Protocols for Processes

Programming in the Large for Open Systems

Munindar P. Singh
(Students: Amit K. Chopra, Nirmit V. Desai, Ashok U. Mallya)

singh@ncsu.edu

Department of Computer Science
North Carolina State University
http://www.csc.ncsu.edu/faculty/mpsingh/
Why Processes and Protocols?

- Heavy interest from IT practitioners
  - Standardization efforts
  - Any number of products
- Current industry approaches are impoverished: scripting languages
  - No special abstractions for dealing with open systems: autonomy, heterogeneity, dynamism
- That is, not designed for SOAs
Programming in the Large

About creating large software systems; the main challenges of modern software engineering

- Traditional emphases
  - Built by large teams
  - Long-lived and stateful components

- Proposed emphases
  - Special treatment of open systems: autonomy, heterogeneity, dynamism
  - Long-lived and stateful components interacting in subtle ways
The Essential Tension

- **Reusability** requires
  - Context freedom
  - Encapsulation

- **Usability** (usefulness) requires
  - Context sensitivity
  - Varieties of context include organizations, laws, and the real world

- Main idea
  - The components have a life of their own
  - The interactions are what matter
A Process is …

- **Orchestration:** a partial order of actions under the control of a central conductor
  - Akin to a workflow or flow in BPEL
- **Choreography:** an exchange of messages among participants
  - Akin to a conversation as described by WS-Chor
- **Collaboration:** a joint set of activities among business partners
  - Akin to real business; essential for SOAs
Emphases of Collaboration

- Dynamic Organizations
- Rule-Based Commitment Protocols: Flexibility
- Commitment Protocols: Content & Compliance
- Protocols: Modularity

Monitoring and compliance
Implementation and enactment
Modeling and validation
Innovations: 1

- **Protocols**: Conceptually decentralized, reusable, encapsulations of processes
- **Commitments**: Content for protocols
  - Support reuse via abstractions for refinement and aggregation of protocols
  - What the protocol should accomplish
  - What deviations are legitimate and what aren’t
  - Operational semantics for commitments
NetBill and Escrow Protocols

- C: rfq
  M: offer
  C: accept
  M: goods
  C: pay
  M: receipt

- (B, E): deposit
  (E, S): secured
  (S, B): goods
  (B, E): goods NOK
  (B, S): goods return
  (S, E): released
  (E, B): refund
  (E, S): pay
Innovations: 2

- **Rule-Based Reasoning:**
  - Expressing protocols flexibly
  - Accommodating context
  - Deciding specific actions by applying policies

- **Spheres of Commitment:**
  - Modeling organizations
  - Enacting protocols
  - Monitoring and verifying compliance

- Processes $= $ Protocols $+$ Policies
Enhanced NetBill

Compiled from a commitment machine for NetBill
Contributions (In Progress)

- Specification language for protocols
- Formal semantics based on commitments
- Protocol algebra to support refinement and aggregation

**Engineering:** not full automation, but tools for
- Modeling and validation of protocols
- Modeling and validation of processes
- Enactment via Spheres of Commitment
- Monitoring and compliance
Trends and Assessment

- Increasing # of business protocols
  - IOTP, Escrow, SET, NetBill, …
  - RosettaNet: 107 Partner Interface Processes (PIPs)
  - ebXML Business Process Specification Schema (BPSS)

- Intended to be legally binding

- Generally highly limited: two party, request-response protocols

- No commitments; no formal semantics

- Limited support for modeling or enactment
Simple Scenario and Example Run

- A customer (C) looks up a book at a vendor (B) and is quoted price and availability
- C orders the book from B
- B ships to C
- C pays B

Diagram:

- $S_0$ to $S_1$: reqQuote(c,b,g)
- $S_2$ to $S_1$: sendQuote(b,c,g,p)
- $S_2$ to $S_3$: sendAccept(c,b,p)
- $S_4$ to $S_3$: sendGoods(b,c,g)
- $S_4$ to $S_5$: sendMoney(c,b,p)

Customer, $c$  
Bookstore, $b$
Process View: Flow or Protocol

Select → Pay
<table>
<thead>
<tr>
<th></th>
<th>Bank</th>
<th>Bookstore</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Customer</td>
</tr>
</tbody>
</table>

Pay → Ship
<table>
<thead>
<tr>
<th></th>
<th>Shipper</th>
<th>Customer</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Bookstore</td>
</tr>
</tbody>
</table>

Ship → Pay
<table>
<thead>
<tr>
<th></th>
<th>Pay</th>
</tr>
</thead>
</table>

Pay → Send Receipt
<table>
<thead>
<tr>
<th></th>
<th>Send Receipt</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Customer</td>
</tr>
<tr>
<td></td>
<td>Bookstore</td>
</tr>
</tbody>
</table>

