Rapid Development of Spreadsheet-based Web Mashups

Woralak Kongdenfha\textsuperscript{1}, Boualem Benatallah\textsuperscript{1}, Julien Vayssière\textsuperscript{2}, Régis Saint-Paul\textsuperscript{3}, Fabio Casati\textsuperscript{4}

\textsuperscript{1}CSE, University of New South Wales, Sydney, Australia
\textsuperscript{2}SAP Research, Brisbane, Australia
\textsuperscript{3}CREATE-NET, Trento, Italy
\textsuperscript{4}DIT, University of Trento, Italy

22 April 2009
Integrated Applications

- Applications and data sources are autonomously developed and deployed
- Proprietary technologies (communication protocols, data formats, business and presentation logic)
- Costly development and maintenance of integrated applications especially in large and dynamic environments

SOA - Main Principles

- Compliance to standards (protocols and languages)
- Virtualisation (functionality, data)
- Composition (reuse)
Web2.0, Mashups

- SOA: connects systems by process-based composition
- Web 2.0 enables communities and sharing among users
- Mashups: Proliferation of creating websites by composing other websites as seen in ProgrammableWeb.com (novel form of composition)
- Key Enabler of Web 2.0 is the APIs (e.g., SOAP, REST)
- Mashups are typically developed by professional programmers (using AJAX and HTML)
- Needs to bring benefits of composition to end-users
Spreadsheet-based Web data mashups

- **Aim:** to bring the benefits of composition to end-users
- **How:** provide a spreadsheet-based framework for Web mashups development
- **Why:** spreadsheets is one of the most successful end-user programming environment [SSM05, EAKC06, NM90, JBB03]
  - flexible data model,
  - incremental approach for building fairly complex applications with immediate feedback,
  - analysis and manipulation functions spanning various application domains, e.g., financial, statistics.
Challenges

- Accessing and representing complex data within spreadsheets
- Synchronization of spreadsheet data and Web data
- Reuse-driven of spreadsheet-based Web mashups
- Easy-manipulation of complex data in spreadsheets
Contributions

- Mapping language between data services/spreadsheet presentation (EDBT'08)
- Component model for spreadsheet-based Web data mashups
- Spreadsheet-based Web mashup patterns
- Drag-and-drop reorganization of structured data in spreadsheets
Outline

- Background and Challenges
  - **SpreadATOR approach**
    - Mapping language between data services/spreadsheet presentation (EDBT’08)
    - Component model for spreadsheet-based Web data mashups
    - Spreadsheet-based Web mashup patterns
    - Drag-and-drop reorganization of structured data in spreadsheets
  - Related Work
  - Conclusion and Future Work
interposes an ER-based data model (data services [CN08]) between spreadsheets and heterogeneous data sources; thus enables uniform data access

proposes a mapping language between data service and spreadsheet data models; thus allows accessing and representing structured data within spreadsheets
Component model for spreadsheet-based Web data mashups

Services

Pull/Subscribe
Data access

Data View Component

Operations
- RefreshView()
- ModifyAttr()

Events
- ViewUpdated()

Interaction Component

Data to Tabular
Interaction rules
Tabular to Data
Interaction rules

Mapping
Specification

Inter-component
Synchronization

Presentation Component

Operations
- RefreshTable()

Events
- Modified()

Presentation

Render the presentation
Handles user interactions

W. Kongdenfha et. al.
1. Data view components

- allow accessing data from external data sources accessible through data services
- use a variant of the ER model to describe structure of underlying data sources
  
  
  GoogleNews(title, link, guid, category, pubDate, description)

- adapters are required when accessing data from non-data service sources
- two types of data view components: push component and pull component
- consist of a set of operations and events that allows its interactions with other components of the model
  
  operations: dv:RefreshView(), dv:ModifyAttr()
  
  events: D_ViewUpdated()
2. Presentation components

- allow displaying data in the tabular grid of spreadsheets
- **presentation specification**
  - describes how the contents of a data view component is mapped to the tabular display of spreadsheet
  - is modeled with the following constructs: ATTRIBUTE, VALUE, RECORD, SET, SHEET

