Interaction-Oriented Programming

Concepts, Theories, and Results on Commitment Protocols

Munindar P. Singh

singh@ncsu.edu

Department of Computer Science
North Carolina State University
http://www.csc.ncsu.edu/faculty/mpsingh/

A Brief History of Programming

- **Applications:** Control of computations hidden in code; integration a nightmare
- **Workflows:** Control abstracted out; integration still difficult
- **Standards-driven orchestration:** Integration improved; antithetical to autonomy
- **Messaging:** Integration simplified by MoM and transformations; limited support for autonomy
- **Choreography:** Model conversations over messages; limited support for autonomy
- **IOP:** Interactions as first-class entities to promote autonomy, heterogeneity, dynamism

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-CDL
- *Collaboration:* a joint set of activities among business partners
  - Akin to real business; essential for SOAs
  - Process $=$ Protocol $+$ (each partner's) Policies

A Protocol is a Choreography, But

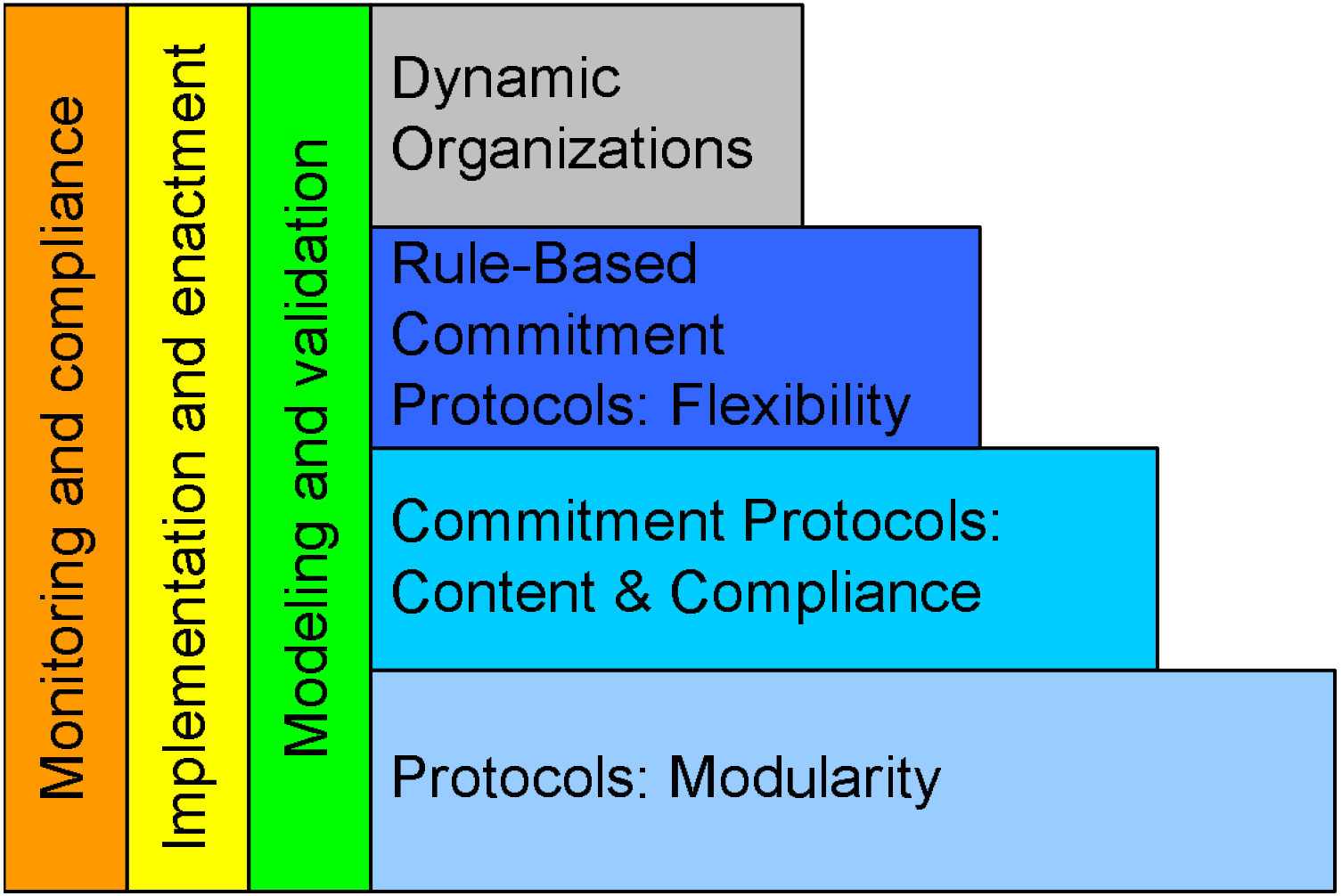
Contentful: interactive, reusable portion of a process in the collaborative view

- Describes interactions as classes describe objects
- Specifies well-defined roles
  - Capturing obligations on an endpoint
  - Setting local policies while complying with a protocol
- Specifies messages and how they affect interaction state
- Stored in a repository, a reusable asset
- Refined and composed for implementation

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
- Shortcomings
  - Generally highly limited: two party, request-response protocols
  - No commitments; no formal semantics
  - Limited support for modeling, enactment

