User-Driven Situational Service Mashups

Guiling Wang
Research Center for Cloud Computing,
North China University of Technology,
Beijing, China
wangguiling@ict.ac.cn
Background

- Situational Applications\(^1\)/Situated Software\(^2\)

  "situational application is "good enough" software created for a narrow group of users with a unique set of needs. ... As the requirements of a small team using the application change, the situational application often also continues to evolve to accommodate these changes."

- For example, situational data integration application:
  - Data sources cannot be enumerated exhaustively
  - Requirements are in great variety

  "Mashup is a new application development method that allows non-professional users to build applications by combining functionalities offered by more than one source to deal with situational and ad-hoc problems."

\(^{1}\)http://en.wikipedia.org/wiki/Situational_application
IT Developer Driven -> User Driven

- **IT Developer Driven**
  - It is difficult for the IT developer to build a system to satisfy the diverse, transient user requirements which cannot be determined in advance.

- **User Driven**
  - To support end-users, business users, and casual programmers using short development life cycles to solve emergent or ad-hoc problems, and are updated frequently as needs evolve.
  - End-user programming promises a development fashion for creating situational applications.
A Motivating Application Scenario

- Who is the Criminal Suspect?
Context Information as the Clues

- Context information about the crime: location of the crime/crime items, fingerprint, monitoring video, mobile phone signal
  - Location of the crime is far from the place where the crime items are found
  - The criminal must have transported the crime items
  - Compare vehicles passed through the road and vehicles of the victim’s social relationships
There are so many potential data sources, how to select from them?

It is difficult to process and combine the data sources for non-professional programmers. Which step should they take next?

For the different criminal cases, the situational information is deemed different. How to adapt to the current situation?

- Get the social relationship of the victim, including the relatives and recent contacts
- Get the vehicle information of the victim
- Get the information of the vehicles passing through the road between crime scene and the place of crime items
- Get the suspect list by comparing the vehicle information
Previously Tedious/Difficult Work -> Present...

1. Select situation descriptions
2. Select the recommendation data services “GetContacts”, “GetRelatives”
3. Accept the recommended operation “Union”
4. Accept the recommended data services “GetVehicles”, “MonitorVehicles”
5. Accept the recommended operation “intersect”
6. Get the criminal suspects

With the help of the “Data Service Space (DSS)” …
Some Research Issues:

- It will be very helpful if we **provide some recommendations on what to do next**. How to do recommendations?
  - How to describe the situation? Given the description of the situation, **recommend a set of data services related with the situation**. Given partial mashups, **recommend the operations/data services?**
  - Where does the data service come from? It is complex to access various data sources, including HTML pages, databases, APIs, etc. How to **provide a uniform abstraction of data sources**?
  - It is complex for end users to express the service composition plan of situational applications. How to **provide a visualized programming environment for end users to mashup the data services**?
Research Agenda

- Support end users or business users to develop the situational application, and aid them by suggesting helpful recommendations.

- **Data Service Modeling**
  - provide a convenient way to access various data sources

- **Data Service Composition**
  - provide an easy-to-use data service composition environment

- **Recommendation**
  - Recommend data services and operations

- [http://113.11.194.86/DataServiceSpace/index.jsp](http://113.11.194.86/DataServiceSpace/index.jsp)
Data Services

- Nested table model (intuitive and closer to the real world)
- Semi/Automatically transformation of HTML/JSON/XML
- Both input and each column of the output nested table is associated with a set of tags
Interactive Data Service Generation Method for HTML Pages: GRUB

- Similarity assumption, supervised learning
Evaluation

- **Effectiveness**
  - Precision=98.20%, Recall=98.95% for record block,
    Precision=Recall=91.34% for record attribute. GRUB has basically as effective as typical related works.

- **Usability**
  - Users don’t need to master programming knowledge, such as HTML language and regular expression.
  - Generate a data service within two minutes.

- **Flexibility**
  - supports both record-level and page-level information extraction while no related works support both of them up to now.
How to provide an easy-to-use data service composition environment?

- The traditional service composition environment is for IT users, not for business users or end users
- There are no adequate data service composition environment for business users
  - **Data flow programming pattern**: users often feel awkward in relating the input with output
  - **Spreadsheet programming pattern**: two-dimentional spreadsheet can not process and present the complex data structure
  - **Tree-like programming pattern**: hard to understand the relationship between tree nodes

A new visualized data service operation environment is needed. It requires **agility and rich expressive power** at the same time.
Graphical Data Service Composition Environment Using Nested Tables

- Programming by Example
- Multi-granularity user operations: Table, Column, Cell
Nested Table Algebra

- The set of nested tables $S$ together with the data operations defined on the set $(\zeta, \alpha, \beta, \gamma, \rho, \mu, \nu, c, \omicron, \kappa, \delta, \tau, \lambda, \times, \cup, -, \varsigma)$ compose the nested table algebra, represented as $\langle S, \zeta, \alpha, \beta, \gamma, \rho, \mu, \nu, c, \omicron, \kappa, \delta, \tau, \lambda, \times, \cup, -, \varsigma \rangle$.

