From Component-Based to Service-Oriented Computing: Towards Self-Evolution

Eugenio Zimeo

University of Sannio - ITALY

eugenio.zimeo@unisannio.it

FACS 07 - Sophia Antipolis - Sept. 21st, 2007

research centre on software technology

Component-Oriented Model

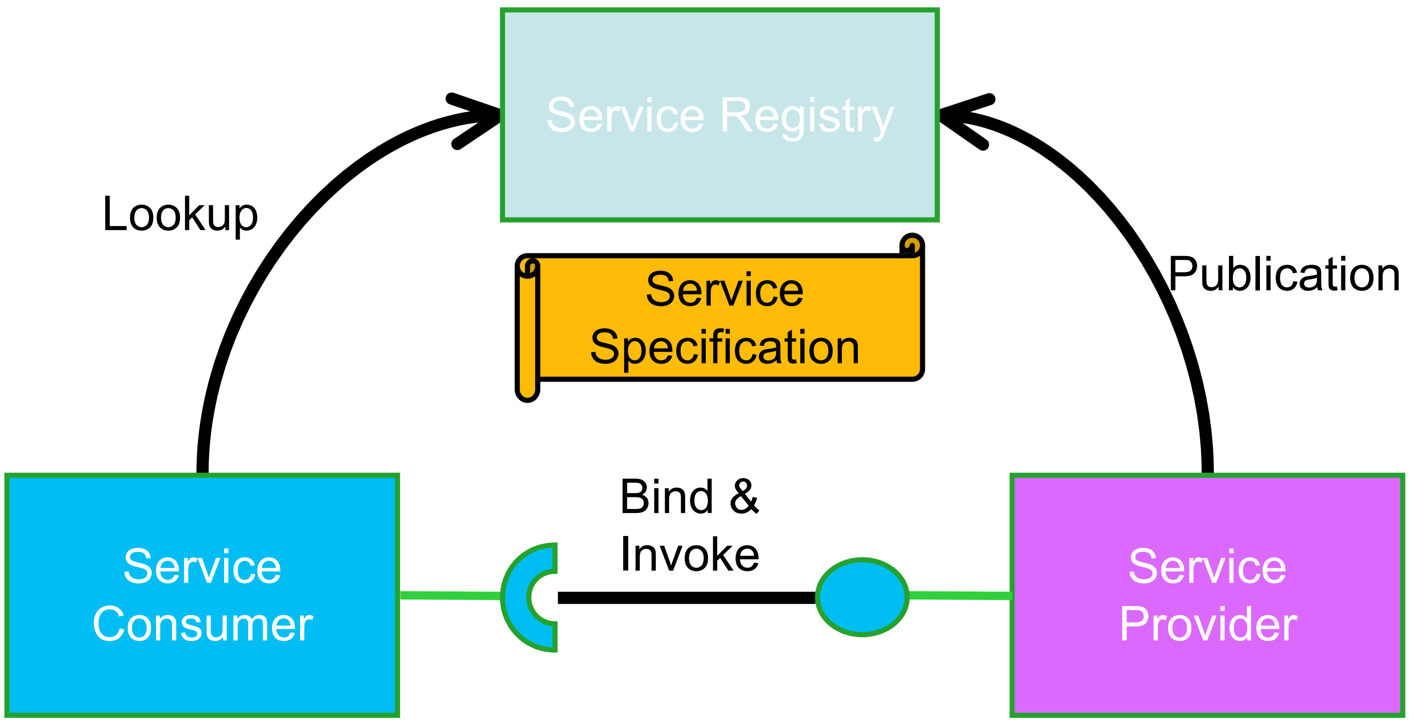
- A component is a self contained entity that interacts with its environment through well-defined interfaces
- A component type
  - consistent piece of code
  - non-functional concerns configuration
  - defined interfaces (required and provided)
- A component instance
  - Content: business code
  - Container: manage non functional concerns
    - Binding, Lifecycle, Persistence, Security, Transaction …

Component model: benefits

- The model helps the implementation and maintenance of complex software systems
  - Focus on application building block definition
  - Creating reusable software building blocks
  - Separation of concerns - between functional (business code) and non-functional aspects
  - Avoid monolithic application - applications are created by composing (existing) components

Service-Oriented Model

Service: Contract of defined behavior



Service-Oriented Model

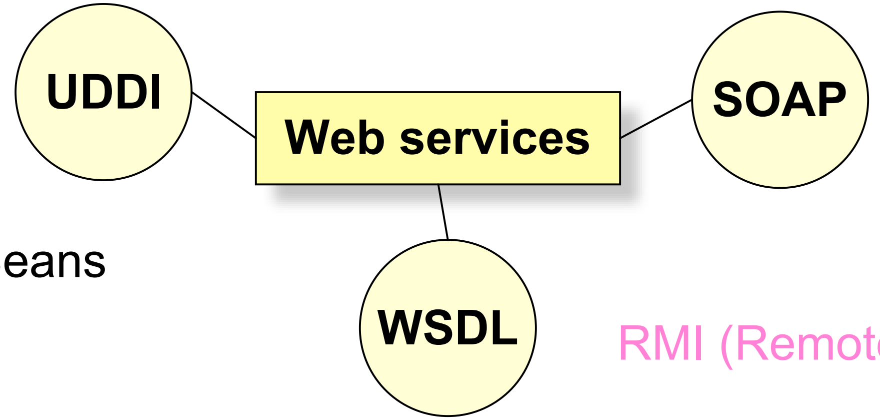
- Ideal for dynamic environments
  - Loose-coupling
    - *Design by Contract*
  - Late-binding
    - At runtime, on demand
  - Hide heterogeneity
- Issues
  - Dynamic in nature
    - Service arrive/disappear dynamically
  - Service dependencies are unreliable and ambiguous
    - No service found or multiple found
  - Requestors do not directly instantiate service instances
    - Common service or different instances

Web Services Model

Web services are encapsulated, loosely coupled Web “components” that can bind dynamically to each other

Why Web Services?

Web Services



Jini

Enterprise Java Beans

RMI (Remote Method Invocation)

Microsoft DCOM

CORBA (Common Object Request Broker Architecture)

Open Software Foundation DCE (Distributed Computing Environment)

Sun ONC/RPC (Open Network Computing)

