Automated Collaboration on the Semantic Web

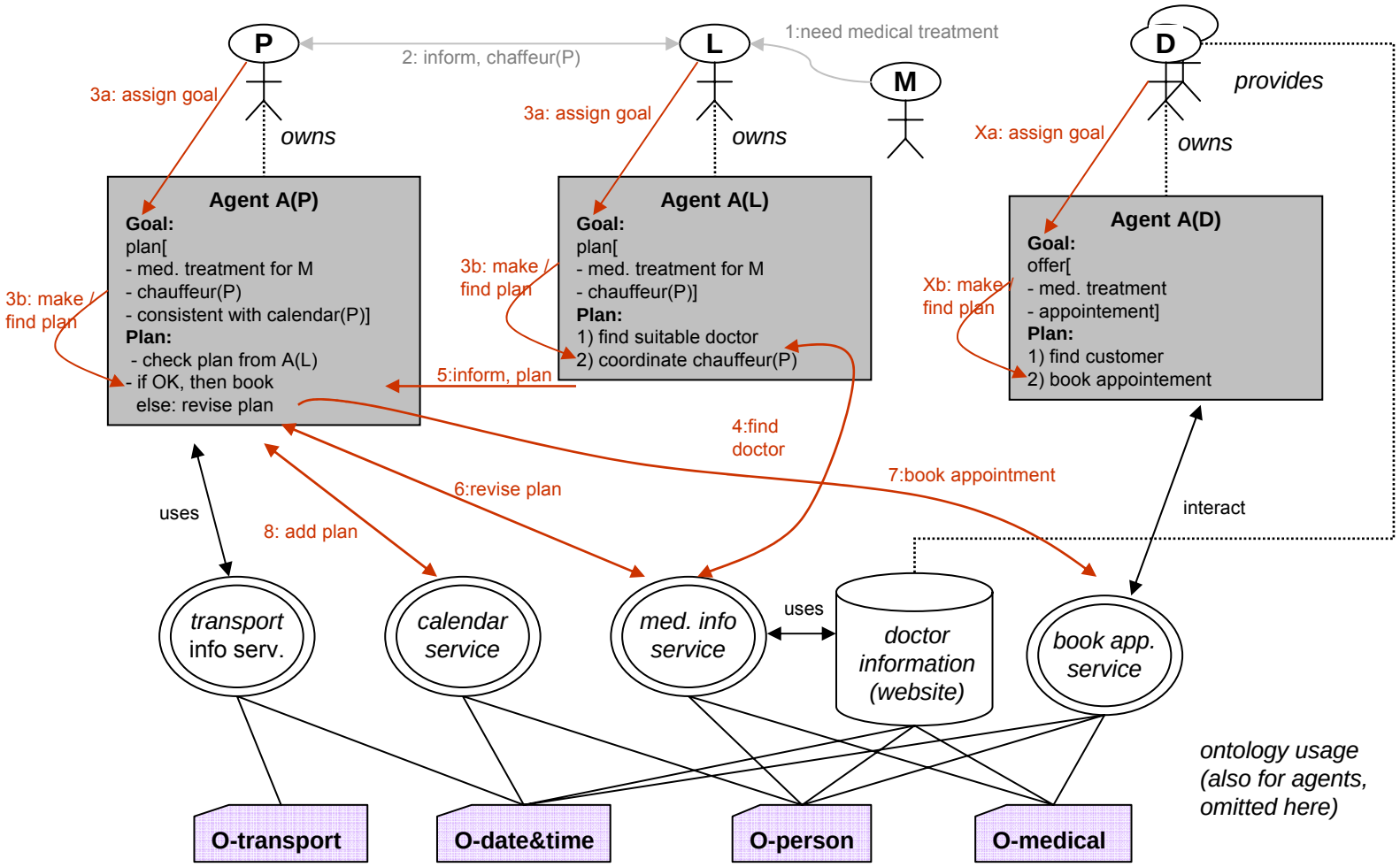
Michael Stollberg
DERI – Digital Enterprise Research Institute
University of Innsbruck, Austria

Doctoral Symposium at the 5th International Conference on Web Engineering (ICWE 2005)
Sydney, Australia, July 24th

Content

1. Aim, Approch, and Scope
2. Model, Elements, and Components
3. Collaboration Establishment
4. Semantic Web Fred (Prototype)
5. Conclusions

