Comparing Workflow Specification Languages: A Matter of Views

Victor Vianu
UCSD/Webdam/INRIA/LSV
• Title: work with Serge and Pierre Bourhis
  ICDT 2011
• More on workflow views: inferring specifications
  preliminary ideas with Luc and Serge
• AXML as a query language
  ongoing work with Serge and Pierre Bourhis
Comparing Workflow Specification Languages: A Matter of Views

- Framework for comparing workflow specifications: based on views
- Specific results for AXML workflow specification mechanisms: guards, automata, temporal logic
- Comparison to IBM’s Tuple Atifacts
What is a data-centric workflow?

• States (with data)
• Events (with data)
• Transitions
Specifications of data-centric workflows

Two examples

• Active XML
• Tuple Artifacts (IBM)
Workflow specification mechanisms for AXML

- **BAXML**: static constraints only
- **GAXML**: function calls and returns controlled by guards
- **AAXML**: allowed transitions described by an automaton
- **TAXML**: workflow constrained by temporal property of history
GAXML by example
GAXML by example

query on Order and Catalog

guard: product available?

Bill
Credit check
Deliver
GAXML by example
GAXML by example
GAXML by example

Mail-Order-Center

Catalog

Customer
  "Joe"

Order

Product
  "ipod80G"

Payment
  "VISA"  "$399"

Rating
  "Good"

!Deliver
GAXML by example

Guard: rating good or excellent?
GAXML by example

Guard: rating good or excellent?
GAXML by example
IBM’s Tuple Artifacts

Tuple artifacts:
• evolving tuples of data values
• local states: evolving relations
• fixed underlying database
• services (pre/post conditions on tuple, local state, and a global relational database)
System with two artifacts

<table>
<thead>
<tr>
<th>Database</th>
<th>Local State</th>
</tr>
</thead>
</table>

### Table: DATABASE

<table>
<thead>
<tr>
<th>Product</th>
<th>Customer</th>
</tr>
</thead>
<tbody>
<tr>
<td>prod_id</td>
<td>customer_id</td>
</tr>
<tr>
<td>min_order_qty</td>
<td>status</td>
</tr>
<tr>
<td>desired_price</td>
<td>credit_rating</td>
</tr>
</tbody>
</table>
Events: services

- initiate order
- negotiation round
- approve final price
- ship order

current snapshot of artifact system  available services
Services evaluate their pre-conditions in parallel

Current snapshot of artifact system

Available services

- initiate order
- negotiation round
- approve final price
- add line item
One qualifying service is non-deterministically picked for execution

current snapshot of artifact system

available services
Post-condition Is Satisfied

- initiate order
- negotiation round
- approve final price
- add line item

current snapshot of artifact system
available services
successor snapshot of artifact system
Example: Evolution of QUOTE Artifact

### QUOTE

<table>
<thead>
<tr>
<th>order#</th>
<th>li_prod</th>
<th>li_qty</th>
<th>ask</th>
<th>bid</th>
<th>final</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### li_quotes

<table>
<thead>
<tr>
<th>prod</th>
<th>quantity</th>
<th>negotiated price quote</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### state

- negotiating
- approval_pending
- archiving
- idle
Example: Evolution of QUOTE Artifact

<table>
<thead>
<tr>
<th>order#</th>
<th>li_prod</th>
<th>li_qty</th>
<th>ask</th>
<th>bid</th>
<th>final</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>P17</td>
<td>250</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

li_quotes

<table>
<thead>
<tr>
<th>prod</th>
<th>quantity</th>
<th>negotiated price quote</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

state

- negotiating
- approval_pending
- archiving
- idle
Example: Evolution of QUOTE Artifact

<table>
<thead>
<tr>
<th>order#</th>
<th>li_prod</th>
<th>li_qty</th>
<th>ask</th>
<th>bid</th>
<th>final</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>P17</td>
<td>250</td>
<td>$25</td>
<td>$15</td>
<td></td>
</tr>
</tbody>
</table>

li_quotes

<table>
<thead>
<tr>
<th>prod</th>
<th>quantity</th>
<th>negotiated price quote</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example: Evolution of QUOTE Artifact

<table>
<thead>
<tr>
<th>order#</th>
<th>li_prod</th>
<th>li_qty</th>
<th>ask</th>
<th>bid</th>
<th>final</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>P17</td>
<td>250</td>
<td>$23</td>
<td>$17</td>
<td></td>
</tr>
</tbody>
</table>