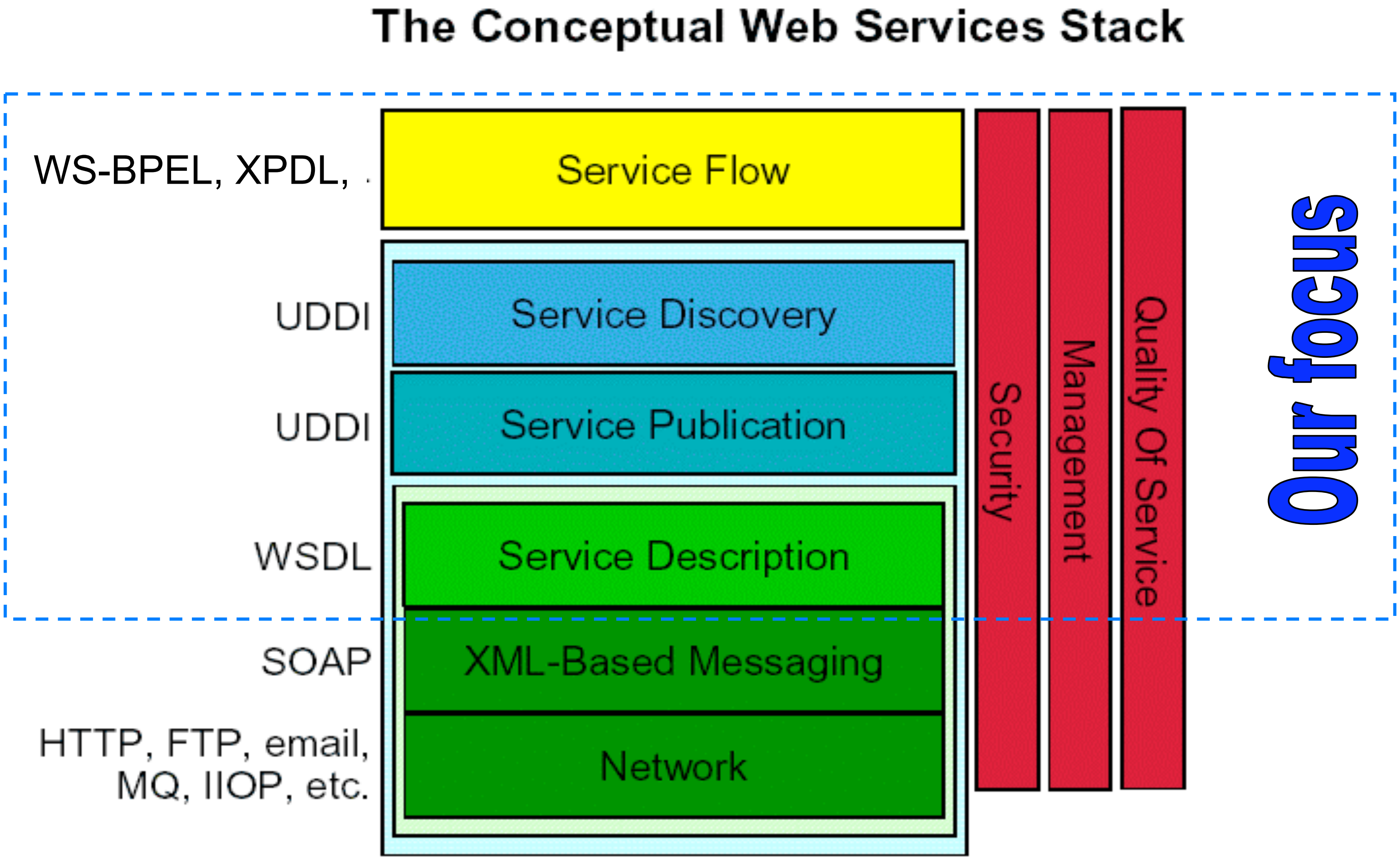
IP, UDP, TCP

Web Services Technology

A very brief overview

research centre on software technology

Web Service Stack

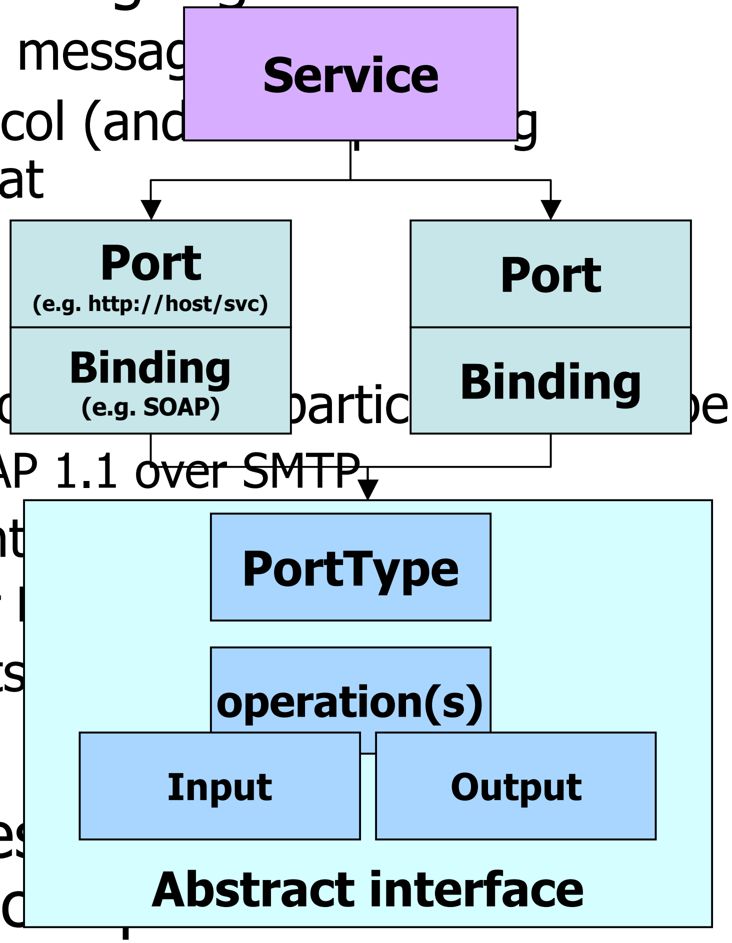


Service description: WSDL

- WSDL goes beyond traditional IDL languages
  - Abstract definitions of operations and messag
  - Concrete binding to networking protocol (and g
    endpoint address) and message format

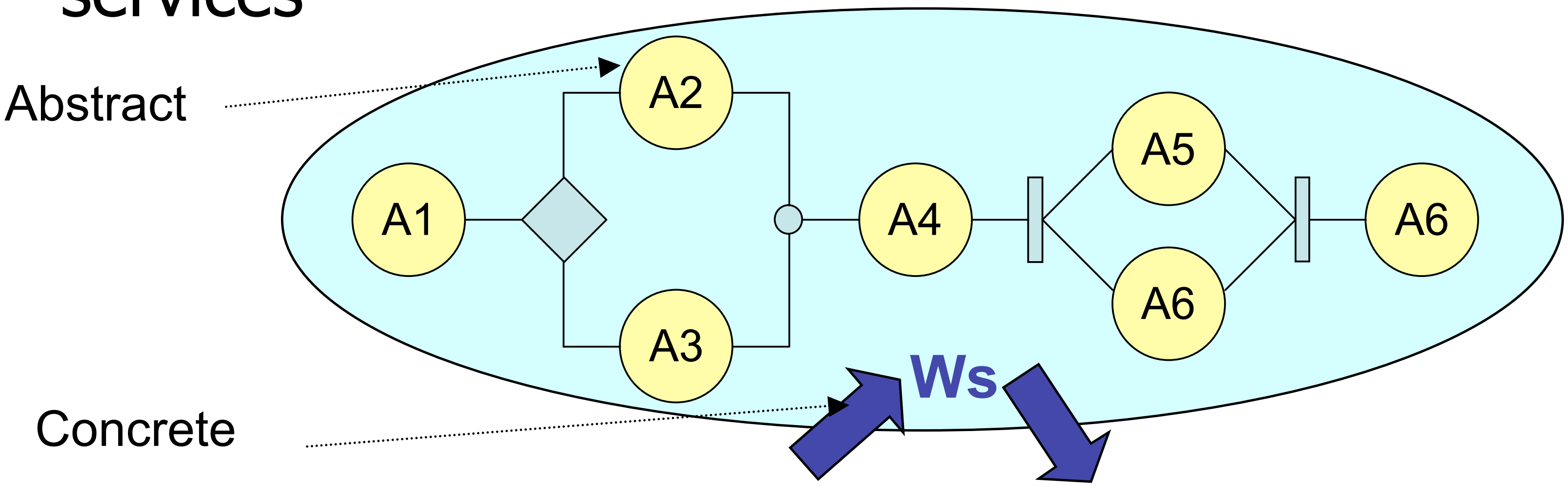
- Component model (binding)
  - Binding: concrete protocol and data fo partic e
    - example: SOAP 1.1 over HTTP or SOAP 1.1 over SMTP
  - Port: a single communication endpoint
    - Endpoint address for binding, URL for
  - Service: aggregate set of related ports