Motivating Example

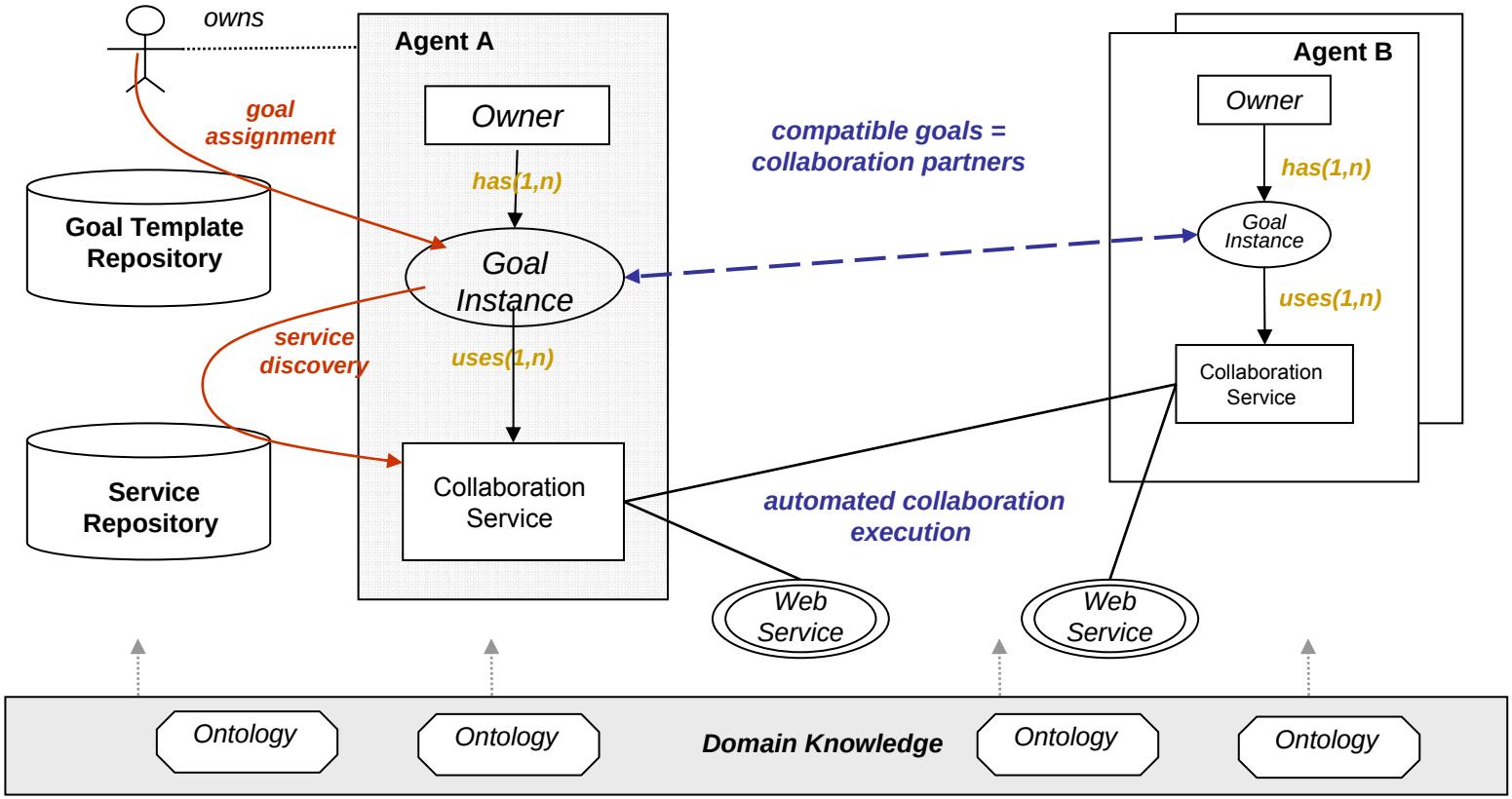


Aim, Approach, and Scope

- Framework & Prototype for Automated Collaboration on the Semantic Web
  - Model of Agency
  - Semantic Element Description
  - Semantically driven Collaboration Establishment
  - Prototype

- Extension of the Web Service Modeling Ontology WSMO
  - comprehensive framework for Semantic Web Services
  - core elements: Ontologies, Goals, Web Services, Mediators
  - serves as basis for element descriptions
  - compatibility / usability of WSMO technologies for SW(S)

Conceptual Architecture



Ontologies as Data Model

- **Ontologies are used as the 'data model'**
  - all element descriptions rely on ontologies
  - all data interchanged in collaborations usage are ontologies
  - semantic information processing & ontology reasoning
- **Ontology Language WSML (Web Service Modeling Language)**
  - conceptual syntax for describing WSMO elements
  - well-layered logical language for axiomatic expressions (WSML Layering)
  - Ontology design: modularization & decoupling
- **Ontology Technology needed:**
  - ontology management (editing, maintenance)
  - ontology instance data handling
  - ontology integration techniques (mapping, merging, alignment)

Ontology Specification

- **Non functional properties** item description for management
- **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)

Model of Agency

electronic representative of a real world entity
that wants to achieve individual objectives
by collaborative interactions with others

- **Requirements**
  - represent owner (optional: agent collaboration)
  - Goals for specifying collaboration objectives (via task delegation by owners)
  - dynamic usage of Collaboration Services as facilities for participating in automated collaborative interactions over the (Semantic) Web
  - semantic description (all data based on ontologies)
- **MoA extends "Collaborative Interface Agents"**
  - general agents properties: autonomous, social, re-& proactive
  - re- & productivity by interaction with user and other agents
  - extensions: semantic description, Collaboration Services, Coll. Management

Agent Semantic Description

Agents

electronic representative of real world entity that wants to achieve an objective by automated collaboration

