Lecture Notes

Course Number: CSC 513
Instructor: Dr. Singh
Lecture Number: 20
Express as a sequence diagram (a.k.a. message sequence chart)
EASY

(NESTED)

EACH ARROW CORRESPONDS TO A MESSAGE

DOESN'T RESPECT NESTING
CONVENTIONAL SEQ DIAGRAMS

- Encode API calls
- Include important objects
- Include local actions (self calls)

WHAT WE NEED

- Encode messages
- Include business partners (entire we understand are autonomous)
- Hide local action emphasizing interactions among business partners

INTERACTIONS AMONG AUTONOMOUS PARTIES
BE UNDERSTOOD AT A BUSINESS LEVEL
ARE THE BUSINESS PROTOCOLS

→ NEED FORMAL WAY TO REPRESENT SUCH PROTOCOLS
WAYS TO EXPRESS

ACI : accept or reject

 LOOP :

NESTING :

STANDARD
(UML 2.0)
SEPAR DIAGRAMS
INCLUDES
CONSTRUCTS
SUCH AS
THES
STATECHARTS (STATE DIAGRAMS IN UML 2.0)

- BUSINESS RELATIONSHIPS
- GENERALIZE OVER FSM

- CONDITION (OR GUARD) ON A TRANSITION

- NEST = STATE
- PARALLEL STATE

© North Carolina State University, All Rights Reserved
NE57

(Bad Example)

SESSION

WORK DAY
CLASS

use lab

Can we
library

Class day

Lab

Nonclass day
Traditional use of statecharts

- involves low-level details of
  - the object state
  - the object behavior

What we want

state of an interaction
expressed in the state of a "contract"

<table>
<thead>
<tr>
<th>Low-Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>api -&gt; Low-Level</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Tight Coupling</td>
</tr>
<tr>
<td>Limit Possible Implementations</td>
</tr>
<tr>
<td>Based on the Corresponding Notion of Correctness</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>High-Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>business</td>
</tr>
<tr>
<td>interacts</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Loose Coupling</td>
</tr>
<tr>
<td>Yields more Flexibility</td>
</tr>
</tbody>
</table>
E-BUSINESS PROCESSES e.g. PURCHASE, Fulfillment, Financing...

Each, when thought of as an app, is NOT helpfully described as P-L-D (or variants) but IS described via the partners and their interactions.

CONVENTIONAL: 

Buyer | Broker | Seller | Bank | Shygen

Moral equivalent of one tier and P-L-D

MONOLITHIC BUSINESS PROCESS

One dominant party runs the show
Idea:
- Construct these interactions on top of suitable business interactions
- Construct corresponding arch concepts: component, interaction pattern, commands
- Step away from operational details
- Characterize **business relationships** (in a non-operational way)

Identify commitments of a party to another

\[ G(\text{debtor, creditor, antecedent, consequent}) \]

When a commitment is active,

The debtor would bring about the consequent provided the antecedent holds

I.e. if the antecedent becomes true, then the consequent should become true

\[ G(\text{Amazon, Madhura, ($19), (Brave New World)}) \]

Typically, we expect Madhura pays Amazon ships

But in general there is no requirement about who should act

© North Carolina State University, All Rights Reserved
Declarative Representation: WHAT (Outcomes)
Procedural (Operational): HOW (Steps)

May have many enactments (on operationalizations)

Ordering between antecedent and consequent

(i) Madhura pays; then Amazon ships

(ii) Amazon ships; then Madhura pays

(iii) Madhura pays via credit card #2

Additional input involved via UPS from warehouse #3

The commitment is fundamentally neutral about the above; applies only on the debtor

(v) Amazon ships; Nothing else
\[ p \rightarrow q \]

<table>
<thead>
<tr>
<th></th>
<th>( p )</th>
<th>( q )</th>
<th>( p \rightarrow q )</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
</tbody>
</table>

\( \therefore \text{pay} \rightarrow \text{ship} \)

|   | \( F \) | \( T \) | \( T \) |
Peer-to-Peer Computing

Symmetric client-server: (callbacks) each party can be the client of the other

Asynchrony: while the request-response paradigm corresponds to pull, asynchronous communication corresponds to push

Generally to place the entire intelligence on the server (pushing) side

Federation of equals: (business partners) when the participants can enact the protocols they like

Protocols defined in terms of business relationships e.g. place or fulfill an order, not just send a message
Message-Oriented Middleware: 1

**MoM** Messaging Middleware

**Asynchrony** (unlike phone, like email)

- **Queues**: point to point, support posting and reading messages
- **Topics**: logical multicasts, support publishing and subscribing to application-specific topics; thus more flexible than queues
- Can offer **reliability guarantees of delivery or failure notification to sender**
  - Analogous to store and forward networks
- Some messages correspond to event notifications
Message-Oriented Middleware: 2

- Varies in reliability guarantees
- Usually implemented over databases
- Can be used through an invocation-based interface (i.e., registered callbacks)
Message-Driven Beans

A standardized receiver for messages

- Clients can’t invoke them directly; must send messages to them
- No need for specialized interfaces, such as `home`, `remote`, ...  
  `vs Entity Beans`
- Easy interface to implement: mainly `onMessage()`, but limited message typing
- Stateless: thus no conversations
Methods for Message-Driven Beans

- `onMessage()`: define what actions to take when a message arrives on the destination this bean is watching