- Allows advertisement of service des
  dynamic discovery and binding of c
  - Used in conjunction with UDDI registry



Service flow

- Creating web processes from composite web services



  - WS-BPEL - WS Business Process Execution Language
  - XPDL - XML Process Definition Language

WS-BPEL (1/2)

WS-BPEL

- WS-BPEL (WS Business Process Execution Language) is a process modeling language.
  - Developed by IBM, Microsoft, and BEA
  - Version 1.1, 5 May 2003
- It supercedes XLANG (Microsoft) and WSFL (IBM).
- It is build on top of WSDL.
  - For descriptions of what services do and how they work, WS-BPEL references port types (interfaces) contained in WSDL documents.

WS-BPEL (2/2)

- WS-BPEL is a block-structured programming language, allowing recursive blocks but restricting definitions and declarations to the top level
- The language defines activities as the basic components of a process definition
- Structured activities prescribe the order in which a collection of activities take place
  - Ordinary sequential control between activities is provided by sequence, switch, and while
  - Concurrency and synchronization between activities is provided by flow
  - Nondeterministic choice based on external events is provided by pick

XPDL (1/2)

XPDL

- XPDL (XML Process Definition Language) is a process modeling language
  - XPDL 1.0 was officially released by the WfMC in October '02
  - XPDL 2.0 was officially approved by the WfMC in October'05
- It is built by exploiting the experience of WPDL (Workflow Process Definition Language), the first WfMC standard interchange language
- Petri Nets influenced the development of XPDL

XPDL (2/2)

- Each step in the process is an activity providing some attributes that give information about
  - who can perform the activity
  - what application or WS should be invoked
  - ....

- To indicate branching, XPDL offers routing activities

- The nodes and transitions can form arbitrarily complex graphs with
  - Sequential Activities
  - Parallel Activities
  - Loops/Cycles
  - Conditional Paths

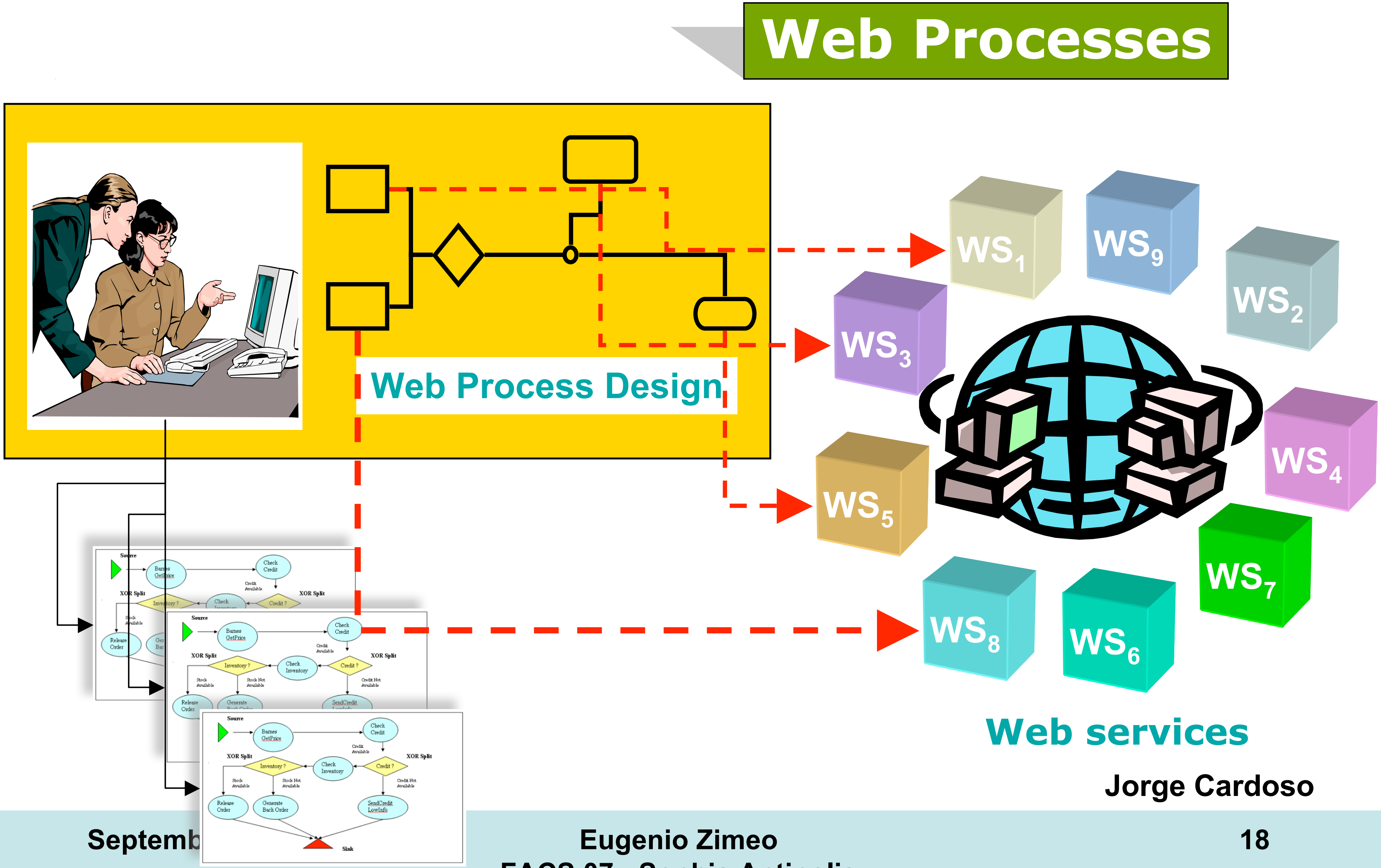
Web Processes

What are Web Processes ?

- **Web Processes** are next generation workflow technology to facilitate the interaction of organizations with markets, competitors, suppliers, customers etc. supporting enterprise-level and core business activities
  - encompass the ideas of both intra and inter organizational workflow
  - created from the composition of Web services
- When all the tasks involved in a Web process are semantically described, we may call such process as **Semantic Web Processes**

