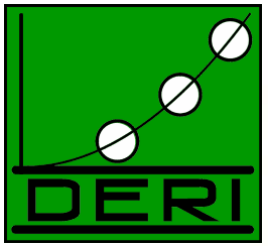


Semantic Web Services Tutorial

Michael Stollberg[1], Matt Moran[2],
John Domingue[3], and Liliana Cabral[3]

5th International Conference on Web Engineering (ICWE 2005)
Sydney, Australia, 2005 July 25

[1]DERI, Innsbruck, Austria
[2]DERI, Galway, Ireland
[3]Knowledge Media Institute, The Open University, UK

Agenda

Part I: Introduction to Semantic Web Services
- Vision of Next Generation Web Technology
- Semantic Web Service Challenges

Part II: The Web Service Modeling Ontology WSMO
- Aims & Design Principles
- Top Level Element Definitions

BREAK

Part III: A Walkthru Example
- Virtual Travel Agency Example
- Roles, Elements, Semantic Web Service technology usage

LUNCH

Part IV: The Web Service Execution Environment WSMX
- Aims & Design Principles
- Architecture & Components

BREAK

Part V: Hands-On Session with IRS III
- IRS III introduction
- Hands-on Session (explanation, hands-on)

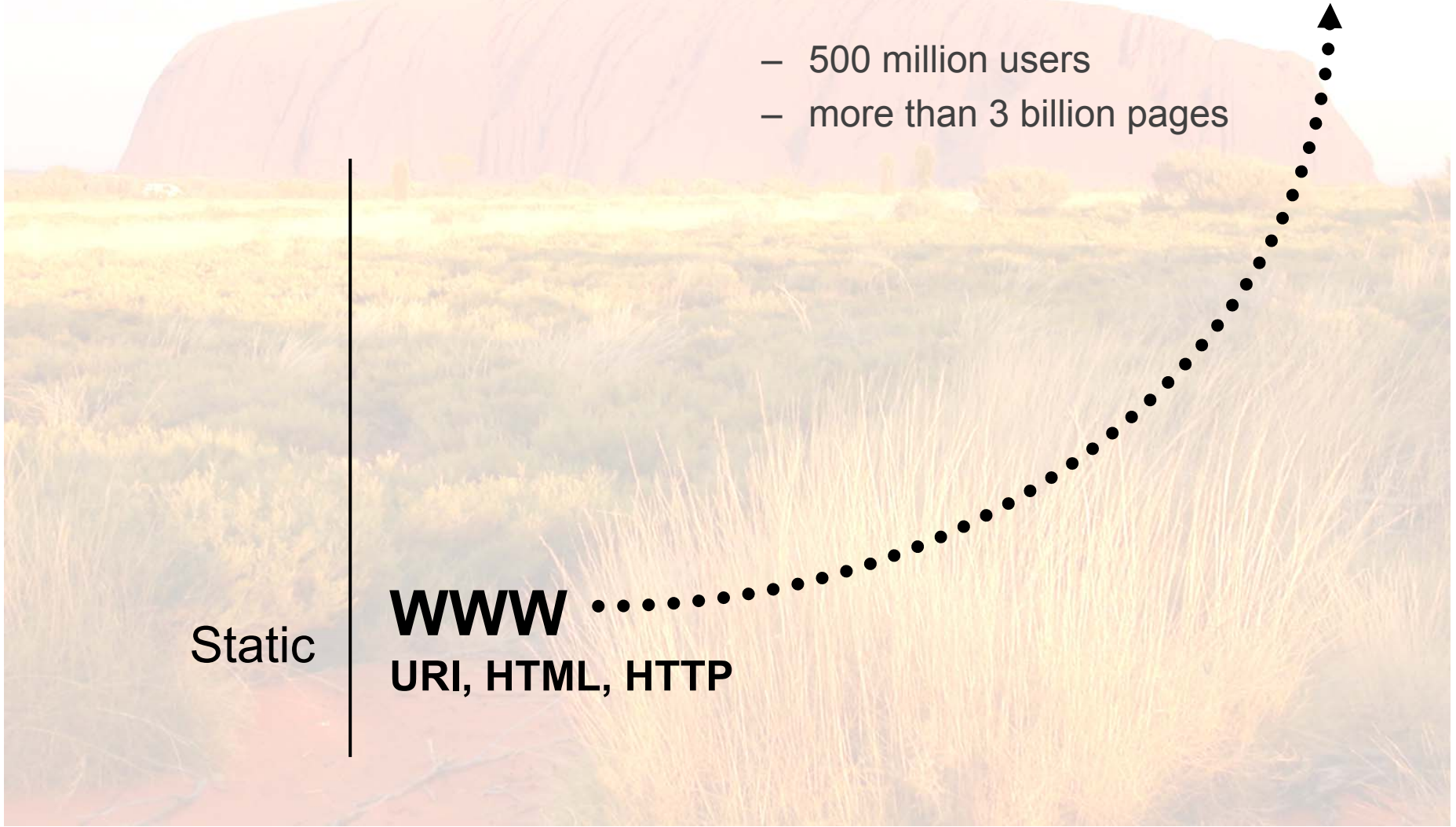
PART I:

Introduction to Semantic Web Services

- The vision of the Semantic Web
- Ontologies as the basic building block
- Current Web Service Technologies
- Vision and Challenges for Semantic Web Services

The Vision

- 500 million users
- more than 3 billion pages

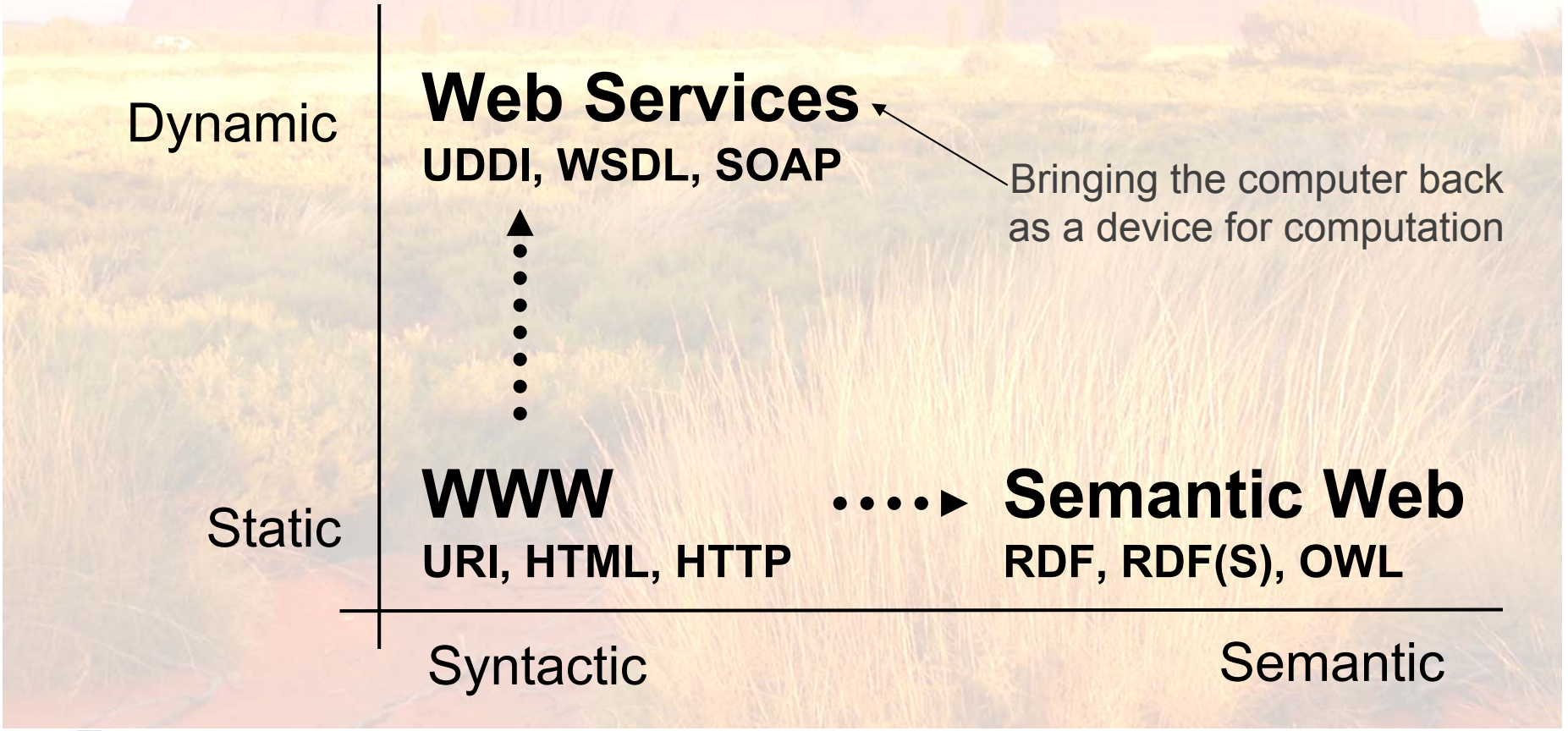


The Vision

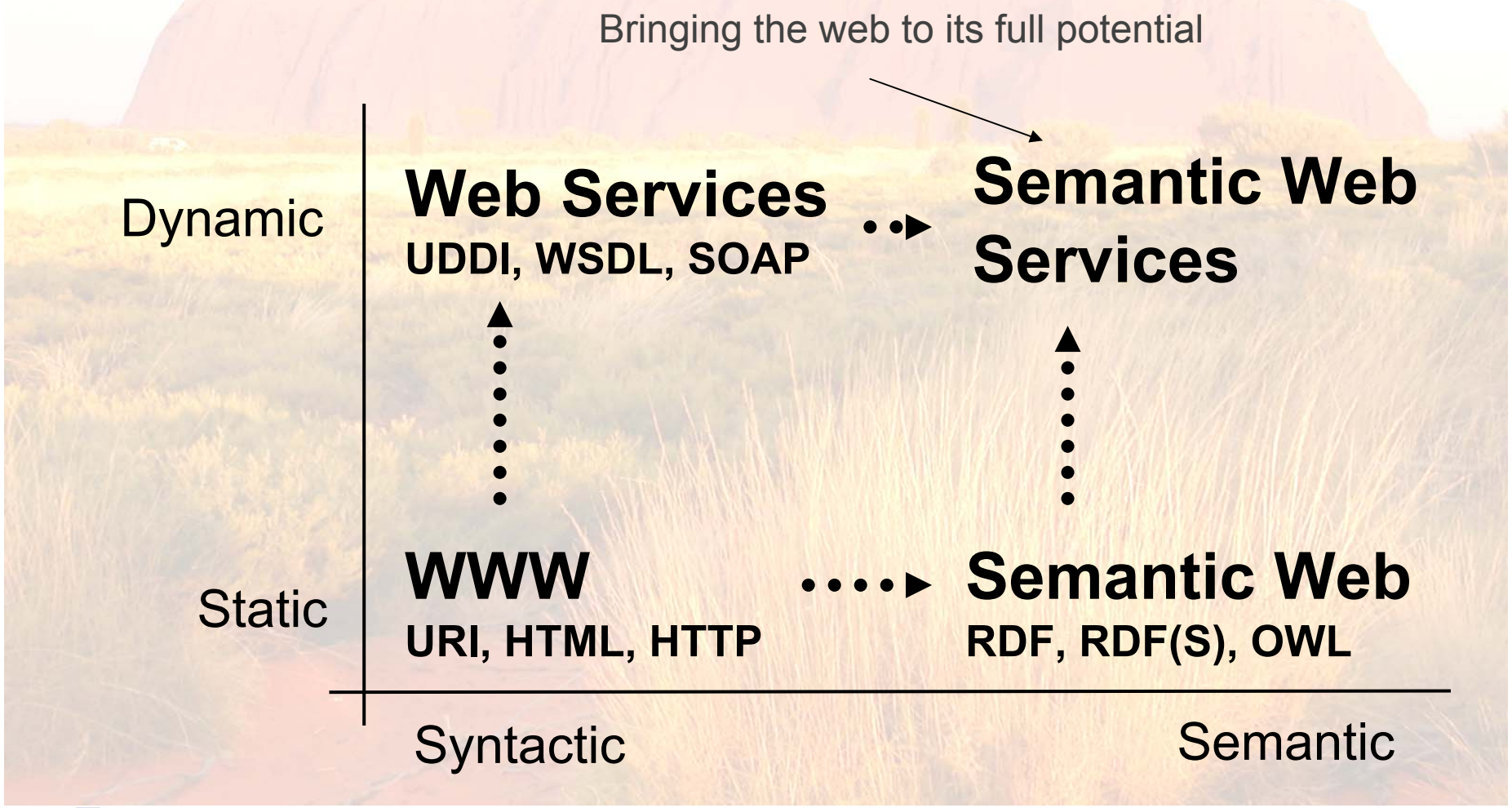
Serious Problems in

- information finding,
- information extracting,
- information representing,
- information interpreting and
- and information maintaining.

The Vision


Dynamic
Static
Web Services
UDDI, WSDL, SOAP
Bringing the computer back as a device for computation
WWW
URI, HTML, HTTP
Semantic Web
RDF, RDF(S), OWL
Syntactic
Semantic

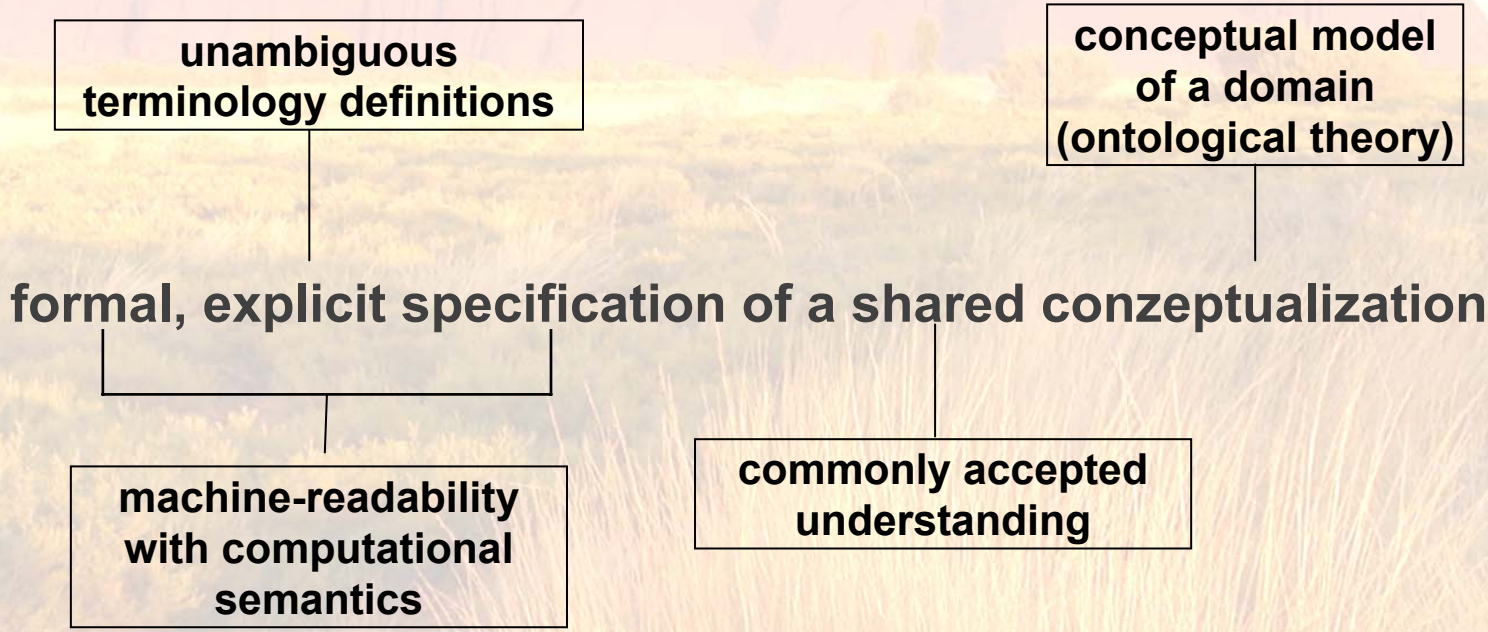
The Vision


Bringing the web to its full potential
Dynamic
Static
Web Services
UDDI, WSDL, SOAP
Semantic Web Services
WWW
URI, HTML, HTTP
Semantic Web
RDF, RDF(S), OWL
Syntactic
Semantic

The Semantic Web

- the next generation of the WWW
- information has machine-processable and machine-understandable semantics
- not a separate Web but an augmentation of the current one
- Ontologies as basic building block

Ontology Definition



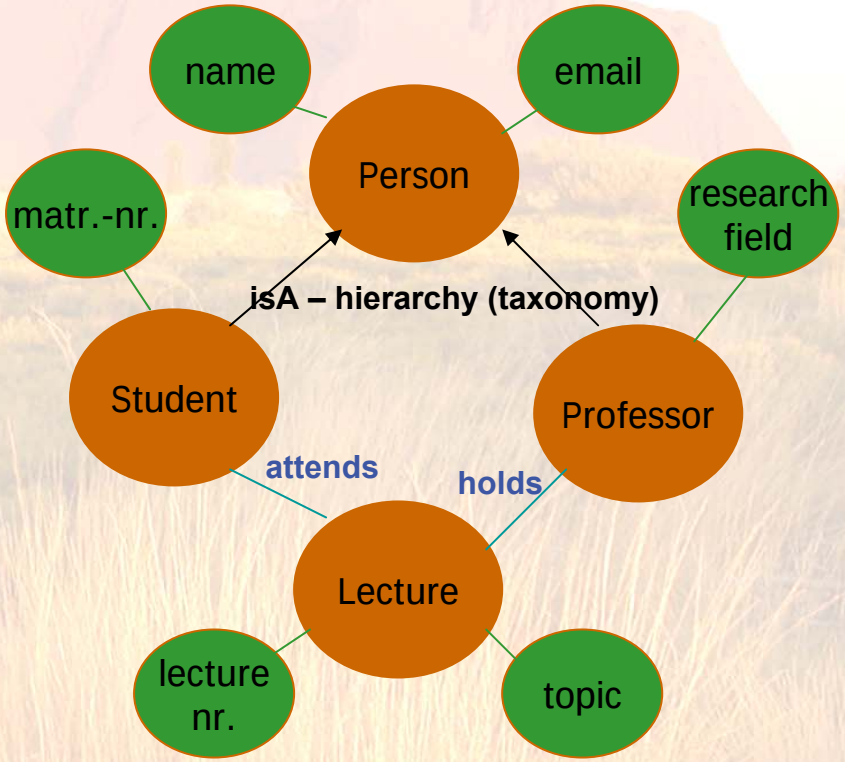
Ontology Example

Concept
conceptual entity of the domain

Property
attribte describing a concept

Relation
relationship between concepts or properties

Axiom
coherency description between Concepts / Properties / Relations via logical expressions



holds(Professor, Lecture) =>
Lecture.topic = Professor.researchField

Ontology Technology

To make the Semantic Web working we need:

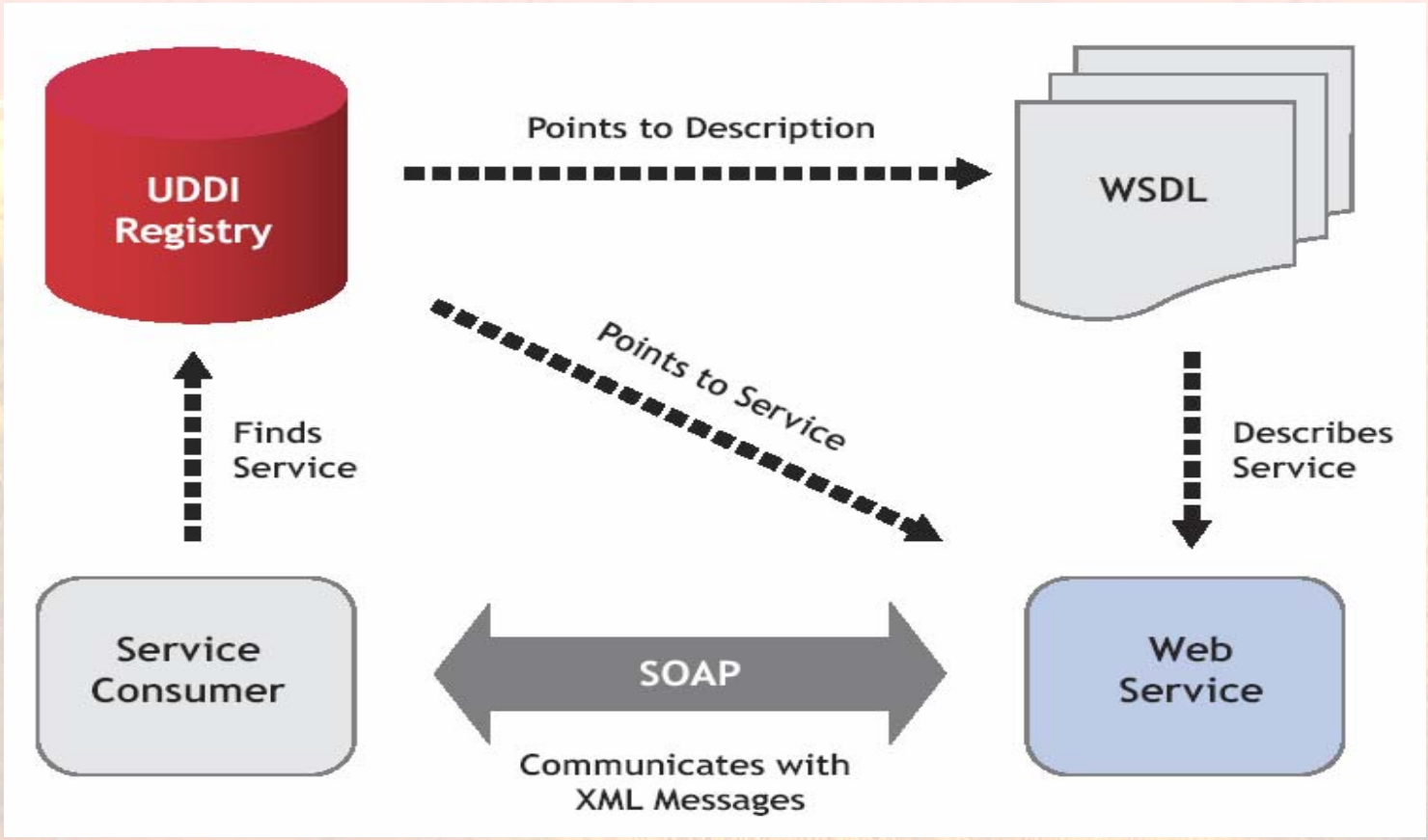
- **Ontology Languages**:
  - expressivity
  - reasoning support
  - web compliance
- **Ontology Reasoning**:
  - large scale knowledge handling
  - fault-tolerant
  - stable & scalable inference machines
- **Ontology Management Techniques**:
  - editing and browsing
  - storage and retrieval
  - versioning and evolution Support
- **Ontology Integration Techniques**:
  - ontology mapping, alignment, merging
  - semantic interoperability determination
- and … **Applications**

Web Services

- loosely coupled, reusable components
- encapsulate discrete functionality
- distributed
- programmatically accessible over standard internet protocols
- add new level of functionality on top of the current web

The Promise of Web Services

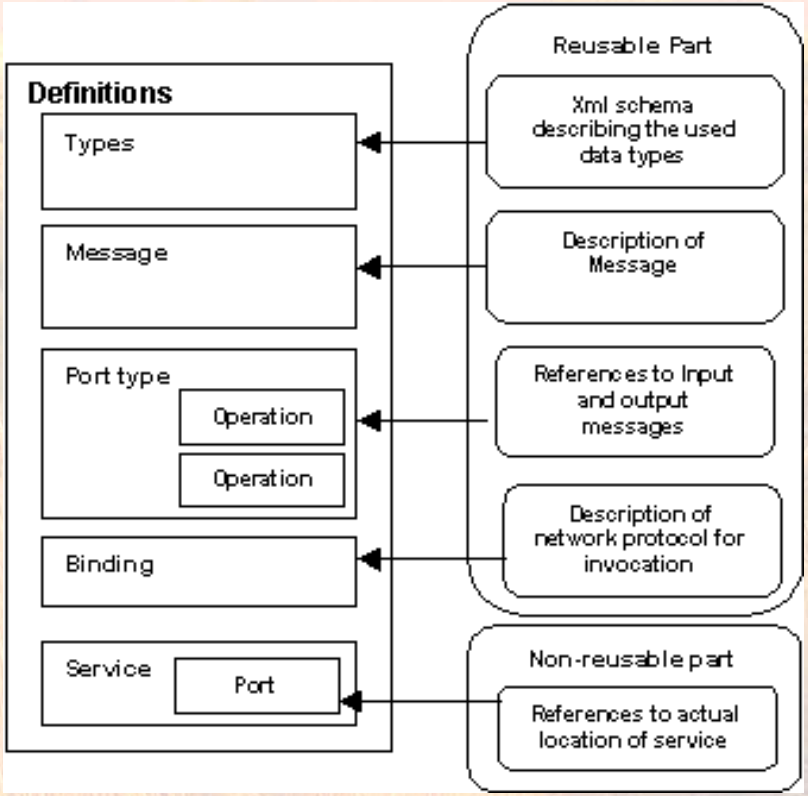
web-based SOA as new system design paradigm



WSDL

- Web Service Description Language
- W3C effort, WSDL 2 final construction phase

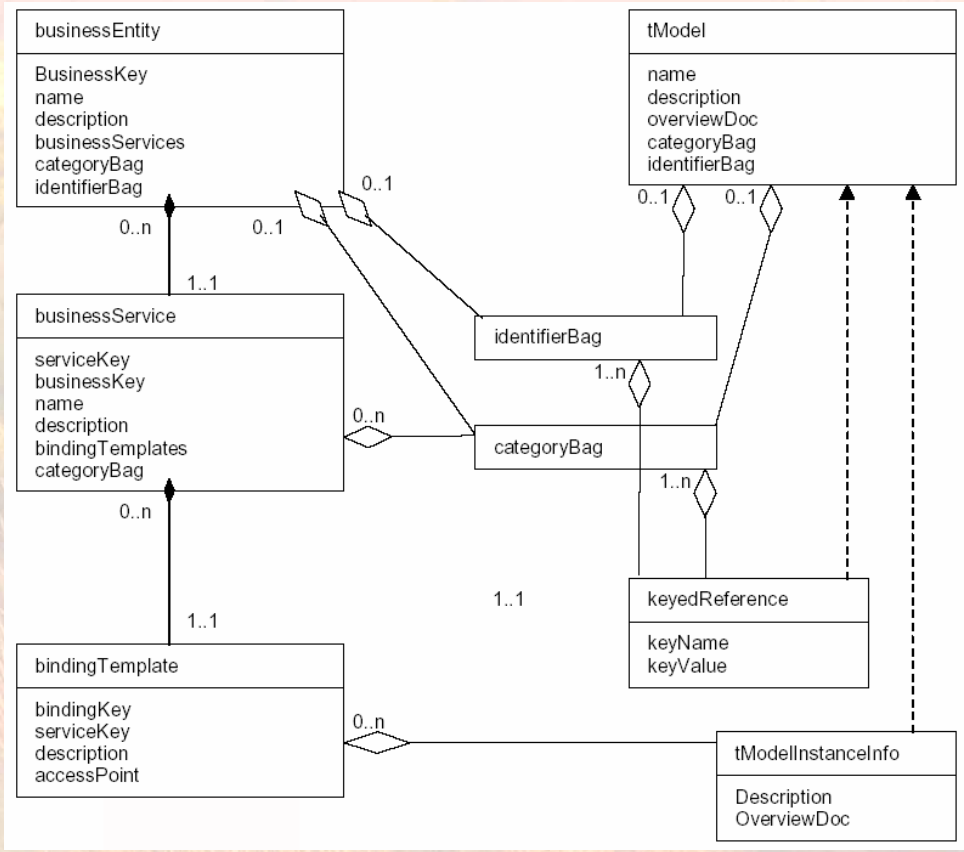
describes interface for consuming a Web Service:
- Interface: operations (in- & output)
- Access (protocol binding)
- Endpoint (location of service)



UDDI

- Universal Description, Discovery, and Integration Protocol
- OASIS driven standardization effort

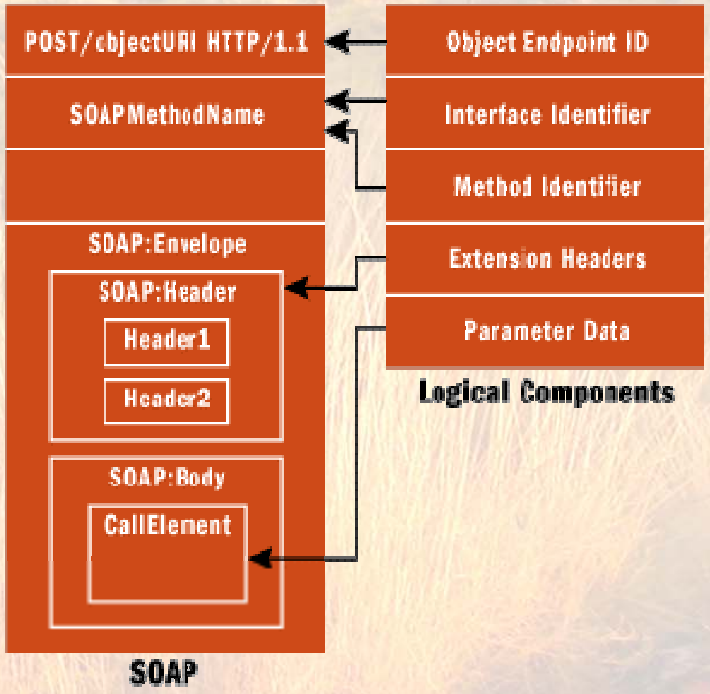
Registry for
Web Services:
- provider
- service information
- technical access



SOAP

- Simple Object Access Protocol
- W3C Recommendation

XML data transport:
- sender / receiver
- protocol binding
- communication aspects
- content



Deficiencies of WS Technology

- current technologies allow usage of Web Services
- but:
  - only syntactical information descriptions
  - syntactic support for discovery, composition and execution

  => Web Service usability, usage, and integration needs to be inspected manually
  - no semantically marked up content / services
  - no support for the Semantic Web

=> current Web Service Technology Stack failed to realize the promise of Web Services

Semantic Web Services

Semantic Web Technology

- allow machine supported data interpretation
- ontologies as data model

+

Web Service Technology

automated discovery, selection, composition,
and web-based execution of services

=> Semantic Web Services as integrated solution for realizing the vision of the next generation of the Web

Semantic Web Services

- define exhaustive description frameworks for describing Web Services and related aspects **(Web Service Description Ontologies)**
- support ontologies as underlying data model to allow machine supported data interpretation **(Semantic Web aspect)**
- define semantically driven technologies for automation of the Web Service usage process **(Web Service aspect)**

Semantic Web Services

Usage Process:

- Publication: Make the description of a Web service available on the Web
- Discovery: Detect suitable services for a solving given task
- Selection: Choose the most appropriate services among the usable ones
- Composition: Combine services to achieve a goal
- Mediation: Solve mismatches (data, protocol, process) among the elements that shall interoperate
- Execution: Invoke services according to consumption interface and programmatic conventions

Semantic Web Services

Execution support:

- Monitoring: Control the execution process
- Compensation: Provide transactional support and undo or mitigate unwanted effects
- Replacement: Facilitate the substitution of services by equivalent ones
- Auditing: Verify that service execution occurred in the expected way

PART II:

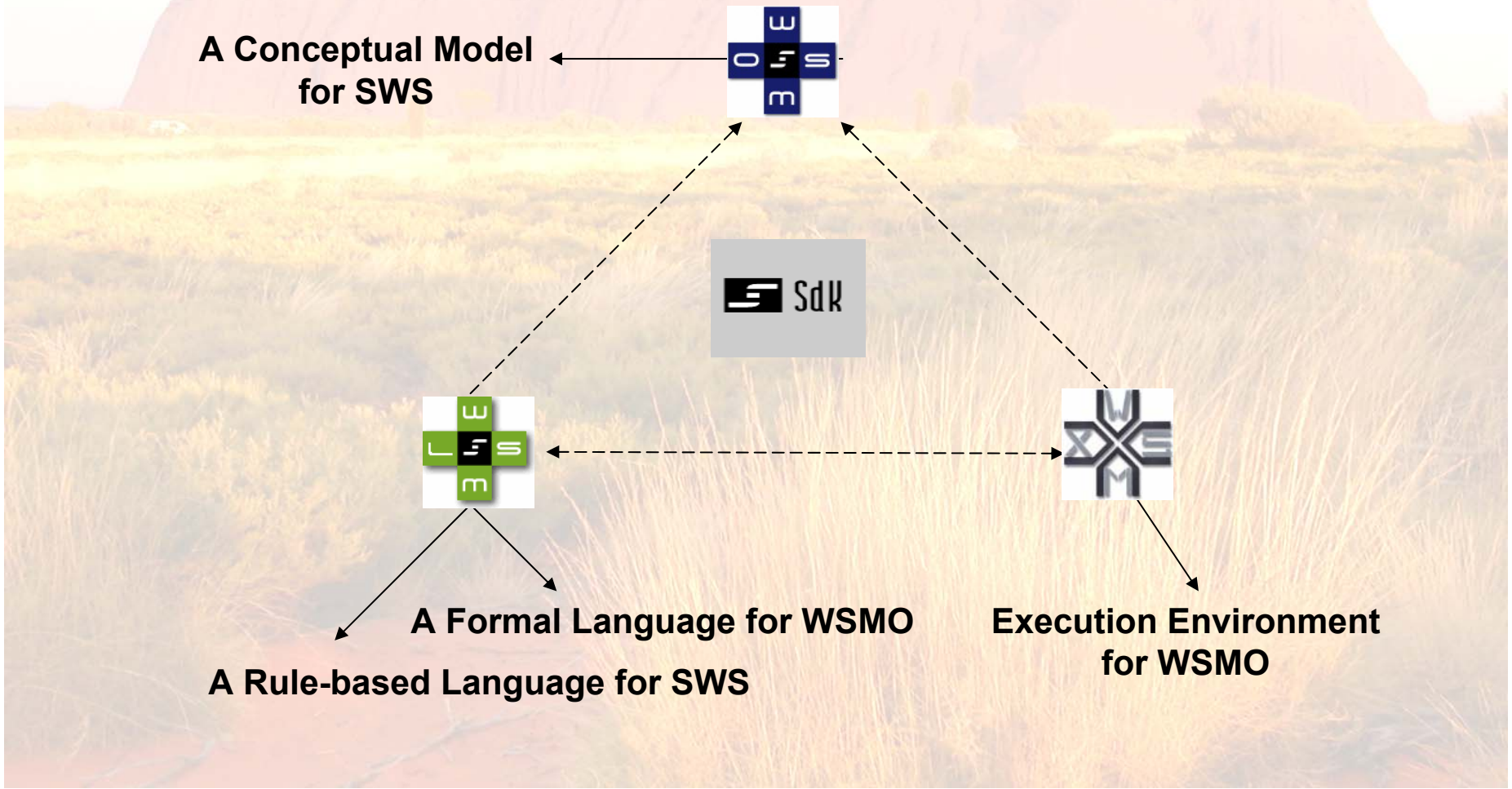
The Web Service Modeling Ontology WSMO

- Aims & Working Groups
- Design Principles
- Top Level Notions
  - Ontologies
  - Web Services
  - Goals
  - Mediators
- Comparison to OWL-S

WSMO is ..

- a conceptual model for Semantic Web Services:
  - ontology of core elements for Semantic Web Services
  - a formal description language (WSML)
  - execution environment (WSMX)
- derived from and based on the Web Service Modeling Framework WSMF
- a SDK-Cluster Working Group
  (joint European research and development initiative)

WSMO Working Groups

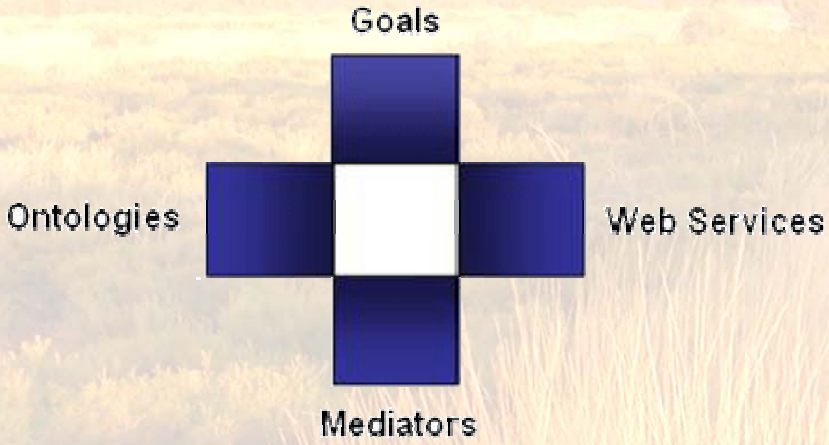


WSMO Design Principles

- Web Compliance
- Ontology-Based
- Goal-driven
- Strict Decoupling
- Centrality of Mediation
- Description versus Implementation
- Execution Semantics

WSMO Top Level Notions

Objectives that a client wants to achieve by using Web Services



Provide the formally specified terminology of the information used by all other components

Semantic description of Web Services:
- **Capability** *(functional)*
- **Interfaces** *(usage)*

Connectors between components with mediation facilities for handling heterogeneities

WSMO D2, version 1.2, 13 April 2005 (W3C submission)

Non-Functional Properties

every WSMO elements is described by properties that contain relevant, non-functional aspects

- Dublin Core Metadata Set:
  - complete item description
  - used for resource management
- Versioning Information
  - evolution support
- Quality of Service Information
  - availability, stability
- Other
  - Owner, financial, etc.

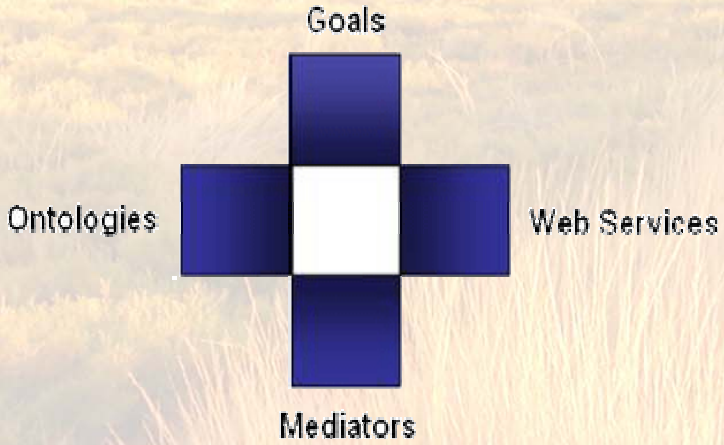
Non-Functional Properties List

- Dublin Core Metadata
  - Contributor
  - Coverage
  - Creator
  - Description
  - Format
  - Identifier
  - Language
  - Publisher
  - Relation
  - Rights
  - Source
  - Subject
  - Title
  - Type
- Quality of Service
  - Accuracy
  - NetworkRelatedQoS
  - Performance
  - Reliability
  - Robustness
  - Scalability
  - Security
  - Transactional
  - Trust
- Other
  - Financial
  - Owner
  - TypeOfMatch
  - Version

WSMO Ontologies

Objectives that a client wants to achieve by using Web Services

Provide the formally specified terminology of the information used by all other components



Semantic description of Web Services:
- **Capability** *(functional)*
- **Interfaces** *(usage)*

Connectors between components with mediation facilities for handling heterogeneities

Ontology Usage & Principles

- **Ontologies are used as the 'data model' throughout WSMO**
  - all WSMO element descriptions rely on ontologies
  - all data interchanged in Web Service usage are ontologies
  - Semantic information processing & ontology reasoning
- **WSMO Ontology Language WSML**
  - conceptual syntax for describing WSMO elements
  - logical language for axiomatic expressions (WSML Layering)
- **WSMO Ontology Design**
  - Modularization: import / re-using ontologies, modular approach for ontology design
  - De-Coupling: heterogeneity handled by **OO Mediators**

Ontology Specification

- **Non functional properties** (see before)
- **Imported Ontologies** importing existing ontologies where no heterogeneities arise
- **Used mediators** OO Mediators (ontology import with terminology mismatch handling)

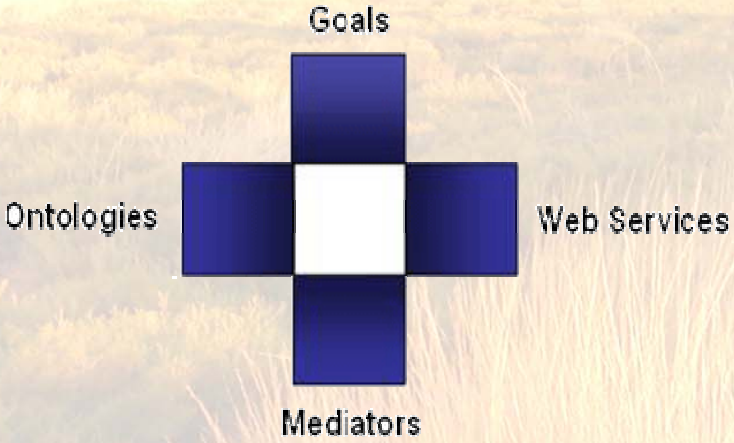
Ontology Elements:

Concepts	set of concepts that belong to the ontology, incl.
Attributes	set of attributes that belong to a concept
Relations	define interrelations between several concepts
Functions	special type of relation (unary range = return value)
Instances	set of instances that belong to the represented ontology
Axioms	axiomatic expressions in ontology (logical statement)

WSMO Web Services

Objectives that a client wants to achieve by using Web Services

Provide the formally specified terminology of the information used by all other components



Semantic description of Web Services:
- **Capability** *(functional)*
- **Interfaces** *(usage)*

Connectors between components with mediation facilities for handling heterogeneities

WSMO Web Service Description

- complete item description
- quality aspects
- Web Service Management

Non-functional Properties

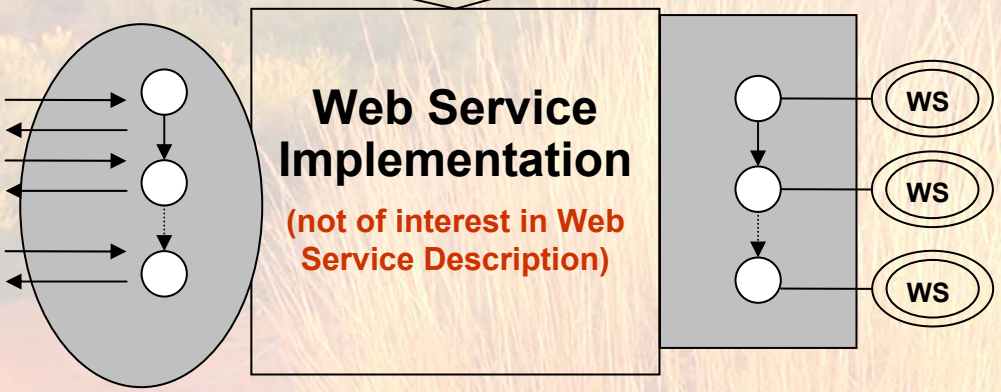
DC + QoS + Version + financial

- Advertising of Web Service
- Support for WS Discovery

Capability

functional description

client-service interaction interface for consuming WS
- External Visible Behavior
- Communication Structure
- 'Grounding'



realization of functionality by aggregating other Web Services
- functional decomposition
- interaction with aggregated WS

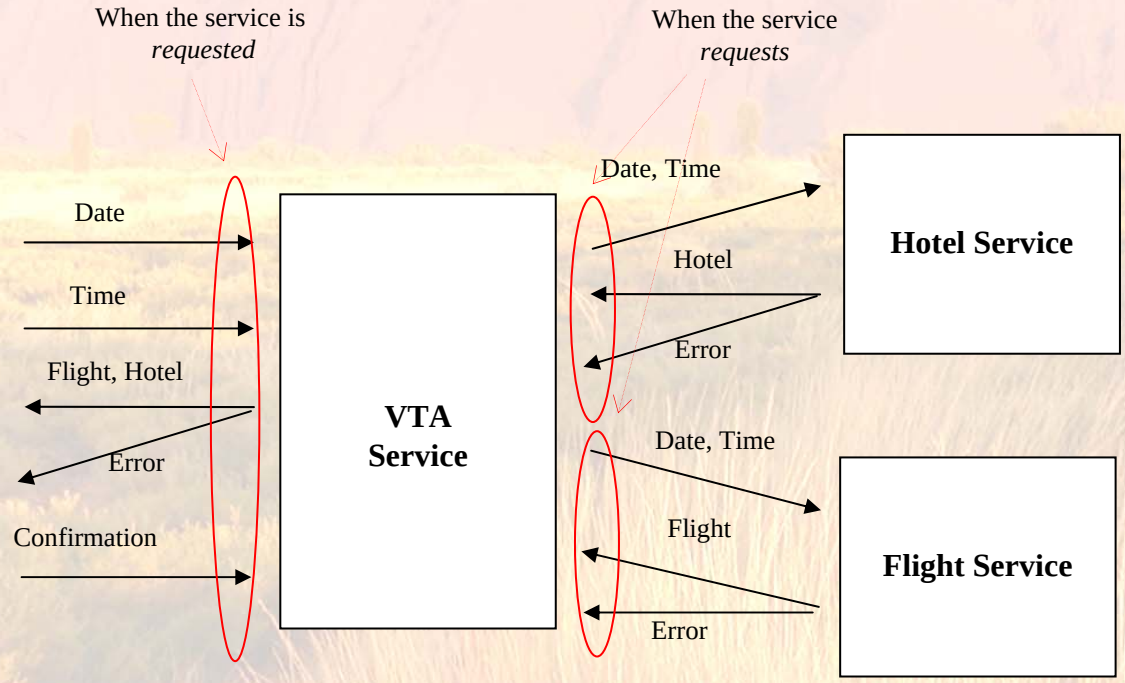
Choreography --- Service Interfaces --- **Orchestration**

Capability Specification

- **Non functional properties**
- **Imported Ontologies**
- **Used mediators**
  - *OO Mediator:* importing ontologies with mismatch resolution
  - *WG Mediator:* link to a Goal wherefore service is not usable a priori
- **Pre-conditions**
  what a web service expects in order to be able to provide its service (conditions over the input)
- **Assumptions**
  conditions on the state of the world that has to hold before the Web Service can be executed
- **Post-conditions**
  describes the result of the Web Service in relation to the input, and conditions on it
- **Effects**
  conditions on the state of the world that hold after execution of the Web Service (i.e. changes in the state of the world)

Choreography & Orchestration

- VTA example:



- **Choreography** = how to interact with the service to consume its functionality
- **Orchestration** = how service functionality is achieved by aggregating other Web Services

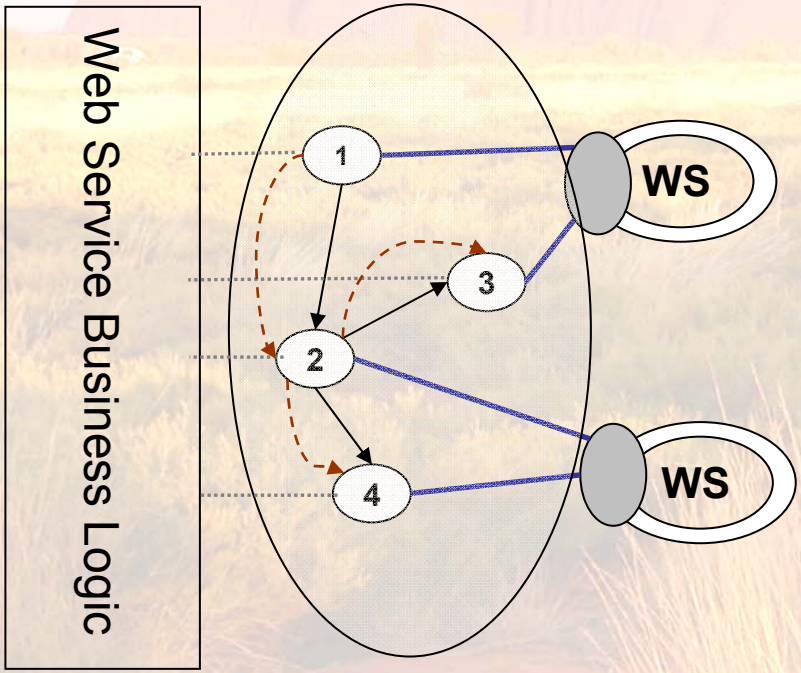
Choreography Aspects

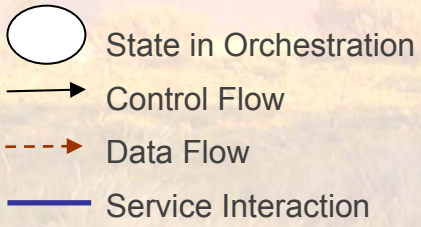
Interface for consuming Web Service

- **External Visible Behavior**
  - those aspects of the workflow of a Web Service where Interaction is required
  - described by workflow constructs: sequence, split, loop, parallel
- **Communication Structure**
  - messages sent and received
  - their order (communicative behavior for service consumption)
- **Grounding**
  - executable communication technology for interaction
  - choreography related errors (e.g. input wrong, message timeout, etc.)
- **Formal Model**
  - reasoning on Web Service interfaces (conversation validation)
  - allow mediation support on Web Service interfaces

Orchestration Aspects

Control Structure for aggregation of other Web Services





- decomposition of service functionality
- interaction with aggregated Web Services
- all service interaction via choreographies

WSMO Web Service Interfaces

- service interfaces are concerned with service consumption and interaction
- Choreography and Orchestration as sub-elements of Web Service Interface
- common requirements for service interface description:
  1. represent the dynamics of information interchange during service consumption and interaction
  2. support ontologies as the underlying data model
  3. appropriate communication technology for information interchange
  4. sound formal model / semantics of service interface specifications in order to allow operations on them.

