

# TripCom and Semantic Web Services



WP4

October 9, 2006

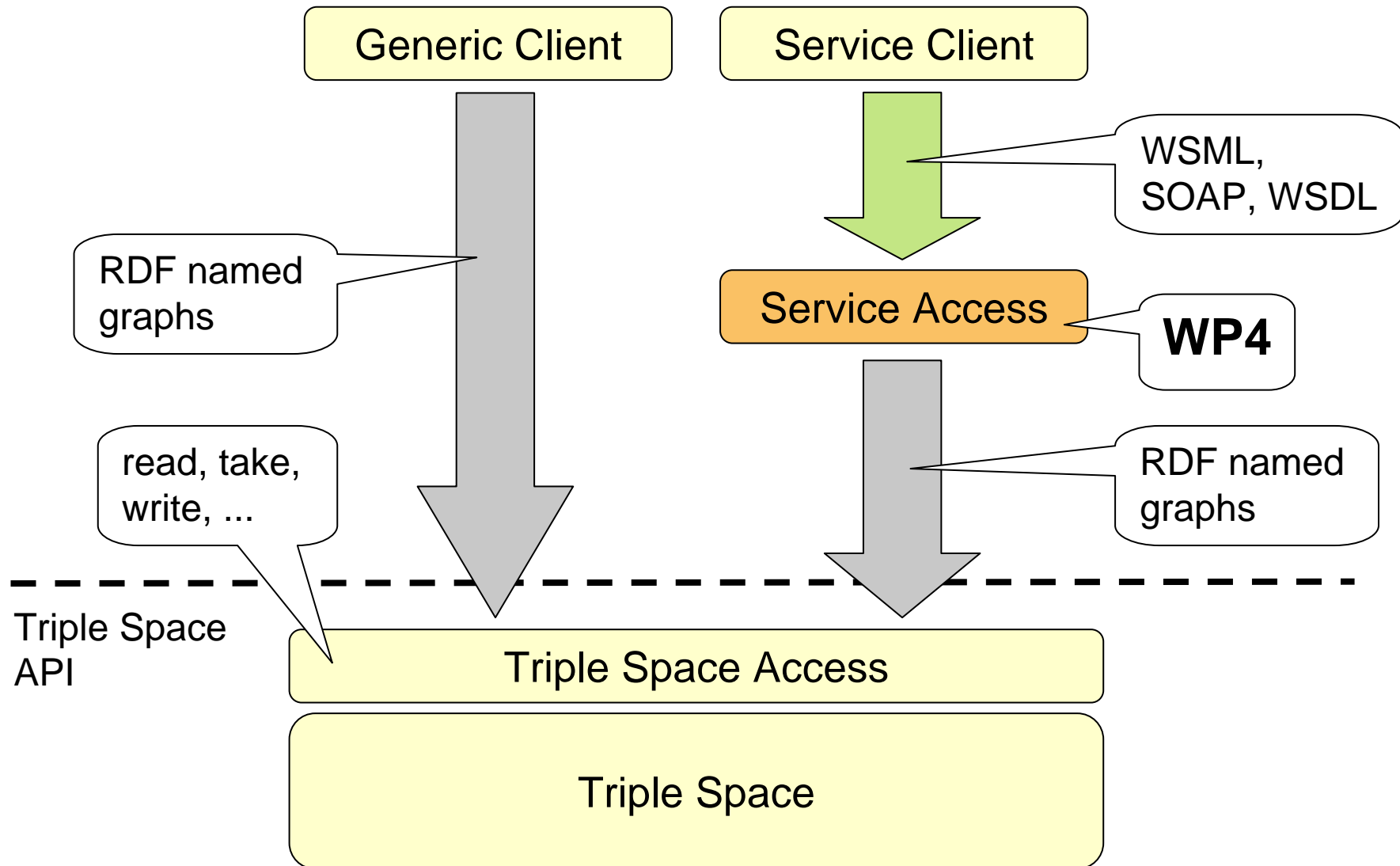


- Introduction
  - Role of WP4 in TripCom
- SOA and Web services
  - Service oriented computing
  - Service Bus
- Use cases for TripCom
  - In Semantic Web Services
  - In the context of WSMX
- Requirements

