressiveness of our language extension, we also introduce stereotypes inspired by MATA [21]. These stereotypes allow a precise description of Web workflows adaptations. The stereotypes that can be used in PS-based models are:

- «create»: applied to any model element, specifying the creation of an element. This tag is implicit in elements belonging to a PS-based model that are not a role.
- «delete»: applied to any model element, specifying the deletion of an element that matches.
- «context»: used for specifying those elements that are mandatory; it avoids creating an element inside., forcing it to match an element in the base diagram. This is an alternative to a role prefix.
- «regex»: used in a role definition for a regular expression[9] (regex) matching strategy. When the role name is annotated with this stereotype, a comparison operation between the role and the base element is performed by the role's regex expression instead of a literal comparison.

This last stereotype, helps reducing the system space elements to match with.

For example, in Figure 6, we show a generalization of Web application requirements that provides the option for donating when performing a Web workflow. This introduces a banner between two roles describing the donation goal and allows traversing towards a donation form.

5.2 Bindings computation

So far, we have defined how to specify adaptations in Web workflow requirements. In this section, we present how the elements matching PS-based model's roles are computed. We refer to each pair of matching base element (interaction or transition) and role element as a *binding*, and the set of *bindings* that satisfy a whole PS-based model as a *joinpoint*. We name these points as *joinpoint* inspired in the joinpoint concept of Aspect Oriented Programming (AOP)[7]. It is noteworthy that a PS-based diagram can have several valid *joinpoints* for a given base diagram.

In Figure 7, we present a Java like pseudo-code that summarizes the *joinpoint* computation. The algorithm has as a restriction that both base and PS-based model must have a starting *Interaction* defined.

The algorithm is quite straightforward because it implements a well-known backtracking solution. It aims at matching the whole PS-based model against a base one. The backtracking strategy provides facilities for generating combinations of pairs of base and role elements. The code shows exhaustive usage of getter and setter methods that corresponds to instance variables and relationships shown in Figure 2.

The most noteworthy aspect is the introduction of a stack for resuming interaction's transitions processing. When the matching