Service Interface Description Approach

- **Ontologies as data model:**
  - all data elements interchanged are ontology instances
  - service interface = evolving ontology
- **Abstract State Machines (ASM) as formal framework:**
  - dynamics representation: high expressiveness & low ontological commitment
  - core principles: state-based, state definition by formal algebra, guarded transitions for state changes
  - overcome the “Frame Problem”
- **further characteristics:**
  - not restricted to any specific communication technology
  - ontology reasoning for service interoperability determination
  - basis for declarative mediation techniques on service interfaces

Service Interface Description Model

- Vocabulary Ω:
  - ontology schema(s) used in service interface description
  - usage for information interchange: in, out, shared, controlled
- States ω(Ω):
  - a stable status in the information space
  - defined by attribute values of ontology instances
- Guarded Transition GT(ω):
  - state transition
  - general structure: ***if*** (condition) ***then*** (action)
  - different for Choreography and Orchestration
  - additional constructs: *add*, *delete*, *update*

Service Interface Example

Communication Behavior of a Web Service

Ω_{in} **hasValues** {
concept A [
att1 **ofType** X
att2 **ofType** Y]
…}

Ω_{out} **hasValues** {
concept B [
att1 **ofType** W
att2 **ofType** Z]
…}

Vocabulary:
- Concept A in Ω_{in}
- Concept B in Ω_{out}

State ω_1

a **memberOf** A [
att1 **hasValue** x
att2 **hasValue** y]

Guarded Transition $GT(\omega_1)$

IF (a **memberOf** A [
att1 **hasValue** x])
THEN
(b **memberOf** B [
att2 **hasValue** m])

State ω_2

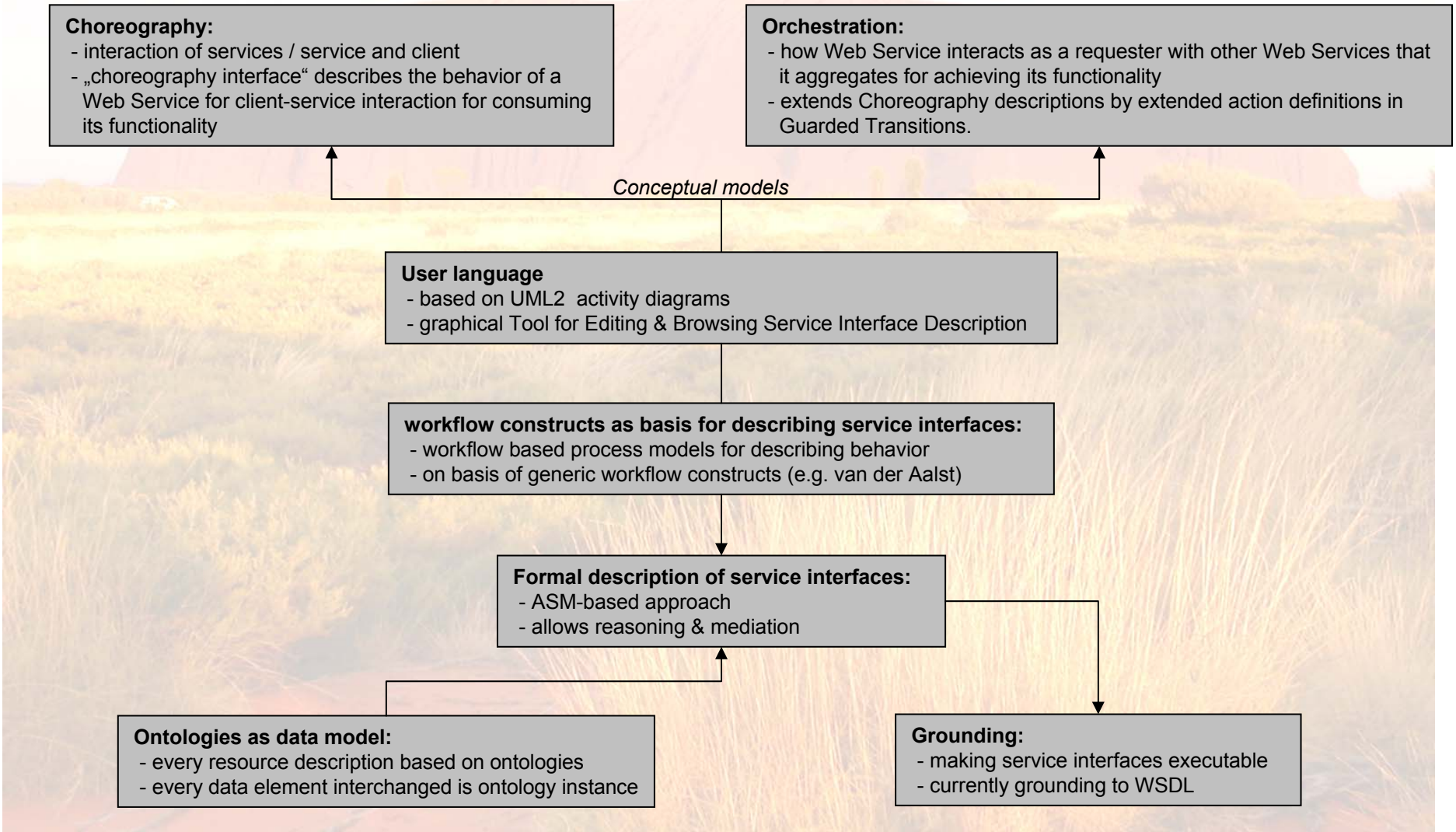
a **memberOf** A [
att1 **hasValue** x,
att2 **hasValue** y]

b **memberOf** B [
att2 **hasValue** m]

received ontology instance ***a***

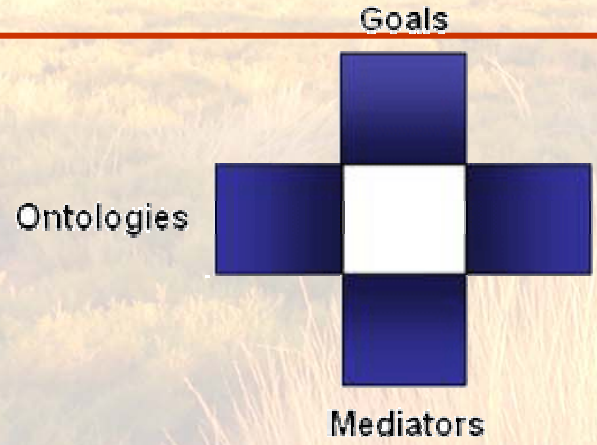
sent ontology instance ***b***

Future Directions



WSMO Goals

Objectives that a client wants to achieve by using Web Services



Provide the formally specified terminology of the information used by all other components

Semantic description of Web Services:
- **Capability** *(functional)*
- **Interfaces** *(usage)*

Connectors between components with mediation facilities for handling heterogeneities

Goals

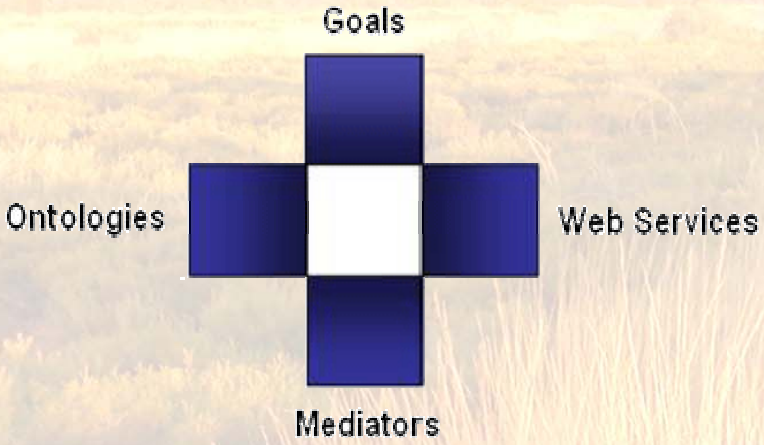
- **Ontological De-coupling of Requester and Provider**
- **Goal-driven Architetcure:**
  - requester formulates objective independently
  - 'intelligent' mechanisms detect suitable services for solving the Goal
  - allows re-use of Services for different purposes
- **Derived from different AI-approaches for intelligent systems**
  - **Intelligent Agents (BDI Architectures)**
  - **Problem Solving Methods**
- **Requests may in principle not be satisfiable**
- **Ontological relationships & mediators used to link goals to web services**
- **Goal Resolution Process open to implementations**

Goal Specification

- **Non functional properties**
- **Imported Ontologies**
- **Used mediators**
  - *OO Mediators:* importing ontologies with heterogeneity resolution
  - *GG Mediator:*
    - Goal definition by reusing an already existing goal
    - allows definition of **Goal Ontologies**
- **Requested Capability**
  - describes service functionality expected to resolve the objective
  - defined as capability description from the requester perspective
- **Requested Interface**
  - describes communication behaviour supported by the requester for consuming a Web Service (Choreography)
  - Restrictions / preferences on orchestrations of acceptable Web Services

WSMO Mediators

Objectives that a client wants to achieve by using Web Services



Provide the formally specified terminology of the information used by all other components

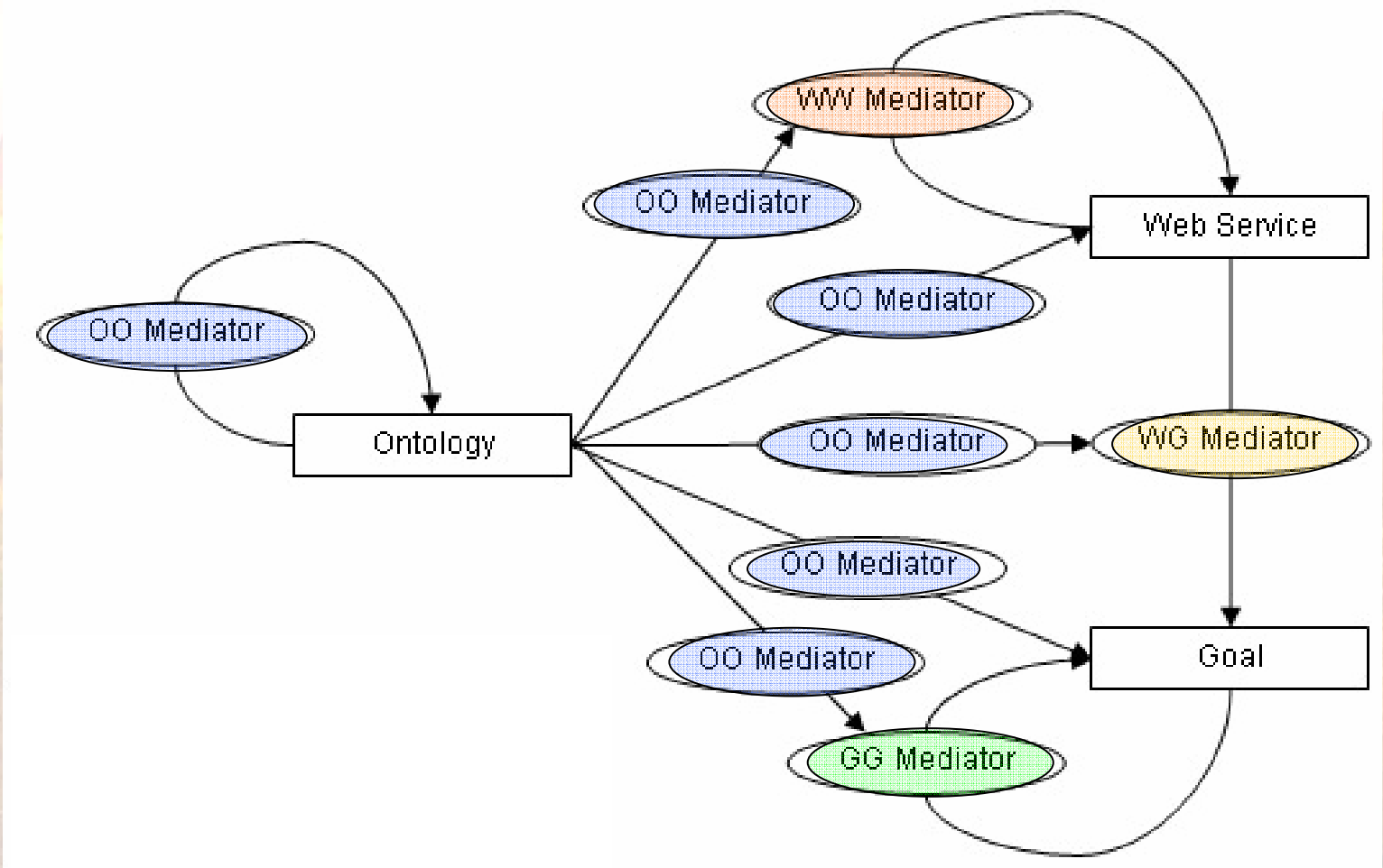
Semantic description of Web Services:
- **Capability** *(functional)*
- **Interfaces** *(usage)*

Connectors between components with mediation facilities for handling heterogeneities

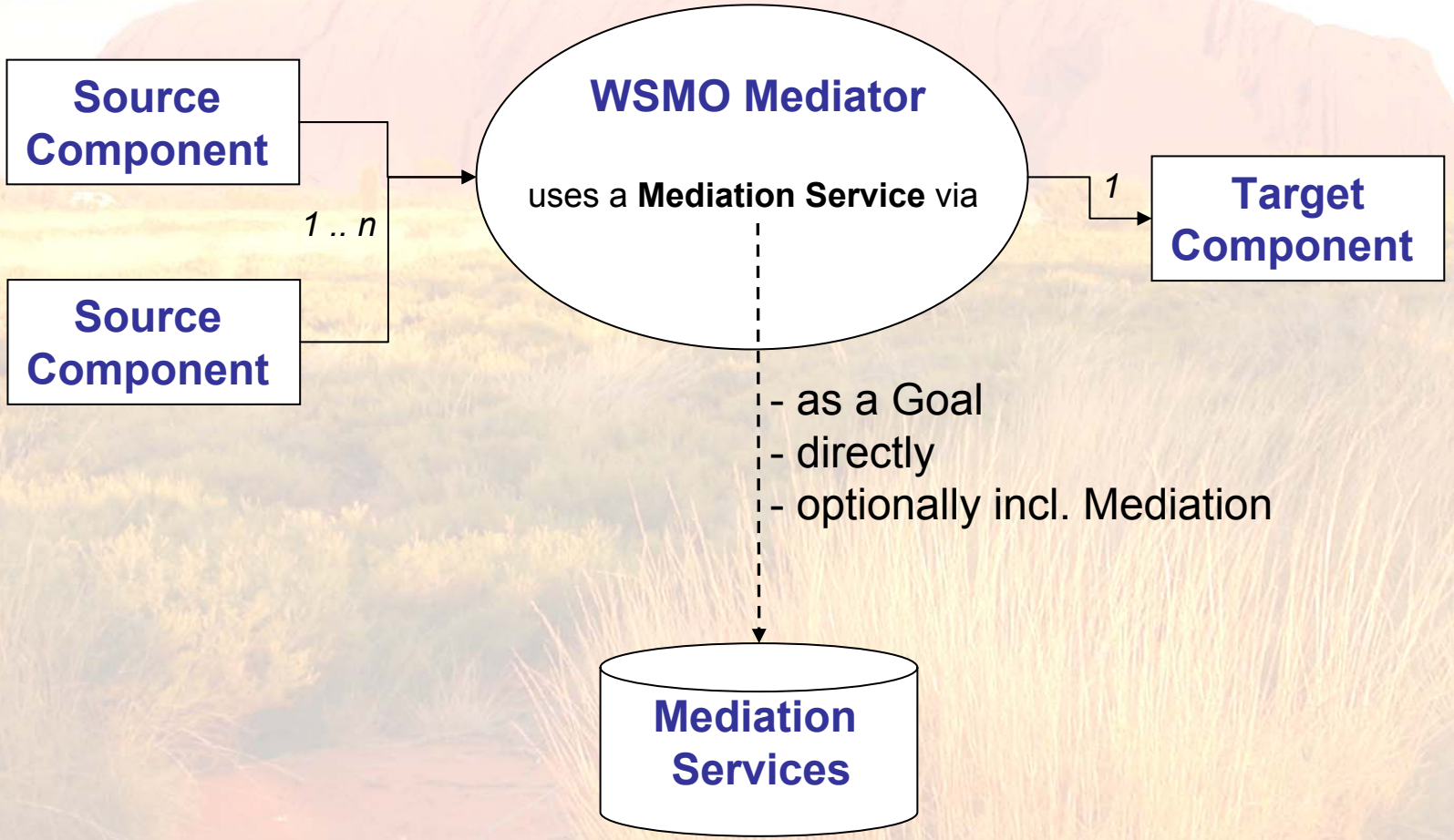
Mediation

- **Heterogeneity …**
  - Mismatches on structural / semantic / conceptual / level
  - Occur between different components that shall interoperate
  - Especially in distributed & open environments like the Internet
- **Concept of Mediation**:
  - ***Mediators*** as components that resolve mismatches
  - Declarative Approach:
    - Semantic description of resources
    - 'Intelligent' mechanisms that resolve mismatches independent of content
  - Mediation cannot be fully automated (integration decision)
- **Levels of Mediation within Semantic Web Services** (WSMF):
  - **(1) Data Level:** mediate heterogeneous Data Sources
  - **(2) Protocol Level:** mediate heterogeneous Communication Patterns
  - **(3) Process Level:** mediate heterogeneous Business Processes

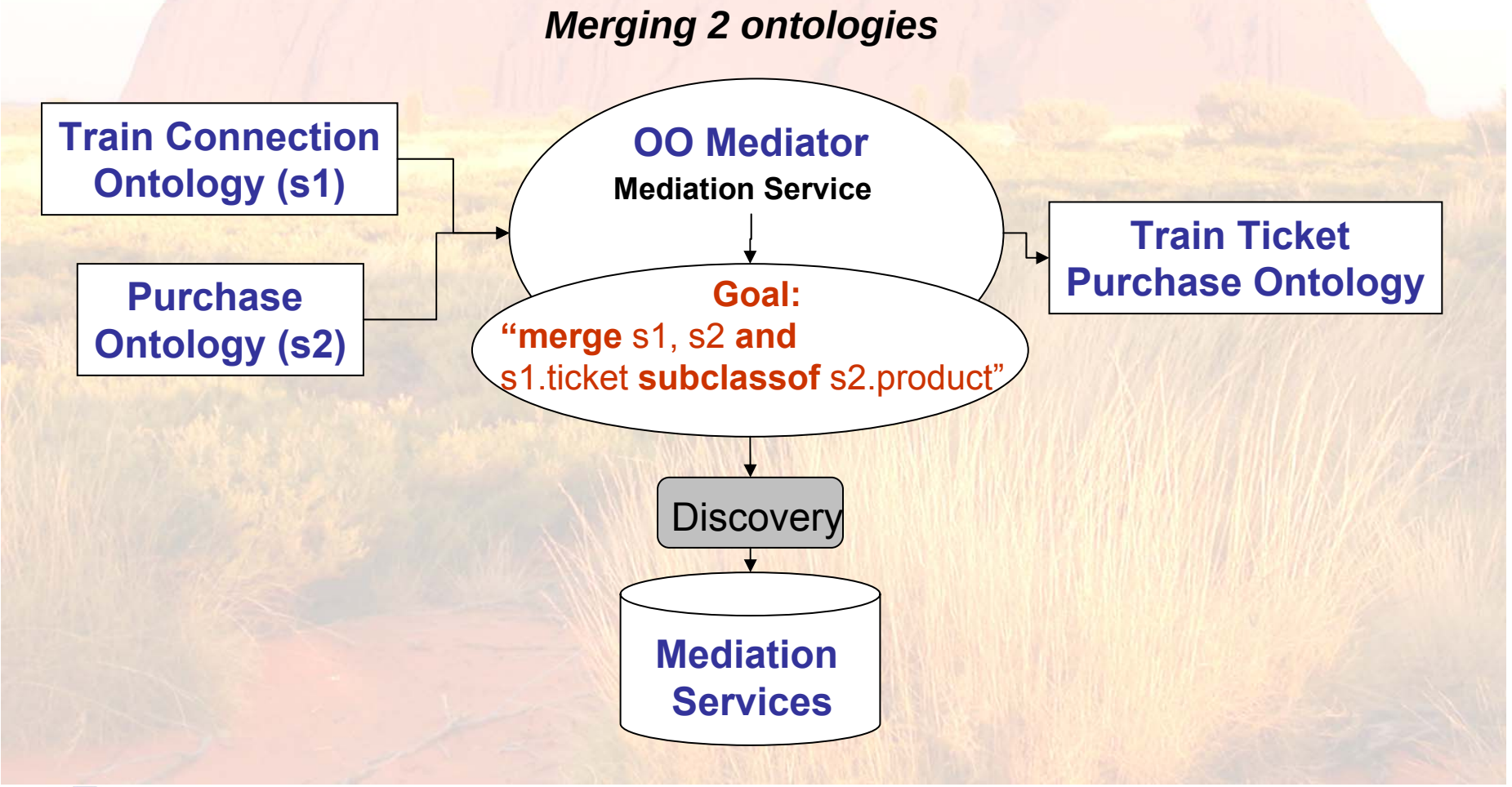
WSMO Mediators Overview



Mediator Structure



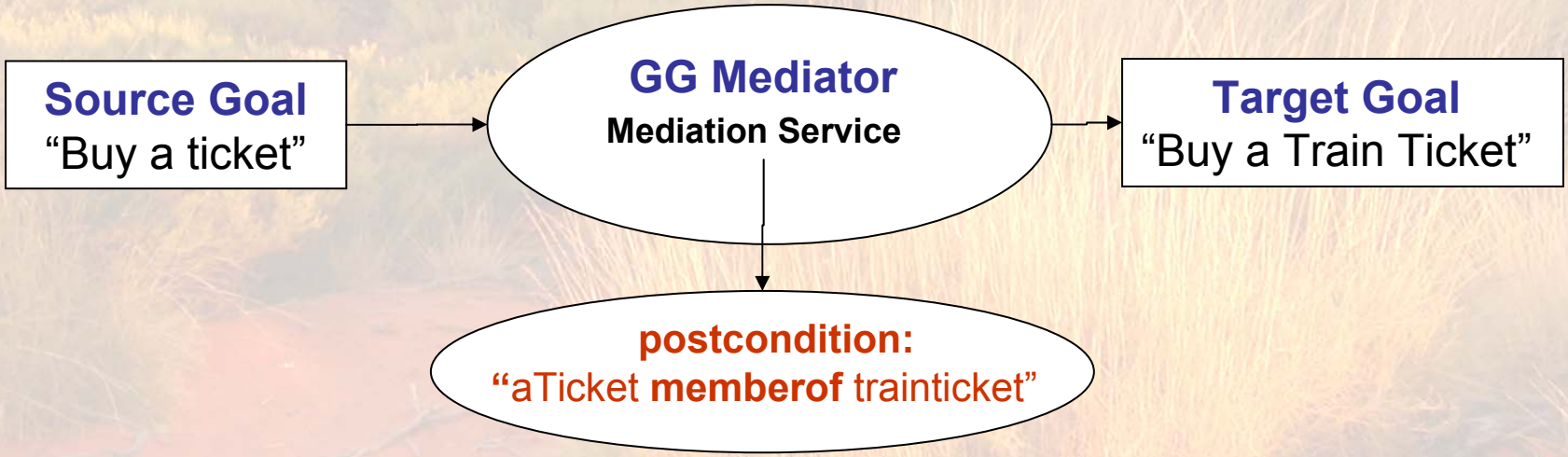
OO Mediator - Example



GG Mediators

- **Aim:**
  - Support specification of Goals by re-using existing Goals
  - Allow definition of **Goal Ontologies** (collection of pre-defined Goals)
  - Terminology mismatches handled by OO Mediators

- **Example: Goal Refinement**



WG & WW Mediators

- **WG Mediators:**
  - link a Web Service to a Goal and resolve occurring mismatches
  - match Web Service and Goals that do not match a priori
  - handle terminology mismatches between Web Services and Goals

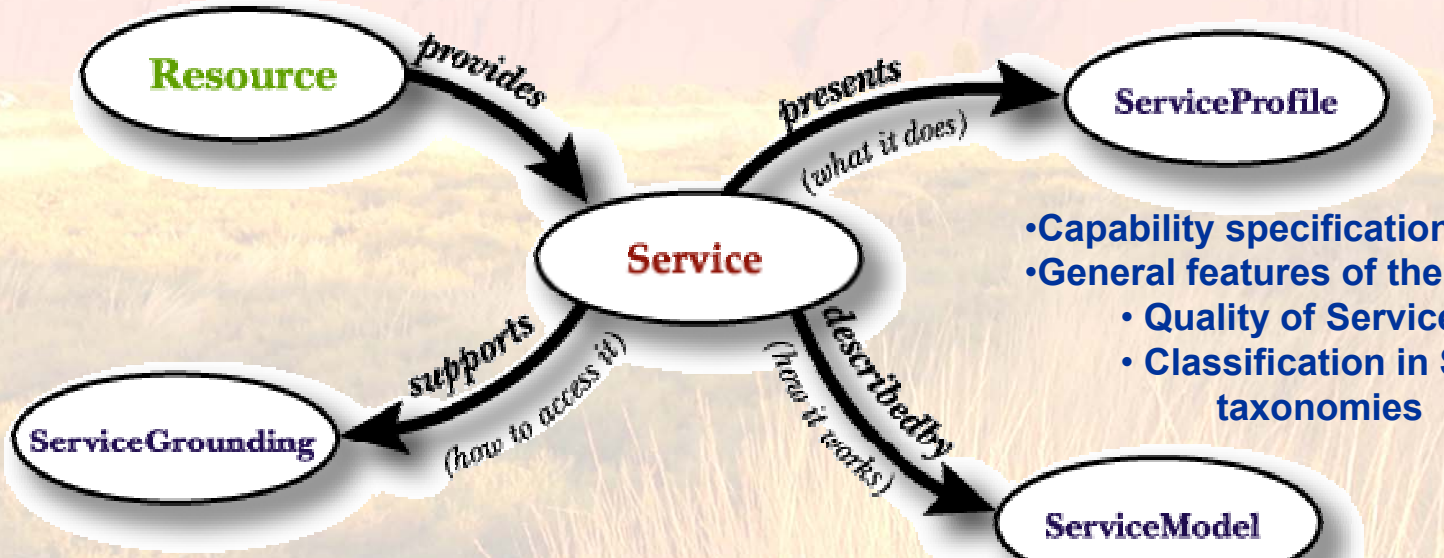
  ⇒ broader range of Goals solvable by a Web Service

- **WW Mediators:**
  - enable interoperability of heterogeneous Web Services

  ⇒ support automated collaboration between Web Services

  - **OO Mediators** for terminology import with data level mediation
  - Protocol Mediation for establishing valid multi-party collaborations
  - Process Mediation for making Business Processes interoperable

Comparison to OWL-S



- Capability specification
- General features of the Service
  - Quality of Service
  - Classification in Service taxonomies

- Mapping to WSDL
  - communication protocol (RPC, HTTP, …)
  - marshalling/serialization
  - transformation to and from XSD to OWL

- Control flow of the service
  - Black/Grey/Glass Box view
- Protocol Specification
- Abstract Messages

Perspective

- OWL-S is an ontology and a language to describe Web services
  - Strong relation to Web Services standards
    - rather than proposing another WS standard, OWL-S aims at enriching existing standards
    - OWL-S is grounded in WSDL and it has been mapped into UDDI
  - Based on the Semantic Web
    - Ontologies provide conceptual framework to describe the domain of Web services and an inference engine to reason about the domain
    - Ontologies are essential elements of interoperation between Web services
- WSMO is a conceptual model for the core elements of Semantic Web Services
  - core elements: Ontologies, Web Services, Goals, Mediators
    - language for semantic element description (WSML)
    - reference implementation (WSMX)
  - Mediation as a key element
  - Ontologies as data model
    - every resource description is based on ontologies
    - every data element interchanged is an ontology instance

OWL-S and WSMO

OWL-S profile ≈ WSMO capability +
goal +
non-functional properties

- OWL-S uses Profiles to express existing capabilities (advertisements) and desired capabilities (requests)
- WSMO separates provider (capabilities) and requester points of view (goals)

OWL-S and WSMO

OWL-S Process Model ≈ WSMO Service Interfaces

- Perspective:
  - OWL-S Process Model describes operations performed by Web Service, including consumption as well as aggregation
  - WSMO separates Choreography and Orchestration

- Formal Model:
  - OWL-S formal semantics has been developed in very different frameworks such as Situation Calculus, Petri Nets, Pi-calculus
  - WSMO service interface description model with ASM-based formal semantics
  - OWL-S Process Model is extended by SWRL / FLOWS

both approaches are not finalized yet

OWL-S and WSMO

OWL-S Grounding ≈ current WSMO Grounding

- OWL-S provides default mapping to WSDL
  - clear separation between WS description and interface implementation
  - other mappings could be used
- WSMO also defines a mapping to WSDL, but aims at an ontology-based grounding
  - avoid loss of ontological descriptions throughout service usage process
  - ‘Triple-Spaced Computing’ as innovative communication technology

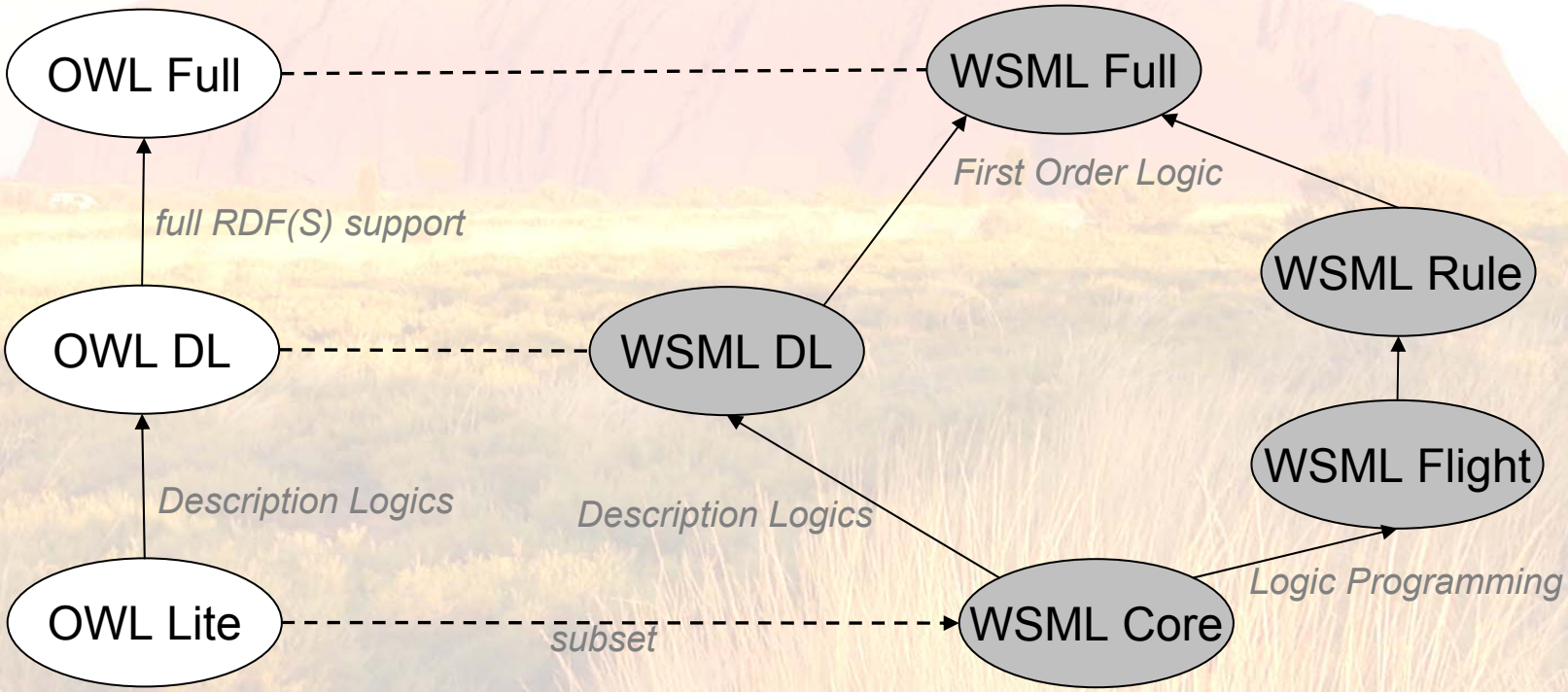
Mediation in OWL-S and WSMO

- OWL-S does not have an explicit notion of mediator
  - Mediation is a by-product of the orchestration process
    - E.g. protocol mismatches are resolved by constructing a plan that coordinates the activity of the Web services
  - …or it results from translation axioms that are available to the Web services
    - It is not the mission of OWL-S to generate these axioms
- WSMO regards mediators as key conceptual elements
  - Different kinds of mediators:
    - OO Mediators for ensuring semantic interoperability
    - GG, WG mediators to link Goals and Web Services
    - WW Mediators to establish service interoperability
  - Reusable mediators
  - Mediation techniques under development

Semantic Representation

- OWL-S and WSMO adopt a similar view on the need of ontologies and explicit semantics but they rely on different logics:
  - OWL-S is based on OWL / SWRL
    - OWL represent taxonomical knowledge
    - SWRL provides inference rules
    - FLOWS as formal model for process model
  - WSMO is based on WSML a family of languages with a common basis for compatibility and extensions in the direction of Description Logics and Logic Programming

OWL and WSML



- WSML aims at overcoming deficiencies of OWL
- Relation between WSML and OWL+SWRL to be defined

Summary

	OWL-S	WSMO	*current Web Service technologies*
Discovery *detection of suitable WS*	Profile	Goals and Web Services (capability)	*UDDI API*
Consumption & Interaction *How to consume & aggregate*	Process Model	Service Interfaces (Choreography + Orchestration)	*BPEL4WS / WS-CDL*
Invocation *How to invoke*	Grounding+ WSDL/SOAP	Grounding (WSDL / SOAP, ontology-based)	*WSDL / SOAP*
Mediation *Heterogeneity handling*	-	Mediators	-

PART III:

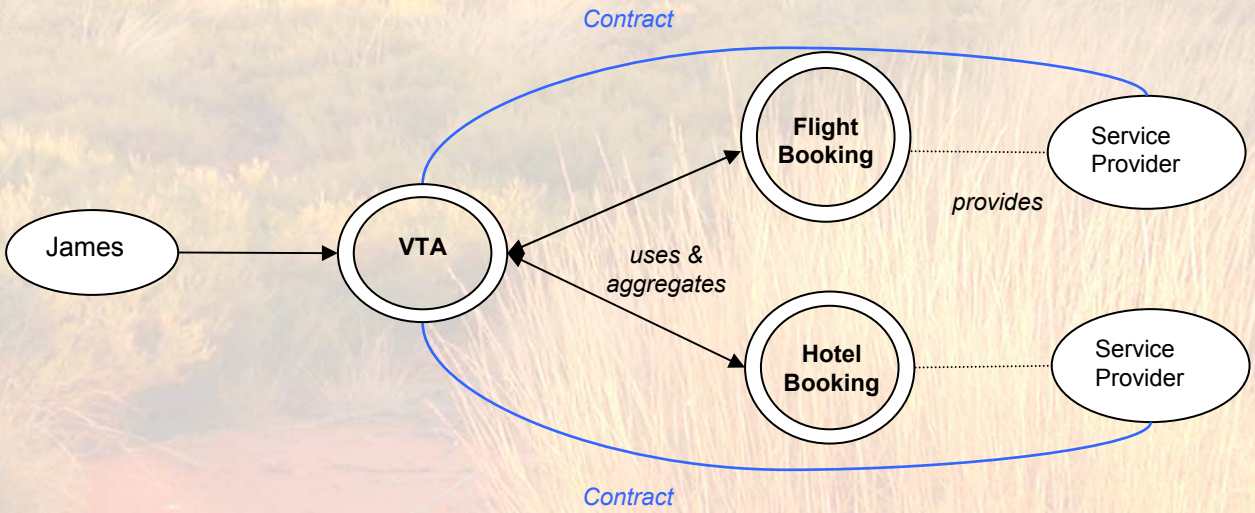
A Walkthrough Example

- Virtual Travel Agency Use Case Overview
- Semantic Web Services Modeling
- Discovery
- Conversation Validation
- Mediation

Virtual Travel Agency Use Case

- James is employed in DERI Austria and wants to book a flight and a hotel for the ICWE conference
- the start-up company VTA provides tourism and business travel services based on Semantic Web Service technology

=> how does the interplay of James, VTA, and other Web Services look like?