Emphases of Collaboration

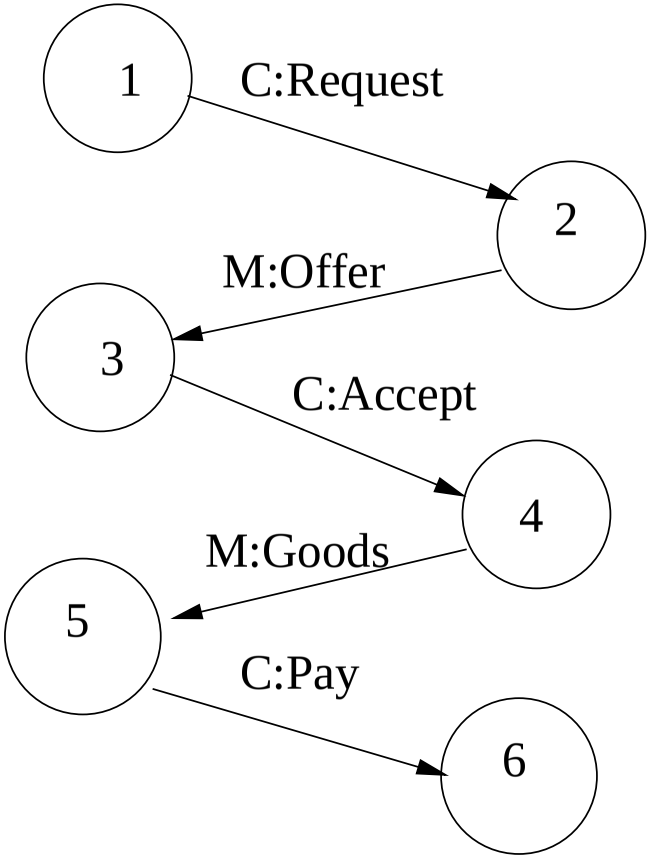


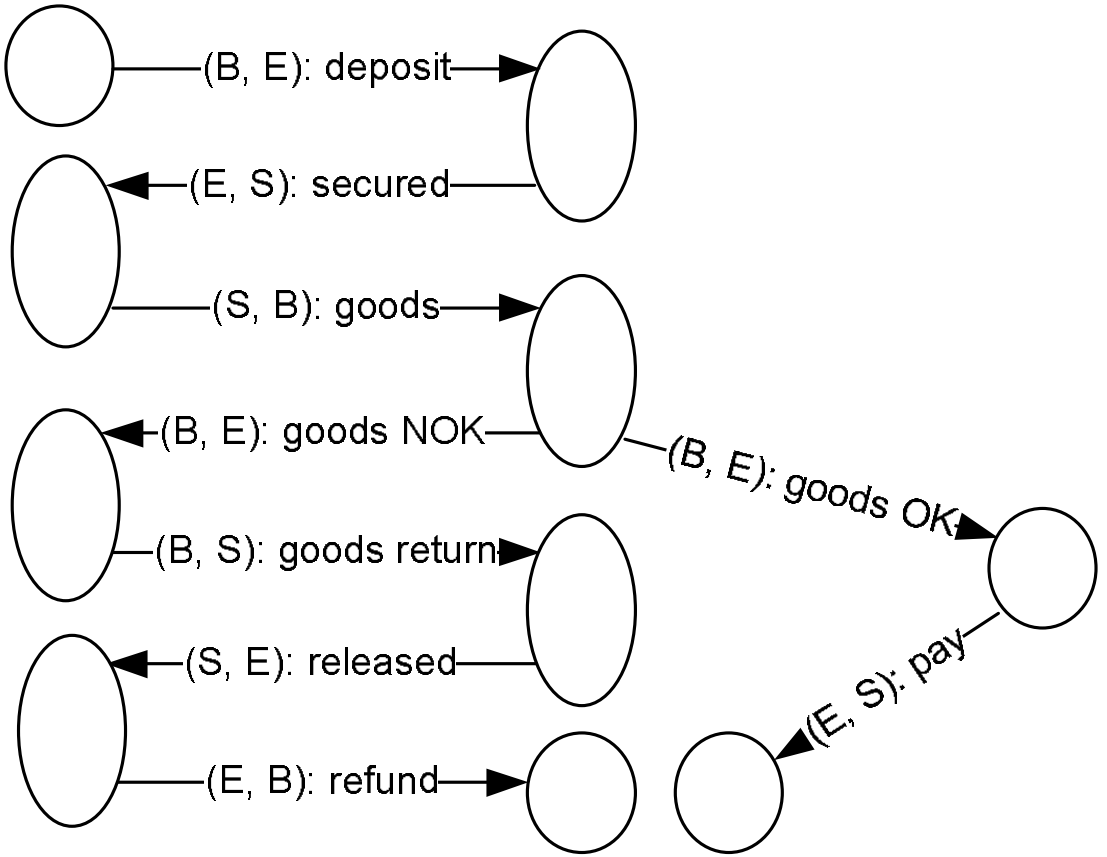
Commitments

Atoms of contracts

- Support reuse via abstractions for *refinement* and *aggregation* of protocols
- Content for protocols
  - Specify what the protocol should accomplish
  - Identify deviations are legitimate and what aren't
- Manipulated depending on context: assign, delegate, release . . .