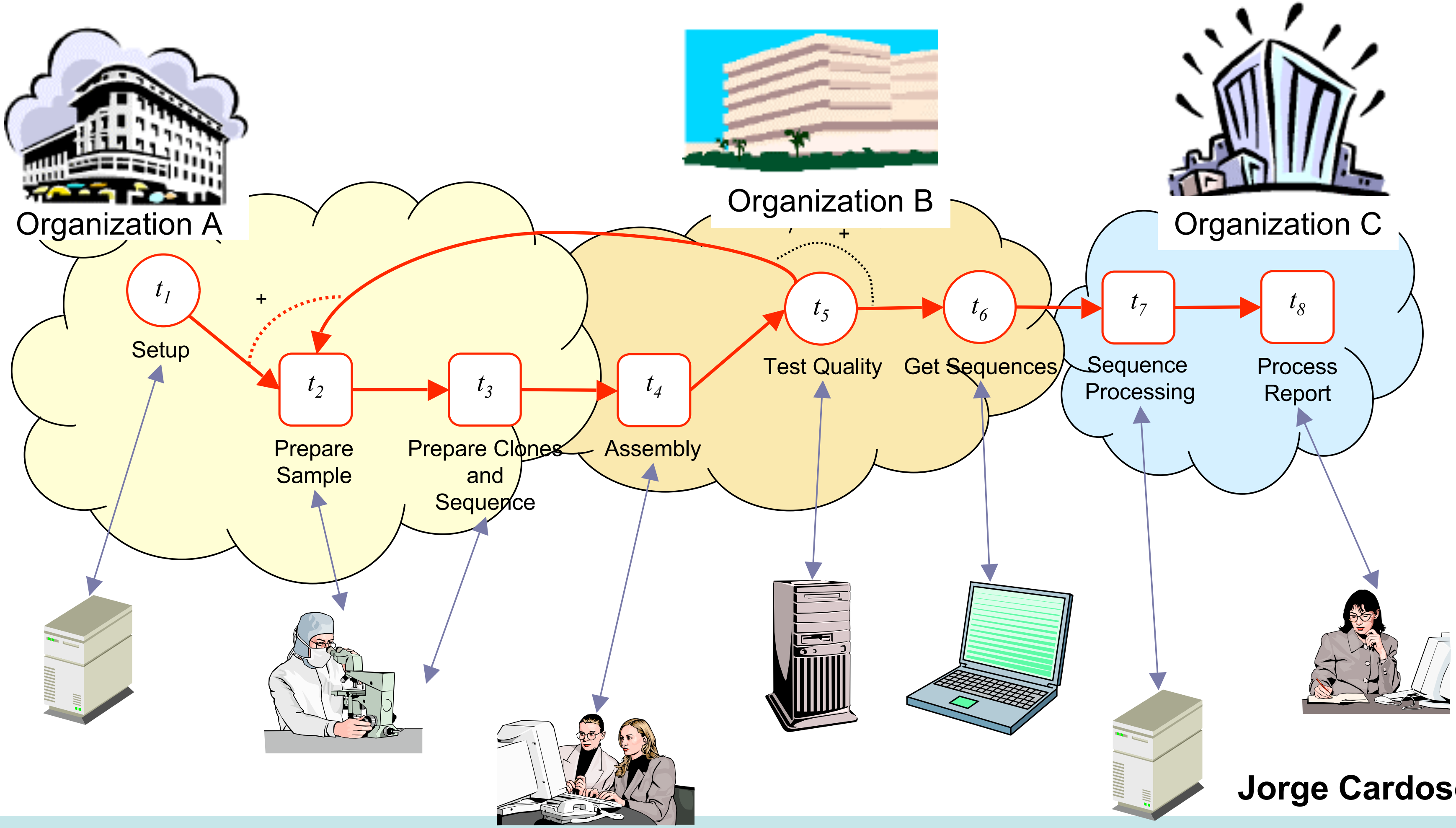
Jorge Cardoso

Web Processes: composition



Web Processes: deployment

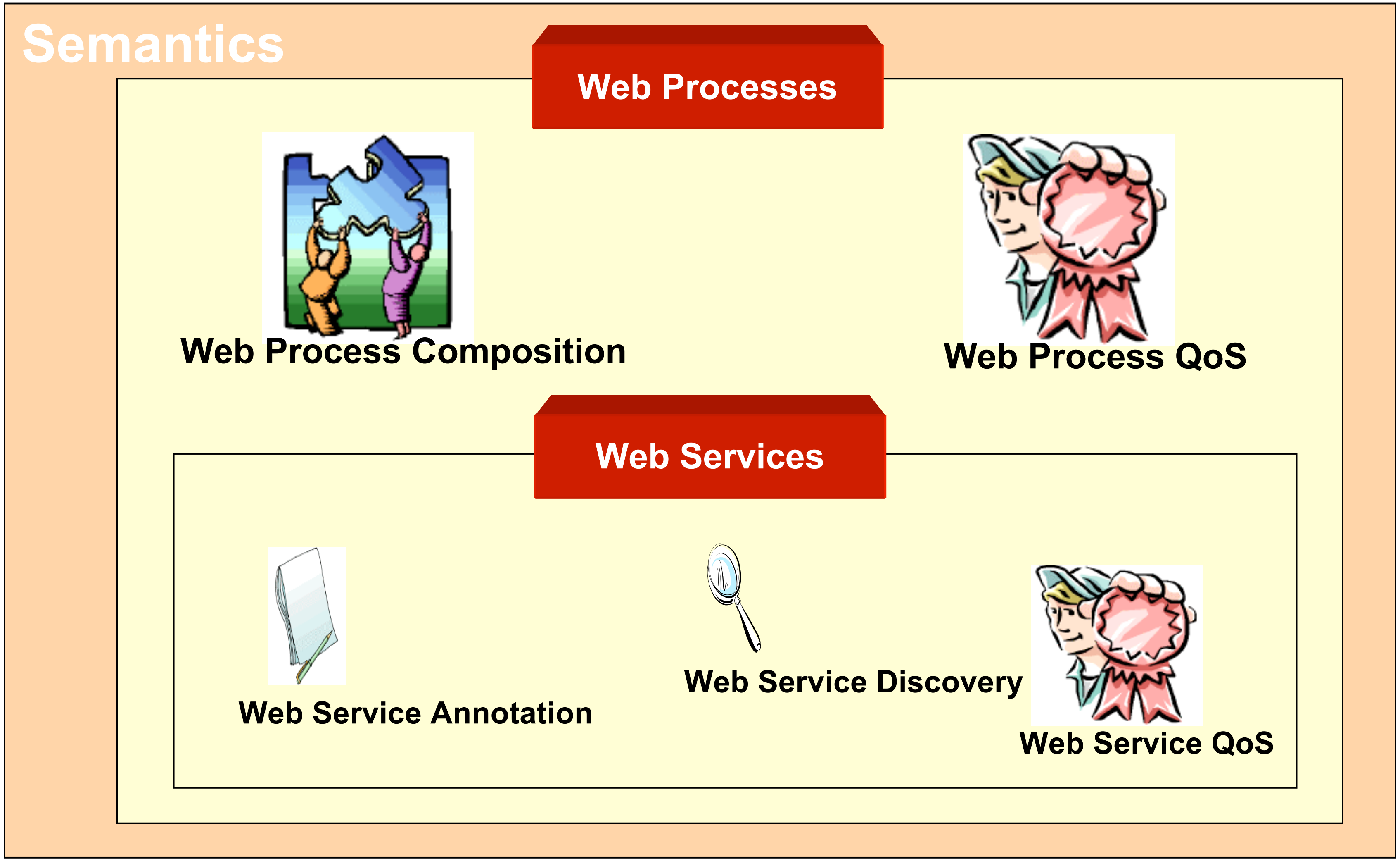
Web Processes



Jorge Cardoso

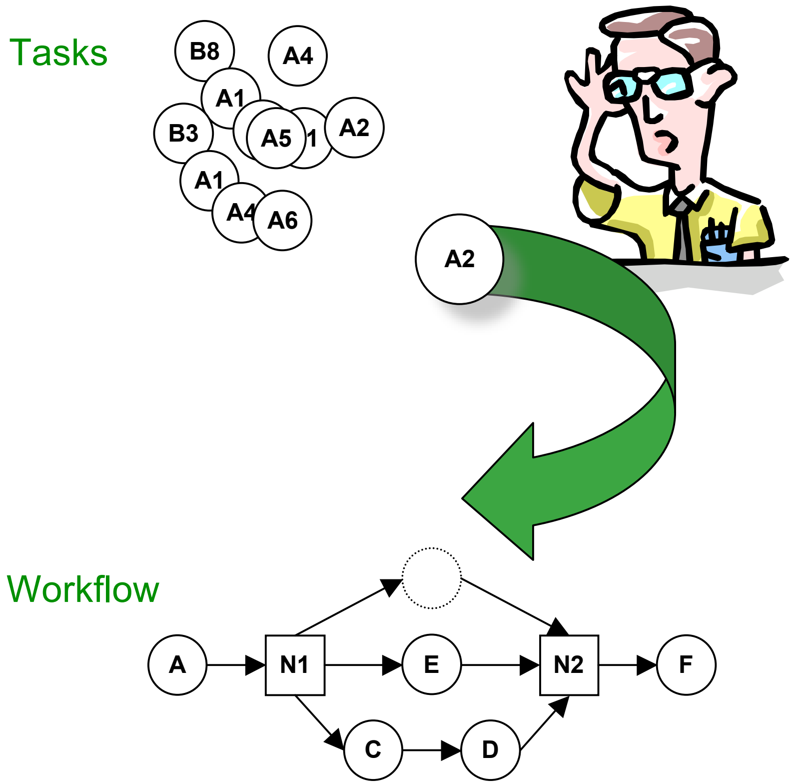
Semantic Web Processes

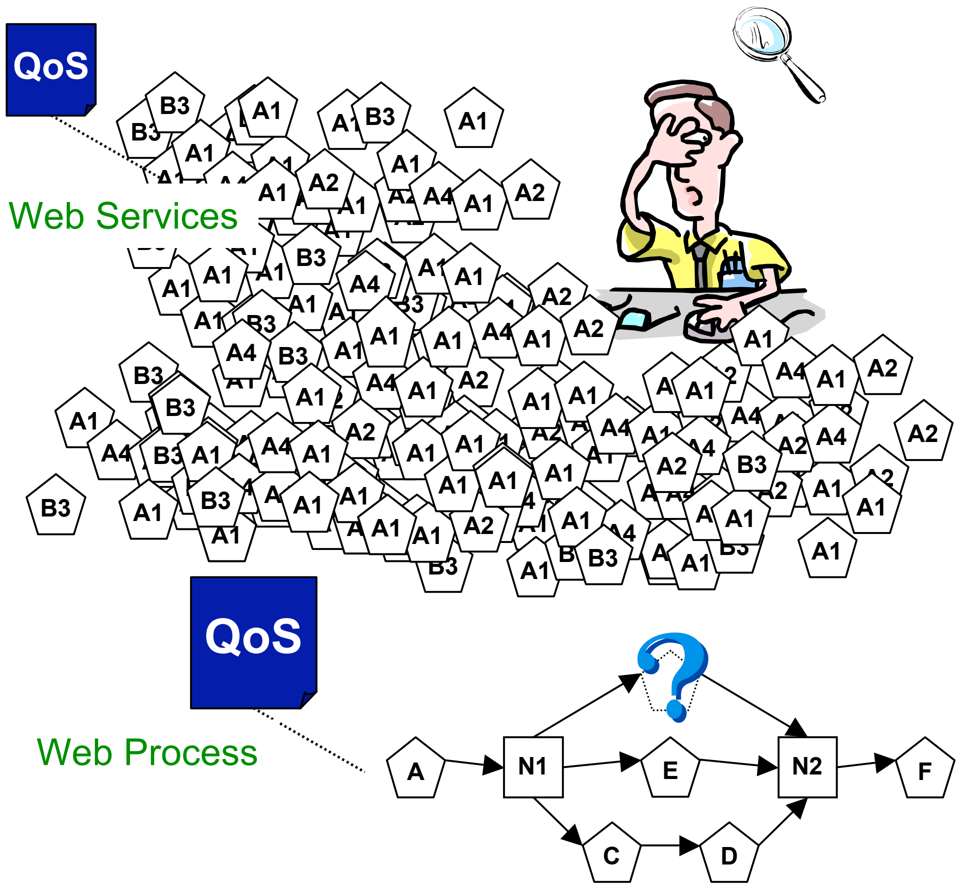
RCOST Research Interests



Service binding: new requirements