- **li_quotes**
  - **prod**: 
  - Quantity
  - Negotiated price quote

- **state**
  - Negotiating
  - Approval_pending
  - Archiving
  - Idle
Example: Evolution of QUOTE Artifact

<table>
<thead>
<tr>
<th>order#</th>
<th>li_prod</th>
<th>li_qty</th>
<th>ask</th>
<th>bid</th>
<th>final</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>P17</td>
<td>250</td>
<td>$20</td>
<td>$20</td>
<td></td>
</tr>
</tbody>
</table>

QUOTE

<table>
<thead>
<tr>
<th>li_quotes</th>
<th>state</th>
</tr>
</thead>
<tbody>
<tr>
<td>pro</td>
<td>quantity</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- negotiating
- approval_pending
- archiving
- idle
Example: Evolution of QUOTE Artifact

<table>
<thead>
<tr>
<th>order#</th>
<th>li_prod</th>
<th>li_qty</th>
<th>ask</th>
<th>bid</th>
<th>final</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>P17</td>
<td>250</td>
<td>$20</td>
<td>$20</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>li_quotes state</th>
</tr>
</thead>
<tbody>
<tr>
<td>prod</td>
</tr>
<tr>
<td>------</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

- negotiating
- approval_pending
- archiving
- idle
Example: Evolution of QUOTE Artifact

<table>
<thead>
<tr>
<th>order#</th>
<th>li_prod</th>
<th>li_qty</th>
<th>ask</th>
<th>bid</th>
<th>final</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>P17</td>
<td>250</td>
<td>$20</td>
<td>$20</td>
<td>$20</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>li_quotes</th>
<th>state</th>
</tr>
</thead>
<tbody>
<tr>
<td>prod</td>
<td>quantity</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- negotiating
- approval_pending
- archiving
- idle
Example: Evolution of QUOTE Artifact

<table>
<thead>
<tr>
<th>order#</th>
<th>li_prod</th>
<th>li_qty</th>
<th>ask</th>
<th>bid</th>
<th>final</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>P17</td>
<td>250</td>
<td>$20</td>
<td>$20</td>
<td>$20</td>
</tr>
</tbody>
</table>

QUOTE

li_quotes

state

- negotiating
- approval_pending
- archiving
- idle

Negotiating price quote

Approving pending

Archiving

Idle
How to compare different workflows?

Building a beehive

Building an ant nest
Use Views!

bee  →  insect  →  ant
hive  →  dwelling  →  nest

Building a beehive  Building an ant nest
Workflow views

• Views on states: restructure, hide
• Views on events: restructure, hide → silent
View within AXML

!order !order !deliver !deliver
!credit-check !deliver \( \varepsilon \)
View mapping AXML to Tuple Artifacts

function call !s                               service call s
Comparing workflow specification languages

- Define views mapping to a common abstraction
- Define workflow simulation

A bit more complicated for tree of runs
Workflow specification mechanisms for AXML

- **BAXML**: static constraints only
- **GAXML**: function calls and return controlled by guards
- **AAXML**: allowed transitions described by an automaton
- **TAXML**: workflow constrained by temporal property of history
Main result on AXML

BAXML can simulate GAXML, AAXML*, TAXML*

In other words:
static constraints can simulate all other mechanisms

* Modulo very minor restrictions
BAXML vs Tuple Artifacts

• BAXML can simulate Tuple Artifacts with respect to previous view

• Tuple Artifacts cannot simulate BAXML view: keep just names of services/function calls

Note: the more information is kept in the view, the harder to simulate
Another use of views:
controlled exposure of specification

- **Adapt** complex specifications to needs of users
- **Hide** private information about internal workflow

**Issue:** how to *explain* a view to its users
Example

- BAXML view: keep just tree of function calls
How to explain the view

• Ideally: if it is regular, finite-state transition system whose unfolding is the infinite tree

• If the tree of runs is not regular, can it be specified in a more powerful but still reasonable way? Can it be approximated? Is the set of infinite runs regular?

• How about more complex views with data?
Conclusion

Views rule (once again)!

- Allow comparing different workflow models
- Customization of workflows
- Abstraction that can be used in verification