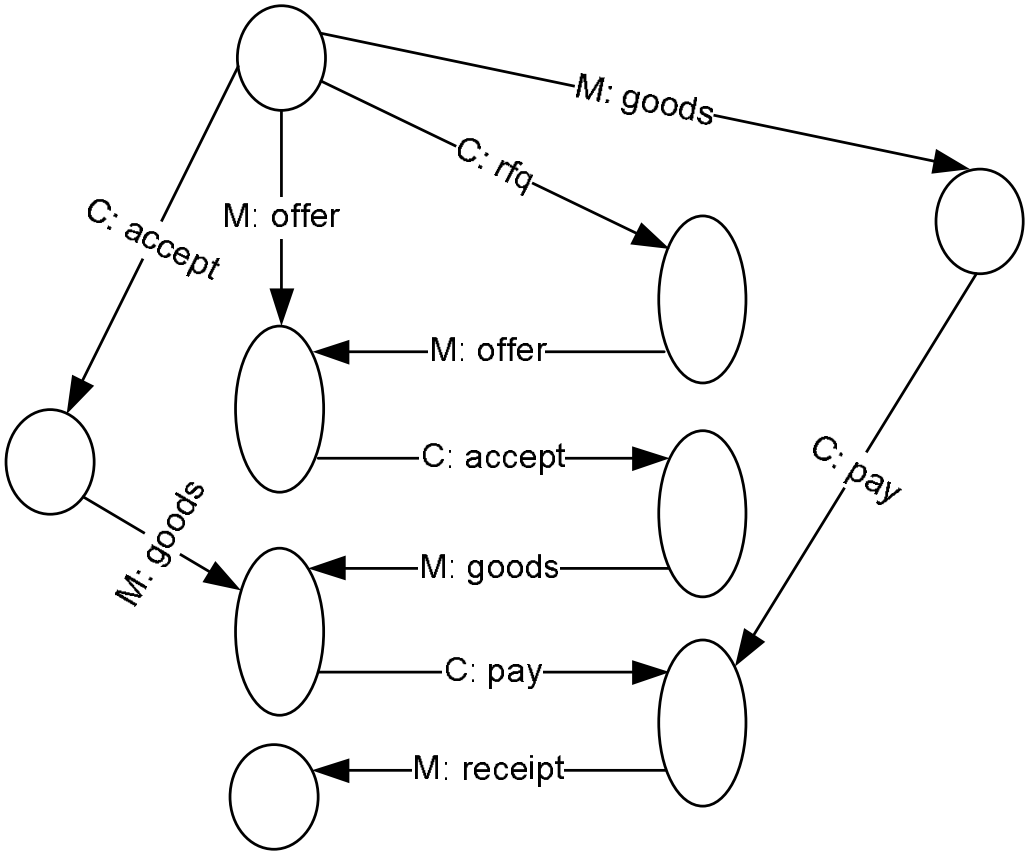
NetBill and Escrow Protocols





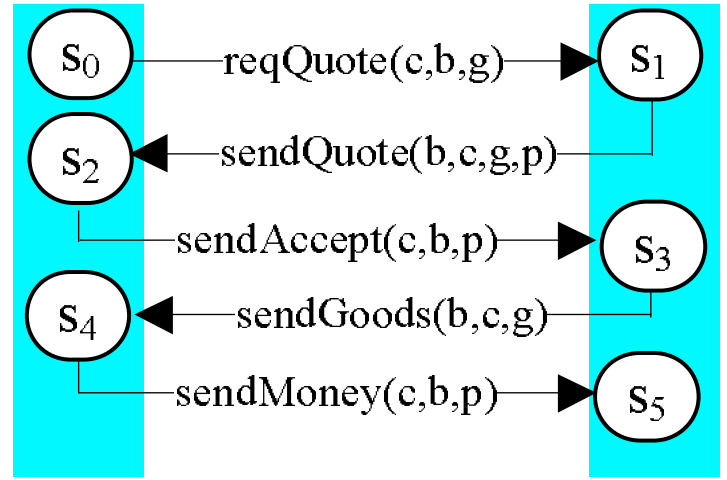
Protocols as Commitment Machines

- Roles; propositions; messages; states; good states
- Theory of action about messages and states
- Transitions by logical inference