Before	Now





Service binding: overall process

- Access to the set of available services (**services space**)

- **Match** the desired service description with each one of the available services description

- Assign the matching degree and **rank** the result set

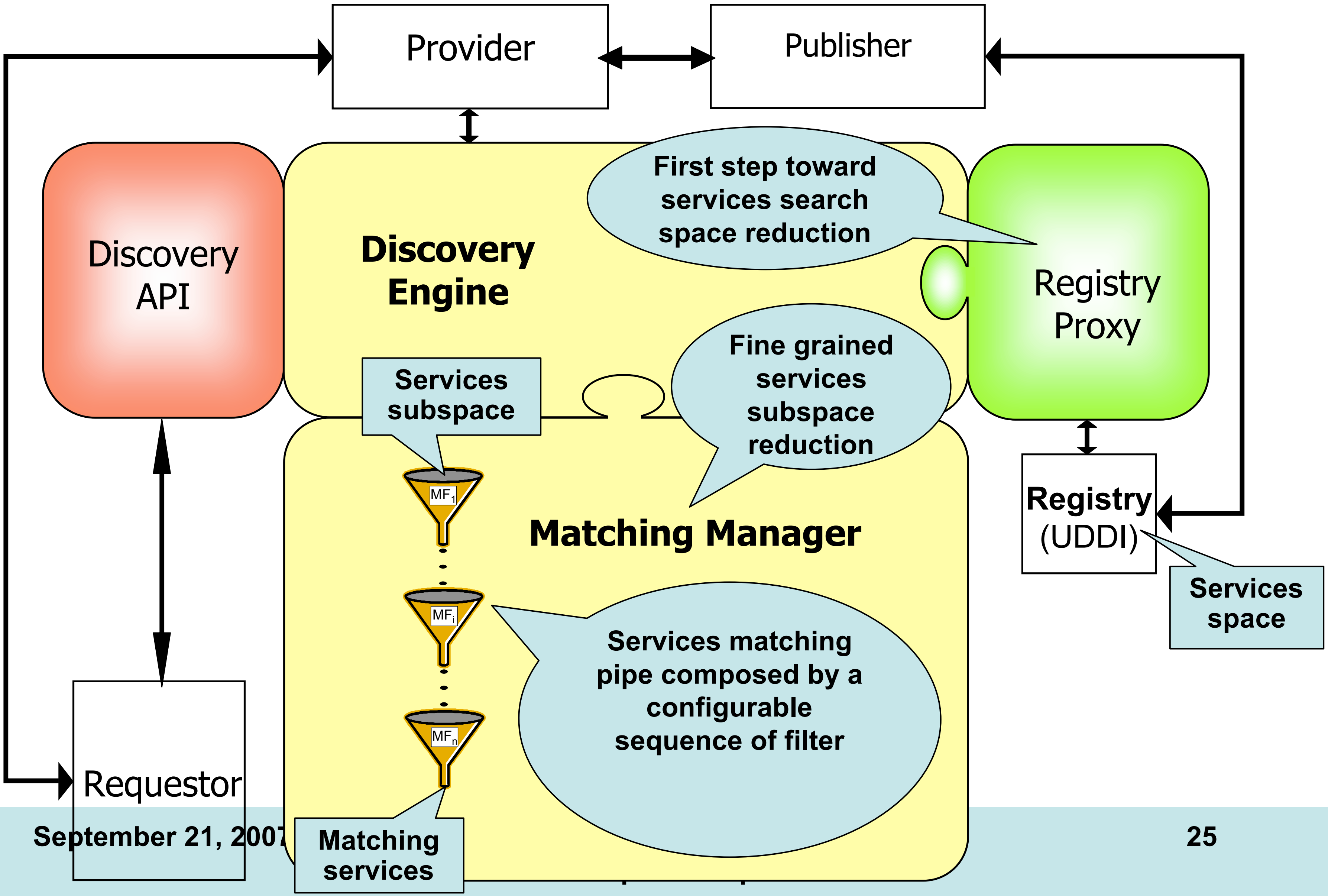
- **Choose** the service that better fits the request