# Introduction

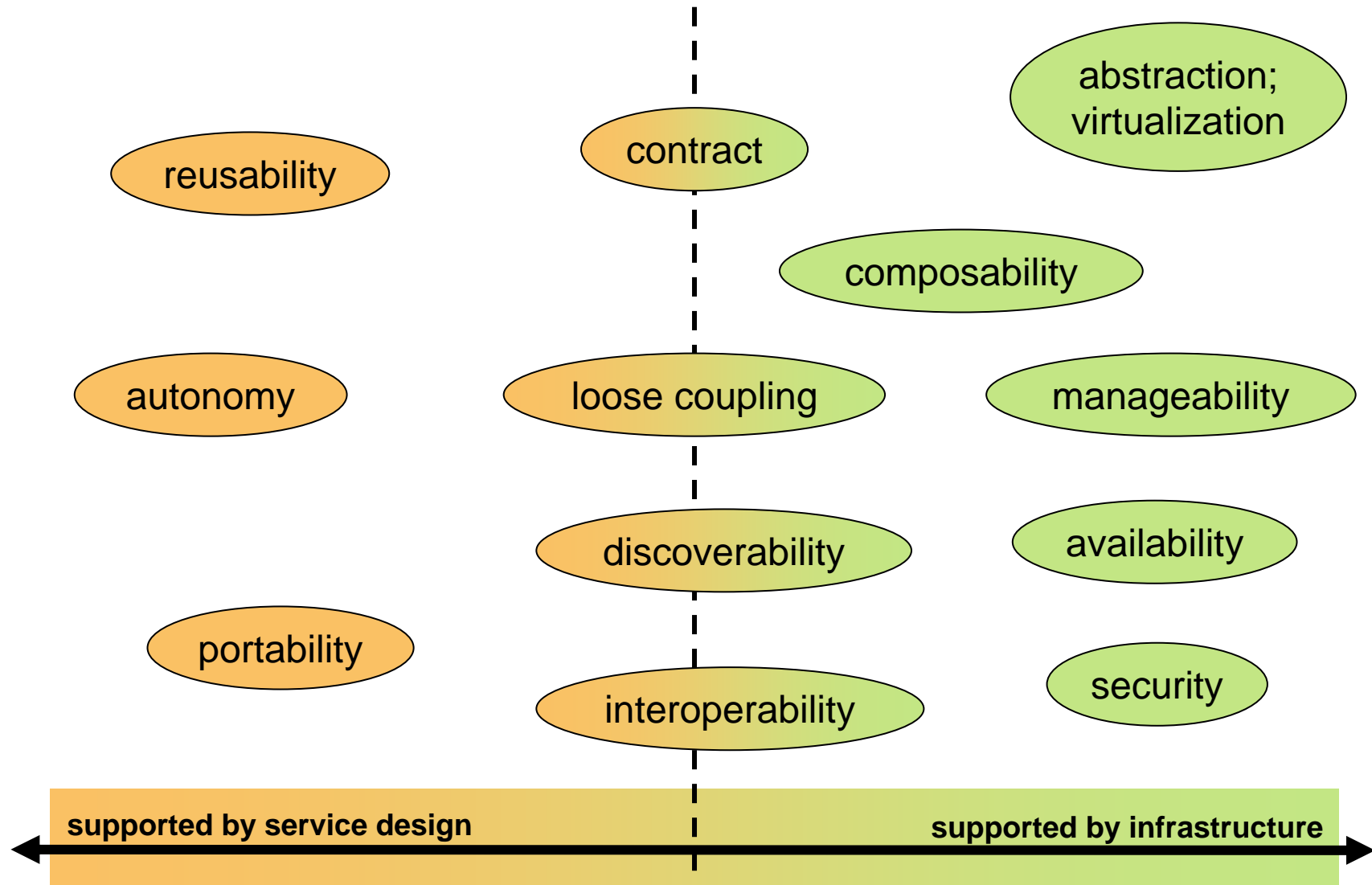
- "Integration of TripCom with Semantic Web Services"
- WP4 provides an interface for "service-style" interaction, which is complementary to the lower level TS API (read, write, take, ...)
  - Provides service clients with **essential SOA and WS features**
    - Publication and discovery of service descriptions
    - Binding mechanism
    - Invocation of service implementations
- Integration of TripCom and WSMX is the **key deliverable** of WP4

