A Reference Model for Cross-Organizational Coordination Architectures

Eddy Truyen, Wouter Joosen
Outline

- Cross-organizational coordination
- Motivation for a reference model of cross-organizational coordination
- Overview of Reference model
- Lessons Learned
Cross-organizational service provisioning

Virtual Enterprise

Service Consumer

Service Provider

Contract

Invocation

Monitor

Control

Result

(c) Paul Grefen
Cross-organizational Coordination

• Definition
  Ø *the managing of dependencies between autonomous organizations in order to foster harmonious interaction between them“*

• Coordination involves a certain *agreement, i.e., a set of rules of engagement, that must be complied with by all participating organizations for the business relationship to be exploited in a safe and harmonious way.*
Coordination architectures

- Agreement Language
  - The agreement language is used to specify agreements
- Agreement
  - The agreement is established and executed by the coordination middleware
- Coordination Middleware
Motivation for a reference model

• A reference model ...

$\exists$ is defined as a standard decomposition of a problem into parts that cooperatively solve the problem. [Bass/Clemens/Kazman].

$\exists$ is an important prerequisite for bootstrapping standardization processes

• Within a business domain, a reference architecture for cross-org coordination can be defined

$\exists$ A reference architecture is a commonly accepted decomposition into components with standardized interfaces

$\exists$ A reference model is a first step to obtain such reference architecture.
My motivation for a reference model

Understanding:
- Before the study:
  - What I do know
  - What I don't know
- After the study:
  - My understanding of cross-org understanding

DISCLAIMER
Minimal Representative Data set

- Ponder [Sloman]
- Law-governed Interaction [Minsky]
- Business contract language and architecture (BCL) [Milosevic]
- Crossflow project [Grefen]
- FSM-based contracts in the Tapas project [Molina-Jimenez]
- GlueQoS [Wohlstdler]
- T-BPEL [S. Tai]
- Slang [Skene]
- Rapide [Luckham]
- Finesse [Berry]
- [Yildiz]
Themes of the study

- Mandatory

Diagram:
- Coordination Architecture
  - Agreement Type
  - Agreement Language
  - Coordination Middleware
Agreement types

- **Business Interaction Flows**
  - Message types
  - Allowed sequences of messages

- **Modal Constraints**
  - Obligations, Authorizations, Prohibitions, Timings

- **Quality of Service**
  - Availability
  - Response time
  - ....

- **Choice of technical protocols**
  - e.g., Transaction processing alternatives:
    - direct, queued, compensation-based
Agreement Language

- Management Concepts
- Conceptual Model
- Computational Model
- Implementation Technology

(optional)
Policies: Example from Ponder

[Obligation Policy] "A seller is required to ship an ordered item within 10 days to a Purchaser after the Purchaser has paid the order".

Ponder:

+Obligation(Seller)

{onEvent(OrderPaidbyPurchaser):
  action(OrderedItemSentToPurchaser)
  within(10 days)}
Contracts: Security example from GlueQoS

Client and server organization negotiate about how to set up an secure channel

 Authentication: protects against unauthorized access
 Client-puzzle protocol (CPP): defends against denial-of-service attacks

GlueQoS:

Server:
- let cpu = SystemMonitor.cpuUsage, puzzlemax = 16
- in (  
  (CPP[size = cpu*puzzleMax/2] and Authentication)
  xor (Authentication and (cpu < 0.5) )
  xor CPP[size = cpu*puzzleMax]   )

Client 1:
- (Authentication)

Client 2:
- (CPP[size <=4] xor
- (CPP[size <=4] and Authentication)  )
Conceptual and Computational Model

Organization-neutral

Event patterns

State mappings

Organization-specific
Management concepts

Diagram:

- Management Concepts
  - Domains
  - Conflict Management
    - Application-specific Conflicts
    - Modality Conflicts
  - Groups
Agreement Language -- Language technology

- XML, e.g. CrossFlow
- WS-Policy and WSPolicyattachment, e.g. GlueQoS
- Object-oriented programming languages, e.g. Ponder
- the Meta Object Facility, e.g. SLAng
- Finite state machines, e.g. Tapas
- Prolog. E.g. Law-governed interaction
- Propietary, Business Contract Language
Coordination Middleware Functionalities

- Coordination Middleware
  - Functionalities
    - Establishment
      - Validation
      - Negotiation
      - Enforcement
    - Execution
      - Dismantling
      - Monitoring
      - Enactment
  - Architectural design
Architectural Design

Event Interceptor ➞ Controller

Coordination Middleware

Functionalities

Architectural design

Placement of Controller

Behavior

- Decentralized
- Centralized
- Reactive
- Proactive
Summary

- Coordination Architecture
  - Agreement Type
  - Agreement Language
  - Coordination Middleware
    - Conceptual Model
      - Management Concepts
    - Computational Model
      - Implementation Technology
    - Functionalities
      - Architectural Design
Lessons learned

**Differences between policy and contracts**
- Agreement types
- Middleware functionalities
- Management concepts
⇒ Opportunities for uniting these approaches

**Enactment**
- Open problem especially with respect to cross-organizational business process

**Complete coordination architectures**
- Agreement types
- Middleware functionalities