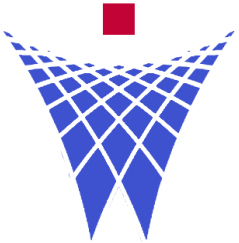


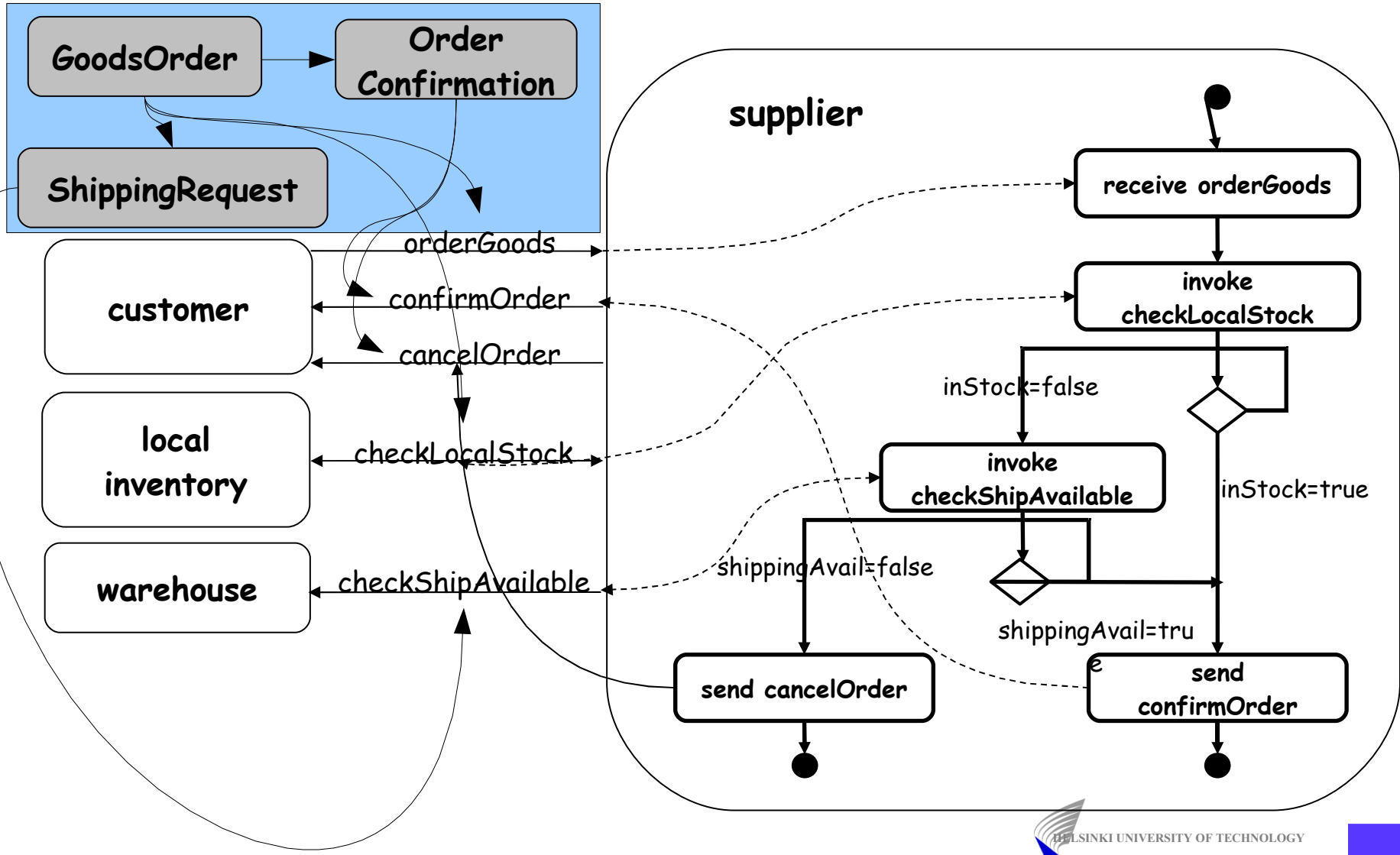
Need for Semantics

WSM[O/L/X]

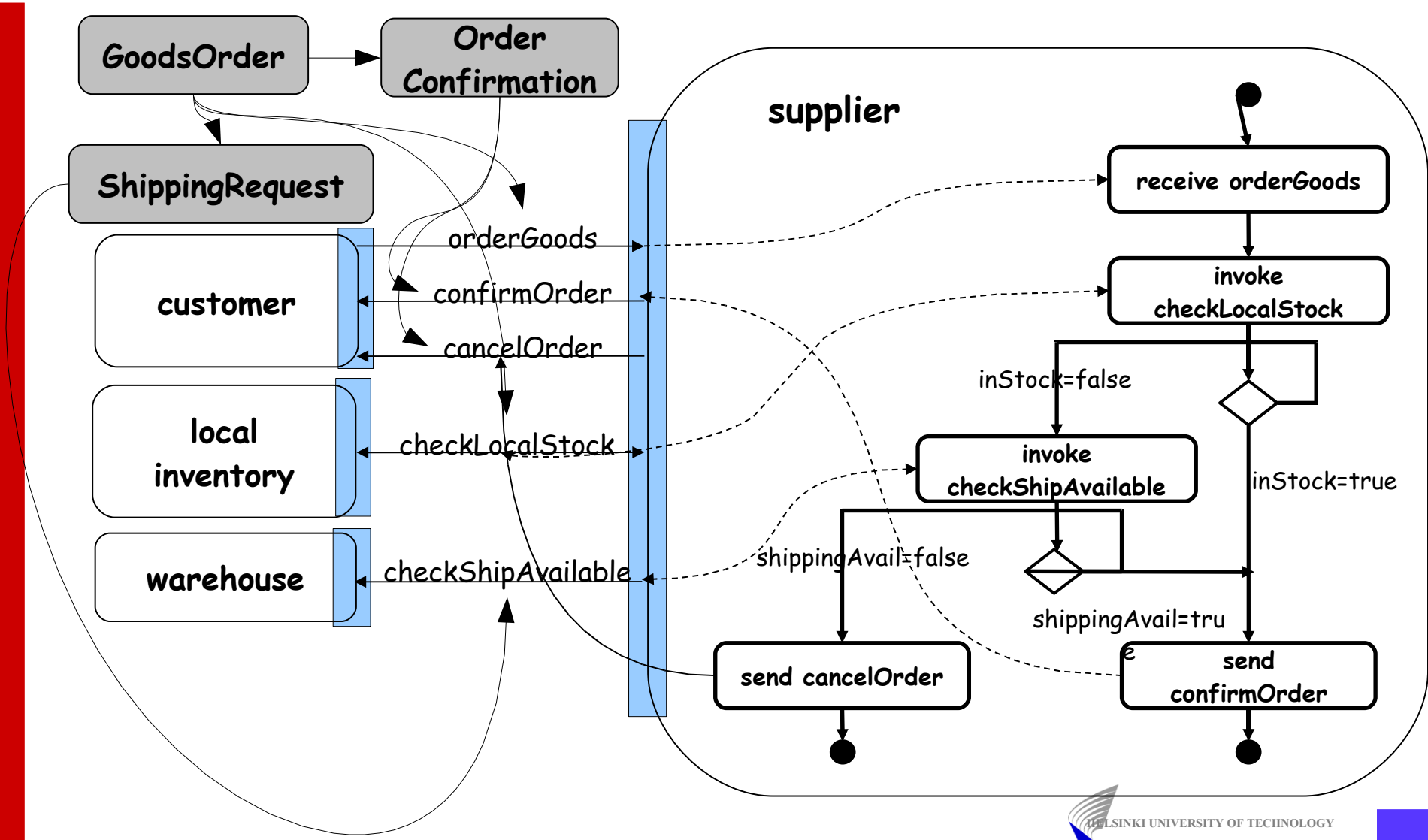
Eetu Mäkelä

Semantic Computing Research Group
http://www.seco.tkk.fi/

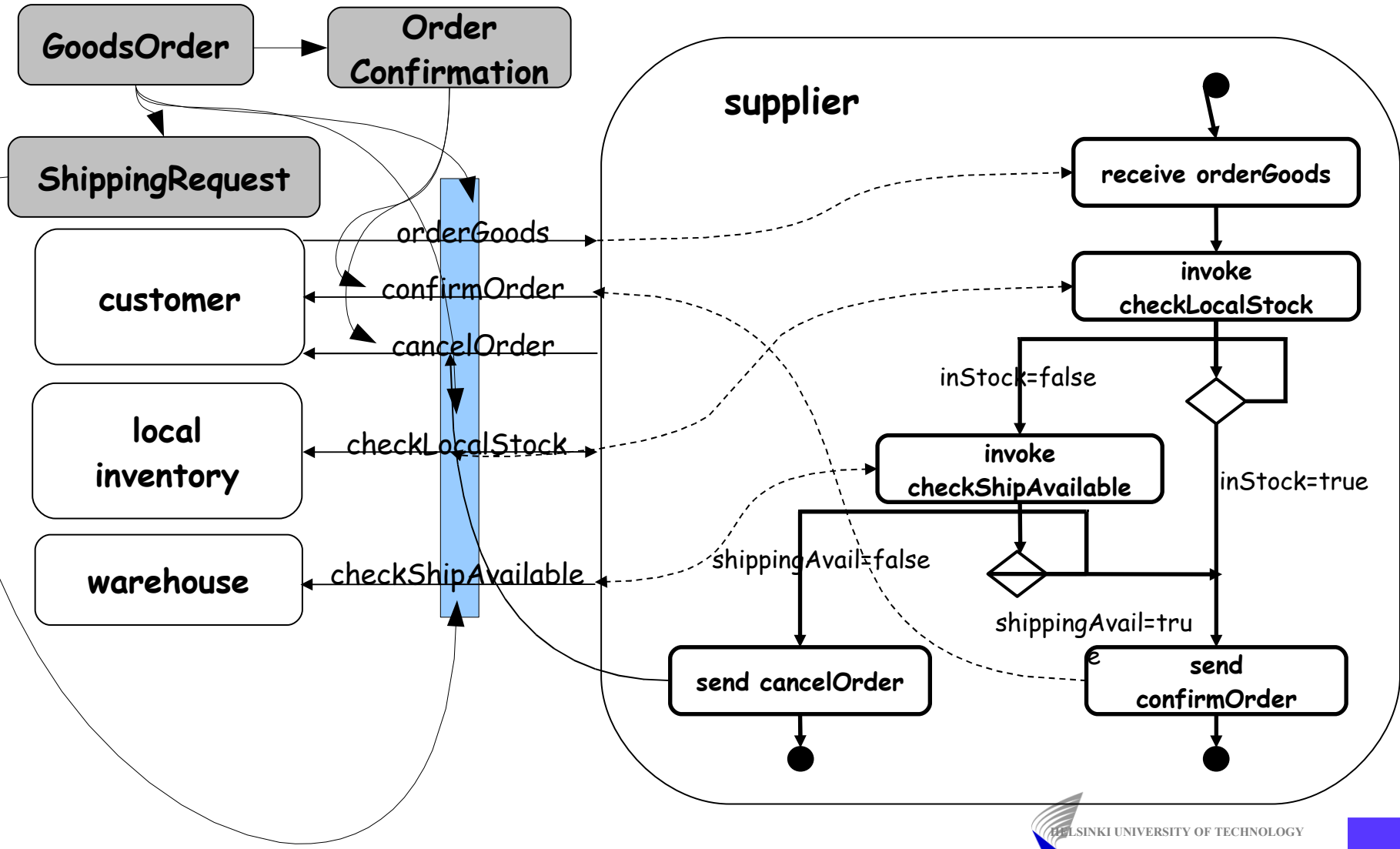
What is needed for minimizing the cost of connecting service modules and mazimizing reuse: Common Data Encoding, Data Semantics



Needed: Common Interfaces, Service Semantics



Needed: Common communication protocols and patterns



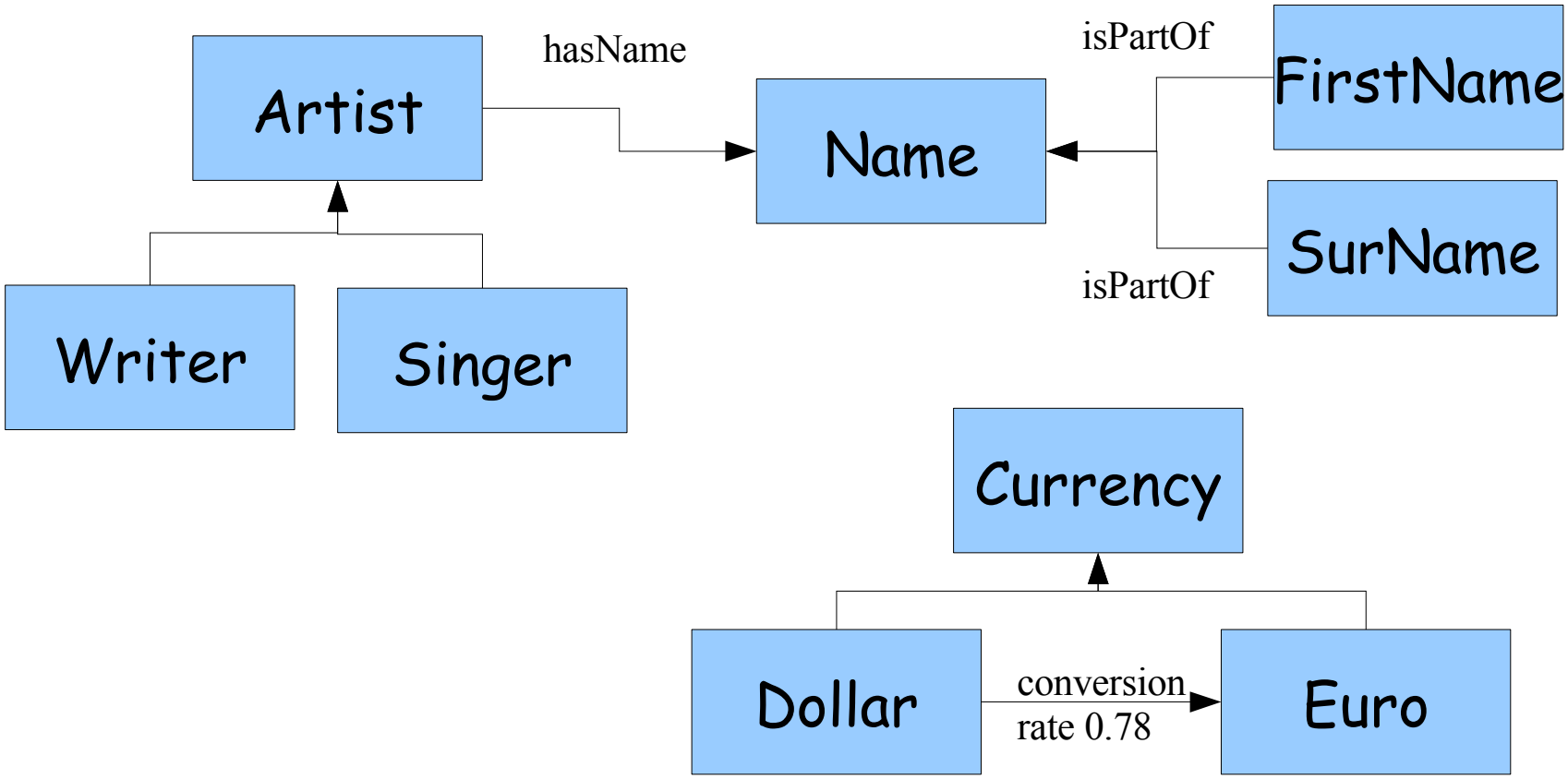
Web Services are a global effort to standardize what is needed