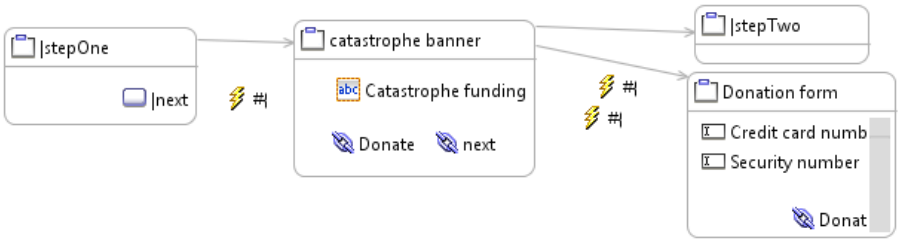

Figure 6. Donation requirement model using Pattern Specification

feasibility of role interaction is evaluated, roles' transitions and recursively the target of role's transition must be checked. This must be done in this way because to ensure a PS-model matching we must check whether all of its roles and transitions are matched. So, when a role transition is about to be processed, current role interaction is pushed in the stack for resuming the processing of its pending role transitions later on. For example, Figure 8 shows a "|X" interaction with two role transitions. When the algorithm is going to process the role transition from "|X" to "|Y" interaction, "|X" interaction is pushed for a later resuming. When the "|Y" interaction is finally processed, the stack is popped for resuming the "|X" interaction and so the pending transition from "|X" to "|Z" is processed.

```
function resolveJoinPoint(base, adaptationDiagram){
List joinpoints = new ArrayList ();
for (Interaction interaction : interactions)
  Joinpoint joinpoint = new Joinpoint(base,adaptationDiagram);
  Stack emptyStack = new Stack ();
  resolveInteractionJoinpoint(interaction,  adaptationDiagram.getStarting(),
  joinpoint, joinpoints, emptyStack);
}
function resolveInteractionJoinpoint (interaction, roleInteraction,
joinpoint, joinpoints, resumableInteractions)
    if (interaction matches roleInteraction)
      joinpoint.addBinding(interaction, roleInteraction);
      resolveTransitionJoinpoint(interaction, roleInteraction, joinpoint,
      joinpoints, resumableInteractions);
      joinpoint.removeLastInteractionBinding();

function resolveTransitionJoinpoint (interaction, roleInteraction,
joinpoint, joinpoints, resumableInteractions)
    List forwardRolTrans = roleInteraction.getRoleForwardTransitions();
    if (forwardRolTrans.isEmpty())
      if (joinpoint.isValid())  joinpoints.add(joinpoint.clone());
     else if (resumableInteractions.isNotEmpty())
        InteractionBinding pop = resumeableInteractions.pop();
        resolveTransitionJoinpoint(pop.getInteraction(),
        pop.getRoleInteraction(), joinpoint, joinpoints,
        resumableInteractions);
        else
          for (Transition roleInteractionTransition : forwardRolTrans)
            if ( interaction.getForwardTransitions().isNotEmpty())
              List matchingTrans =
                          matchingTrans (interaction,roleInteraction);
              if (matchingTrans.isNotEmpty())
                for (Transition transition : matchingTrans)
                  if (transition not bound in the joinpoint)
                    joinpoint.add(transition, roleInteractionTransition);
                    resumableInteractions.push(last stack's interaction);
                    resolveInteractionJoinpoint
                    (transition.getTargetInteraction(),
                    roleInteractionTransition.getTargetInteraction(),
                    joinpoint,  joinpoints, resumableInteractions);
                    joinpoint.removeLastTrasitionBinding();
```

Figure 7. Pseudo-code for computing joinpoint

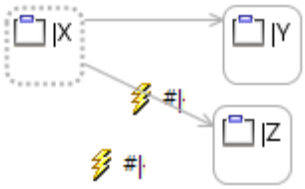

Figure 8. Simple PS-based diagram

For example, the execution of the algorithm presented in Figure 7 over the base diagram (simplified) in Figure 9 using the PS-model shown in Figure 8 will resolve four joinpoint:

- A bound to |X, B bound to |Y and C bound to |Z
- A bound to |X, B bound to |Z and C bound to |Y
- B bound to |X, D bound to |Y and E bound to |Z
- B bound to |X, D bound to |Z and E bound to |Y

Finally, depending on the stakeholders' needs, these joinpoints are interpreted based on the concepts presented in Section 4.2 for producing a woven model.

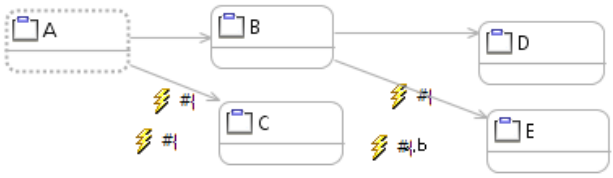

Figure 9. Simple base diagram with tree topology

In the case of the AR presented in Figure 6, this requirement can be instantiated on base requirement shown in Figure 3 using one of the available joinpoints:

- *Products* bound to |*stepOne* and Packaging bound to |*stepTwo*
- *Packaging* bound to |*stepOne* and *Delivery* bound to |*stepTwo*
- *Delivery* bound to |*stepOne* and *Order Status* bound to |*stepTwo*

Notice that there are several joinpoint available because the generality of adaptation model.

5.3 Discussion

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 [22].

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 helps 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. Adaptation characterization

Workflow adaptations present themselves often in a unique way but there is a basic set of common adaptations that can be characterized in order to reuse the knowledge gained as long as they are resolved.

We will present a basic context-awareness features that can be incorporated in workflows and we will model illustrative examples using Pattern Specification extension presented in previous section.

6.1 Generic workflow entity modification

Workflow changes compromise the definition of steps and transitions by introducing or removing new attributes, and guards among others. Although this is a generic requirement definition, it is a simple change present in most requirement adaptations.

For example, let's suppose a new promotion that applies a discount to products in the e-commerce site is available. For a given basic product detail (shown in Figure 10), the promotion requirements demand a proper user notification of promotion details such as discount amount, description and legal terms access. This kind of entity modification can be modelled using the Pattern Specification extension as shown in Figure 11. In this case, a discount and promotion description labels, as well as a link to the "*Promotion terms*" interaction are introduced into a matching *ProductDetail* interaction. Additionally, a new interaction is appended for presenting the promotion's terms.