Class agent
- hasNonFunctionalProperties **type** nonFunctionalProperties
- importsOntology **type** ontology
- usesMediator **type** ooMediator
- owner **type** owner
- collaboration **type** collaboration
- history **type** collaboration

Class owner
- hasNonFunctionalProperties **type** nonFunctionalProperties
- owner **type** instance
- preference **type** axiom
- policy **type** axiom
- serviceUsagePermission **type** service

Class collaboration
- hasNonFunctionalProperties **type** nonFunctionalProperties
- goal **type** goal *single-valued*
- service **type** service

SWF

Goals

Objective to be achieved that a user (agent owner) delegates to an agent for automated resolution

- **Definition and Usage**
  - denotes state of the world that is to be achieved (formal / machine-readable specification)
  - serves as facility for automated usage of Collaboration Service
  - semantic formal / machine-readable specification for automated processing (desired state defined by logical expressions on domain ontologies)

- **Goal Driven Architecture**
  - objective definition independent of available facilities (decoupling "desires" and "requests")
  - automated goal resolution (dynamic discovery of partners and resources for resolution)
  - allows dynamic use & re-use of Services for different purposes
  - derived from different AI-approaches for intelligent systems (agent BDI architectures, PSMs, Planning / Service Composition)

- **Goal Templates and Goal Instances**
  - Goal Templates as "pre-defined schemas of Goals resolvable in system / application"
  - user defines Goal Instance for concrete objectives by instantiating a Goal Template and assigning it to an agent for automated resolution

Goals Semantic Description

Goal Template

predefined schema of user objective, described as WSMO 1.0 goal

Class goalTemplate **is-a** wsmov1.0Goal

hasNonFunctionalProperties **type** nonFunctionalProperties
importsOntology **type** ontology
usesMediator **type** {ooMediator, ggMediator}
hasPostcondition **type** axiom
hasEffect **type** axiom

Goal Instance

concrete objective assigned to an agent, instantiated Goal Template, client for service usage

Class goalInstance **sub-Instance** goalTemplate

hasNonFunctionalProperties **type** nonFunctionalProperties
importsOntology **type** ontology
usesMediator **type** ooMediator
hasPostcondition **type** axiom
hasEffect **type** axiom
hasSubmission **type** instance
hasResult **type** instance
goalResolutionStatus **type** goalResolutionStatus

Collaboration Service Description

- complete item description
- quality aspects
- service management

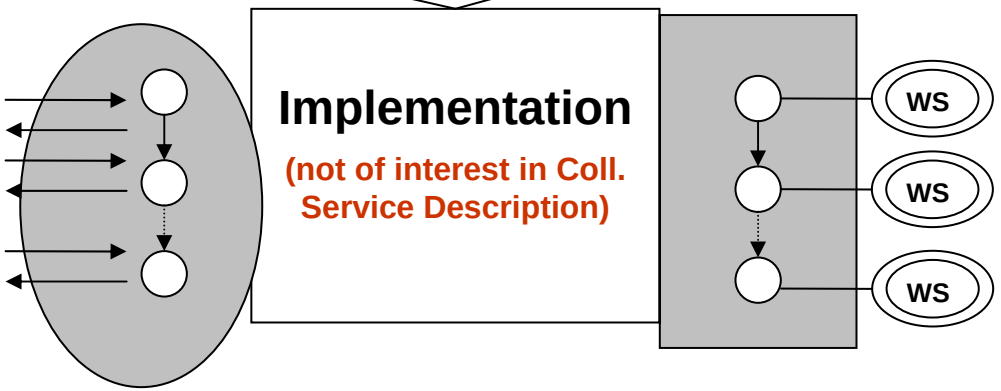
Non-functional Properties

DC + QoS + Version + financial

- service advertisement
- support for service discovery

Capability

functional description



interface for service consumption (agents = clients):
- external visible behavior
- communication structure
- 'grounding'

realization of functionality by:
- interaction with CS of other agents
- usage of other web rescoures / WS

Client Interface --- Service Interfaces --- **Orchestration**

(sub-class of WSMO Choreography) *(sub-class of WSMO Orchestration)*

SWF

Collaboration Service Description

Collaboration Services

facility an agent uses for participating in an automatically executed collaboration
interacts with other agents and utilizes (Semantic) Web resources via its orchestration

Class collaborationService **is-a** wsmoService
- hasNonFunctionalProperties **type** nonFunctionalProperties
- importsOntology **type** ontology
- usesMediator **type** ooMediator
- hasCapability **type** capability *single-valued*
  - hasSharedVariables **type** sharedVariable
  - hasPrecondition **type** axiom
  - hasAssumption **type** axiom
  - hasPostcondition **type** axiom
  - hasEffect **type** axiom
- hasInterface **type** interface
  - clientInterface **type** serviceInterfaceDescription
  - orchestration **type** serviceInterfaceDescription

Class serviceInterfaceDescription **sub-Class** wsmoServiceInterface
- hasNonFunctionalProperties **type** nonFunctionalProperties
- importsOntology **type** ontology
- usesMediator **type** ooMediator
- hasVocabulary **type** concept_in, concept_out, concept_controlled
- hasState **type** ontology
- hasGuardedTransition **type** if (condition) then *update* – for client interface
  if (condition) then *operation(CS, R, update)* – for orchestration