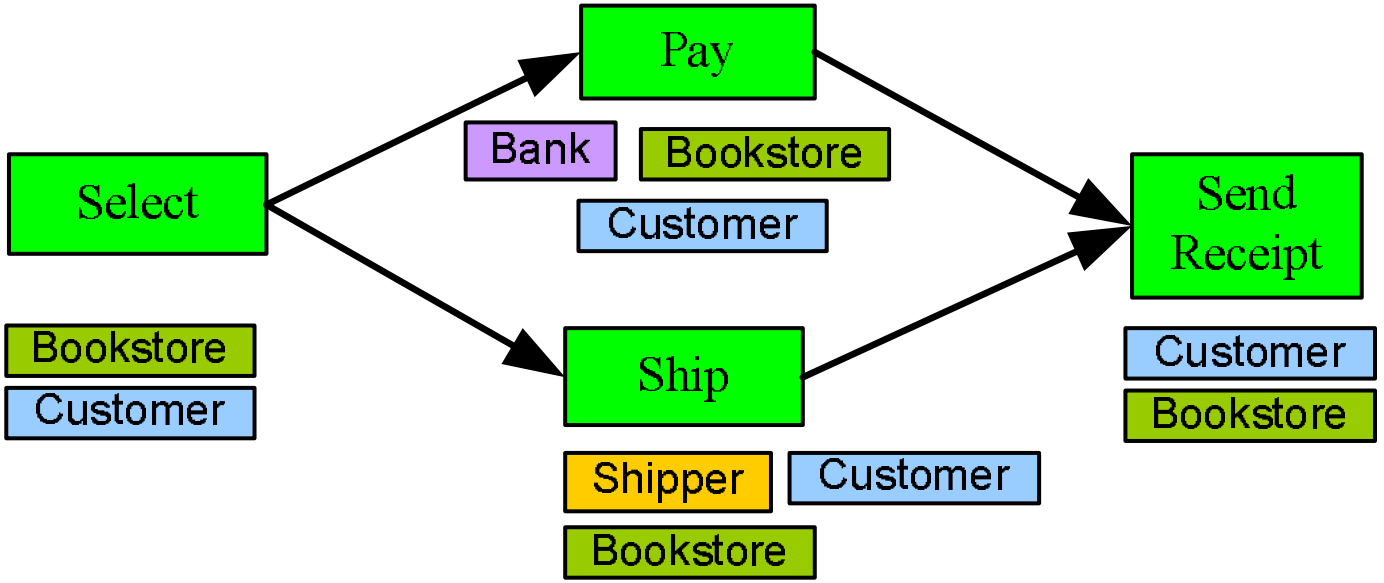


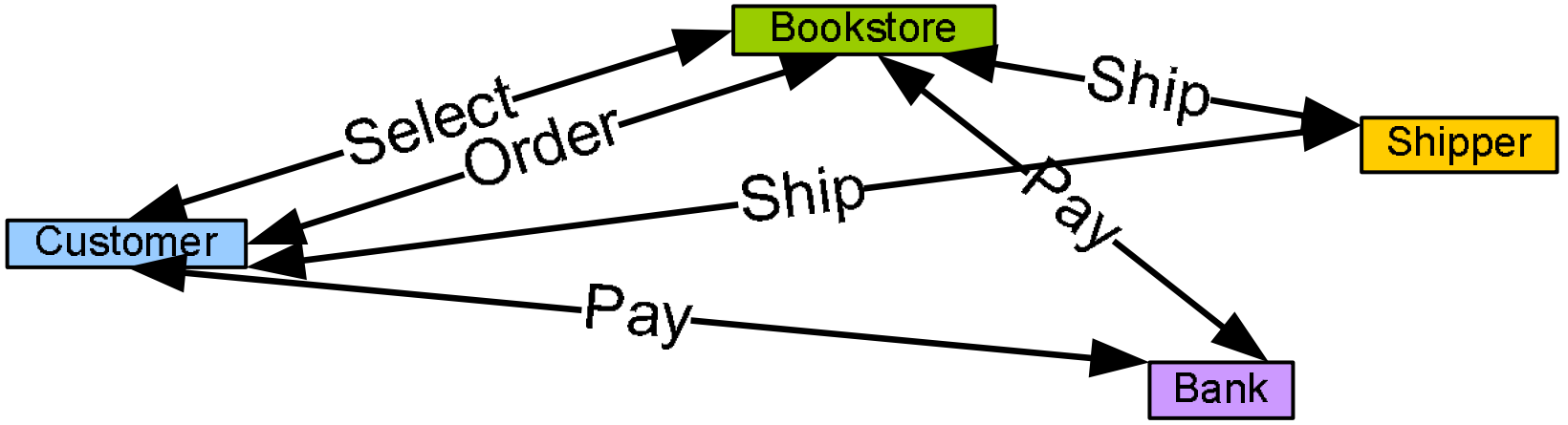
Simple Scenario and Example Run

- 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



Process View: Flow or Protocol





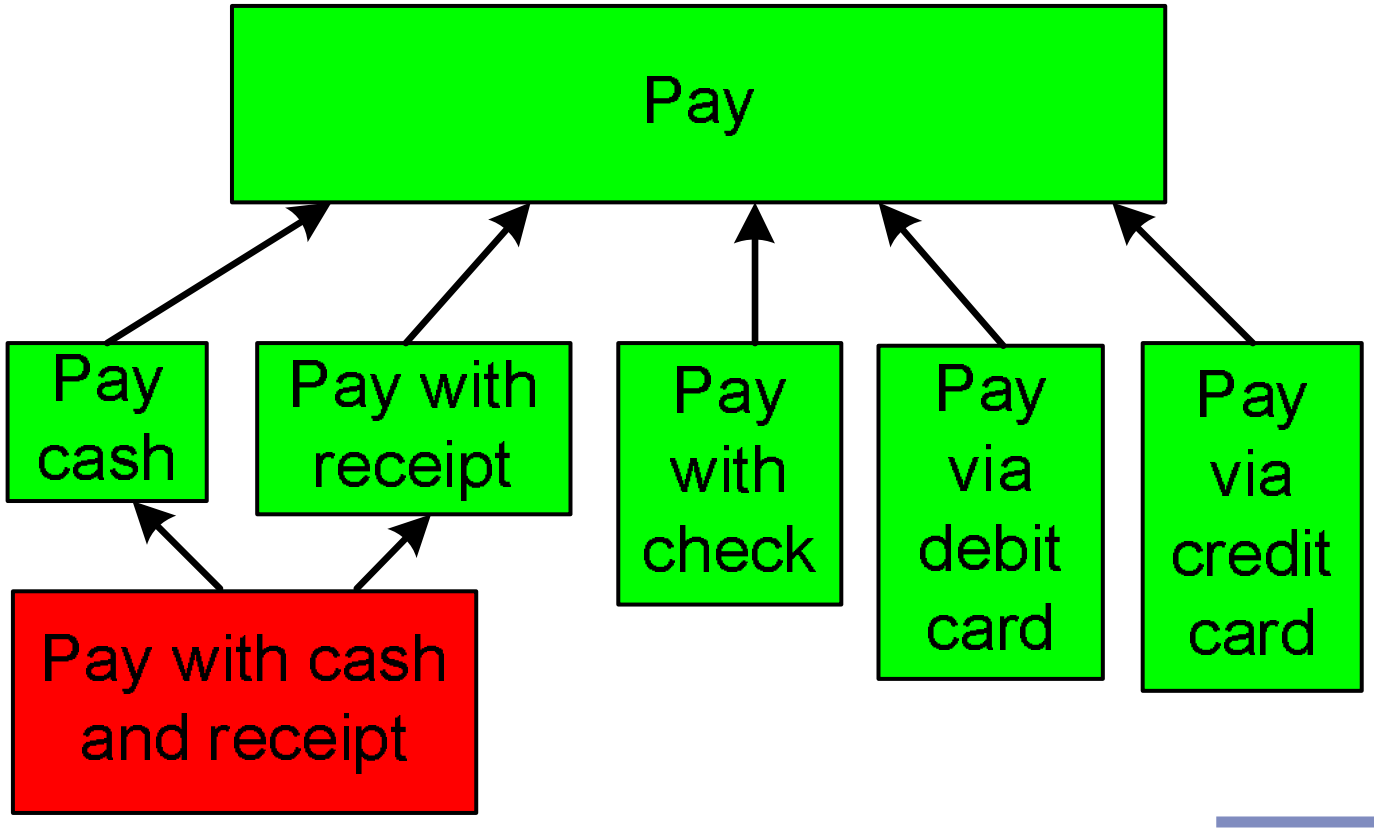
Challenges: Modeling

- *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

