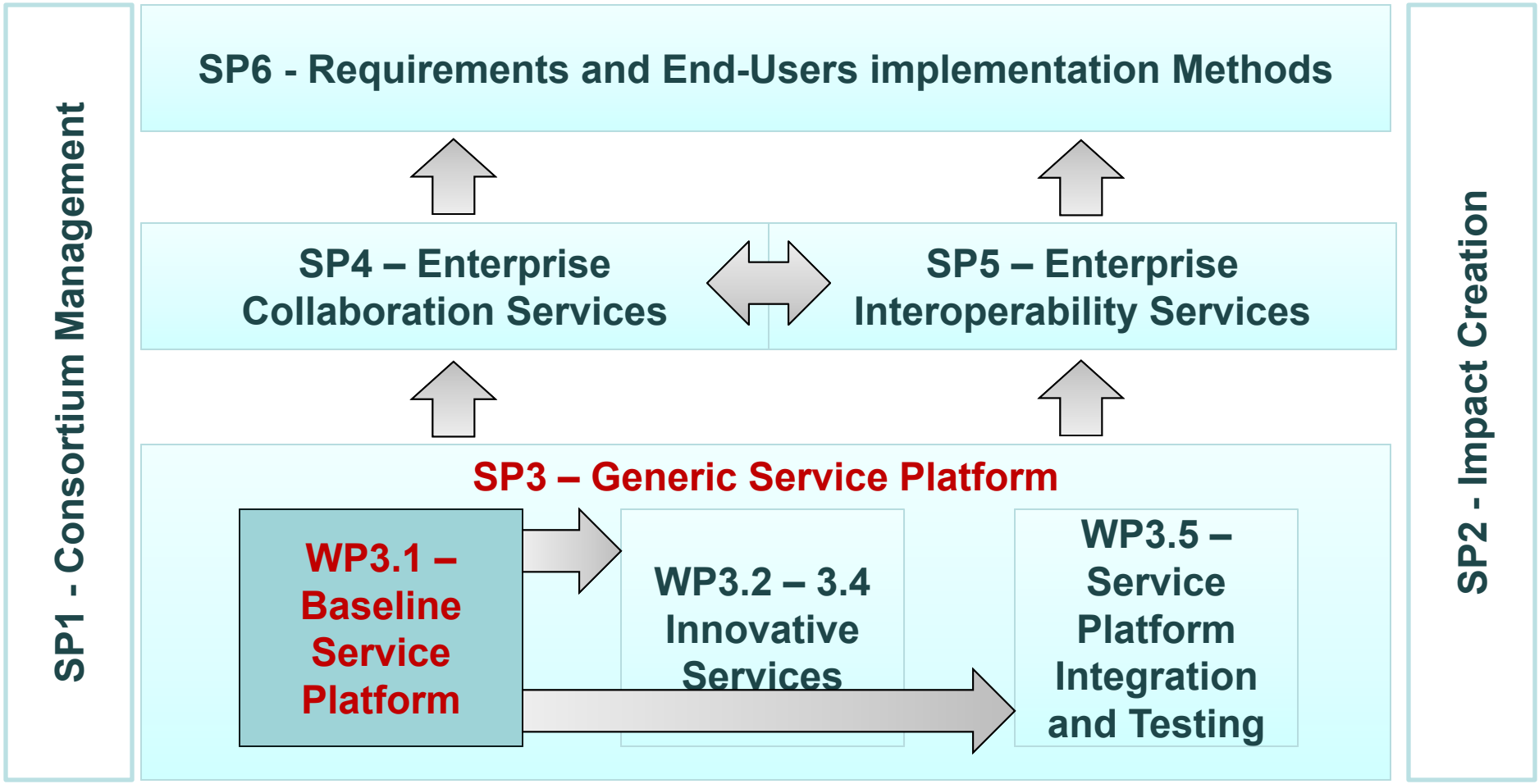
Enterprise COllaboration & INteroperability

SP3 & Semantic Web Services

Overview

COIN General Assembly, Budapest, 05.05.2009
Srdjan Komazec, UIBK

Context


SP1 - Consortium Management
SP6 - Requirements and End-Users implementation Methods
SP4 – Enterprise Collaboration Services
SP5 – Enterprise Interoperability Services
SP3 – Generic Service Platform
WP3.1 – Baseline Service Platform
WP3.2 – 3.4 Innovative Services
WP3.5 – Service Platform Integration and Testing
SP2 - Impact Creation

Outline

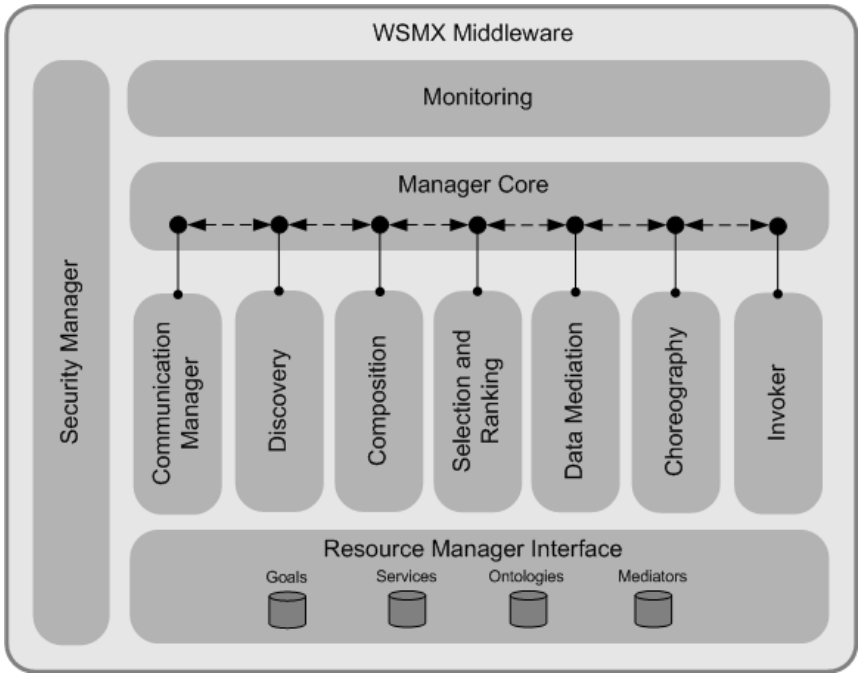
- Objectives
- Deliverables
- Service Baseline Platform Assets
- Service Baseline Platform Architecture
- Results
- Web Service Execution Environment

Objectives of SP3

- Analysis of assets, requirements and design of the Baseline Service Platform based on
  - Semantically Enabled Service oriented Architecture (SESA)
  - Digital Business Ecosystem platform
  - Multi-Agent framework
  - TrustCoM platform
- Develop the Baseline Service Platform
- Provide an initial Scenario for the Platform
- Pave the road for SP3 Innovative Services

SESA

- Provide support for Web Service lifecyle adopting semantics (Web Service Modeling Ontology)
- Automation of
  - Service Discovery
  - Service Invocation
  - Service Ranking
  - Service Composition
  - …
- Web Service Modelling eXecution environment (WSMX) is the Reference Implementation
- Ongoing Standardization in OASIS SEE TC



Some fundamental characteristics are missing to allow its adoption in the Enterprise context