- Common data encoding: XML, WSDL
- Common data semantics: (WSDL)
- Common communications protocols: SOAP, WS-ReliableMessaging, WS-Security
- Common interfaces: WSDL, WS-Policy
- Common service semantics: (WSDL) (WS-Transactions), (WS-Security), (WS-Policy)

The Need for Semantics: Data Semantics

- Suppose we have the data object “Book” that has the fields “Writer name”, “Book name” and “Price (USD)”
- What happens if someone else expects an object with fields “Artist first name”, “Artist surname”, “Work name”, and “Price (EUR)”?
- If the fields were semantically annotated, common translators could be used to automatically map data objects to each other

With ontologies (defining classes and relationships) and transformation rules, it is possible to encode data semantics in a common manner, and translate between encodings



Name:=concat(FirstName, SurName)
amountIn(Euro):=conversionRate(X,Euro)*amountIn(X)

Description Logics (OWL) vs Horn Logics (SWRL, Prolog etc)

- Description logics describe restrictions about classes and their relationships between each other
- Horn logics describe what follows if certain prequisites are true
- Neither is a subset of the other
- It's impossible to assert that persons who study and live in the same city are “home students” in OWL
  - This can be done easily using rules: studies(X,Y), lives(X,Z), loc(Y,U), loc(Z,U) -> homeStudent(X)
- Rules cannot assert the information that a person is either a man or a woman
  - This information is easily expressed in OWL: Man (class) disjointWith Woman (class)

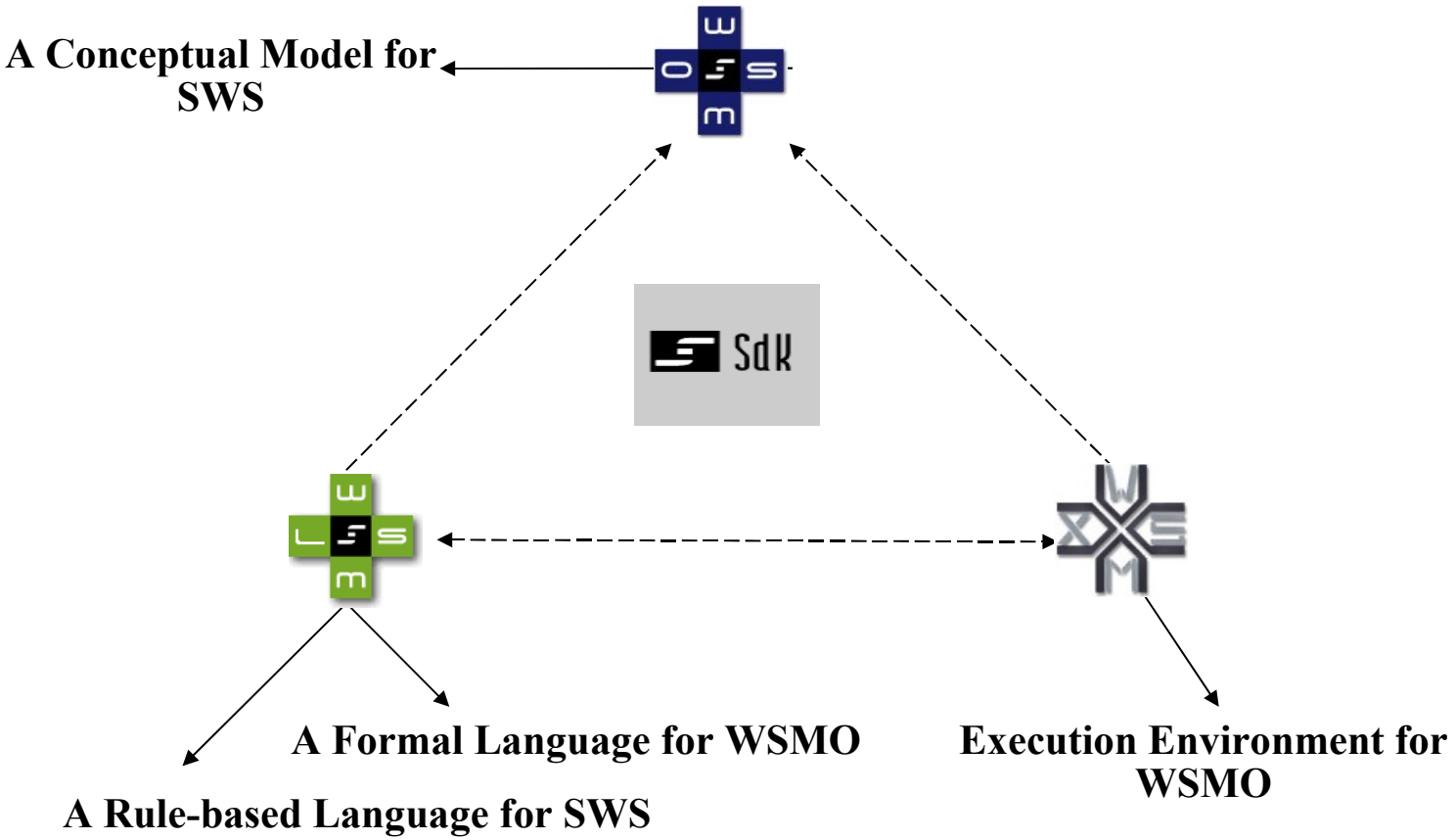
The Need for Semantics: Service Semantics

- Currently, we have a service that takes in a number, two strings of text, and returns a number. The number is termed “amount”, the strings “currency1” and “currency2”
- Linking data semantics to interface variables solves the problem of what the data actually means
- But we'd also like to know what the service *does*
- So, semantically annotate that this is a currency converter service (with e.g. STRIPS)
- Formally: With the prequisites of recognized currencies and an amount, the output will be the amount in currency one transformed into currency two

Service Semantics, cont.

- Also:
  - You need to use this encryption to contact this service
  - You need this choreography to interact with this service
  - Non-parameter requisites (STRIPS again): “For this stock trading service to function, the world needs to be in such a state that the relevant stock market is open”

WSMO Working Groups



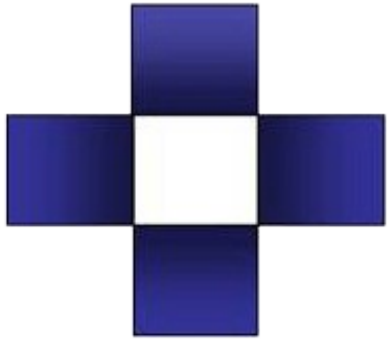
WSMO Top Level Notions

Objectives that a client wants to achieve by using Web Services

Goals

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

Ontologies



Web Services

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

Mediators

Connectors between components with mediation facilities for handling heterogeneities

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

Contents

- WSMO ontologies and the WSML ontology language
- WSMO Web Services
  - discovery
  - choreography, orchestration and **mediation**
- WSMX execution environment
- Comparison between OWL-S and WSMO

WSMO 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**

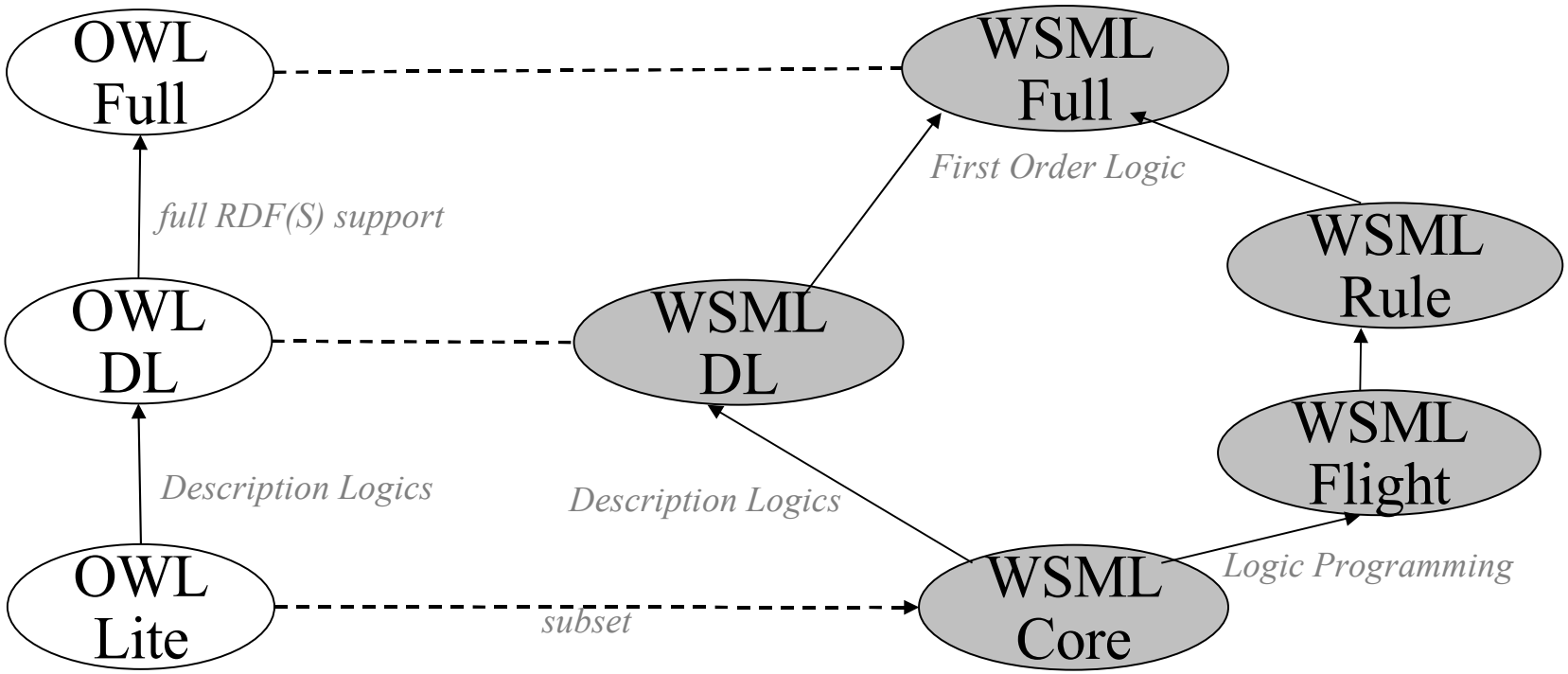
Web Service Modeling Language

- Aim – to provide a language (or a set of interoperable languages) for representing the elements of WSMO:
  - Ontologies, Web services, Goals, Mediators
- WSML provides a formal grounding for the conceptual elements of WSMO, based on:
  - *Description Logics*
  - *Logic Programming*
  - *First-Order Logic*
  - *Frame Logic*

Rationale of WSML

- Provide a Web Service Modeling Language based on the WSMO conceptual model
  - Concrete syntax
  - Semantics
- Provide a Rule Language for the Semantic Web
- Many current Semantic Web languages have
  - undesirable computational properties
  - unintuitive conceptual modeling features
  - inappropriate language layering
    - RDFS/OWL
    - OWL Lite/DL/Full
    - OWL/SWRL

OWL and WSML



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

Contents

- WSMO ontologies and the WSML ontology language
- WSMO Web Services
  - discovery
  - choreography, orchestration and **mediation**
- WSMX execution environment
- Comparison between OWL-S and WSMO

Automated WS discovery

- The task
  - Identify possible web services W **which are able to provide the requested service** S for its clients
- An **important issue …**
  - „*being able to provide a service*“ has to be **determined** based on **given descriptions only** (*WS, Goal, Ontos*)
  - Discovery can ***only be as good*** *as* these descriptions
    - **Very detailed WS descriptions**: are precise, enable highly accurate results, are more difficult to provide; *in general, requires interaction with the provider (outside the pure logics framework)*
    - **Less detailed WS descriptions**: are easy to provide for humans, but usually less precise and provide less accurate results

Ease of provision

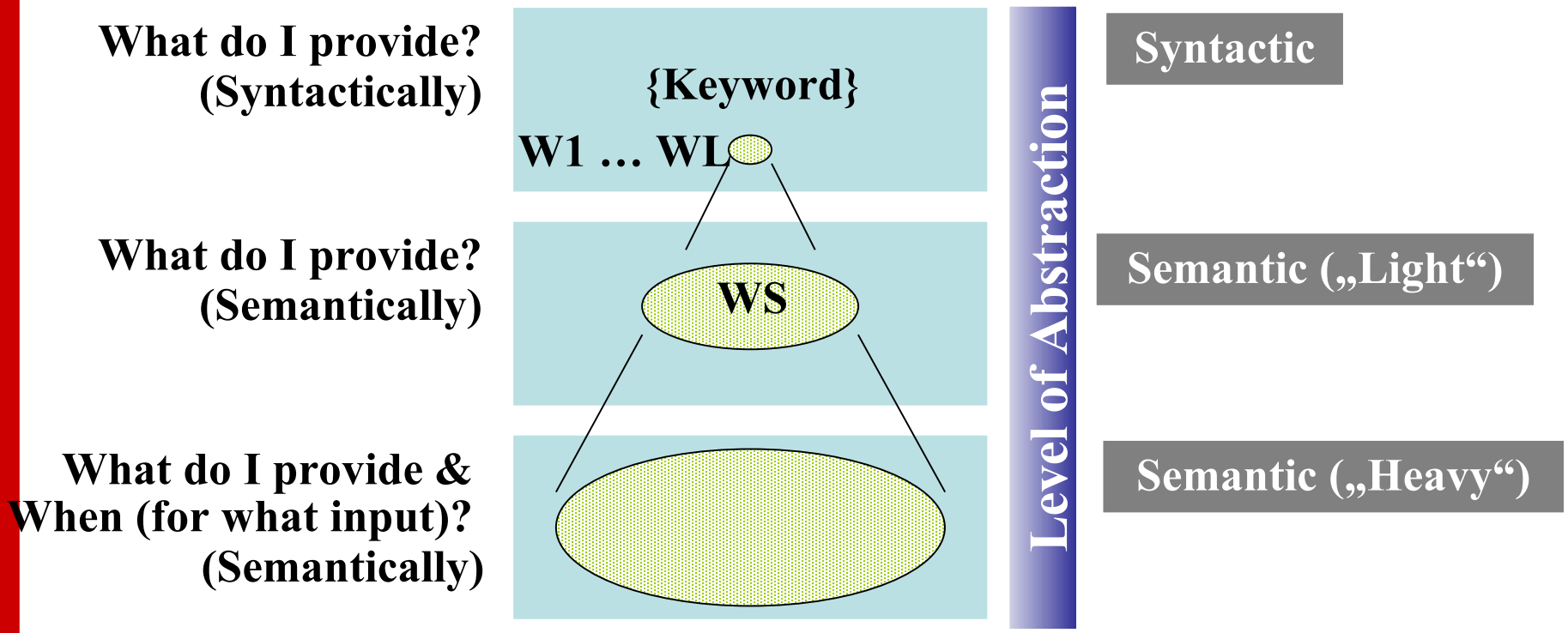
Possible Accuracy

Goals

- **Ontological De-coupling of Requester and Provider**
- **Goal-driven Architecture:**
  - 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**