Send Receipt → Bookstore
<table>
<thead>
<tr>
<th></th>
<th>Bookstore</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Customer</td>
</tr>
</tbody>
</table>

Bookstore → Customer
|     | Customer |

Customer → Select
|     | Select |

Select → Order
|     | Order |

Order → Ship
|     | Ship |

Ship → Pay
|     | Pay |

Pay → Bank
|     | Bank |

Bank → Customer
Challenges: Modeling

- **Refinement**: pay by credit card versus pay
- **Extensibility**: verify C’s attributes, e.g., age
- **Adjustment**: receive payment before shipping; receive book before paying

Alternative execution examples:
- B arranges for a shipper (S) to deliver the book to C
- C pays via bank (K)
- Compose a process from the above
Refinement of Protocols

Selection criteria for protocols

- **Functional**: pay versus ship
- **Nonfunctional**: payer trusts payee or not

![Diagram showing pay methods]

- Pay
- Pay with cash
- Pay with receipt
- Pay with check
- Pay via debit card
- Pay via credit card
- Pay with cash and receipt
Aggregation of Protocols

- A simplified protocol may be revealed to a give role
- Decisions could be taken internally but not exposed
Example Run: Pay via Bank

Customer's Bank, $k$

Customer, $c$

Bookstore, $b$

$S_0 \xrightarrow{\text{reqQuote}(c,b,g)} S_1$

$S_2 \xrightarrow{\text{sendQuote}(b,c,g,p)}
\xleftarrow{\text{sendAccept}(c,b,p)} S_3$

$S_4 \xrightarrow{\text{sendGoods}(b,c,g)}
\xleftarrow{\text{sendMoney}(k,b,p)} S_5$

$S_21 \xrightarrow{\text{authPay}(c,b,p)}$
Example Run: Shipper Protocol

Receiver, v  Sender, m  Shipper, s
Example Run: Composed Purchase

Bank, $k$  Customer, $c$  Bookstore, $b$  Shipper, $x$

$S_0$  reqQuote$(c, b, g)$
$S_1$  sendQuote$(b, c, g, p)$
$S_2$  sendAccept$(c, b, g, p)$
$S_3$  reqQuote$(b, x, [gc])$
$S_4$  authPay$(x, p)$
$S_5$  sendMoney$(k, x, p)$
$S_6$  sendMoney$(b, x, px)$
$S_{11}$  sendQuote$(x, b, [gc], px)$
$S_{12}$  sendAccept$(b, x, [gc], px)$
$S_{13}$  sendGoods$(b, g, x)$
$S_{14}$  sendGoods$(x, c, g)$

Singh et al., October 2004 – p.20/27
Challenges: Enactment

- Behaving adaptively: decide dynamically to ship before payment to trusted Cs

- Handling exceptions
  - External problems: cannot ship book
  - Context-sensitivity: not legal for kids
  - Detecting violations: no payment; book arrives damaged
  - Correcting violations: remind, complain, refund, . . .

- Exploiting opportunities: combine orders from same C
Example Run: Return and Refund

Example: Uniform Commercial Code (UCC) allows returns with refunds for goods that are received damaged
Architecture

Maintains protocol state: Commitments and propositions, roles being played, ...

Ex: Business policies, pricing policies

Binds to roles, interacts with other roles.

Rule Base
- Internal Policy
- Protocol Rules

Knowledge Base
- updates
- queries

Main
- consults

Agent Playing a Role
- Rules dictated by protocols being enacted

Messages

Local domain

Public domain

Roles
- Proposotions

Commitments
- Rules

Protocol Specified in OWL-P

© Singh et al., October 2004 – p.24/27
Ongoing Work

- A language, OWL-P, OWL for Protocols
  - Roles
  - Messages: content as propositions and commitments
  - Rules to describe messages and roles
- Tool to generate skeletons from OWL-P
- Operational semantics in $\pi$-calculus
- Rule-based policies that help agents satisfy their protocol roles
- Protocol algebra to support refinement and aggregation
Processes = Protocols + Policies

- Operational patterns
  - Time outs, remind, garbage collect, . . .
  - Decisions to manipulate: delegate, assign, . . .
  - Winograd & Flores and other such

- Methodologies, e.g., enhancing Tropos:
  - Cover functional reqs via protocols
  - Refine protocols for nonfunctional reqs
  - Enact protocols dynamically based on agent policies and context
Newer papers in ICWS, ICSOC, AAMAS address parts of the above vision