# Role of WP4 in TripCom (2)

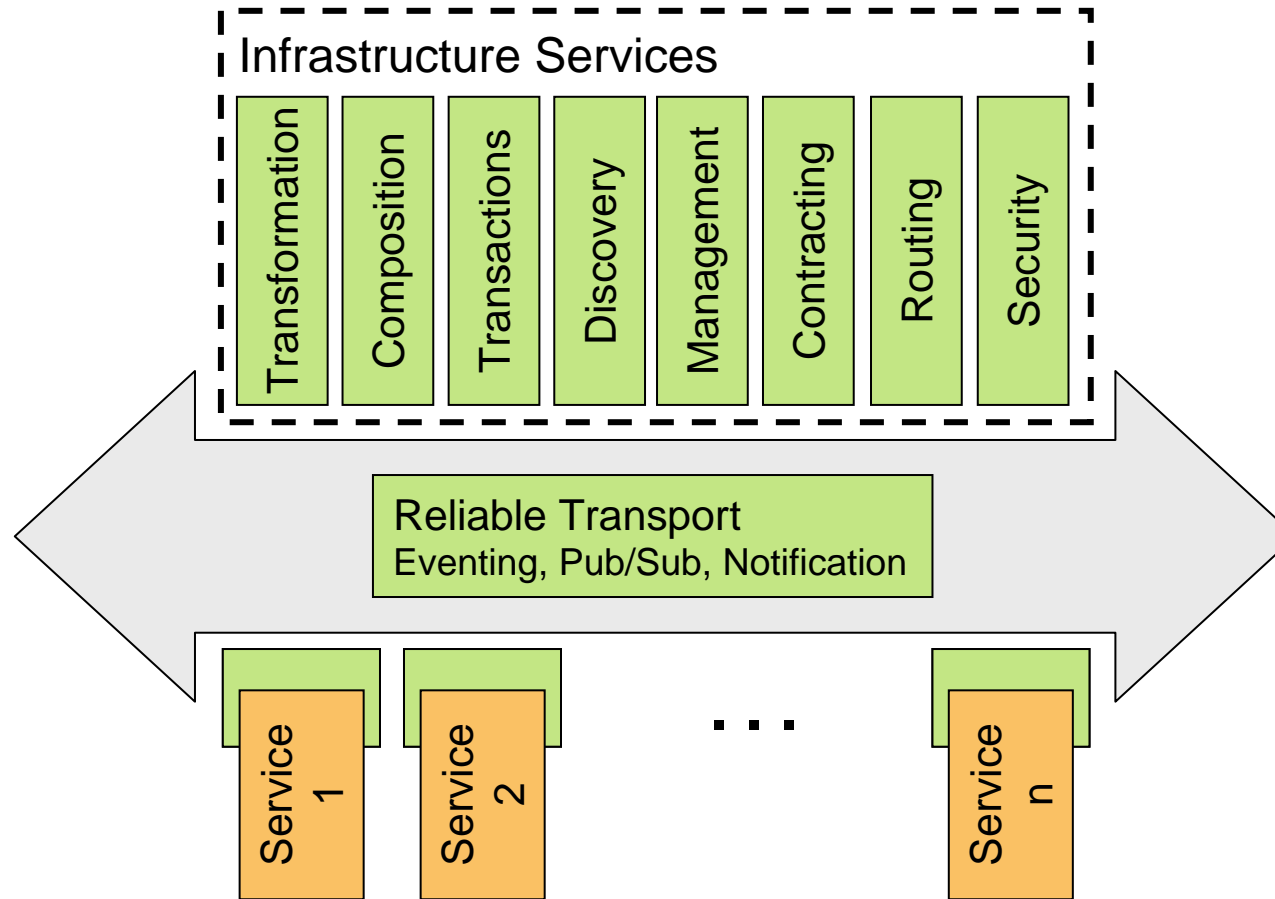


# SOA and Web services

# Service oriented computing



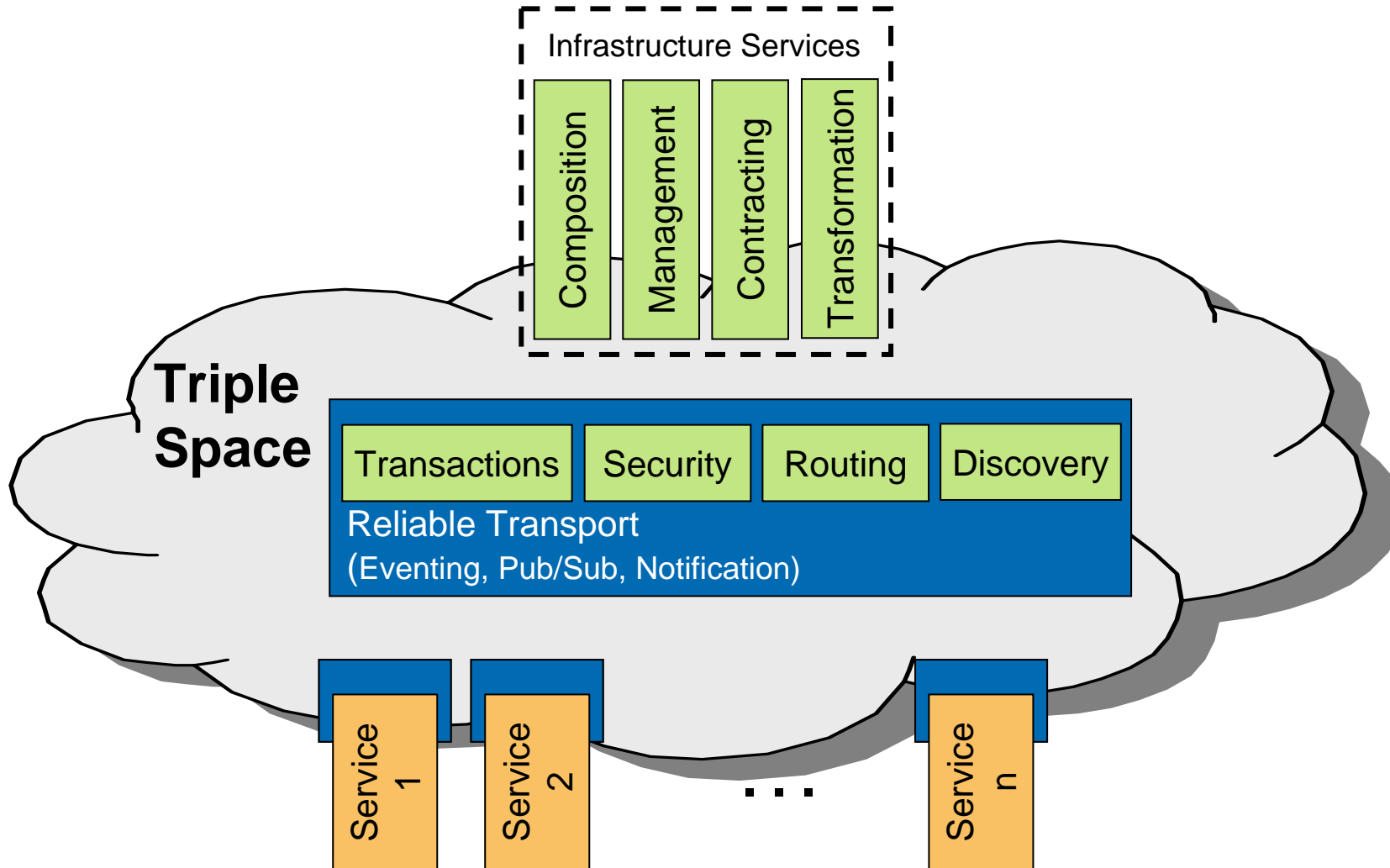
based on <http://www.serviceorientation.org>



"middleware platform for realizing service oriented computing based on Web service standards"

- Communication bus is realized through **message-oriented middleware (MOM)**
  - Reliable and asynchronous message-based communication
  - Communication partners are wired through "communication channels"
    - Queues
    - Topics
  - Syntactic message transformation (e.g. XSLT)
  - Messages are explicitly routed to their destination via message routers