Domain Ontologies

- All terminology that we use in resource descriptions need to be based on ontologies; also, all information interchanged should be ontology instances
- Modular ontology design
  - an ontology should define terminology of a domain
  - an ontology should be shared & agreed in a community
  - modular ontologies are combined in specific
- Domain Ontologies needed for this Use Case:
  - Trip Reservation Ontology
  - Location Ontology
  - Date and Time Ontology
  - Purchase Ontology
  - … possibly more

Trip Reservation Ontology

- defines the terminology for trips (traveling, accomodation, holiday / business travel facilities) and reservations
- provided by community of interest (e.g. Austrian Tourism Association)

- main concepts:
  - TRIP
    - describes a trip (a journey between locations)
    - passenger, origin & destination, means of travel, etc.
  - RESERVATION
    - describes reservations for tickets, accomodation, or complete trips
    - customer, trip, price, payment
  - RESERVATION REQUEST
  - RESERVATION OFFER
  - RESERVATION CONFIRMATION

- uses other ontologies:
  - Location Ontology for origin & destination specification
  - Date and Time Ontology for departure, arrival, duration information
  - Purchase Ontology for payment related aspects

Trip Reservation Ontology

```
namespace {_"http://www.wsmo.org/ontologies/tripReservationOntology",
 dc      _"http://purl.org/dc/elements/1.1#",
 xsd     _"http://www.w3.org/2001/XMLSchema#",
 loc     _“http://www.daml.org/2003/09/factbook/factbook-ont#",
 dt      _"http://www.wsmo.org/ontologies/dateandtime.wsml#",
 po      _"http://www.wsmo.org/ontologies/purchase.wsml#"}

ontology _"http://www.wsmo.org/ontologies/tripReservationOntology#"

nonFunctionalProperties
 dc#title hasValue "Trip Reservation Ontology"
 dc#creator hasValue _”http://www.deri.org”
 dc#description hasValue “domain ontology for travel and accomodation reservation"
 dc#publisher hasValue “Austrian Toursim Association"
 version hasValue "$Revision 1.17 $"
endNonFunctionalProperties

importsOntology   {_"http://www.wsmo.org/ontologies/dateandtime.wsml",
                                  _"http://www.wsmo.org/ontologies/purchase.wsml“}

usesMediator {_"http://www.wsmo.org/mediators/owl2wsml.wsml“}
```

Trip Reservation Ontology

```
concept trip
passenger impliesType po#person
origin impliesType loc#location
destination impliesType loc#location
departureDate ofType dt#dateandtime
returnDate ofType dt#dateandtime
meansOfTransport impliesType meansOfTransport
accomodation impliesType accomodation

concept reservation
 nonFunctionalProperties
  dc#description hasValue "reservations for tickets, accomodation, or complete trips“
  dc#relation hasValue reservationItemDef
 endNonFunctionalProperties
customer impliesType po#customer
reservationItem impliesType wsml#true
price impliesType po#price
payment impliesType po#payment
axiom reservationItemDef
 definedBy
  forall{?x, ?y} (?x memberOf reservation[reservationItem hasValue ?y] impliedBy
     (?y memberOf ticket) or (?y memberOf accomodation) or (?y memberOf trip) ).
```

Ontology Modelling Remarks

- Ontology Design for the Semantic Web
  - "real ontologies, no crappy data models" (Dieter Fensel)
  - (re-)use existing, widely accepted ontologies
- Ontology Design is a very difficult and challenging tasks
  - determine agreed conceptualization of domain
  - correct formalization (e.g. misuse of is_a / part_of relations)

  => requires expertise in knowledge engineering
- Ontology Engineering Methodologies & Technology Support essential
  - editing, browsing, maintenance
  - storage and retrieval
  - ontology evolution support
  - ontology integration techniques

Goal Description

- "book flight and hotel for the ICWE 2005 for James"
- goal capability postcondition: get a trip reservation for this

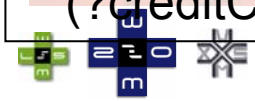
```
goal _"http://www.wsmo.org/examples/goals/icwe2005"
  importsOntology {_"http://www.wsmo.org/ontologies/tripReservationOntology", …}
  capability
    postcondition
      definedBy
        ?tripReservation memberOf tr#reservation[
          customer hasValue fof#james,
          reservationItem hasValue ?tripICWE] and
         ?tripICWE memberOf tr#trip[
           passenger hasValue fof#james,
           origin hasValue loc#innsbruck,
           destination hasValue loc#sydney,
           meansOfTransport hasValue ?flight,
           accomodation hasValue ?hotel] and
        ?flight[airline hasValue tr#staralliance] memberOf tr#flight and
        ?hotel[name hasValue "Capitol Square Hotel"] memberOf tr#hotel .
```

VTA Service Description

- book tickets, hotels, amenities, etc.
- capability description (pre-state)

```
capability VTAcapability
 sharedVariables {?creditCard, ?initialBalance, ?item, ?passenger}
 precondition
  definedBy
   ?reservationRequest[
        reservationItem hasValue ?item,
        passenger hasValue ?passenger,
        payment hasValue ?creditcard,
    ] memberOf tr#reservationRequest and
    ((?item memberOf tr#trip) or (?item memberOf tr#ticket)) and
    ?creditCard[balance hasValue ?initialBalance] memberOf po#creditCard .

 assumption
  definedBy
   po#validCreditCard(?creditCard) and
   (?creditCard[type hasValue po#visa] or ?creditCard[type hasValue po#mastercard]).
```



VTA Service Description

- capability description (post-state)

```
postcondition
    definedBy
      ?reservation[
          reservationItem hasValue ?item,
          customer hasValue ?passenger,
          payment hasValue ?creditcard
       ] memberOf tr#reservation .

  assumption
    definedBy
     reservationPrice(?reservation, "euro", ?tripPrice) and
      ?finalBalance= (?initialBalance - ?ticketPrice) and
      ?creditCard[po#balance  hasValue  ?finalBalance] .
```

Web Service Discovery

Semantic Web Service Discovery

find appropriate Web Service for automatically resolving a goal as the objective of a requester

- Aims:
  - high precision discovery
  - maximal automation
  - effective discoverer architectures
- Requirements:
  - infrastructure that allows storage and retrieval of information about Web services
  - description of Web services functionality
  - description of requests or goals
  - algorithms for matching requesters for capabilities with the corresponding providers

Discovery Techniques

- different techniques available
  - trade-off: ease-of-provision <-> accuracy
  - resource descriptions & matchmaking algorithms

Key Word Matching

match natural language key words in resource descriptions

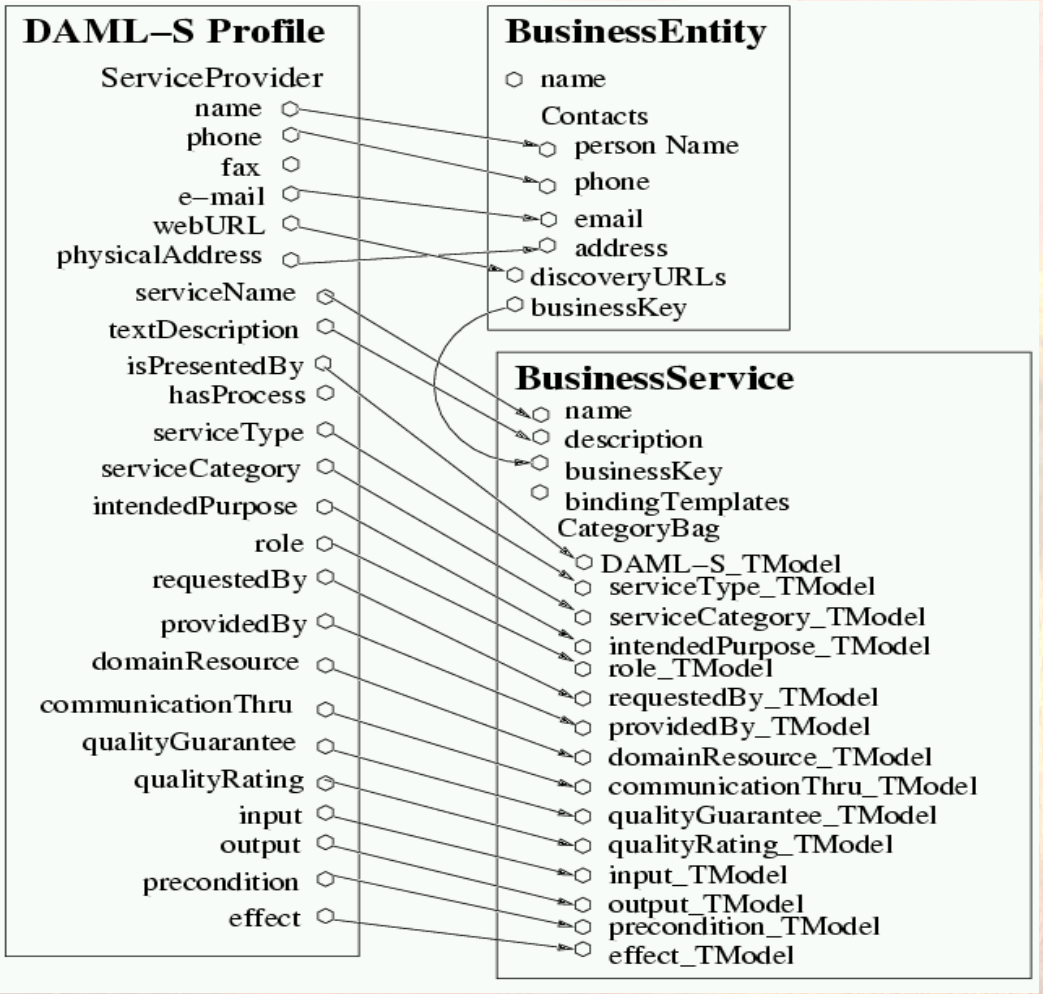
Controlled Vocabulary

ontology-based key word matching

Semantic Matchmaking

… what Semantic Web Services aim at

Semantic Web Services in UDDI



- Mapping semantic resource descriptions into UDDI
- OWL-S Service Profile mapping to UDDI
- WSMO elements to UDDI mapping (for all top level elements)

⇒ ***mapping semantic descriptions to syntactic repository***

⇒ ***allows retrieval of structural information***

Controlled Vocabulary

WSMO non-functional properties

- Ontology keywords in non-functional properties
  - dc#subject contains main ontology concepts related to Web Service
  - allows pre-filtering similar to OWL-S Profile Hierarchy, but on basis on ontologies (“controlled vocabulary”)

- Example
  - a Web Service for selling train tickets in Austria

    dc#subject **hasValue** _{tc#trainticket, po#purchase, loc#austria}
  - does not precisely describe Web Service functionality

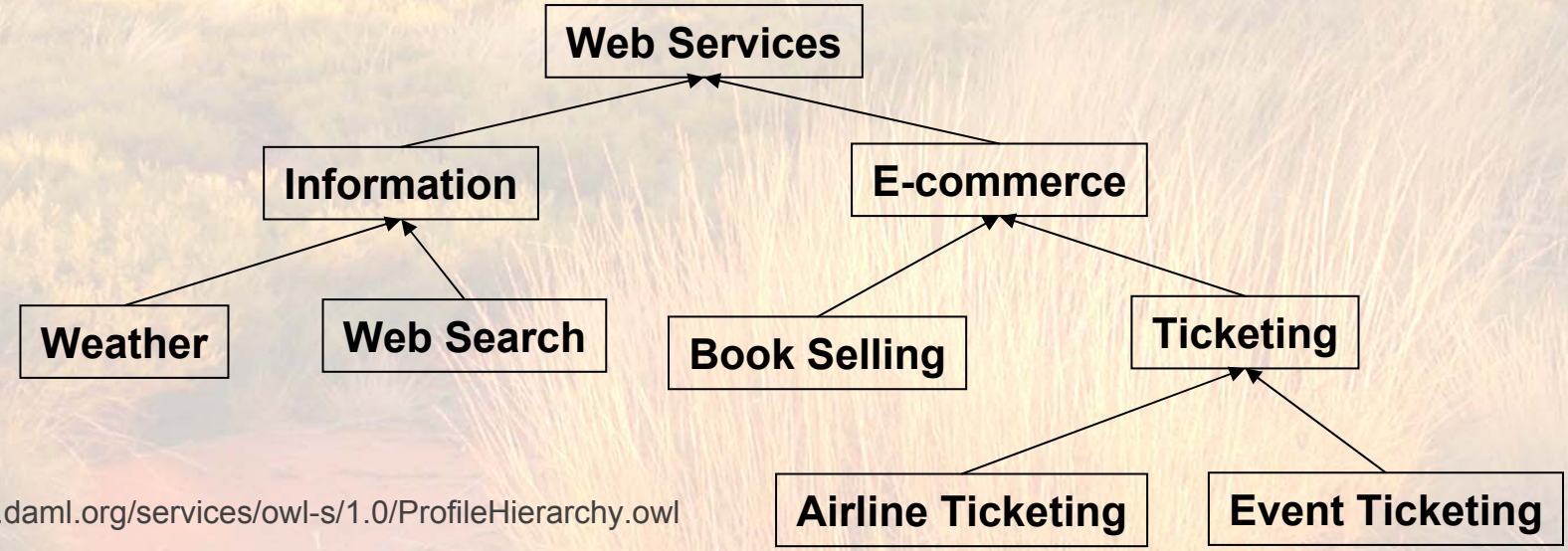
  => accuracy of discovery result meager

Lara, R., Lausen, H.; Toma, I.: (Eds): *WSMX Discovery*. WSMX Working Draft D10 v0.2, 07 March 2005.

Controlled Vocabulary

OWL-S Profile Hierarchies

- hierarchy of Web Services
  - functional similarities (domain, in- / outputs)
  - allows pre-filtering of services on basis of categorization

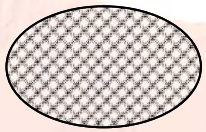


http://www.daml.org/services/owl-s/1.0/ProfileHierarchy.owl

Matchmaking Notions & Intentions

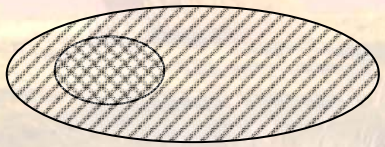
Exact Match:

G, WS, O, M ⊨ ∀x. (G(x) <=> WS(x))



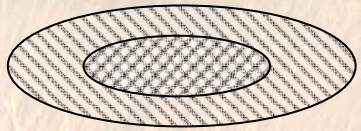
PlugIn Match:

G, WS, O, M ⊨ ∀x. (G(x) => WS(x))



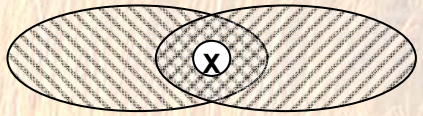
Subsumption Match:

G, WS, O, M ⊨ ∀x. (G(x) <= WS(x))



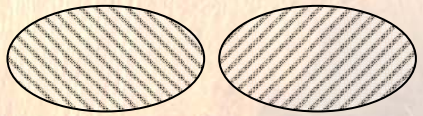
Intersection Match:

G, WS, O, M ⊨ ∃x. (G(x) ∧ WS(x))



Non Match:

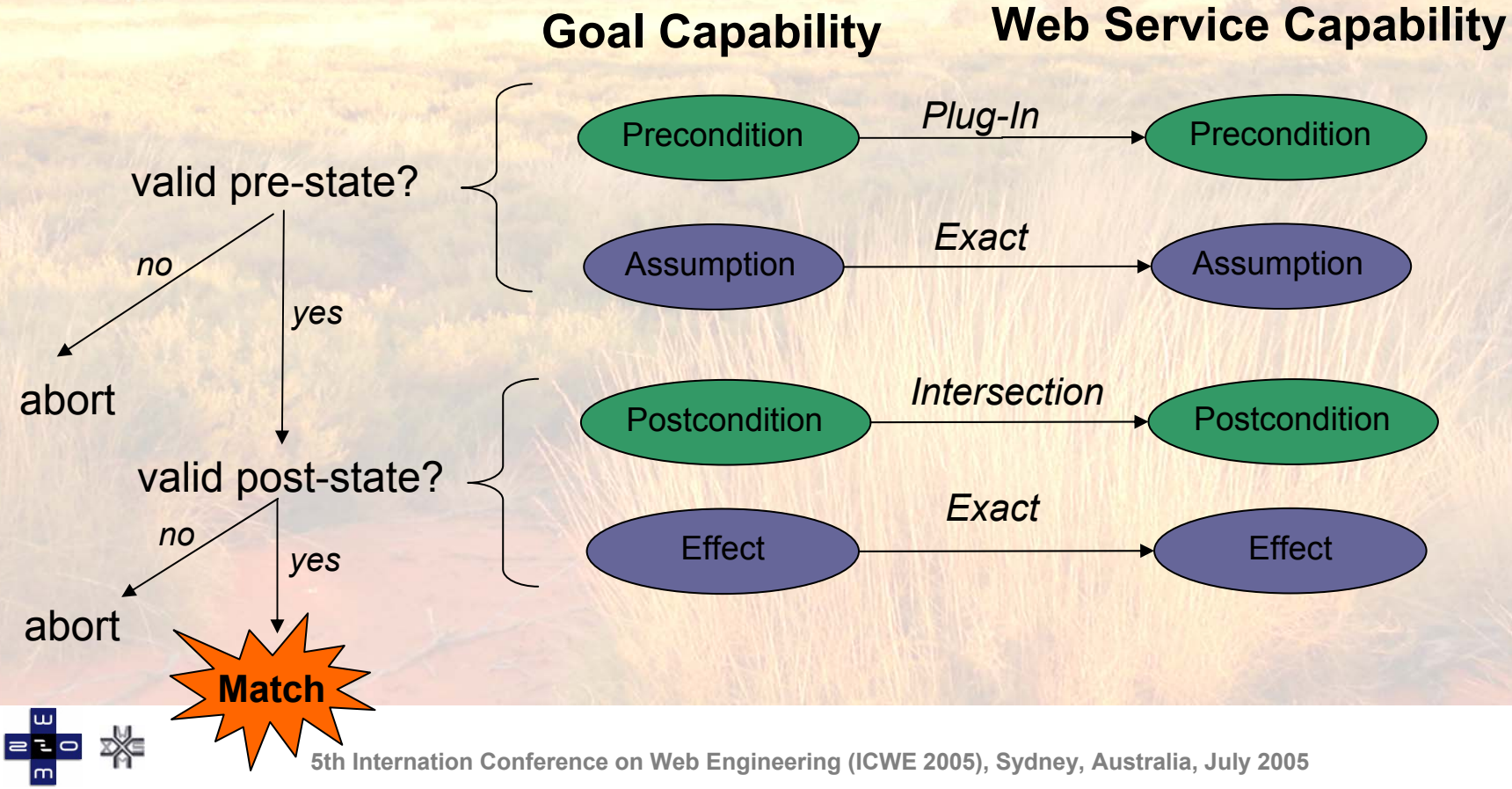
G, WS, O, M ⊨ ¬∃x. (G(x) ∧ WS(x))



Keller, U.; Lara, R.; Polleres, A. (Eds): *WSMO Web Service Discovery*. WSML Working Draft D5.1, 12 Nov 2004.

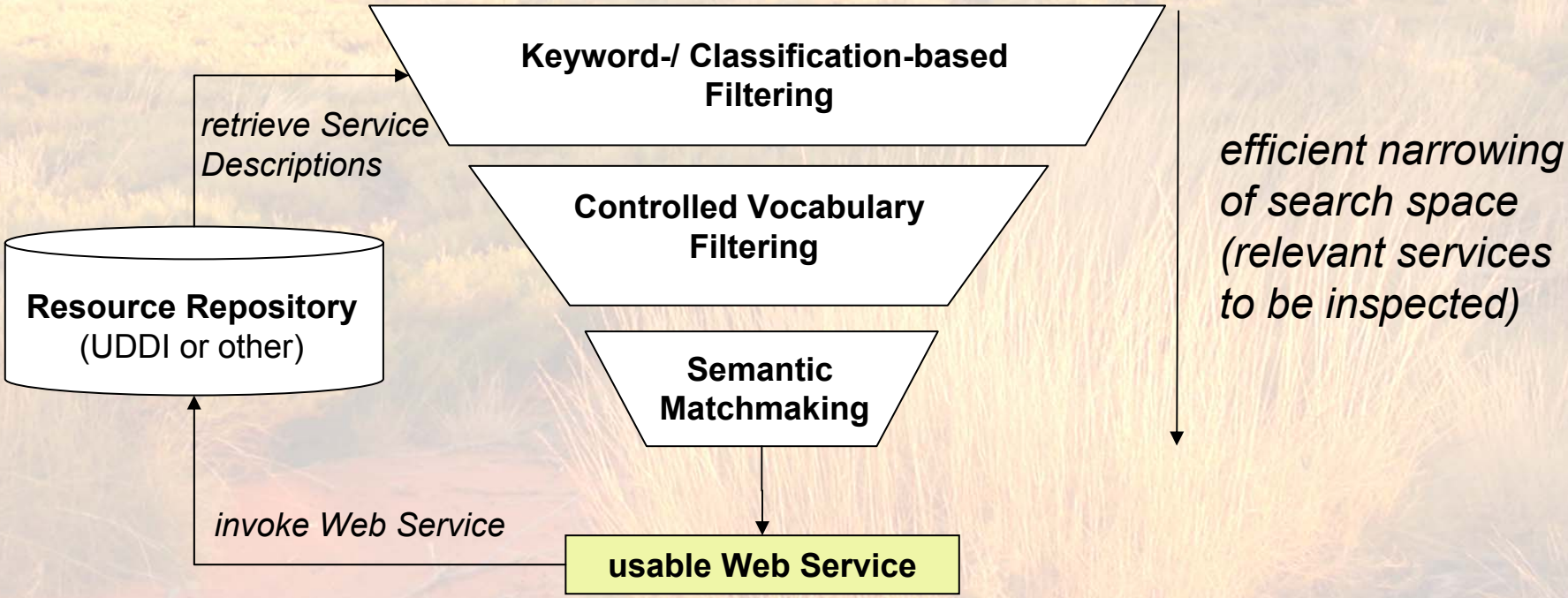
Discovery Approach

- Matchmaking Notion to be used defined for each goal capability element
- **Basic Procedure:**

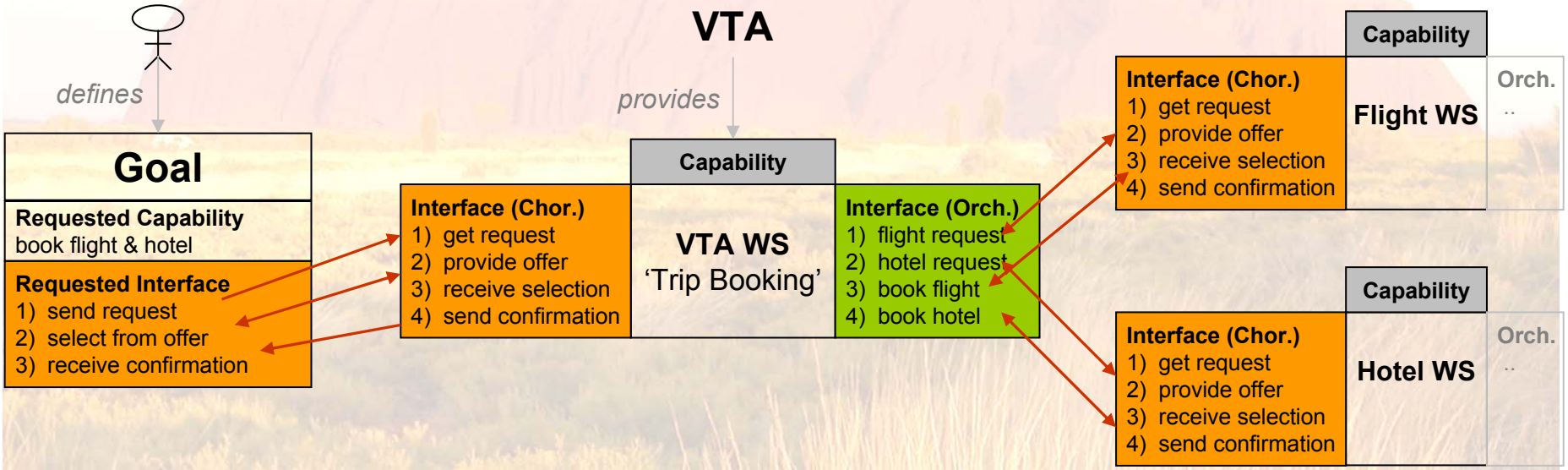


Discoverer Architecture

- Discovery as central Semantic Web Services technology
- Integrated Discoverer Architectures (under construction):



Service Interfaces



Choreography Interface: how entity can interact

Choreography: interaction between entities

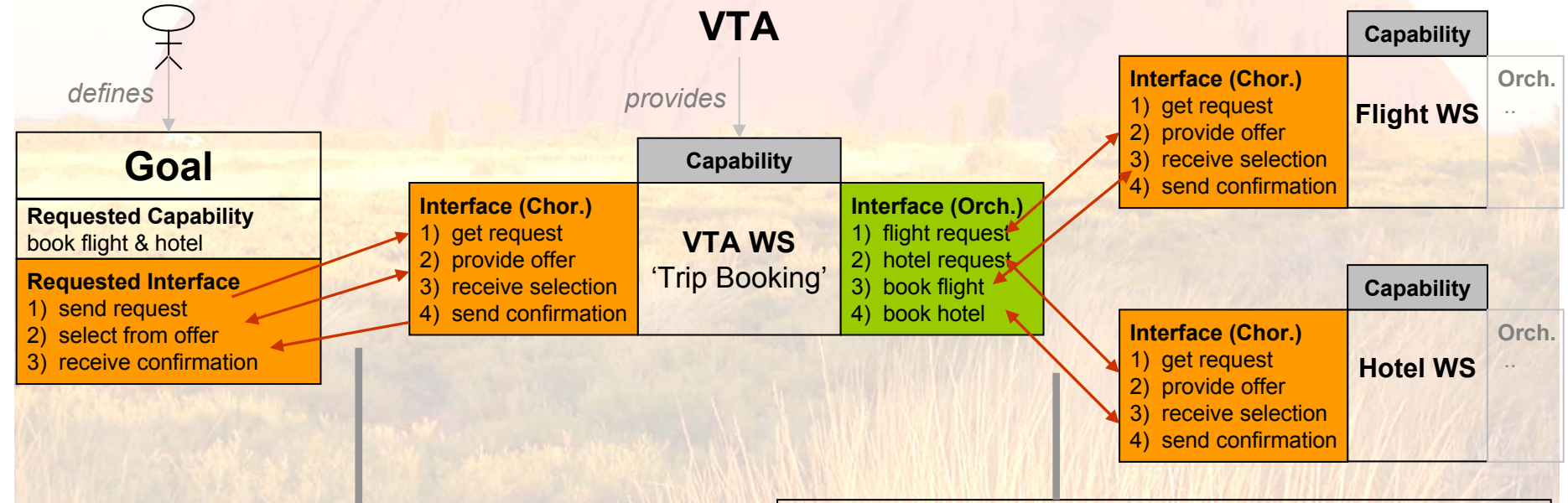
Orchestration: service aggregation for realizing functionality

VTA Service Description

- Choreography Interface
- transition “get request” to “provide offer”

```
choreography VTABehaviorInterface
 importsOntology {_"http://www.wsmo.org/ontologies/tripReservationOntology“}
 vocabularyIn {reservationRequest, …}
 vocabularyOut {reservationOffer, …}
guardedTransitions VTABehaviorInterfaceTransitionRules
  if (reservationRequest memberOf tr#reservationRequest[
      customer hasValue ?Customer,
      reservationItem hasValue ?Trip] and
      ?Trip memberOf tr#trip[
         passenger hasValue ?Passenger,
         origin hasValue ?LocationInAustria] and
     ?LocationInAustria memberOf loc#location[
         inCountry hasValue loc#austria]
  then reservationOffer memberOf tr#reservationOffer[
     customer hasValue ?Customer,
     reservationItem hasValue ?Trip ] .
```

Choreography Discovery



- both choreography interfaces given ("static")
- correct & complete consumption of VTA

=> existence of a valid choreography?

- VTA Orchestration & Chor. Interfaces of aggregated WS given

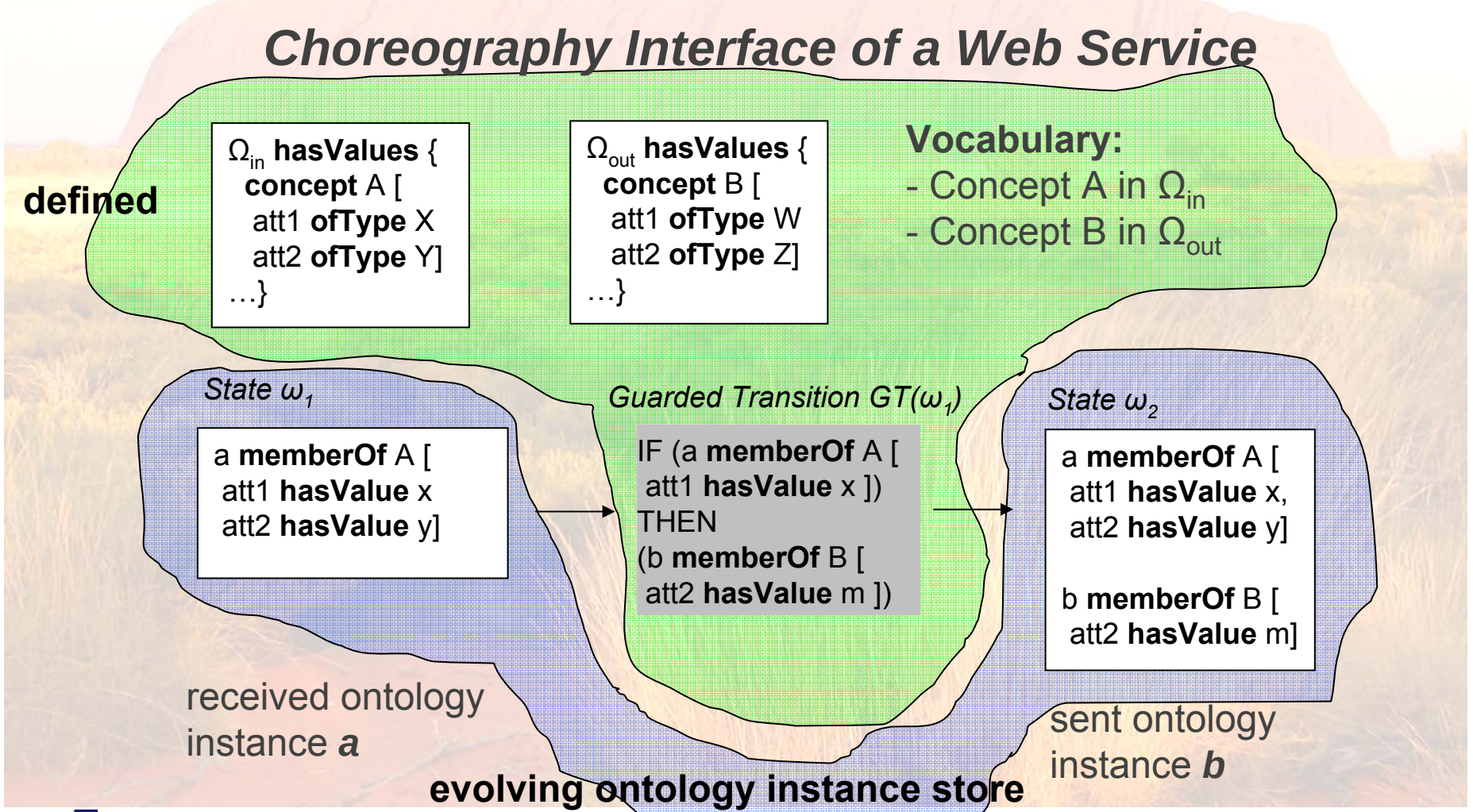
=> existence of a valid choreography between VTA and each aggregated WS?

- **Choreography Discovery** as a central reasoning task in Service Interfaces
- 'choreographies' do not have to be described, only existence determination

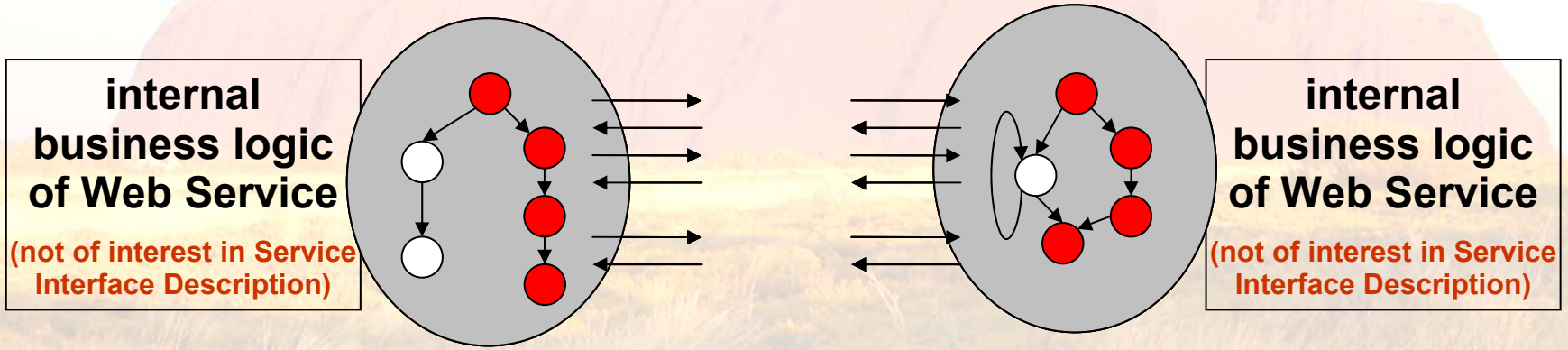
WSMO Service Interface Description Model

- common formal model for Service Interface description
  - ontologies as data model
  - based on ASMs
  - not restricted to any executable communication technology
- general structure:
  - Vocabulary Ω:
    - ontology schema(s) used in service interface description
    - usage for information interchange: in, out, shared, controlled
  - States ω(Ω):
    - a stable status in the information space
    - defined by attribute values of ontology instances
  - Guarded Transition GT(ω):
    - state transition
    - general structure: ***if*** (condition) ***then*** (action)
    - different for Choreography and Orchestration
    - additional constructs: *add, delete, update*

Service Interface Example



Choreography Discovery



- a valid choreography exists if:
  1) **Information Compatibility**
     - compatible vocabulary
     - homogeneous ontologies
  2) **Communication Compatibility**
     - start state for interaction
     - a termination state can be reached without any additional input

Information Compatibility

If choreography participants have compatible vocabulary definitions:

- $\Omega_{in}(S1)$ and $\Omega_{shared}(S1) = \Omega_{out}(S2)$ and $\Omega_{shared}(S2)$
- determinable by Intersection Match from Discovery

$$SI_{S1}, SI_{S2}, O, M \models \exists x.\ (\Omega_{S1(in\ U\ shared)}(x) \wedge \Omega_{S2(out\ U\ shared)}(x))$$

- more complex for multi-party choreographies

Prerequisite: choreography participants use homogeneous ontologies:

- semanticInteroperability(S1, S2, …, Sn)
- usage of same ontologies in Service Interfaces or respective OO Mediators

Communication Compatibility

- Definitions (for "binary choreography" (only 2 services), more complex for multi-party choreographies)

Valid Choreography State:

$\omega_x(C(S1, S2))$ **if** informationCompatibility $(\Omega S1(\omega_x), \Omega S2(\omega_x))$

  - means: action in GT of S1 for reaching state $\omega_x(S1)$ satisfies condition in GT of S2 for reaching state $\omega_x(S2)$, or vice versa

Start State:

$\omega_\emptyset(C(S1, S2))$ **if** $\Omega S1(\omega_\emptyset)=\emptyset$ and $\Omega S2(\omega_\emptyset)=\emptyset$ and $\exists\ \omega_1(C(S1, S2))$

  - means: if initial states for choreography participants given (empty ontology, i.e. no information interchange has happened), and there is a valid choreography state for commencing the interaction

Termination State:

$\omega_T(C(S1, S2))$ **if** $\Omega S1(\omega_T)$=noAction and $\Omega S2(\omega_T)$=noAction and $\exists\ \omega_T(C(S1, S2))$

  - means: there exist termination states for choreography participants (no action for transition to next state), and this is reachable by a sequence of valid choreography states

- Communication Compatibility given if there exists a start state and a termination state is reachable without additional input by a sequence of valid choreography states

Communication Compatibility Example

James' Goal Behavior Interface

$\Omega_{S1}(\omega\emptyset) = \{\emptyset\}$

if ∅ **then** request

$\Omega_{S1}(\omega1) = \{request(out)\}$

if cnd1(offer) **then** changeReq

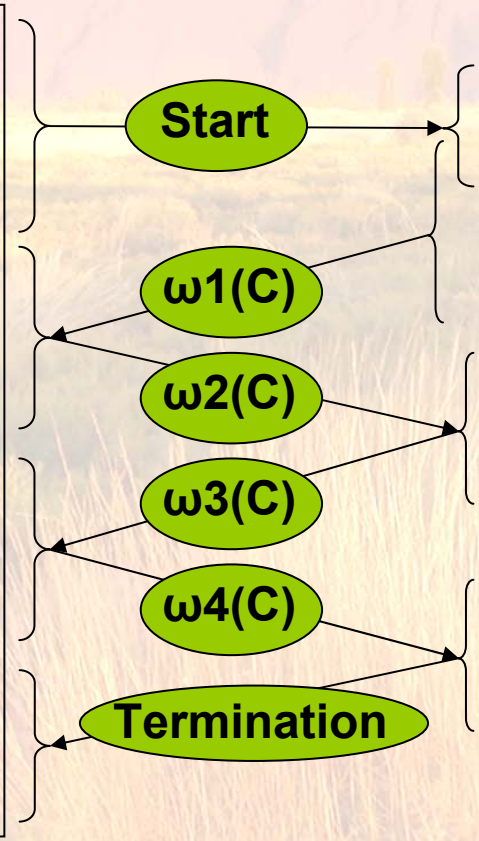
$\Omega_{S1}(\omega2a) =$
{offer(in), changeReq(out)}

if cnd2(offer) **then** order

$\Omega_{S1}(\omega2b) =$
{offer(in), order(out)}

if conf **then** ∅

$\Omega_{S1}(\omega3) = \{offer(in), conf(in)\}$



VTA Behavior Interface

$\Omega_{S2}(\omega\emptyset) = \{\emptyset\}$

if request **then** offer

$\Omega_{S2}(\omega1) =$
{request(in), offer(out)}

if changeReq **then** offer

$\Omega_{S2}(\omega2a) =$
{changeReq(in),offer(out)}

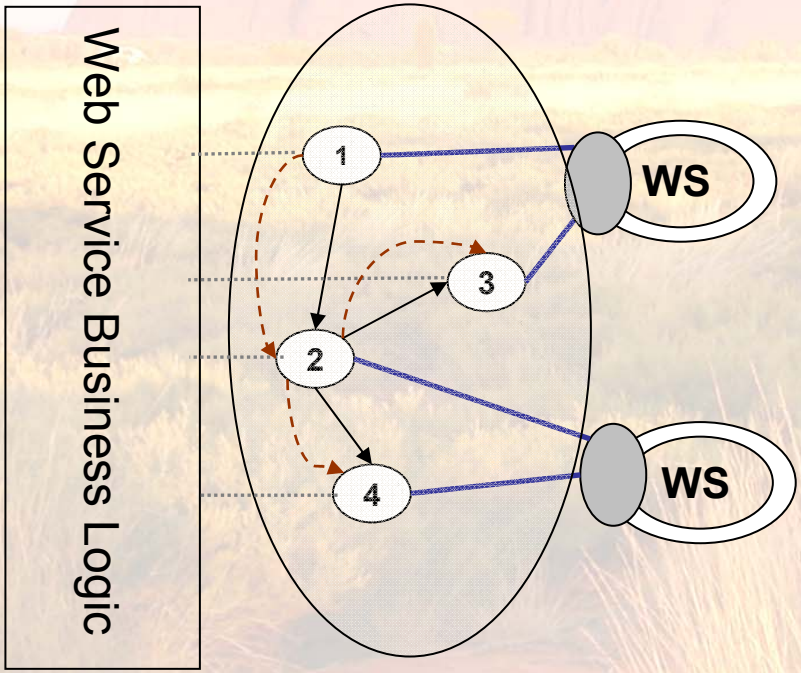
if order **then** conf

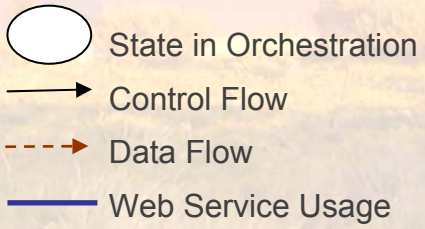
$\Omega_{S2}(\omega2b) =$
{order(in), conf(out)}

existence of a valid Choreography

Orchestration

Interaction with aggregated Web Services + Control Structure





- formally described service functionality decomposition
- only those aspects of WS realization wherefore other WS are aggregated
- aggregated WS used via their behavior interface

Orchestration Description & Validation

- Orchestration Description:
  - interaction behavior of “Orchestrator” with “orchestrated Web Services”
  - formal description as Choreography, extended Guarded Transitions
  - Orchestration Guarded Transitions general structure:

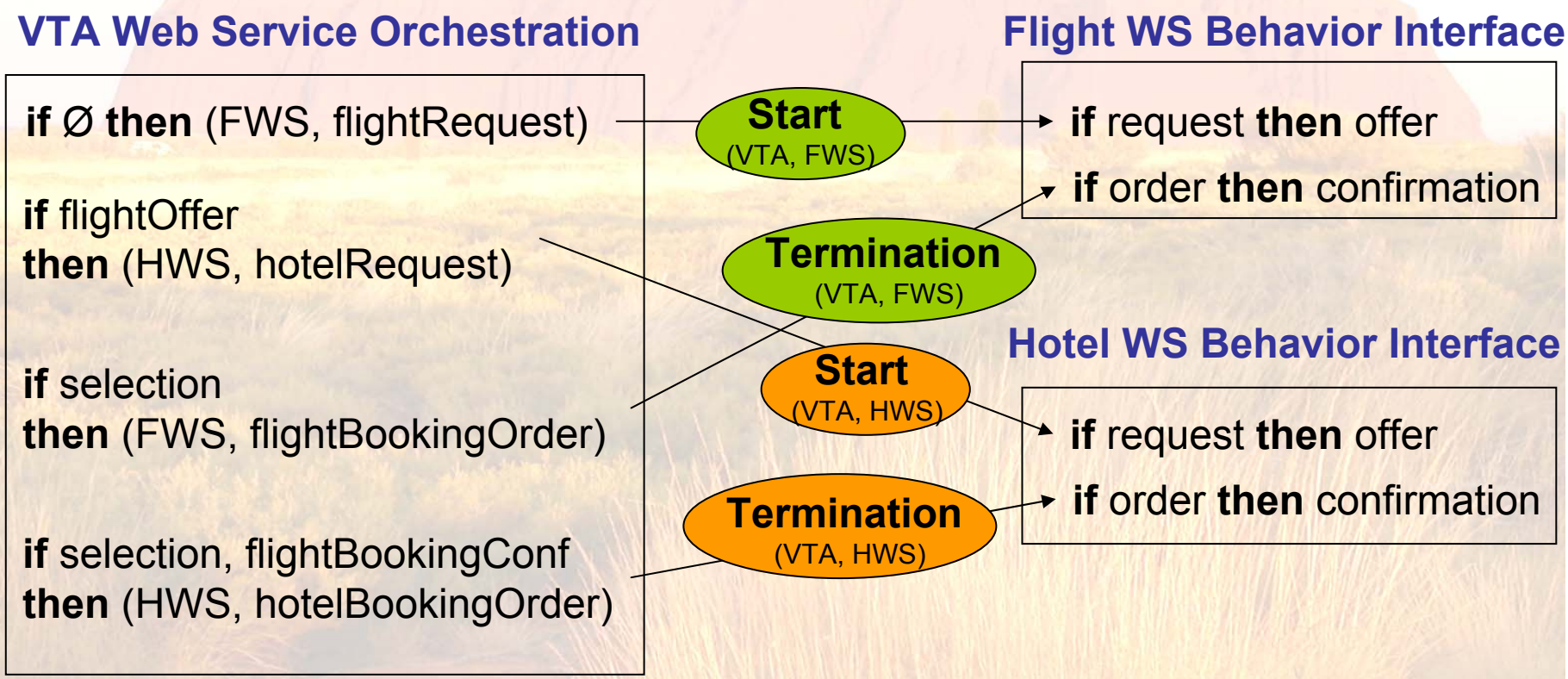
    if condition **then** operation

    Operation = (Orchestrator, Web Service, Action)
  - Orchestrator serves as client for aggregated Web Services

- Orchestration Validation:
  - need to ensure that interactions with aggregated Web Service can be executed successfully