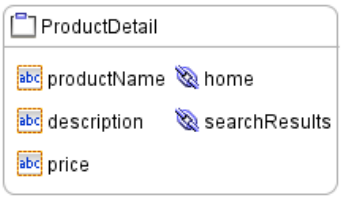

Figure 10. Product Detail interaction

6.2 Device awareness

Implicit decisions can be taken or steps added/removed in a workflow depending on the user's device. Mobile devices provide different facilities from traditional PC devices having access to resources like location information, cameras, sound recording, etc. but have different usage restrictions. For example, when driving a car, it is not easy to introduce any data in Mobile devices but register current GPS's location. In this scenario, a workflow can be aware of the user's context in order to simplify data gathering. If we want to report an urban incident by using a mobile device, the reporting workflow can sense the current mobile location avoiding entering that piece of information. In the other hand, when reporting by using a PC, the location and date must be filled out by the user.

For example "Reducing checkout process for smartphone" (shown in Figure 12) introduces enhancements over the main workflow; the "*input of delivery address*" task is avoided by using the closest registered address to the current location. In Figure 12, this requirement is modelled overriding the default navigation presented in Figure 3 where the specification of delivery information (*Delivery* interaction) is by-passed, and, instead, the *Order Status* interaction is exhibited after selecting *Packaging* configuration. This "by passing" is achieved by defining a transition that goes from *Packaging* 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 3.

6.3 Partner workflow awareness

There are situations in which workflows belonging to different organizations must interact with each other in order to satisfy inter- organizations needs defined by an embracing organization. This kind of workflows is called Inter-Organizational Workflows (IOC) [5]. In some cases, Web workflows that implement isolated business process could have failed taking into account the Inter-Organizational context, and so integration aspects such as share resources, tasks and business rules may not be well defined.

Being aware of the whole context may lead to a more effective workflow's goal satisfaction. This kind of inter-organization awareness requires a deep analysis for detecting shared concepts (resources, tasks, decisions, etc.) and requirement modelling (describing structure and states) that can result in a conflict of interest between workflows. For this task we propose to face inter-workflow adaptation using the Webspec PS extension to represent those integration aspects as first-class citizens.

Figure 11. Discount promotion enhancements

Figure 12. WebSpec diagram for "Reduced checkout process for smartphones" requirement

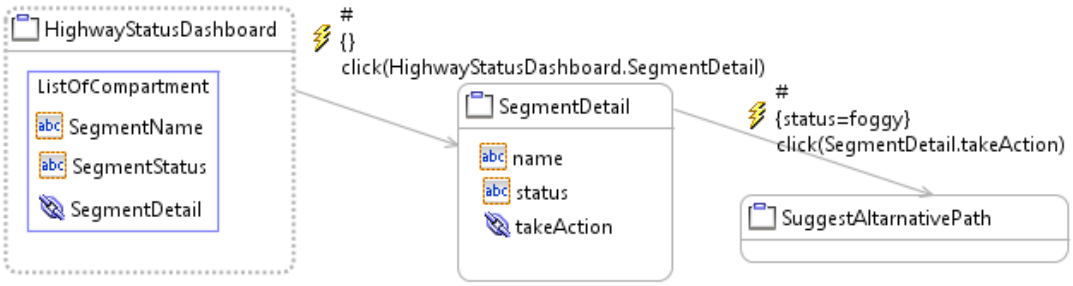

Figure 13. Highway decision making requirement for foggy days

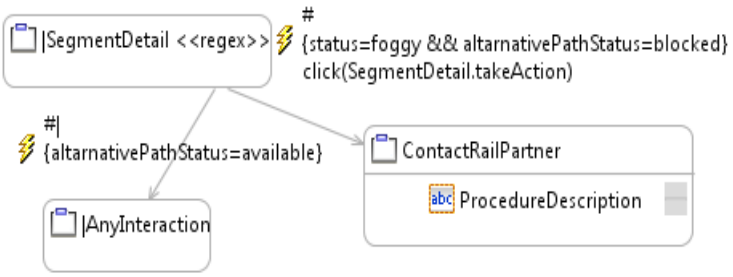

Figure 14. Railway workflow awareness adaptation

The example of Section 1 illustrates the situation where highway and train's street are shared concepts. If the ASFINAG's workflow suggests using as alternative path the Railroad crossing's street because there is fog along the highway, it would come to a conflictive decision when a train stops working blocking the railroad crossing's street and suggesting passengers to walk toward highway for assistance. For solving this situation, new workflow features should be modelled for enhancing the ASFINAG's workflow with a set of guards that check whether the street (resource under consideration) is available in the decision making step.