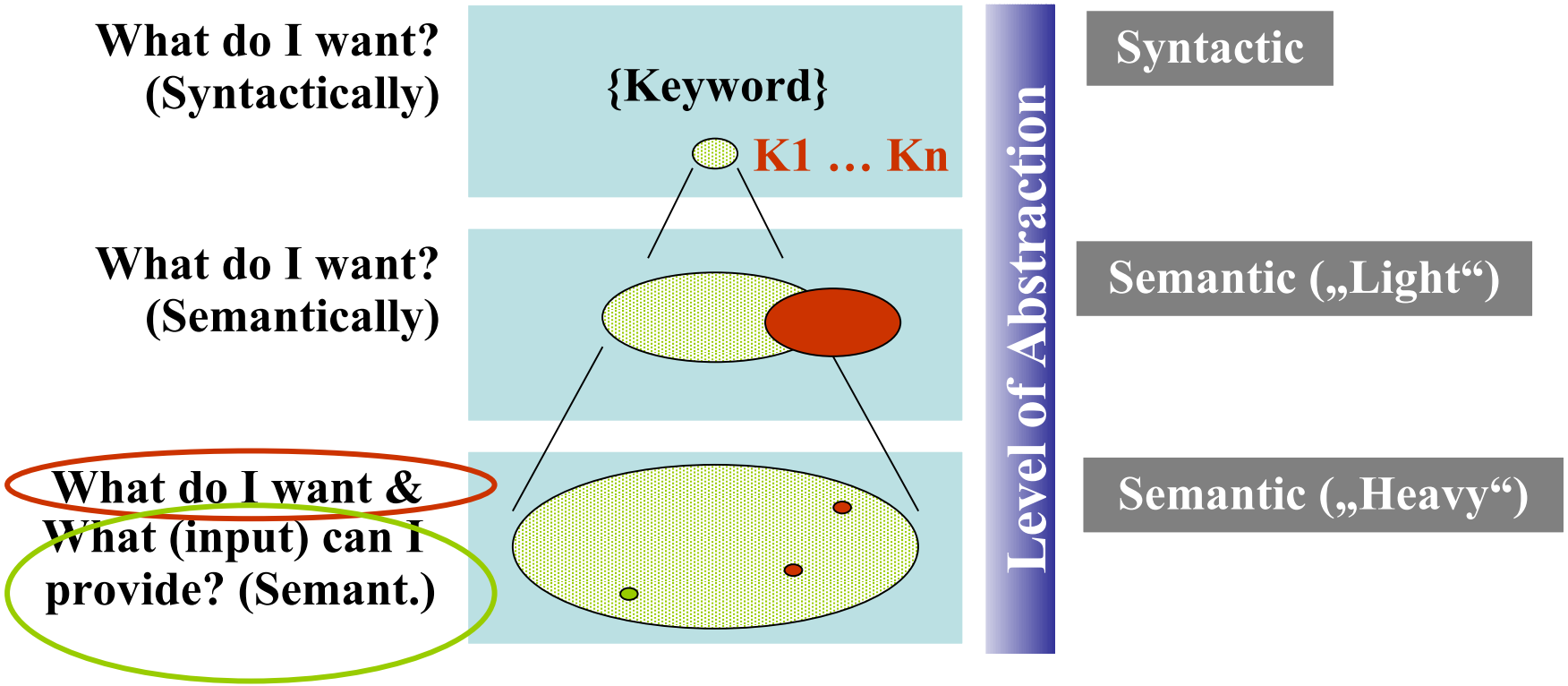
Descriptions and Discovery (II)

- **Service provider side**:
  - Capability description & levels of abstraction



Descriptions and Discovery (III)

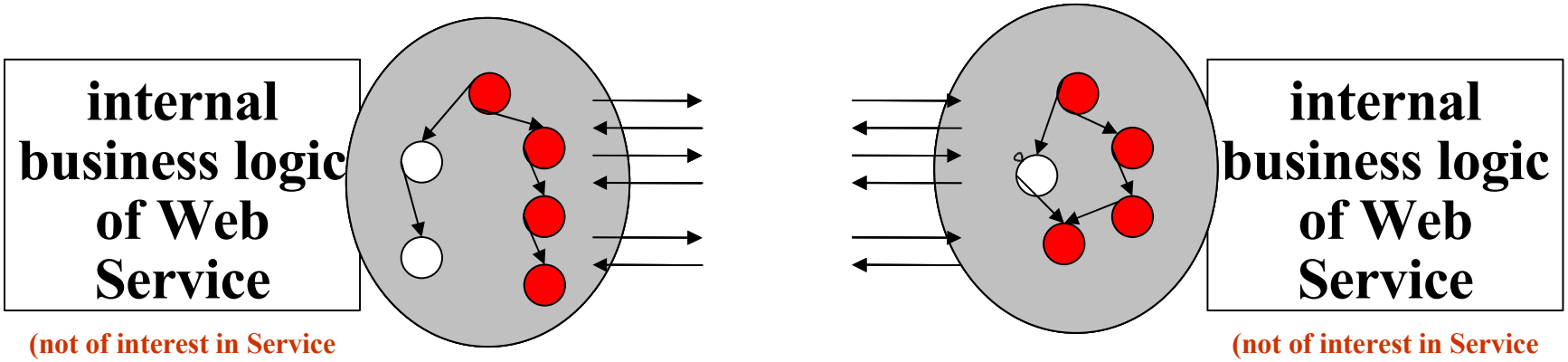
- **Service requester side**: Goal description



Contents

- WSMO ontologies and the WSML ontology language
- WSMO Web Services
  - discovery
  - choreography, orchestration and **mediation**
- WSMX execution environment
- Comparison between OWL-S and WSMO

Choreography Discovery



(not of interest in Service Interface Description)

(not of interest in Service Interface Description)

- 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

Orchestration Validation Example

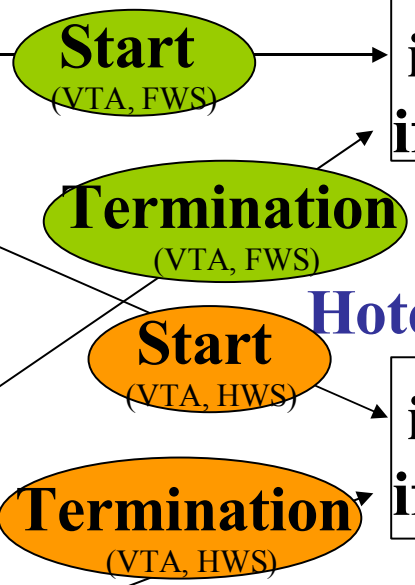
VTA Web Service Orchestration

if Ø **then** (FWS, flightRequest)

if flightOffer
then (HWS, hotelRequest)

if selection
then (FWS, flightBookingOrder)

if selection, flightBookingConf
then (HWS, hotelBookingOrder)



Flight WS Behavior Interfac

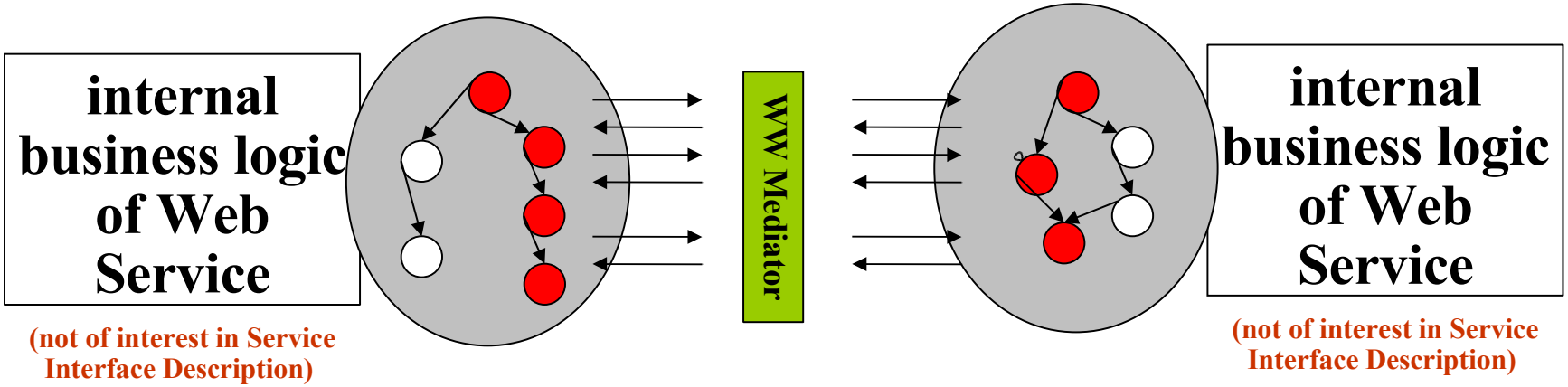
if request **then** offer
if order **then** confirmation

Hotel WS Behavior Interfac

if request **then** offer
if order **then** confirmation

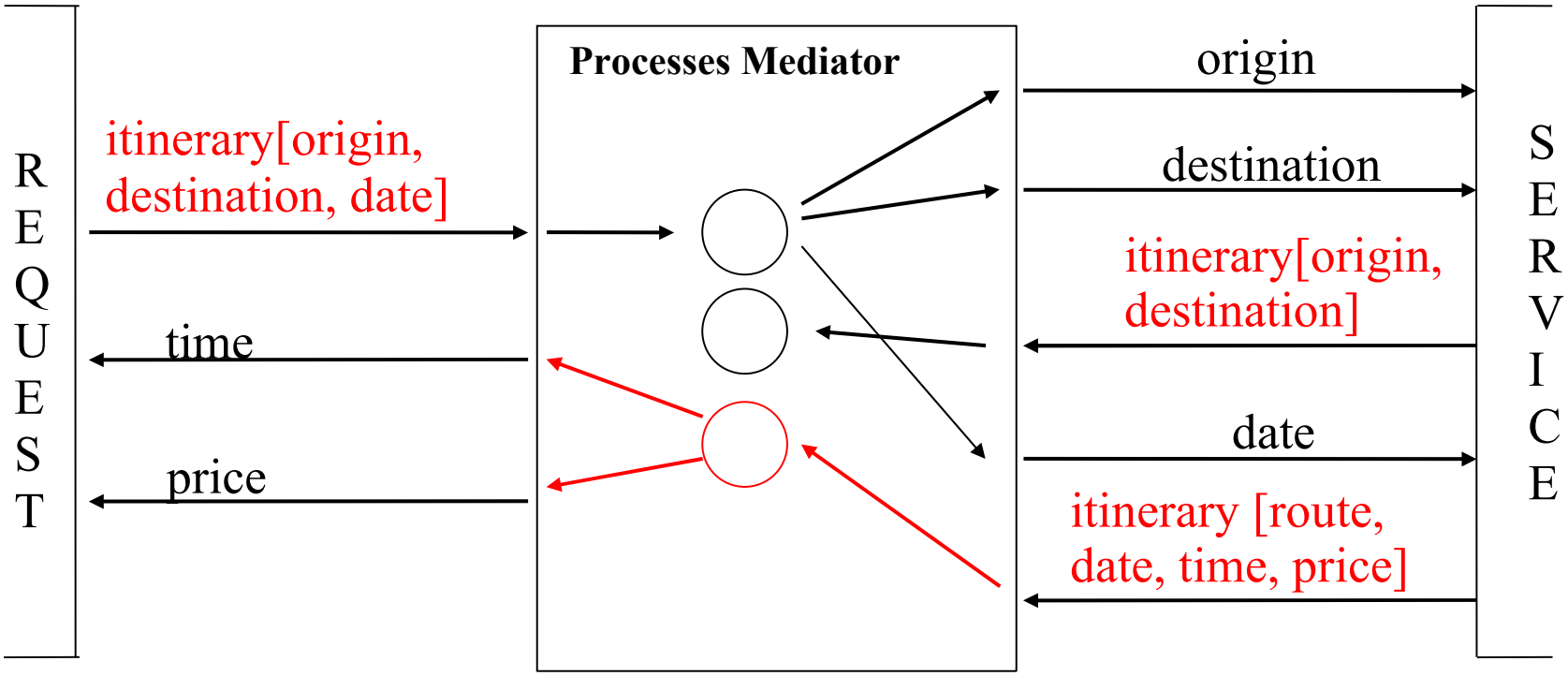
Orchestration is valid if valid choreography exists for interactions between Orchestrator and each aggregated Web Service, done by choreography discovery

Protocol & Process Level Mediation

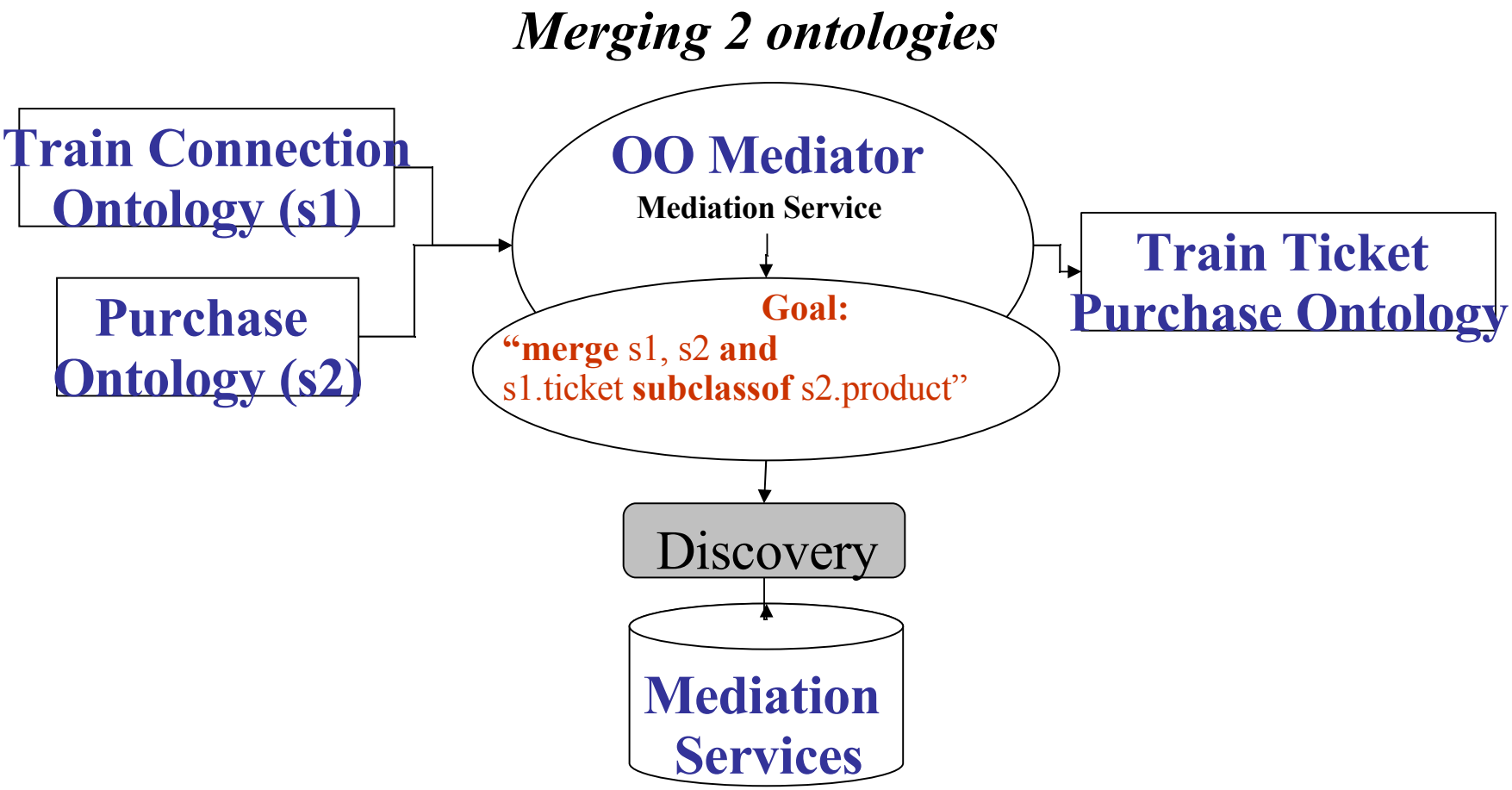


- if a choreography does not exist, then find an appropriate WW Mediator that
  - resolves possible mismatches to establish Information Compatibility (OO Mediator usage)
  - resolves process / protocol level mismatches in to establish Communication Compatibility