- Communication through **persistent storage** of **semantic data**
  - Middleware understands semantics of "messages"
  - Semantic transformation of messages
- Support for "**repeatable read**" pattern
  - Information is published to the space
  - Multiple requesters can non-destructively consume the information
- Random access
- **Content-based** access (using semantics)
- Administrative effort
  - Some MOM-based implementations require cumbersome setup and wiring of middleware components
  - This is different in the space approach: now **pull instead of push** as in messaging
    - clients know the information they want



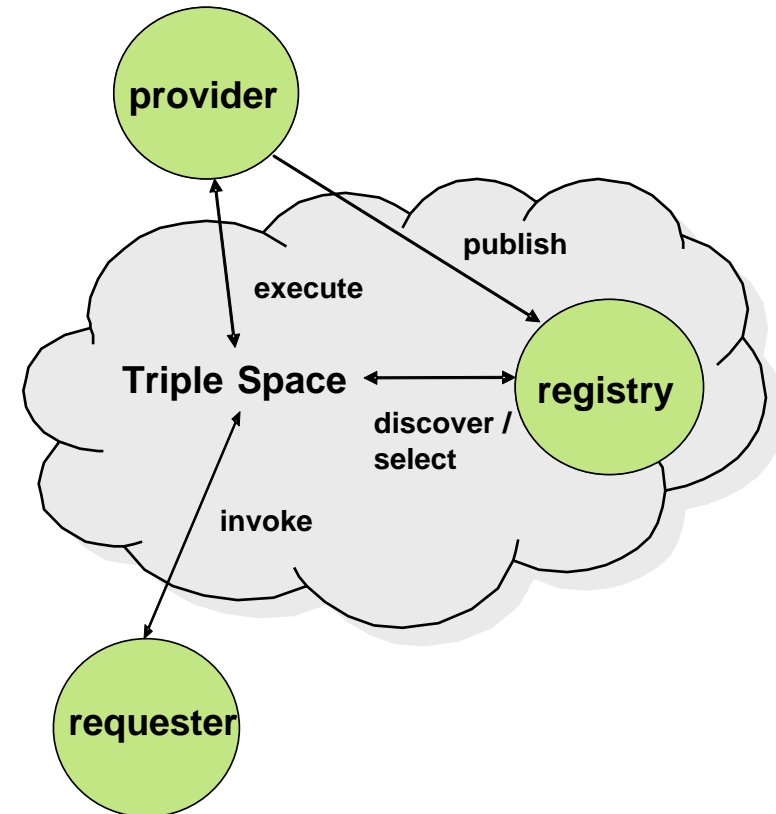
# Use Cases for TripCom in Semantic Web Services

- TripCom provides new underpinning for a "Semantic" Service Bus
  - I. Semantically enriched **SOA features**
    - publish, discover, bind, execute
  - II. **Communication** platform for reliable and asynchronous Web service communication
    - Web service binding
  - III. **Repository** for Web service metadata
    - WSML
    - (SA-)WSDL
    - (abstract) BPEL
    - Policies
    - ...

- Goal
  - Provide a platform for **Semantic Web Services interaction**
- Benefit
  - Semantic discovery/selection of
    - Service implementations
    - Service interfaces
  - Persistent and reliable Web service interaction
- What needs to be done?
  - TripCom-based implementation of essential SOA features
    - Semantic description of service interfaces
    - Publication of service interface descriptions
    - Discovery/selection of service interfaces and implementations
    - Service invocation