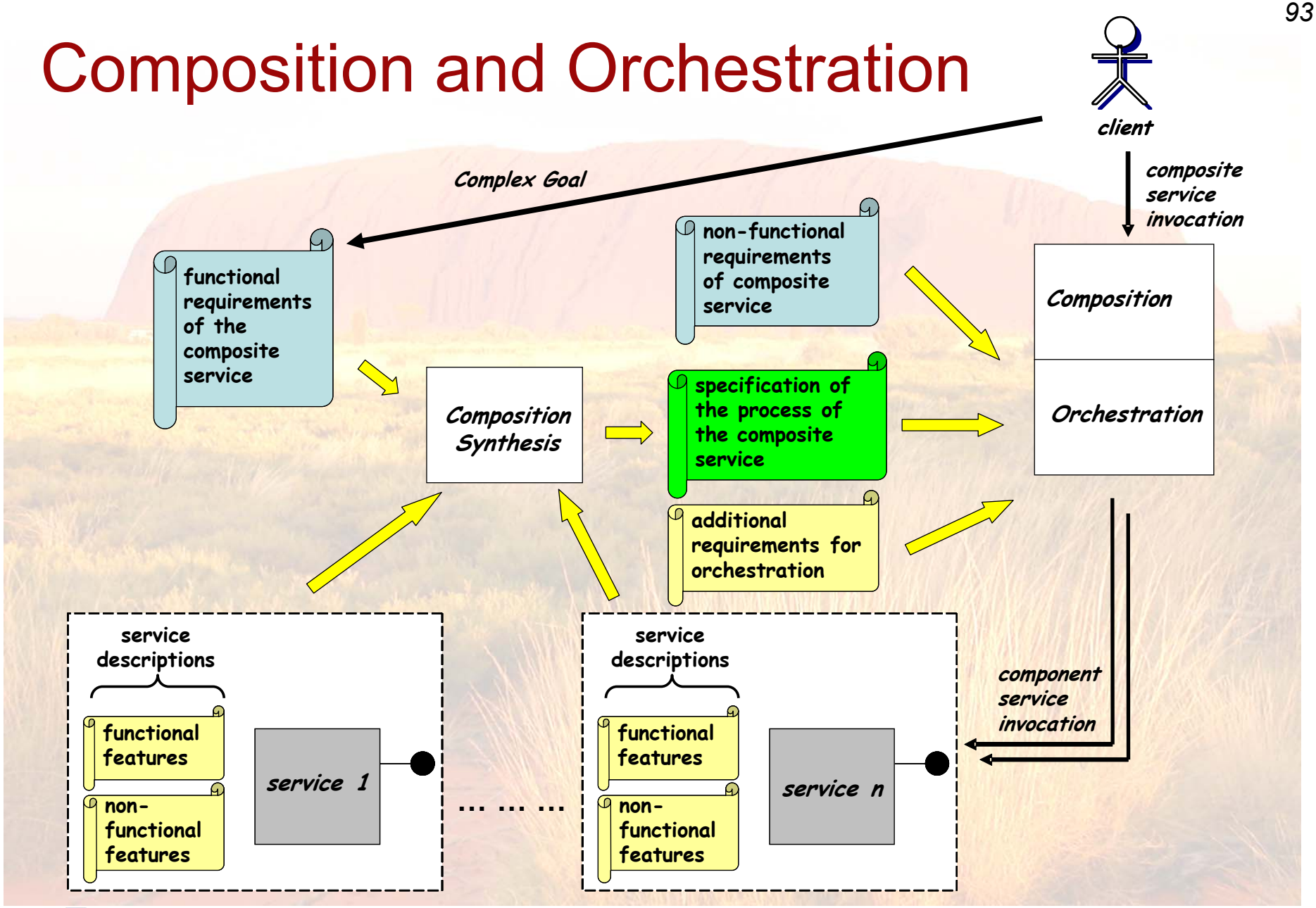
  => Choreography Discovery for all interaction of Orchestrator with each aggregated Web Service

Orchestration Validation Example



Orchestration is valid if valid choreography exists for interactions between Orchestrator and each aggregated Web Service, done by choreography discovery

Composition and Orchestration



Service Composition and Orchestration

- Web Service Composition:
  - the realization of a Web Service by dynamically composing the functionalities of other Web Services
    - The new service is the *composite* service
    - The invoked services are the *component* services
  - a composite service can provide the skeleton for a Web Service (e.g. the VTA Web Service)

- Current Composition techniques only partially cover aspects for valid orchestrations:
  - functional Web Service composition (on capability descriptions)
  - dynamic control and data flow construction for composite Web Service
  - delegation of client / goal behavior to component services

=> Orchestration Validation needed to ensure executability of Web Service aggregations

Mediation

- Heterogeneity as inherent characteristic of (Semantic) Web:
  - heterogeneous terminology
  - heterogeneous languages / formalisms
  - heterogeneous communication protocols and business processes
- WSMO identifies Mediators as top level element, i.e. central aspect of Semantic Web Services
  - levels of mediation: data, protocol, processes
  - WSMO Mediator types
- Approach: declarative, generic mismatch resolution
  - classification of possible & resolvable mismatches
  - mediation definition language & mediation patterns
  - execution environment for mappings

Data Level (OO) Mediation

- Related Aspects / Techniques:
  - Ontology Integration (Mapping, Merging, Alignment)
  - Data Lifting & Lowering
  - Transformation between Languages / Formalisms

- Data Level Mismatch Classification
  - Conceptualization Mismatches
    - same domain concepts, but different conceptualization
    - different levels of abstraction
    - different ontological structure

    => *resolution only incl. human intervention*
  - Explication Mismatches
    - mismatches between:

      T (Term used) **D** (definition of concepts), **C** (real world concept)

    => *automated resolution partially possible*

Ontology Mapping Language

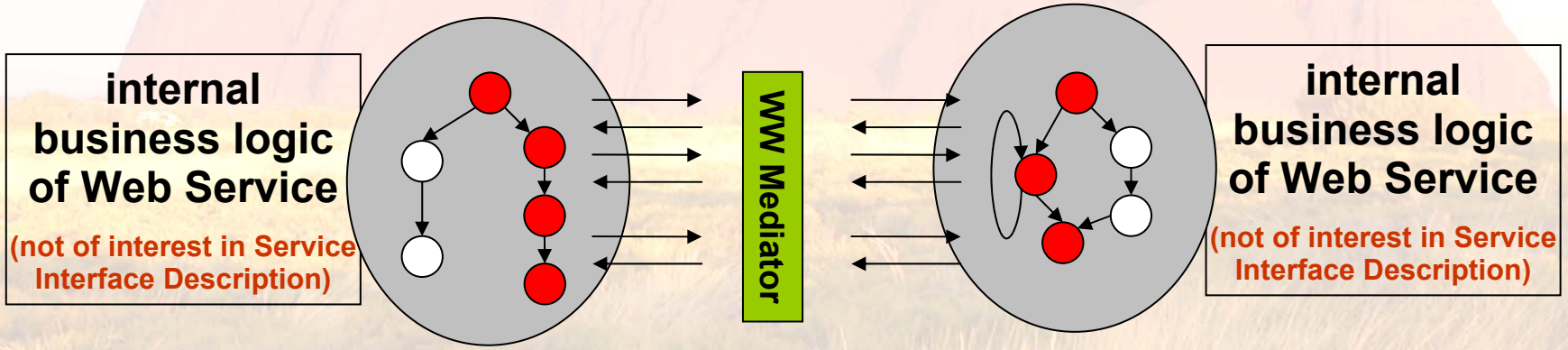
- Language Neutral Mapping Language
  - mapping definitions on meta-layer (i.e. on generic ontological contructs)
  - independent of ontology specification langauge
  - “Grounding” to specific langauges for execution (WSML, OWL, F-Logic)
- Main Features:
  - Mapping Document (sources, mappings, mediation service)
  - direction of mapping (uni- / bidirectional)
  - mapping between Ontology Constructs:
    - classMapping, attributeMapping, relationMapping (between similar constructs)
    - classAtrributeMapping, classRelationMapping, classInstanceMapping
    - instanceMapping (explicit ontology instance transformation)
  - Conditions / logical expressions for data type mismatch handling, restriction of mapping validity, and complex mapping definitions
  - Mapping operators:
    - =, <, <=, >, >=, and, or, not
    - inverse, symmetric, transitive, reflexive
    - join, split

Mapping Language Example

classMapping(**unidirectional** o2:Person o1.Adult
attributeValueCondition(o2.Person.age **>=** 18))

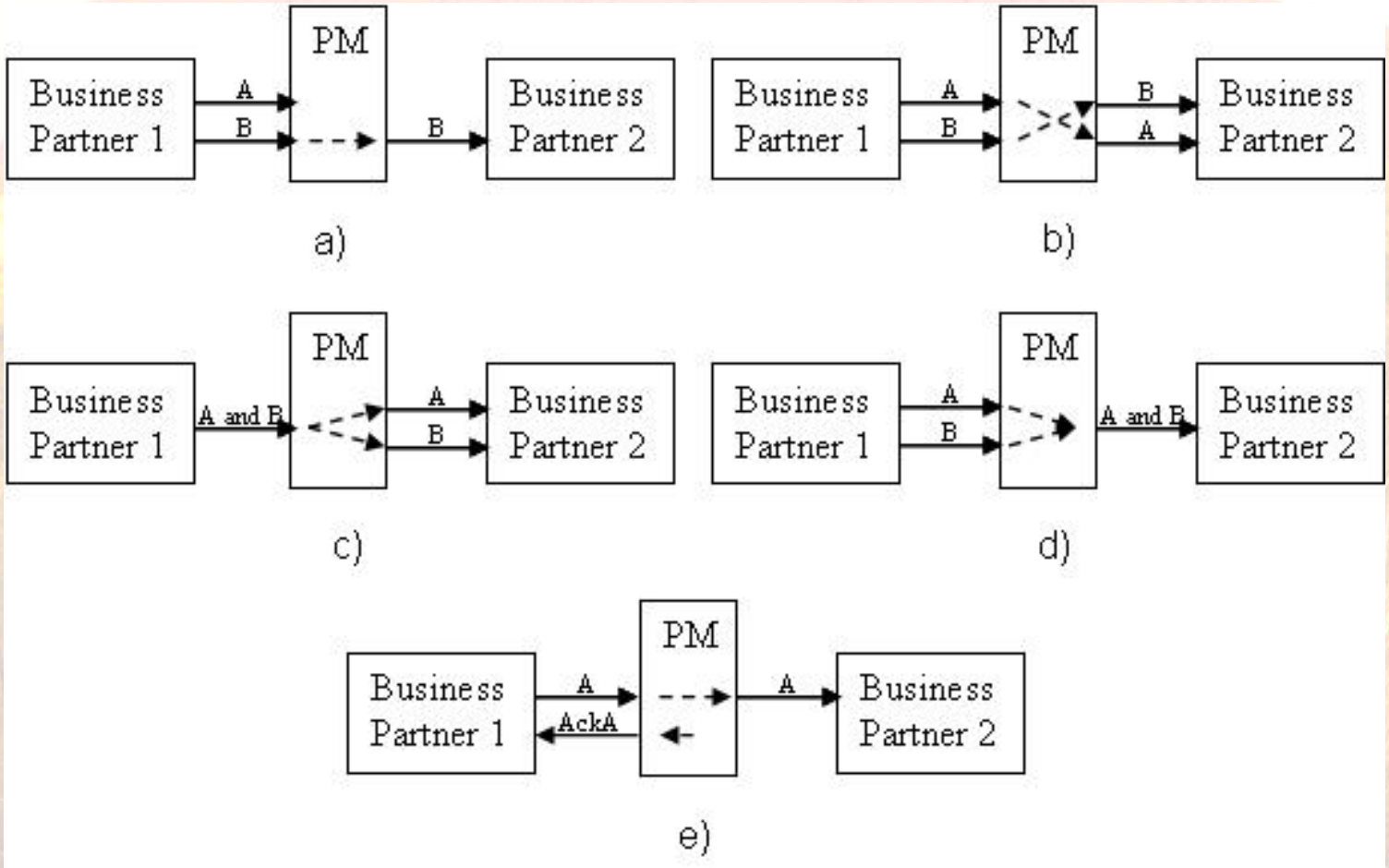
this allows to transform the instance 1234 of ontology O2 into a valid instance of ‘adult’ in ontology O1

Protocol & Process Level Mediation

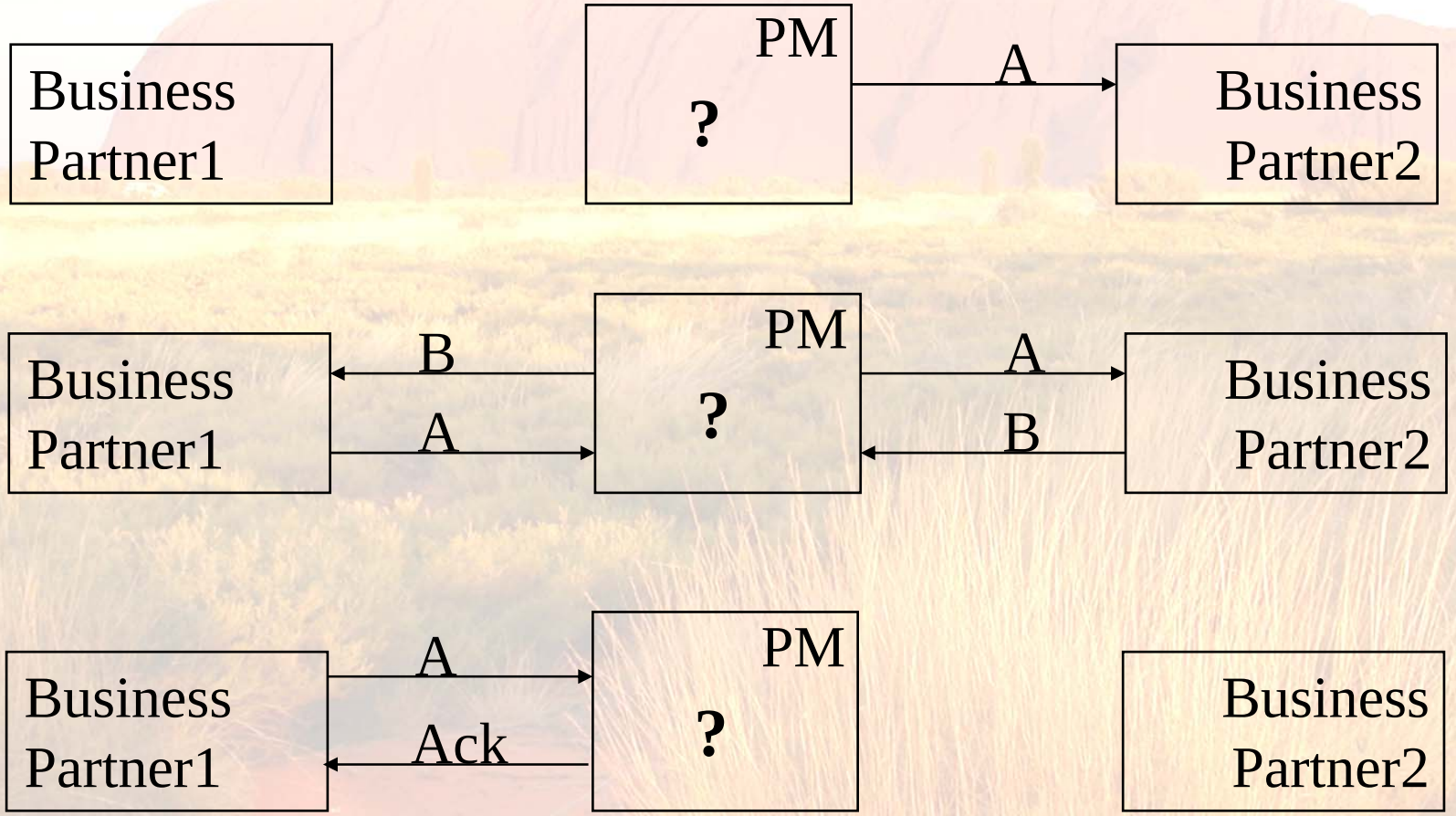


- if a choreography does not exist, then find an appropriate WW Mediator that
  - resolves possible mismatches to establish Information Compatibility (OO Mediator usage)
  - resolves process / protocol level mismatches in to establish Communication Compatibility

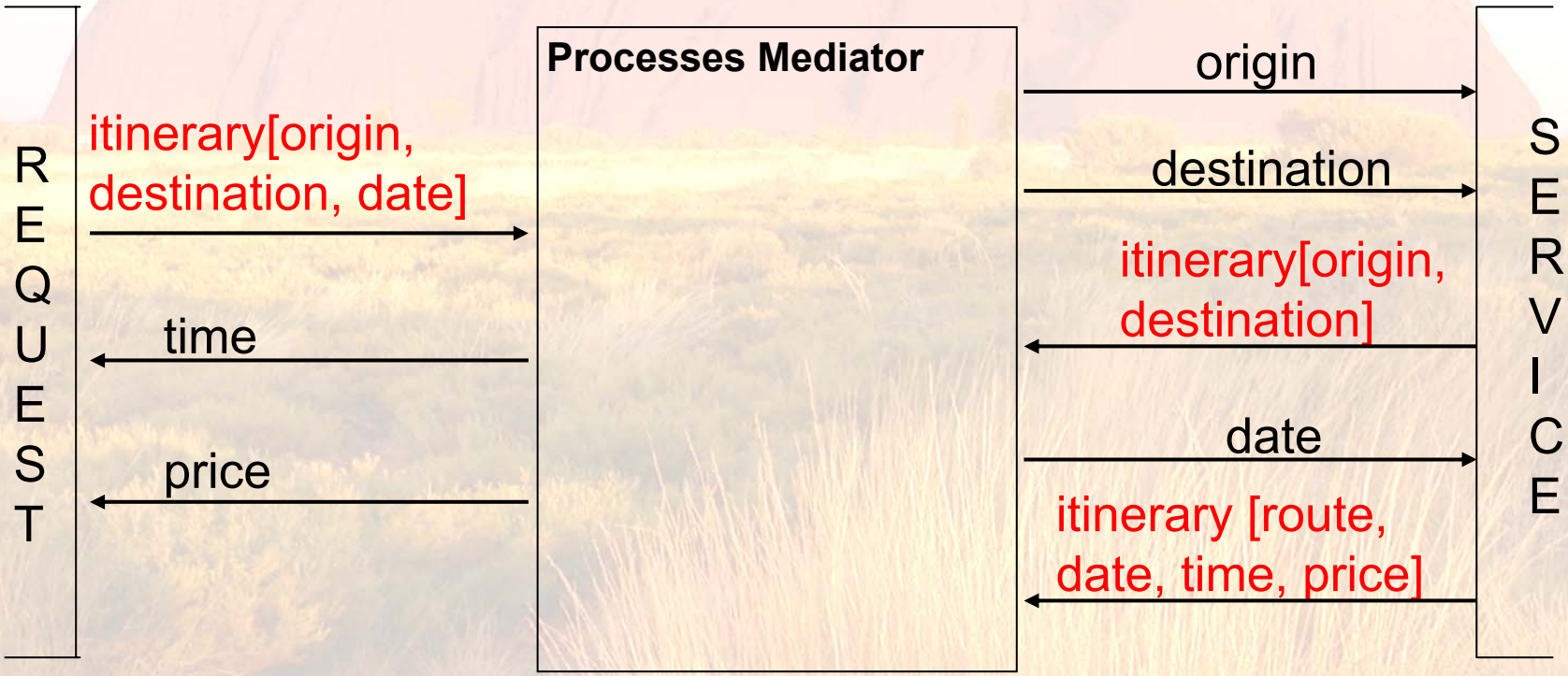
Process Mediation – Addressed Mismatches



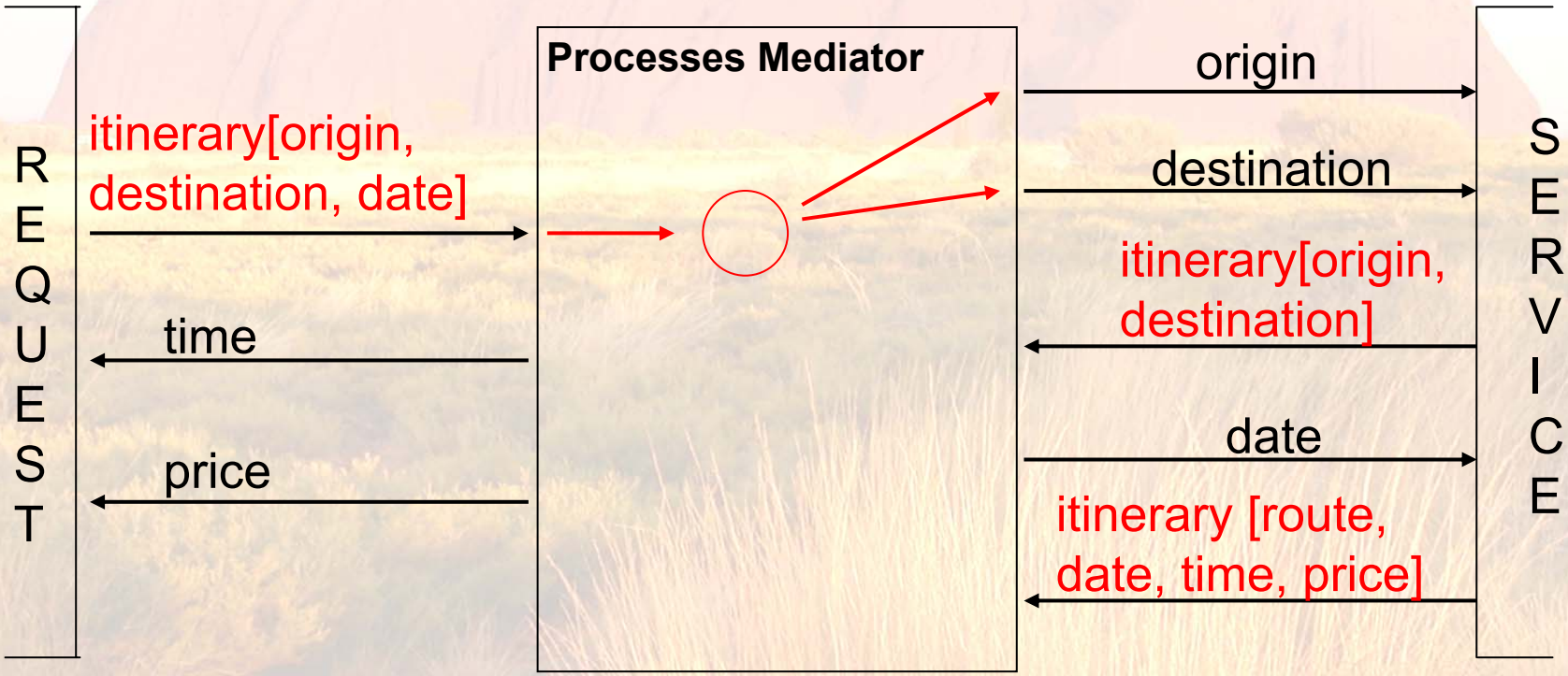
Unsolvable Mismatches



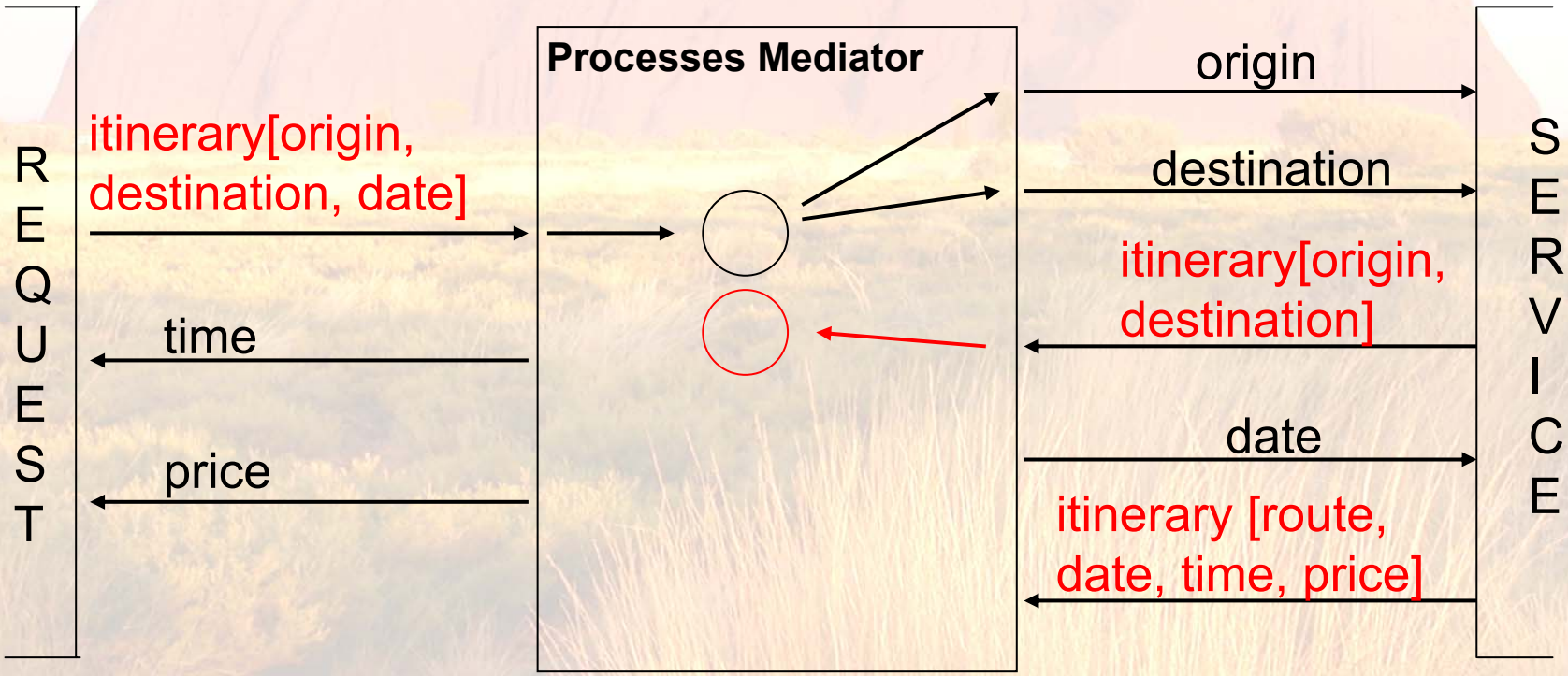
Process Mediation Example



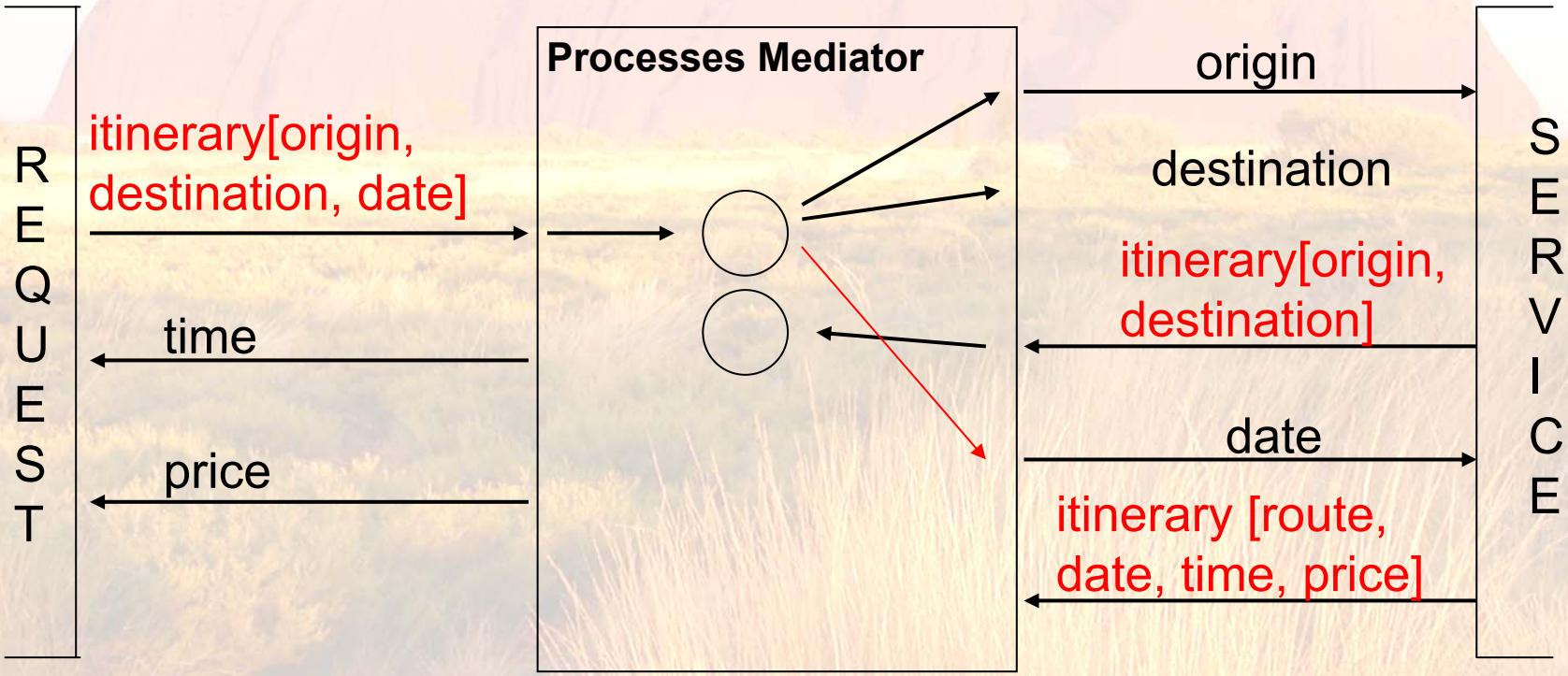
Process Mediation Example



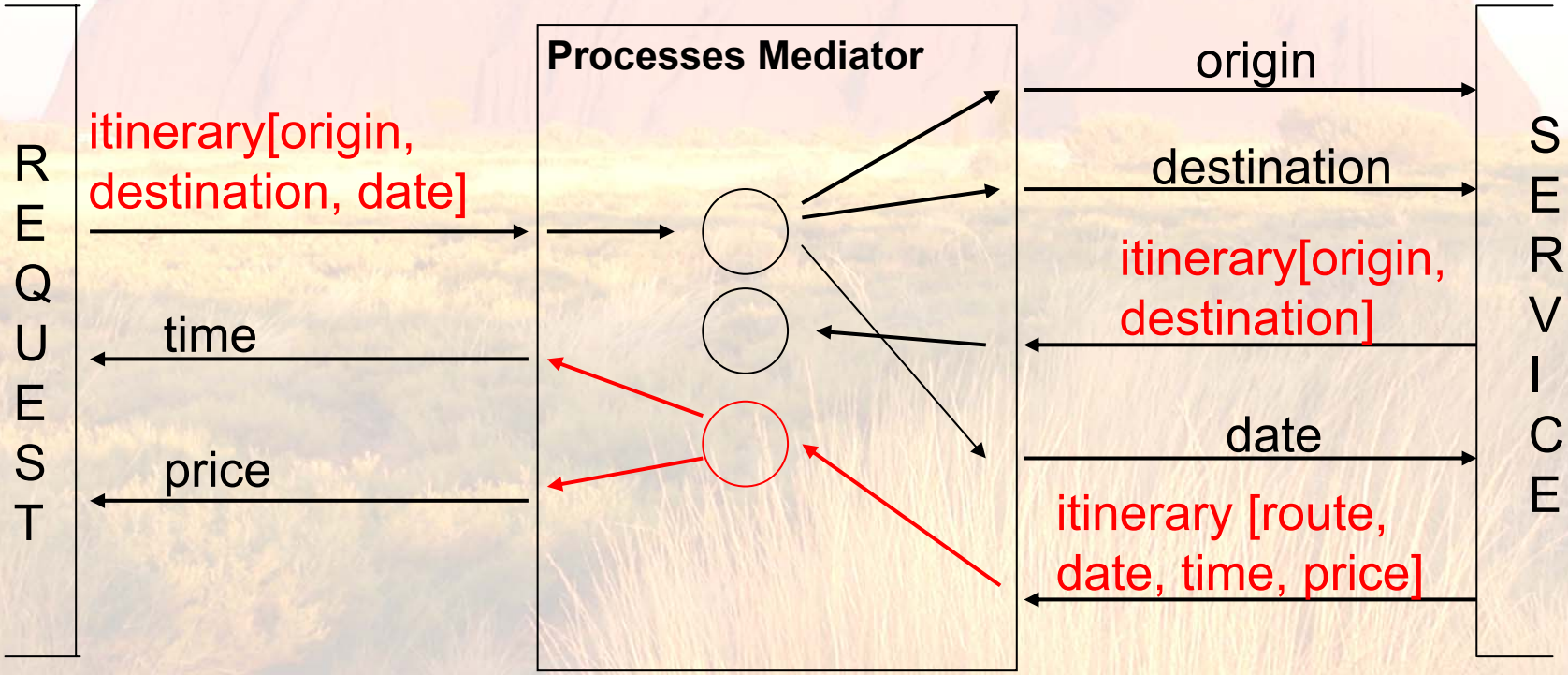
Process Mediation Example



Process Mediation Example



Process Mediation Example



Conclusions

- Semantic Web Service descriptions require
  - expertise in ontology & logical modeling

  => tool support for users & developers under development
  - understanding of Semantic Web Service technologies
    - what it does, and how it works
    - which are the related descriptive information

- Semantic Web Service technologies aim at automation of the Web Service usage process
  - users only define goal with tool support
  - 'intelligent' SWS middleware for automated Web Service usage

- state of the art in technology & tool development
  - theoretical approaches are evolving & converging
  - prototypical SWS technologies existent
  - industrial strength SWS technology suites aspired in upcoming efforts

PART IV:

The Web Service Execution Environment (WSMX)

- Introduction, background and motivation
- Structural architecture
- Dynamic behaviour
- Future plans
- Demos

Introduction, Background and Motivation

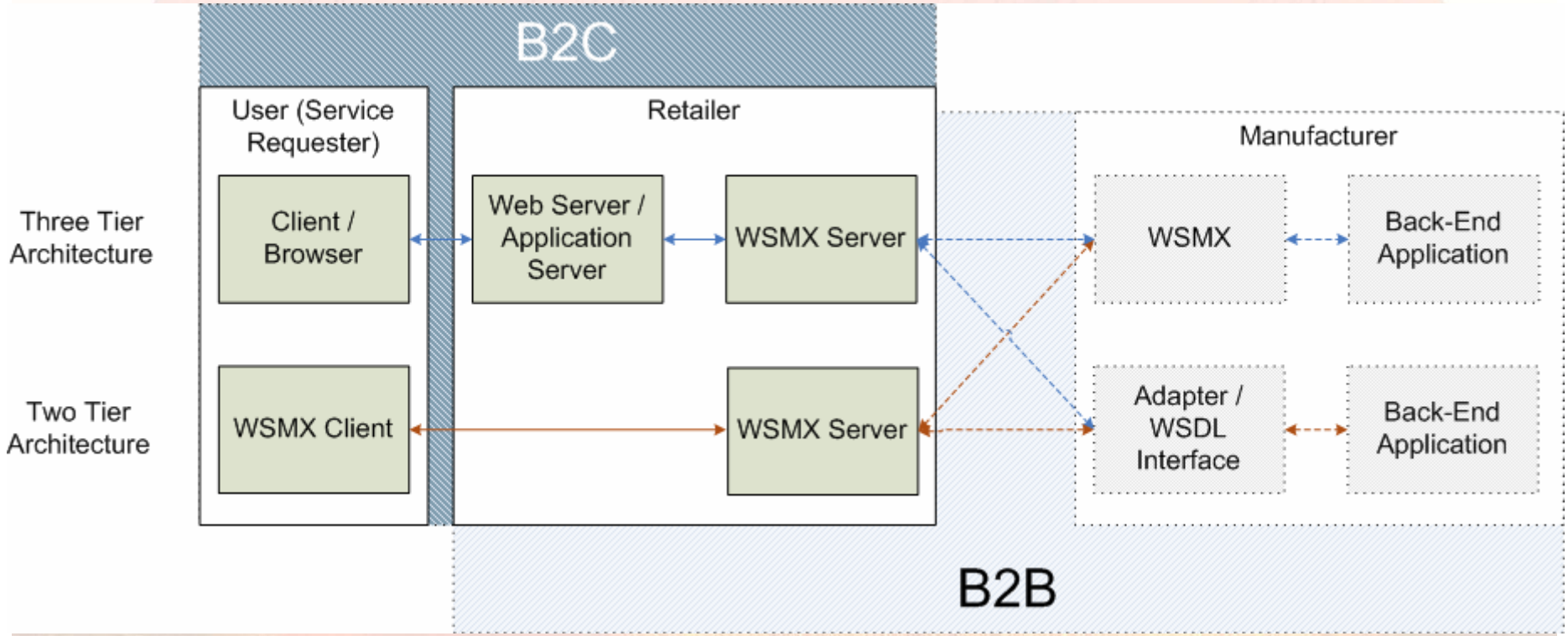
WSMX Introduction

- Software framework for runtime binding of service requesters and service providers
- WSMX interprets service requester's goal to
  - discover matching services
  - select (if desired) the service that best fits
  - provide mediation (if required)
  - make the service invocation
- Is based on the conceptual model provided by WSMO
- Has a formal execution semantics
- SO and event-based architecture based on microkernel design using technologies as J2EE, Hibernate, Spring, JMX, etc.

WSMX Motivation

- Provide **middleware 'glue'** for Semantic Web Services
  - Allow service providers focus on their business
- Provide a **reference implementation** for WSMO
  - Eat our own cake
- Provide an environment for **goal based** service discovery and invocation
  - Run-time binding of service requester and provider
- Provide a flexible **Service Oriented Architecture**
  - Add, update, remove components at run-time as needed
- Keep **open-source** to encourage participation
  - Developers are free to use in their own code
- Define **formal execution semantics**
  - Unambiguous model of system behaviour

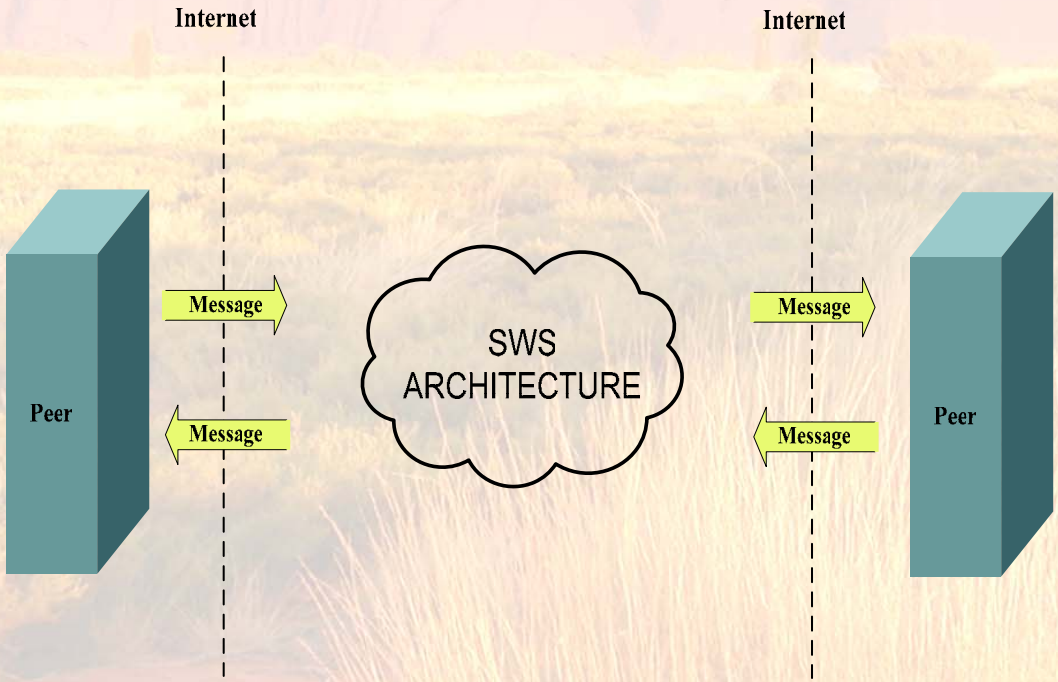
WSMX Usage Scenario



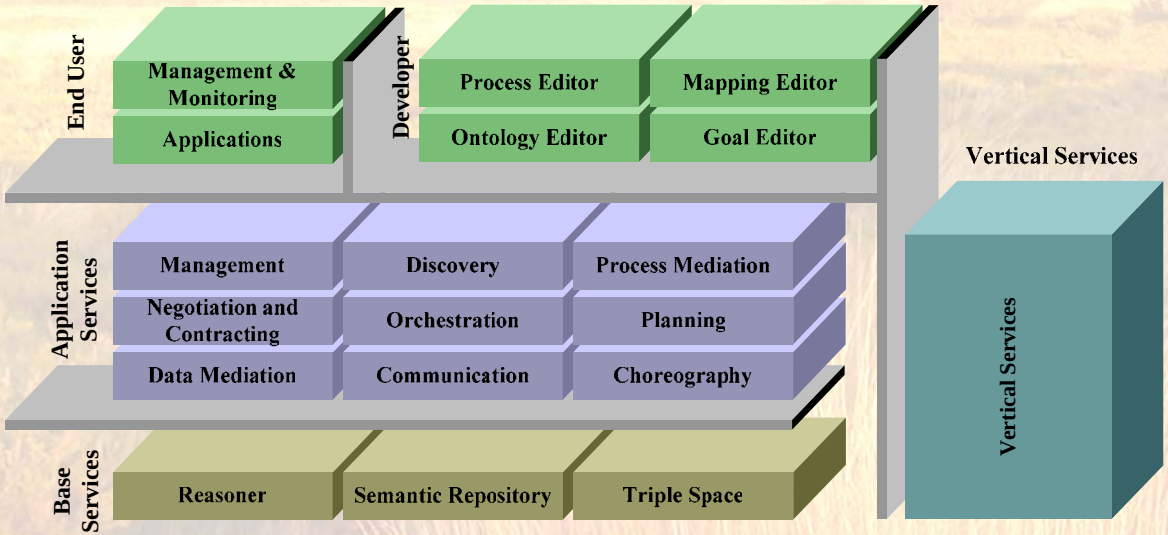
WSMX Usage Scenario - P2P

- A P2P network of WSMX ‘nodes’
- Each WSMX node described as a SWS
- Communication via WSML over SOAP
- Distributed discovery – first aim
- Longer term aim - distributed execution environment

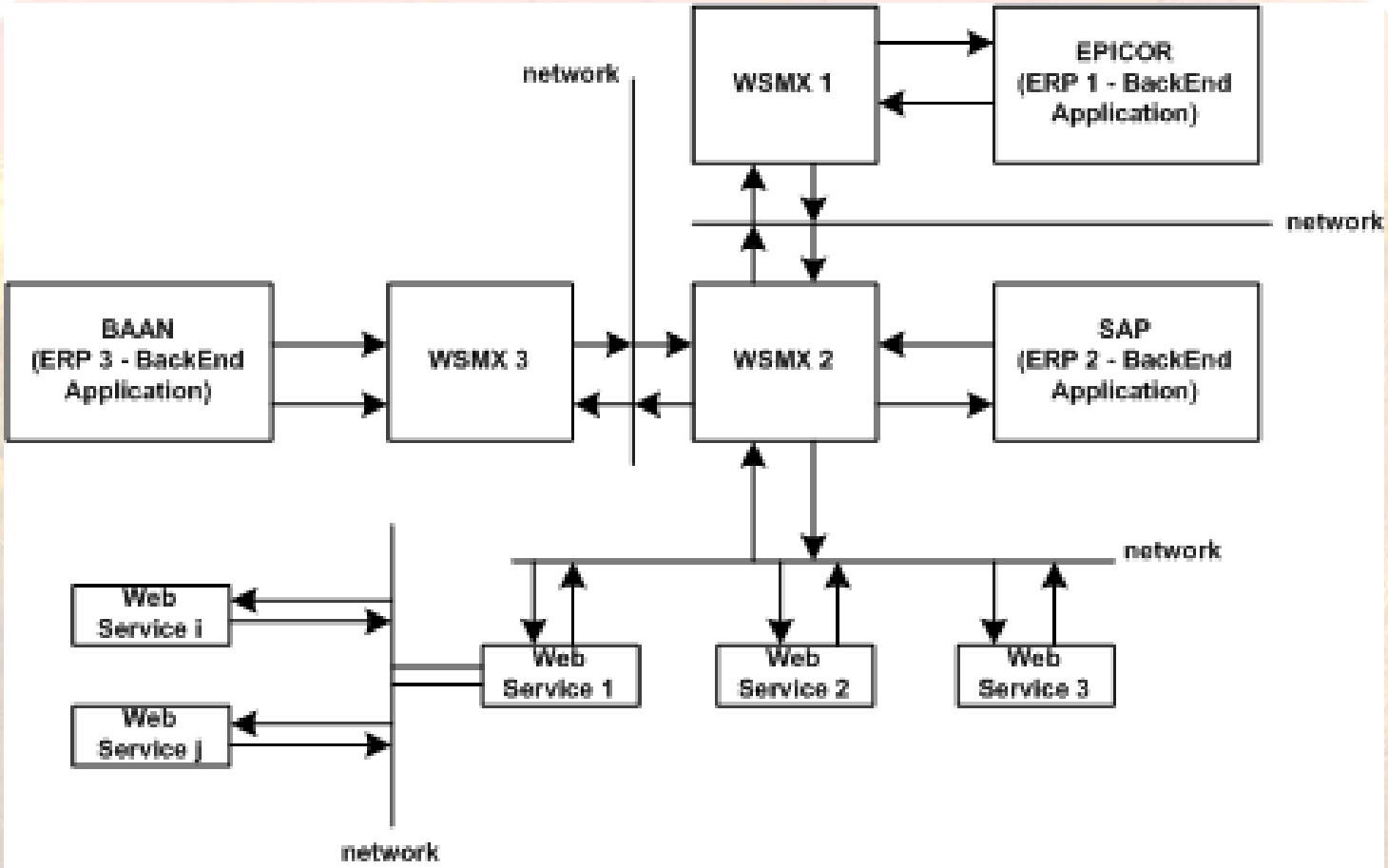
WSMX Usage Scenario - P2P



WSMX Usage Scenario - P2P

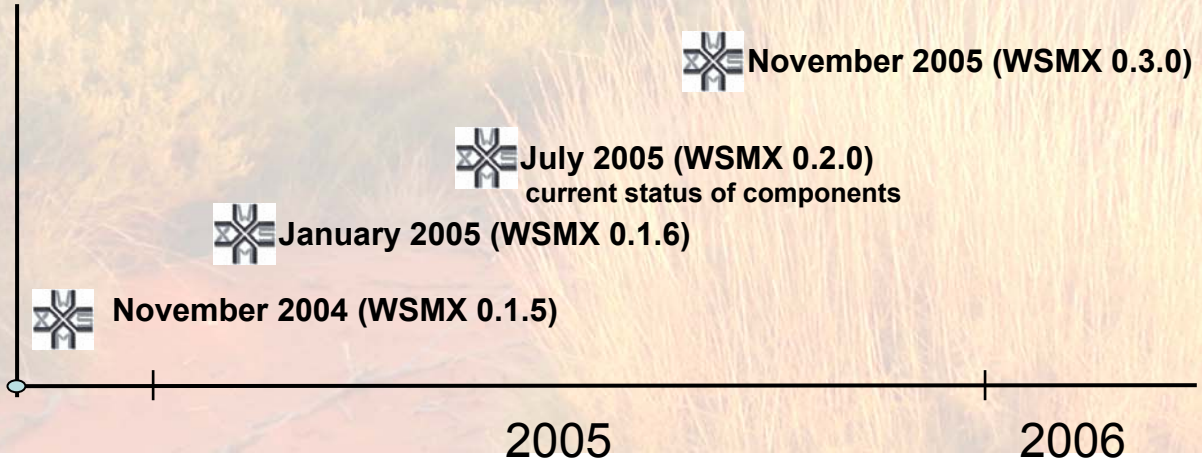


WSMX Usage Scenario - P2P



Development Process & Releases

- The development process for WSMX includes:
  - Establishing its conceptual model
  - Defining its execution semantics
  - Develop the architecture
  - Design the software
  - Building a working implementation
- Planned releases:



Structural Architecture

Design Principles

Strong Decoupling & Strong Mediation

autonomous components with mediators for interoperability

Interface vs. Implementation

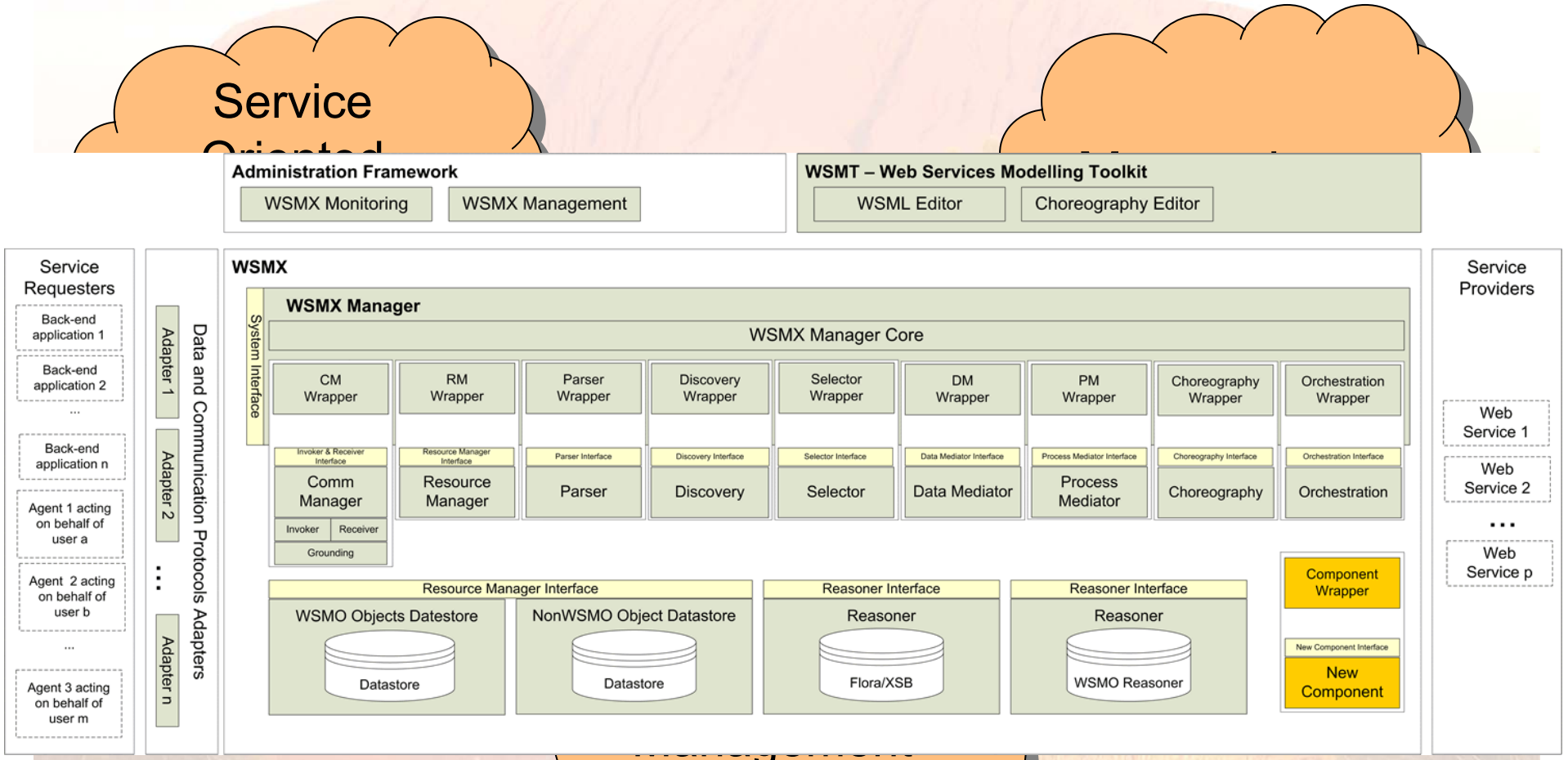
distinguish interface (= description) from implementation (=program)

Peer to Peer

interaction between equal partners (in terms of control)

WSMO Design Principles == WSMX Design Principles == SOA Design Principles

WSMX Architecture



Benefits of SOA

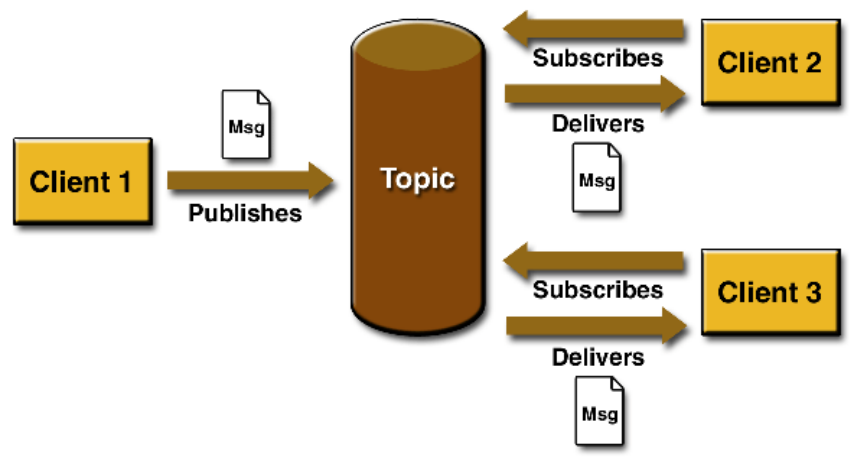
- Better reuse
  - Build new functionality (new execution semantics) on top of existing Business Services
- Well defined interfaces
  - Manage changes without affecting the Core System
- Easier Maintainability
  - Changes/Versions are not all-or-nothing
- Better Flexibility

Service Oriented State

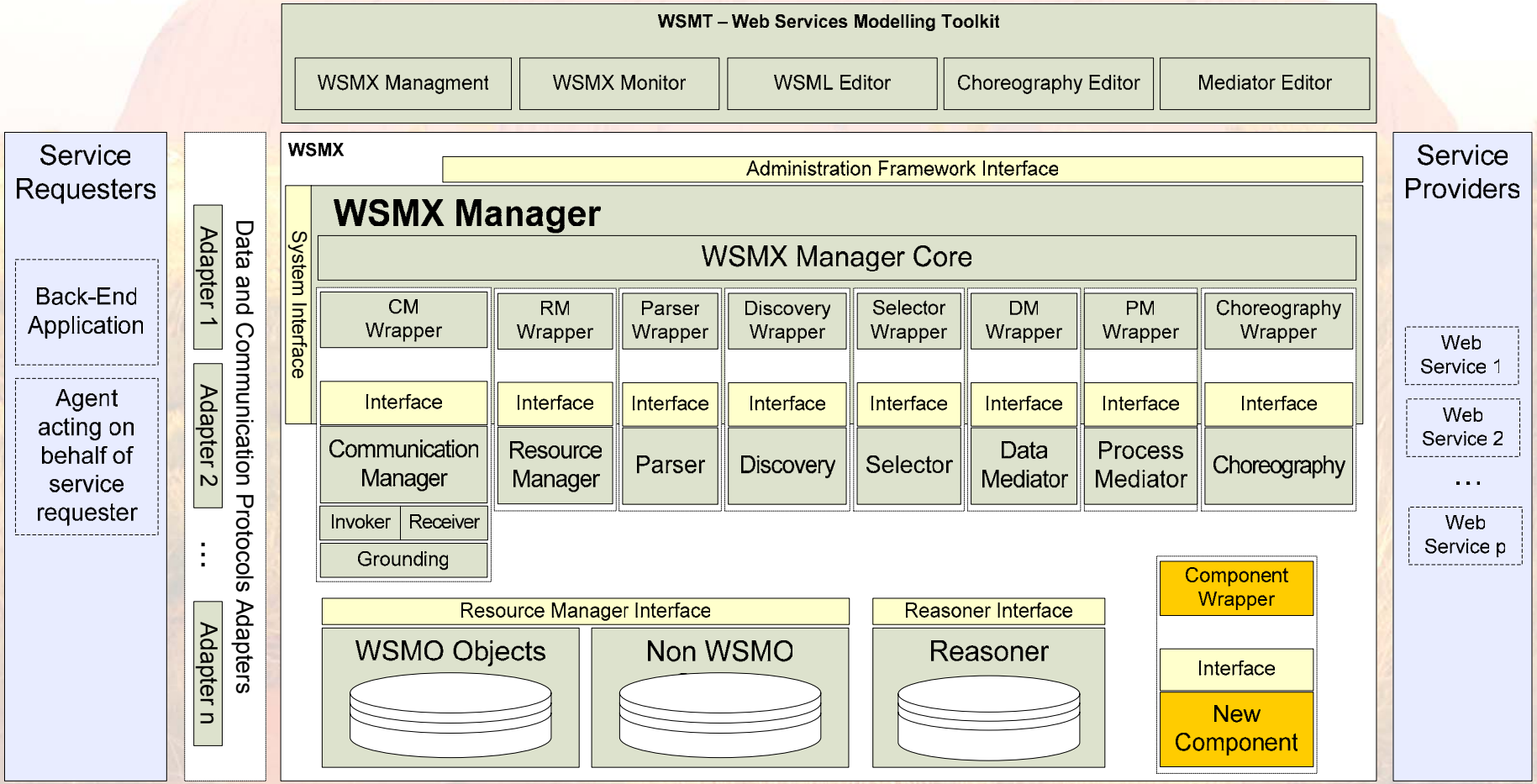
- The interface to the service is implementation-independent
- The service can be dynamically invoked
  - Runtime binding
- The service is self-contained
  - Maintains its own state

Messaging

- Messaging is peer-to-peer facility
- Distributed communication
  - Loosely coupled
- Sender does not need to know receiver (and vice versa)
- Asynchronous mechanism to communicate between software applications



Components & System Architecture



Selected Components

- Adapters
- Parser
- Invoker
- Choreography
- Process Mediator
- Discovery
- Data Mediator
- Resource Manager

Adapters

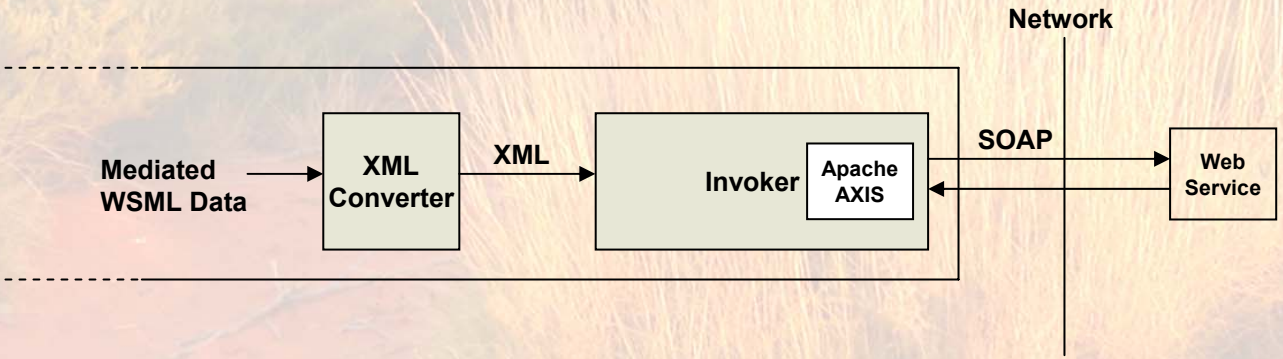
- To overcome data representation mismatches on the communication layer
- Transforms the format of a received message into WSML compliant format
- Based on mapping rules

Parser

- WSML compliant parser
  - Code handed over to wsmo4j initiative http://wsmo4j.sourceforge.net/
- Validates WSML description files
- Compiles WSML description into internal memory model
- Stores WSML description persistently (using Resource Manager)

Communication Mgr – Invoker

- WSMX uses
  - The SOAP implementation from Apache AXIS
  - The Apache Web Service Invocation Framework (WSIF)
- WSMO service descriptions are grounded to WSDL
- Both RPC and Document style invocations possible
- Input parameters for the Web Services are translated from WSML to XML using an additional XML Converter component.



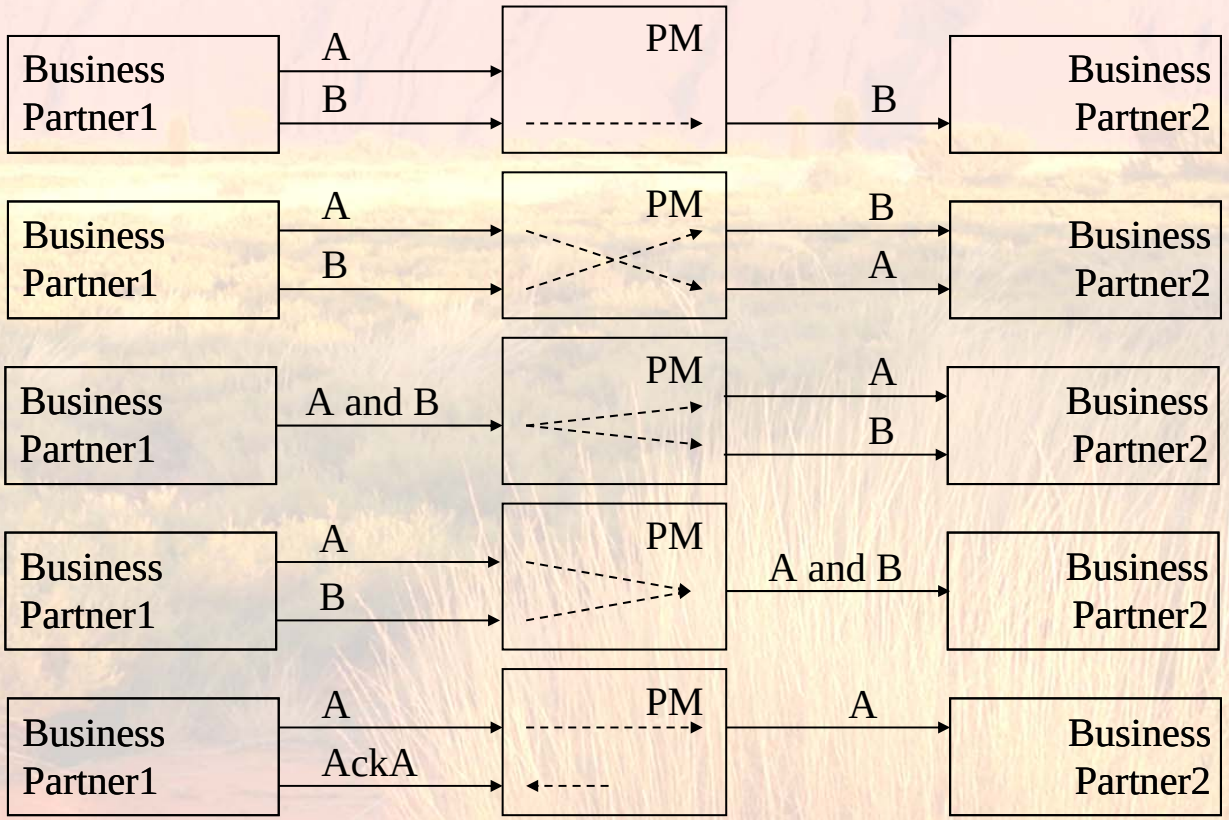
Choreography

- Requester and provider have their own observable communication patterns
  - Choreography part of WSMO
- A choreography instance is loaded for each
  - Both requester and provider have their own WSMO descriptions
- The Choreography component examines a service's choreography to determine next step in communication
- The Choreography component raises events for the Invoker to make actual service invocations

Process Mediator

- Requester and provider have their own communication patterns
- Only if the two match precisely, a direct communication may take place
- At design time equivalences between the choreographies' conceptual descriptions is determined and stored as set of rules
- The Process Mediator provides the means for runtime analyses of two choreography instances and uses mediators to compensate possible mismatches

Process Mediator



Discovery

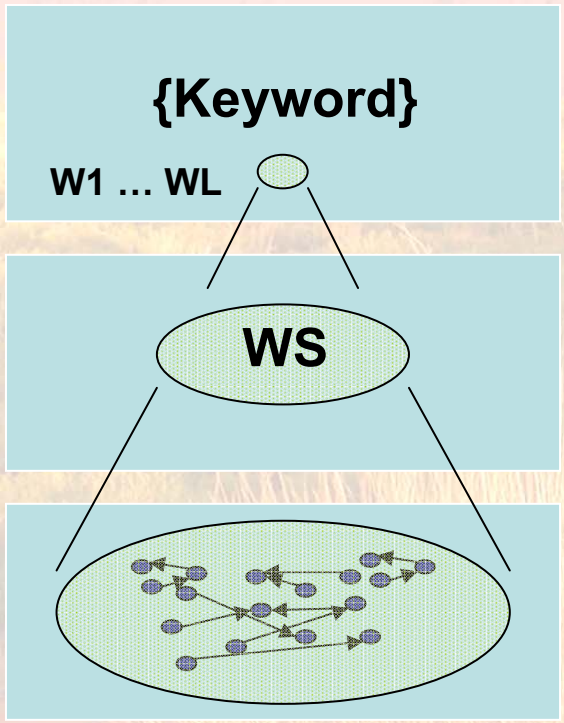
- Responsible for finding appropriate Web Services to achieve a goal (discovery)
- Current discovery component is based on simple matching
- Advanced semantic discovery in prototypical stage

Discovery

Keyword-based with Natural Language Processing (NLP)

Coarse grained Service and Goal descriptions

Fine grained Service and Goal descriptions



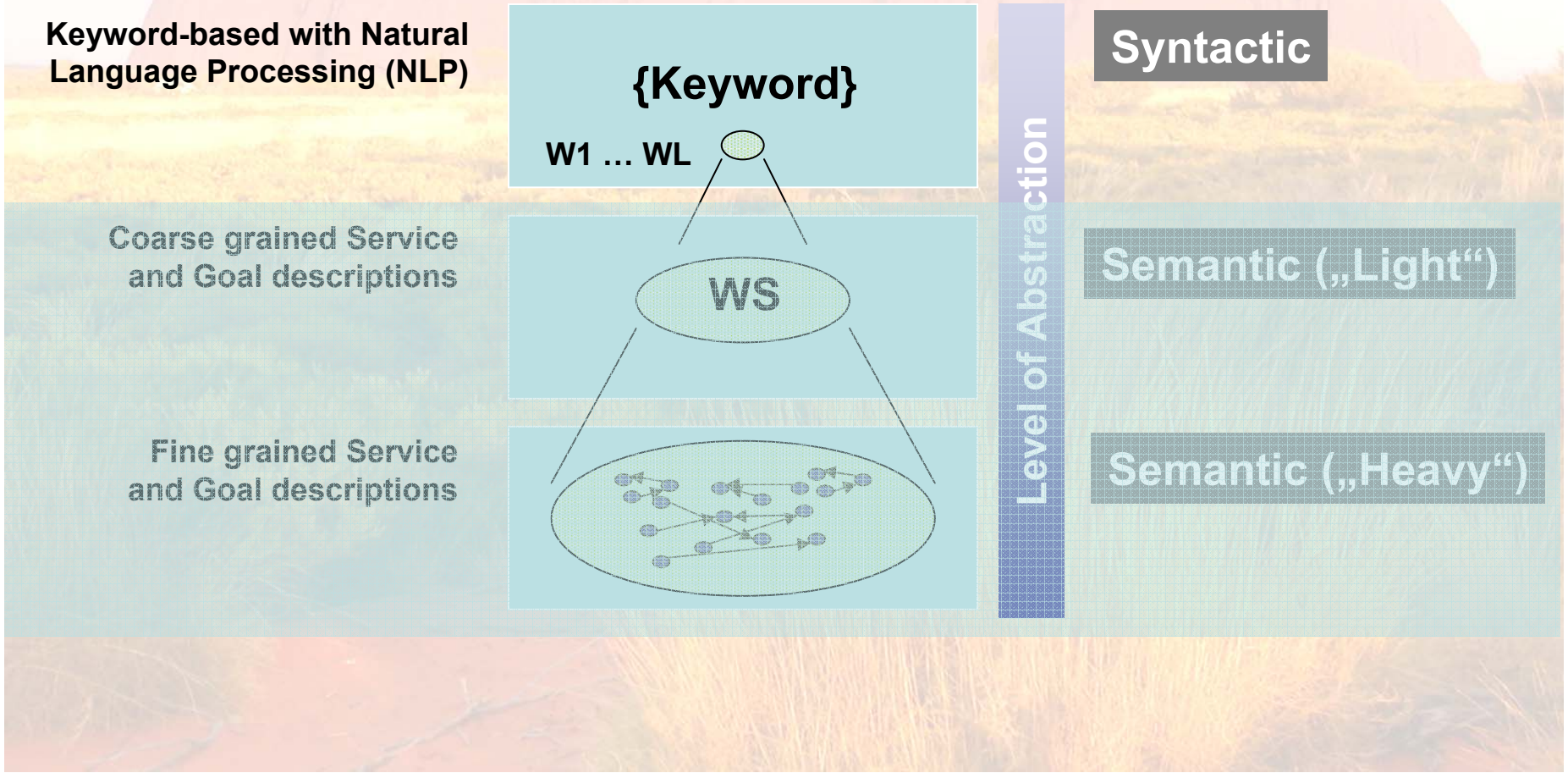
Level of Abstraction

Syntactic

Semantic („Light“)

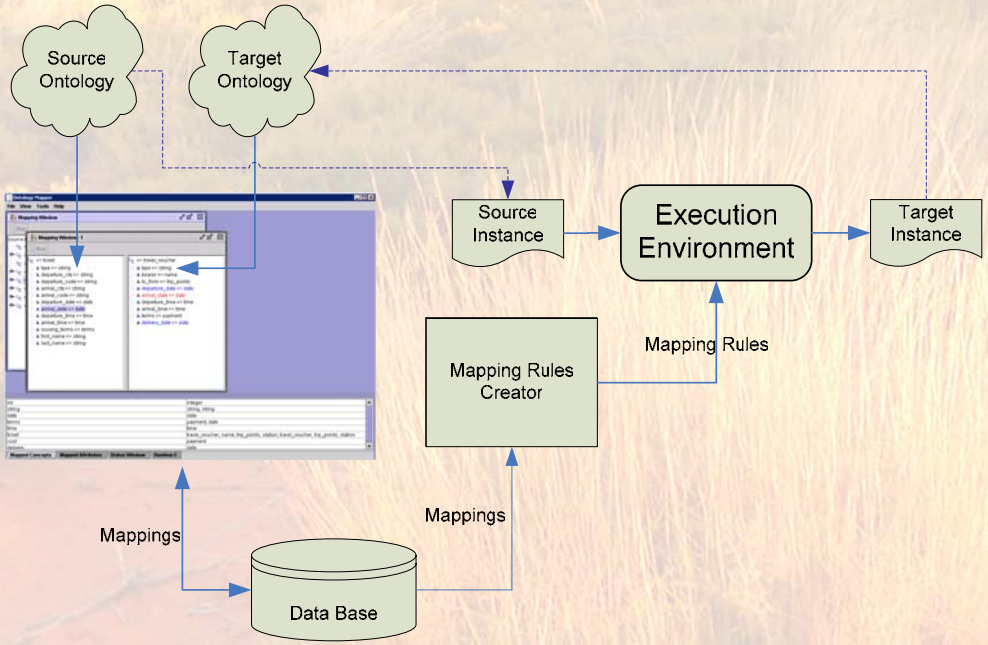
Semantic („Heavy“)

Discovery


Keyword-based with Natural Language Processing (NLP)
{Keyword}
W1 … WL
Coarse grained Service and Goal descriptions
WS
Fine grained Service and Goal descriptions
Level of Abstraction
Syntactic
Semantic („Light“)
Semantic („Heavy“)

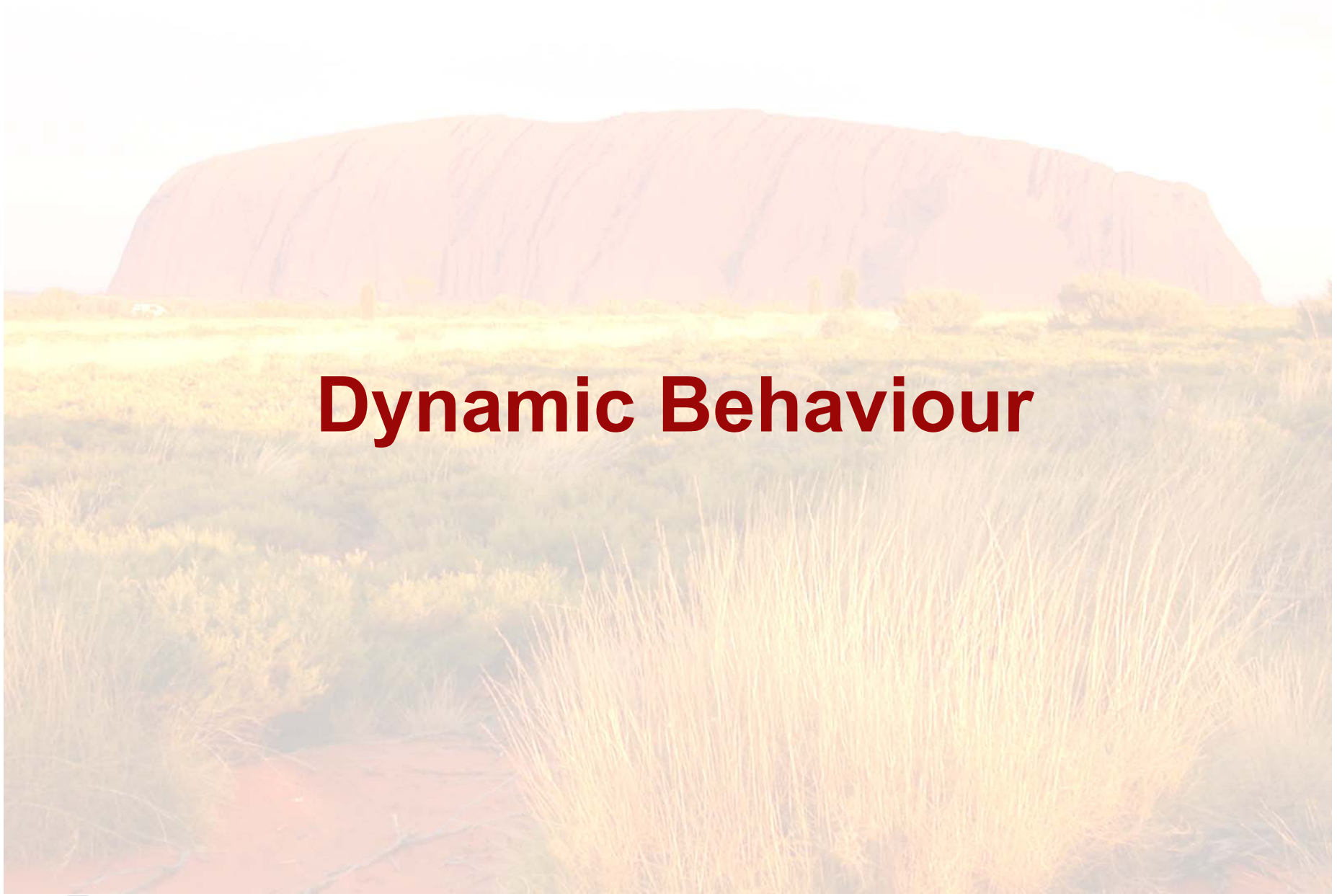
Data Mediator

- Ontology-to-ontology mediation
- A set of mapping rules are defined and then executed
- Initially rules are defined semi-automatic
- Create for each source instance the target instance(s)



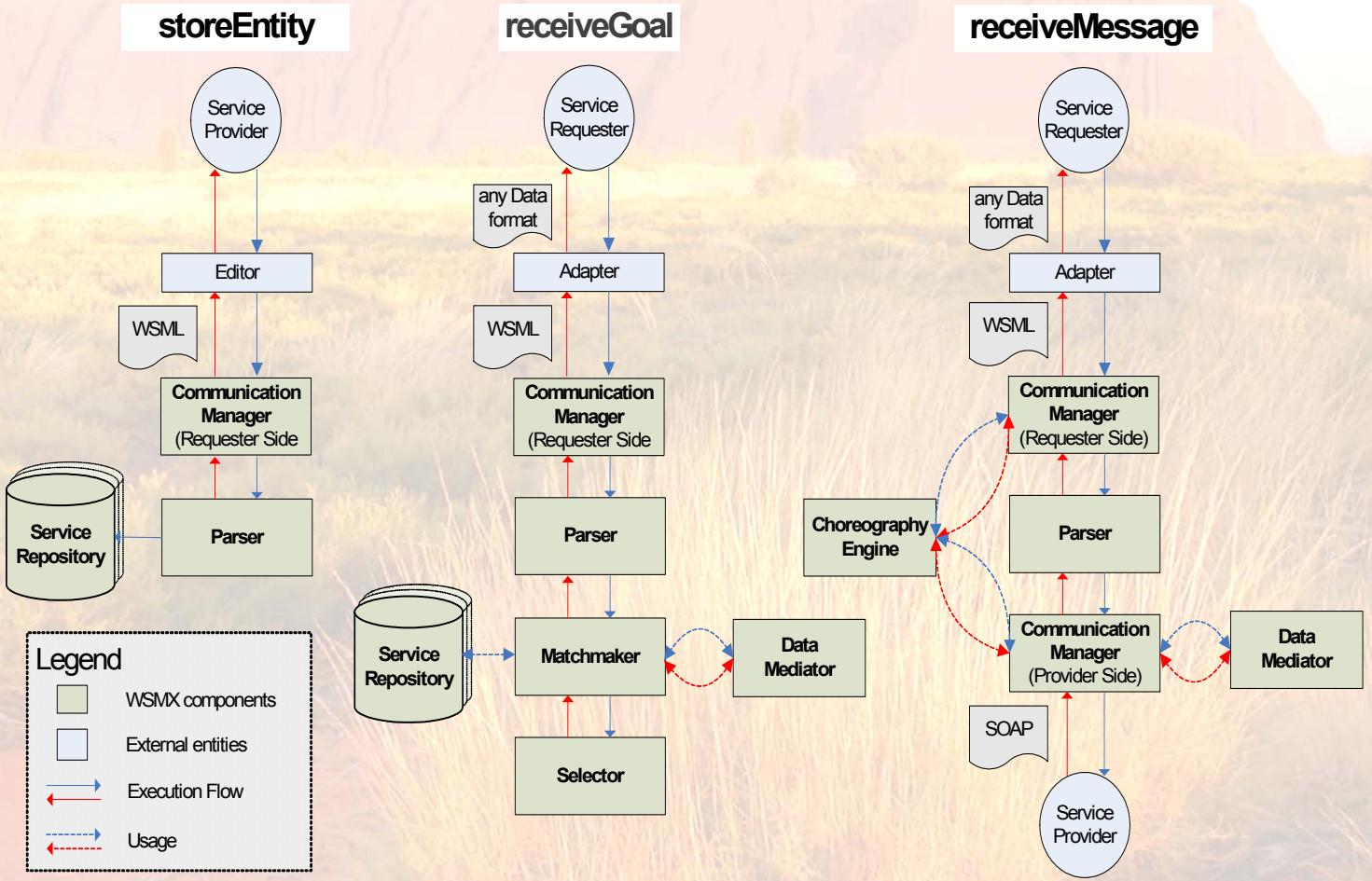
Resource Manager

- Stores internal memory model to a data store
- Decouples storage mechanism from the rest of WSMX
- Data model is compliant to WSMO API
- Independent of any specific data store implementation i.e. database and storage mechanism



Dynamic Behaviour

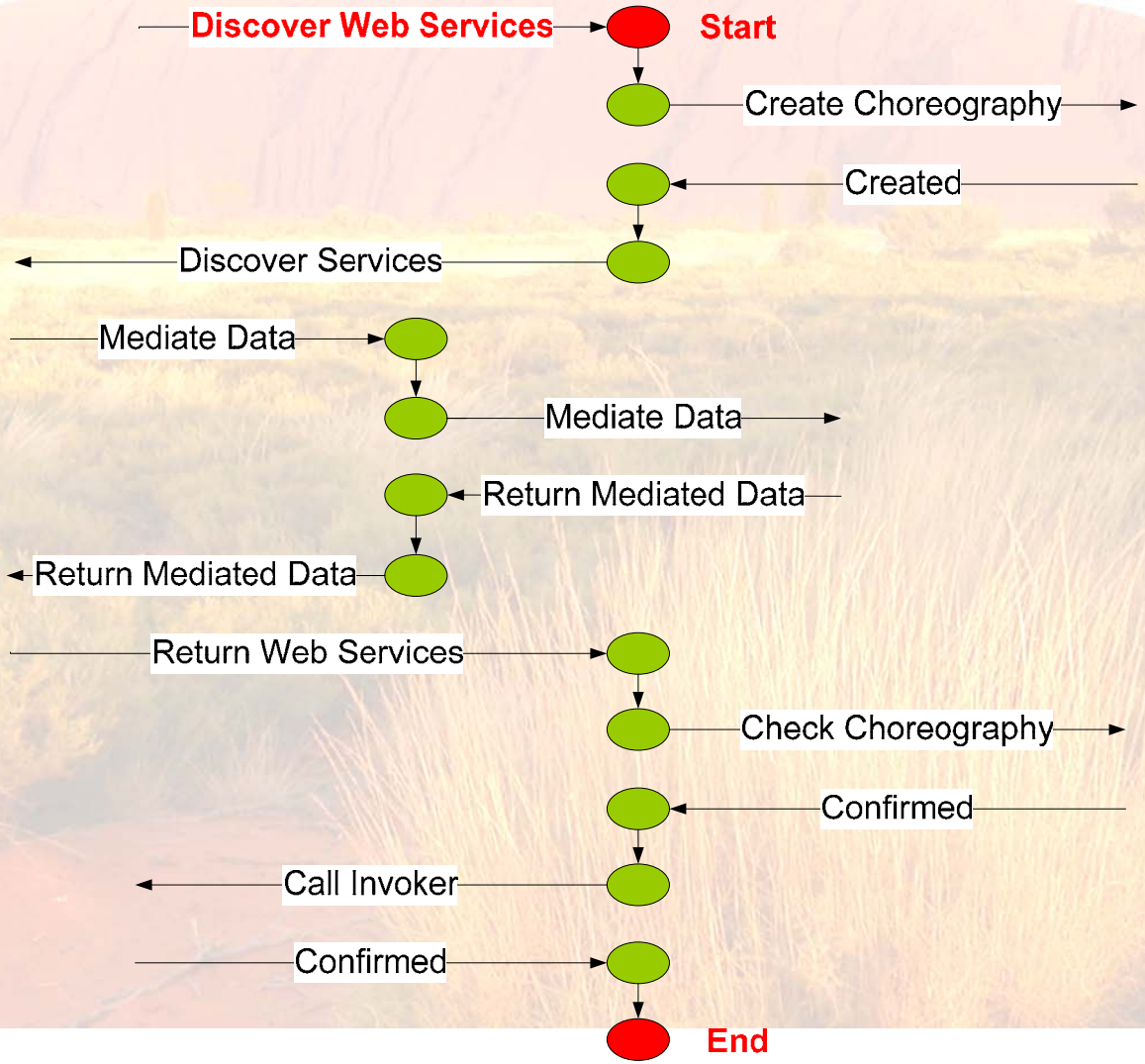
System Entry Points



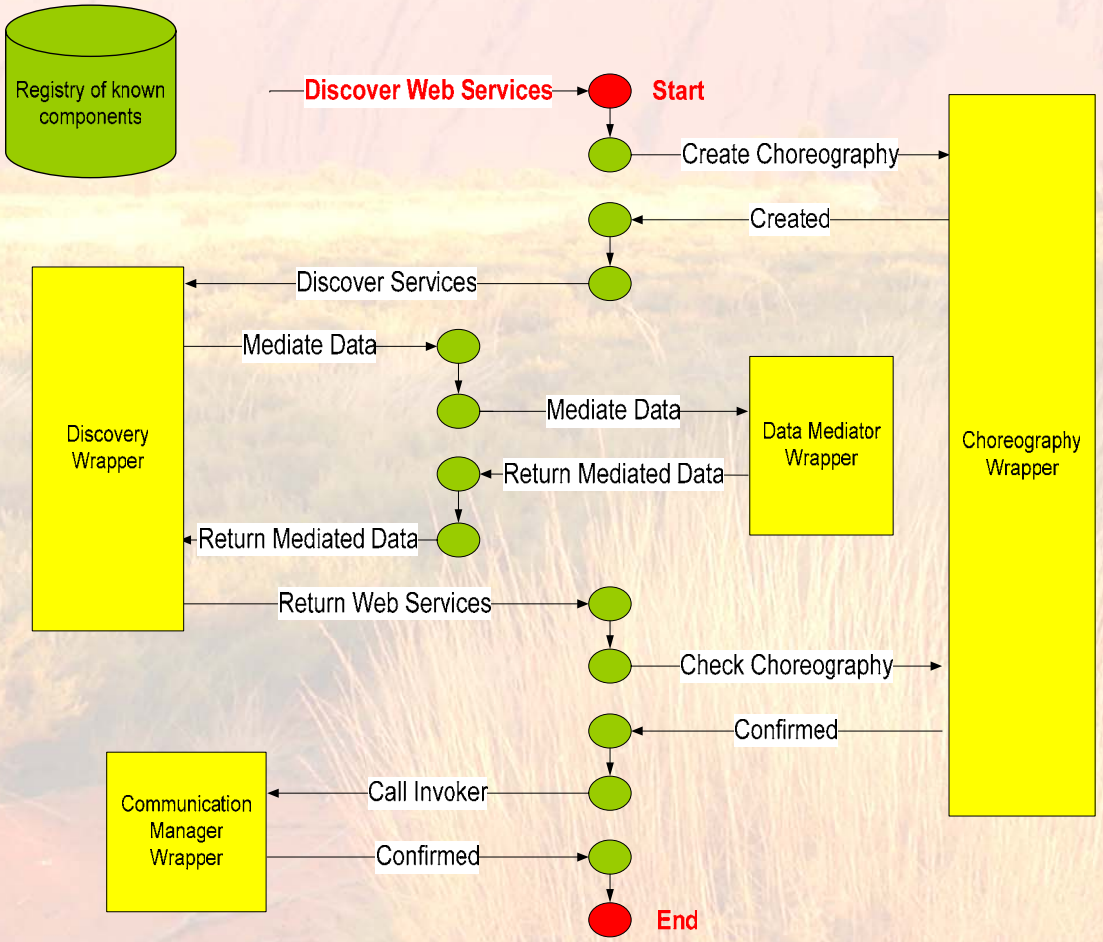
System entry points

- storeEntity(WSMOEntity):Confirmation
  - provides an administration interface for storing any WSMO-related entities (Web Services, Goals, Ontologies)
- realizeGoal(Goal, OntologyInstance):Confirmation
  - service requester expects WSMX to discover and invoke Web Service without exchanging additional messages
- receiveGoal(Goal, OntologyInstance, Preferences):WebService[]
  - list of Web Services is created for given Goal
  - requester can specify the number of Web Services to be returned
- receiveMessage(OntologyInstance,WebServiceID, ChoreographyID):ChoreographyID
  - back-and-forth conversation to provide all necessary data for invocation
  - involves execution of choreographies and process mediation between service interfaces