WSMX: what is missing

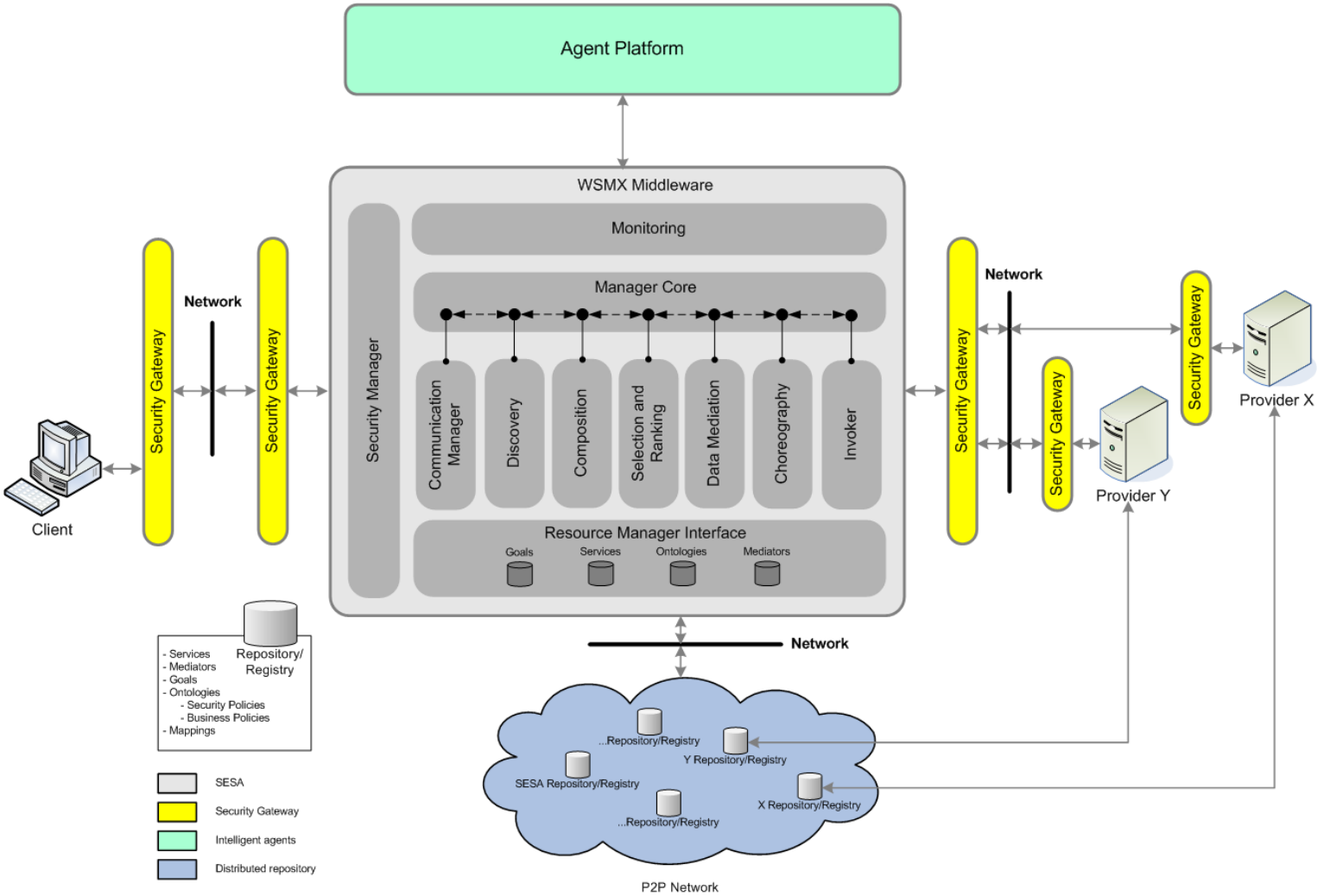
- Support for Security
  - Authorization/authentication
  - Trust
  - Encrypted Messaging

  TrustCom
- Support for Monitoring
  - Service Quality Monitoring
  - Fault Monitoring
- Scalable Grounding Mechanism
  - Lifting and Lowering of XML to/from WSML
  - Easy deployment of grounding
- Support for Pervasive and Scalable Model Repositories
  - Distributed Repositories and Registries
  - Data replication
  - Model reuse and sharing

  DBE
- Support for Complex Business Interaction
  - Business Process
  - Negotiation

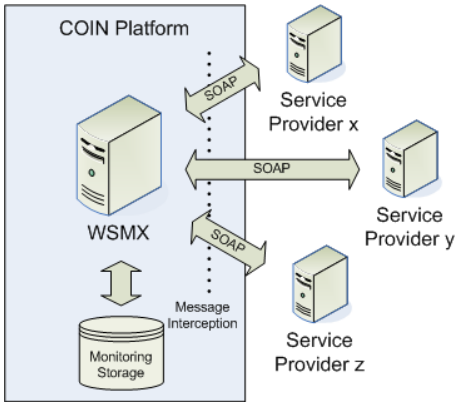
  Multi-Agent framework

Service Baseline Platform Architecture



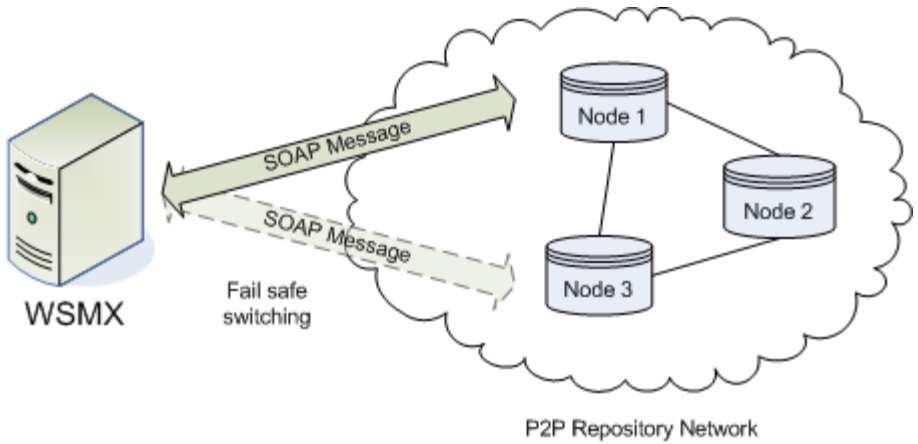
Results

- WSMX Grounding
  - Decoupled solution based on SAWSDL and XSLT
- WSMX Monitoring
  - RDF storage for storing data regarding the service invocations



Results

- WSMX and P2P Repository Integration
  - SOAP based integration
  - P2P Repository is yet another WSMX Resource Manager implementation
  - Support for fail-safe operation

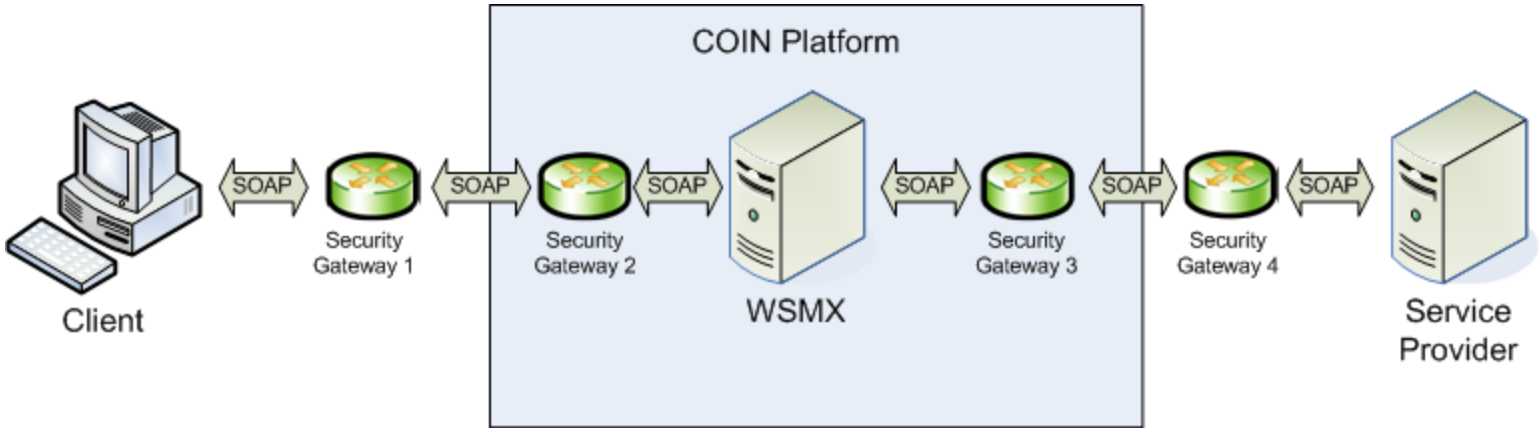


Results

- WSMX and Agent platform
  - SOAP based integration
  - WSMX delegates specific tasks to the Agent platform and vice versa

