What is a Choreography?

A choreography is a way to organize a multiparty web application in a manner that emphasizes the interactions among the constituent parties.

A choreography specifies

- Desired interactions from a shared viewpoint
- Parties involved as roles
- Types of messages the roles send one another
- Any constraints on the occurrence and ordering of the messages
Model of Commitments

We specify a model where parties

- Create publicly observable commitments with one another
- Manipulate commitments with one another
- Adopt several roles during an enactment

A commitment is represented as

\[ c = C \text{ (debtor, creditor, antecedent, consequent)} \]

An example

\[ c = C \text{ (buyer, seller, goods, pay)} \]
Commitments and Operations on Them

The commitment $c$ can be manipulated

- $create\ (c)$
- $cancel\ (c)$
- $release\ (c)$
- $assign\ (c, z)$
- $delegate\ (c, z)$
- $detach\ (c)$
- $update\ (c)$
- $discharge\ (c)$
Commitments and Operations on Them

The life cycle of a commitment

- **commitment**
  - **null**
    - **create**
    - **expire**
  - **active**
    - **conditional**
      - **antecedent**
    - **detached**
      - **consequent**
        - **consequent_timeout**
        - **terminated**
        - **discharged**
        - **violated**
      - **cancel**
      - **release**
Commitments and Operations on Them

Sample enactments for a commitment

\[ c = C(buyer, seller, goods, pay) \]

1. **offer**
   - buyer
   - seller
   - create(c)

2. **goods**
   - [c exists]
   - detach(c)

3. **pay**
   - discharge(c)

4. **reject**
   - cancel(c)

[c does not exist]
Message Meanings and Relationships

Creating a Commitment

\[
c = C(x, y, p, q)
\]

1. \textit{create}(c)
Section II: Model and Semantic Patterns

Semantic Patterns

Message Meanings and Relationships

Progressing a Commitment

\[
c = C(x, y, p, q)
\]

1. partial detach (c)

2. full detach (c)

3. partial discharge (c)

4. delegate (c, z)

5. assign (c, z')

\[ c \text{ exists } \]

Anup Kalia, Ashok Mallya, Munindar Singh

© 2012 by SOC Lab, NCSU
Message Meanings and Relationships

Updating a Commitment

\[
c = C(x, y, p, q)
\]

2. \textit{update}(c)

- \[ c \text{ exists } \]
- \[ c \text{ does not exist} \]
- \[ c' \text{ exists} \]
Message Meanings and Relationships

Completing a Commitment

\[ c = C(x, y, p, q) \]

- 2. cancel(c)
- 3. release(c)
- 4. full discharge(c)

\[ c \text{ does not exist} \]

[ c exists ]
Message Meanings and Relationships

Advancing a Commitment

\[ c = C(x, y, p, q) \]

1. Alt
2. do action
3. do action
Canceling a Commitment

sd Cancel (Commitment c)

c = C(x, y, p, q)

x: Debtor
y: Creditor
z: Debtor’s Supervisor

1. do: cancel(c)
2. sanction: 1
3. escalate: 1
4. sanction: cancel(c): 3
Releasing a Commitment

\[ c = C(x, y, p, q) \]

\begin{enumerate}
\item \textbf{do:} \textit{release}(c)
\item \textbf{request:} \textit{release}(c)
\item \textbf{do:} \textit{release}(c)
\item \textbf{do:} \textit{release}(c)
\item \textbf{refuse:} 2
\end{enumerate}
Delegating a Commitment

Delegating a Commitment

\[ c = C(x, y, p, q) \]

1. request: \( \text{delegate}(c, a) \)
2. agree: 1
3. do: \( \text{delegate}(c, a) \)
4. refuse: 1

\( x: \) Debtor
\( y: \) Creditor
\( a: \) Debtor
Assigning a Commitment

- Analogous to a delegation pattern
- Here the creditor initiates an assignment by sending a message to the debtor
- Can perform the assignment unilaterally
A Travel Planning Business Scenario

- A passenger requests a travel agency to book air line tickets, a hotel room, and a rental car
- The travel agency interacts with airlines, hotels, and rental cars
- The passenger makes payment to the travel agency for its services
- The travel agency pays the airline, the hotel, and the rental car company
M1: Identify Stakeholders as Roles

