Capturing and Validating Personalization Requirements in Web Applications

Esteban Robles Luna, LIFIA, UNLP, Argentina
Irene Garrigós, DSLI, UA, Spain
Gustavo Rossi, LIFIA, UNLP, Argentina
Agenda

• Motivation & Background.
• Specifying personalization requirements with WebSpec.
• Deriving the Personalization Model.
• Implementation.
• Conclusions & further work.
Agenda

• Motivation & Background.
• Specifying personalization requirements with WebSpec.
• Deriving the Personalization Model.
• Implementation.
• Conclusions & further work.
Motivation

- Complex Web applications tend to **change fast** because:
  - Different kind of users → different goals and need.
  - Evolution of requirements.
  - Economic aspects.

- Special care must be put in requirements during the development cycle to develop successful web applications (McDonald et.al, 2001, Lowe et.al, 2003).
Context

• As applications change fast:
  • Requirements definition is shallowly considered OR it is consider only in the beginning of the development.
  • Requirements related with Personalization are shallowly documented.

• Problems
  • Difficult to implement the requirement.
  • Impossible to validate whether the requirements have been correctly implemented in the application.
<table>
<thead>
<tr>
<th>Agenda</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Motivation &amp; Background.</td>
</tr>
<tr>
<td>• Specifying personalization requirements with WebSpec.</td>
</tr>
<tr>
<td>• Deriving the Personalization Model.</td>
</tr>
<tr>
<td>• Implementation.</td>
</tr>
<tr>
<td>• Conclusions &amp; further work.</td>
</tr>
</tbody>
</table>
Approach using WebTDD

WebTDD

1. Create Mockups + WebSpec
2. Derive Interaction Tests
3. Run Tests against Application
4. Create/Improve:
   - Presentation Models
   - Navigation Models
   - Data Models
   - User/Personalization Models
5. Generate Application
6. Run Tests against Application
WebSpec - Introduction

• WebSpec is a DSL (Domain Specific Language) that allows specifying navigation, interaction and UI aspects in Web applications.

• It has a defined syntax and semantics.

• Instead of using a textual version of the language, we use a mixed version of graphical elements and text.
A diagram defines a set of scenarios that must be satisfied by the application.

A scenario represents a sequence of actions that the user can execute and must be satisfied by the application.
WebSpec - Basic concepts

- WebSpec has two key concepts:
  - An *interaction* represents a point where the user can interact with the application by using its interface objects (widgets).
  - A *navigation* from one interaction to another can be activated by executing a sequence of actions such as: clicking a button, adding some text in a text field, etc.
WebSpec - Invariants

• We can specify invariants on each interaction to specify which properties must be satisfied by the application when the user reach that interaction point.

• Example:

```
Cart.total = ${price} && Cart.productSize = "1"
```

• Cart.total = ${price} && Cart.productSize = “1”
WebSpec - Personalization specification

- For specifying personalization we use a special variable `${user}` which holds information about the user activity.

- We use the prefix DD for domain dependent information, e.g.: `${user}.DD.booksBought`
Agenda

• Motivation & Background.
• Specifying personalization requirements with WebSpec.
  • Example.
• Implementation.
• Conclusions & further work.
Each scenario starts in an initial point and is build following the navigations.
E-commerce application - Offer discounts

• The user can perform different actions which are specified in the navigations.

• For instance, the user can navigate to the book list and go back to the Home page.
• In the book list the user can navigate to a book details page.

• From the book detail we can navigate to the cart. We store how many books the user has bought in the following sentence:

\[
\text{${user}.DD.booksBought:=${user}.DD.booksBought + 1}
\]
Finally, we specify when the discount must be shown using the books bought variable in the BookDetail page:

\[ \text{BookDetail.discount.visible} \leftrightarrow (\$\{\text{user}\}.\text{DD.booksBought} \geq 2) \]
Agenda

• Motivation & Background.
• Specifying personalization requirements with WebSpec.
• Deriving the Personalization Model.
• Implementation.
• Conclusions & further work.
• Once the personalization requirements have been specified and the tests have been generated, we focus on how to automatically derive concrete software artefacts that implement the personalization functionality from the personalization requirements.

• In this way, the mismatch between requirements and the developed application is avoided.

• The generation of such software artefacts leads to an application that satisfies the personalization requirements expressed in the WebSpec diagrams.
Generating the PM

• In this case, the software artefacts generated from the personalization requirements are personalization rules. We have chosen to specify these rules using the PRML (Personalization Rules Modelling Language) language.

• PRML is a rule-based high level language devised to specify personalization in an orthogonal way upon Web applications, independently of the underlying methodology.

• PRML has been successfully used in several Web methodologies and applied to several Web systems and an engine to perform and validate these rules has been implemented.
Generating the PM

• By automatically generating the personalization model, we provide the designer a first set of personalization rules that he can refine or modify later.
• This helps avoiding many manual errors and inconsistencies.
• In order to transform WebSpec diagrams into PRML rules, we use the MOF 2.0 Query/View/Transformation language (QVT)
From a diagram we can obtain a set of PRML rules that set a start point for the development of the personalization functionality.

PRML Rules are obtained using QVT between WebSpec and PRML’s metamodels.

Set content QVT rule:

```
u = webSpecExpToPRMLExp(e)
```
For example the diagram generates 2 PRML rules:

  `setContent(UM.User.booksBought, UM.User.booksBought + 1)`
  endWhen
The invariant in the BookDetail interaction defines a new PRML rule:

**Invariant:**
BookDetail.discount.visible \(\iff\) \((${\text{user}}).\text{DD.booksBought} \geq 2\)

**PRML rule:**
When LoadElement.BookDetail(NM.Book book) do
  If (UM.User.booksBought >= 2) then
    book.Attributes.selectAttribute(discount)
  endIf
endWhen
Because WebSpec is formally defined we can automatically derived a set of tests.

To make the validation independently of the development process we derive interaction tests.

These tests perform the same actions that a user would do it directly in a Web browser and also allow us to assert that some properties hold on a page.
Agenda

• Motivation & Background.
• Specifying personalization requirements with WebSpec.
• Deriving the Personalization Model.
• Implementation.
• Conclusions & further work.
WebSpec Eclipse plugin

- WebSpec has been implemented as an Eclipse plugin allowing an easy integration with different languages.

- We have used **EMF** and **GMF for the editors** and **Sable** for parsing the text (http://sablecc.org/).

- Some aspects of the UI of the editors were configured to improve their usability.
Agenda

• Motivation & Background.
• Specifying personalization requirements with WebSpec.
• Deriving the Personalization Model.
• Implementation.
• Conclusions & further work.
Conclusions

• We have presented the main problems of personalization requirements in Web application development.

• As a proposal to resolve those problems we have presented WebSpec a multipurpose requirement artifact for Web applications.

• We have shown its definition, the derivation of PRML rules and the derivation of interaction test for requirement specification.
Further work

- Field experiences.
- Improve the derivation process to the user model and PRML rules.
- Define a *lite* version to be used by customers.