- **Bind** the service for invoking its functionalities

Matching Problem

- The problem of determining whether a given service description conforms to another service description
- Provider describes its service with a service description *t* that we call **target** description
- Requestor formulates its request to a matchmaker following two basic approaches
  - ➢ Service description as query
  - ➢ Query language statements as query
    - ✓ We call that query **template** description *T*, whatever form it has
- It is essential to distinguish what we have to match with respect to
  - Our problem is to match a template against a set of targets
- … when a target match a template?
  - We assume that a target match a template when these descriptions are "compatible"

Discovery Process



Semantics for Web Processes

- Data/Information Semantics
  - **What:** Formal definition of data in input and output messages of a web service
  - **Why:** for discovery and interoperability
  - **How:** by annotating *input/output data* of web services using ontologies

- Functional/Operational Semantics
  - Formally representing capabilities of web service
  - for discovery and composition of Web Services
  - by annotating *operations* of Web Services as well as provide *preconditions* and *effects*

- Execution Semantics
  - Formally representing the execution or flow of a services in a process or operations in a service
  - for analysis (verification), validation (simulation) and execution (exception handling) of the process models
  - using *State Machines*, *Petri nets*, *activity diagrams* etc.

- QoS Semantics
  - Formally describing operational metrics of a web service/process
  - To select the most suitable service to carry out an activity in a process
  - using *QoS model* for web services

State of the art

For Functional Requirements

- OWL-S
- METEOR-S
- WSDL-S
- WSMO

QoS

Ontology	Goals	Upper QoS concepts
DAML QoS	DAML-S complement	QoS profile QoS property And QoS metrics
WS QoS	Web service discovery	QoS vocabulary
QoSOnt	Service based system	Base and unit, attribute and usage domain
QoS ontology	Agent based system	Upper, Middle and lower

onQoS

The Upper Ontology

- QoS Parameter
- QoS Metric
- Measurement Process
- Scale
- Scale Value
- QoS Metric Function

The Middle Ontology

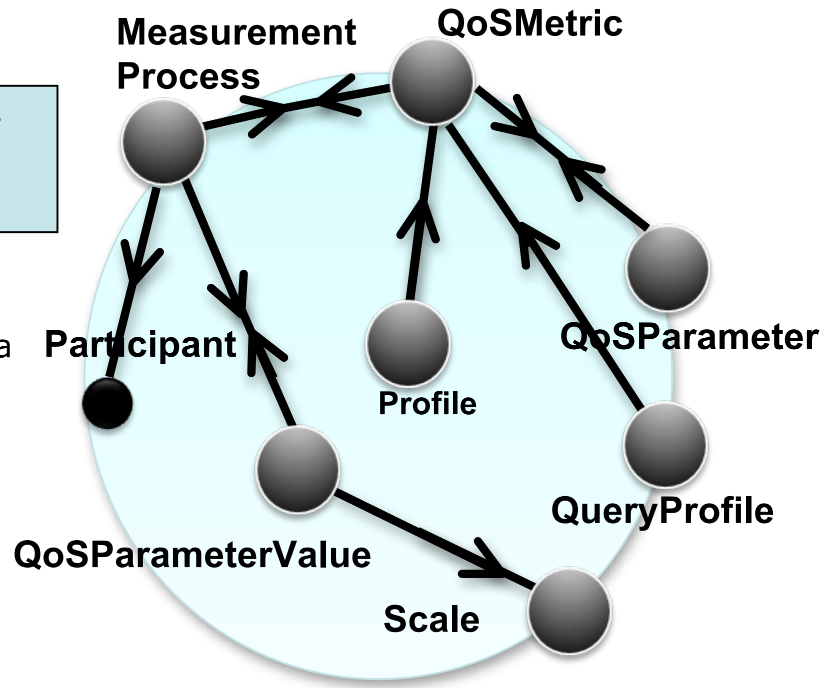
- The QoS Metric Middle Ontology
- The QoS Metric Function Middle Ontology
- The QoS Scale Vocsbulary Ontology