Capability Specification

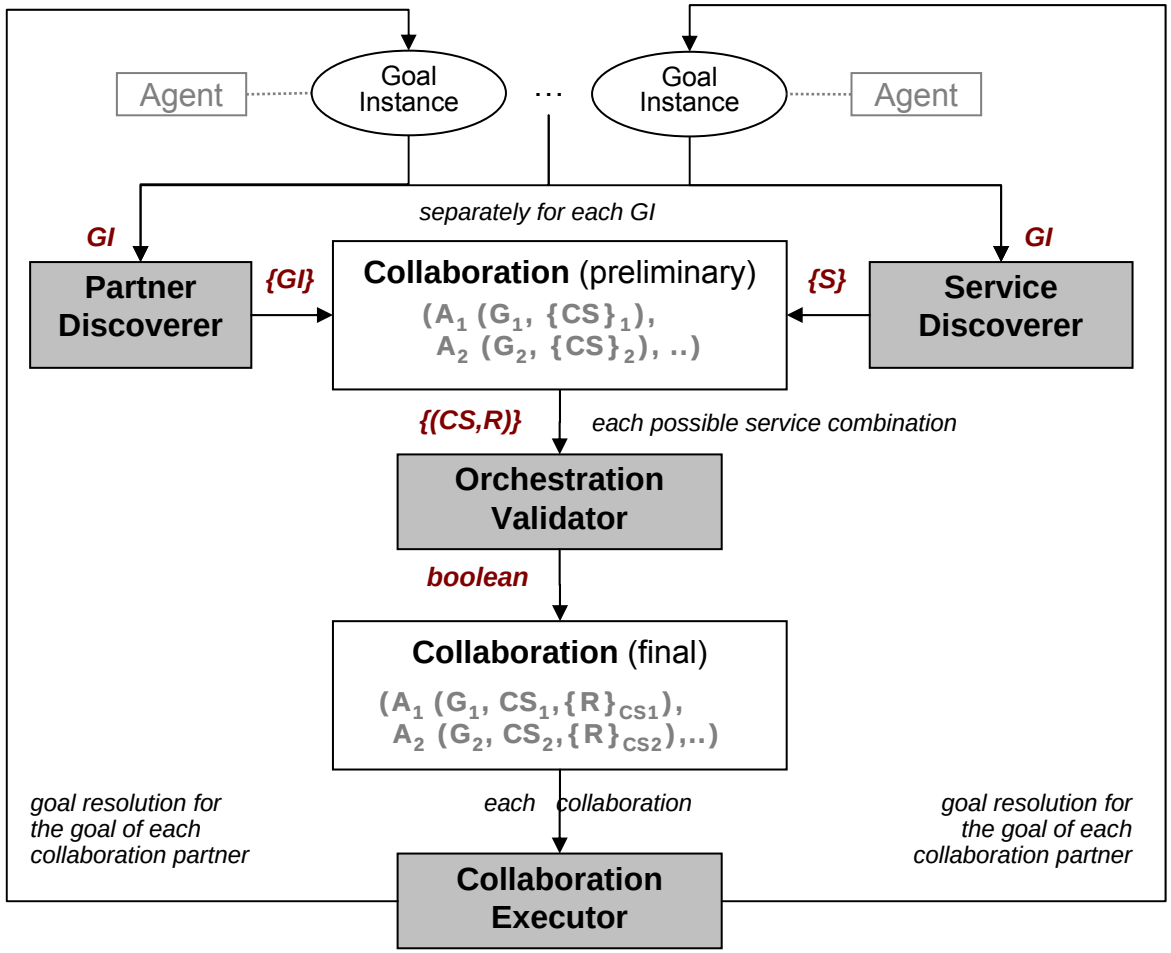
- **Non functional properties**
- **Imported Ontologies**
- **Used mediators**
  - *OO Mediator:* importing ontologies with mismatch resolution
  - *WG Mediator:* link to a Goal wherefore service is not usable a priori
- **Pre-conditions**
  what a web service expects in order to be able to provide its 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)

Service Interface Description Model (adopted from WSMO)

- 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*

SWF

Collaboration Management



Semantically Driven Discovery

find appropriate (Web) facility for automatically resolving a goal as the objective of a requester

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

SWF

Action and Object Knowledge

- Knowledge Types Distinction
  - Action = activity to be performed on object (e.g. buy, sell, ..)
  - Object = entity that action is performed on (e.g. product, ticket, ...)
- Action and Object Knowledge is different
  - "Action" defines what is to be done; interacting entities need to have **compatible actions** (e.g. buy <-> sell)

  => Action-Resource Ontology
  - "Object" defines whereon an action is to be performed; interacting entities need to have **not-contradicting object definitions**

  => Set-theoretic Object Matchmaking

- **Combination is needed** (agents / resources might have not-contradicting objects but incompatible actions)