Results

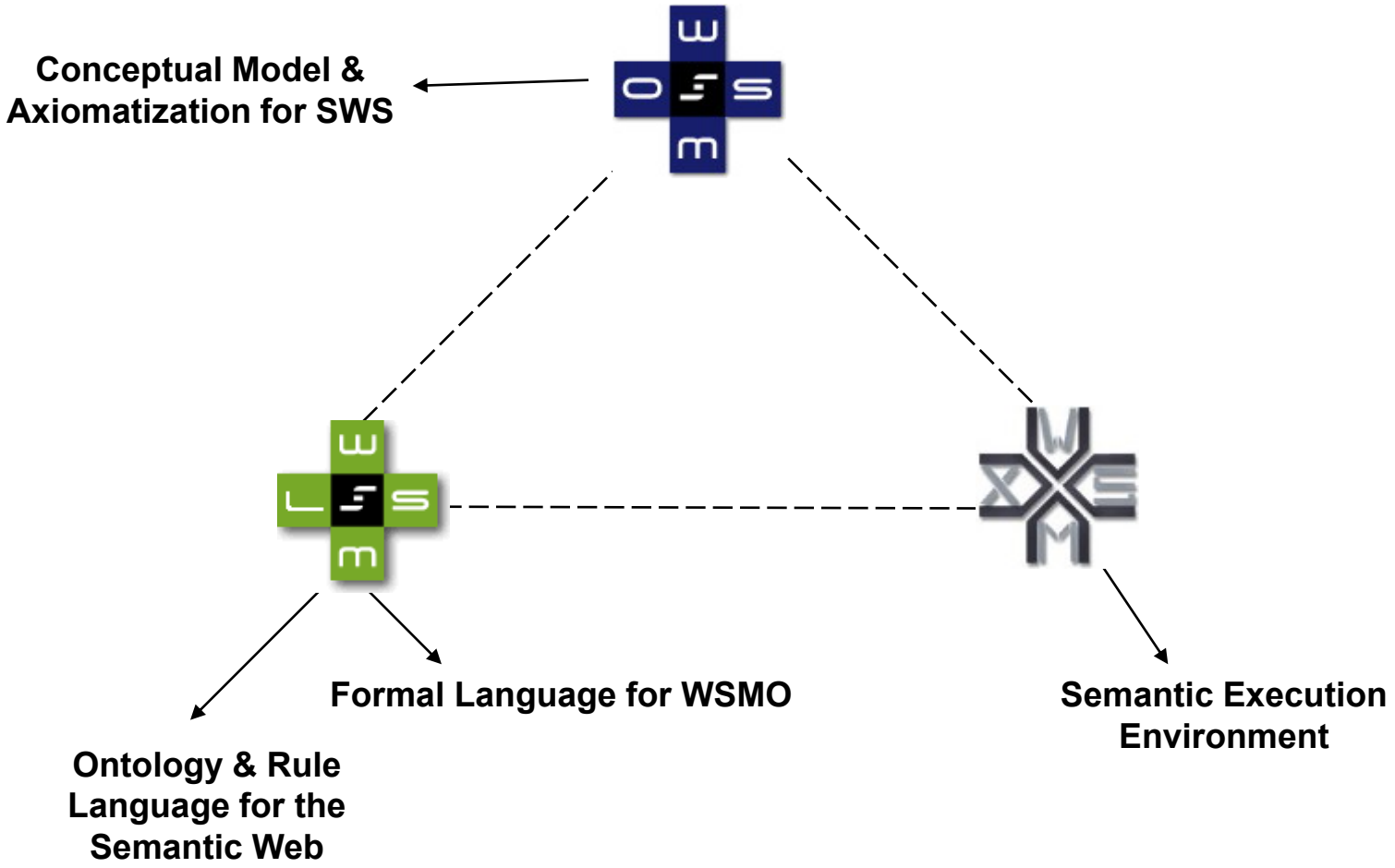
- WSMX and Security Gateways
  - SOAP-based integration,
  - Service requesters, WSMX and service providers are shielded by the pairs for security gateways



WSMX Overview

Introduction

Relation to WSMO and WSML



Introduction

WSMX…

- … is comprehensive software framework for runtime binding of service requesters and service providers,
- … interprets service requester's goal to
  - discover matching services,
  - select (if desired) the service that best fits,
  - provide data/process mediation (if required), and
  - make the service invocation,
- … is reference implementation for WSMO,
- … has a formal execution semantics, and
- … is service oriented, event-based and has pluggable architecture
  - Open source implementation available through Source Forge,
  - based on microkernel design using technologies such as JMX.

Introduction

Design principles

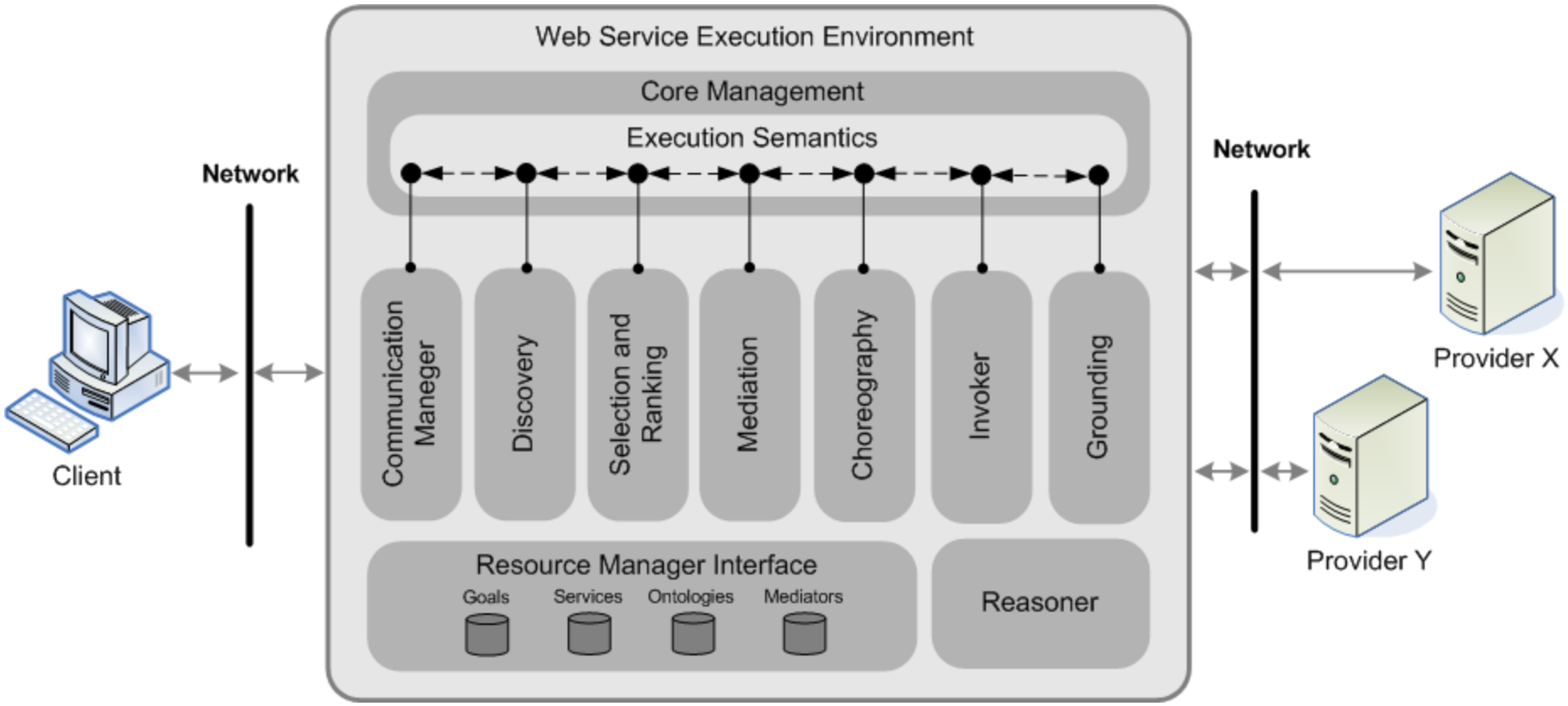
- Service-oriented principle
  - Service reusability, loose coupling, abstraction, composability, autonomy, discoverability,
- Semantic Principle
  - Rich and formal description of information and behavioral models enabling automation of certain tasks by means of logical reasoning,
- Problem-solving principle
  - Goal-based discovery and invocation of services, and
- Distributed principle
  - Executing process across a number of components/services over the network, thus promoting scalability and quality of process.