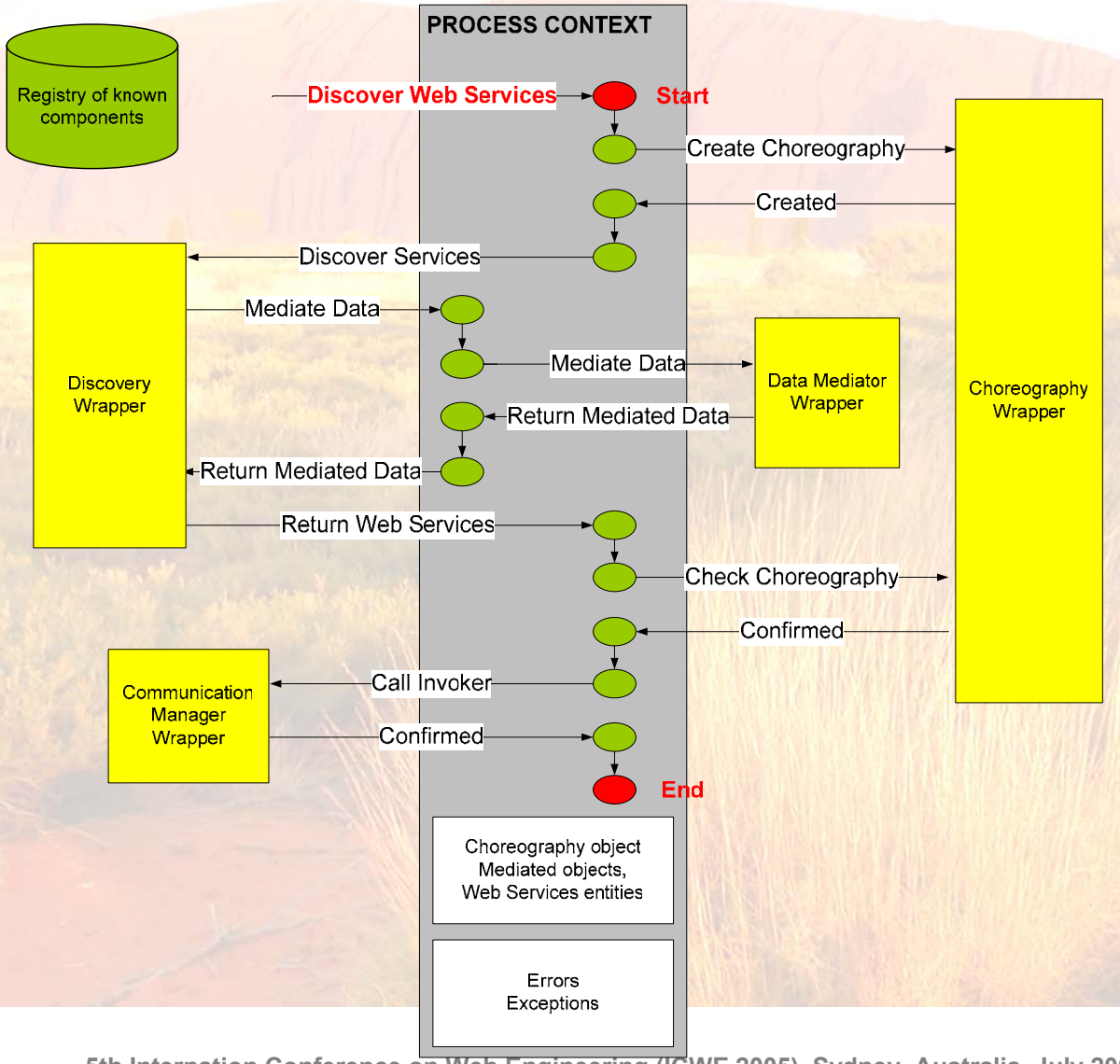
Define “Business” Process



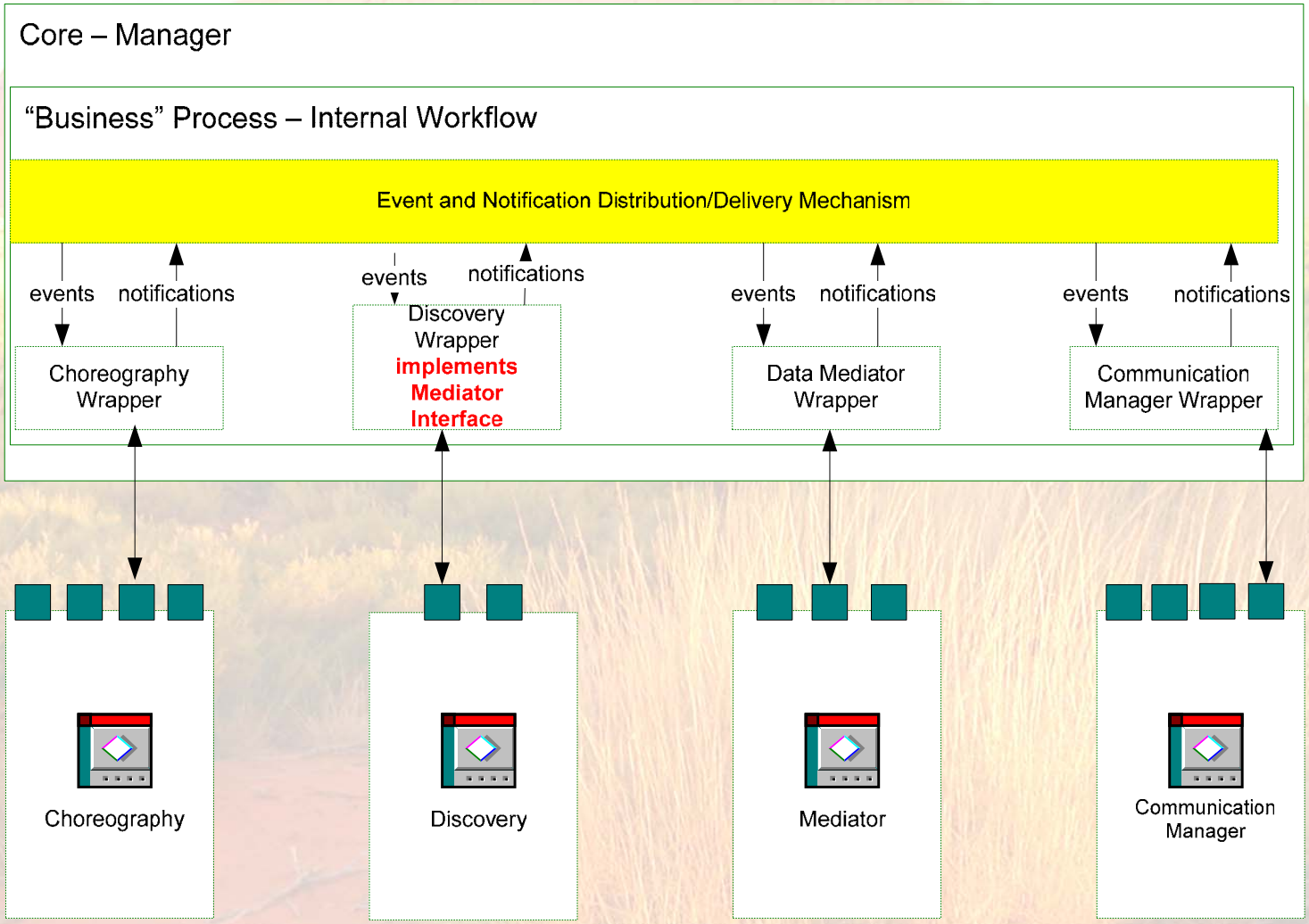
Generate Wrappers for Components



Context Data

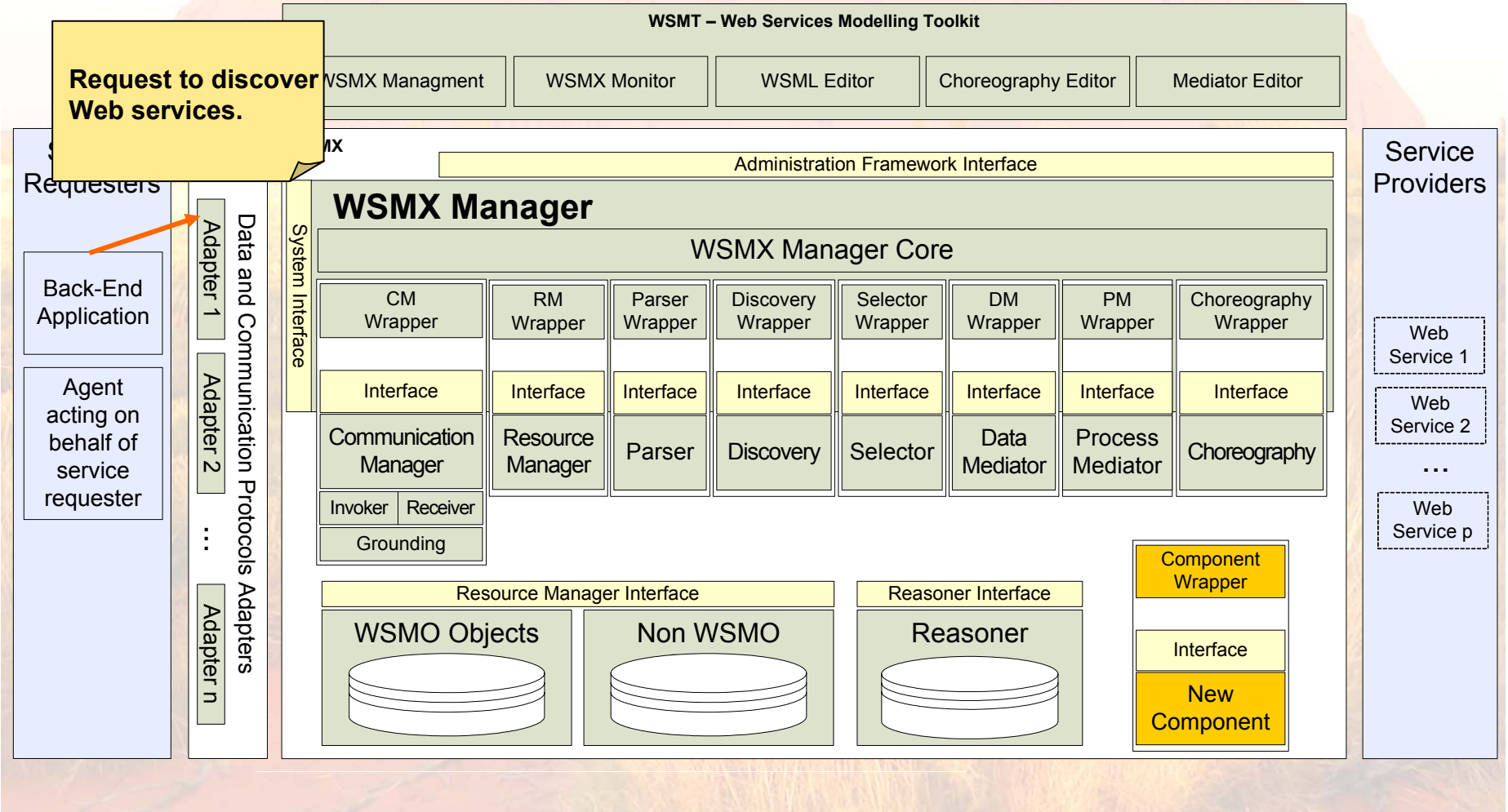


Event-based Implementation

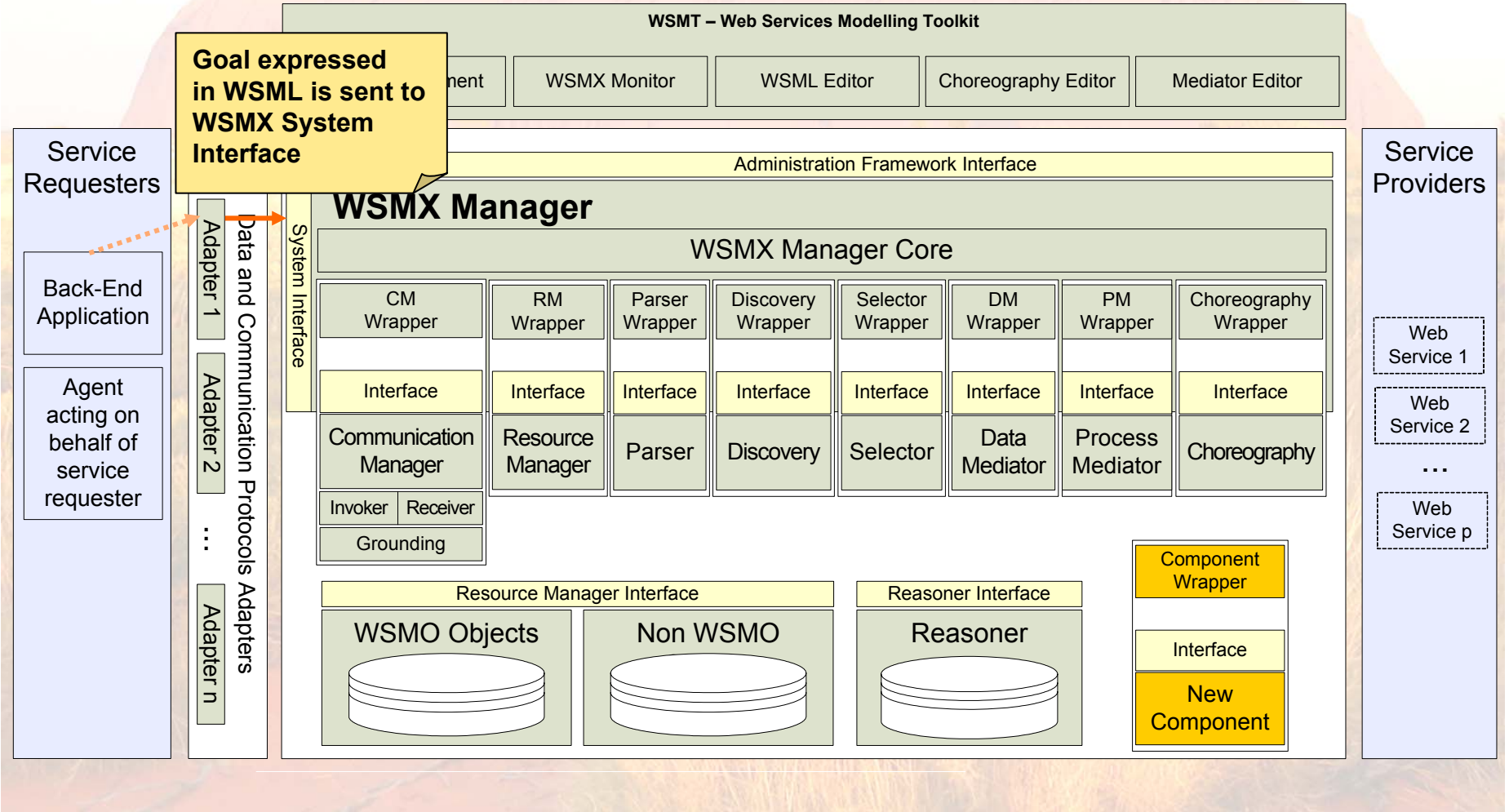


Walk Through

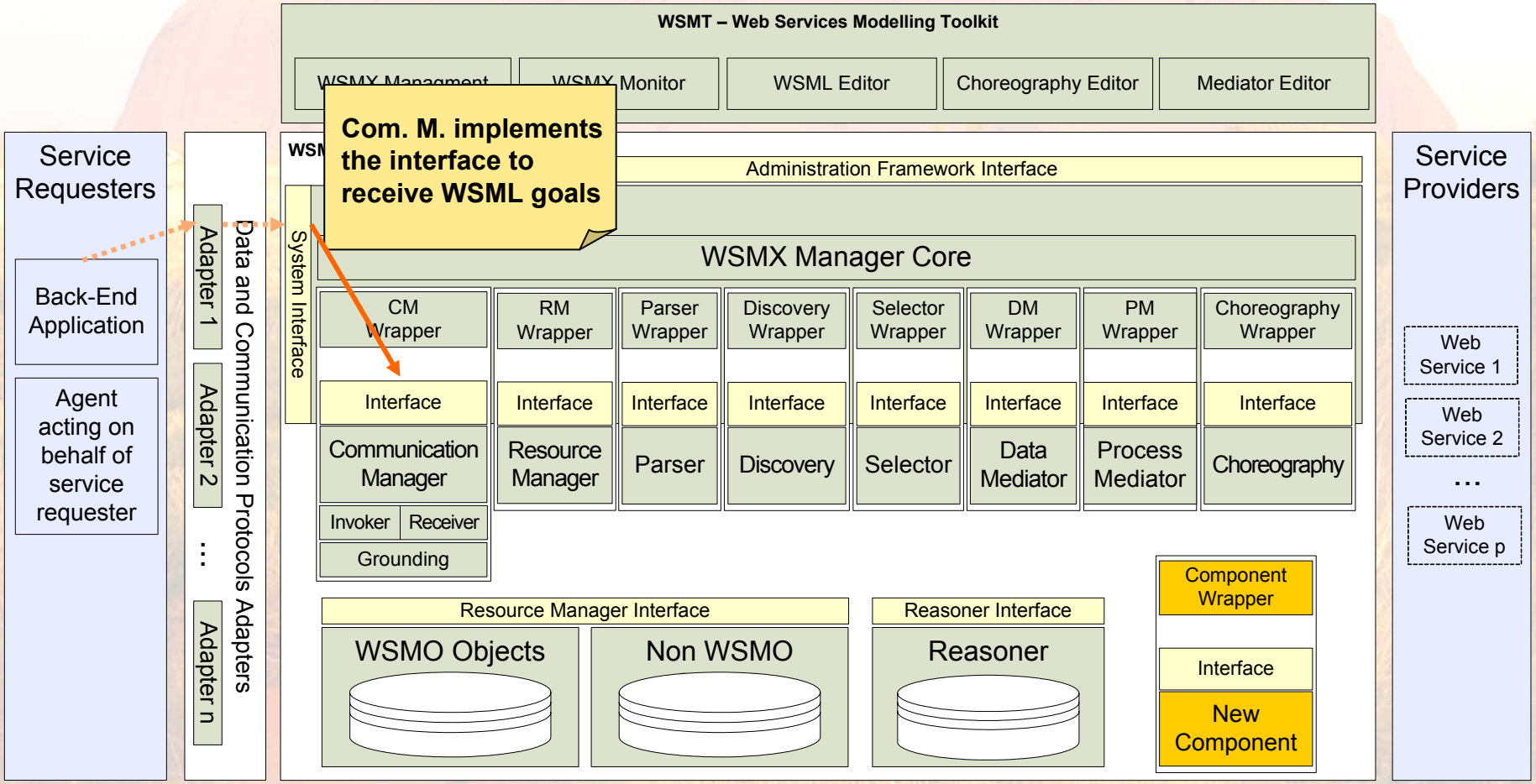
Execution Semantics



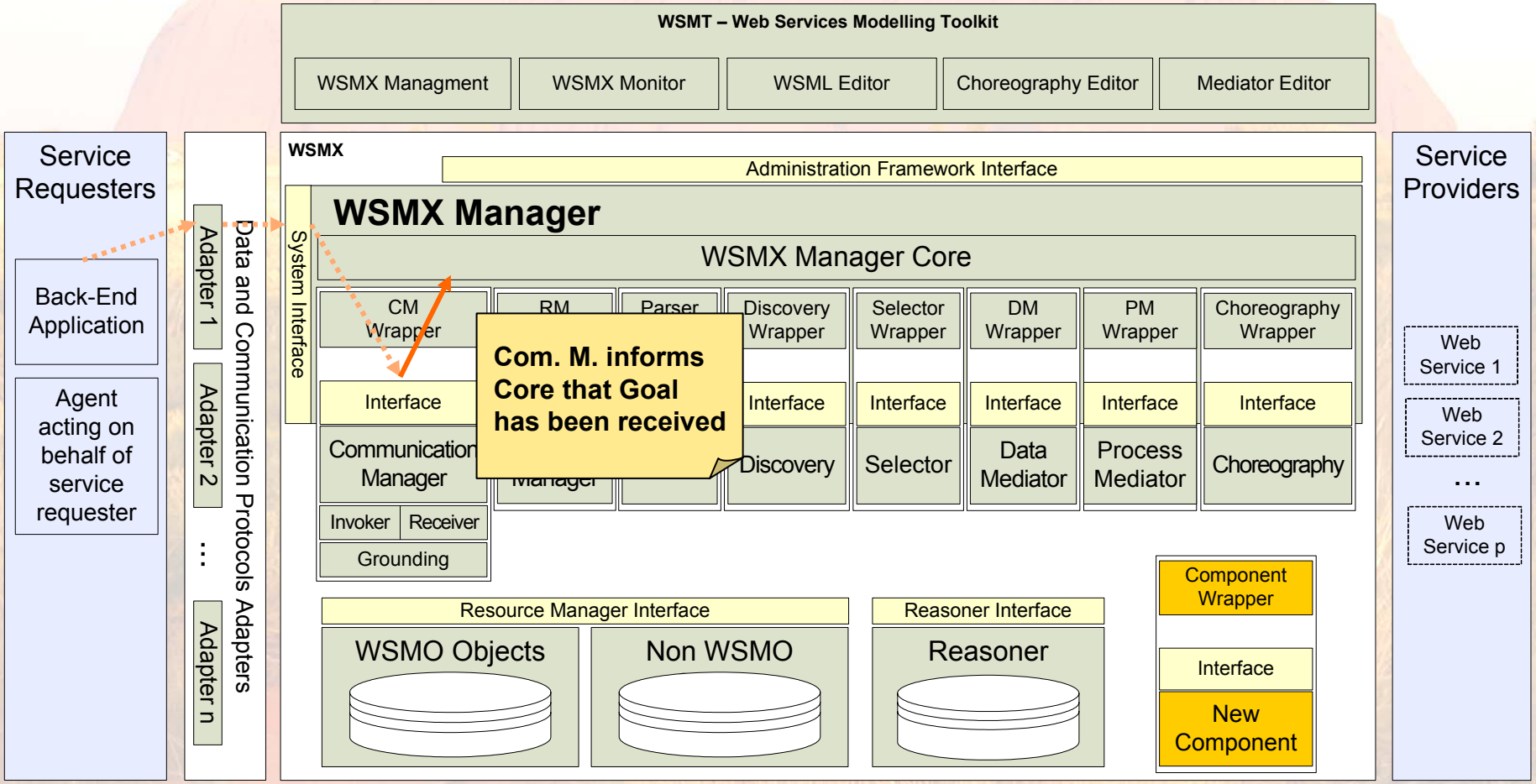
Execution Semantics



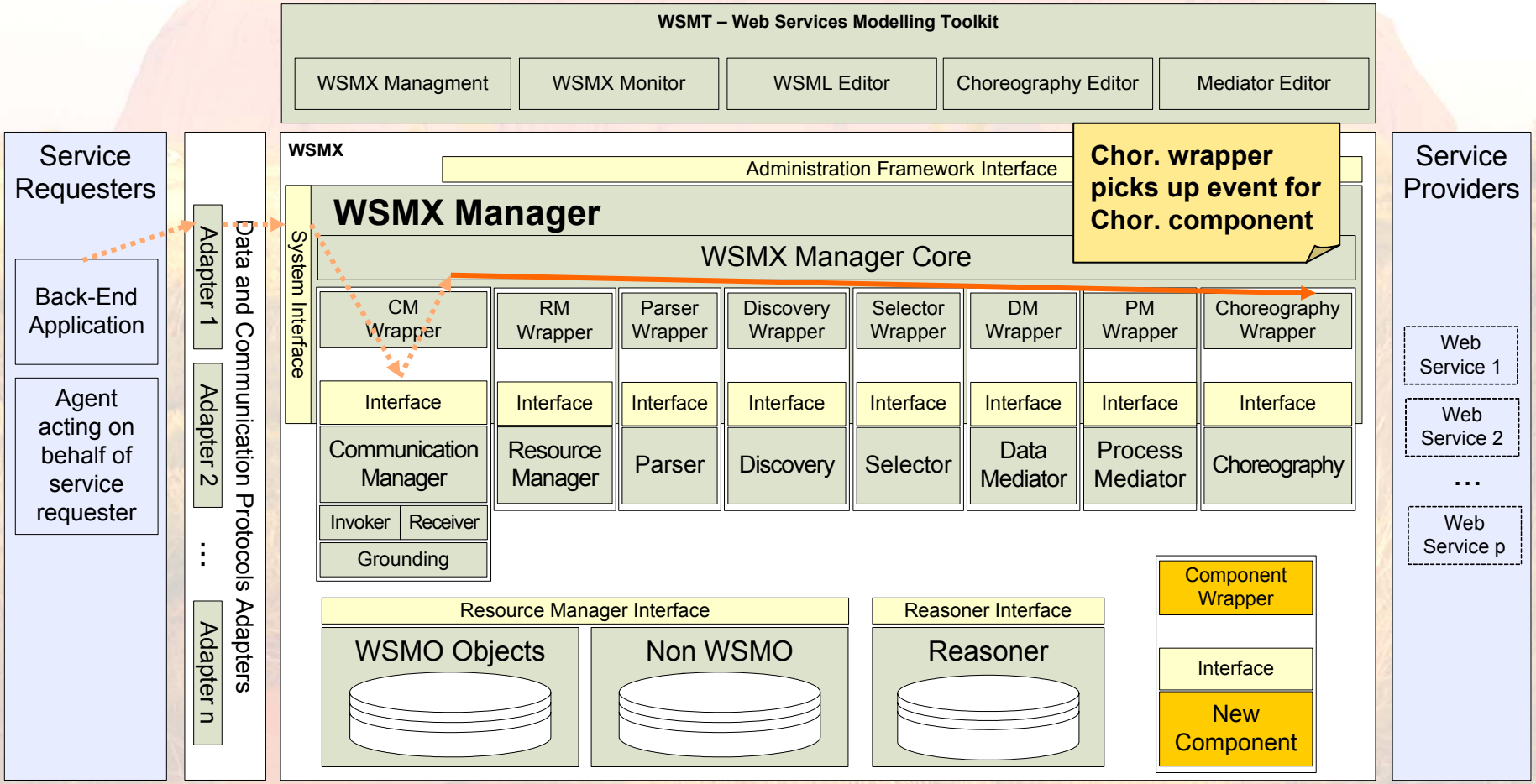
Execution Semantics



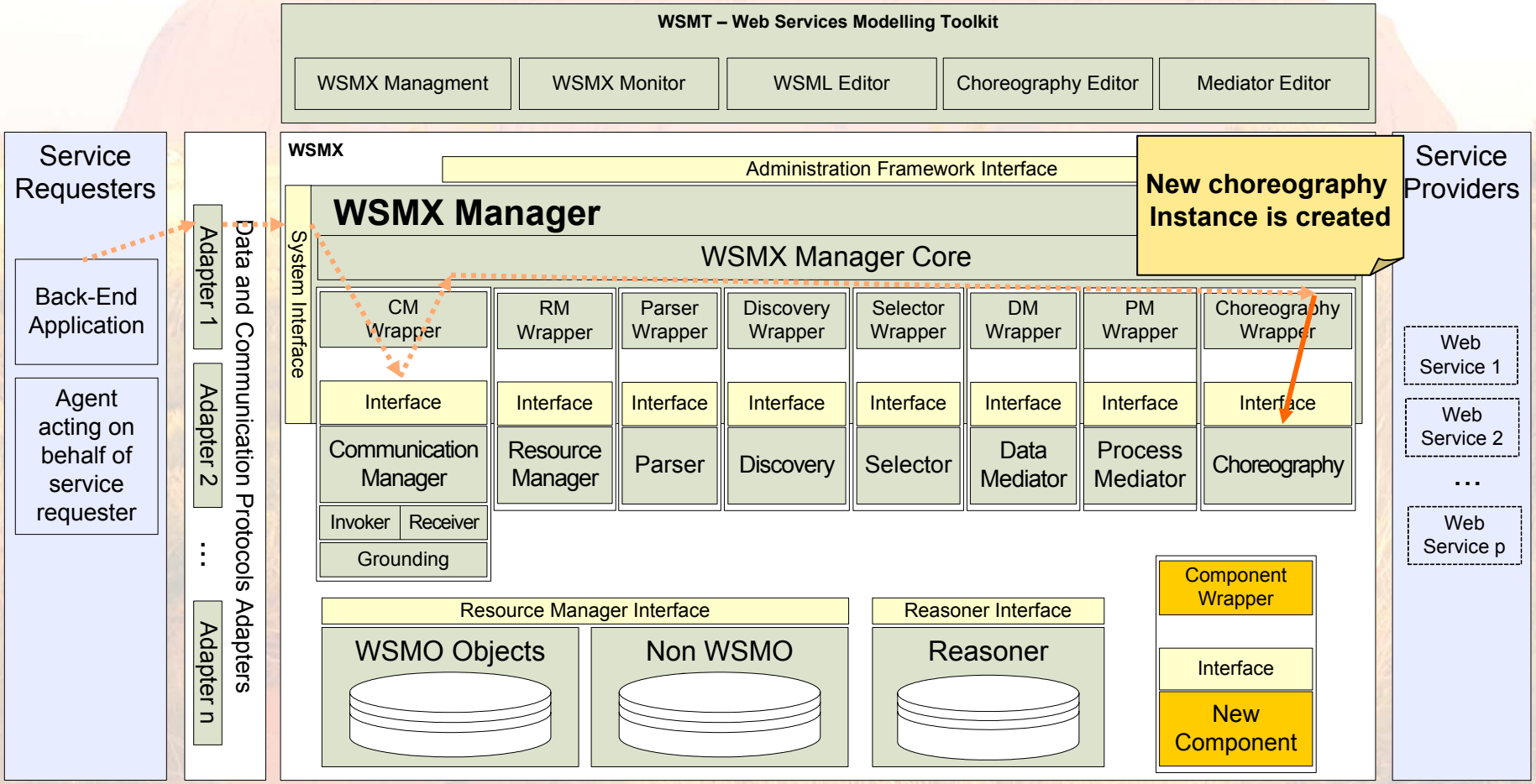
Execution Semantics



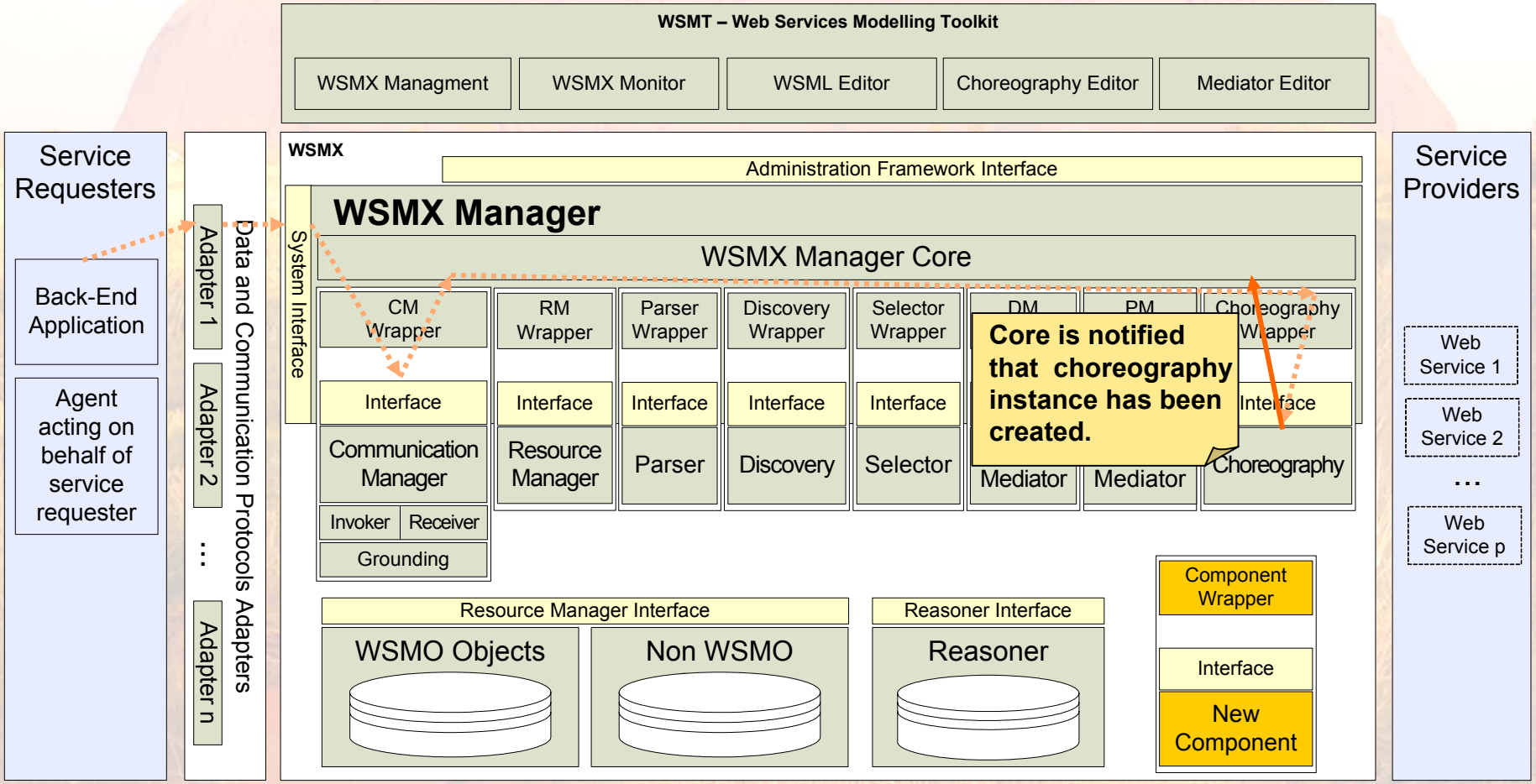
Execution Semantics



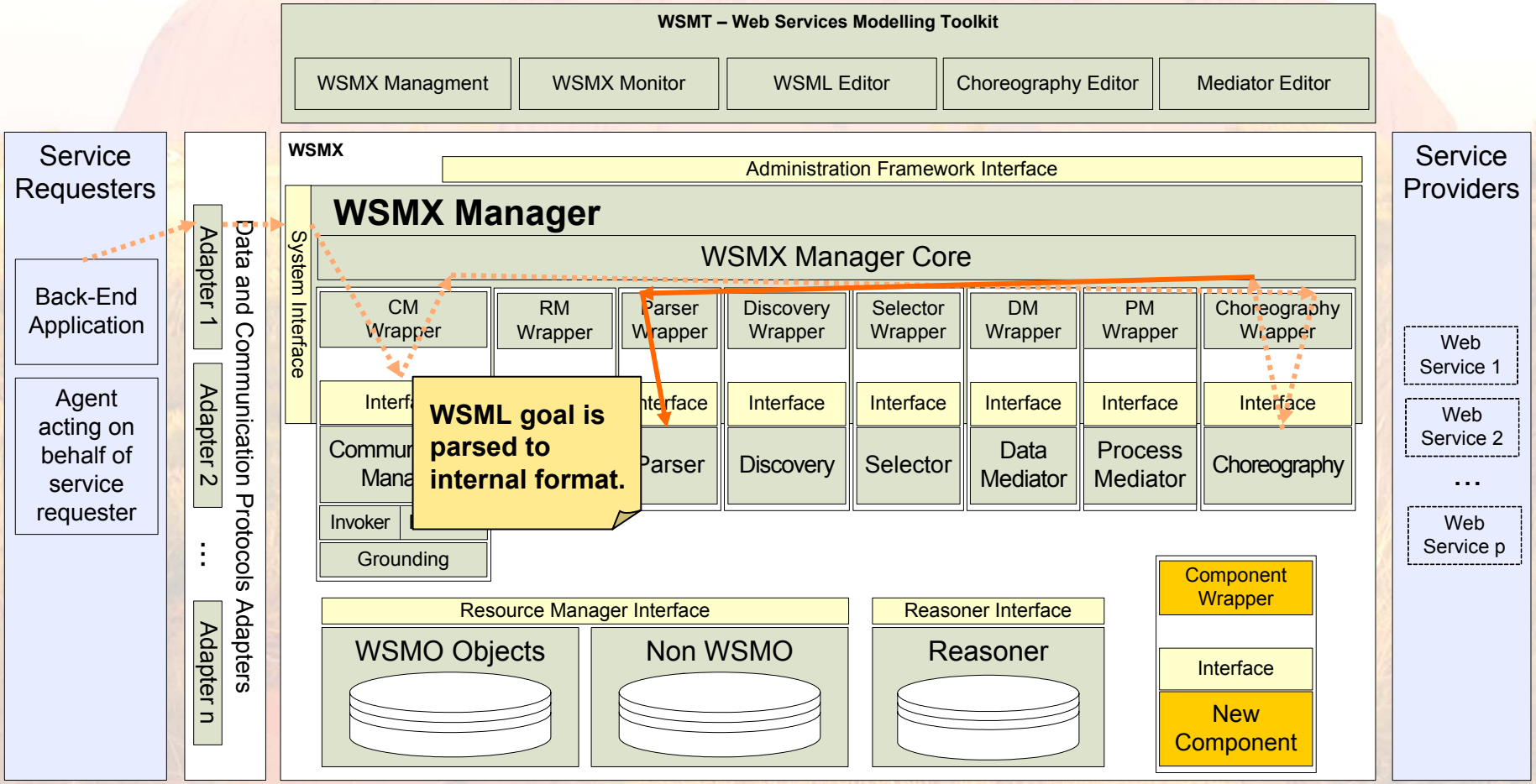
Execution Semantics



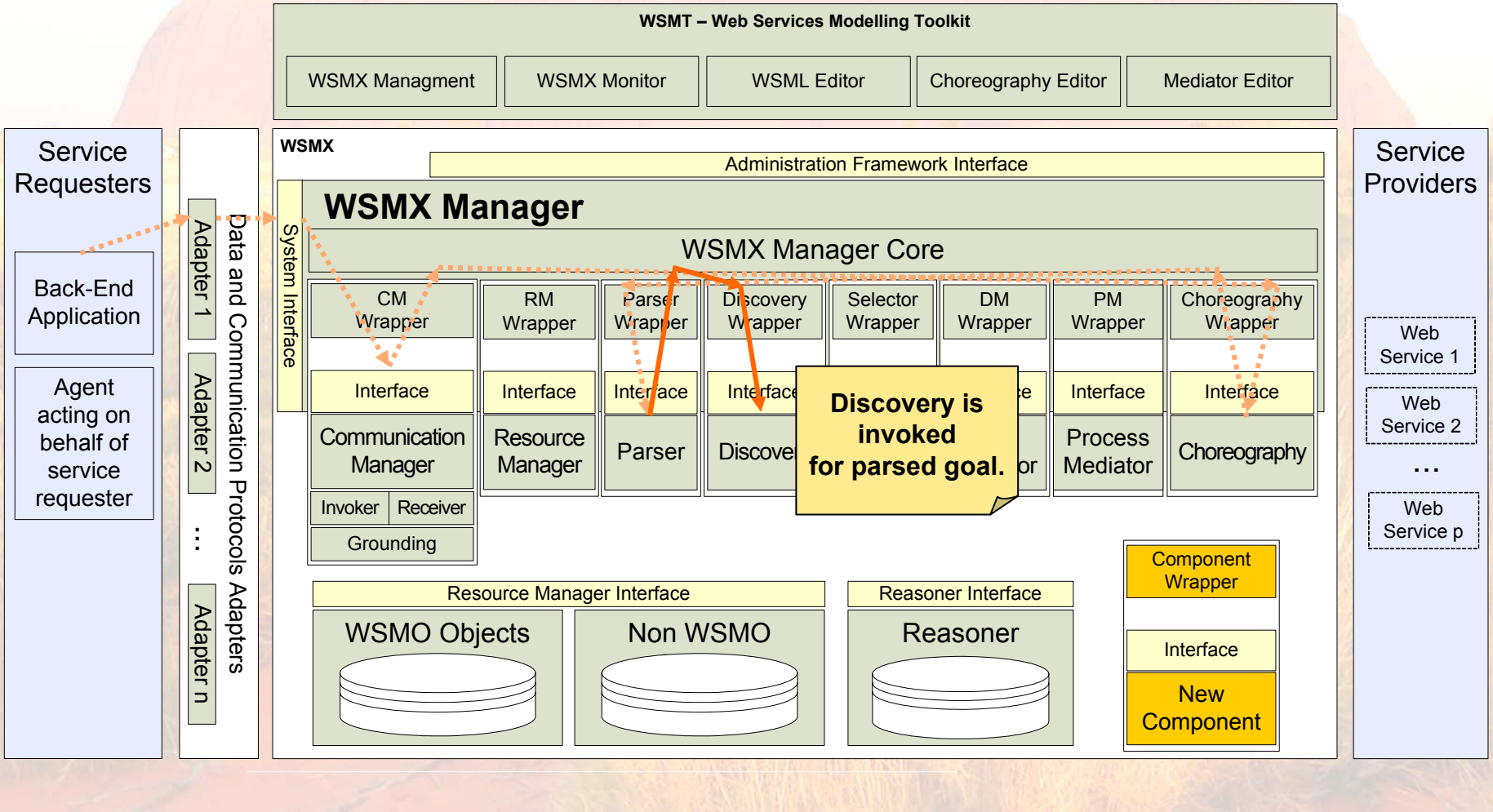
Execution Semantics



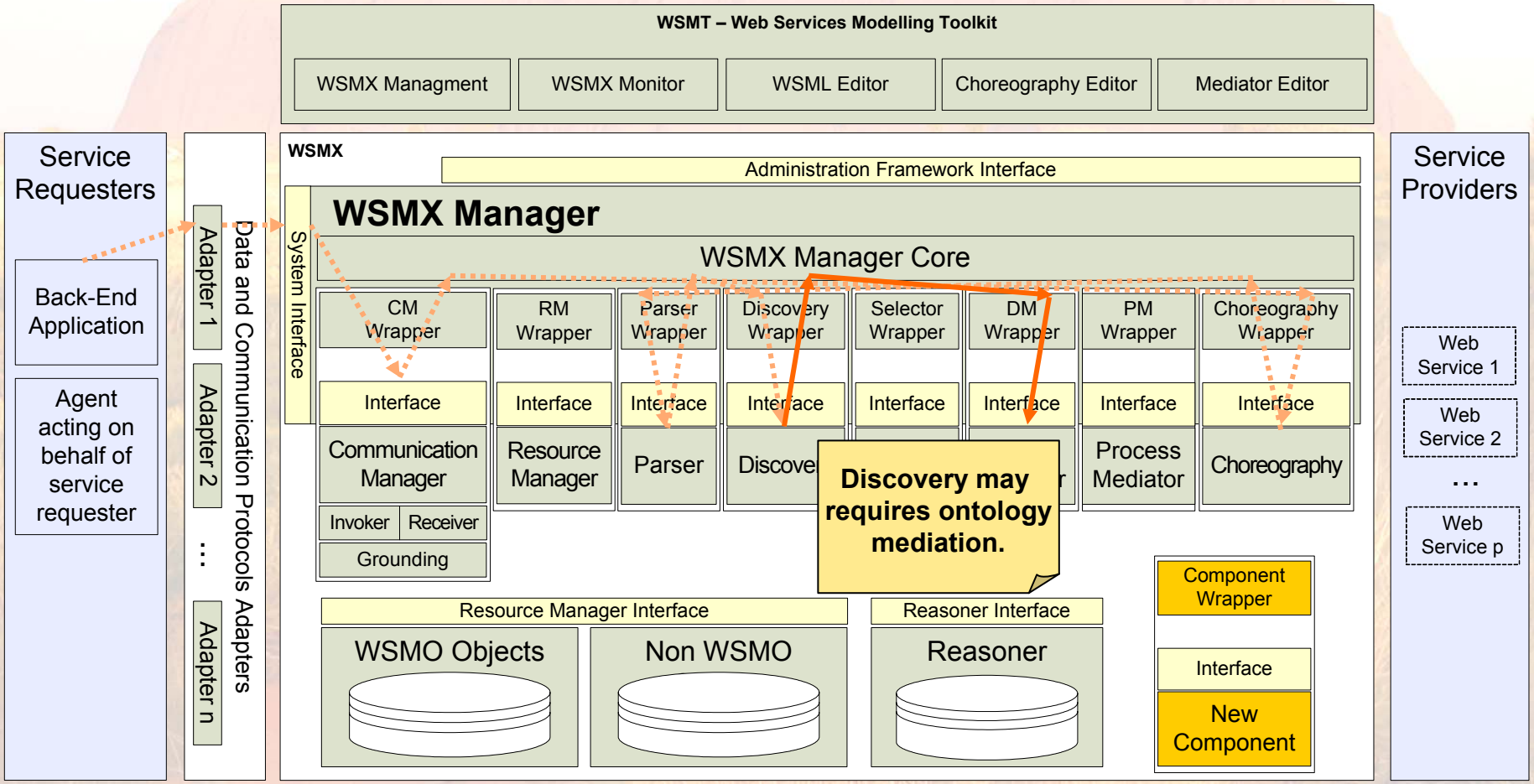
Execution Semantics



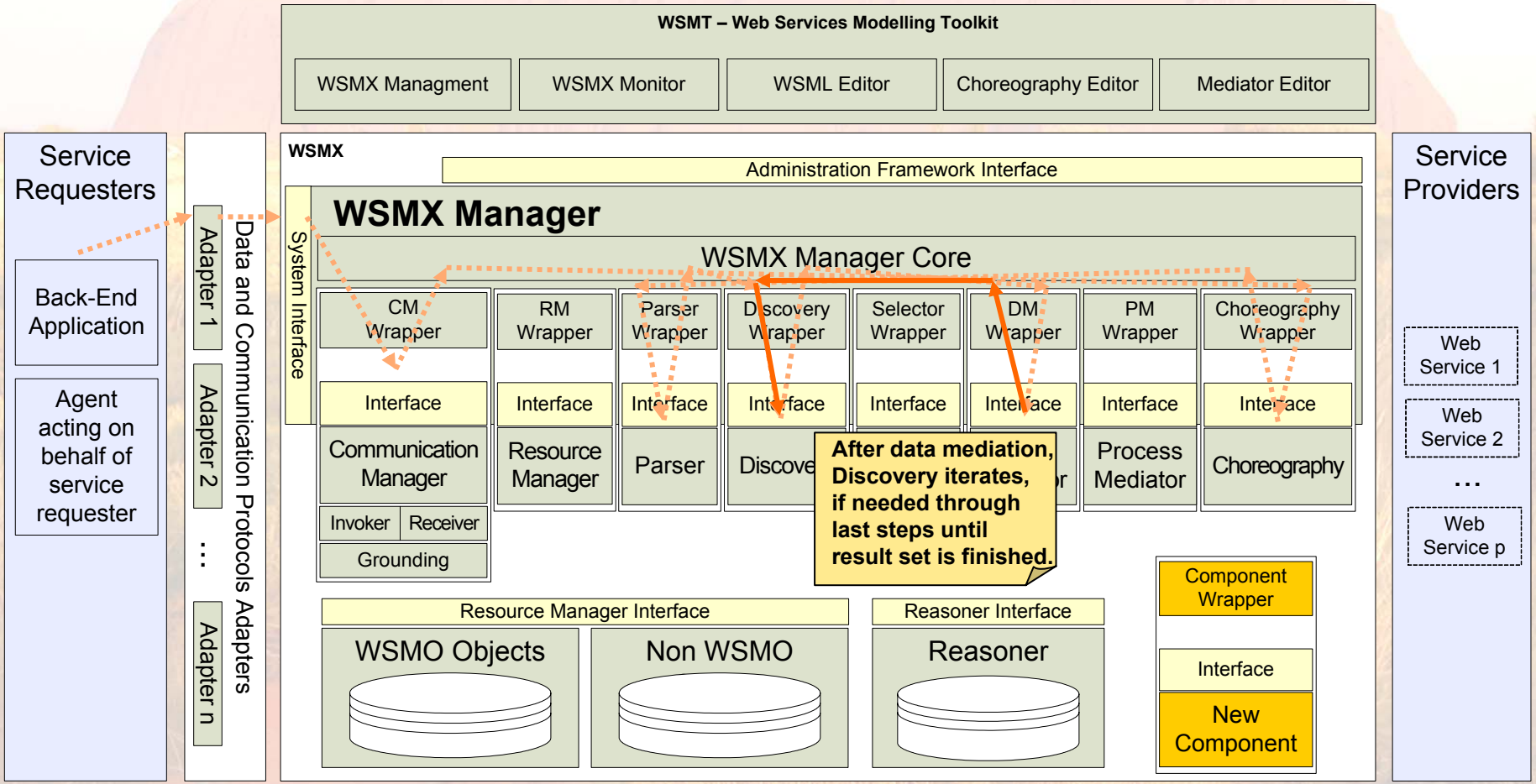
Execution Semantics



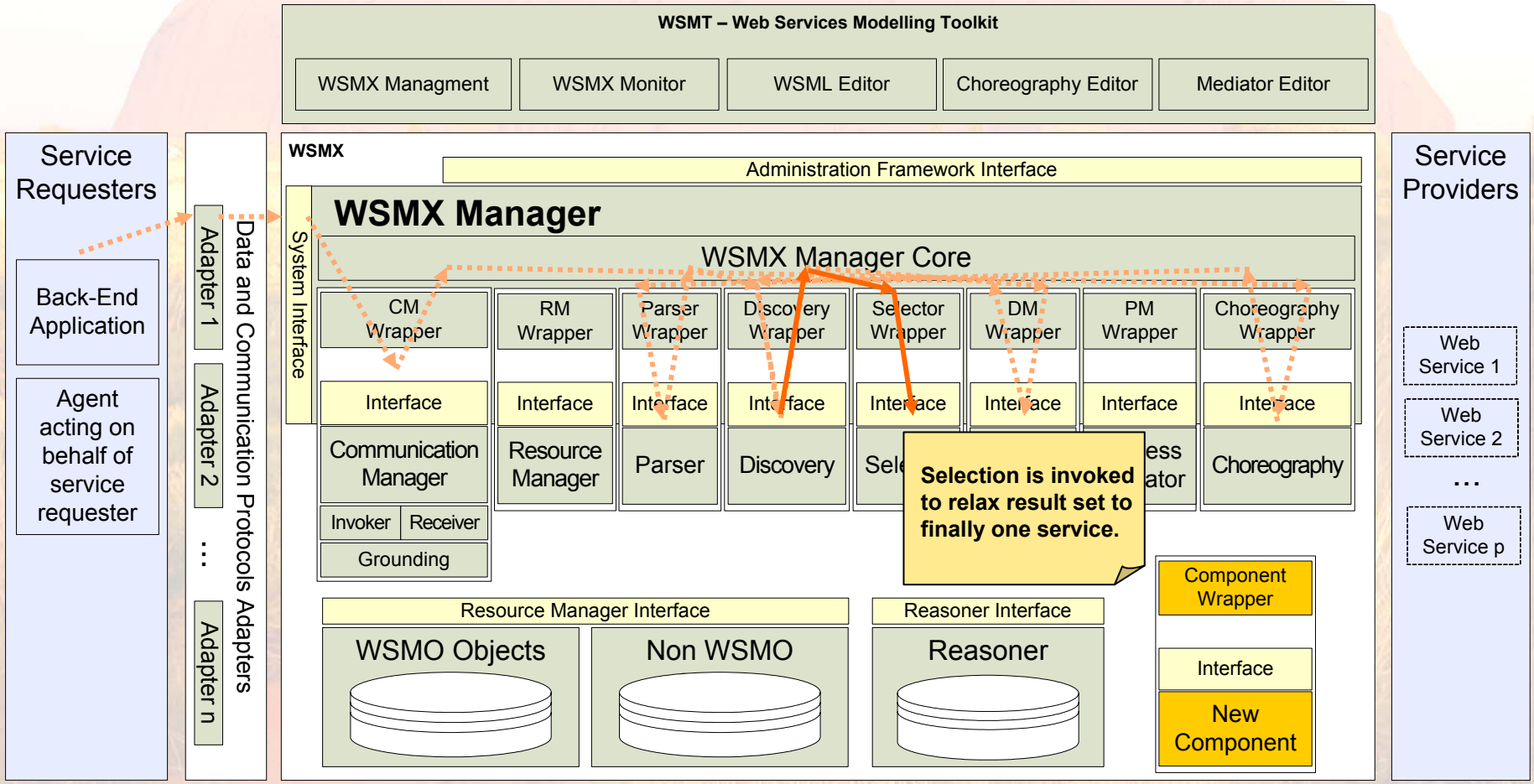
Execution Semantics



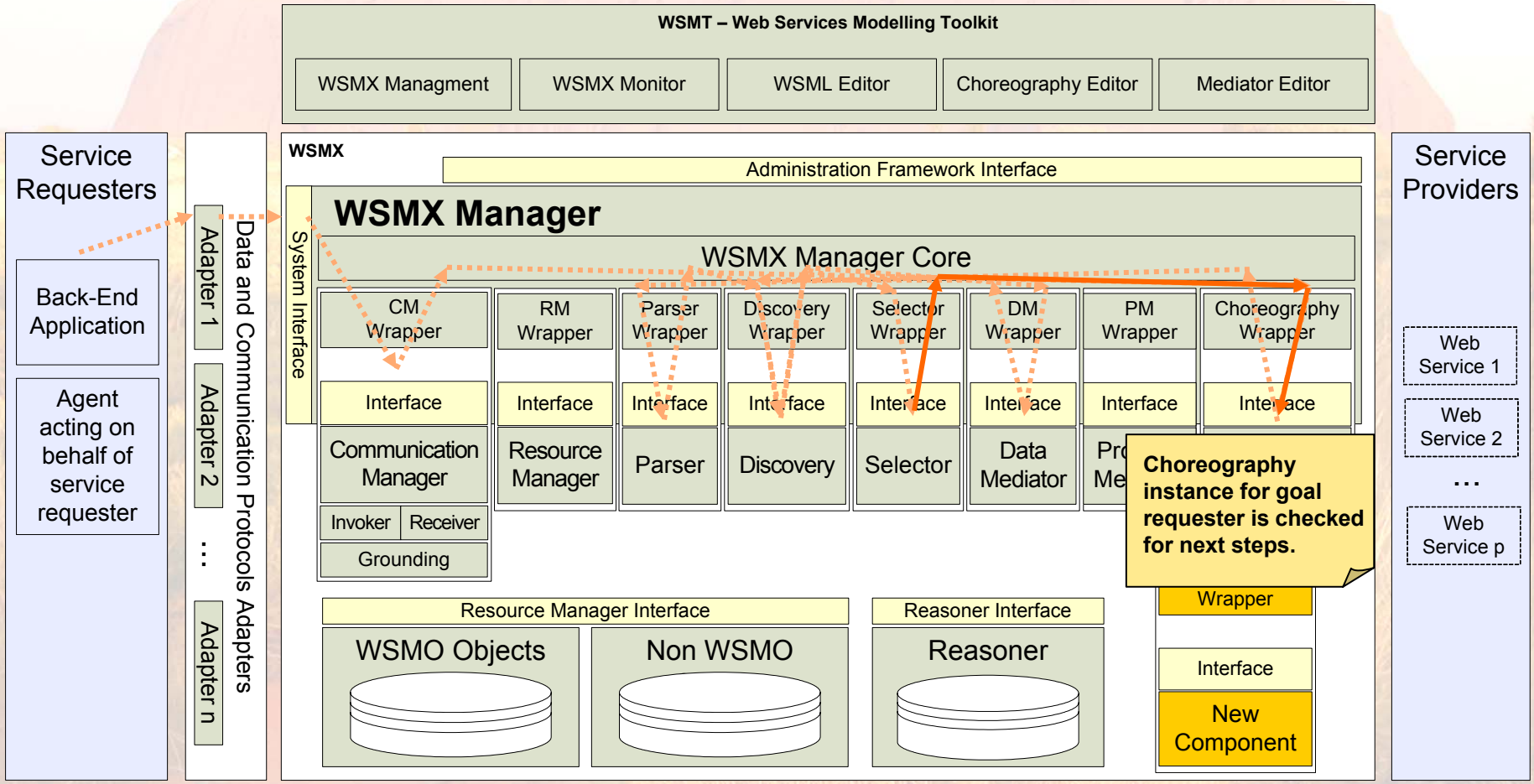
Execution Semantics



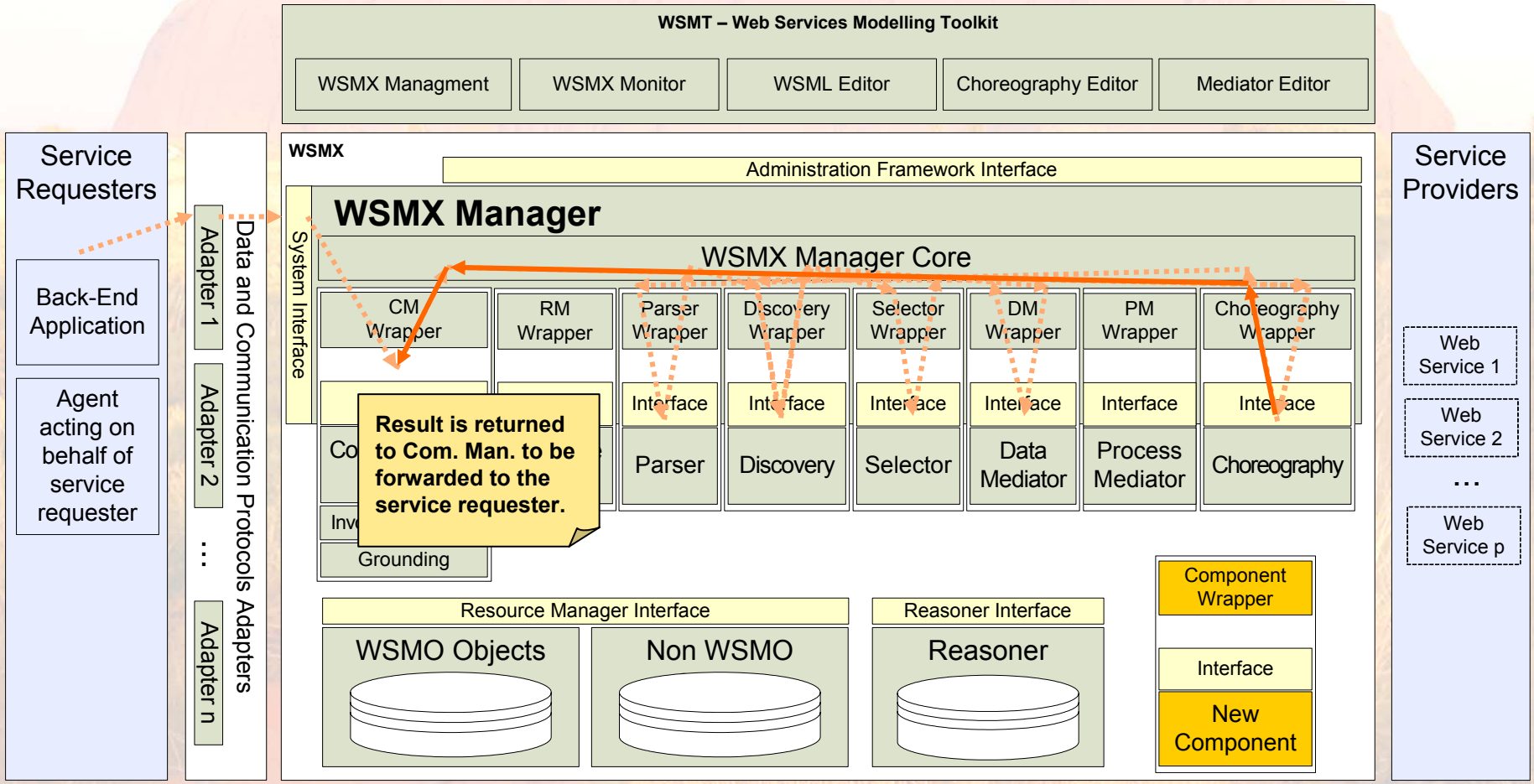
Execution Semantics



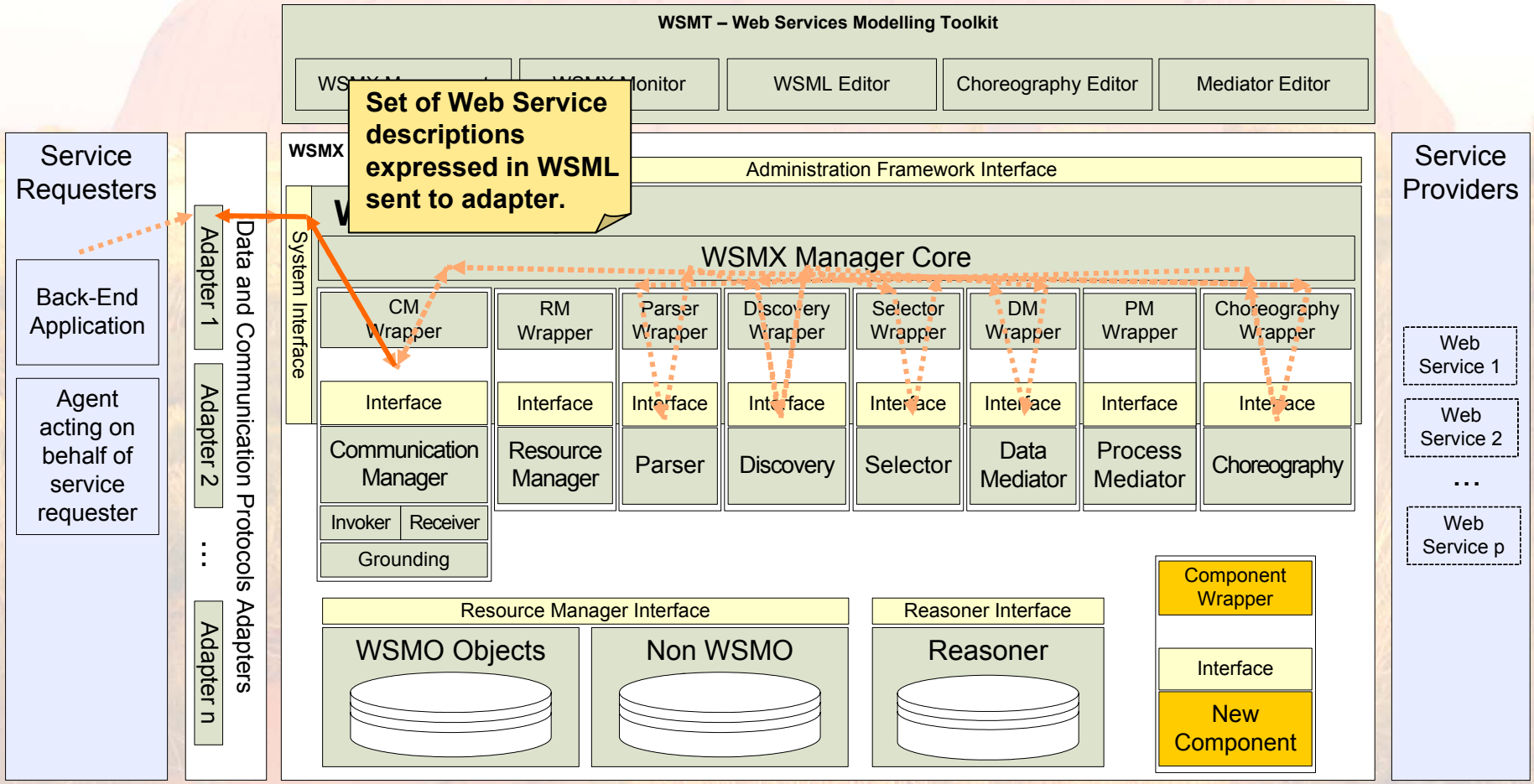
Execution Semantics



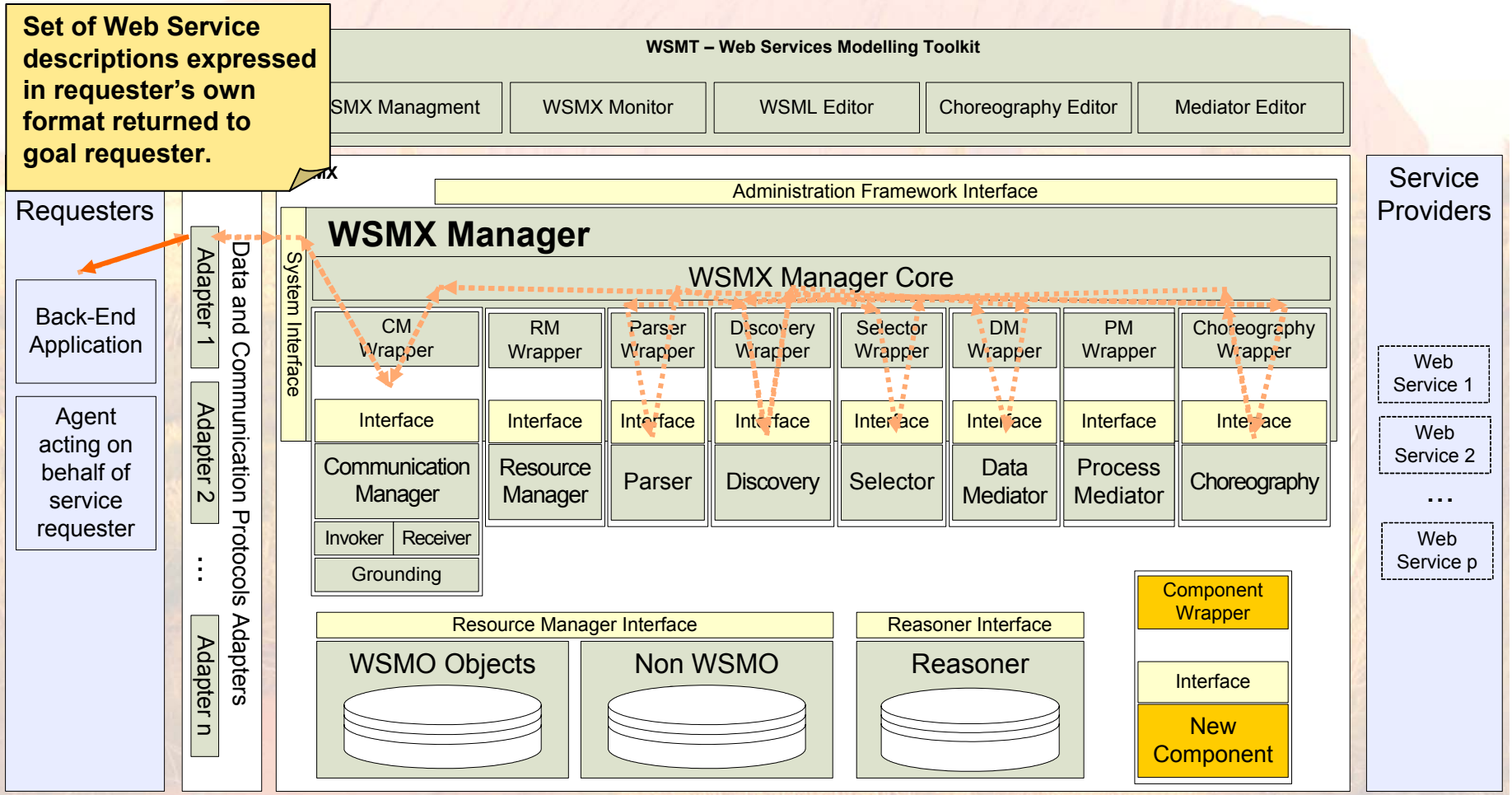
Execution Semantics



Execution Semantics



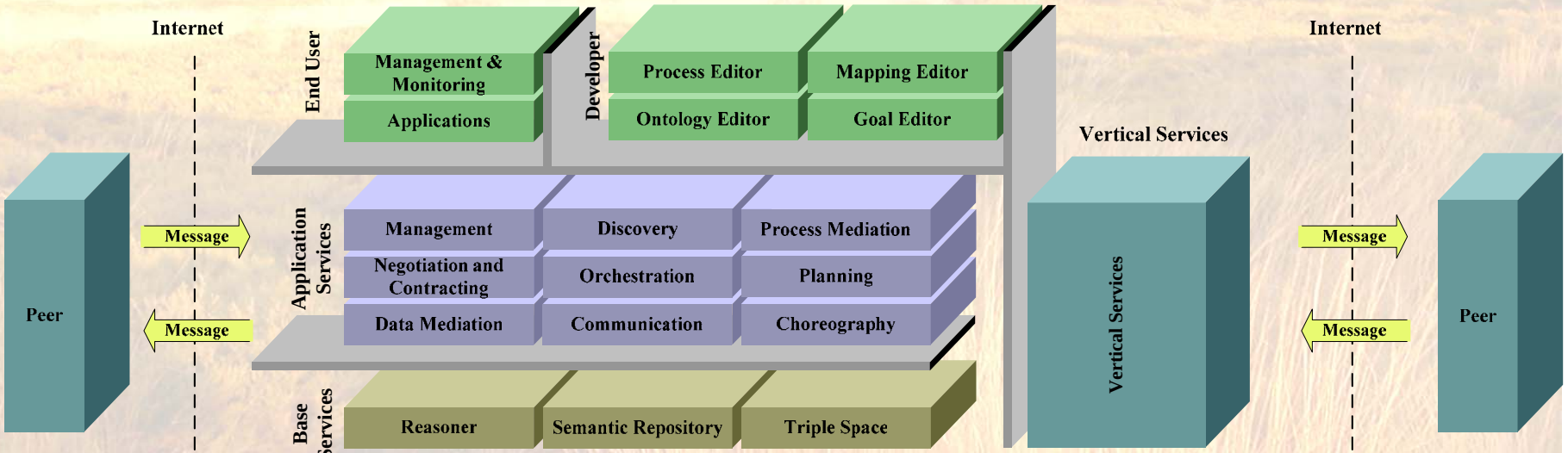
Execution Semantics



Future Plans & Conclusions

WSMX Usage Scenario - P2P

- Complete the functionality for all the boxes



Conclusions

- Conceptual model is WSMO (with some add-ons)
- End to end functionality for executing SWS
- Has a formal execution semantics
- Real implementation
- Open source code base at SourceForge
- Event-driven component architecture
- Growing functionality - developers welcome ☺

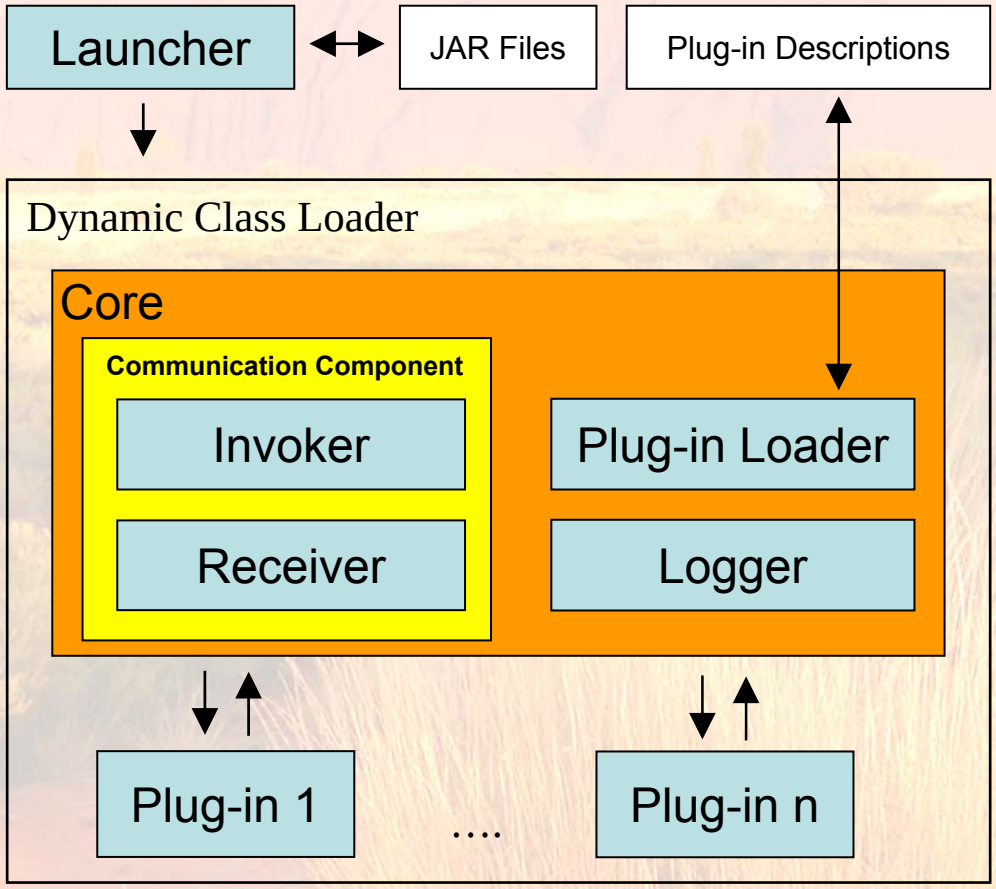
WSMX @ Sourceforge.net

Demos – WSMX and WSMT

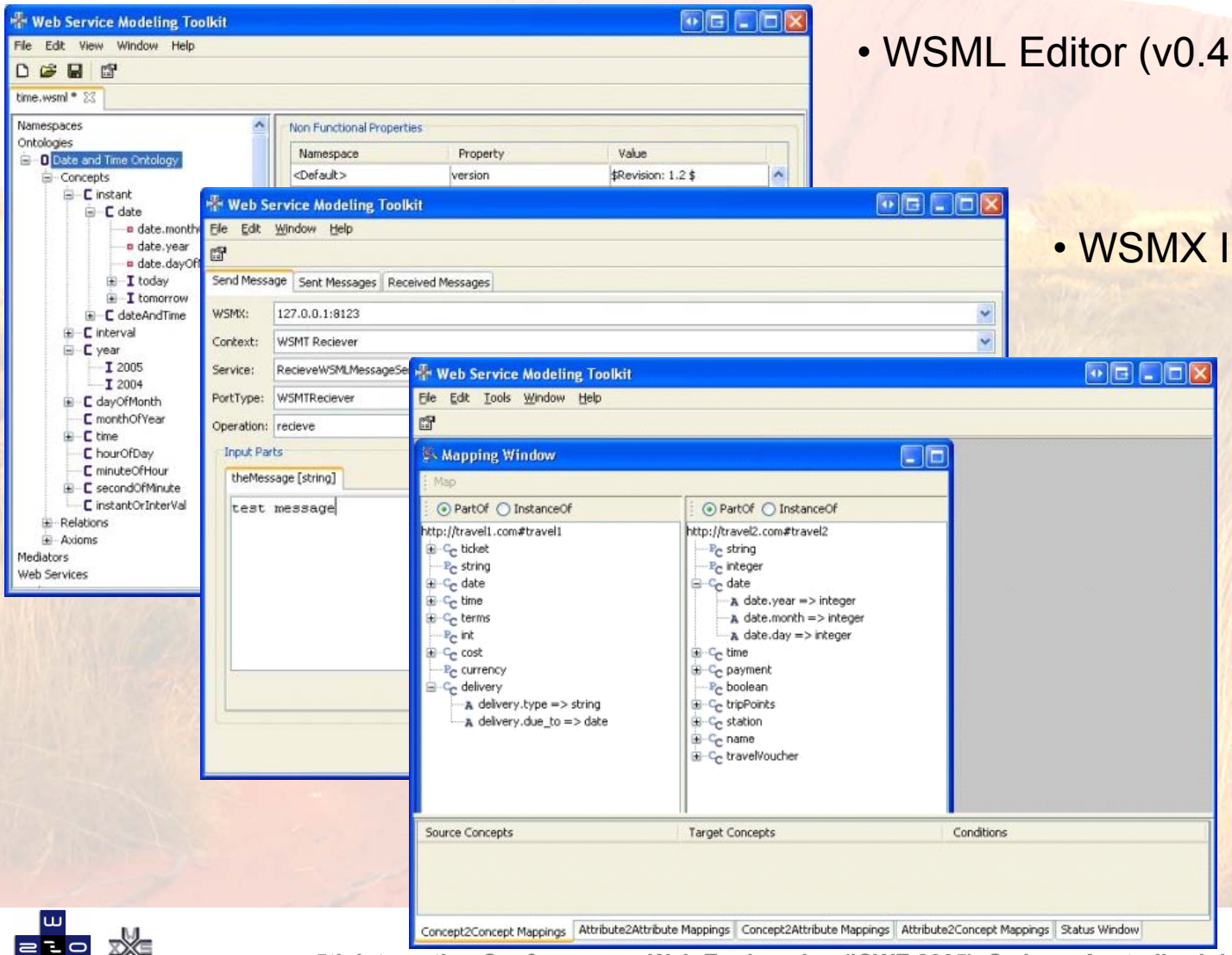
Web Service Modelling Toolkit

- WSMT is a lightweight framework for rapid creation and deployment of homogenous tools for Semantic Web Services.
- Aims
  - Reduce application overhead
  - Provide common reusable functionality
  - Allow new tools to be added dynamically
  - Encourage the creation of tools
- Open source at:
  - http://sourceforge.net/projects/wsmx/

WSMT Architecture



Current Tools



- WSML Editor (v0.4)
- WSMX Invoker (v0.1)
- WSMX Data Mediation Mapping Tool (v0.2.2)

PART V:

Hands-On Session

- **Internet Reasoning Service (IRS III):**
  - **system overview**
  - **demonstration**
- **Hands-on Session:**