- consists of a set of operations and events
  - operations: modifyVALUE(), insertATTR()
  - events: P_VALUEChanged(), P_VALUEDeleted()
3. Interaction components

- the role of interaction components is to synchronize data view and presentation components
- **Data-to-Tabular Mappings (DTM) definition**
  - binds contents of a data view component to presentation specification

\[
\begin{align*}
\langle C_x, C_y \rangle &= dv:\text{getObjects() } & & \text{(1)} \\
\langle C_x+k, C_y+1 \rangle &= dv:\text{getAttrName}(o_1,a_k) \ ; \ 0 < k < dv:\text{countAttrs}(o_1) & & \text{(2)} \\
\langle C_x+k, C_y+2+j \rangle &= dv:\text{getAttrValue}(o_j,a_k) \ ; \ 0 < j < dv:\text{countObjs( } \langle C_x, C_y \rangle ) \ , \ 0 < k < dv:\text{countAttrs}(o_j) & & \text{(3)}
\end{align*}
\]

- **Interaction rules** establish publish/subscribe relationships between data view and presentation components

**Presentation-data interaction rule**

\[
\begin{xml}
<\text{interaction} \ \text{publisher=“StockTable”} \ \text{event=“P_VALUEChanged”} \ 
\text{subscriber=“StockDataView”} \ \text{operation=“dv:modifyValue()” }>
\end{xml}
\]

**Data-presentation interaction rule**

\[
\begin{xml}
<\text{interaction} \ \text{publisher=“StockDataView”} \ \text{event=“D_ViewUpdated()”} \ 
\text{subscriber=“StockTable”} \ \text{operation=“ui:Refresh()” }>
\end{xml}
\]
Outline

Background and Challenges

SpreadATOR approach

- Mapping language between data services/spreadsheet presentation (EDBT’08)
- Component model for Web data mashups
- **Spreadsheet-based Web mashup patterns**
- Drag-and-drop reorganization of structured data in spreadsheets

Related Work

Conclusion and Future Work
Benefits of the proposed component model
- enables the synchronization of spreadsheet data and Web data
- enables different tabular presentations
- enables bulk loading

However, it is difficult for non-professional users to build such a component model

We propose the notion of spreadsheet-based Web mashup patterns
- each pattern captures default behavior for creating Web data mashups (default data access method, presentation, and interaction features)
- shifts the development efforts from scratch to reuse
Spreadsheet-based Web mashup patterns (characterization)

- CAISE’08: analysis of tabular presentations frequently found in spreadsheet applications
- WWW’09: captures common spreadsheet data presentations as presentation components

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>StockDetails</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Symbol</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Change in Volume</td>
<td>40.5%</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Change in Price</td>
<td>21.5%</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Symbol</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Symbol</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Symbol</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Symbol</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Summary of stock quotes

<table>
<thead>
<tr>
<th>Worksheet 1</th>
<th>Worksheet 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>1</td>
<td>Stockinfo</td>
</tr>
<tr>
<td>2</td>
<td>RATE</td>
</tr>
<tr>
<td>3</td>
<td>BEAT</td>
</tr>
<tr>
<td>4</td>
<td>THOR</td>
</tr>
<tr>
<td>5</td>
<td>LHCG</td>
</tr>
<tr>
<td>6</td>
<td>NTLS</td>
</tr>
</tbody>
</table>

Detail of a particular stock in another worksheet

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Selected stock’s ID</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>RATE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>BEAT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>THOR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>LHCG</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>NTLS</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fast access to detailed information of an item of interest (index pattern)

Stock details shown in a list (Repeater pattern)

A list of stocks and their related contact information organized in a nested structure (Hierarchical pattern)