Lifecycle

1. Discovery - determines usable services for a request,
2. Composition - combine services to achieve a goal,
3. Selection - chooses most appropriate service among the available ones,
4. Mediation- solves mismatches (data, protocol, process) hampering interoperation,
5. Choreography – interactions and processes between the service providers and clients,
6. Grounding – lifting and lowering between the semantic and syntactic data representations, and
7. Invocation - invokes Web service following programmatic conventions.

WSMX

Current middleware status



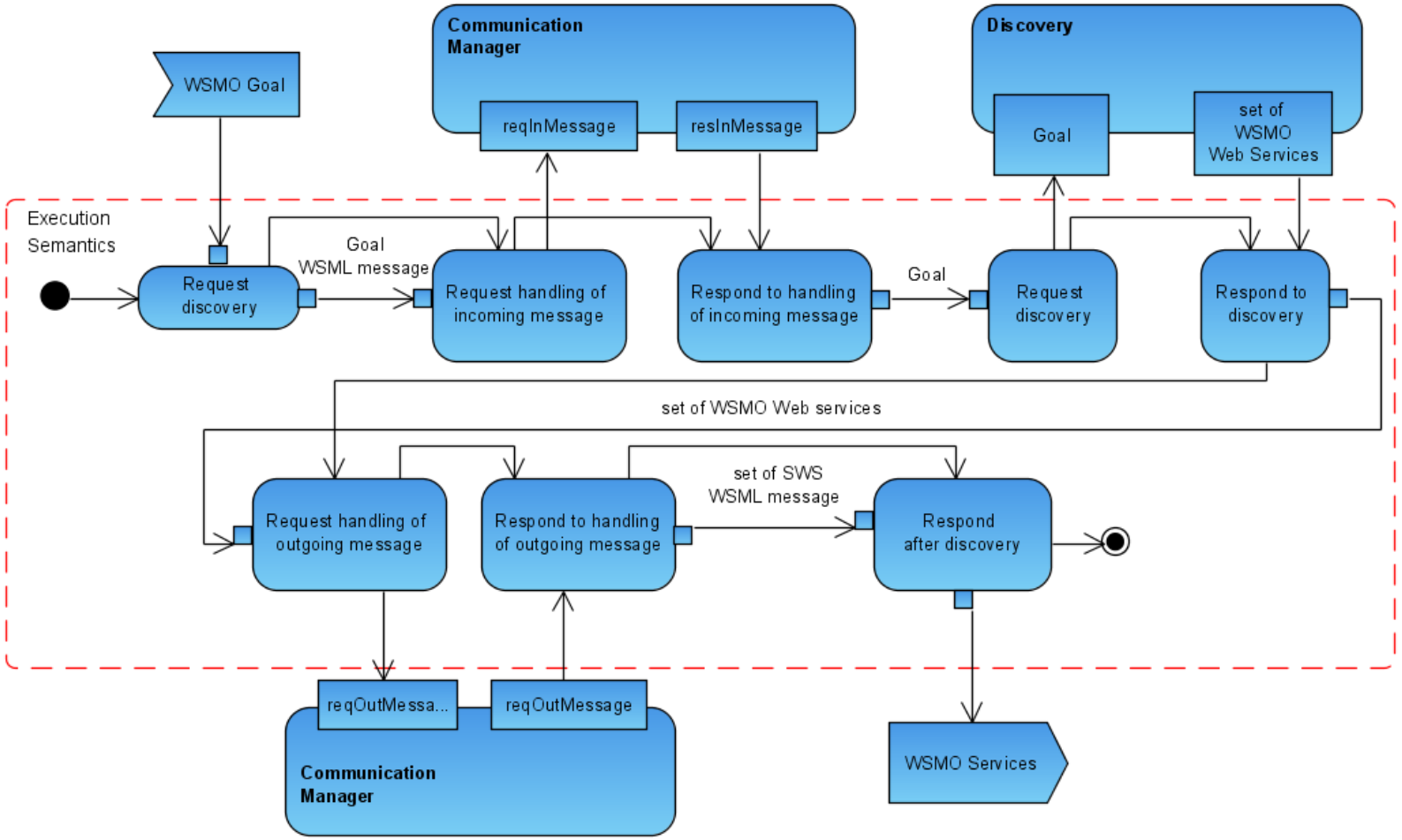
Execution Semantics

Execution Semantics

- Formal description of the operational behavior of the system in terms of computational steps
  - Greater flexibility in SESA implementations,
  - Foundations for model testing,
  - Executable representation, and
  - Improved model understanding among humans.
- Mandatory execution semantics
  - Goal-Based Web Service Discovery
  - Web Service Invocation
  - Goal-Based Service Execution

Execution Semantics

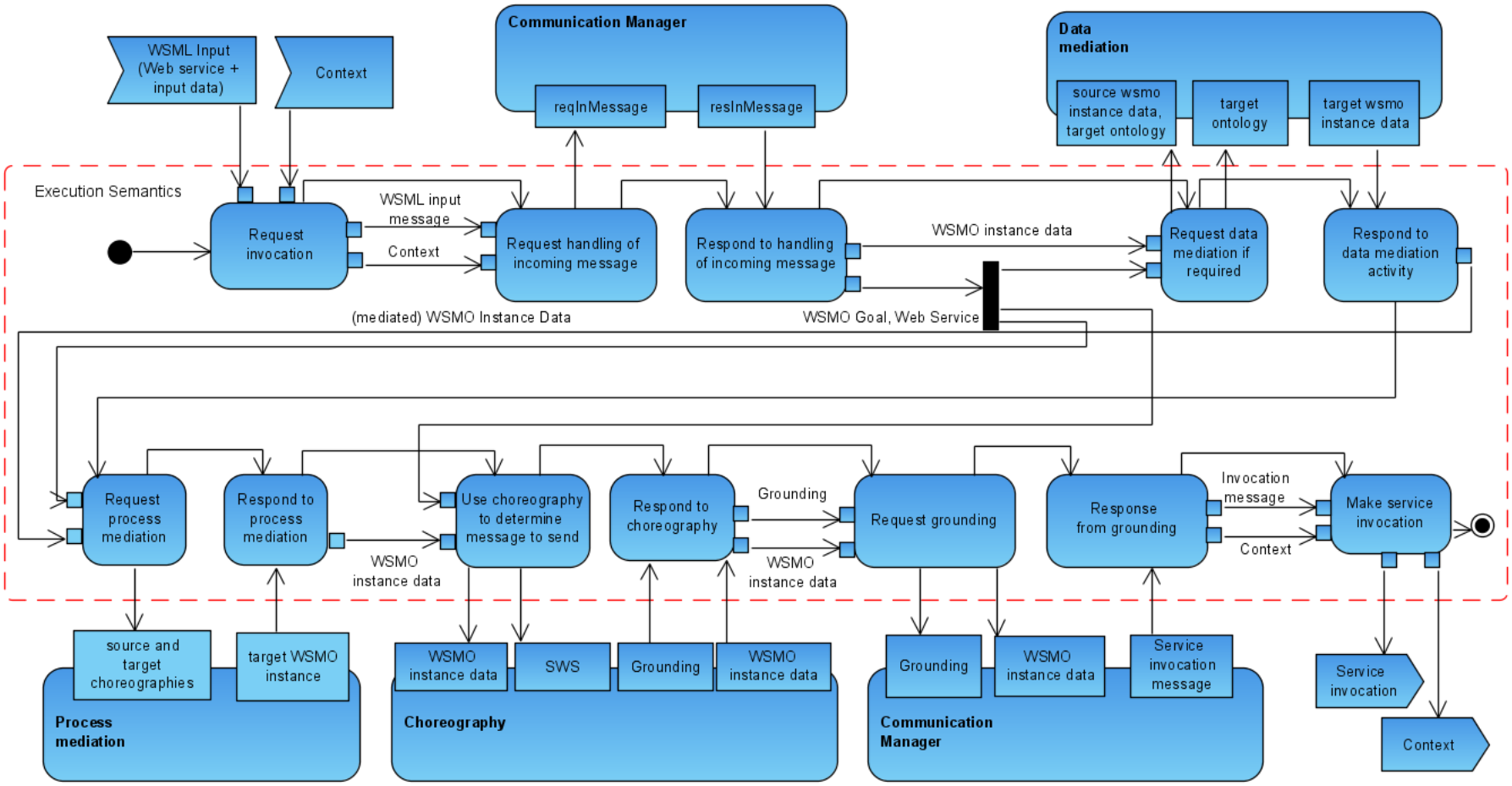
Goal-Based Web Service Discovery



Fensel, D.; Kerrigan, M.; Zaremba, M. (Eds): *Implementing Semantic Web Services: The SESA Framework. Springer 2008*.