  - **Introduction to Use Case Scenario**
  - **Hands-on Session tasks**
  - **Exercises**

IRS-III:

A framework and platform for building Semantic Web Services

John Domingue and Liliana Cabral

The Internet Reasoning Service is an infrastructure for publishing, locating, executing and composing *Semantic Web Services*

Design Principles

- Ontological separation of User and Web Service Contexts
- Capability Based Invocation
- Ease of Use
- One Click Publishing
- Agnostic to Service Implementation Platform
- Connected to External Environment
- Open
- Complete Descriptions
- Inspectable
- Interoperable with SWS Frameworks and Platforms

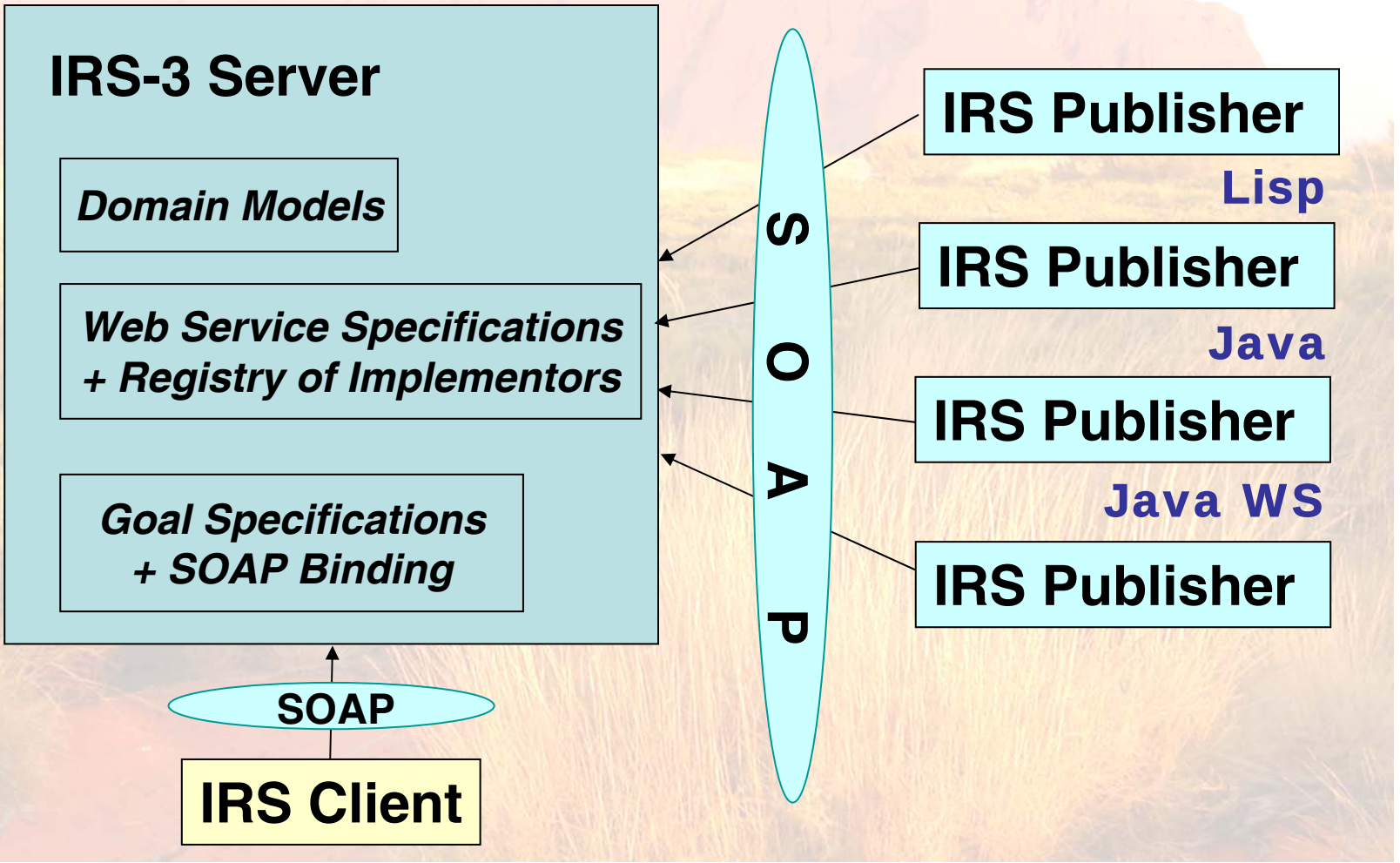
Features of IRS-III (1/2)

- Based on Soap messaging standard
- Provides Java API for client applications
- Provides built-in brokering and service discovery support
- Provides *capability-centred* service invocation

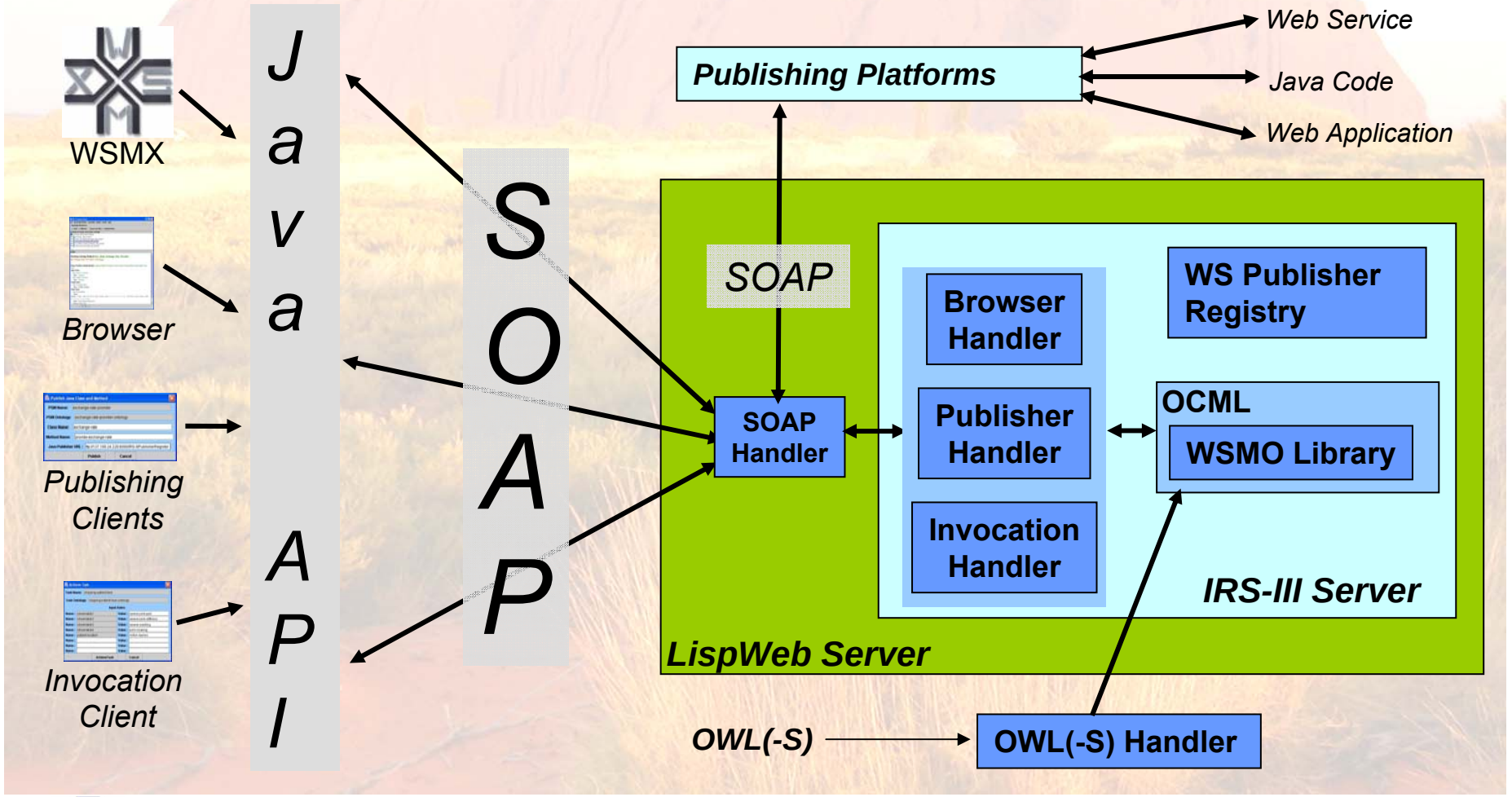
Features of IRS-III (2/2)

- Publishing support for variety of platforms
  - Java, Lisp, Web Applications, Java Web Services
- Enables publication of ‘standard code’
  - Provides clever wrappers
  - One-click publishing of web services
- Integrated with standard Web Services world
  - Semantic web service to IRS
  - ‘Ordinary’ web service

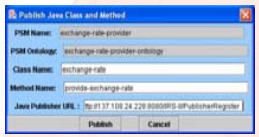
IRS-III Framework



IRS-III Architecture



Publishing Platform Architecture



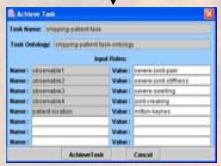
Publishing Clients

SOAP

SOAP

IRS-III Server

SOAP



Invocation Client

SOAP Handler

Service Registrar

Service Invoker

WS Service Registry

IRS-III Publishing Platform

HTTP Server

Web Service 1

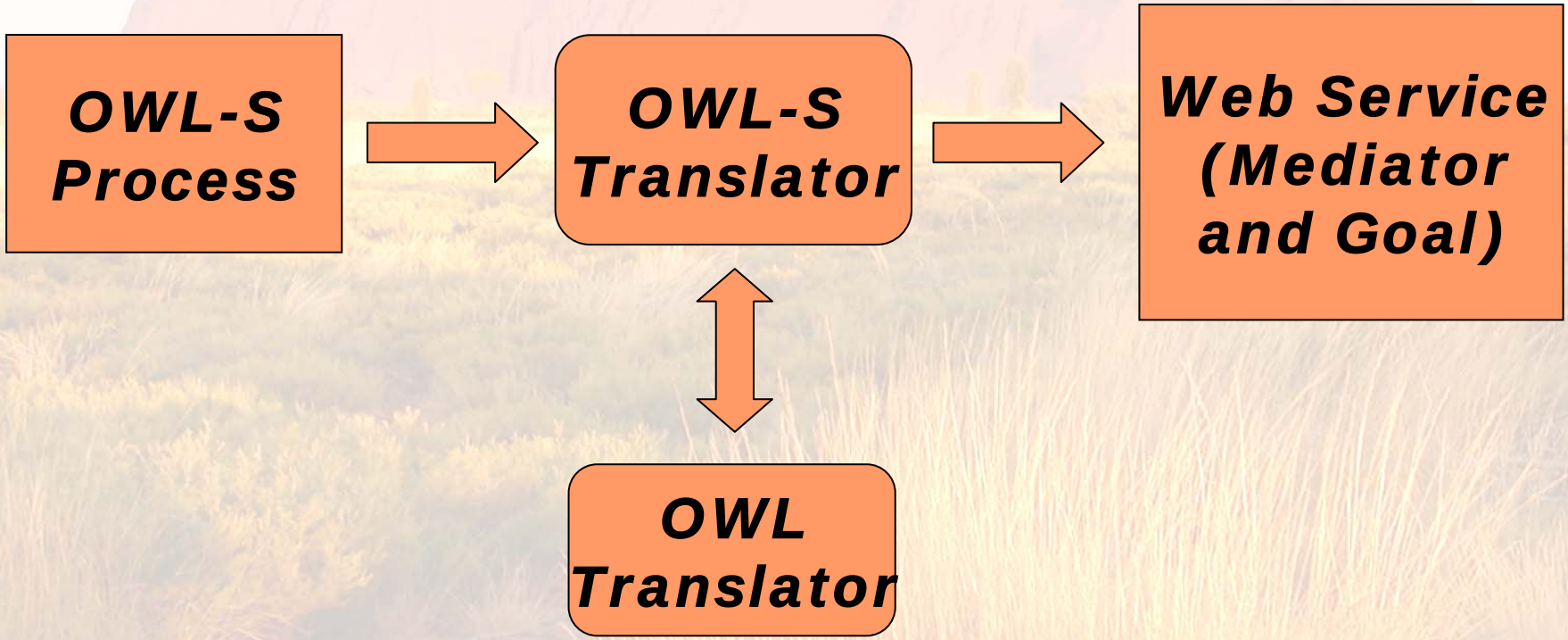
Web Service 2

Web Service 3

IRS-III/WSMO differences

- Underlying language OCML
- Goals have inputs and outputs
- IRS-III broker finds applicable web services via mediators
  - Used mediator within WS capability
  - Mediator source = goal
- Web services have inputs and outputs ‘inherited’ from goal descriptions
- Web service selected via assumption (in capability)

OWL-S 1.0 Translation



OWL Process to Web Service

- IOPEs are translated to:

  has-input, has-output, has-precondition and has-postcondition

  in the capability of a Web service.

- The type and condition definitions at the range of the above roles are translated by the OWL to OCML translator.

- Simple goal and mediators can be generated (optional) as template for later development.