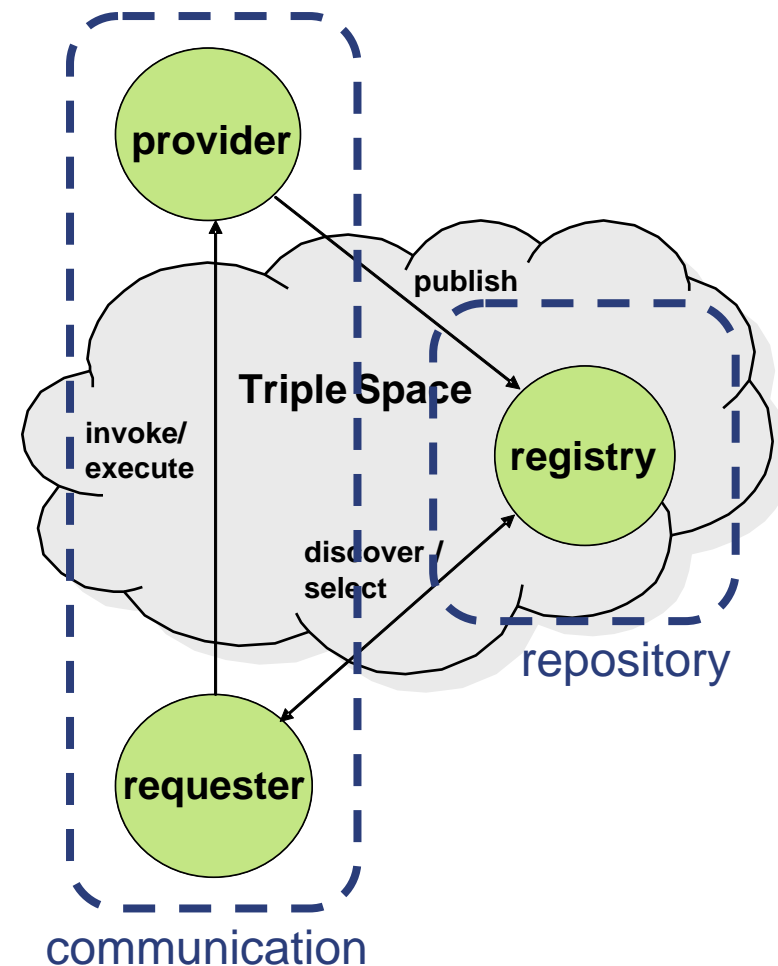
# I. Implicit service discovery/selection

- Actual idea behind a service bus (“**virtualization**”)
- Service requester **just invokes a goal** through TripCom's Service API
  - TripCom performs service discovery, selection
  - based on functionality provided by WSMX
    - WSMX Integration API
- Similar to WSMX's **invokeGoal** entry point



# I. Explicit service discovery/selection

- Service requester **explicitly calls**
  - **discover**
  - **select**
  - **invoke**
- Service requester can decide which functionality provided by TripCom to use



- Goal
  - Provide platform for **persistent and asynchronous Web service communication** between service requester and service provider
- Benefit
  - Transport is inherently asynchronous and persistent, based on publication + subscription/polling
  - Semantics can be used e.g. for message transformation and message routing
  - Reliable messaging is a key technology in EAI
- What needs to be done?
  - Definition and implementation of a **TripCom binding for Web services**

- Binding specifies
  - Message format and serialization
  - Message exchange patterns
  - Mapping to transport protocol, which includes
    - Endpoint addressing
    - Operation and parameter mapping
- E.g. SOAP/HTTP, POJO/JMS
- **"TripCom" binding: RDF/TripleSpace**
- Benefit
  - "not leaving the semantic layer" - Annex