A basic Web workflow for Highway resources management is shown in Figure 13. In this case, it is possible to browse from a dashboard to distinct highway's segments. When a specific segment has the status "foggy", the system must suggest highway's drivers to take an alternative path. A railway partner aware adaptation for the workflow described in Figure 13, is shown in Figure 14; it takes into account a critical railroad crossing's street status (workflows shared resource). The *SegmentDetail* interaction is enriched with a new navigation that is browsed when *alternativePathStatus* is blocked; that is, a broken down train have blocked a railroad crossing. The original navigation for "froggy" status is enhanced with a predicate that evaluates the alternative path's availability keeping the requirement correct.

6.4 Temporal awareness

So far, we have shown how to introduce changes in Web workflows requirements models. Very often adaptation requirements are valid only during fixed periods of time. For instance, the Donation AR (shown in Figure 6) will be available from a specific date up to the completion of a goal, e.g. having collected a concrete sum of money. (See a more thorough analysis of this kind of "volatile" requirements in [11,17]).

When dealing with this kind of temporal awareness, changes in workflow are introduced to support business events which occur in a time frame and must be removed when that time frame is over. This sort of adaptations compromises the modification of workflow's states and transitions. For instance, new transitions can be introduced, removed or modified changing their guards.

To support this activation/deactivation process we use a specific DSL named Activation Rules [17]. The active period of an AR is defined using two rules: one rule that specifies when the AR will be active and another one defines when the requirement is not longer needed. Both rules are described by temporal and/or custom business events. When a given event expression is satisfied, a set of WebSpec scenarios are enabled/disabled. Next we present a template for a rule definition:

WHEN

(Event_Pattern_Expression)

THEN

(CONNECT | DISCONNECT)

SCENARIO 1, SCENARIO 2,..., SCENARIO N

For example suppose we are interested on enabling donation requirement (shown in Figure 6) when for a precise date up to donation's objective is achieved. The Activation Rule for enabling the donation looks like:

WHEN

*Time is *-May-25 23:59*

THEN

CONNECT

Donation Requirement

For disabling donation, we should use next Activation Rule.

WHEN

DonationCompleteEvent()

THEN

CONNECT

Donation Requirement

With these rules, adaptation requirement modelling is now complete taking into account temporal aspects.

7. SUPPPORTING TOOL

As part of this research work we have developed an extension for our WebSpec tool [20] that gives support to Pattern Specification concepts and stereotypes as described in Section 4.2. WebSpec is a tool built on top of the Eclipse platform[6]. This platform implements an OSGI architecture [15] where application's concerns are packaged as bundles that can be plugged in at any time. By taking advantage of this architecture, the WebSpec tool was developed providing, at first, the core set of building blocks that allow designing and validating models, deriving tests, and performing simulations. The extension provides the following new features:

- *WebSpec model composition*: WebSpec diagrams describe scenarios depending on user stories. After requirement gathering and analysis steps, these WebSpec models can be combined in order to provide a whole system view. This whole system must be consistent since the tool uses this resultant model to validate its consistency.
- *WebSpec model weaving:* this feature implements Pattern Specification support by processing WebSpec templates and introducing its improvements into base diagrams.

Next we describe WebSpec model weaving extension.

7.1 WebSpec model weaving extension

The weaving extension reasons over two diagrams: a base WebSpec and a PS-based one. A PS-based diagram describes an AR as a set of changes (introduction and modification of Interactions and Transitions) that must be applied in the base WebSpec diagram.