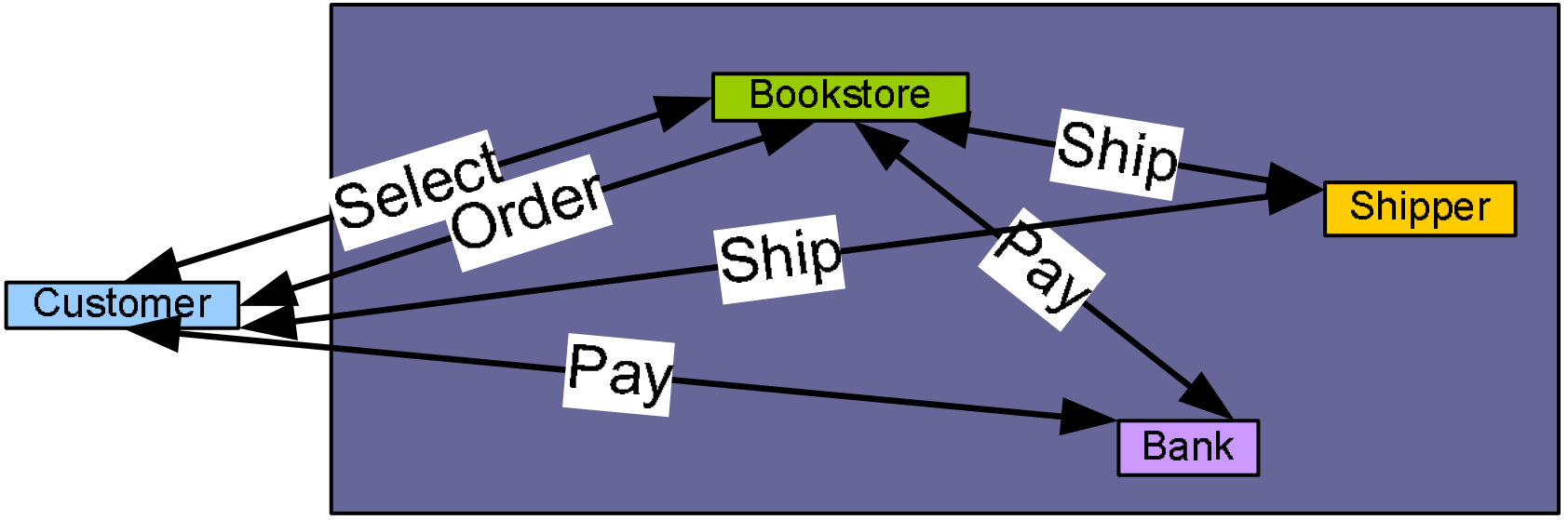
Selection criteria for protocols

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

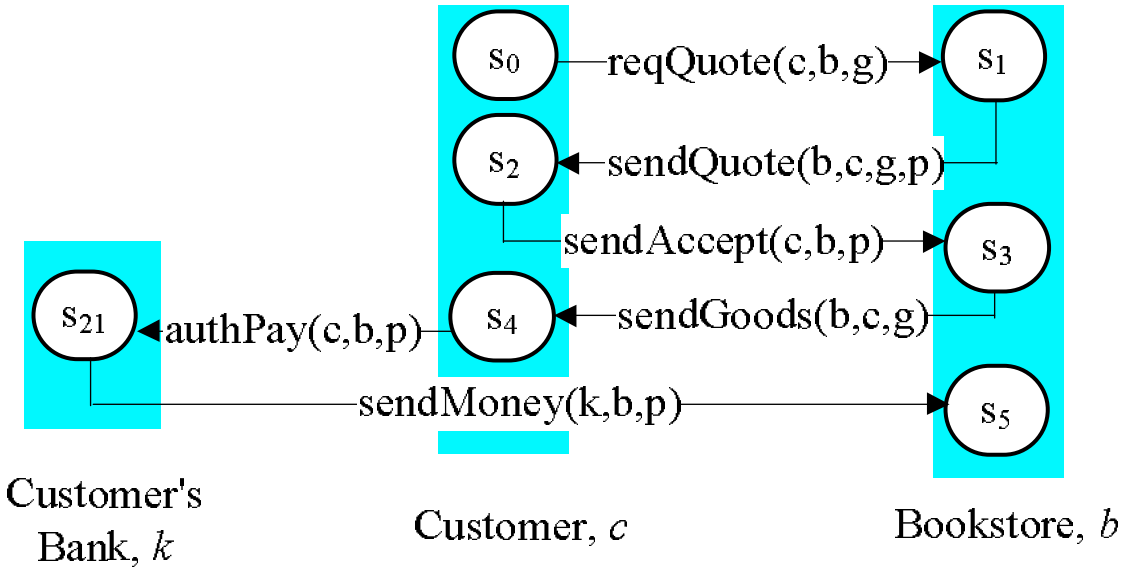


Aggregation of Protocols

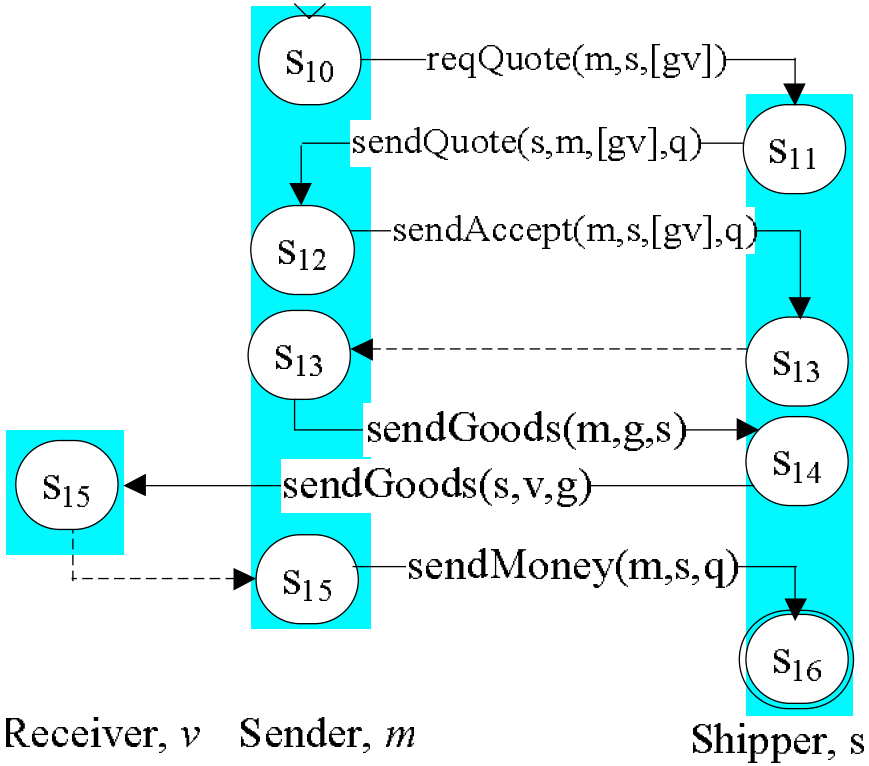
- A simplified protocol may be revealed to a give role
- Decisions could be taken internally but not exposed



