What is a Service?

- Traditional, as in WS: Abstraction of a computational object
- Improved conventional: Abstraction of a "capability"
- Real life: Participant in a service engagement *Business Relationships*
  - Independent parties *business partners*
  - Symmetric relationships: complementary capabilities and goals
Service Engagements

Crucial to the modern economy; major trend in computing

- Business interactions characterized by
  - Autonomous parties
  - Coproduction
  - Contractually constrained
  - Symmetric relationships: complementary capabilities and goals

- Contrast with web services, which merely abstract a computational object
Challenges for Policies and Decision Making

- **No unique locus:** separate policies for each autonomous participant
- **Dependence on business relationships**
- **Complexity of modeling**
  - Specifying vocabulary pertinent to service engagements
  - Determining where policy decisions apply
- **Idea:** Architecture for governance centered on **conversations**
  - **Domain-specific policies:** Incorporate monitoring and responding to events
  - **Generic policies:** Altering business relationships
Elements of a Service Engagement

- Transactional: main purpose and enactment of the engagement
- Structural: partnerships and contracts
- Contextual: setting of the engagement
Traditional Approaches
Quite Unlike a Real-Life Service Engagement

- Take participants flows (e.g., in BPEL) as units of abstraction
  - Mix private policies and public interactions
  - Proprietary: may not be available for reuse
  - Context-laden: even when available, cannot be readily reused
- Focus on low-level (e.g., WS-CDL) or data-level meanings (e.g., OWL)
  - Ignore business-level significance of messages
  - Ambiguous; not verifiable
A Real-Life Service Engagement

Expressed as Interacting Flows

AGFIL (Insurance Company)

- Notify Lee CS
- Obtain claim form
- Check claim form
- Amend estimate
- Reconcile info
- Finalize claim

Lee CS (Claim Handler)

- Obtain details
- Estimate < 500
- Contact garage
- Assign adjustor
- Agree repair
- Check invoice

EuropAssist

- Notify AGFIL
- Assign garage
- Validate info
- Gather info

(Call Center)

- Receive car
- Estimate repair cost
- Inspect car
- Repair car
- Invoice

(Cone of many)

Mechanic
Introducing Protocols and Policies

Centered on Interaction

- Interaction protocols are modules of abstraction
  - Separate from policies, which are inherently private
  - Help identify policy points where policies apply
  - Modular, reusable

- Express interaction meanings chiefly via commitments, which
  - Are atoms of contractual relationships
  - Enable correctness checking of contracts
  - Yield precise meanings and verifiability
What is a Protocol?
A Description of Business-Level Interactions

A reusable unit of interaction

- Analogous to an abstract class or interface for objects
- Specifies well-defined roles: \textit{Buyer, Seller, Bank, Shipper}
- Specifies messages among the roles and how they affect interaction state: \textit{e.g., place order, pay, fulfill order}
  - Capturing commitments on an endpoint (business partner playing a role)
  - Setting local policies while complying with a protocol
- Stored in a repository, i.e., as an asset or resource in its own right
- Refined and composed for implementation
Commitments

```
Later

Org

Has context

Has creditor

Principal

Has debtor

Commitment

Consequent

Antecedent

Practical Commitment

Dialectical Commitment

ship
```
Commitment Life Cycle (and Patterns)

\[
C(d, c, T, T', consequent, consequent, consequent, pay, pen)
\]

(a) Commit

(b) Relieve
Commitment Operations

- `create(CC(x, y, p, q))` establishes the commitment
- `detach(CC(x, y, p, q))` turns it into a base commitment
- `discharge(CC(x, y, p, q))` satisfies the commitment
- `cancel(CC(x, y, p, q))` cancels the commitment
- `release(CC(x, y, p, q))` releases the debtor from the commitment
- `delegate(z, CC(x, y, p, q))` replaces `x` by `z` as the debtor
  - `x` remains ultimately responsible (in our work)
- `assign(w, CC(x, y, p, q))` replaces `y` by `w` as the creditor
Patterns for Delegate

(a) Transfer responsibility

(b) Retain responsibility

(c) Escalate

(d) Withdraw delegation
Contextual Patterns: Penalize and Revert

(a) Penalize

(b) Revert offer

*Added was implicit earlier*
\[ CD = C(s, b, eBay, pay, goods) \]

No goods : violated

\[ C(eBay, b, eBay, violated(a), create(C_1)) \]

\[ C_1 : C(s, b, eBay, pay, goods) \]

\[ C_1 : refund the payment + 10\% \]
Define an engagement for a purchase (service)

Buyer pays / Seller delivers some item

- Negotiate
- Execute

Order —> Ship —> Pay

Operational

C (Buyer, Seller, goods, pay)
C (Seller, Buyer, pay, goods)

Treat Ship & Paying as delegations
A Purchase Service Engagement

(a) Pair of conditional commitments describing purchase

(b) Introducing bank and shipper via delegations of commitments

(c) Allowing buyer to skip payment or get a refund upon returning goods
A Real-Life Service Engagement (Repeated)