There is a *weaving processor* responsible for calculating the set of points in the diagram that the PS-based model matches. As a PS-based diagram is a template with several hooks, there can be several combinations of base model elements where PS-model can be instantiated. We can think each combination as an analogy of AOP's joinpoints.

When the tool takes as input the example presented in section 4.3, it will resolve and present the available joinpoints as shown in Figure 15. After computing the available joinpoints, the tool prompts user to choose which one will be used for the PS-based model instantiation.

Next, the *weaving processor* uses the selected endpoint for instantiating the PS-based model. In the instantiation process the following rules are applied:

1. If a role interaction (belonging to the PS-model) matches a given interaction and all of role widgets belonging to the role interaction match distinct base interaction's widgets, all of non-role widgets belonging to the role interaction are cloned into the base interaction. In this case, the base interaction is augmented with widgets. Rules 3 and 4 are applied over forward transitions that have as a source the role interaction.
2. If a Non-role interaction of a PS-model doesn't exist in base model, then it is cloned in the base model understanding it is an introduction.
3. Transitions and interactions belonging to the PS-model with the same name to one belonging to the base model, override the base version. Rule 1 is applied recursively over the transition's target interaction.
4. If a role transition matches one belonging to the base model, the base transition is augmented with the role transition's constraint and actions. Rule 1 is applied recursively over the transition's target interaction.

It is worth to note that these rules are evaluated over a given PS-model and their applicability was already validated in the first step executed by the weaving processor.

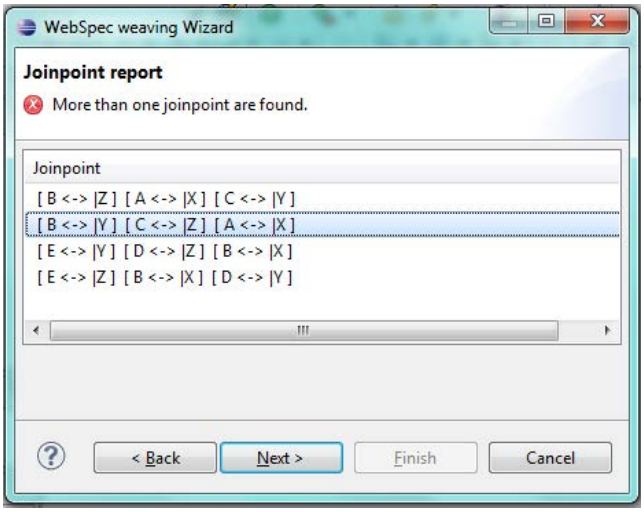

Figure 15. List of resolved joinpoint shown by the tool

8. CONCLUSIONS & FUTURE WORK

In this work we presented a novel approach for modelling Workflows in Web applications for both traditional requirements as well as crosscutting ones. By using WebSpec diagrams, workflows were modelled as a set of interactions representing their steps, and transitions for defining interactions' connections. In this work, we proposed a PS extension for WebSpec and a set of new stereotypes that allow easily specifying crosscutting workflow's behaviour. On the other hand, the approach allows modelling requirements associated to Inter-Organization Workflows [5] that, as we are aware, do not have supporting tools. We have implemented a set of tool extensions that supports the approach. The approach allows composing diagrams based on PS with base WebSpec diagrams. From the outcome woven model, a set of test can be generated in order to assess the implemented Web workflow.

We plan to perform assessments to validate our ideas and measure benefits of its application exploiting WebSpec's features such as test generations and simulations.

UML class diagrams and business process models can be sketched from WebSpec diagrams. 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.

We plan to compare the outcome obtained from the requirement gathering tasks using our approach (based on Web requirement models) against traditional lexical software requirement specification. We will measure improvements by means of studying documentation's quality metrics such as ambiguity, completeness, correctness, achievement, traceability, among others. We plan to analyze the advantages in traceability between model elements since our model for formalizing requirements also allows deriving design models following a model driven approach. Finally, we are studying how Web workflow requirements can ease agile development by inferring required story points[4] for a given requirement.

9. ACKNOWLEDGMENTS

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. It also received funding from the Argentinian Mincyt in the context of Argentina-Austria Cooperation and Agencia's PICT 2187.