Steps in M1

- **Input** A business scenario and domain expertise
- **Output** A set of autonomous entities whose interaction is being modeled

In travel planning business scenario roles identified are Passenger, Travel Agency, Airline, Hotel, Car Rental Agency
M2: Record one or more Typical Discourse

Steps in M2

- **Input** The output of Step M1, business scenario, and domain expertise
- **Output** Illustrative discourses of the interaction being modeled in terms of messages: one primary and zero or more secondary
The travel planning example (happy path)

<table>
<thead>
<tr>
<th>ID</th>
<th>S</th>
<th>R</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>P</td>
<td>T</td>
<td>I’ll pay $a$ if you send me itinerary $i$</td>
</tr>
<tr>
<td>2</td>
<td>T</td>
<td>A$_1$</td>
<td>I’ll pay $a_1$ if you send me ticket $t$ for $P$</td>
</tr>
<tr>
<td>3</td>
<td>T</td>
<td>H$_1$</td>
<td>$P$ will pay $a_2$ if you make a room resv $r$ for $P$</td>
</tr>
<tr>
<td>4</td>
<td>A$_1$</td>
<td>T</td>
<td>I have no tickets</td>
</tr>
<tr>
<td>5</td>
<td>T</td>
<td>A$_2$</td>
<td>I’ll pay $a_1$ if you send me ticket $t$ for $P$</td>
</tr>
<tr>
<td>6</td>
<td>A$_2$</td>
<td>T</td>
<td>Here’s ticket $t$ for $P$</td>
</tr>
<tr>
<td>7</td>
<td>T</td>
<td>R</td>
<td>$P$ will pay $a_3$ if you make a car resv $rc$ for $P$</td>
</tr>
<tr>
<td>8</td>
<td>H$_1$</td>
<td>T</td>
<td>Reservation $r$ done for $P$</td>
</tr>
<tr>
<td>9</td>
<td>R</td>
<td>T</td>
<td>Reservation $rc$ done for $P$</td>
</tr>
<tr>
<td>10</td>
<td>T</td>
<td>P</td>
<td>Here’s itinerary $i$</td>
</tr>
<tr>
<td>20</td>
<td>P</td>
<td>T</td>
<td>Here is $a$ I owe you for itinerary $i$</td>
</tr>
</tbody>
</table>
Guidelines for creating a primary discourse

Create a discourse that

- represents a positive outcome
- reflects the social or organizational relationships among the roles
- omit messages that do not add to an observer’s knowledge of the interaction
- does not include roles that do not add to an observer’s knowledge of the interaction
M3: Record One or More Discourses Demonstrating Exceptions

Steps in M3

- **Input** The output of Step M2, business scenario, and domain expertise
- **Output** Discourses exhibiting exceptions in the interactions being modeled
## The trip planning example (exceptions)

<table>
<thead>
<tr>
<th>ID</th>
<th>S</th>
<th>R</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>P</td>
<td>T</td>
<td>Please remove car reservation from i</td>
</tr>
<tr>
<td>12a</td>
<td>T</td>
<td>R</td>
<td>Please cancel reservation rc</td>
</tr>
<tr>
<td>13</td>
<td>T</td>
<td>P</td>
<td>Here’s the updated itinerary i</td>
</tr>
<tr>
<td>14a</td>
<td>T</td>
<td>H1</td>
<td>Room reservation has to be canceled</td>
</tr>
<tr>
<td>15</td>
<td>T</td>
<td>P</td>
<td>Will an alternative room do?</td>
</tr>
<tr>
<td>16</td>
<td>P</td>
<td>T</td>
<td>yes</td>
</tr>
<tr>
<td>14b</td>
<td>H1</td>
<td>T</td>
<td>OK to cancel room reservation</td>
</tr>
<tr>
<td>17</td>
<td>T</td>
<td>H2</td>
<td>P will pay you $a_2 if you reserve room r' for P</td>
</tr>
<tr>
<td>18</td>
<td>H2</td>
<td>T</td>
<td>Reservation r’ done for P</td>
</tr>
<tr>
<td>19</td>
<td>T</td>
<td>P</td>
<td>Here’s the updated itinerary i</td>
</tr>
</tbody>
</table>
M4: Annotate Messages in Discourses

