Managing Volatile Requirements in Web Applications

Matías Urbieta, University of La Plata - Argentina
Damiano Distante, Unitelma Sapienza University - Italy
Gustavo Rossi, University of La Plata - Argentina
Wieland Schwinger, Johannes Kepler University - Austria
Volatile Requirements

• Web applications are characterized by continuous evolution.

• Evolution might involve the satisfaction of unforeseen temporary requirements which arise after the application has been deployed, remain valid for a certain period of time after which they are removed to be reactivated afterwards or to disappear forever.

• We call such requirements “Volatile Requirements” and the associated functionalities “Volatile Functionalities”.

• Examples:
  • Fundraising campaign in response to a catastrophe.
  • Special marketing campaign in ecommerce websites.
  • Special offers for Christmas celebration and back to school seasons.
  • etc.
Example – Video and St. Valentine Reminder Complementing the Page Presenting a CD in Amazon

Norah Jones’s CD

You Save: $3.99 (47%)


Want it delivered
minutes, and choose
next 2 hours and 45
mns

Fishbowl Performance Video

Free Audio Stream: Norah Jones’s song ‘Thinking About You.’ Listen now (requires Windows Media Player).

St. Valentine Reminder

Norah Jones shares her favorite music with Amazon customers. See all artists’ picks on our Music You Should Hear page.

Discover more great Blue Note artists.

A special deluxe version of Not Too Late is also available.

You can also watch an Amazon.com exclusive video of Dave Koz performing the beloved song “After”.

Find perfect Valentines for sweethearts, friends, and family, as well as videos and artists’ picks, in our Valentine’s Gifts for Music Lovers store.

Customer image from Kevin S.

See all 3 customer images

Share your own customer images
Code Tangling and Related Maintenance Issues

In order to have the video and the St. Valentine concerns inserted on the CD interface, we need to edit the interface code and insert the new functionality at the right place.

As a consequence, the code belonging to the core concern has knowledge on volatile concerns, and therefore both concerns (core and volatile) are tangled.

Now suppose the volatile functionality needs to be removed, and consider that it involves also other design and implementation artifacts than user interface...
Our Approach – Key Concepts

• **Volatile requirements as first-class requirements**
  • We address them at design level before going down to the implementation level

• **Obliviousness**
  • Core concerns are oblivious of volatile functionalities. The design and implementation of the core’s functionalities are separated from the ones corresponding to volatile functionalities

• **Seamless integration**
  • We want volatile functionalities to be integrate with core functionality in a seamless way, in order to allow:
    • removing them in an easy way
    • composing them using a transformation language
  • Inspired on well-known techniques for advanced separation of concerns
Our Approach - Core and Volatile Models
### Our Approach - Integration Solutions

Our approach to integration solutions involves weaving concerns across different layers of the software architecture. Each layer has specific modeling and weaving solutions tailored to its needs.

<table>
<thead>
<tr>
<th>Layer</th>
<th>Modeling</th>
<th>Weaving Models</th>
<th>Concern Implementation</th>
<th>Concern Weaving Implementation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conceptual</td>
<td>UML class Diagram</td>
<td>Inversion of Control</td>
<td>Commands and Decorators</td>
<td>---</td>
</tr>
<tr>
<td>Navigational</td>
<td>OOHDM Navigation Diagram</td>
<td>Navigational Affinities</td>
<td>Navigational Nodes</td>
<td>Navigational Affinities (XML configuration file)</td>
</tr>
<tr>
<td>Interface</td>
<td>ADV and ADV-Chart</td>
<td>ADV Integration Specification</td>
<td>XML-based Documents</td>
<td>XSL Transformations</td>
</tr>
</tbody>
</table>

* Each technology has its own runtime weaving engine

---

Let’s focus on the Interface Layer
Specifying Interfaces with Abstract Data Views (ADVs)

- In OOHDM, the user interface is specified using Abstract Data Views (ADVs) which support an object-oriented model for interface objects.
- An ADV is defined for each node class to indicate how each node attribute or sub-node will be presented to the user.
- An ADV can be seen as an Observer of the node expressing its perception properties.
Weaving Web Interface Designs with ADVs

- The idea of our approach:
  - Core and volatile interfaces (at the ADV and implementation layers) are woven by executing an integration specification (XML file), which is realized using XSL transformations or style sheets.

Integration Specification

IntegrationFor: Concern name. affinity name
Target: ADV target name
Add: ADV source name | Insertion Specification
Relative to: ADV name
Position: [above | bottom | left | right]
Weaving Web Interface Designs

**ADV CD**

<table>
<thead>
<tr>
<th>Picture</th>
<th>Title: string</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cover: bitmap</td>
<td>Performer: String</td>
</tr>
<tr>
<td></td>
<td>ListPrice: String</td>
</tr>
</tbody>
</table>

ADV addToShopCart (asButton)

ADV CustomerReviews

**Integration Specification**

Integration: NorahJonesIntoCd
Target: ADV CD
Add: ADV Fishbowl
Relative to: CD.CustomerReviews
Position: above

ADV Fishbowl

ADV Video

Play STOP
Example XSL Transformation 1

XSL transformation that inserts the Fishbowl Video Performance component above the Customer Review row in a JSP Page of a CD

• The XPATH expression (point-cut) `//tr[contains(.,'Customer Reviews')]` refers to the table row containing the text "Customer Reviews"

• The template (advice) leaves the existing elements of that row unchanged and inserts above a new one with the Fishbowl element
Fig. 12. CD stylesheet and CDNode instance.
Managing Volatile Requirements’ Life Cycle

An activation rule looks as follows:

WHEN
   (Event_Pattern|Expression)
THEN
   (CONNECT | DISCONNECT)
   Concern concern_Name
   NAV_Affinity Affinity_Name
   UI_Integration Integration_Name

To specify the volatility pattern of a VF (i.e., when a VF has to be activated and when has to be deactivated) we use Activation Rule

WHEN
   Time is *-Apr-14 00:00
THEN CONNECT Concern MothersDay
   NAV_Affinity MothersDay, UI_Integration MothersDayHomePage

WHEN
   Time is *-May-14 23:59
THEN
   DISCONNECT Concern MothersDay
   NAV_Affinity MothersDay, UI_Integration MothersDayHomePage
The CAZON Volatile Functionality Integration and Management Framework
Conclusions

• Volatile functionalities have to be treated as a first-class entities since the moment they are gathered

• Core and volatile UI models can be designed separately and then obliviously composed

• The proposed approach provides solutions to cover the whole life-cycle of volatile functionalities in Web applications

• Our proposal makes it possible to introduce new volatile functionalities in Web applications “on the fly”, and enables non-technical people to control their activation rules at runtime
Future Work

• Integrating our approach in model-driven Web engineering methods other than OOHDM

• Conducting assessments studies to measure benefits of our approach

• Incorporating volatile functionality into RIAs and dealing with more complex crosscutting concerns
Q & A