Action-Resource Ontology

```
concept action
  compatibleAction symmetric ofType action
concept buy subConceptOf action
  compatibleAction symmetric ofType sell
concept sell subConceptOf action
  compatibleAction symmetric ofType buy
```

Action Taxonomy

```
concept resource
  hasAction ofType set action
concept goal subConceptOf resource
concept service subConceptOf resource

concept buyergoal subConceptOf goal
   hasAction ofType set buy
concept sellerservice subConceptOf service
   hasAction ofType set sell
```

Resource Taxonomy

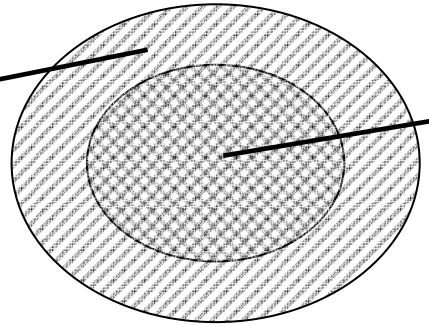
all resources are defined as instances of resource types

- **allows determining action compatibility / equality of agents & resources**
- **supports handling multi-party collaborations**

Object Matchmaking

Set-Based Resource Descriptions

Information Space
all possible instances
of used ontologies



Description Notion
all possible instances that
satisfy restricted information space

```
postcondition
  definedBy
    exists ?PurchaseItem(?PurchaseItem[
      item hasValue ?PurchaseFurniture
    ] memberOf swfmo:product) and
    exists ?PurchaseFurniture(?PurchaseFurniture[
      material hasValues {wood},
    ] memberOf furn:chair) and
  ?X[
     purchaseItem hasValue ?PurchaseItem,
     buyer hasValue kb:MichaelStollberg,
     purchasePayment hasValue kb:MSCreditCard
   ] memberOf swfmo:purchaseContract .
```

Goal Instance Postcondition

- Objective: receive a purchase contract for a wooden chair for Michael Stollberg, payment with credit card

- meta-varibale X (dynamically quantified by matchmaking notion)

- restrictions on several ontology notions

- WSML syntax

Set Theoretic Matchmaking Notions

1. Exact Match:

   $D_Q, D_R, O, M \models \forall x.\ (D_Q <=> D_R)$

2. PlugIn Match:

   $D_Q, D_R, O, M \models \forall x.\ (D_Q => D_R)$

3. Subsumption Match:

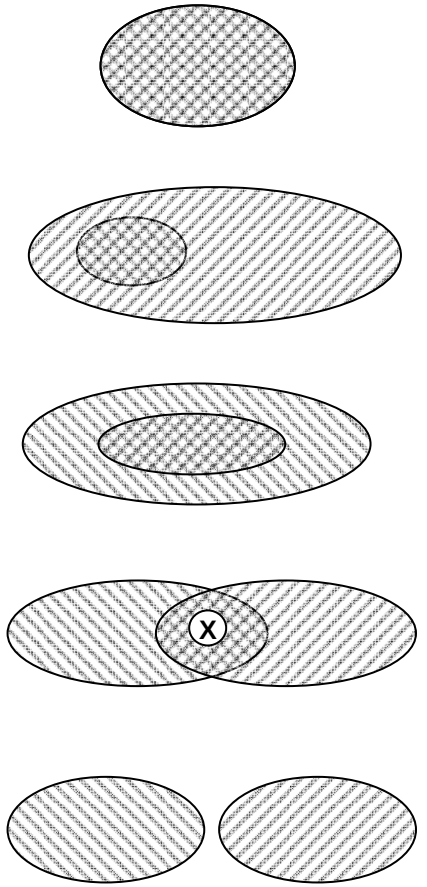
   $D_Q, D_R, O, M \models \forall x.\ (D_Q <= D_R)$

4. Intersection Match:

   $D_Q, D_R, O, M \models \exists x.\ (D_Q \wedge D_R)$

5. Non Match:

   $D_Q, D_R, O, M \models \neg\exists x.\ (D_Q \wedge D_R)$



Realization in VAMPIRE

Structure of a Proof Obligation for discovery

1. **Needed Knowledge Resources**
   - include Ontology Theory and Universe
   - Knowledge Base optionally

2. **Resource Description notion to be matched**
   - Request Resource & Result Resource
   - Only those notions that are to be matched

