Migrating Multi-page Web Applications to Single-page Ajax Interfaces

Ali Mesbah
Arie van Deursen

Software Engineering Research Group
Delft University of Technology
Classical Web Applications

Every state change implies:

- Client sends request to Server
- Server builds entire page
- Server sends new page to Client
- Client refreshes the entire page
Web Applications with Rich Interaction

Recent trend: rich user interaction on web

- Google suggest / gmail, Flickr, Yahoo! Mail

Ajax: “Asynchronous JavaScript And XML”

- Asynchronous server communication using XMLHttpRequest
- Standards based (HTML, CSS, XML)
- Dynamic display using DOM
- JavaScript engine binds everything
**Classic Web Application Model (Synchronous)**

**Client**
- User activity
- Data transmission
- System processing
- Data transmission
- User activity

**Server**
- System processing
- Data transmission
- User activity

**Ajax Web Application Model (Asynchronous)**

**Client**
- Browser UI
- User activity
- Input
- Display
- Input
- Display
- Client-side processing

**Ajax Engine**
- Data transmission
- Data transmission
- Data transmission
- Data transmission
- Data transmission

**Server**
- Server-side processing
- Server-side processing
- Server-side processing
- Server-side processing
Ajax is here to stay

After Ajax was coined in February 2005:

• an overwhelming number of articles appeared
• numerous libraries and frameworks written
• many web applications have adopted one or more of the ideas underpinning Ajax

Despite all the attention:

• there is a lack of a precisely described set of architectural formalisms for Ajax
• no support for migration
SPIAR-based Architectural View

Processing View

Client Browser

Rep. Model

△update

event △update

SPUI

△update

HTTP

△C

△S

Server Application

Decoder

△update

Encoder

Com. Model

△update invoke

Processor
“Ajaxification”

- Analyze current (multi-page) user interaction
  - Create scenarios
  - Extract and simplify Navigational Path
  - Identify candidate UI components
- Devise an intermediate UI model
- Transform to target models

- Problem/Focus: *how to identify appropriate candidate single-page components from a page-sequence interface web application?*
Ajaxification Process

1. Classic Web Application
   → Nav. Model Extraction
   → Nav. Model

2. UI Component Model Identification
   → Candidate UI Components

3. Single-page UI Model Definition
   → Single Page UI Model

4. Target UI Model Transformation
   → Target UI Model e.g., BXXML, GWT

5. Generate
   → Ajax Interface

Developer

Single Page UI Metamodel
Page Classification Notions

Two pages are the same if they have:

- *Textual Identity:* exactly the same HTML code,
- *Syntactical Identity:* exactly the same structure, ignoring the text between tags, according to a comparison of the syntax trees,
- *Syntactical Similarity:* similar structure, according to a similarity metric, computed on the syntax trees.
Shortcoming

<table class="product">
  <tr>
    <td>Chow</td>
    <td>550.95</td>
  </tr>
  <tr>
    <td>Poodle</td>
    <td>300</td>
  </tr>
  <tr>
    <td>Bulldog</td>
    <td>400</td>
  </tr>
  <tr>
    <td>Dalmatian</td>
    <td>600</td>
  </tr>
</table>
Schema-based Clustering

Classic Web application

Retrieve pages

HTML pages

For each page

Page

Search Nav Links

Extract Schema

Page

Levenshtein Edit Distance

Page Links

Detect Schema Clones

Pair Page Clones

Cluster Clones

Clusters
Component Identification

- Extract Schemas of pages
- Levenshtein Edit Distance to detect Clones
- Transitive Closure of Clone relation to Cluster
- Cluster Reduction
- Differencing (A --> B) to find the *fragments*
- UI Element Identification

Results in:
- A list of candidate UI components along the navigational path
- Can be used to define a single-page UI model
## Results

<table>
<thead>
<tr>
<th>Method</th>
<th>Threshold</th>
<th>RCR</th>
<th>ICR</th>
<th>Precision (%)</th>
<th>Recall (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>STS</td>
<td>0.89</td>
<td>6</td>
<td>3</td>
<td>66</td>
<td>54</td>
</tr>
<tr>
<td>STS</td>
<td>0.91</td>
<td>9</td>
<td>4</td>
<td>69</td>
<td>82</td>
</tr>
<tr>
<td>STS</td>
<td>0.93</td>
<td>7</td>
<td>8</td>
<td>46</td>
<td>63</td>
</tr>
<tr>
<td>MMS</td>
<td>0.97</td>
<td>7</td>
<td>1</td>
<td>87</td>
<td>63</td>
</tr>
<tr>
<td>MMS</td>
<td>0.98</td>
<td>11</td>
<td>0</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>MMS</td>
<td>0.99</td>
<td>9</td>
<td>2</td>
<td>82</td>
<td>82</td>
</tr>
</tbody>
</table>

RCR: Relevant Clusters Retrieved  
ICR: Irrelevant Clusters Retrieved  
STS: Syntax Tree Similarity  
MMS: Meta-model Similarity
Concluding Remarks

• We have proposed a migration process
• Our classification approach is in two ways different:
  • schema-based similarity notion and clustering
  • find the clusters along the navigational path
• We have a static Client-side analysis approach

Ongoing/Future work:
• Dynamic analysis + larger case studies
• Server-side implications of migrating
• Model-driven engineering in support of migration/development (meta-models, transformations, …)
More Information