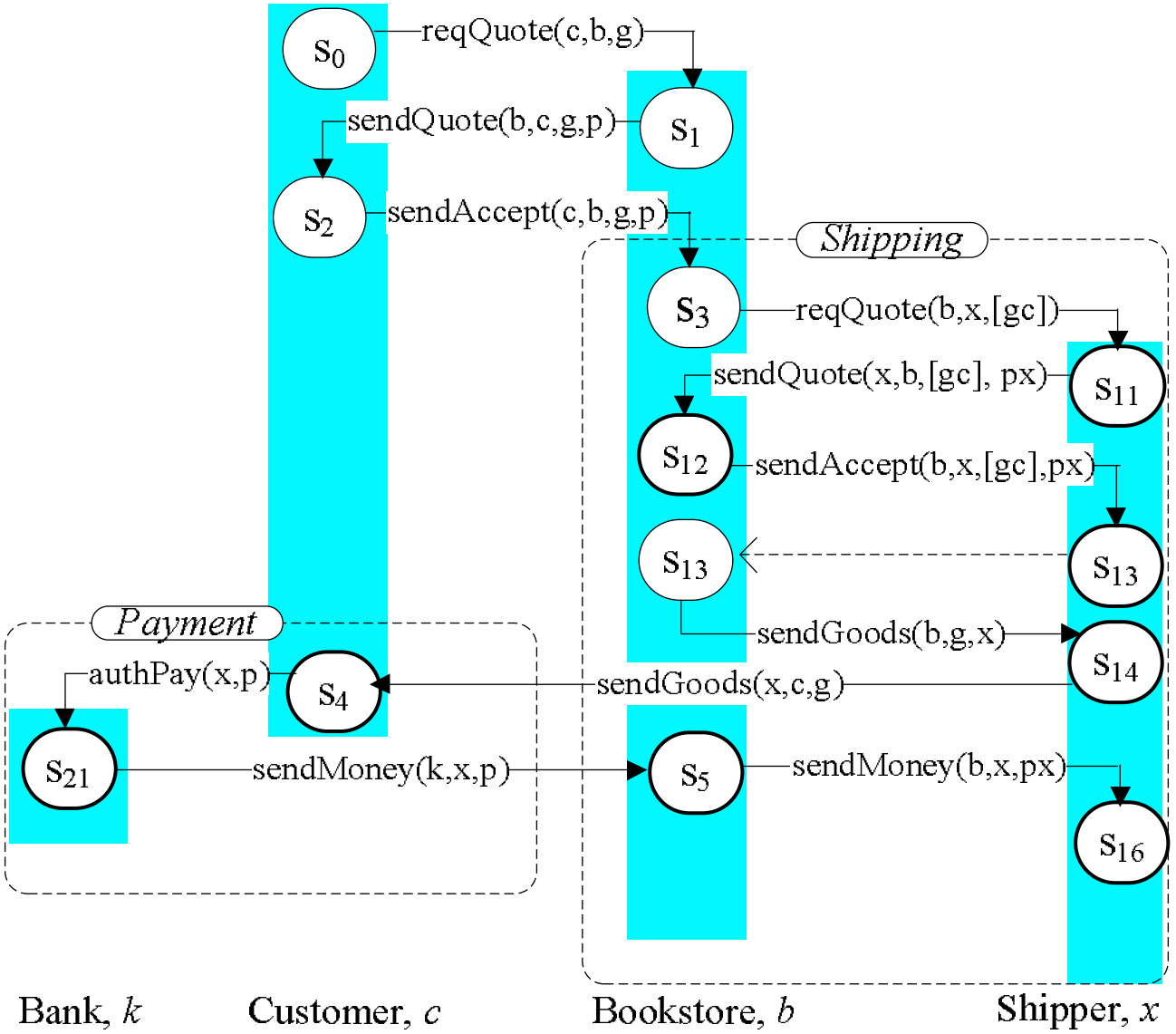
Example Run: Pay via Bank



Example Run: Shipper Protocol



Example Run: Composed Purchase

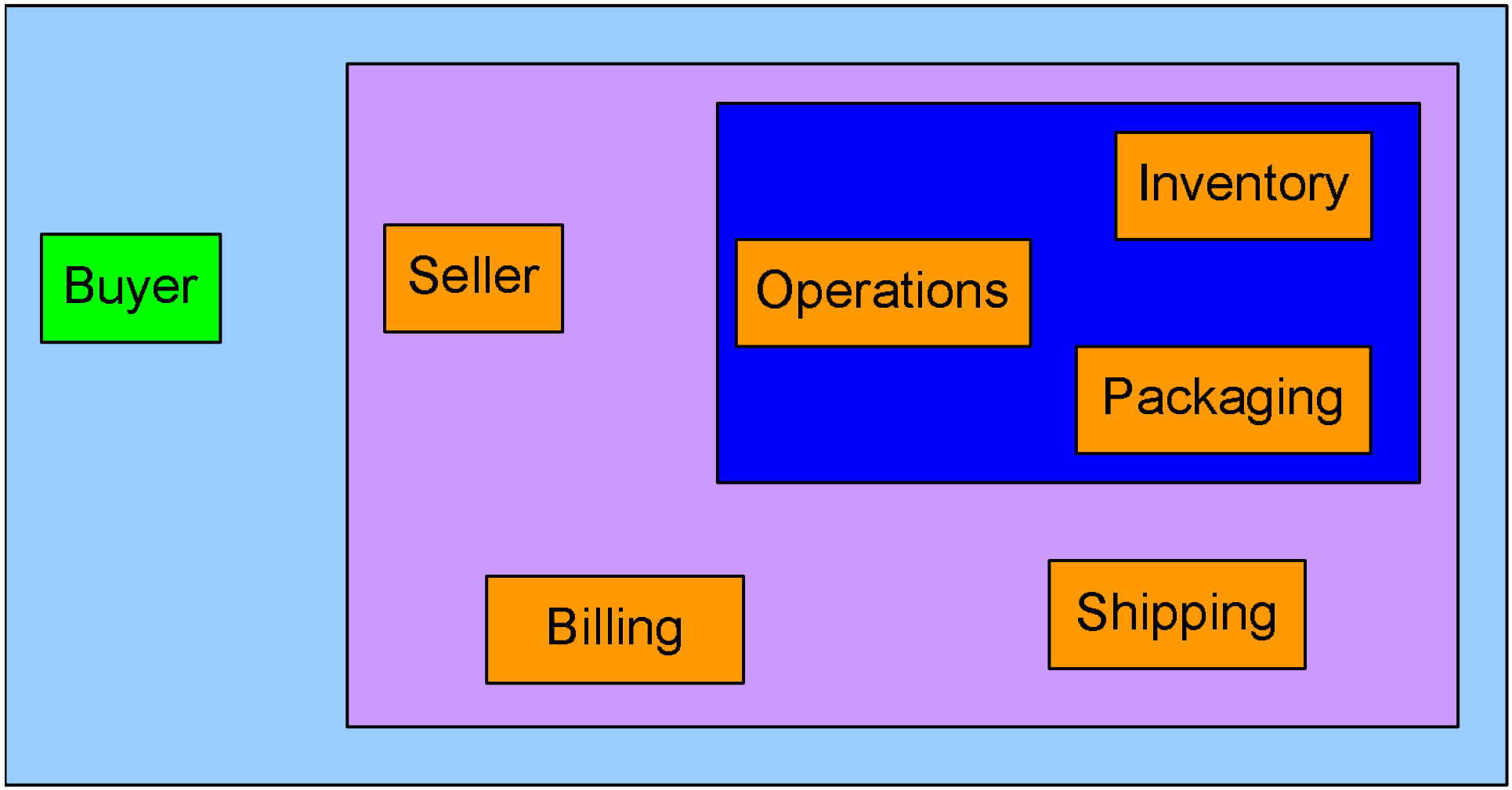


Challenges: Enactment

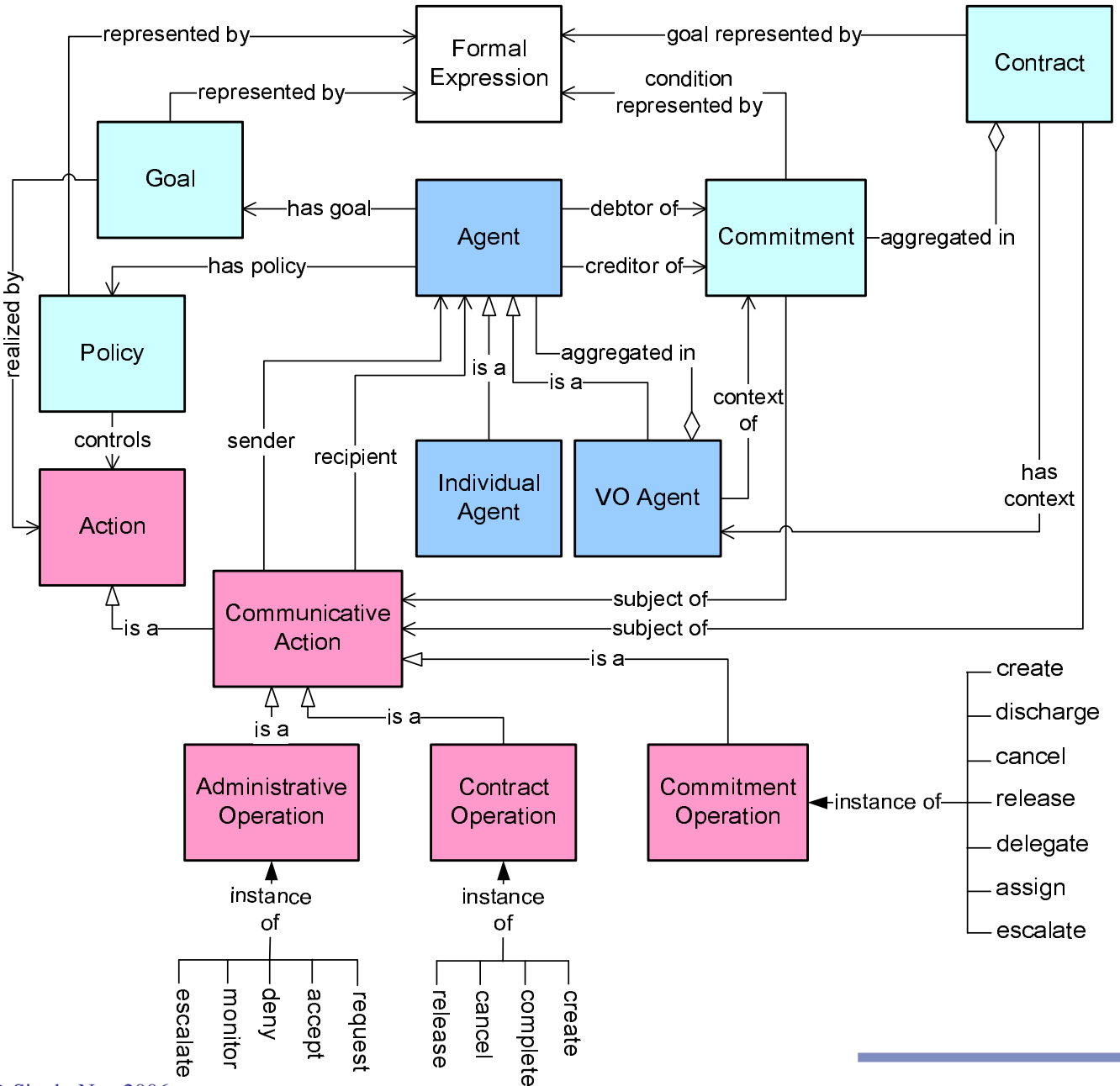
- *Behaving adaptively:* decide dynamically to ship before payment to trusted Cs
- *Handling exceptions*
  - External problems: cannot ship book
  - Context-sensitivity: not legal for kids
  - Detecting violations: no payment; book arrives damaged
  - Correcting violations: remind, complain, refund, . . .
- *Exploiting opportunities:* combine orders from same C

Organization Example

Spheres of Commitment