Process Mediation Example



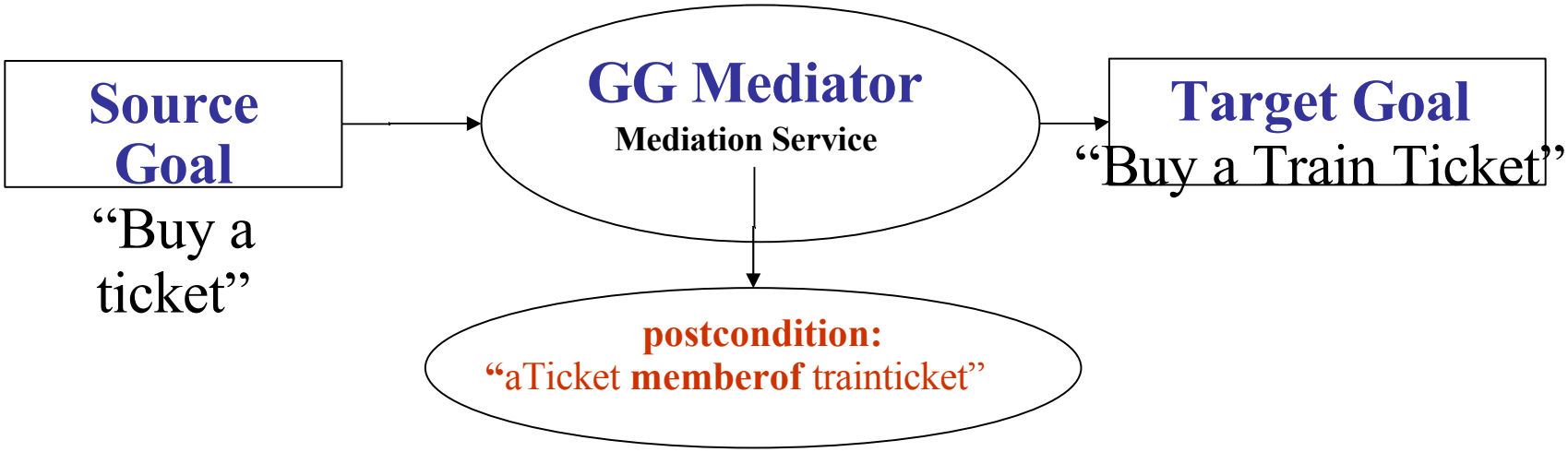
OO Mediator - Example



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

Contents

- WSMO ontologies and the WSML ontology language
- WSMO Web Services
  - discovery
  - choreography, orchestration and **mediation**
- WSMX execution environment
- Comparison between OWL-S and WSMO

WSMX Design Principles

Strong Decoupling & Strong Mediation

autonomous components with mediators for interoperability

Interface vs. Implementation

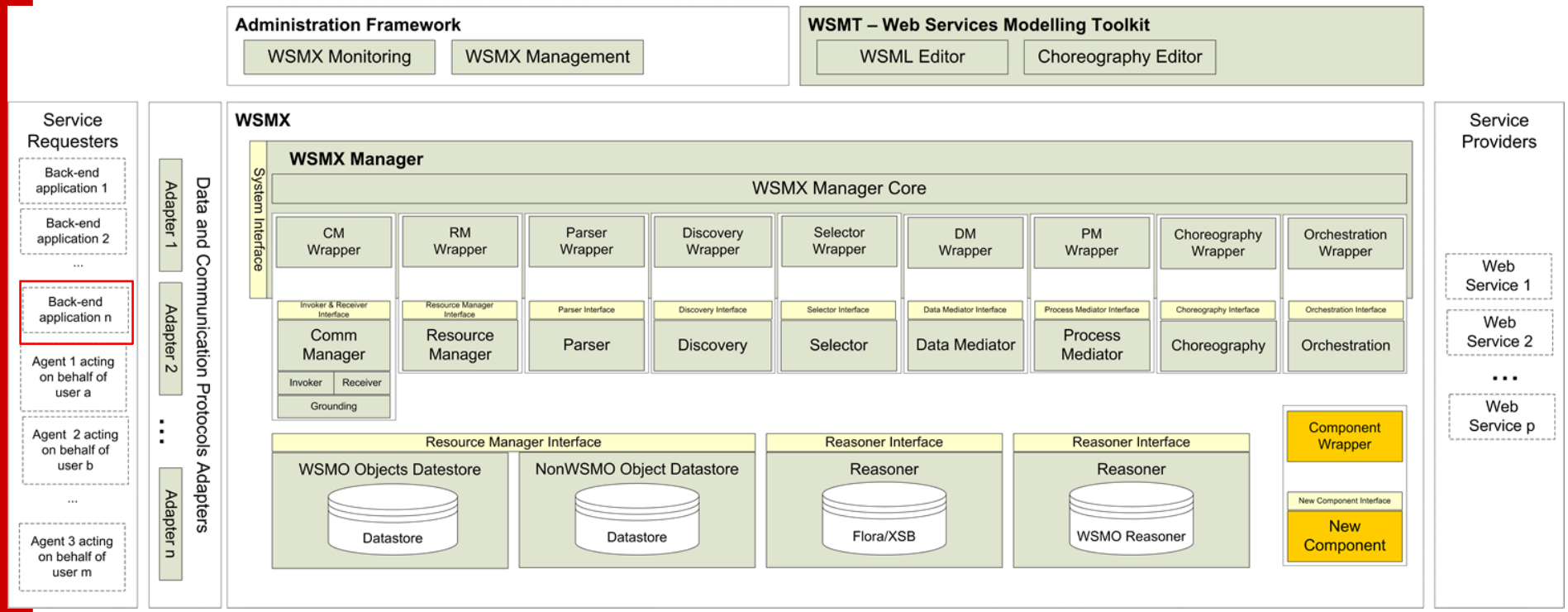
distinguish interface (= description) from implementation (=program)

Peer to Peer

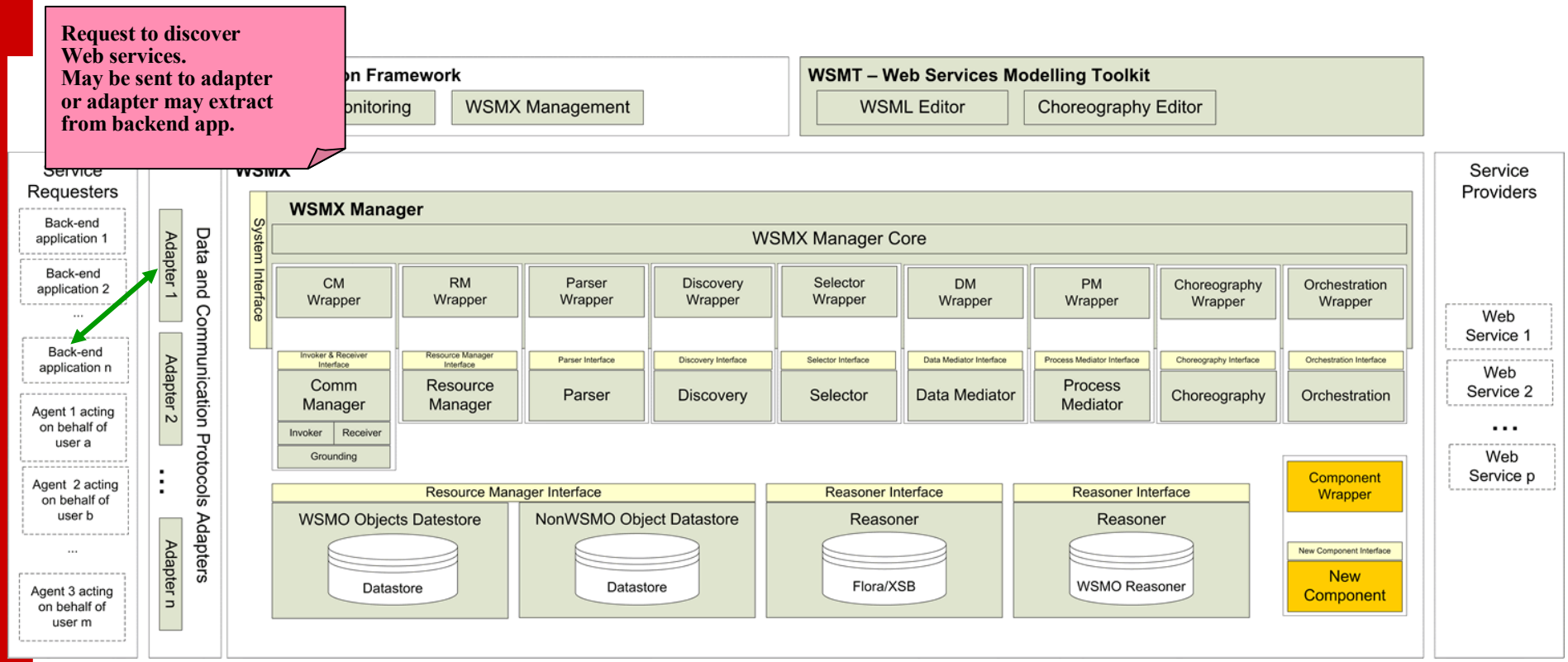
interaction between equal partners (in terms of control)