Execution Semantics

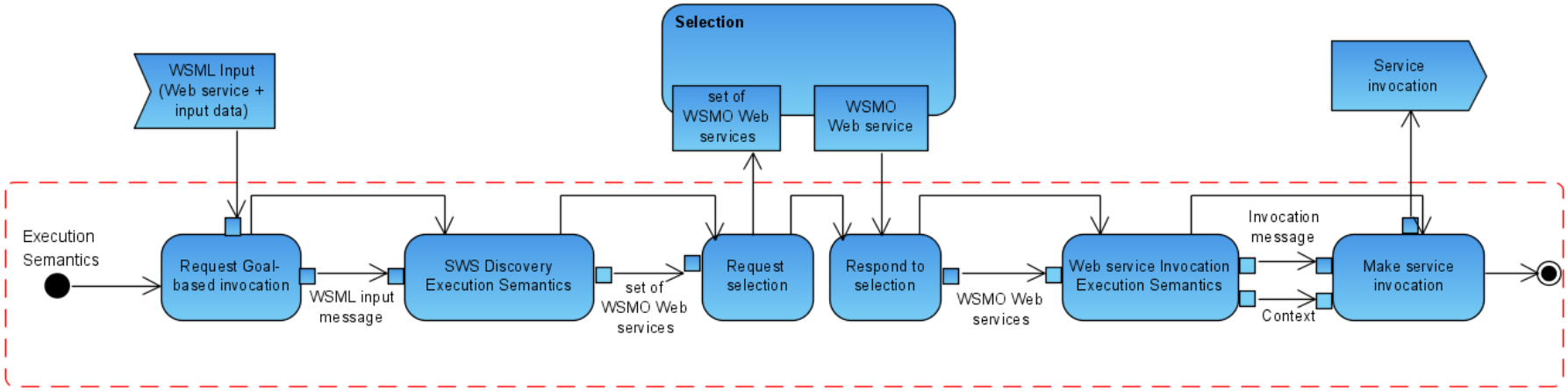
Web Service Invocation



Fensel, D.; Kerrigan, M.; Zaremba, M. (Eds): *Implementing Semantic Web Services: The SESA Framework. Springer 2008*.

Execution Semantics

Goal-Based Service Execution



Fensel, D.; Kerrigan, M.; Zaremba, M. (Eds): *Implementing Semantic Web Services: The SESA Framework. Springer 2008*.

Illustration by larger example

Illustration by larger example

Scenario description

- The goal is to discover a suitable solution for the transportation of a package with defined size and weight
- Candidate Web Services have different constraints regarding the transportation destinations, package size and weight acceptance, as well as pricing schemas
- For more information visit:
  - http://sws-challenge.org/wiki/index.php/Scenario:_Shipment_Discovery

Illustration by larger example

Goal description

I want to have my package shipped from CA, USA to Tunis, Africa size (7/6/4), weight 1 lbs, the cheaper the better.

```
wsmlVariant _"http://www.wsmo.org/wsml/wsml-syntax/wsml-flight"

goal GoalA1

capability GoalA1Capability
  postcondition
    definedBy
     ( ?x[sop#price hasValue ?price] memberOf
sop#PriceQuoteResp
        and sop#isShipped(shipmentOrderReq) ).

interface GoalA1Interface
  choreography GoalA1Choreography
    stateSignature GoalA1StateSignature

    in sop#ShipmentOrderReq
    out sop#ShipmentOrderResp

    transitionRules GoalA1TransitionRules

      forall {?request} with
        (?request memberOf sop#ShipmentOrderReq)
      do
         add(_#1 memberOf sop#ShipmentOrderResp)
      endForall
```

```
ontology GoalRequest

instance shipmentOrderReq memberOf sop#ShipmentOrderReq
  sop#from hasValue soi#MoonContactInfo
  sop#shipmentDate hasValue soi#shipmentDate1
  sop#package hasValue package
  sop#to hasValue soi#SzyslakContactInfo

instance package memberOf so#Package
  so#quantity hasValue 1
  so#length hasValue 7.0
  so#width hasValue 6.0
  so#height hasValue 4.0
  so#weight hasValue 1.0

instance shipmentDate1 memberOf so#ShipmentDate
  so#earliest hasValue "2009-01-21T13:00:00.046Z"
  so#latest hasValue "2009-01-22T13:00:00.046Z"
```