IRS-III Demo

(including OWL-S Import)

John Domingue and Liliana Cabral

SWS Creation & Usage Steps

- Create a goal description
  - (e.g. exchange-rate-goal)
  - Add input and output roles
  - Include role type and soap binding
- Create a wg-mediator description
  - Source = goal
  - Possibly add a mediation service
- Create a web service description
  - Used-mediator of WS capability = wg-mediator above
- Specify Operation <-> Lisp function mapping in Choreography Grounding
- Publish against web service description
- Invoke web service by ‘achieve goal’

Multiple WS for goal

- Each WS has a mediator for used-mediator slot of capability
  - Some WS may share a mediator
- Define a kappa expression for assumption slot of WS capability
- Kappa expression format
  - (kappa (?goal) <ocml relations>)
- Getting the value of an input role
  - (wsmo-role-value ?goal <role-name>)

Defining a Mediation Service

- Define a wg-mediator
- Source = goal
- Mediation-service = goal for mediation service
- Mediation goal
  - Mediation goal input roles are a subset of goal input roles
- Define mediator and WS as normal

Valid Relations

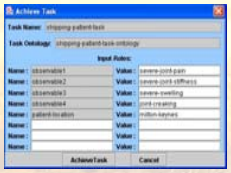
- Classes are unary relations
  - e.g. (country ?x)
- Slots are binary relations
  - e.g. (is-capital-of ?x ?y)
- Standard relations in base (OCML toplevel) ontology

  =, ==, <, >, member

European Currency Assumption

```
(kappa (?goal)
    (member
        (wsmo-role-value
            ?goal
            'has_source_currency)
        '(euro pound)))
```

Goal Based Invocation



Invocation

Solve Goal
Goal -> WG Mediator -> WS/Capability/Used-mediator

Instantiate Goal Description

Exchange-rate-goal
Has-source-currency: us-dollars
Has-target-currency: pound

Web Service Discovery

European-exchange-rate-ws
Non-european-exchange-rate-ws
European-bank-exchange-rate-ws

WS -> Capability -> Assumption expression

Web service selection

European-exchange-rate

Mediation

Mediate input values

'$' -> us-dollar

Invocation

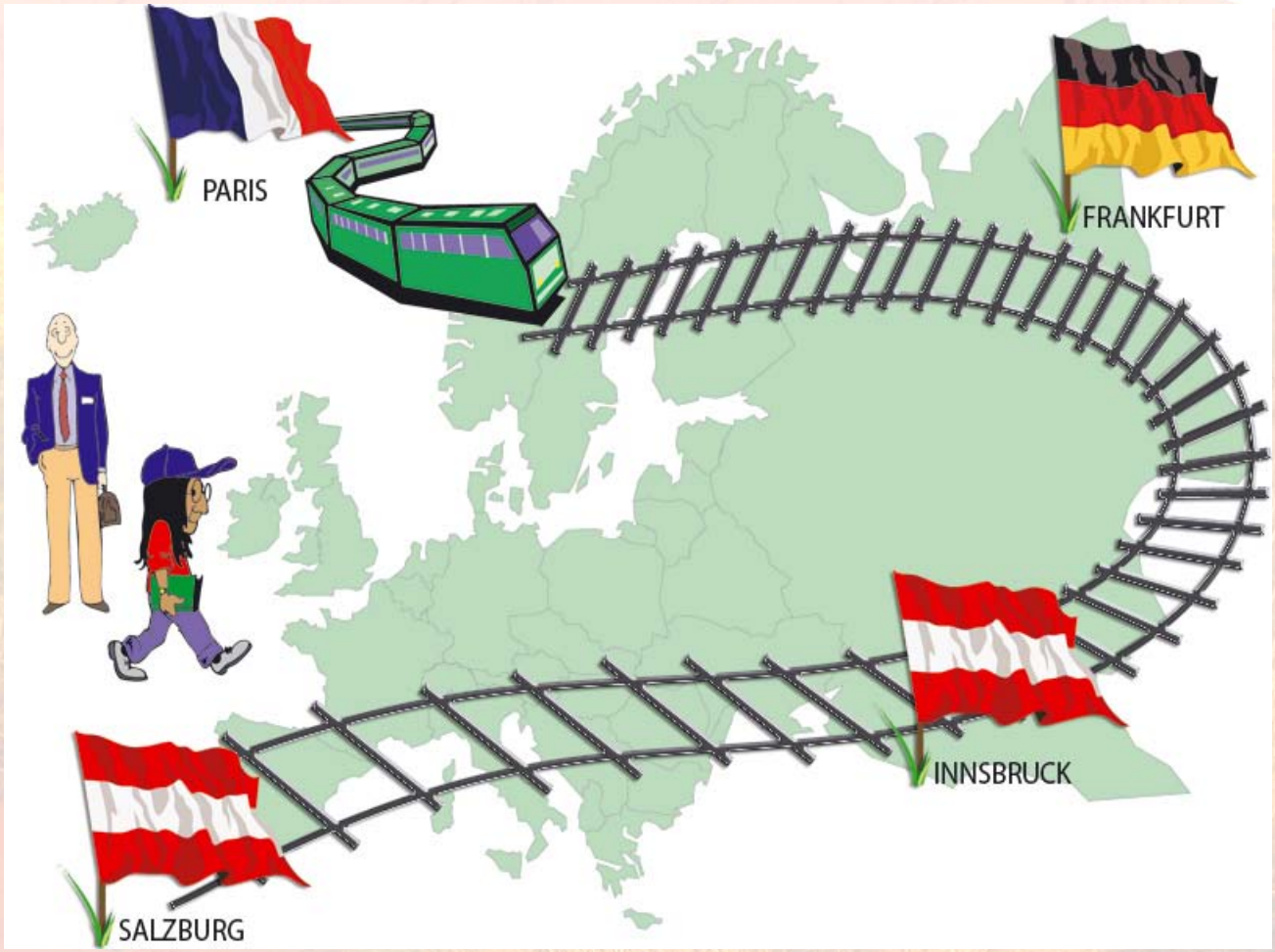
Invoke selected web service

European-exchange-rate

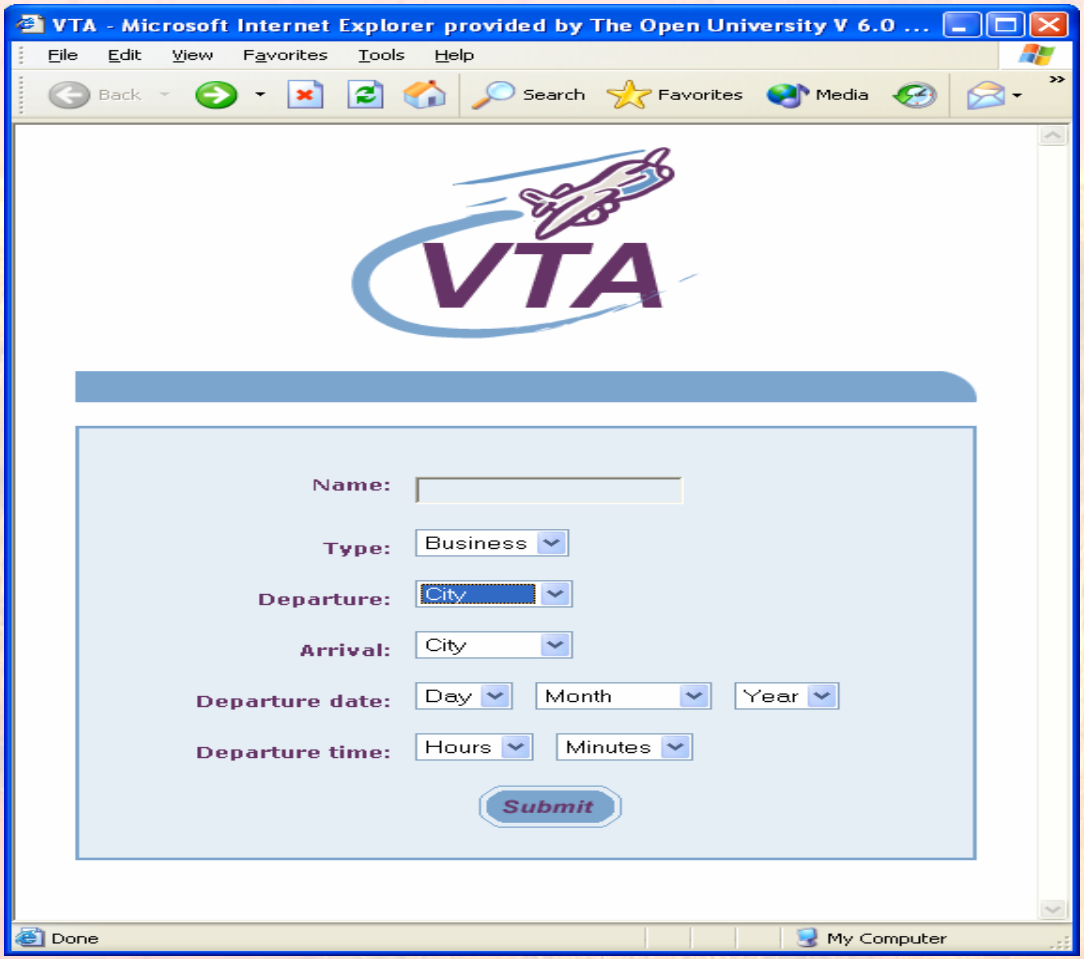
Hands-On Session

John Domingue and Liliana Cabral

European Travel Scenario



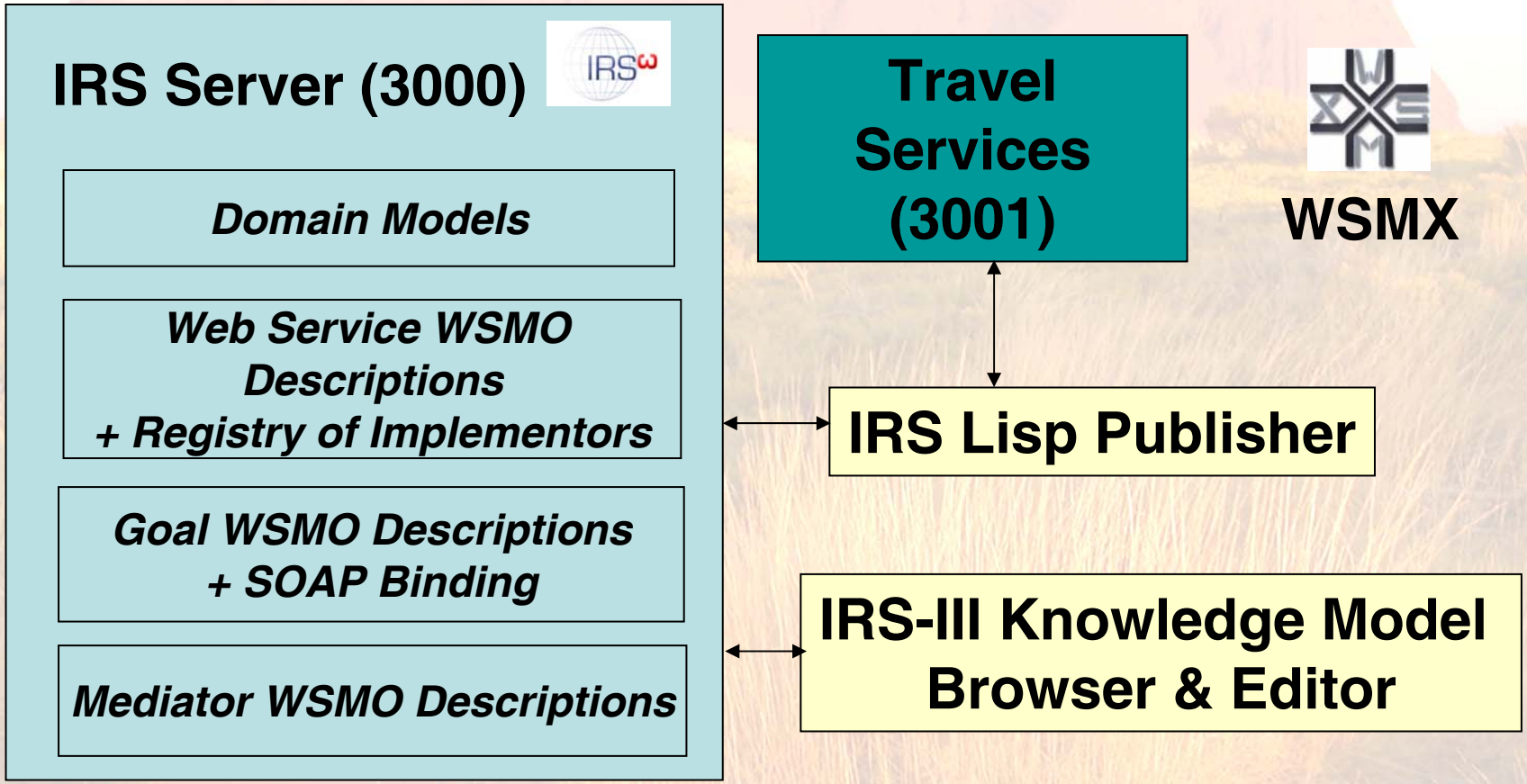
European Travel Demo



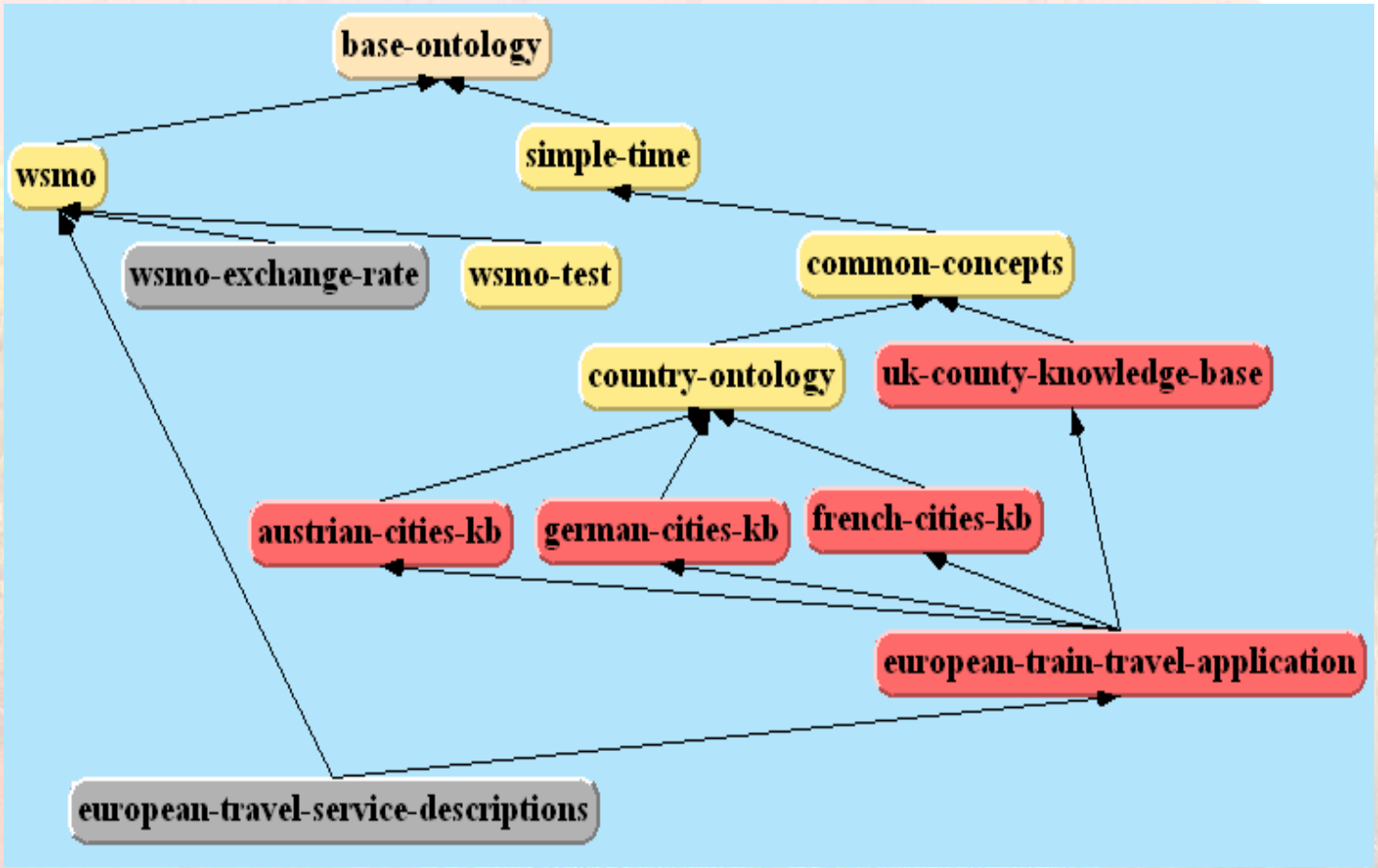
IRS-III Hands On Task

- Develop an application for the European Travel scenario based on SWS. The application should support a person booking a train ticket between 2 European cities at a specific time and date
- Create Goal, Web service and Mediator WSMO descriptions in IRS-III (european-travel-service-descriptions) for available services. Your descriptions should choose a specific service depending on the start and end locations and the type of traveller. Use the assumption slot to do this
- Publish available lisp functions against your descriptions
- Invoke the web services
- Solution to be shown at the end of this session

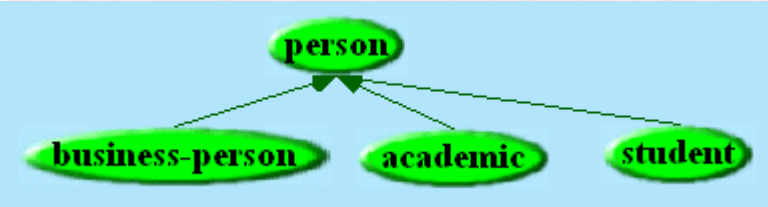
Tutorial Setup



Travel Related Knowledge Models



Key Classes, Relations, Instances



Is-in-country <city> <country> e.g.

(is-in-country berlin germany) -> true

(student <person>) -> true, for john matt michal

(business-person <person>) -> true, for liliana michael

Goals

1- Get train timetable

- Inputs: origin and destination cities (city), date (date-and-time, e.g. (18 4 2004))
- Output: timetable (string)

2- Book train

- Inputs: passenger name (person), origin and destination cities, departure time-date (list-date-and-time, e.g. (20 33 16 15 9 2004))
- Output: booking information (string)

Services

- 1 service available for goal 1
  - No constraints
- 6 services available for goal 2
  - As a provider write the constraints applicable to the services to satisfy the goal (assumption logical expressions)
- 1 wg-mediator mediation-service
  - Used to convert time in list format to time in universal format

Service constraints

- Services 2-5
  - Services for (origin and destination) cities in determined countries
- Service 4-5
  - Need a mediation service to map goal time-date to service time-date
- Services 6-7
  - Services for students or business people in Europe

Available Functions (1/3)

1- get-train-times

paris london (18 4 2004)

"Timetable of trains from PARIS to LONDON on 18, 4, 2004
5:18
…23:36"

2- book-english-train-journey

christoph milton-keynes london (20 33 16 15 9 2004)

"British Rail: CHRISTOPH is booked on the 66 going from MILTON-KEYNES to LONDON at 16:49, 15, SEPTEMBER 2004. The price is 169 Euros."

3- book-french-train-journey

sinuhe paris lyon (3 4 6 18 8 2004)

"SNCF: SINUHE is booked on the 511 going from PARIS to LYON at 6:12, 18, AUGUST 2004. The price is 27 Euros."

Available Functions (2/3)

4- book-german-train-journey

christoph berlin frankfurt 3304251200

"First Class Booking German Rail (Die Bahn): CHRISTOPH is booked on the 323 going from BERLIN to FRANKFURT at 17:11, 15, SEPTEMBER 2004. The price is 35 Euros."

5- book-austrian-train-journey

sinuhe vienna innsbruck 3304251200

"Austrian Rail (OBB): SINUHE is booked on the 367 going from VIENNA to INNSBRUCK at 16:47, 15, SEPTEMBER 2004. The price is 36 Euros. "

Available Functions (3/3)

6- book-student-european-train-journey

john london nice (3 4 6 18 8 2004)

"European Student Rail Travel: JOHN is booked on the 916 going from LONDON to NICE at 6:44, 18, AUGUST 2004. The price is 94 Euros. "

7- book-business-european-train-journey

liliana paris innsbruck (3 4 6 18 8 2004)

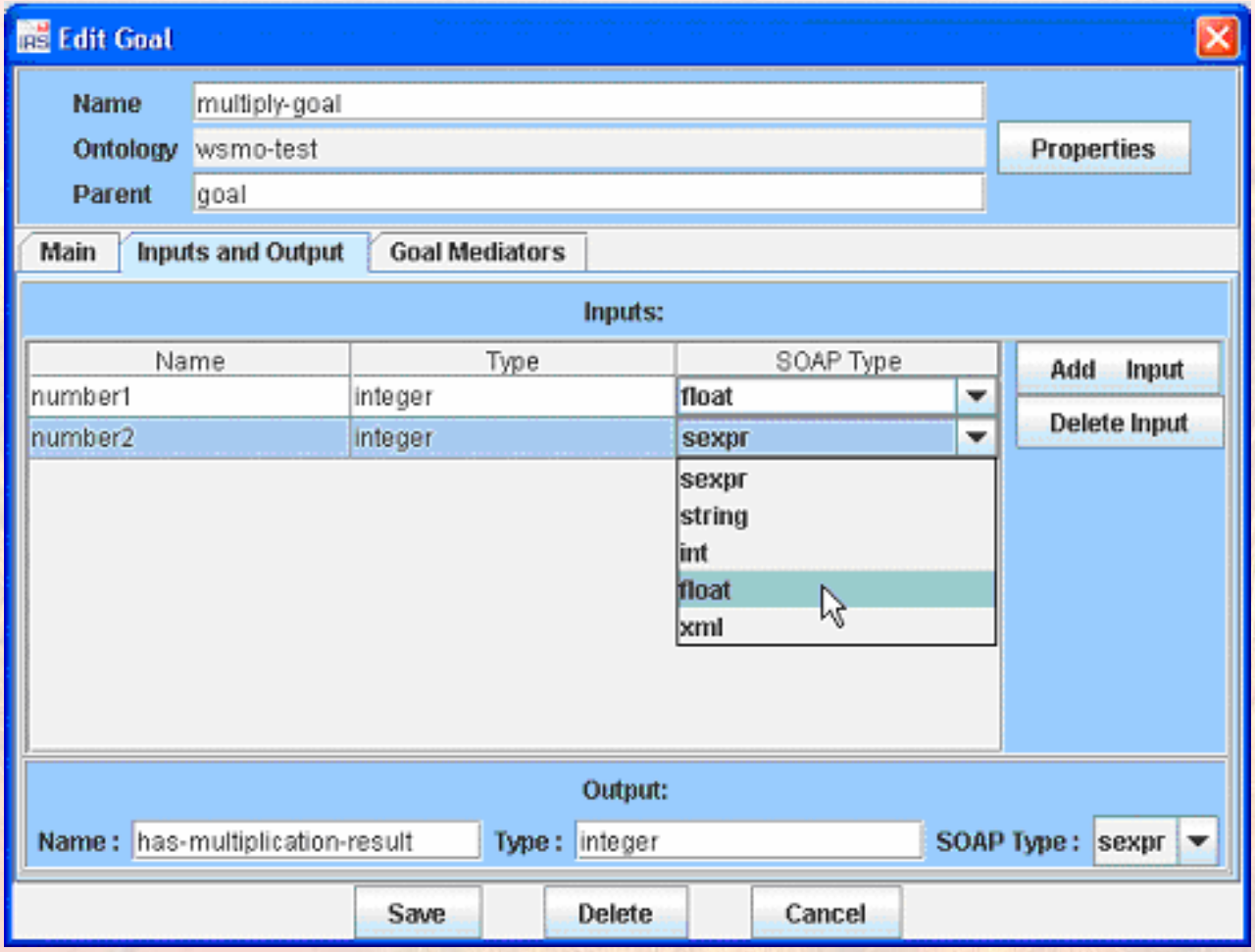
"Business Europe: LILIANA is booked on the 461 going from PARIS to INNSBRUCK at 6:12, 18, AUGUST 2004.
The price is 325 Euros."

8- mediate-time (lisp function) or JavaMediateTime/mediate (java)

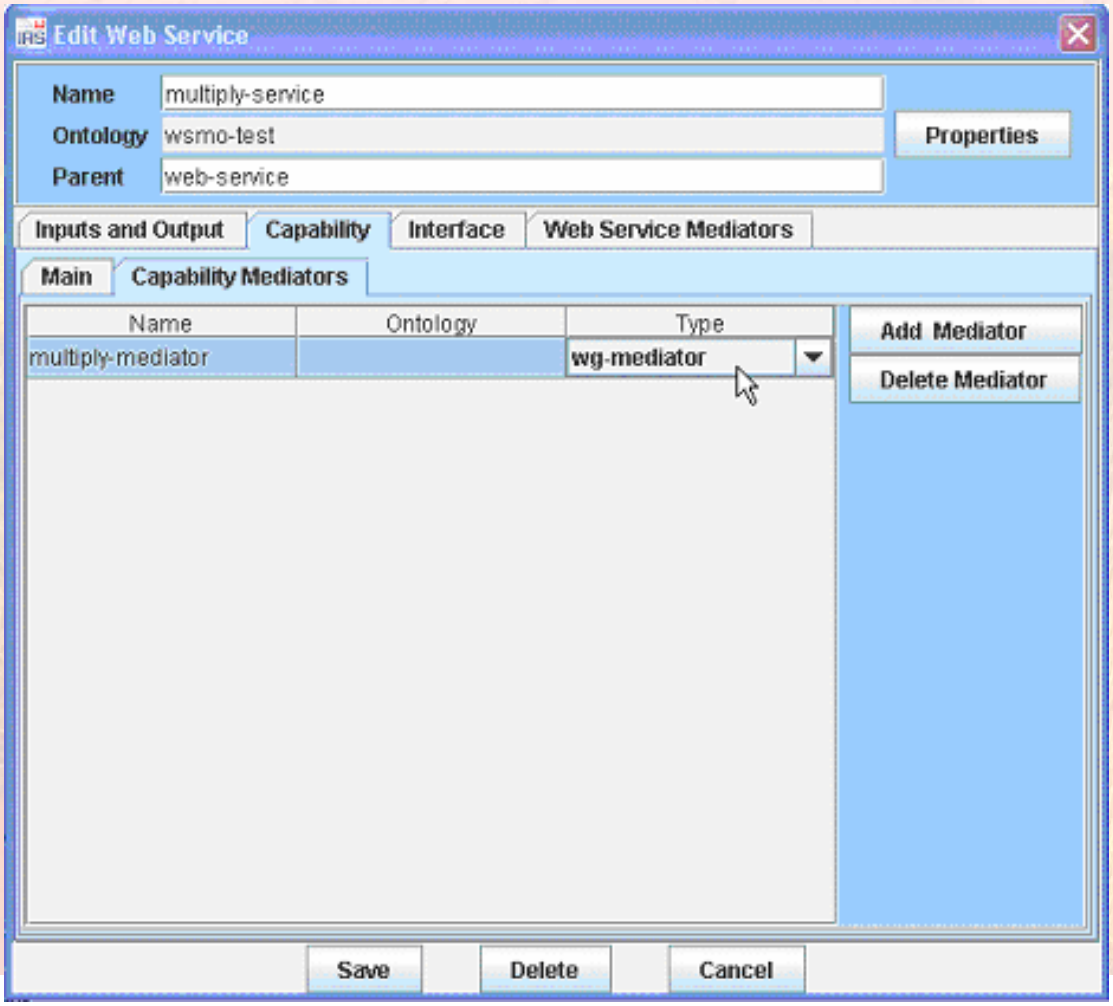
(9 30 17 20 9 2004)

3304686609

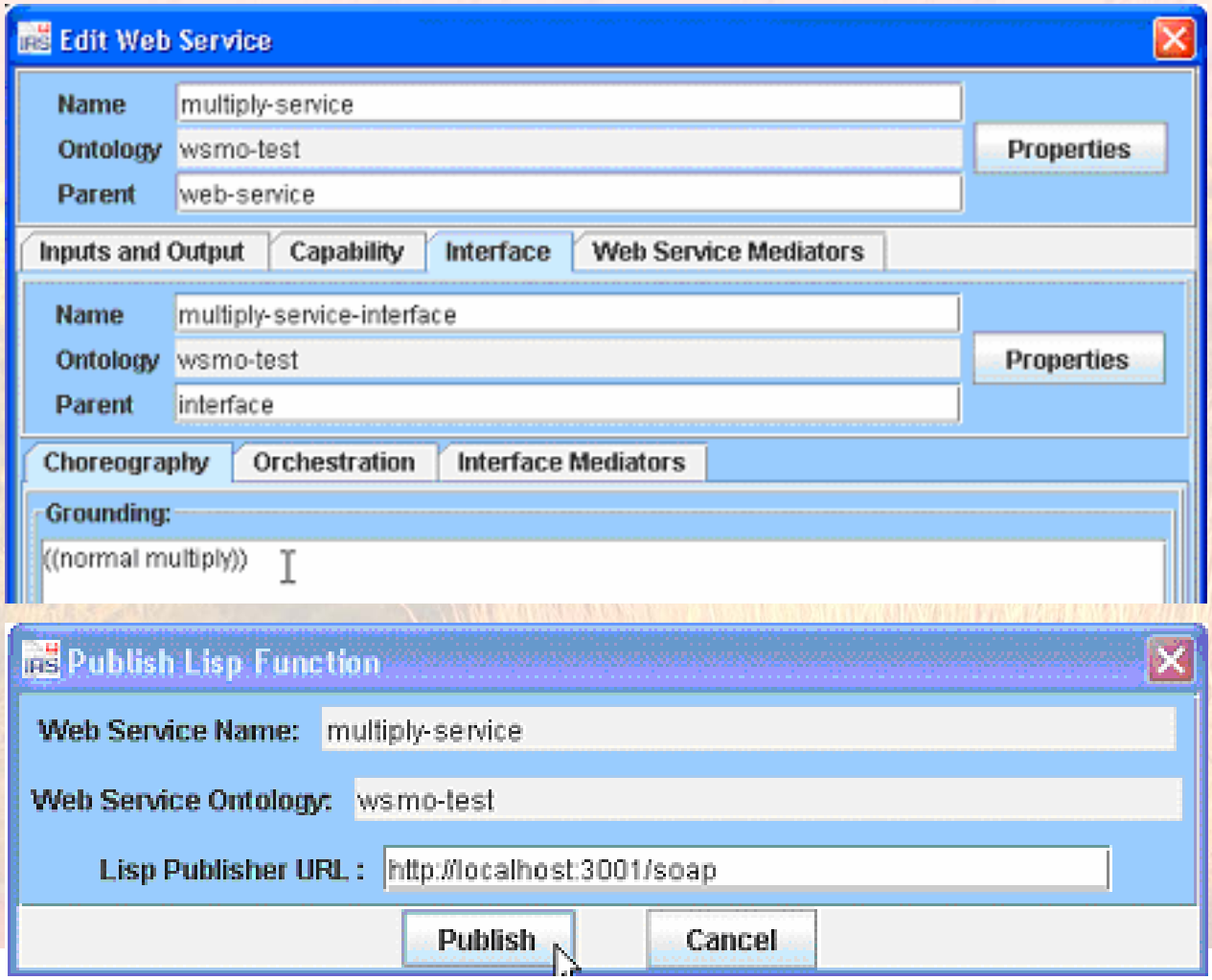
Example: Multiply Goal



Example: Multiply Web Service



Example: Publishing



Tips

- Order matters for input roles
  - Input roles in goal must match order of arguments to function
- Need to specify both input roles and output role
- Be careful with soap binding
  - sexpr as default
  - String for one line output
  - Use xml for multiple line output
- Input roles for web services inherited from goal
- Slot names can not be the same as class names
- Goal <-> web service linking mediator in the capability used mediators

Closing, Outlook, References, Acknowledgements

Tutorial Wrap-up

- The targets of the presented tutorial were to:
  - understand aims & challenges within Semantic Web Services
  - understand Semantic Web Service Frameworks:
    - aims, design principles, and paradigms
    - ontology elements & description
- an overview of Semantic Web Service techniques:
  - element description
  - discovery
  - choreography and service interoperability determination
  - orchestration and composition
- present WSMX a future Web Service based IT middleware
  - design and architecture
  - components design

=> you should now be able to correctly assess emerging technologies & products for Semantic Web Services and utilize these for your future work

Beyond WSMO

- Although WSMO (and OWL-S) are the main initiatives on Semantic Web services, they are not the only ones:
- Semantic Web Services Interest Group
  - Interest group founded at W3C to discuss issues related to Semantic Web Services (http://www.w3.org/2002/ws/swsig/)
  - Standardization Working Group in starting phase
- SWSI: International initiative to push toward a standardization of SWS (http://www.swsi.org)
- Semantic Web services are entering the main stream
  - UDDI is adopting OWL for semantic search
  - WSDL 2 will contain a mapping to RDF
  - The use of semantics is also discussed in the context of standards for WS Policies

SWSI (www.swsi.org)

- SWSI (Semantic Web Services Initiative) is becoming the point of synthesis of the SWS activity around the World
- SWSI includes many participants belonging to both academy and industry from the US and Europe
- SWSI is composed of two committees
  - SWSL which is expected to produce a language for Semantic Web services
  - SWSA which is expected to describe the architectural requirements for Semantic Web services
- OWL-S and WSMO are two main inputs, but contributions include IRS, Meteor-S

Semantics in the Main Stream

- Many WS standardization groups are realizing that they need to add semantic representation
- **UDDI v.next**
  - UDDI v.next is the new version of UDDI
  - UDDI TC has decided to use OWL as a standard language for the representation of business taxonomies
  - OWL-based inference will be used to improve WS search
- **Web Service Description Language v2**
  - The WSDL working group at W3C has decided to add an RDF mapping to WSDL 2
  - The RDF mapping may effectively provide a standard grounding mechanism for OWL-S and WSMO
- **Web Services policies proposals** require a significant amount of inference
  - There have been proposals to use OWL or SWRL as basic languages
  - Or to provide a mapping to semantic Web languages

References WSMO

- The central location where WSMO work and papers can be found is WSMO Working Group: http://www.wsmo.org
- WSMO languages
  - WSML Working Group http://www.wsml.org
- Web Service Execution Environment WSMX
  - WSMX working group : http://www.wsmx.org
  - WSMX open source can be found at: https://sourceforge.net/projects/wsmx/

References WSMO

- [WSMO Specification]: Roman, D.; Lausen, H.; Keller, U. (eds.): ***Web Service Modeling Ontology***, WSMO Working Draft D2, final version 1.2, 13 April 2005.
- [WSMO Primer]: Feier, C. (ed.): ***WSMO Primer***, WSMO Working Draft D3.1, 18 February 2005.
- [WSMO Choreography and Orchestration] Roman, D.; Scicluna, J., Feier, C. (eds.): ***Ontology-based Choreography and Orchestration of WSMO Services***, WSMO Working Draft D14, 01 March 2005.
- [WSMO Use Case] Stollberg, M.; Lausen, H.; Polleres, A.; Lara, R. (ed.): ***WSMO Use Case Modeling and Testing***, WSMO Working Drafts D3.2; D3.3.; D3.4; D3.5, 05 November 2004.
- [WSML] de Bruijn, J. (Ed.): ***The WSML Specification***, WSML Working Draft D16, 03 February 2005.

References OWL-S

- The main repository of papers on OWL-S is at http://www.daml.org/services/owl-s/pub-archive.html that contains many papers produced by the coalition as well as from the community at large

- The main source of information on OWL-S is the Web site http://www.daml.org/services/owl-s

- The rest of this section will report what we believe to be the most influential papers on OWL-S as well as paper referred in this tutorial

References

Overview

- Arroyo, S., Lara, R., Gomez, J. M., Berka, D., Ding, Y. and Fensel, D: "Semantic Aspects of Web Services" in Practical Handbook of Internet Computing. Munindar P. Singh, editor. Chapman Hall and CRC Press, Baton Rouge. 2004.
- Tim Berners-Lee, James Hendler, and Ora Lassila, “The Semantic Web”. Scientific American, 284(5):34-43, 2001.
- Chen, W., Kifer, M., and Warren, D. S. (1993). HILOG: A foundation for higher-order logic programming. Journal of Logic Programming, 15(3):187-230.
- Domingue, J. Cabral, L., Hakimpour, F., Sell D., and Motta, E., (2004) IRS-III: A Platform and Infrastructure for Creating WSMO-based Semantic Web Services WSMO Implementation Workshop (WIW), Frankfurt, Germany, September,2004
- Dieter Fensel, “Ontologies: Silver Bullet for Knowledge Management and Electronic Commerce”, Springer-Verlag, Berlin, 2001.
- Stollberg, M.; Feier, C.; Roman, D.; Fensel, D.: Semantic Web Services – Concepts and Technology. In Proceedings of the EUROLAN Summer School, 2005 (to appear).

References

Ontology & Languages

- [Gruber, 1993] Thomas R. Gruber, “A Translation Approach to Portable Ontology Specifications”, Knowledge Acquisition, 5:199-220, 1993.
- [Grosof et al., 2003] Grosof, B. N., Horrocks, I., Volz, R., and Decker, S. (2003). Description logic programs: Combining logic programs with description logic. In Proc. Intl. Conf. on the World Wide Web (WWW-2003), Budapest, Hungary.
- [Kifer et al., 1995] Kifer, M., Lausen, G., and Wu, J. (1995). Logical foundations of object-oriented and frame-based languages. JACM, 42(4):741-843.
- [Pan and Horrocks, 2004] Pan, J. Z. and Horrocks, I. (2004). OWL-E: Extending OWL with expressive datatype expressions. IMG Technical Report IMG/2004/KR-SW-01/v1.0, Victoria University of Manchester. Available from http://dl-web.man.ac.uk/Doc/IMGTR-OWL-E.pdf.
- [Stencil Group] - www.stencilgroup.com/ideas_scope_200106wsdefined.html

References

Discovery

B. Benatallah, M. Hacid, C. Rey, F. Toumani **Towards Semantic Reasoning for Web Services Discovery**,. *In Proc. of the International Semantic Web Conference (ISWC 2003), 2003*

Herzog, R.; Lausen, H.; Roman, D.; Zugmann, P.: **WSMO Registry.** WSMO Working Draft D10 v0.1, 26 April 2004.

Keller, U.; Lara, R.; Polleres, A. (Eds): **WSMO Web Service Discovery**. WSML Working Draft D5.1, 12 Nov 2004.

Keller, U.; Lara, R.; Lausen, H.; Polleres, A.; Fensel, D.: **Automatic Location of Services**. In Proc. of the 2nd European Semantic Web Symposium (ESWS2005), Heraklion, Crete, 2005.

M. Kifer, R. Lara, A. Polleres, C. Zhao, U. Keller, H. Lausen and D. Fensel: **A Logical Framework for Web Service Discovery.** Proc. 1st. Intl. Workshop SWS'2004 at ISWC 2004,Hiroshima, Japan, November 8, 2004, CEUR Workshop Proceedings, ISSN 1613-0073

Lara, R., Lausen, H.; Toma, I.: (Eds): **WSMX Discovery**. WSMX Working Draft D10 v0.2, 07 March 2005.

Lei Li and Ian Horrocks. **A software framework for matchmaking based on semantic web technology**. In *Proc. of the Twelfth International World Wide Web Conference (WWW 2003), 2003.*

References

Discovery

Lei Li and Ian Horrocks. **A software framework for matchmaking based on semantic web technology**. In *Proc. of the Twelfth International World Wide Web Conference (WWW 2003), 2003*

Daniel J. Mandell and Sheila A. McIlraith. **Adapting BPEL4WS for the Semantic Web: The Bottom-Up Approach to Web Service Interoperation.** In *Proceedings of the Second International Semantic Web Conference (ISWC2003),*

Massimo Paolucci, Takahiro Kawamura, Terry R. Payne, Katia Sycara; **Importing the Semantic Web in UDDI**. In *Proceedings of Web Services, E-business and Semantic Web Workshop, 2002*

Massimo Paolucci, Takahiro Kawamura, Terry R. Payne, Katia Sycara; "**Semantic Matching of Web Services Capabilities**." In *Proceedings of the 1st International Semantic Web Conference* (ISWC2002), 2002

Preist, C.: **A Conceptual Architecture for Semantic Web Services**. In Proceedings of the 3rd International Semantic Web Conference (ISWC 2004), 2004, pp. 395 - 409.

Stollberg, M.; Keller, U.; Fensel. D.: **Partner and Service Discovery for Collaboration on the Semantic Web.** Proc. 3rd Intl. Conference on Web Services (ICWS 2005), Orlando, Florida, July 2005.

References

Choreography, Orchestration, Mediation

[WSMO Service Interfaces Description]: Roman, D.; Scicluna, J.; Feier, C. (eds.): ***Ontology-based Choreography and Orchestration for WSMO Web Services***, WSMO Working Draft D14, final version 0.1, 1 March 2005.

[WSMO Mediators]: Scharffe, F. (ed.): ***WSMO Mediators***, WSMO Working Draft D29, 24 June 2005.

[WSMX Data Mediation]: Mocan, A. (ed.): ***WSMX Data Mediation***, WSMX Working Draft D13.3, 16 May 2005.

[WSMO Process Mediation]: Cimpian, E.; Mocan, A. (ed.): ***Process Mediation in WSMX***, WSMX Working Draft D13.7, 16 May 2005.

Stollberg, M.: **Reasoning Tasks and Mediation on Choreography and Orchestration in WSMO**. In *Proceedings of the 2nd International WSMO Implementation Workshop (WIW 2005)*, Innsbruck, Austria, 2005.

de Bruijn, J. and Polleres, A.. **Towards an Ontology Mapping Specification Language for the Semantic Web**. DERI Technical Report 2004-06-30, 2004.

Cimpian, E. and Mocan, A. **WSMX Process Mediation Based on Choreographies.** In *Proceedings of the 1st International Workshop on Web Service Choreography and Orchestration for Business Process Management at the BPM 2005, Nancy, France, 2005*

Michael Altenhofen, Egon Börger and Jens Lemcke: **An Execution Semantics for Mediation Patterns.** In *Proceedings of the 2nd International WSMO Implementation Workshop (WIW 2005)*, Innsbruck, Austria, 2005.

References WSMX

- The central location where WSMX work, papers, and software can be found is the WSMX working group homepage: http://www.wsmx.org.
- The main documents are:
  - Conceptual Model (http://www.wsmo.org/2004/d13/d13.1/v0.3/)
  - Architecture (http://www.wsmo.org/TR/d13/d13.4/v0.2/)
  - Implementation: open source at http://sourceforge.net/projects/wsmx
  - Documentation (http://www.wsmo.org/TR/d22/v0.2/)
  - Execution Semantics (http://www.wsmo.org/TR/d13/d13.2/)
  - WSMX Toolkit (http://www.wsmo.org/TR/d9/d9.1/v0.2/)
- Further Readings:

Bussler, C. (2003): B2B Integration. Berlin, Heidelberg: Springer.

Haselwanter, T.; Zaremba, Ma.., Zaremba Mi.: *Enabling Components Management and Executions Semantics in WSMX*. In Proceedings of the 2nd International WSMO Implementation Workshop (WIW 2005), Innsbruck, Austria, June 2005.

Zaremba, M. and Bussler, C.: *Towards Dynamic Execution Semantics in Semantic Web Services.* In Proceedings of the WWW 2005 Workshop on Web Service Semantics: Towards Dynamic Business Integration, 2005.

References IRS III tutorial

- J. Domingue, L. Cabral, F. Hakimpour,D. Sell and E. Motta: IRS-III: A Platform and Infrastructure for Creating WSMO-based Semantic Web Services. Proceedings of the Workshop on WSMO Implementations (WIW 2004) Frankfurt, Germany, September 29-30, 2004, CEUR Workshop Proceedings, ISSN 1613-0073, online http://CEUR-WS.org/Vol-113/paper3.pdf.

- J. Domingue and S. Galizia: Towards a Choreography for IRS-III.
- Proceedings of the Workshop on WSMO Implementations (WIW 2004) Frankfurt, Germany, September 29-30, 2004, CEUR Workshop Proceedings, ISSN 1613-0073, online http://CEUR-WS.org/Vol-113/paper7.pdf.

- Cabral, L., Domingue, J., Motta, E., Payne, T. and Hakimpour, F. (2004).
- Approaches to Semantic Web Services: An Overview and Comparisons. In proceedings of the First European Semantic Web Symposium (ESWS2004);
- 10-12 May 2004, Heraklion, Crete, Greece.

- Motta, E., Domingue, J., Cabral, L. and Gaspari, M. (2003) IRS-II: A Framework and Infrastructure for Semantic Web Services. In proceedings of the 2nd International Semantic Web Conference (ISWC2003) 20-23 October 2003, Sundial Resort, Sanibel Island, Florida, USA.

Acknowledgements

The WSMO work is funded by the European Commission under the projects **DIP**, **Knowledge Web**, **SEKT**, **SWWS**, **AKT** and **Esperonto**; by **Science Foundation Ireland** under the **DERI-Lion** project; and by the Austrian government under the **FIT-IT** program.

We would like to thank to all the members of the **WSMO**, **WSML**, and **WSMX** working groups for their advice and input into this tutorial. We dedicate special thanks to Chris Bussler, Michal Zaremba, François Scharffe, Adrian Mocan, and Emilia Cimpian for participation and input during preparation of the tutorial.