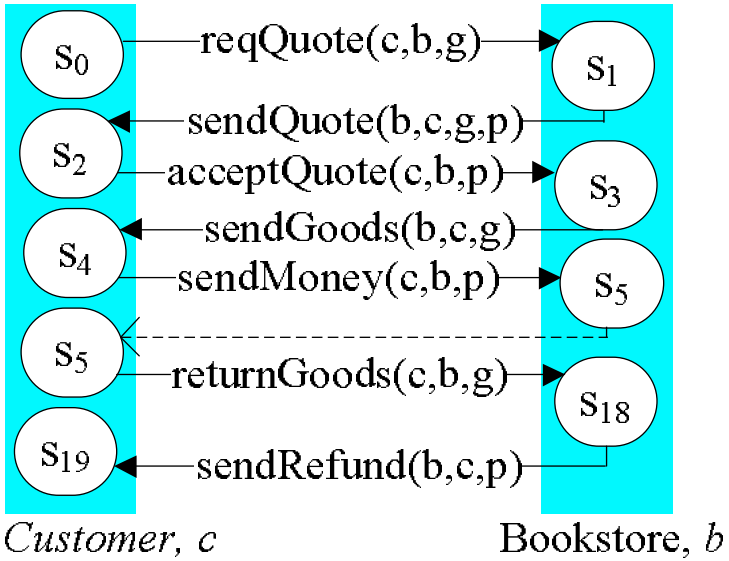


Conceptual Model for Organizations



Enacting Protocols in Organizations

Organizations specify, monitor, verify compliance



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

Contracts and Organizations as Duals

- *Contract:* static entity capturing relationships among agents
  - A contract arises within a VO where the contracting agents are peers
  - The enclosing VO would have been created by a prior contract
- *Organization:* dynamic (evolving) entity: hosts commitments, contracts, authorities
  - Created through a contract
  - Provides a basis for creating, manipulating, and enacting contracts