Illustration by larger example

AchieveGoal execution semantics

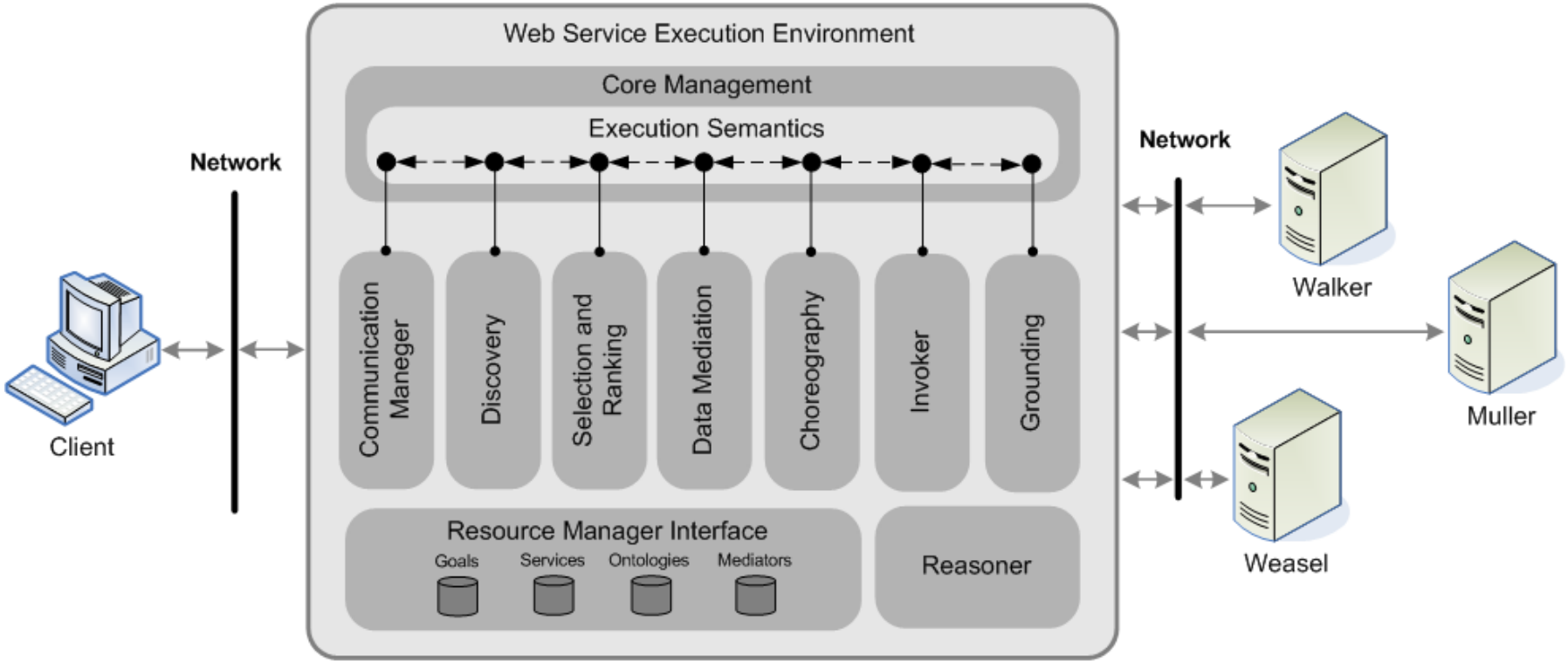


Illustration by larger example

AchieveGoal execution semantics

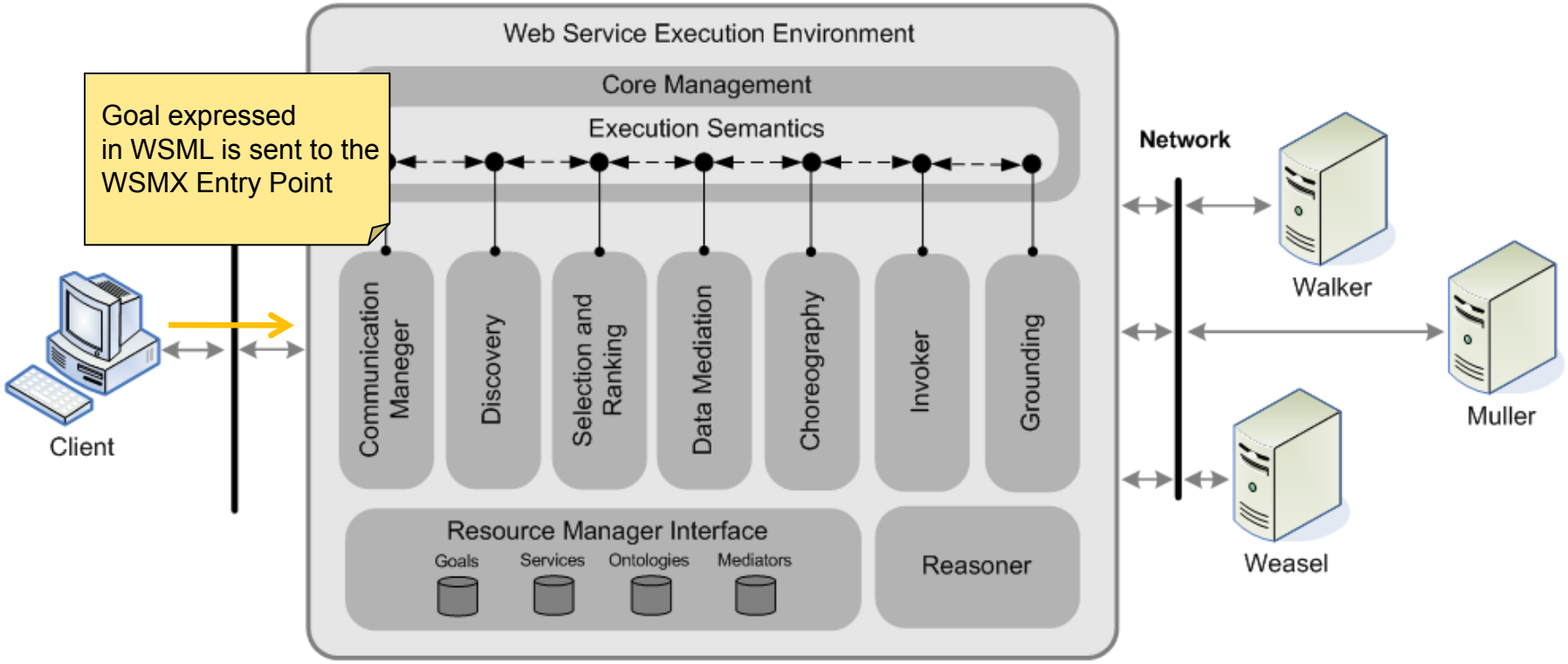


Illustration by larger example

AchieveGoal execution semantics

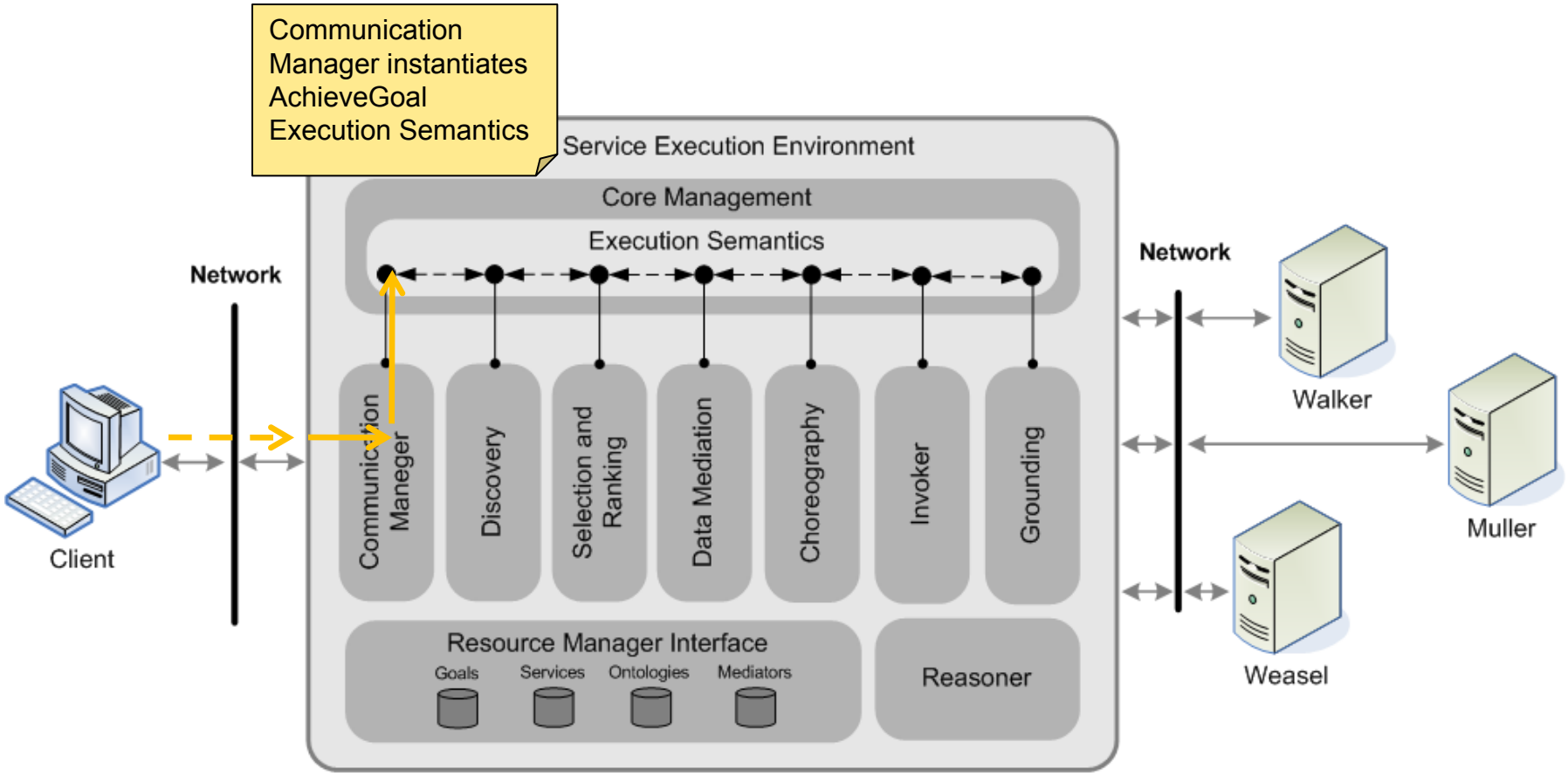


Illustration by larger example

AchieveGoal execution semantics

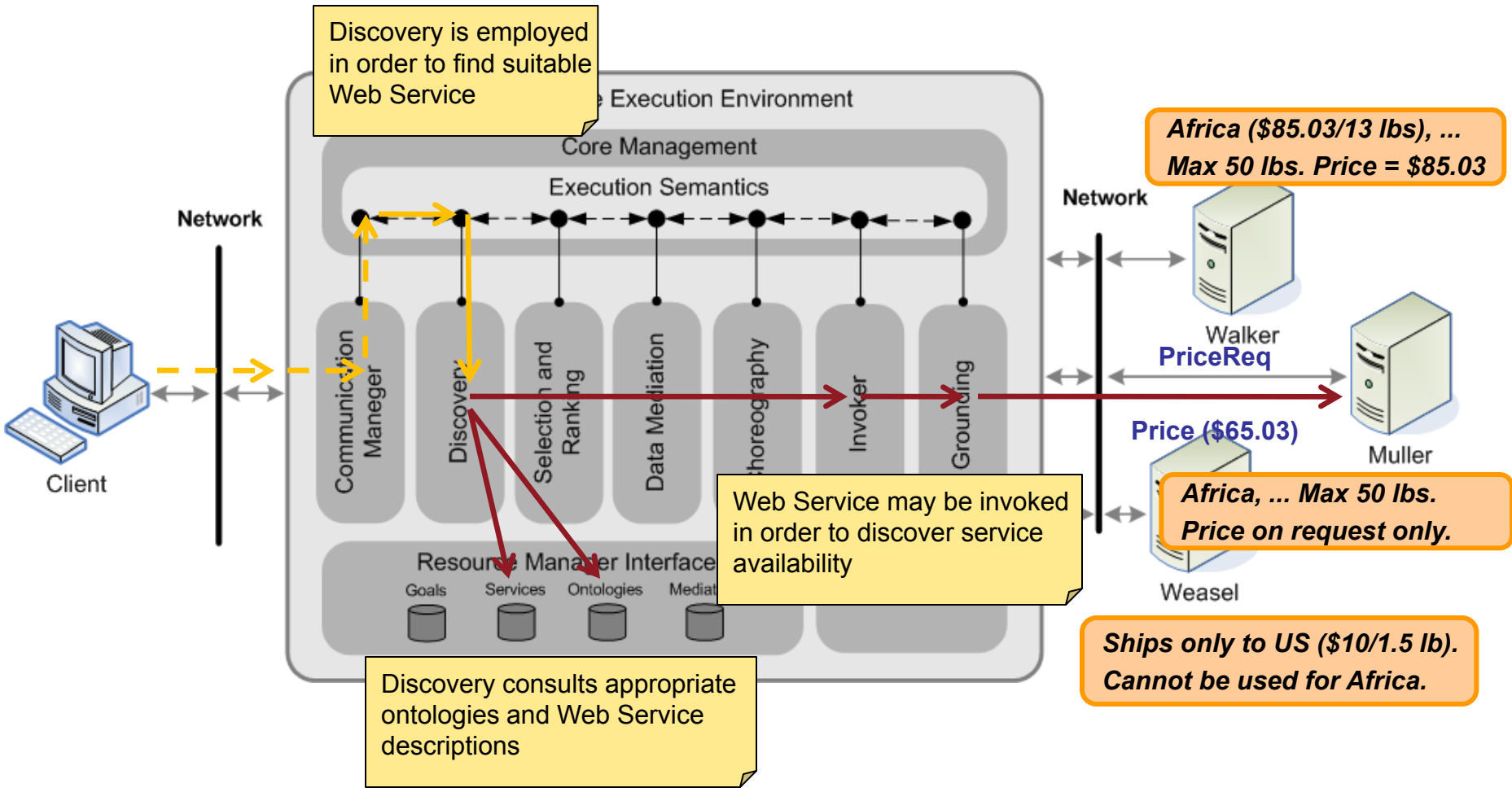


Illustration by larger example

AchieveGoal execution semantics

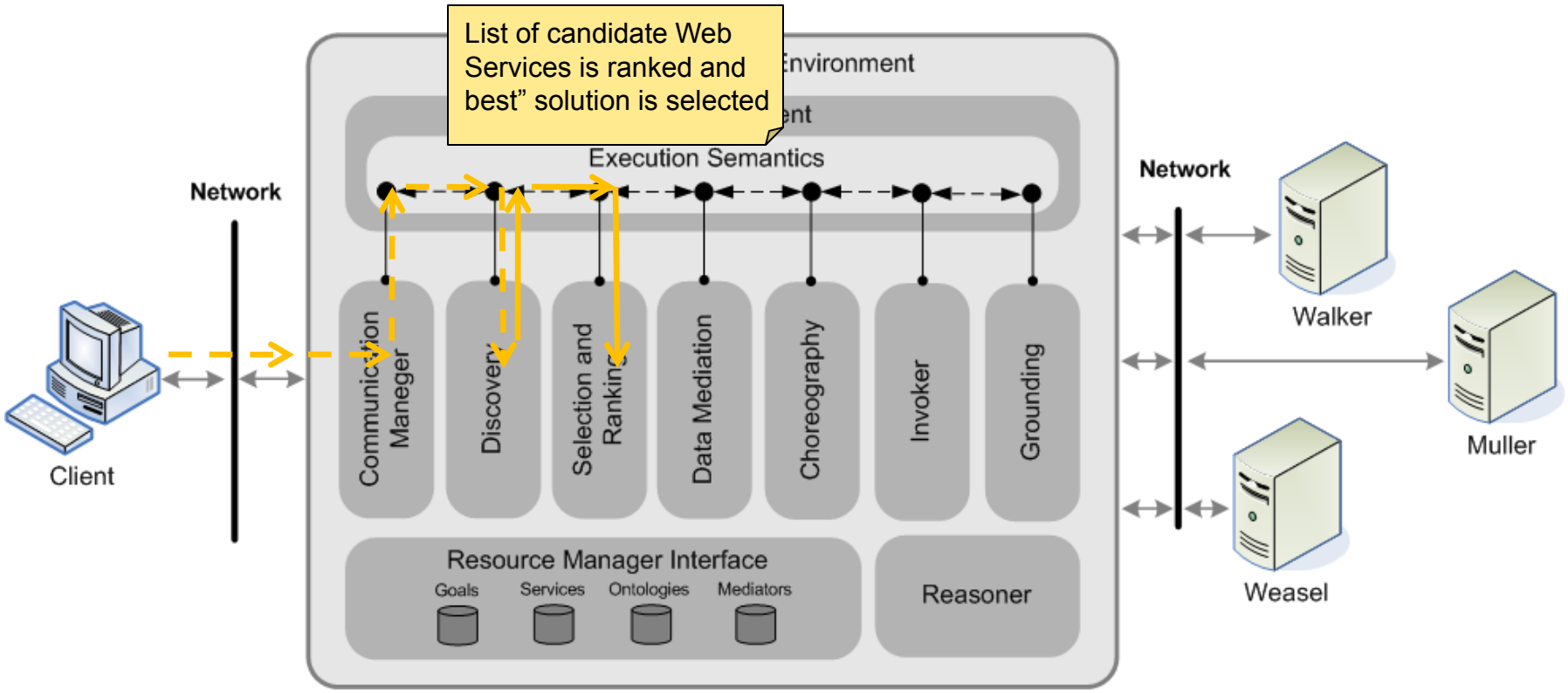


Illustration by larger example

AchieveGoal execution semantics