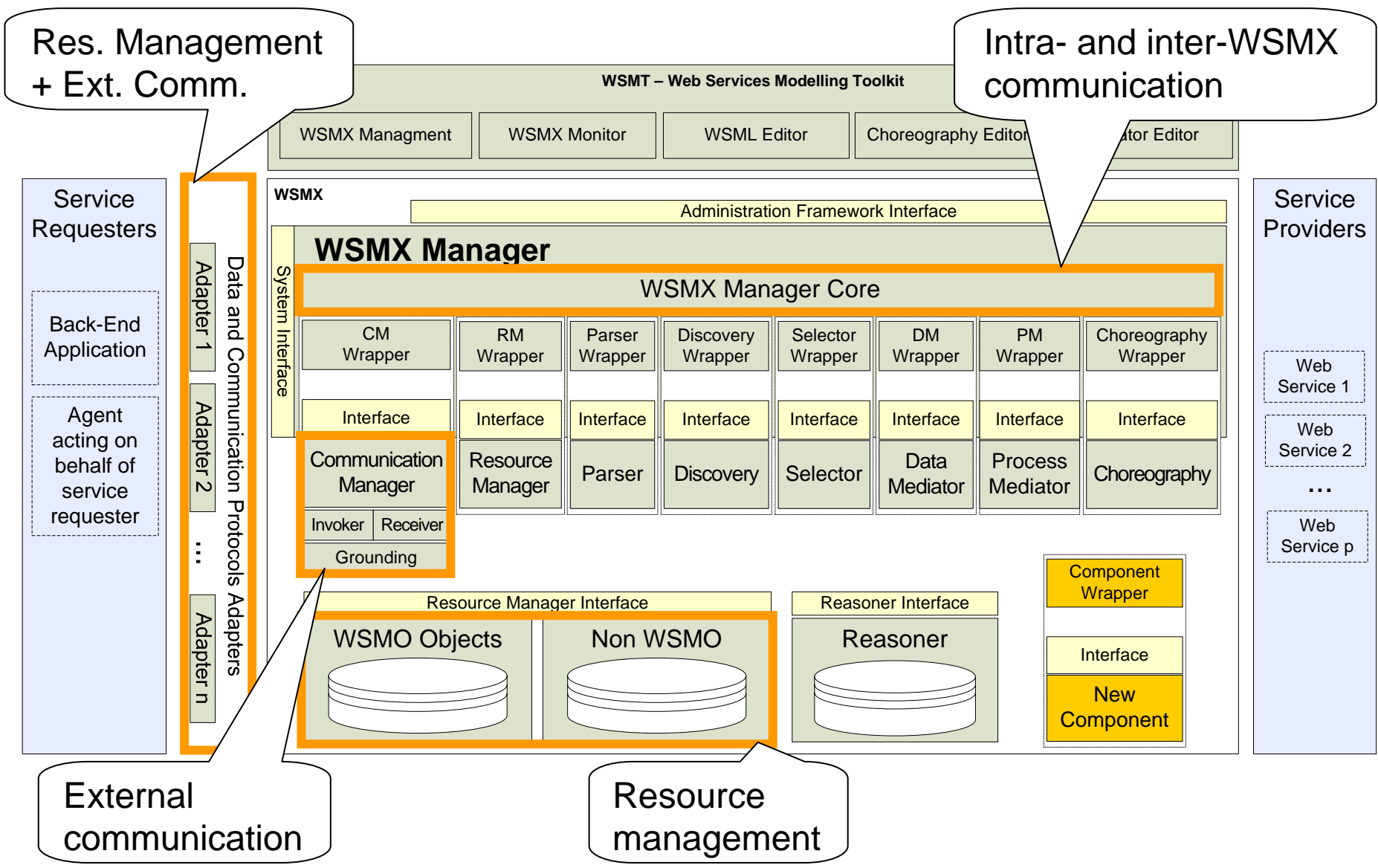
- Goal
  - TripCom can be used as a **repository** (registry) for Web Service metadata, e.g.
    - (SA-)WSDL
    - WSML
    - (abstract) BPEL
    - Policy
- Benefit
  - Semantic query mechanisms
  - Web-scale repository
- What needs to be done?
  - Provide mappings to and from RDF
  - Provide mechanisms for
    - Storage of service metadata
    - Querying and retrieval

# Use Cases for TripCom in WSMX

- Web Services Modeling Execution Environment
- Reference Implementation of the WSMO conceptual model
- Loosely coupled components
  - Discovery
  - Selection
  - Composition
  - Data and Process Mediation
  - WSMX Manager
  - And more ...
- Definition of component interfaces  
(→ integration API)

- **Communication** platform (refinement of WS Communication Use-Case)
  - I. Communication and coordination within WSMX (internal comm. between WSMX components)
  - II. Communication between service requester, WSMX, and service provider (external communication)
  - III. Communication between WSMX systems
- **Resource management** for persistent storage (refinement of WS Repository Use Case)
  - IV. Repository for Web Services, Goals, Mediators and Ontologies

# WSMX and TripCom



# Requirements

- retrieval operations (blocking or non-blocking)
  - based on tuple content
  - based on unique triple/graph identifier
  - based on context (metadata)
- storage operations
  - explicit sub-space definition
  - implicit sub-space definition
- notification/subscription mechanisms
  - ordered message delivery to support "traditional" messaging
- management
  - creation and deletion of spaces
- transactions
  - local transactions → e.g. emulation of *TSpace*-like multiwrite operation
  - distributed transactions → e.g. remote service invocation

**End of Document**