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Contacts</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Prospect</td>
<td>ID</td>
<td>Name</td>
</tr>
<tr>
<td>3</td>
<td>RATE</td>
<td>3088299</td>
<td>56388</td>
</tr>
<tr>
<td>4</td>
<td>BEAT</td>
<td>1111111</td>
<td>444555</td>
</tr>
</tbody>
</table>

A list of news related to a particular stock in another worksheet (Relationship Index pattern)

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>NewsTable</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>title</td>
<td>link</td>
<td>pubDate</td>
</tr>
<tr>
<td>3</td>
<td>BBB warns...</td>
<td><a href="http://new">http://new</a>.</td>
<td>Mon,29..</td>
</tr>
<tr>
<td>4</td>
<td>Filmmaking..</td>
<td><a href="http://new">http://new</a>.</td>
<td>Wed,15..</td>
</tr>
<tr>
<td>5</td>
<td>American...</td>
<td><a href="http://new">http://new</a>.</td>
<td>Thu,16..</td>
</tr>
<tr>
<td>6</td>
<td>Sculpture...</td>
<td><a href="http://new">http://new</a>.</td>
<td>Fri,17..</td>
</tr>
<tr>
<td>7</td>
<td>RATE has...</td>
<td><a href="http://new">http://new</a>.</td>
<td>Wed,01..</td>
</tr>
</tbody>
</table>

Contact information with user-defined labels (Content pattern)

Side-by-side comparison of stock quotes (Table pattern)
1. provide URL of data service
2. browse schema of data service
3. specify query conditions
4. select a presentation component to display data
5. specify sorting condition on the query result
6. specify the number of content should be displayed at a time
7. specify how the data should be refreshed
Spreadsheet-based Web mashup patterns (Table Mashup Pattern)

Data view component
NasdaqStock(Symbol, Volume, Price, News)

Interaction component
\[
\begin{align*}
\langle C_x, C_y \rangle &= dv:\text{getObjects}() \\
\langle C_x+k, C_y+1 \rangle &= dv:\text{getAttribute}(o_1, a_k) \quad 1 \leq k \leq dv:\text{countAttrs}(o_1) \\
\langle C_x+k, C_y+1+j \rangle &= dv:\text{getAttribute}(o_j, a_k) \quad 1 \leq j \leq dv:\text{countObjs}(\langle C_x, C_y \rangle) \\
&\quad 1 \leq k \leq dv:\text{countAttrs}(o_j)
\end{align*}
\]

Formulas generated from Table Mashup Pattern

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td><a href="http://www.nasdaq.com">http://www.nasdaq.com</a>..</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>&lt;&lt;1.B2&gt;&gt;.[0]/_symbol</td>
<td>&lt;&lt;1.B2&gt;&gt;.[0]/_Volumn%Change</td>
<td>&lt;&lt;1.B2&gt;&gt;.[0]/_Price%Change</td>
</tr>
</tbody>
</table>

Presentation component

Contents displayed on the spreadsheet

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>StockTable</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>Symbol</td>
<td>Volumn%Change</td>
<td>Price%Change</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>RATE</td>
<td>40.5%</td>
<td>21.59%</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>BEAT</td>
<td>37.0%</td>
<td>32.17%</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>THOR</td>
<td>27.0%</td>
<td>18.78%</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>LHCG</td>
<td>21.4%</td>
<td>41.35%</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>NTLS</td>
<td>19.3%</td>
<td>34.68%</td>
</tr>
</tbody>
</table>
At runtime, when user edits contents of a VALUE cell, the interaction component performs the following tasks:

- Capturing event from the presentation component
- Invoking operations of subscribing components
Background and Challenges
Integration of spreadsheets with SOA
SpreadATOR approach
- Mapping language between data services/spreadsheet presentation (EDBT’08)
- Component model for Web data mashups
- Spreadsheet-based Web mashup patterns
- **Drag-and-drop reorganization of structured data in spreadsheets**
Example Application
Related Work
Conclusion and Future Work
Drag-and-drop reorganization of structured data