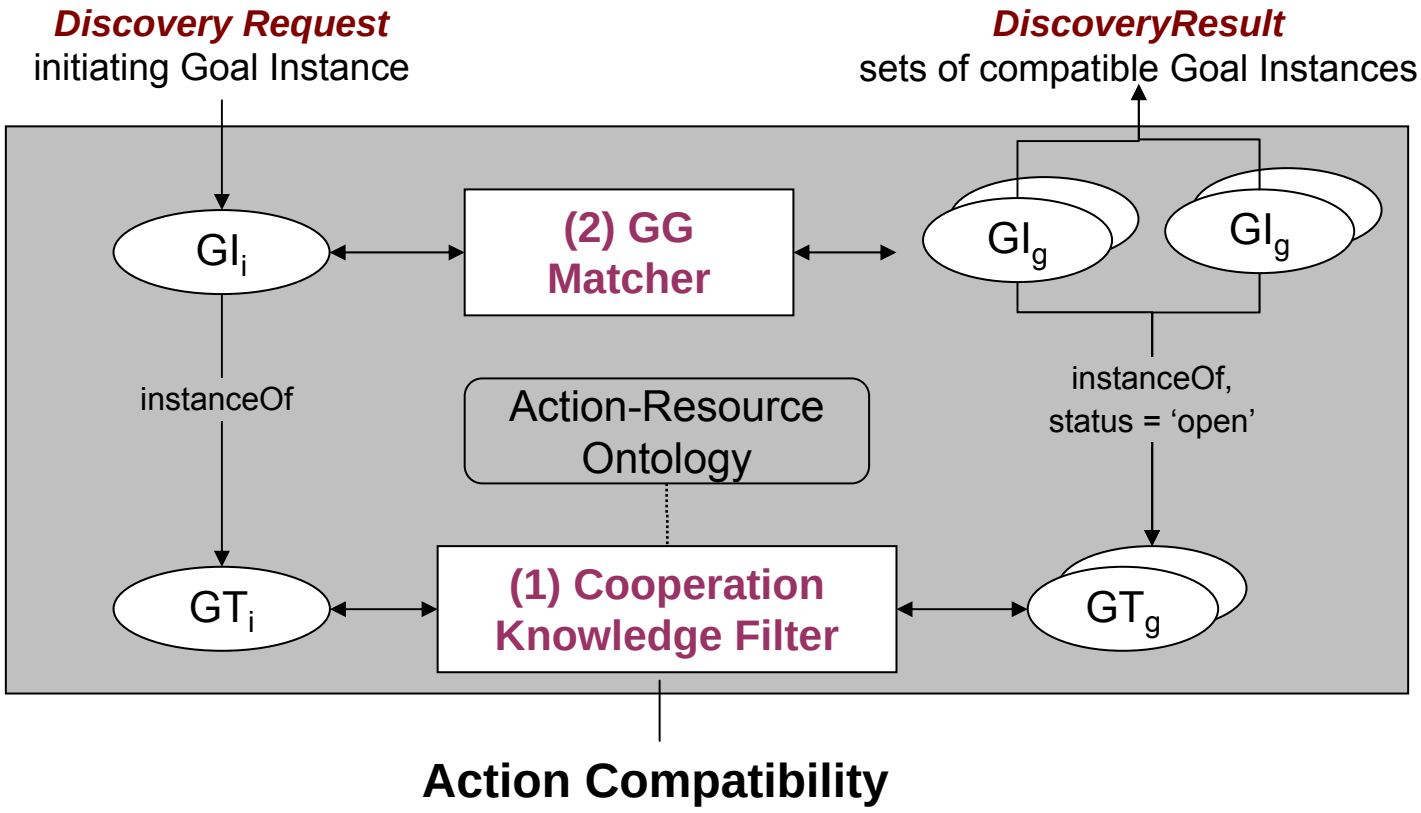
3. **Object Matchmaking notion**
   - Include as specified in the Discovery Request