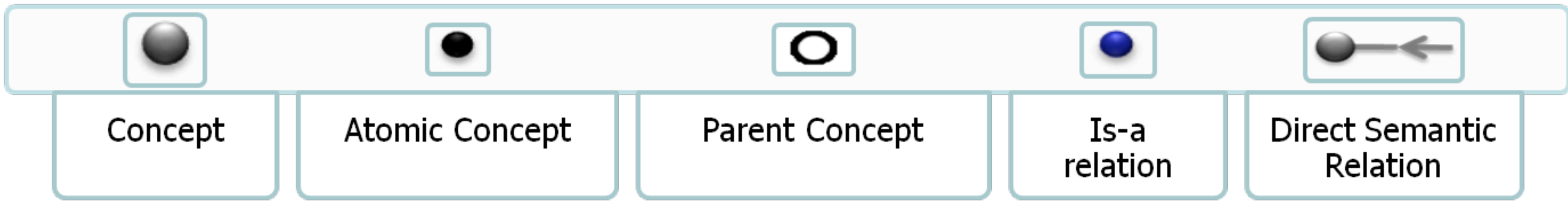
The Lower Ontology

- QoS Network Ontology
- Automotive ontology

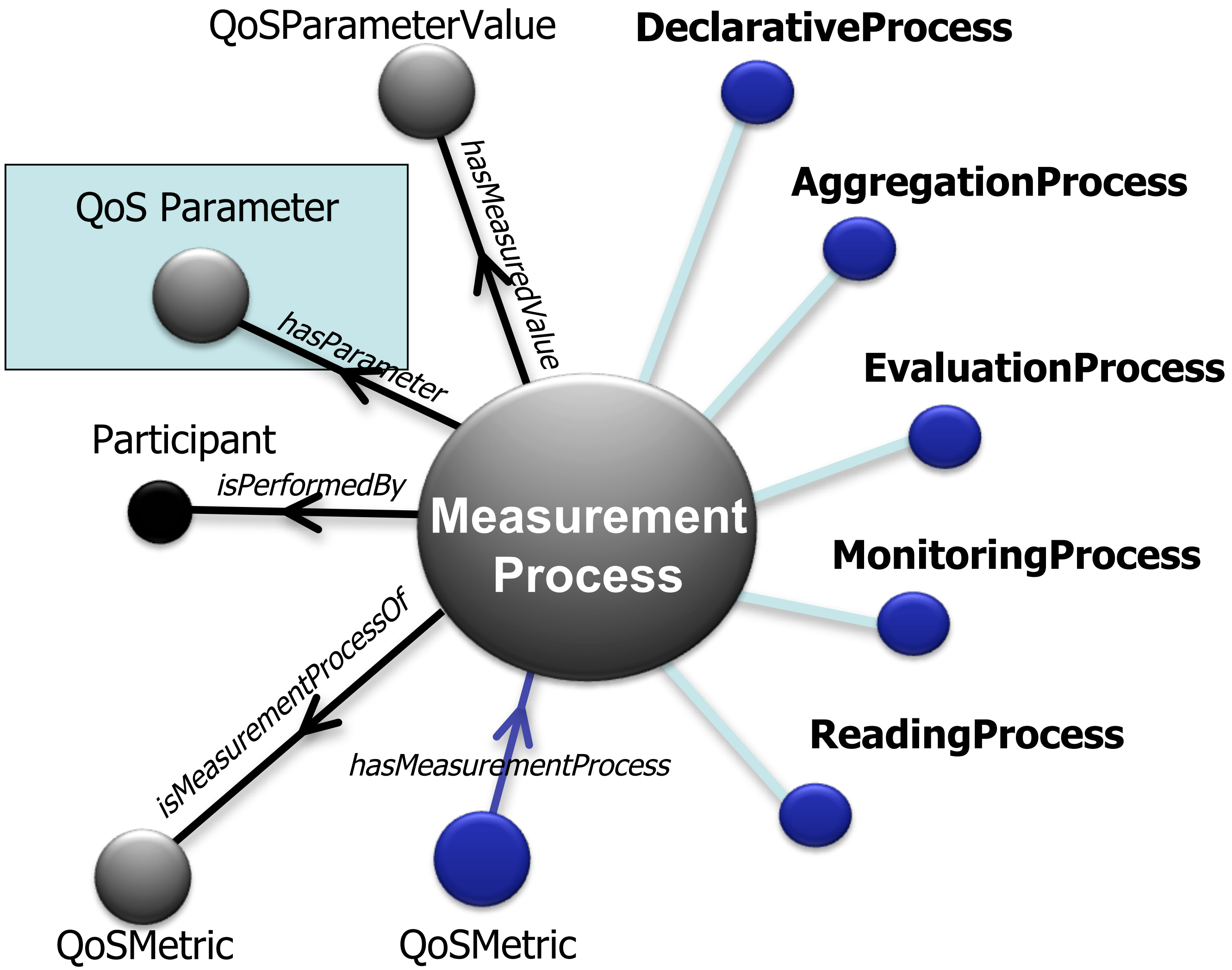
onQoS – a global view

- **QoS parameter** is a measurable QoS characteristic or feature
- **QoS Metric** is a type of measurement which relates to a QoS parameter
- **Measurement Process** is the process by which numbers or symbols are assigned to QoS parameters according to clearly defined rules
- **Scale** specifies the nature of the relationship between a set of values
- **QoSParameterValue** is a number or symbol that identifies a category in which the QoS parameters can be placed basing on a particular attribute
- **Participant** identifies the resource that performs the measurement process
- **Profile** describes a QoS policy through the definition of one or more QoS metrics
- **Query Profile** is a particular Profile that presents a unique QoS metric relating to the overall required QoS.