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

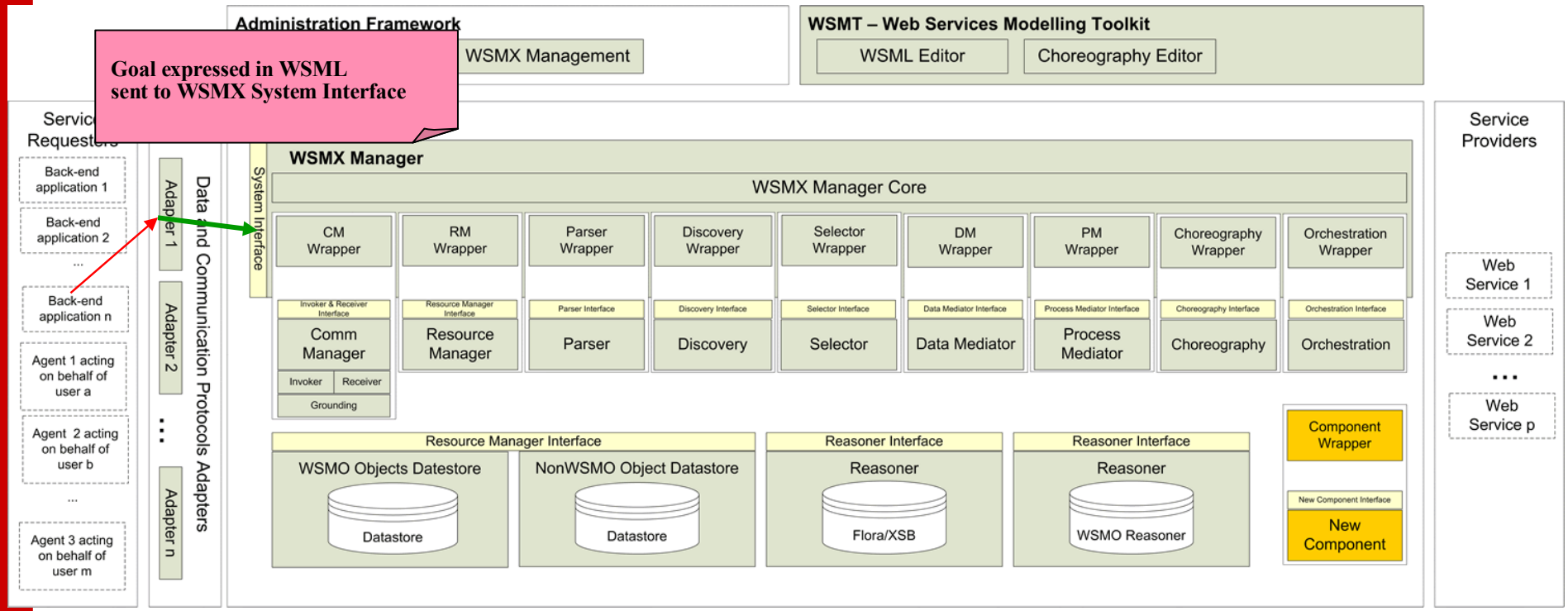
System Architecture



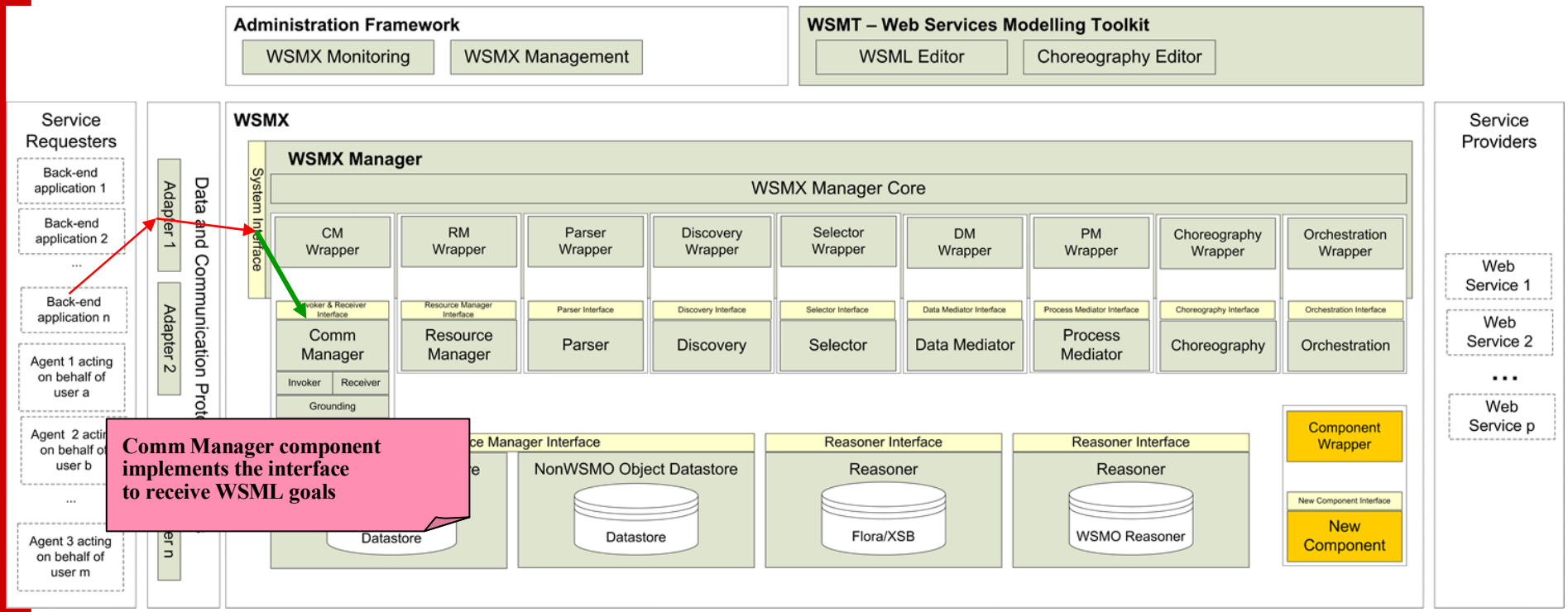
System Architecture



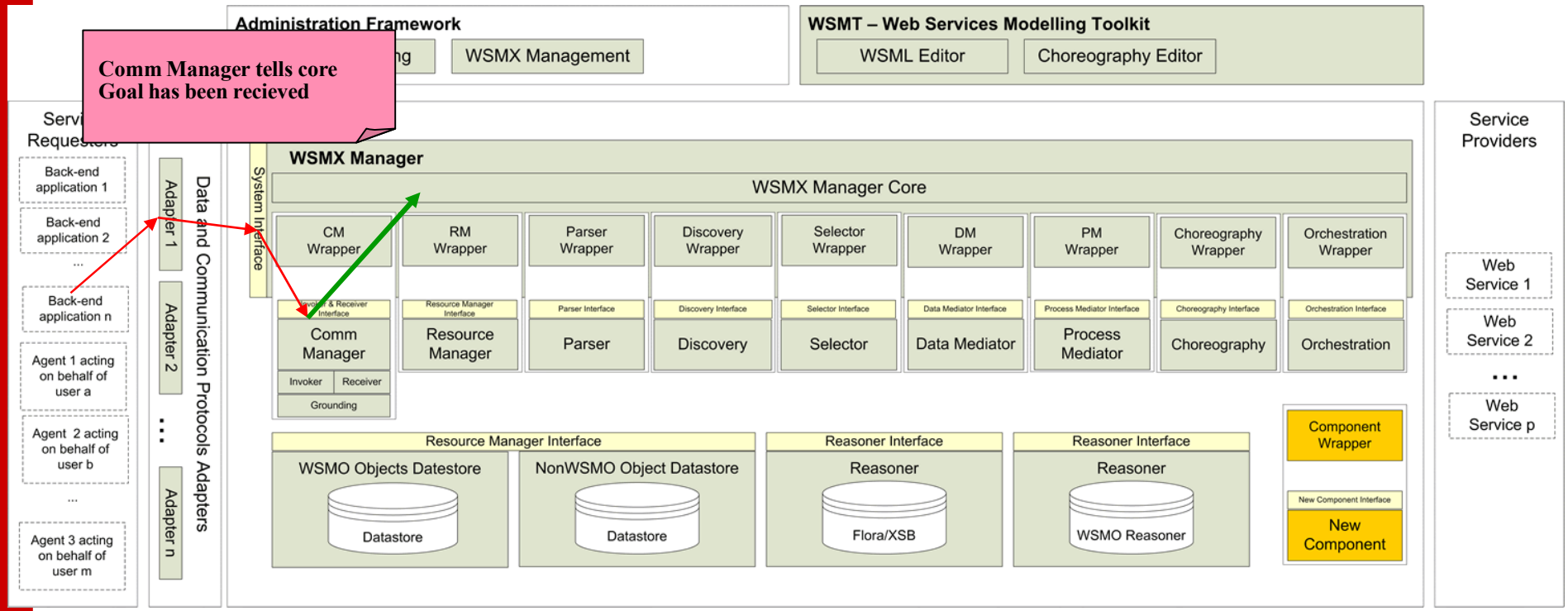
System Architecture



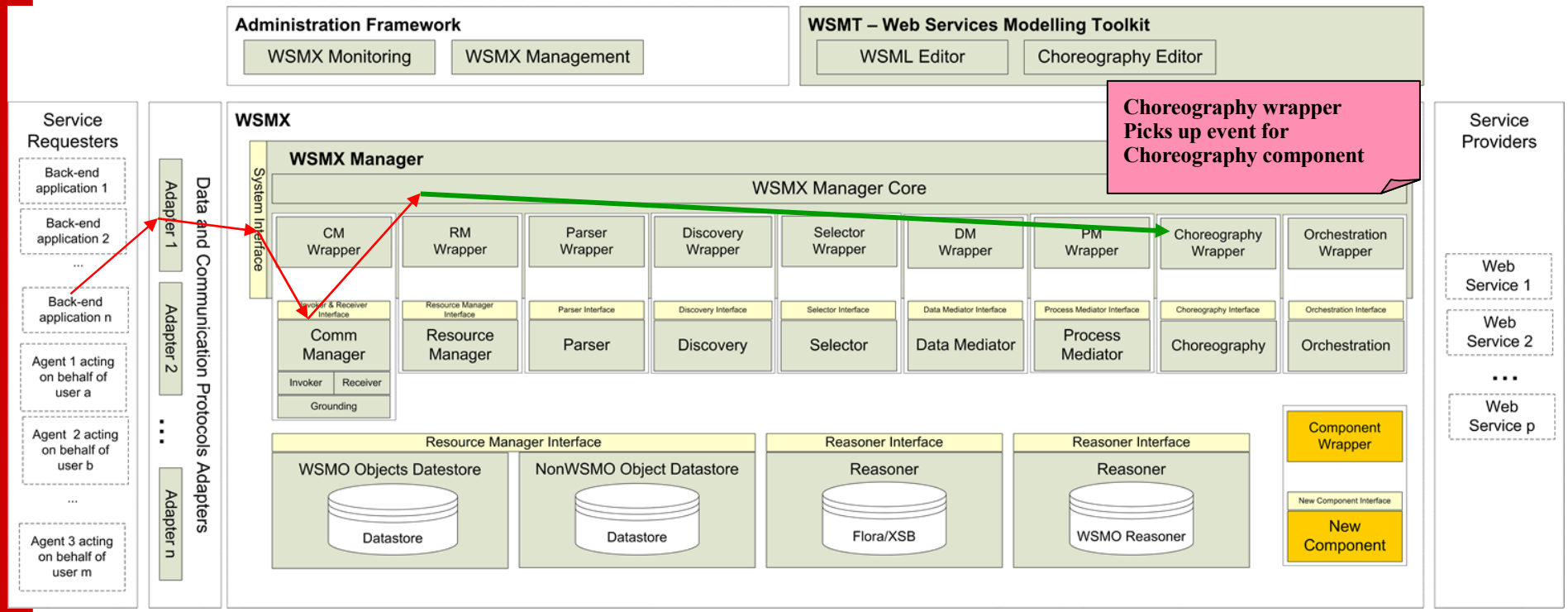
System Architecture



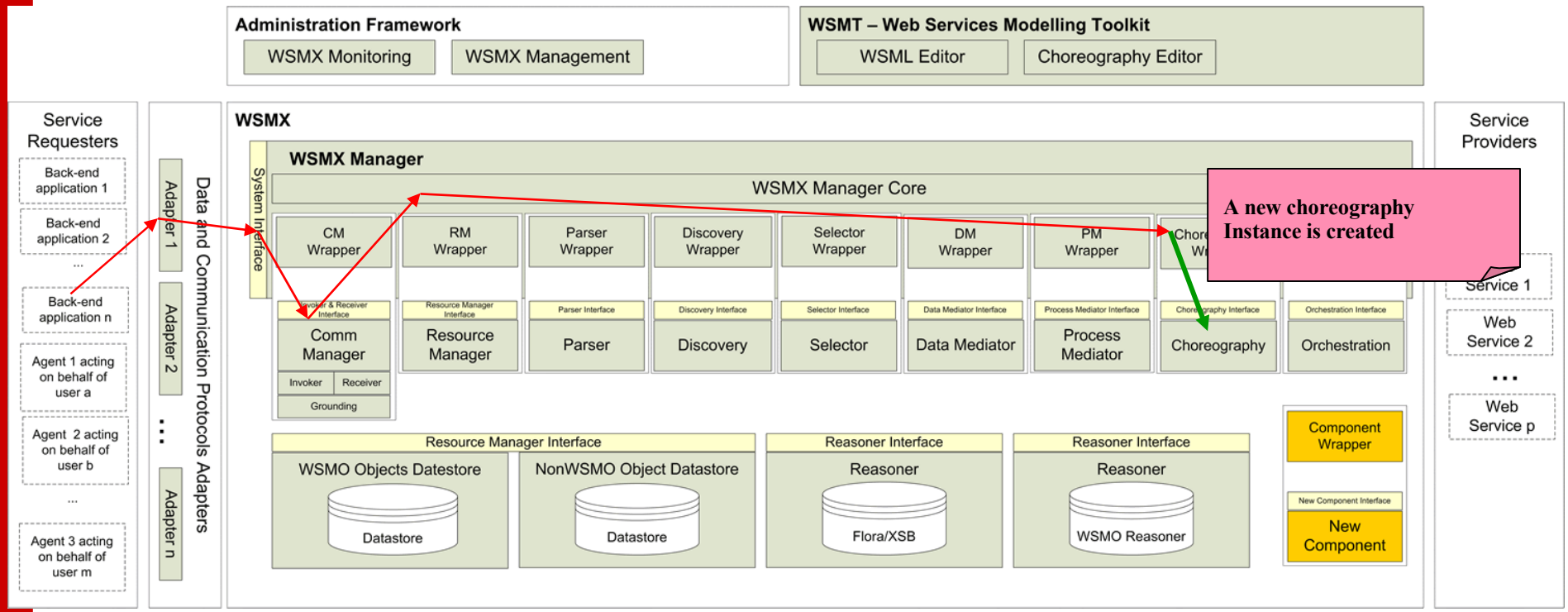
System Architecture



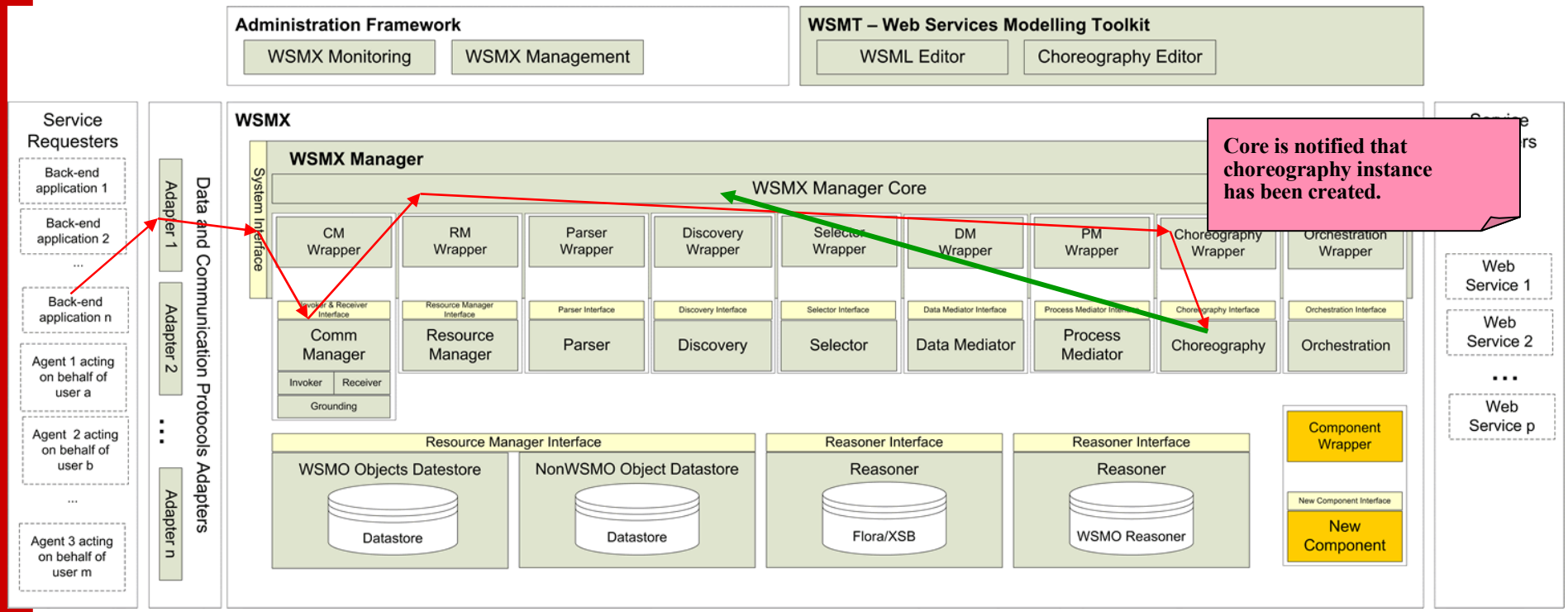
System Architecture