=> *easy to generate dynamically*

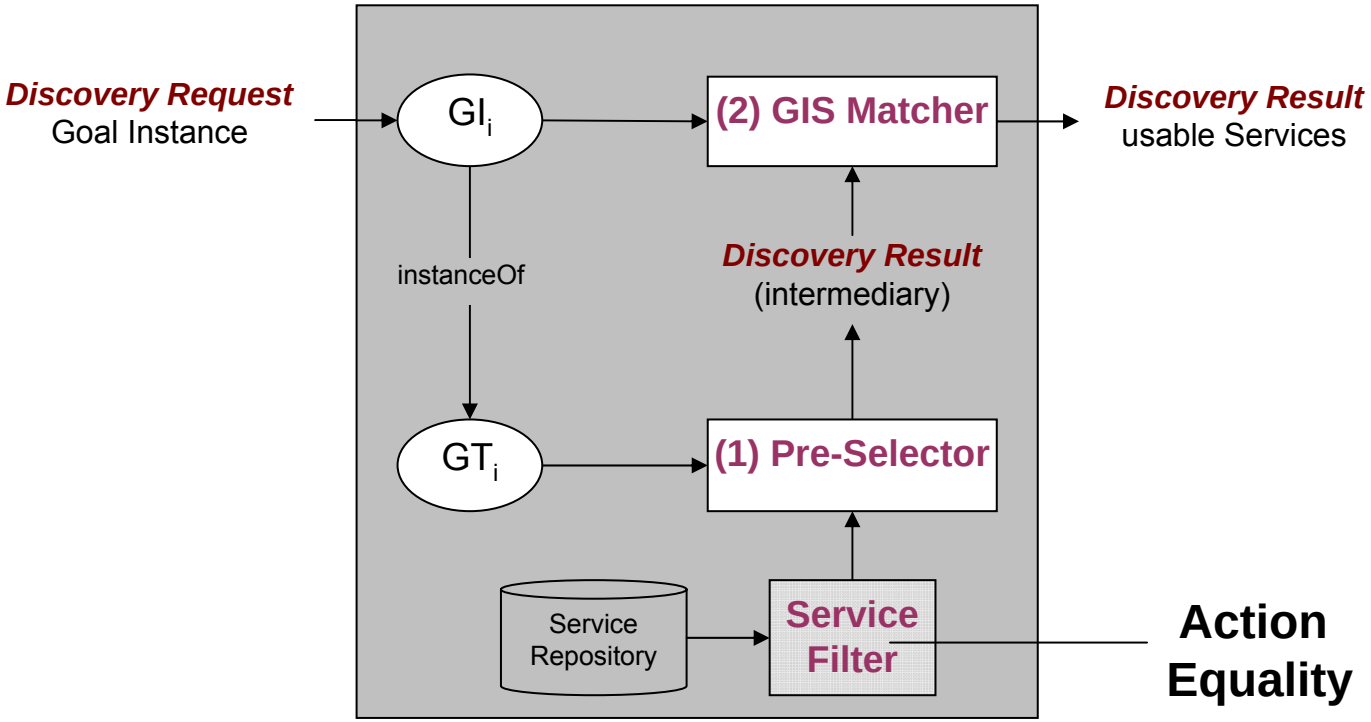
Partner & Service Discovery Components

- run independently, invoked by agents
- apply ARO & OMM as defined above
- Design Principles:
  1. Common Discoverer Architecture
  2. Modularized Functionality
  3. Layered Architecture (stepwise narrow search space)
  4. Reduction of expensive operations (efficiency)

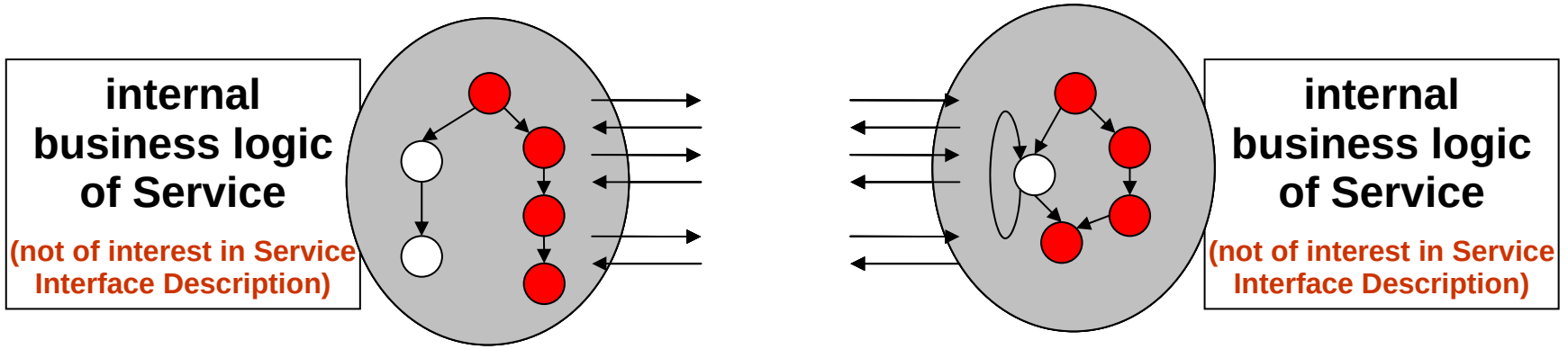
Partner Discoverer Architecture



Service Discoverer Architecture



Choreography Discovery



- a valid choreography (interaction protocol) 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:

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

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

- 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:

$\omega_\varnothing(C(S1, S2))$ **if** $\Omega S1(\omega_\varnothing) = \varnothing$ and $\Omega S2(\omega_\varnothing) = \varnothing$ 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** $\Omega S1(\omega_T) = \text{noAction}$ and $\Omega S2(\omega_T) = \text{noAction}$ 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