- Spreadsheet paradigm supports data manipulation
- When introducing new abstractions in spreadsheets, SpreadATOR also considers providing data manipulation operators for such abstractions
Related Work (1/2)

Existing tools in Spreadsheets (Excel)

- XML mapping tool, SQL importation, Web data importation, Analysis Services
- No support for complex types as they are converted to Excel’s supported types. Hence it is not possible to access the original document
- Lose of the nested structure of data as the tool denormalizes the hierarchical structure into flat-table
- Adherence to various tools dealing with external data importation in spreadsheets gives us hope that SpreadATOR will be adopted

Spreadsheet Data Presentations

- Lakshmanan’s Tabular Data Model [LSGK98, GLS96]
  - provides a powerful data model to present broad classes of tables (relation/spreadsheets)
  - the specification of presentations needs to be done manually
- Gencel [ACKE, EAKC06, EE]
  - a specific table presentation (since the purpose is to ensure spreadsheet correctness)
  - supports for the specification of presentation through templates
- SpreadATOR approach is in between the two by defining a small set of frequently used spreadsheet data presentations and provides support for their specifications
Related Work (2/2)

Inferring spreadsheet data presentation templates

- infer templates from existing spreadsheets [AE06] (enables data exportation)
- SpreadATOR’s data presentation templates can be adopted by such approach

Data-driven Web application development

- ASP.Net, PHP, JSF
  - provide widgets for data presentation
  - require web development skills
- WebML [CFB00]
  - provide presentation patterns for Web applications development
- SpreadATOR follows spreadsheet paradigm and provide interactive importation and presentation with immediate results
SpreadATOR brings the benefits of SOA (Data Services) to end-users by providing a spreadsheet-based framework for Web data mashups development.

The integration of data services and spreadsheets raise challenges in the difference in their data models.

By enabling spreadsheet cells to contain *object reference* and *queries*, SpreadATOR allows the user to access structured data within spreadsheets.

SpreadATOR proposes a new component model that supports the synchronization between spreadsheet data and Web data.

SpreadATOR introduces a set of common spreadsheet-based Web mashup patterns that simplifies mashups development by shifting the development efforts from scratch to that of reuse.
Future Work

- **Presentation components**
  - We observed that Web data are frequently displayed using visualization components such as timeline, maps, etc.
  - We are investigating how to integrate common visualization components into SpreadATOR, while preserving spreadsheet paradigm

- **Composition patterns**
  - SpreadATOR currently focuses on the integration at the data layer
  - To enable service composition within spreadsheets, more complex patterns such as workflow patterns need to be considered
Robin Abraham, Irene Cooperstein, Steve Kollmansberger, and Martin Erwig.
Automatic generation and maintenance of correct spreadsheets.
*Proc. ICSE'05.*

Robin Abraham and Martin Erwig.
Inferring templates from spreadsheets.

Stefano Ceri, Piero Fraternali, and Aldo Bongio.
Web modeling language (webml): a modeling language for designing web sites.

Pablo Castro and Anil Nori.
Astoria: A programming model for data on the web.

Martin Erwig, Robin Abraham, Steve Kollmansberger, and Irene Cooperstein.
Gencel: a program generator for correct spreadsheets.

Gregor Engels and Martin Erwig.
Classsheets: automatic generation of spreadsheet applications from object-oriented specifications.
In *Proc. ASE ’05.*

Marc Gyssens, Laks V. S. Lakshmanan, and Iyer N. Subramanian.
Tables as a paradigm for querying and restructuring (extended abstract).
Simon Peyton Jones, Alan Blackwell, and Margaret Burnett.
A user-centered approach to functions in excel.

Laks V. S. Lakshmanan, Subbu N. Subramanian, Nita Goyal, and Ravi Krishnamurthy.
On query spreadsheets.

Bonnie A. Nardi and James R. Miller.
The spreadsheet interface: A basis for end user programming.

C. Scaffidi, M. Shaw, and B. Myers.
Estimating the numbers of end users programmers.