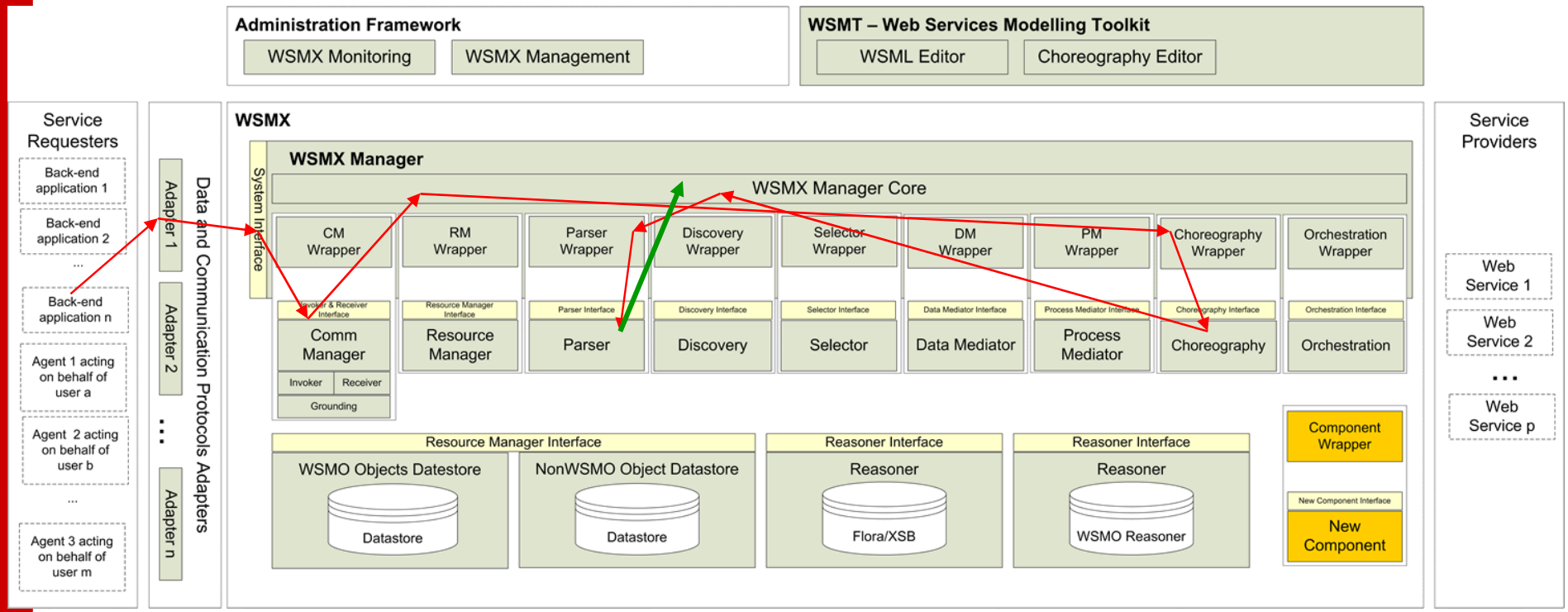
System Architecture



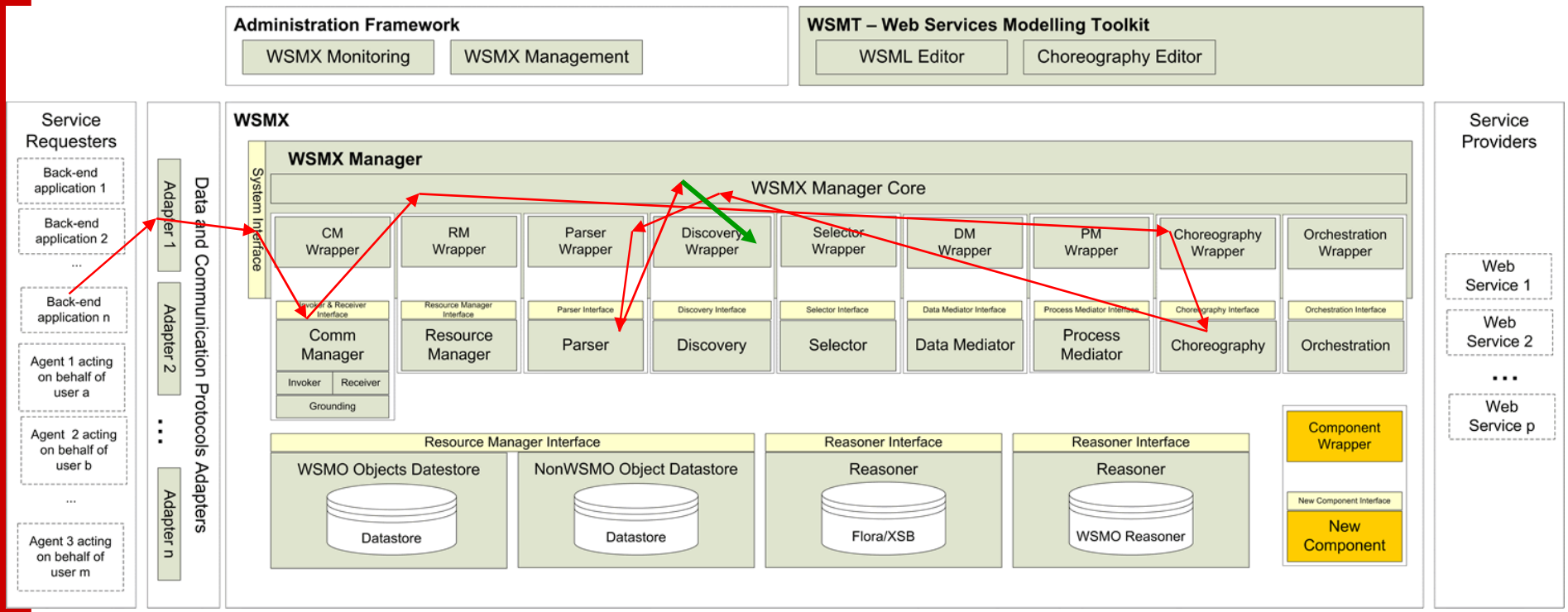
System Architecture



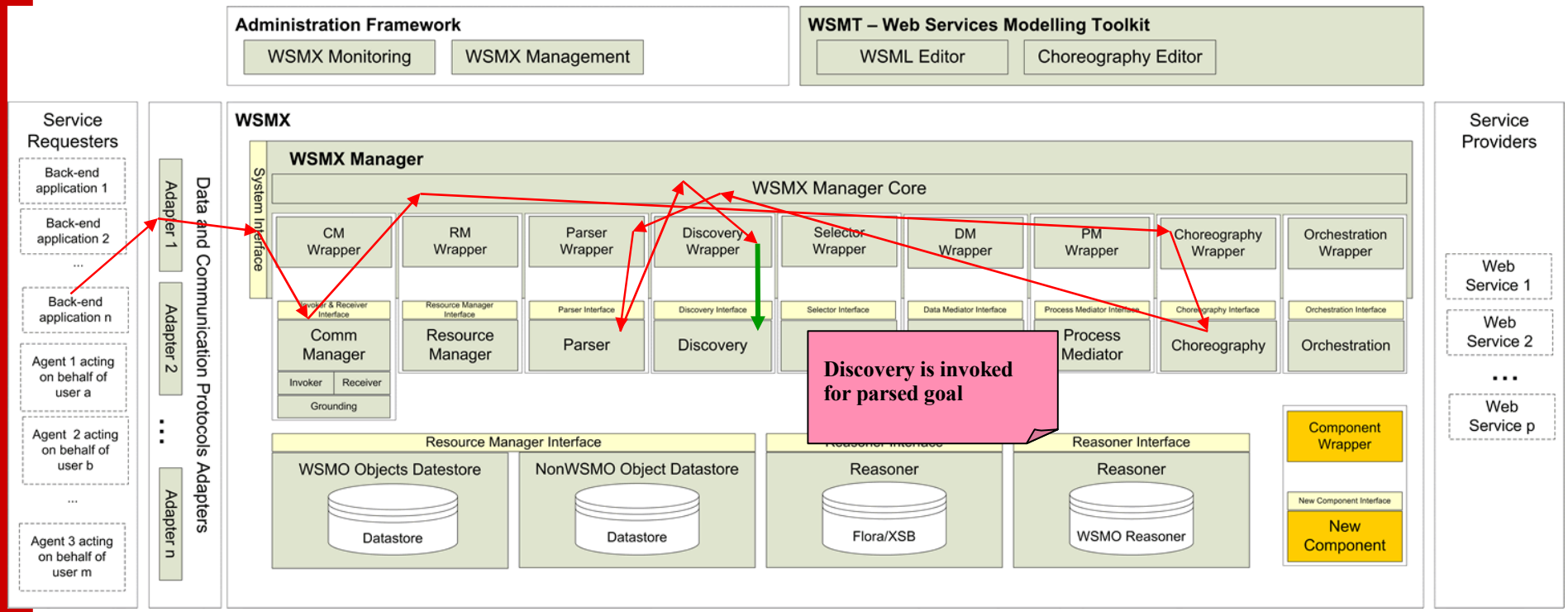
System Architecture



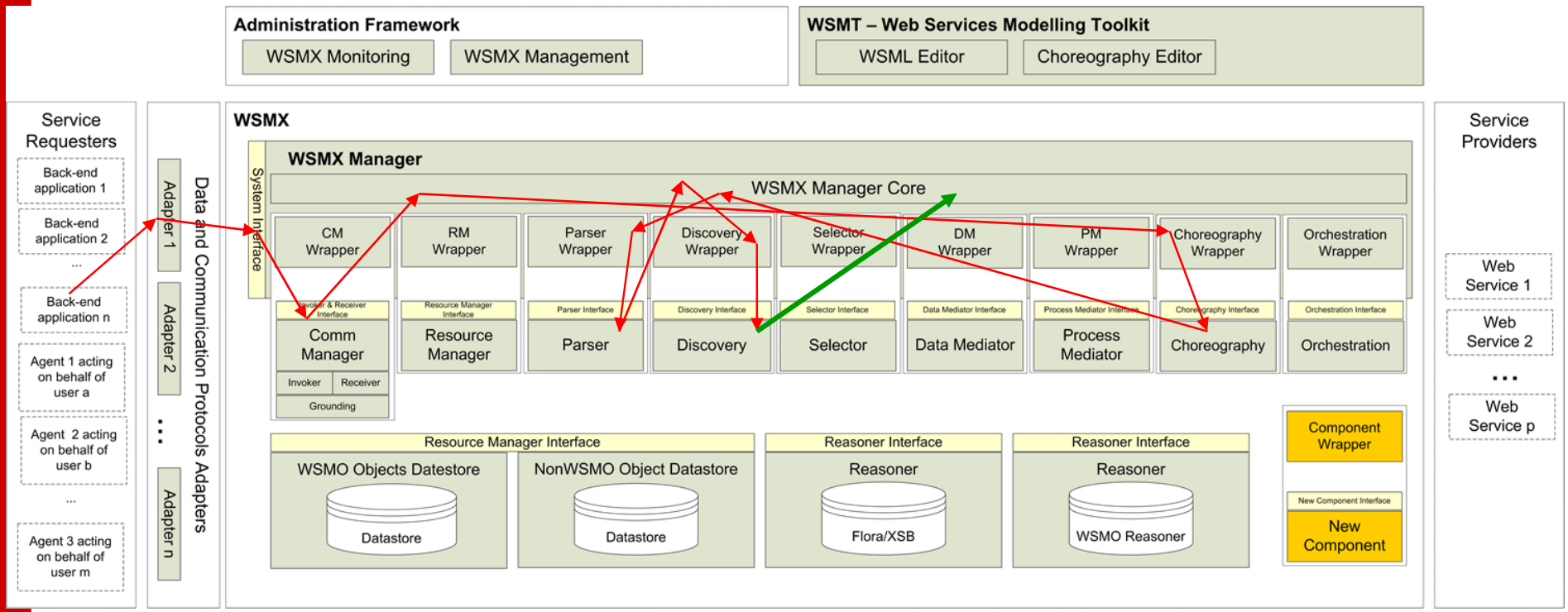
System Architecture



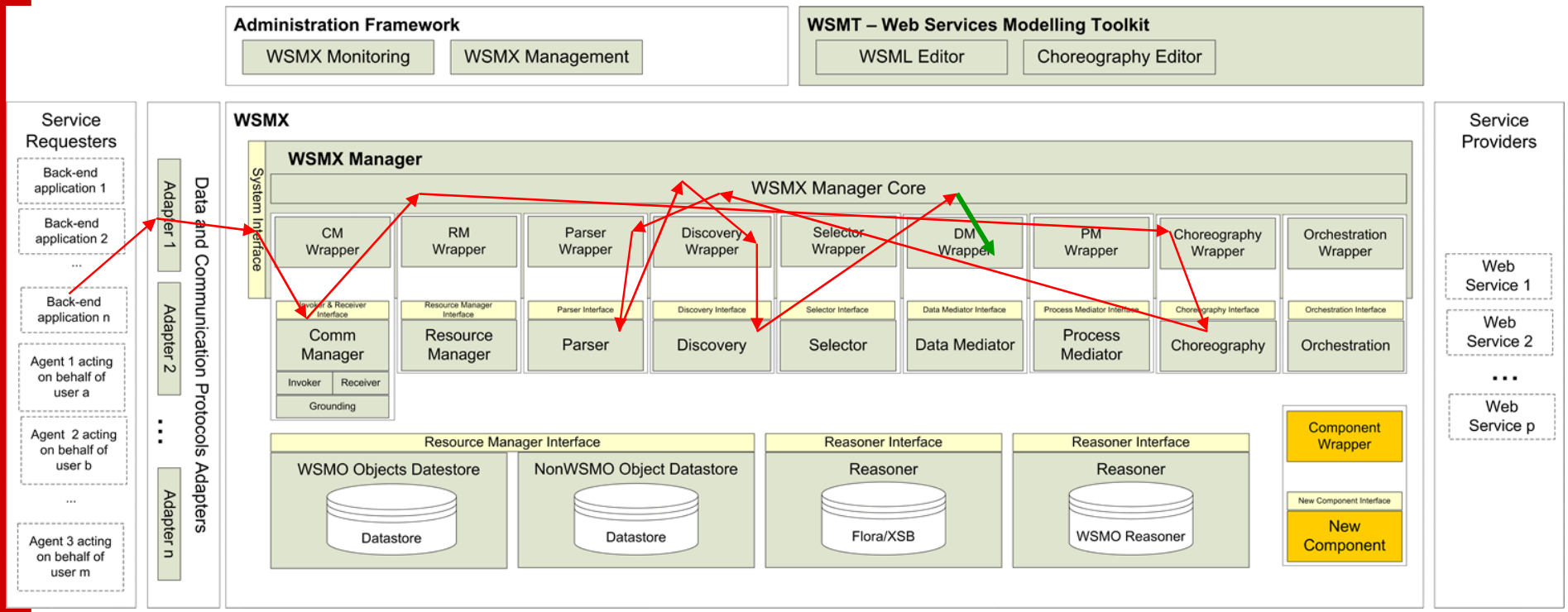
System Architecture



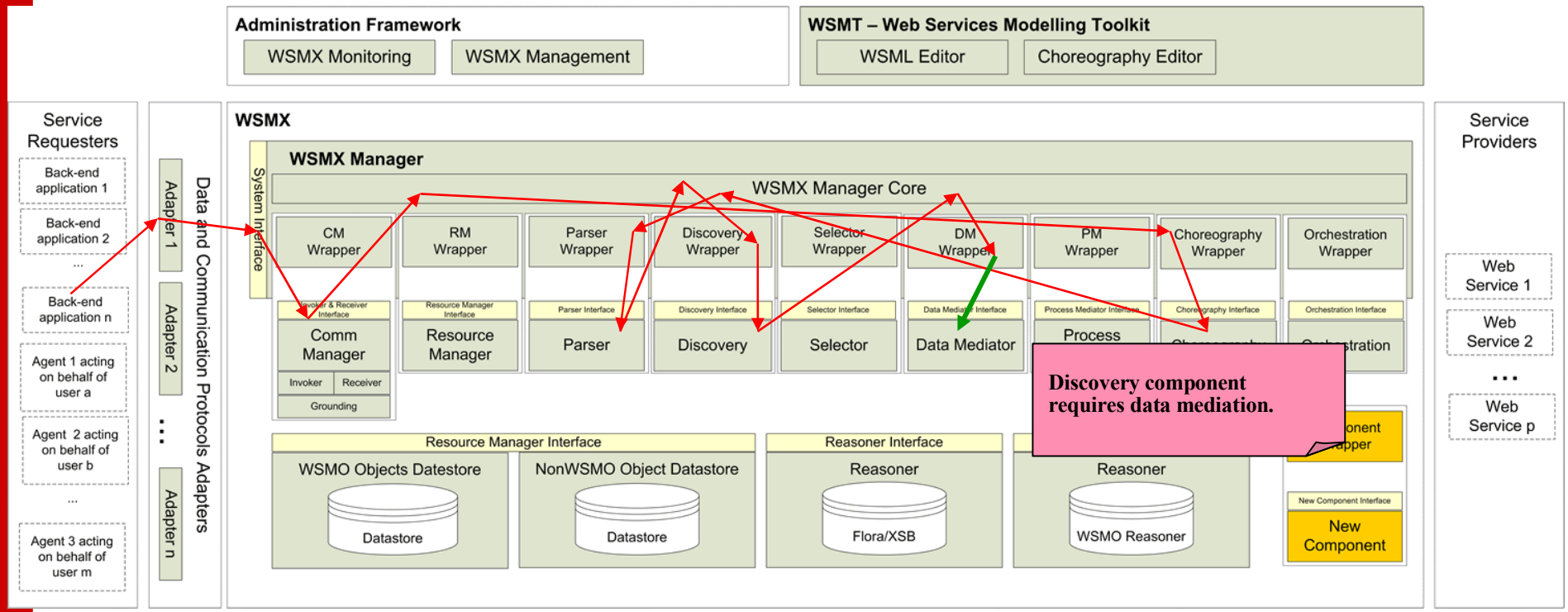
System Architecture



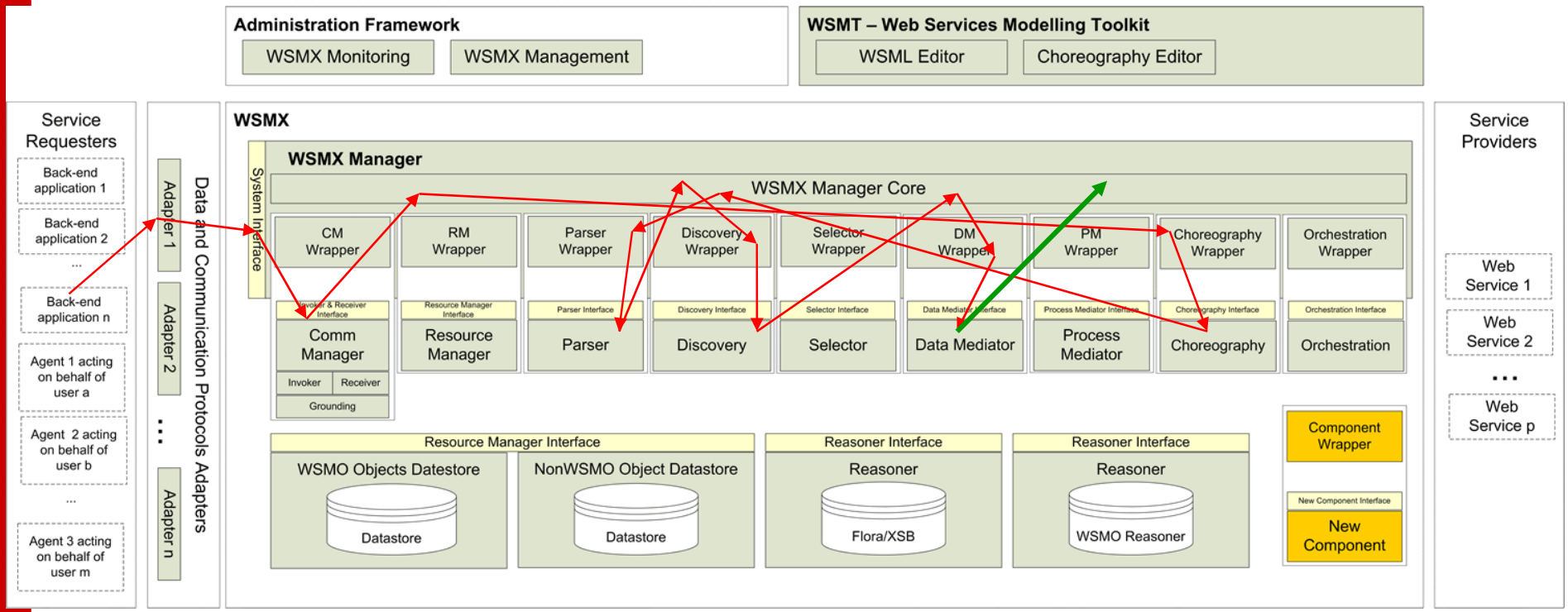
System Architecture



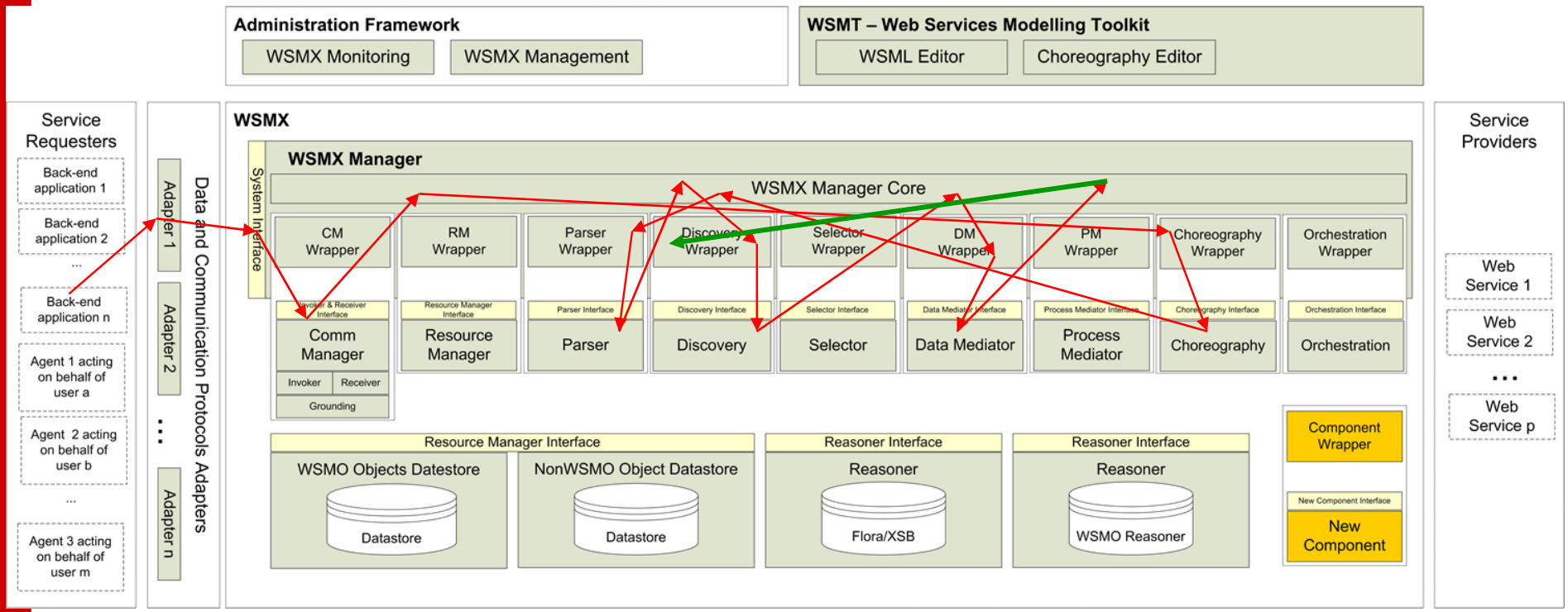