<table>
<thead>
<tr>
<th>Operation Classification</th>
<th>Data Operation</th>
<th>Corresponding basic operation for traditional nested relation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Accession &amp; Normalization</td>
<td>Import ($\zeta$)</td>
<td>-</td>
</tr>
<tr>
<td>Data Transform</td>
<td>AddColumn ($\alpha$)</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>UpdateColumn ($\beta$)</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>DeleteColumn ($\gamma$)</td>
<td>project</td>
</tr>
<tr>
<td></td>
<td>Rename ($\rho$)</td>
<td>rename</td>
</tr>
<tr>
<td></td>
<td>Unnest ($\mu$)</td>
<td>unnest</td>
</tr>
<tr>
<td></td>
<td>Nest ($\nu$)</td>
<td>nest</td>
</tr>
<tr>
<td></td>
<td>Copy ($c$)</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Sort ($\omicron$)</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Truncate ($\kappa$)</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Filter ($\delta$)</td>
<td>select</td>
</tr>
<tr>
<td></td>
<td>LinkService ($\tau$)</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>MergeTuples ($\lambda$)</td>
<td>-</td>
</tr>
<tr>
<td>Binary operations</td>
<td>Cartesian Product ($\times$)</td>
<td>Cartesian product</td>
</tr>
<tr>
<td></td>
<td>Join ($\bowtie$)</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Union ($\cup$)</td>
<td>union</td>
</tr>
<tr>
<td></td>
<td>Difference ($-$)</td>
<td>intersect</td>
</tr>
</tbody>
</table>
Guiling Wang, Shaohua Yang, Yanbo Han: Mashroom: end-user mashup programming using nested tables. WWW 2009: 861-870
Evaluation

User Experiment

Yanbo Han, Guiling Wang, Guang Ji, Peng Zhang: Situational data integration with data services and nested table. Service Oriented Computing and Applications 7(2): 129-150 (2013)
Our On-going Work: an Initial Approach to Recommendation

1.1 Create Tagged Data Services

1.2 Select Context Tags, Assign Values

2. Build Mashups

3. Interactive Execution

Data Service Wrapper

Situation Monitor

Transformer

Mashup Runtime Engine

Recommendation Engine

Data Service Mashup Knowledge Base
Example Situations for our Data Services

- Find previous top-k situations similar to the current situation. Then calculate the recommended data services emerged in these situations.

- Model context as a set of attributes
  - e.g., $C=\{\text{crime location, crime items location, ...} \}$

- A situation can be represented as a vector. Each element in the vector is value of context attribute
  - e.g., $s=<\text{Chaoyang Beijing, Haidian Beijing, ...}>$

- Given two situations $s=<c_1, c_2, ..., c_n>$ and $s’=<c_1’, c_2’, ..., c_n’>$, the similarity between $s$ and $s’$ (represented as $\text{sim}(s,s’)$) is calculated from:
  - $\text{sim}(s,s’)=\sum_{i=1}^{n} w_i * \text{sim}_i(c_i, c_i’)$
Recommend Target/Source Operator based on Statistics of Mashup History Logs

- Recommend the target operator
- Recommend the source operator

- Recommend the related data services
Recommendation based on Statistics of Mashup History Logs

- **MashupPlan** = <Ops, Connectors, PlanIn, PlanOut, Situation>
- **connector** = <Link, Mappings>
  - Link = <Op$_{src}$, Op$_{tgt}$>, Link can be represented as Op$_{src}$ → Op$_{tgt}$ for simplicity,
  - Op$_{src}$ is the source operator of the connector,
  - Op$_{tgt}$ is the target operator of the connector,
  - Mappings = {$m_i | m_i =<$OpIn$_i$, A$_q$>, A$_q$ ∈ T$_{src}$}.
- If two data services can be connected by direct or indirect connectors, we call there exists a hyperlink between these two data services. Connectors is the set of connectors between DS$_{src}$ and DS$_{tgt}$,
  - hyperlink = <DS$_{src}$, DS$_{tgt}$, connectors>

$$Rate(a \rightarrow b) = \begin{cases} \sum_{o_i \in Ops_{DS}} Sim_{a,o_i} \cdot C_{o_i \rightarrow b}, a \in Ops_{DS} \\ \sum_{o_i \in Ops_{op}} C_{a \rightarrow b}, a \in Ops_{operation} \end{cases}$$
Initial Experiment

- Data set
  - Mashup history logs are from Yahoo! Pipes, totally 620 pipes (after pre-processing, 4857 operator connectors)
  - 600 pipes as sample data set, 20 as the test data set

- select k=5 and select 200, 300,..., and 600 pipes
Acknowledgement
Research Center for Cloud Computing, North China University of Technology
Prof. Yanbo Han and other colleagues
Graduated PH.D student: Dr. Shaohua Yang, Dr. Guang Ji
Master student: Sai Zhang, Bo Cao, Meizhen Geng

Thanks for your attention!

Guiling Wang
Email: wangguiling@ict.ac.cn, wangguiling@ncut.edu.cn