Protocols for Processes

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.
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
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
- (B, E): goods OK
- (E, S): released
- (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).

- 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.

Customer, $c$

Bookstore, $b$
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.

```
Pay

Pay 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 given role.
- Decisions could be taken internally but not exposed.
Example Run: Pay via Bank

Customer's Bank, \( k \)  

Customer, \( c \)  

Bookstore, \( b \)

\( s_0 \) \( \xrightarrow{\text{reqQuote}(c,b,g)} \) \( s_1 \)  
\( s_2 \) \( \xrightarrow{\text{sendQuote}(b,c,g,p)} \) \( s_3 \)  
\( s_4 \) \( \xrightarrow{\text{sendAccept}(c,b,p)} \) \( s_3 \)  
\( s_5 \) \( \xrightarrow{\text{sendGoods}(b,c,g)} \) \( s_3 \)  
\( s_5 \) \( \xrightarrow{\text{sendMoney}(k,b,p)} \) \( s_5 \)
Example Run: Shipper Protocol

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

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

$S_0$  $S_1$  $S_2$  $S_3$  $S_4$  $S_5$  $S_{11}$  $S_{12}$  $S_{13}$  $S_{14}$  $S_{16}$

- reqQuote$(c,b,g)$
- sendQuote$(b,c,g,p)$
- sendAccept$(c,b,g,p)$
- reqQuote$(b,x,[gc])$
- sendQuote$(x,b,[gc],px)$
- sendAccept$(b,x,[gc],px)$
- sendGoods$(b,g,x)$
- sendGoods$(x,c,g)$
- sendMoney$(k,x,p)$
- sendMoney$(b,x,px)$
Challenges: Enactment

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

Customer, $c$          Bookstore, $b$

- $S_0 \xrightarrow{\text{reqQuote}(c,b,g)} S_1$
- $S_2 \xrightarrow{\text{sendQuote}(b,c,g,p)} S_1$
- $S_2 \xrightarrow{\text{acceptQuote}(c,b,p)} S_3$
- $S_4 \xrightarrow{\text{sendGoods}(b,c,g)} S_3$
- $S_4 \xrightarrow{\text{sendMoney}(c,b,p)} S_5$
- $S_5 \xrightarrow{\text{returnGoods}(c,b,g)} S_{18}$
- $S_{19} \xrightarrow{\text{sendRefund}(b,c,p)}$
Spheres of Commitment

- Buyer
- Seller
- Billing
- Inventory
- Packaging
- Shipping
Architecture

- Knowledge Base
- Rule Base
  - Internal Policy
  - Protocol Rules
- Protocol Rules
- Main
- Agent Playing a Role
  - Rules dictated by protocols being enacted
  - Binds to roles, interacts with other roles.
- Messages
  - Local domain
  - Public domain
- Commitments
- Propositions
- Roles
- Rules
- Messages

- Maintains protocol state: Commitments and propositions, roles being played, ...
- Ex: Business policies, pricing policies
- Usually several roles per agent
- Usually several protocols, each with multiple roles

© Munindar P. Singh, July 2004 – p.22/25
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 π-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.
Papers on this Topic