System Architecture



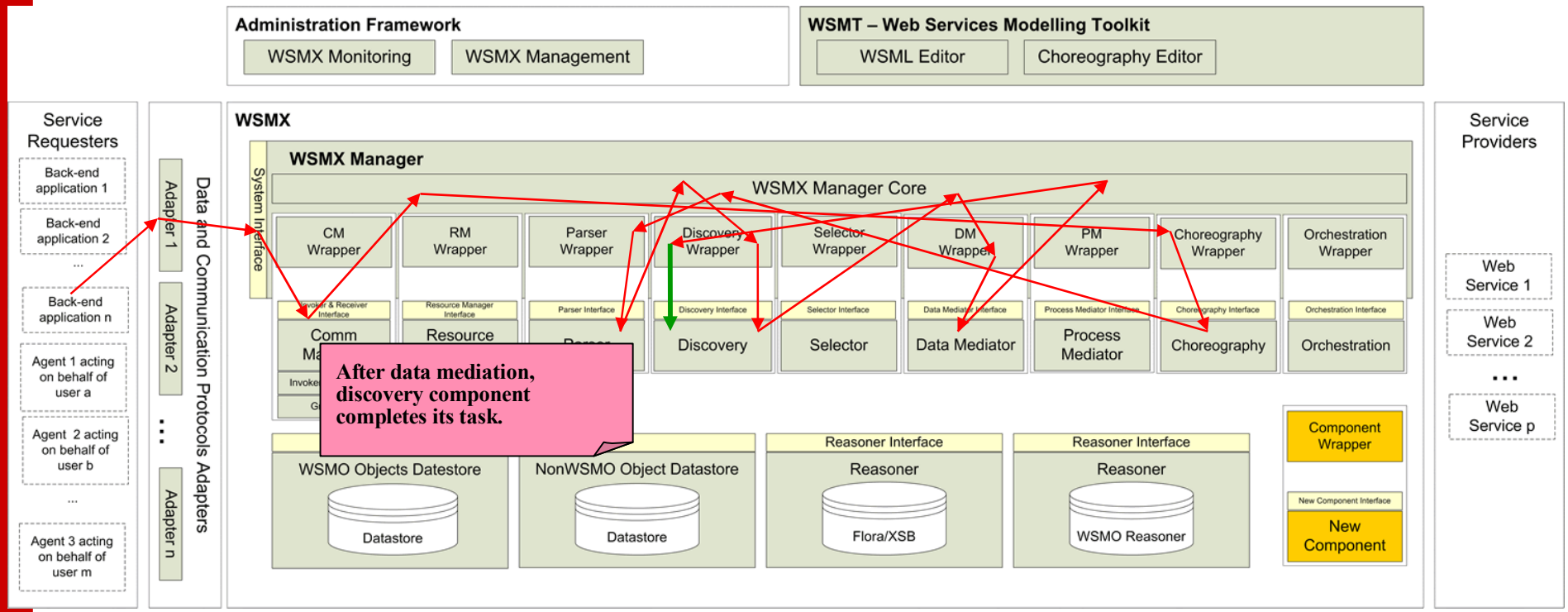
System Architecture



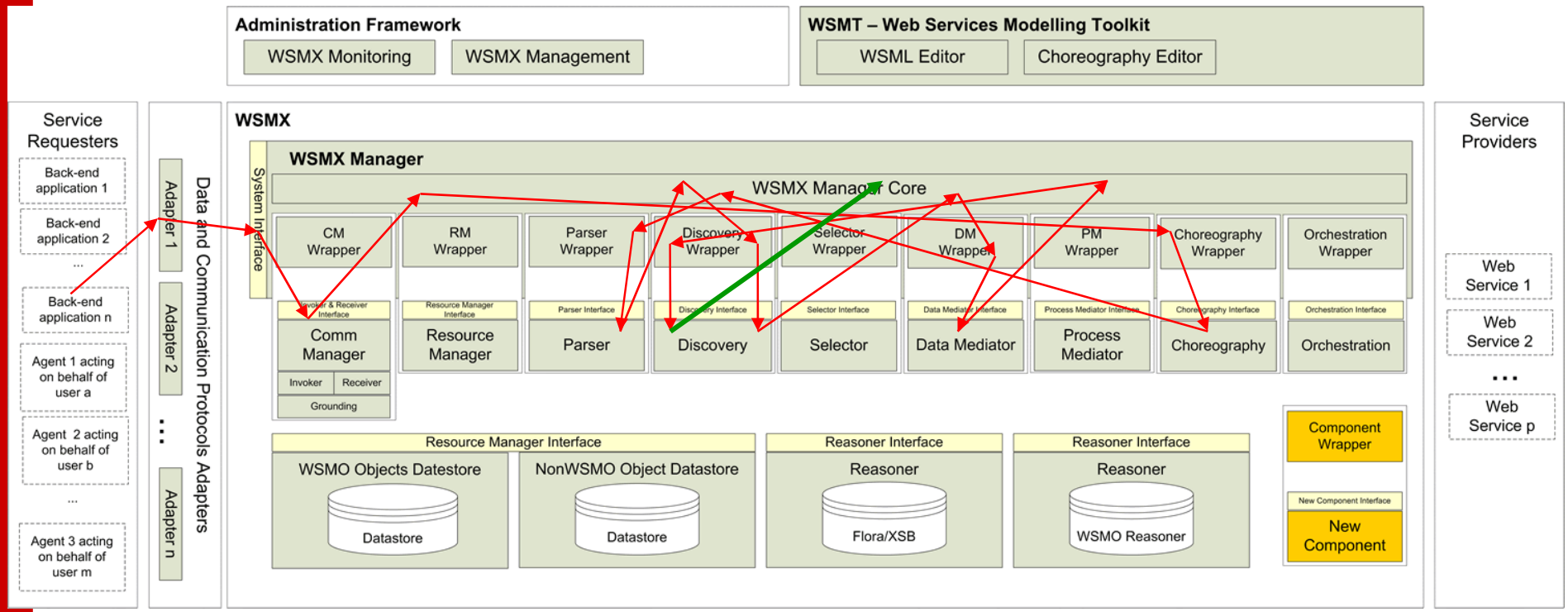
System Architecture



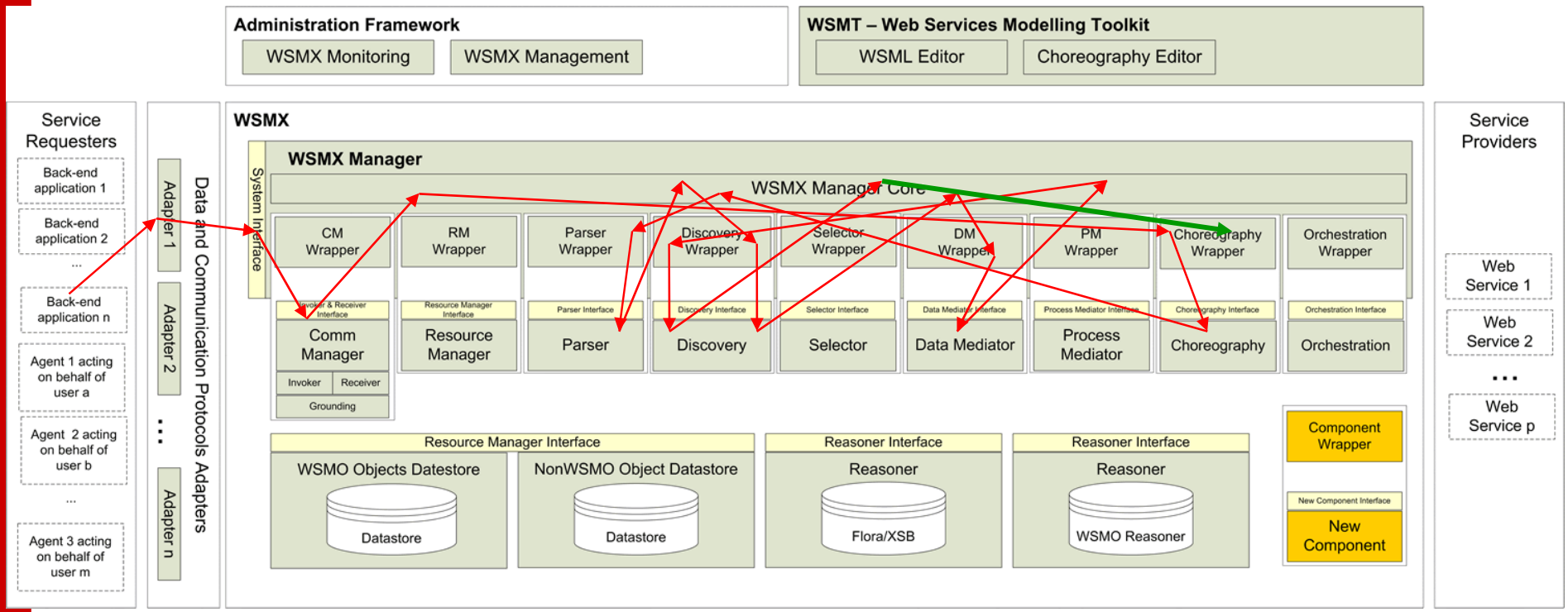
System Architecture



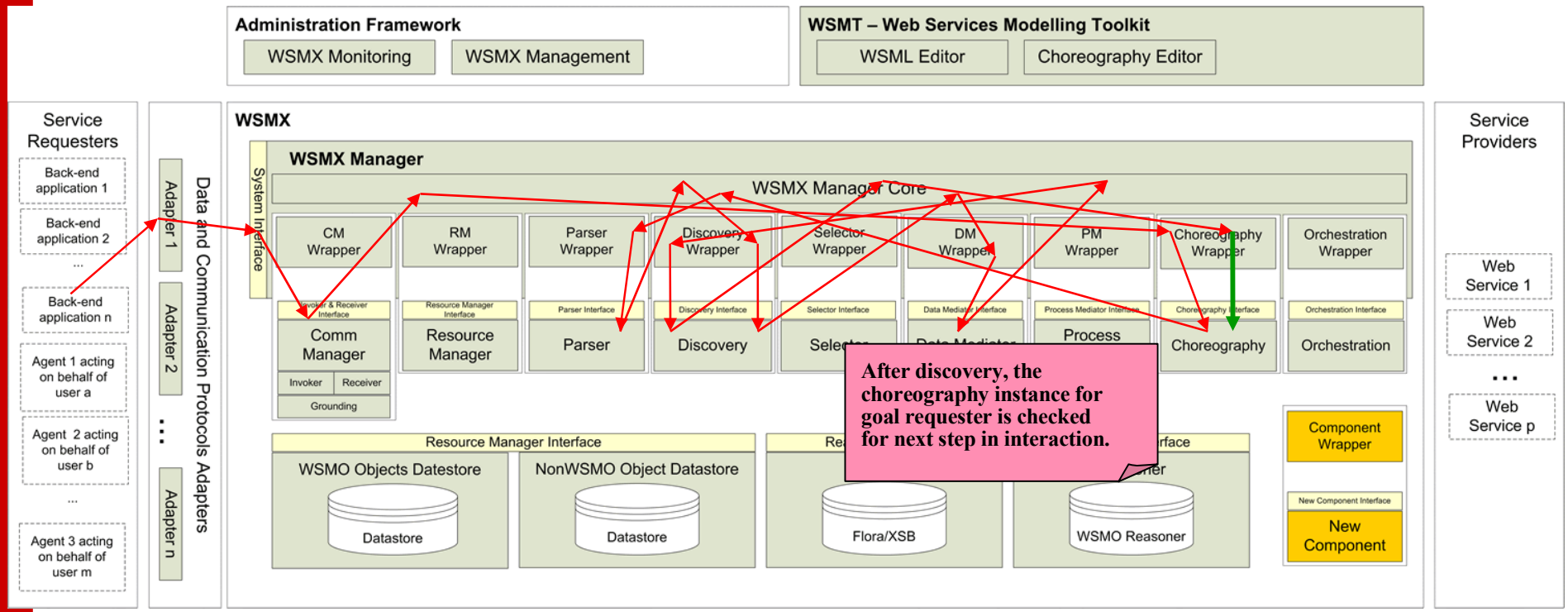
System Architecture



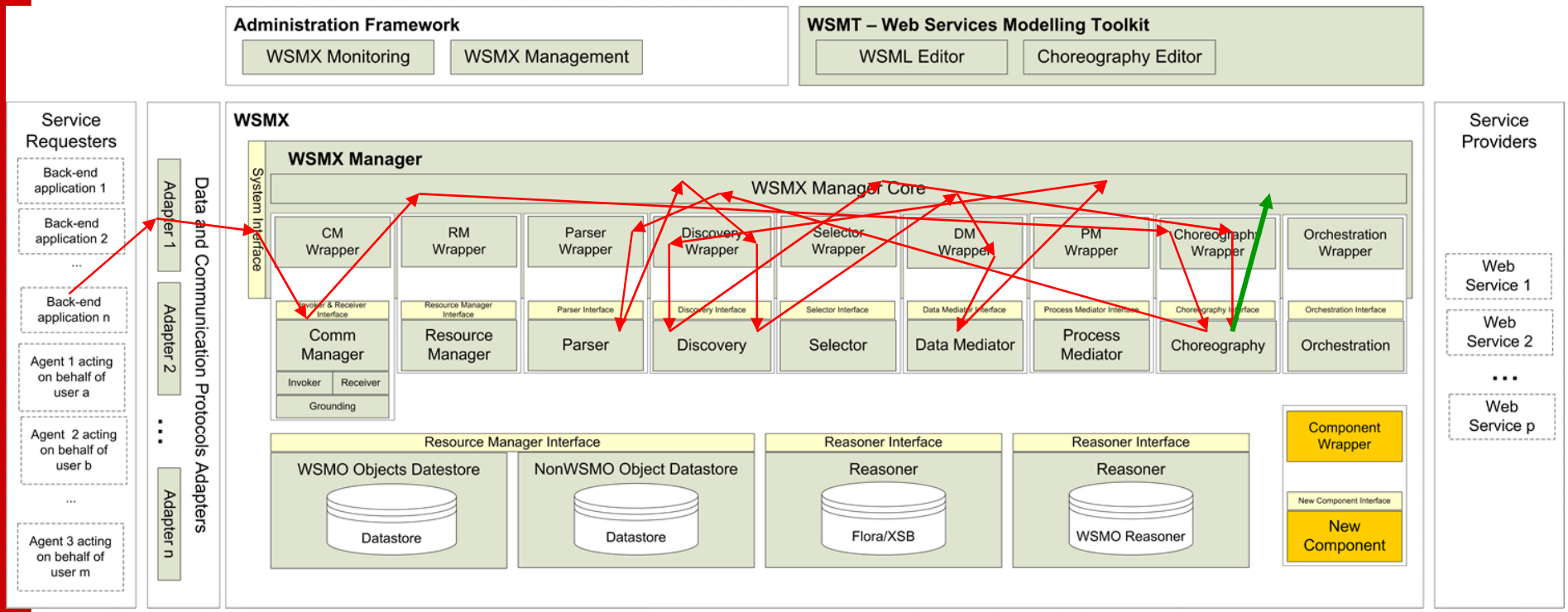
System Architecture



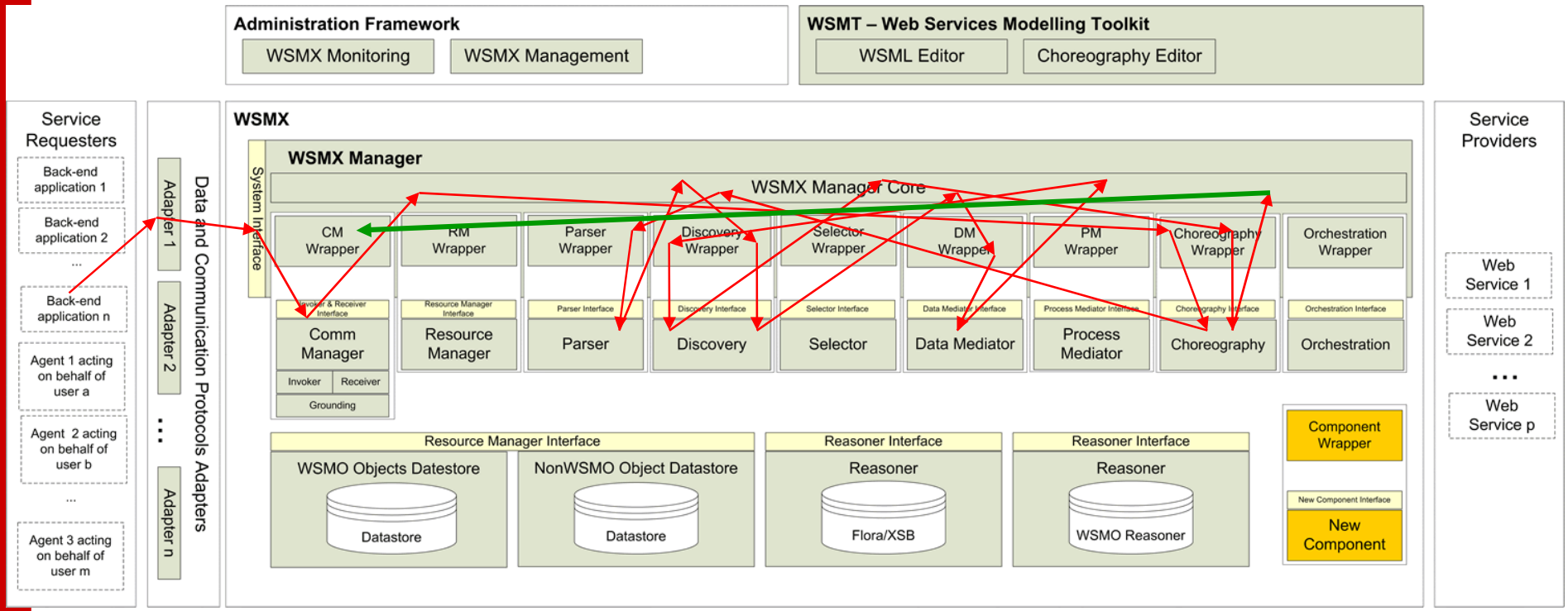
System Architecture



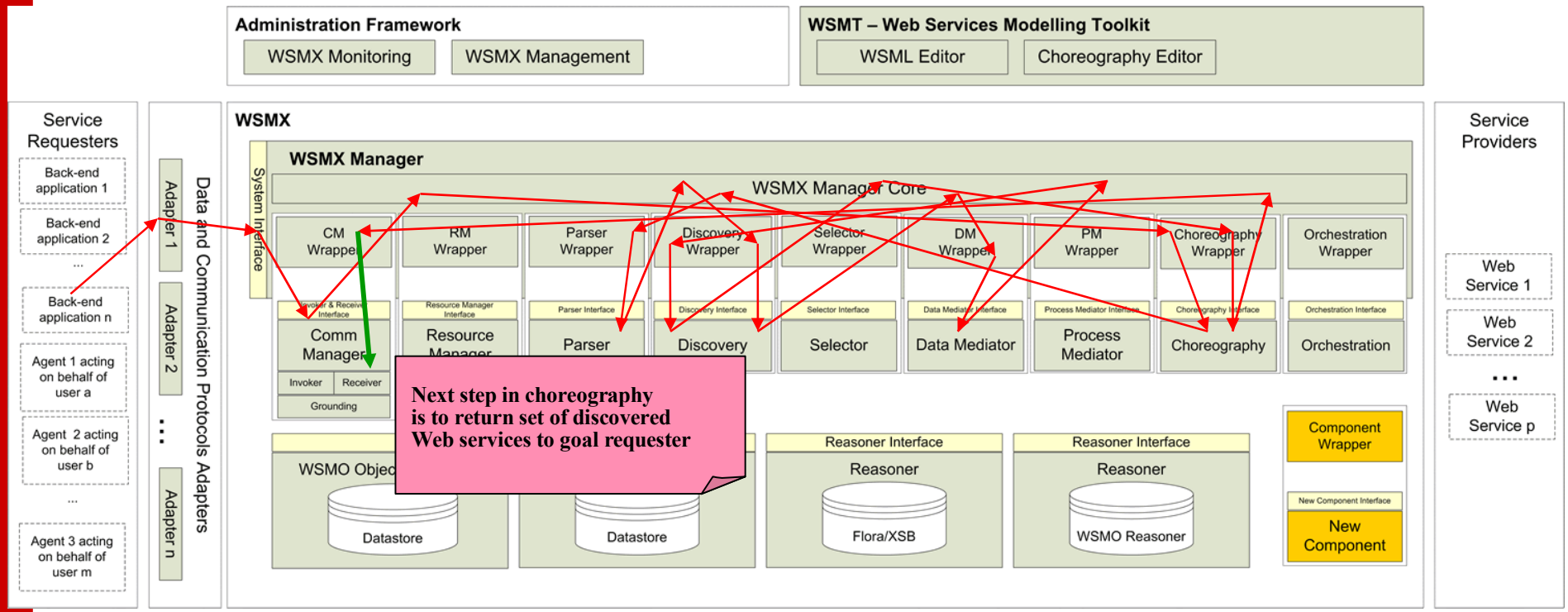