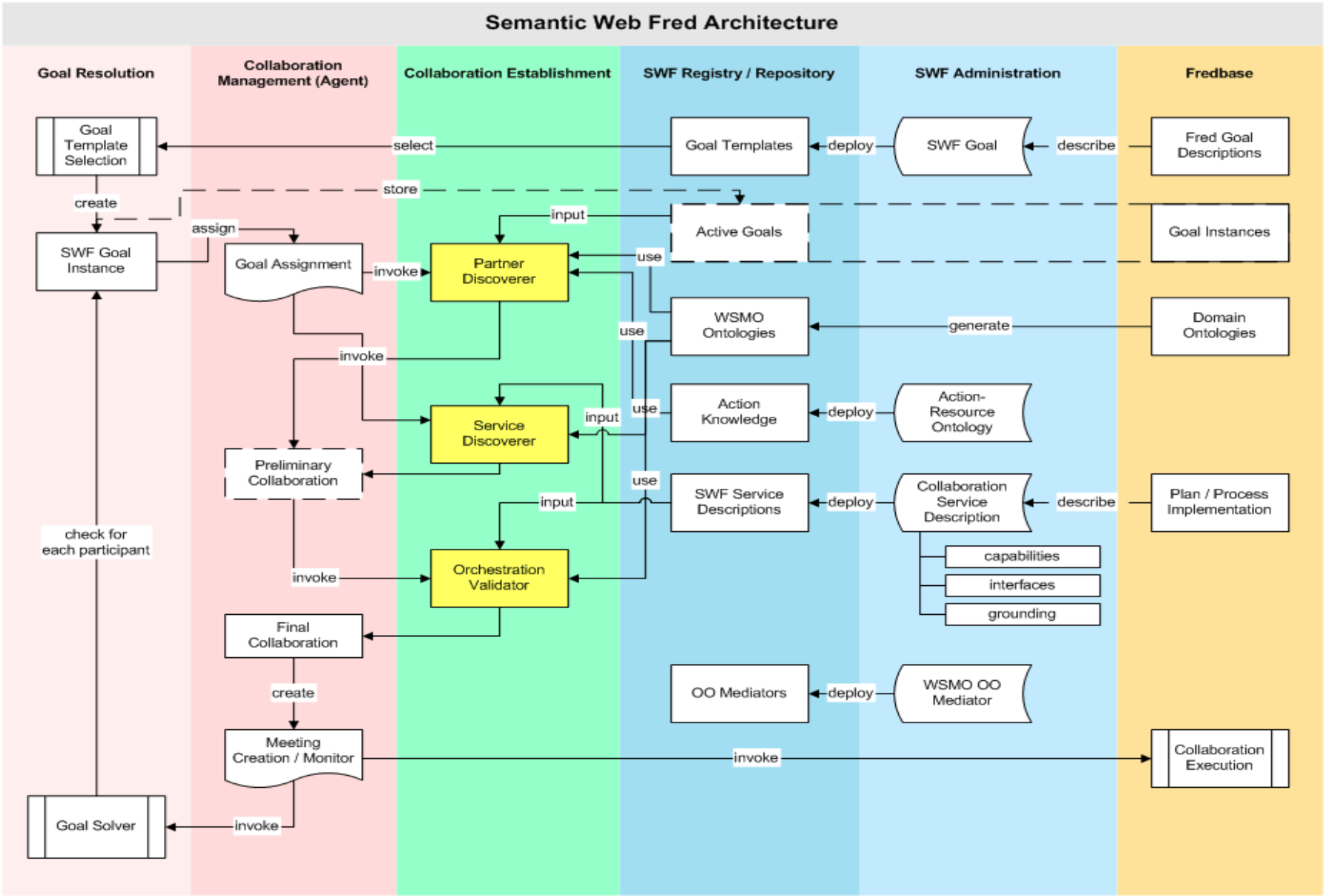
Orchestration Validation

discover valid choreography for all interactions that need to be performed for automated execution of a Collaboration

- **layered architecture / stepwise validation wrt efficiency:**
  - "multi-party choregraphy discovery"
  - Process:
    1. validate orchestration for CS of initiating agent
    2. validate orchestration of CS of each partner agent
  - for each binary choreography:
    1. determine information compatibility (less expensive operation)
    2. if (1), then determine communicatuion compatibility (more expensive operation)

- **under construction:**
  - choreography discovery prototype with VAMPIRE
  - Orchestration Validator implementation only checks on usage of homogeneous ontologies

Semantic Web Fred



Technologies Used

- **System Languages**
  - functional components in Java
  - GUIs in VB6
- **Smart Objects (ontology handling & management technology):**
  - Ontologies handled as Java objects
  - Ontology / SMO management facilities
- **UDDI Repository, conventional RDB as repository**
- **Collaboration Execution**
  - Meeting Rooms (central service, controls & monitors collaboration executions)
  - FIPA ACL as Collaboration Service Interaction Language
- **Web / Semantic Web / Web Service support**
  - Web Service usage via WSDL-Executor
  - WSMO registry(ies) alignment
  - Semantic Web Resource support (e.g. connection to YARS)

Conclusions

- Framework for Automated Collaboration on the Semantic Web
  - epistemology of collaborative interactions
  - coherent technology framework, combining most recent technologies
  - semantically enabled collaboration establishment / management
  - prototype realization & testing
- Main Results:
  - conceptual architecture & element definitions
  - Model of Agency, Goal & Service descriptions and usage
  - semantically enabled collaboration establishment
- Status of work:
  - conceptual / theoretic work nearly completed
  - development in SWF project is ongoing
  - expected finalization: end 2005 / spring 2006

Relation to Web Engineering

- WE aspects mostly in collaboration execution
  - Collaboration / Web Service invocation
  - automated information interchange over Web
  - decentralized / autonomous computation
  - ⇒ currently grounded to conventional Web (Service) technology (NOT what we want)

- insights in "Semantic Web Engineering":
  1. **"inference engines far away from industrial applicability"**
  - only limited / partial **Reasoning Support**
  - tremendous **deficiencies in stability, computation time, scalability**
  - **Reasoning Requirements still unclear** (light-weight object matchmaking vs. complex FOL reasoning)

  2. **"hide logics from users and system developers"**
  - semantic descriptions by **Knowledge Engineers**
  - exhaustive **tool support** required for users and system developers
  - **abandon logical expressivity** if it hampers automation