10. REFERENCES

1. Adams, M., Edmond, D., ter Hofstede, A. 2003. The Application of Activity Theory to Dynamic Workflow Adaptation Issues. In: *PACIS 2003 Proceedings*. Paper 113.
2. Adams, M., ter Hofstede, A. H. M., Edmond, D., M. P. van der Aalst, W. 2006. Worklets: a service-oriented implementation of dynamic flexibility in workflows. In *Proc. of the 2006 Confederated international conference on On the Move to Meaningful Internet Systems: CoopIS, DOA, GADA, and ODBASE*, pp. 291-308.
3. Charfi, A. 2007. Aspect-Oriented Workfow Languages:AO4BPEL and Applications. In *Phd thesis, Fachbereich Informatik, der Technischen Universit at Darmstadt*. http://d-nb.info/985111321.
4. Cohn, M. 2009. *Succeeding with Agile: Software Development Using Scrum (1st ed.)*. Addison-Wesley Professional.
5. Divitini, M., Hanachi, C., Sibertin-Blanc, C. 2001. Inter-organizational workflows for enterprise coordination. In *Coordination of Internet agents*. Springer-Verlag, London, UK pp. 369-398.
6. Eclipse OSGI, http://www.eclipse.org/osgi/, Accessed August 2012.
7. Filman, R., Elrad, T., Clarke, S., Aksit, M. 2004. Aspect Oriented Software Development. Addison Wesley.
8. France, R., Kim, D., Ghosh, S., Song, E. 2004. A UML-Based Pattern Specification Technique. In *IEEE Transactions on Software Engineering*, Volume 30(3).
9. Goyvaerts, J., Levithan, S. 2009. *Regular Expressions Cookbook*, O'Reilly Media, ISBN 978-0-596-52068-7.
10. Luecke, R. 2004. *Crisis Management: Master the Skills to Prevent Disasters*. Harvard Business Press Books. ISBN 978-1591394372.
11. Moreira, A., Araújo, J., Whittle, J. 2006. Modeling Volatile Concerns as Aspects. In *CAiSE*, pp. 544-558.
12. Robles, E., Garrigós, I., Grigera, J., Winckler, M. 2010. Capture and Evolution of Web Requirements Using WebSpec. In *ICWE 2010*, pp. 173-188.
13. *Selenium*. http://seleniumhq.org/ Access August 2012.
14. Sutton, S., Rouvellou, I. 2002. Modeling of Software Concerns in Cosmos. In *Proc of ACM Conf. AOSD 2002*, pp. 127-133, ACM Press.
15. *OSGI*, http://www.osgi.org.
16. Urbieta, M., Escalona Cuaresma, M.J., Robles Luna, E., Rossi,G. 2011. Detecting Conflicts and Inconsistencies in Web Application Requirements. In *ICWE Workshops*,278-288.
17. Urbieta, M., Rossi, G., Distante, M., Ginzburg, J. 2012. Modeling, Deploying, and Controlling Volatile Functionalities in Web Applications. In *International Journal of Software Engineering and Knowledge Engineering* (IJSEKE), vol. 22(1), pp. 129-155.
18. Urbieta, W., Rossi, G., Gordillo, S., Schwinger, W., Retschitzegger, W., Escalona, M. J. 2012. Identifying and Modelling complex Workflow Requirements in Web Applications. In *proc. ICWE'12,* Springer-Verlag, Berlin, Heidelberg, In press.
19. van der Aalst,W., van Hee, K. 2004. *Workflow Management Models, Methods, and Systems*. The MIT Press, ISBN 978-0262720465.
20. WebSpec Language, http://code.google.com/p/webspec-language/. Access August 2012.
21. Whittle, J., Moreira, A., Araújo, J., Rabbi, R., Jayaraman, P., Elkhodary, A. 2007. An Expressive Aspect Composition Language for UML State Diagrams. *Int. Conference on Model Driven Engineering, Languages and Systems* (MODELS), Springer.
22. Wimmer, M., Schauerhuber, A., Kappel, G., Retschitzegger, W., Schwinger, W., Kapsammer, E. 2011. A survey on UML-based aspect-oriented design modeling. *ACM Comput. Surv*. (CSUR) 43(4):28.