System Architecture



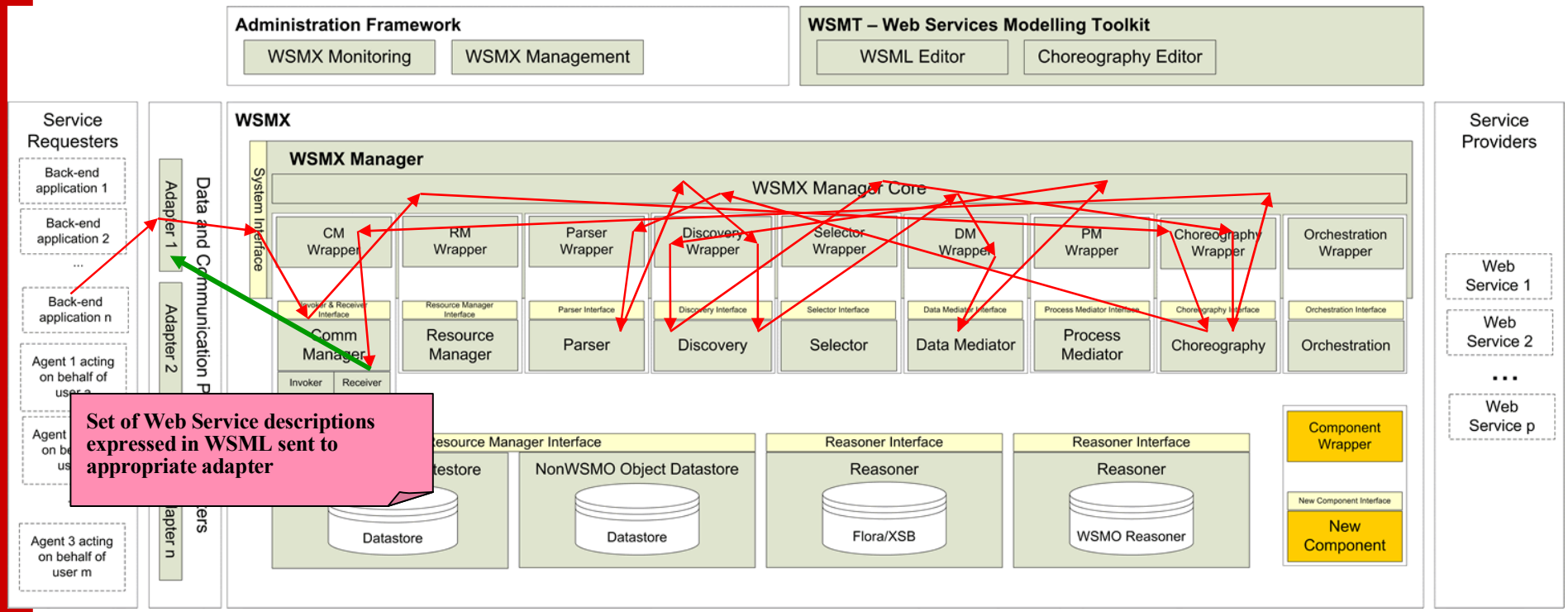
System Architecture



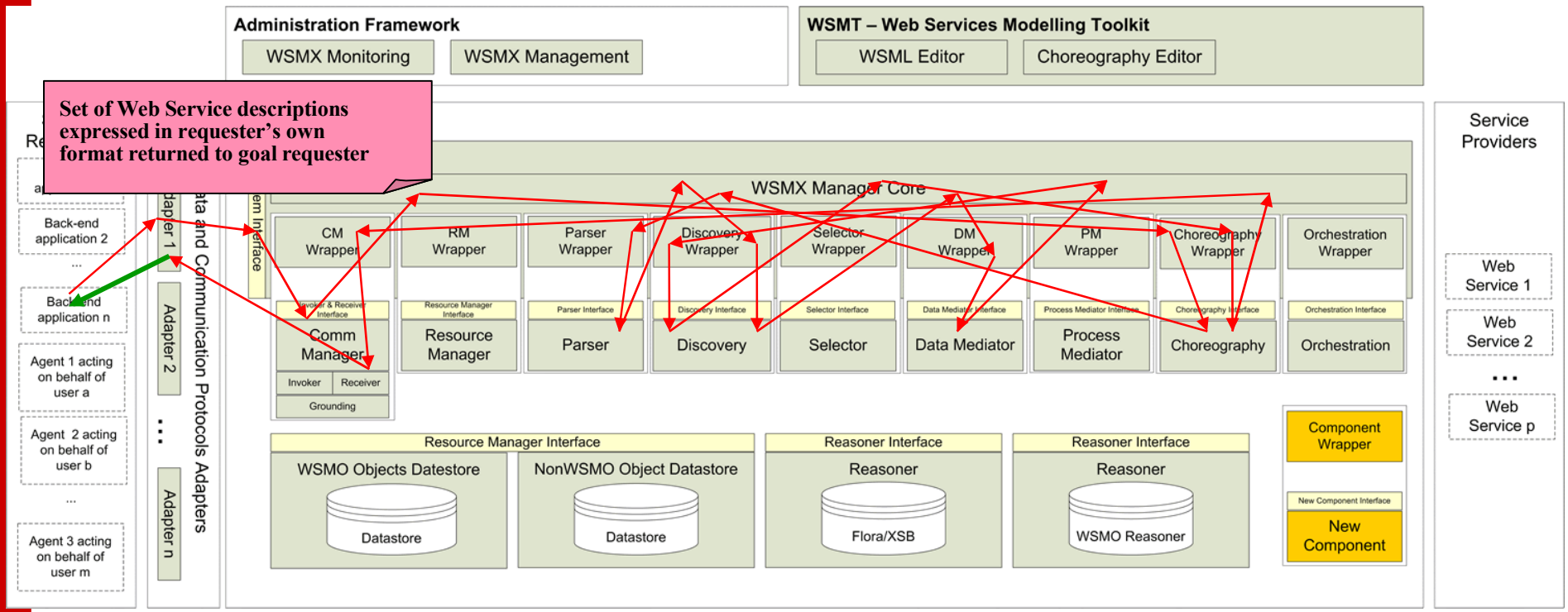
System Architecture



System Architecture



System Architecture



Contents

- WSMO ontologies and the WSML ontology language
- WSMO Web Services
  - discovery
  - choreography, orchestration and **mediation**
- WSMX execution environment
- Comparison between OWL-S and WSMO

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