Steps in M4

- **Input** The output of Step M2 and M3
- **Output** Annotations of each message produced by steps M2 and M3
M4: Annotate Messages in Discourses

Illustrating the message annotations

- Message 1 *creates* a commitment $C_1$ from P to T
- Message 10 *progresses* Message 1 by detaching the commitment $C_1$ and *advances* Messages 1, 6, 8, and 9
- Message 13 *updates* Message 10 and *advances* Message 11
- Message 20 *completes* Message 10 by discharging $C_1$
M5: Validate Discourses

Steps in M5

- **Input** The annotations of step M4 along with the discourses of steps M2 and M3
- **Output** None
## M5: Validate Discourses

### Completeness check for happy path

<table>
<thead>
<tr>
<th>ID</th>
<th>S</th>
<th>R</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>21</td>
<td>T</td>
<td>$A_2$</td>
<td>Here is $a_1$ I owe you for ticket $t$</td>
</tr>
<tr>
<td>22</td>
<td>T</td>
<td>$H_2$</td>
<td>Here is $a_2$ I owe you for room $r'$</td>
</tr>
</tbody>
</table>

### Completeness check for exceptions

<table>
<thead>
<tr>
<th>ID</th>
<th>S</th>
<th>R</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>12b</td>
<td>$R$</td>
<td>$T$</td>
<td>Reservation $rc$ canceled</td>
</tr>
<tr>
<td>14c</td>
<td>$H_1$</td>
<td>$T$</td>
<td>Room $r$ canceled</td>
</tr>
</tbody>
</table>
M6: Map the Discourses to CC Graph

Steps in M6

- **Input** The output of Step M4 and M5
- **Output** A CC graph of the discourse; identification of commitment operation patterns in the interaction
A conversation graph of the discourse
Discharging a commitment

**Conversation graph**

- S
  - p: create(c)
  - q: detach(c)
  - r: discharge(c)
- R

**CC component**

- S
  - p: create(c)
  - q: detach(c)
  - r: discharge(c)
- R
Canceling a Commitment

Conversation graph

1. **create**\((c)\)
2. **detach**\((c)\)
3. **cancel**\((c)\)

CC component

1. **create**\((c)\)
2. **detach**\((c)\)
3. **cancel**\((c)\)
Releasing a Commitment

Conversation graph

CC component

create(c)
detach(c)
release(c)

release(c)

Conversation graph

CC component

create(c)
detach(c)
release(c)

release(c)
Delegating a Commitment
Assigning a Commitment

**Conversation graph**

- **S**: \( p \) \( \text{create}(c) \) \( R \)
- **R**: \( q \) \( \text{assign}(c) \) \( R' \)
- **S**: \( r \) \( \text{discharge}(c) \) \( R' \)

**CC component**

- **S**: \( p \) \( \text{create}(c4) \) \( R \)
- **R**: \( q \) \( \text{assign}(c4,R') \)
- **S**: \( r \) \( \text{discharge}(c5) \)
- **R**: \( r \) \( \text{discharge}(c5) \)

- **S**: \( p \) \( \text{create}(c4) \) \( R \)
- **R**: \( q \) \( \text{detach}(c4) \)
- **S**: \( r \) \( \text{assign}(c4,R') \)
- **R**: \( s \) \( \text{discharge}(c5) \)
Updation a Commitment

Conversation graph

CC component

S → p: create(c) → R

S → q: update(c) → R

S → R

R → S

p

q: update(c)
Generate CC graph from the Conversation Graph

The generated CC graph
M7: Generate a Choreography

Steps in M7

- **Input** The output of Step M6
- **Output** A choreography based on the CC graph generated for the recorded discourses
Section IV: Methodology

M7: Generate a Choreography

Generate Choreography from CC graph

1: create(c1)
10: detach(c1)
13: update(c1)
19: update(c1)
20: disch(c1)

[ c1 exists ]
10. full detach(c1)

alt
19. update(C1)
20. disch(c1)