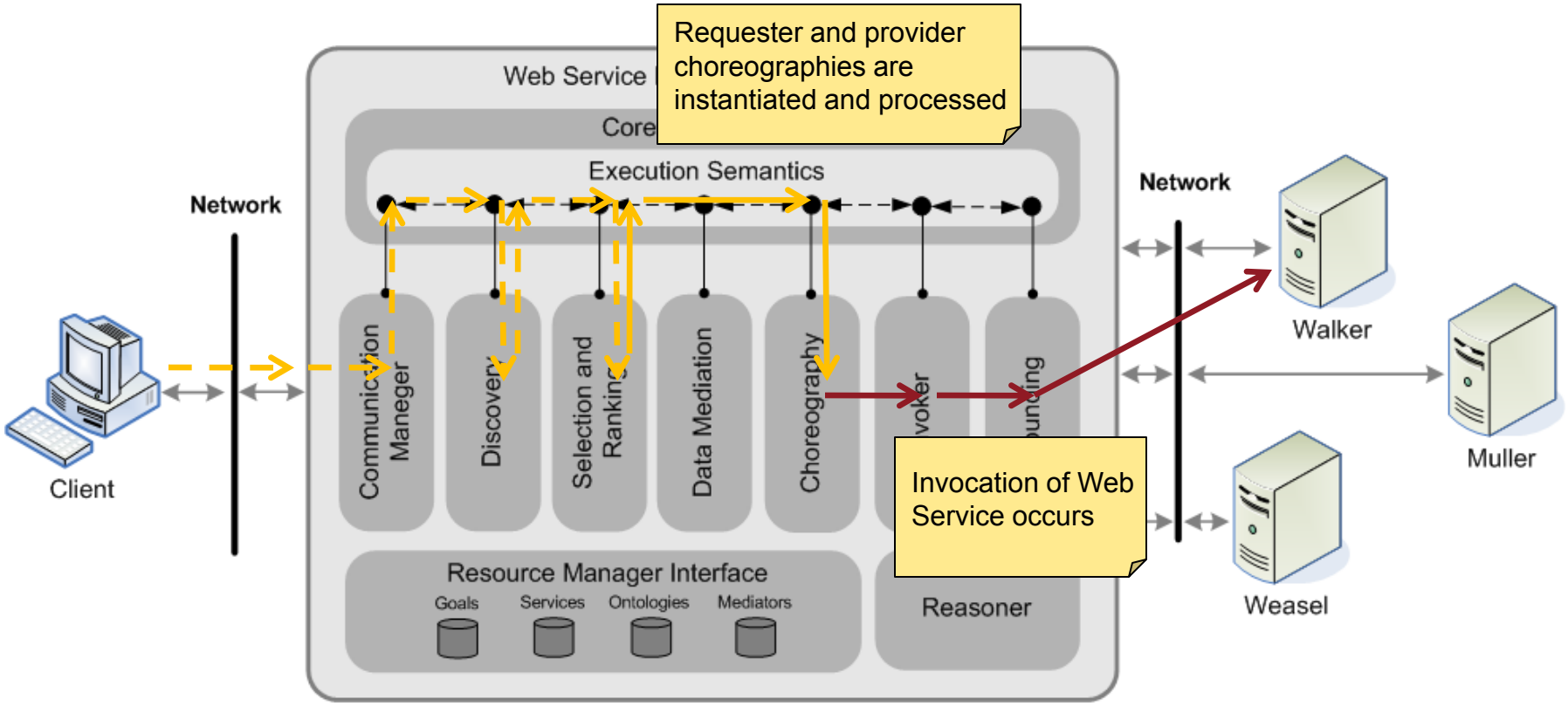


Illustration by larger example

AchieveGoal execution semantics – choreography exec

```
choreography WSMullerShipmentOrderChoreography
   stateSignature WSMullerShipmentOrderStateSignature
   …
   in sop#ShipmentOrderReq withGrounding { _"http://sws-
       challenge.org/shipper/v2/muller.wsdl#wsdl.interfaceMessageReference(muller/ShipmentOrder/in0)"}
   in so#ContactInfo
   in so#ShipmentDate
   in so#Package
   in so#Address
   out sop#ShipmentOrderResp

   transitionRules WSMullerShipmentOrderTransitionRules
   forall {?request} with
      (?request memberOf sop#ShipmentOrderReq)