Process = Protocol + Policies

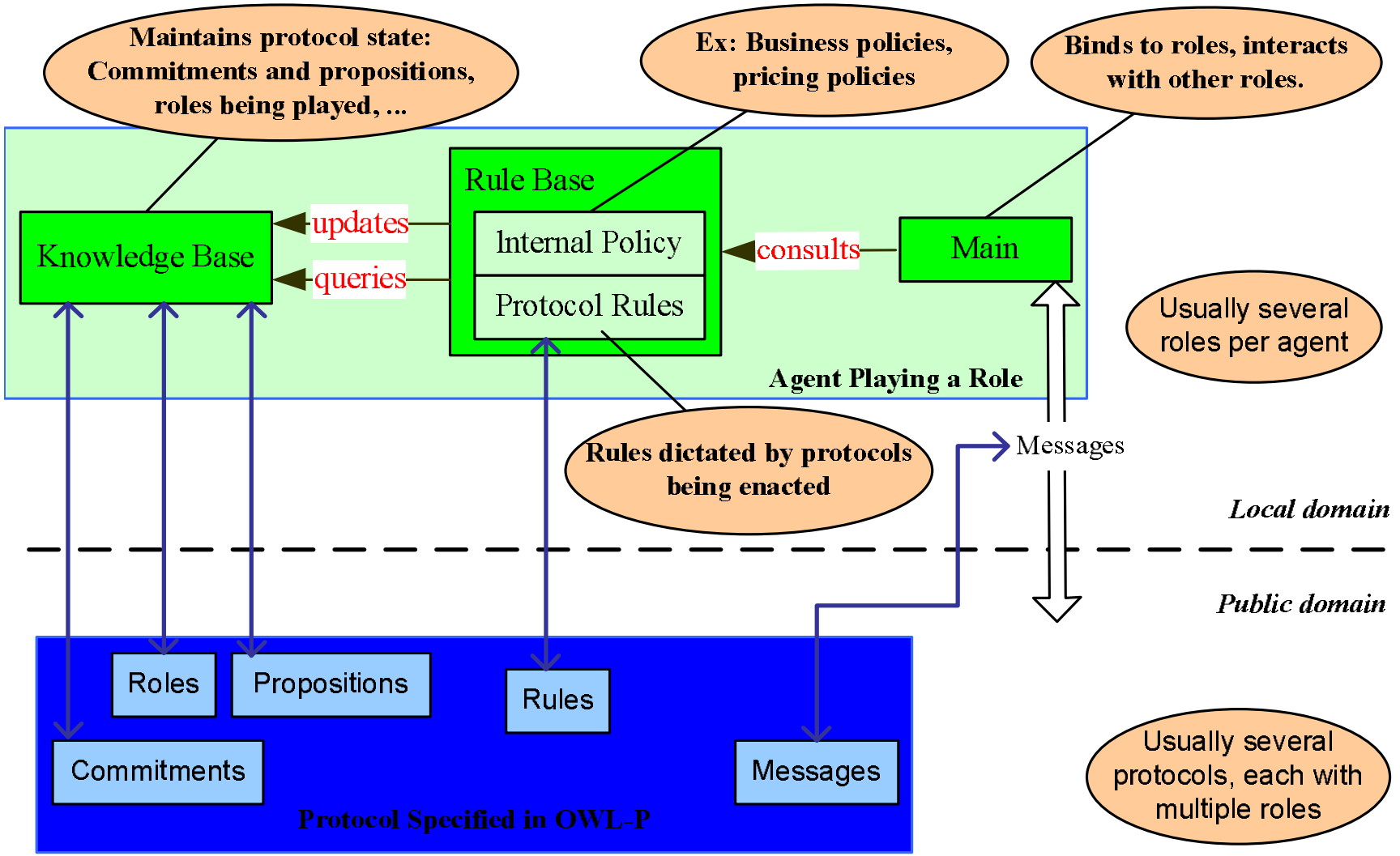
- Operational patterns
  - Time outs, remind, garbage collect, . . .
  - Decisions to manipulate: delegate, assign, . . .
  - Winograd & Flores and other such
- Methodologies, e.g., enhancing Tropos:
  - Cover functional reqs via protocols
  - Refine protocols for nonfunctional reqs
  - Enact protocols dynamically based on agent policies and context

Protocol Semantics and Pragmatics

Proposal being developed

- *Repository:* tools for customized access
- *Recombination:* Encourage novel combinations
  - Formal representations of protocols
  - Power tools for enhancing repository via composition
- *Collective intelligence:* Capture social knowledge about above
  - Correctness of particular compositions
  - Contexts where compositions work or fail

Architecture



Papers on this Topic

- Recent papers in AAAI, AAMAS, ICWS, ICSOC, IJCAI, SCC address parts of the above vision
- "Agent Communication Languages: Rethinking the Principles." *IEEE Computer*, 31(12):40–47, Dec 1998
- "Reasoning About Commitments in the Event Calculus: An Approach for Specifying and Executing Protocols." *Annals Math & AI*, 42(1-3), 2004
- "Verifying Compliance with Commitment Protocols." *J. Autonomous Agents & MAS*, 2(3):217–236, Sep 1999
- "An Ontology for Commitments in Multiagent Systems." *AI & Law*, 7:97–113, 1999

Implementation Agenda

Bridging the gap between current architectures (e.g., ESB or enterprise service bus) and user needs

- Capture and generalize scenarios known to be of user interest
- Develop a repository of validated protocols
- Extend and incorporate current tools: OWL-P (protocols) and MAVOS (multiagent virtual organization system)