   do
        add(_#1 memberOf sop#ShipmentOrderResp)
        delete(?request memberOf sop#ShipmentOrderReq)
   endForall
```

<shipmentOrderReq(soi#MoonContactInfo, soi#shipmentDate1, package, soi#SzyslakContactInfo),
package(1, 7.0, 6.0, 4.0, 1.0),
shipmentDate1(“2009-01-21T13:00:00.046Z”, "2009-01-22T13:00:00.046Z")>

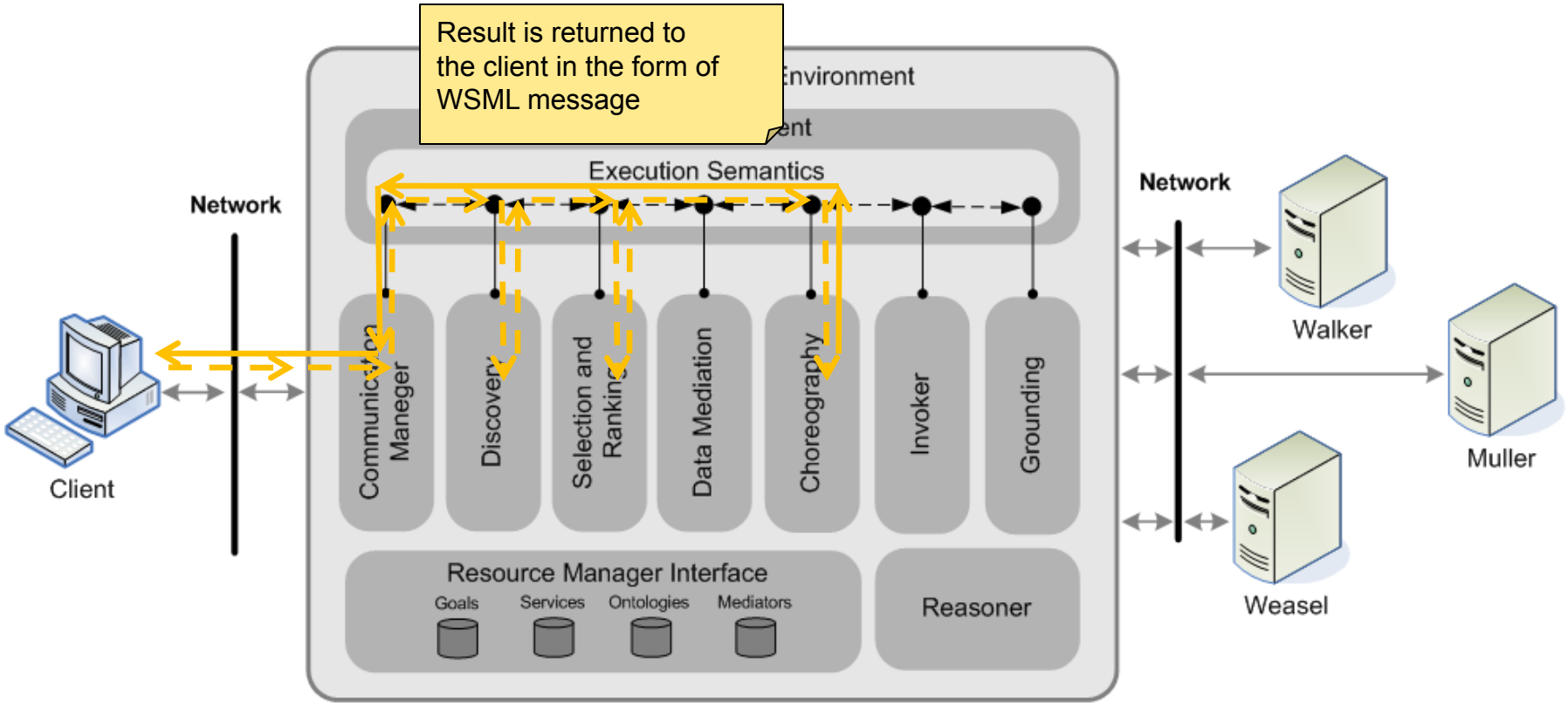
S1

<shipmentOrderResp(“2009-01-21T15:00:00.046Z”, 65.03),
package(1, 7.0, 6.0, 4.0, 1.0),
shipmentDate1(“2009-01-21T13:00:00.046Z”, "2009-01-22T13:00:00.046Z")>

S2

Illustration by larger example

AchieveGoal execution semantics



Questions