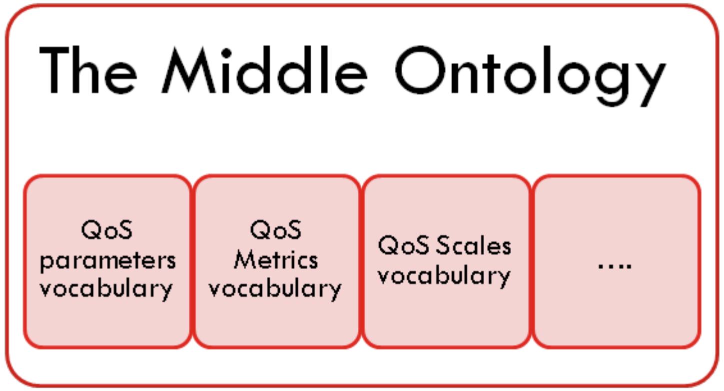


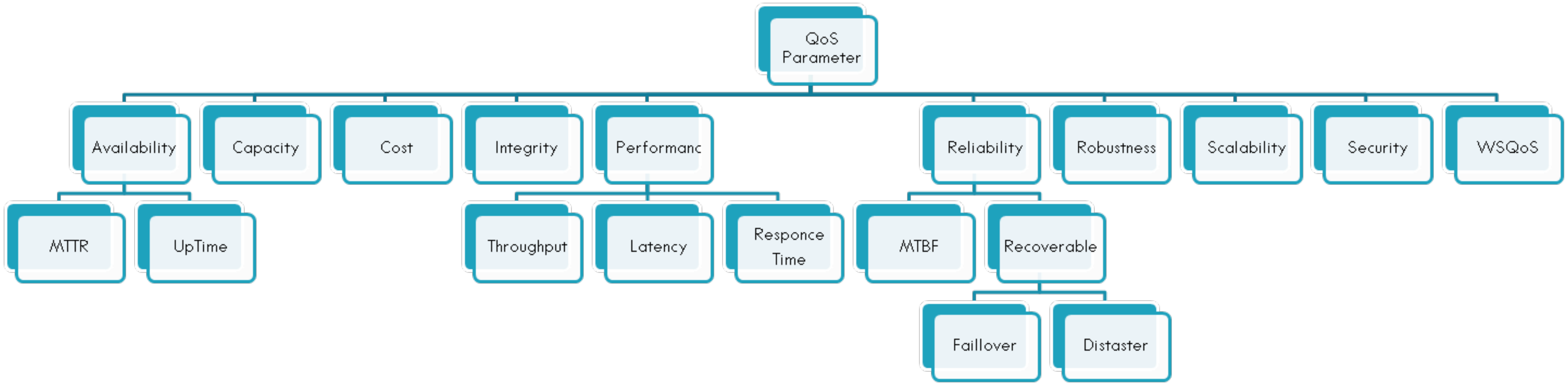
onQoS: MeasurementProcess



onQoS: Middle ontology

- QoS Parameters Vocabulary

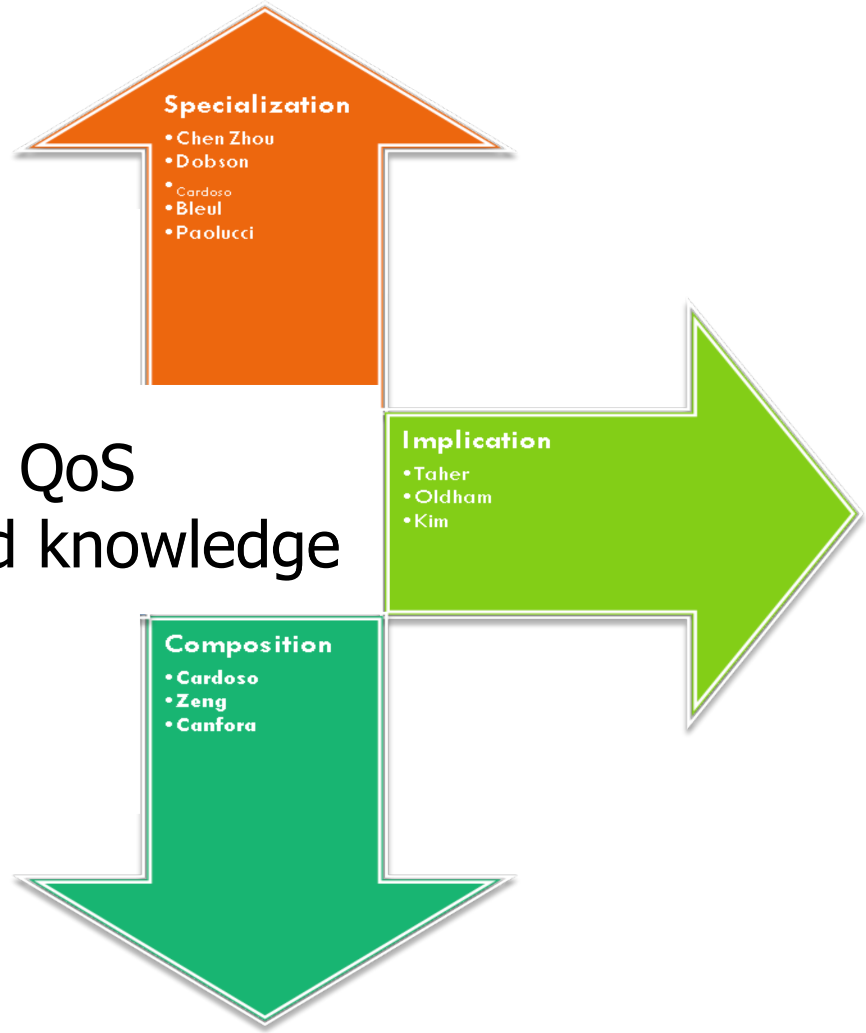




Directions in the Semantic Matching

We identified the Specialization, the Implication and the Composition direction to exploit the QoS knowledge in the matching process

Exploiting QoS formalized knowledge



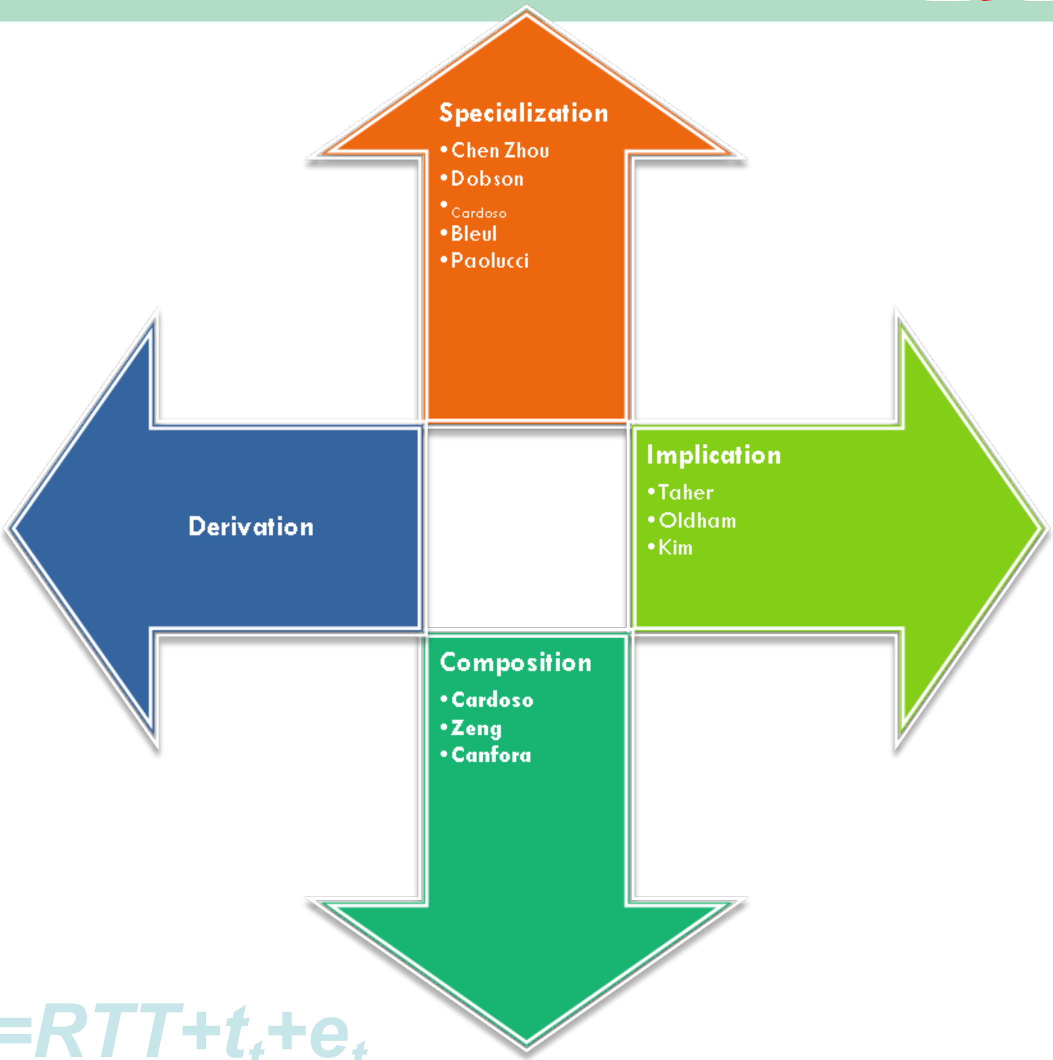
Directions in the Semantic Matching

We identified the Specialization, the Implication and the Composition direction to exploit the QoS knowledge in the matching process

And we introduced a fourth one ….

RTT<=24.9
e_t <= 0.5
t_t <= 7

L<=24.9



$L=RTT+t_t+e_t$

(D3, D1) | (D3, D2)
(D3, D5) | (D4, D1)
(D4, D2) | (D7, D1)
(D7, D2) | (D7, D3)
(D7, D4) | (D7,D5)
(D8,D1) | (D8,D2)
(D8, D3) | (D8,D5)
(D8,D6)

	QoS requirements		QoS requirements
D1	**Authentication** **Authorization** Cost <= 100€ EncStand: RSA, PKI, OpenPGP, Triple-DES **ExecutionTime** <= 0.5 ms FaulRate <= 50% Jitter <= 0.3 ms NetThroughput >= 200 kbps **RTT** <= 14ms Scalability >= 78% **TransmissionTime** <= 7 ms UpTime >= 90%	D3	Cost <= 140 € EncStand: RSA, PKI Jitter <= 0.3 ms **NetLatency** <= 24.9 ms **Privacy** UpTime >= 65%
		D4	Authentication Authorization NetLatency <= 26 ms ThrLatRatio >=3.2 Mbps/s
D2	Cost <= 121 € EncStand: RSA, PKI, OpenPGP ExecutionTime <= 0.6 ms FaulRate <= 50% Jitter <= 1.5 ms Privacy RTT <= 17ms Scalability >= 43% ThrLatRatio >= 3.5Mbps/s UpTime >= 86%	D5	Authentication Authorization EncStand: RSA, PKI, OpenPGP ExecutionTime <= 0.8 ms Jitter <= 2.6 ms NetThroughput >= 10 kbps RTT <= 5 ms TransmissionTime <= 6 ms UpTime >= 65%
		D6	NetLatency <= 22 ms
		D7	Authentication Authorization
		D8	RTT <= 25ms

A new problem:
how can we specify QoS requirements

- Today a standard QoS query language has not yet defined
- How Can We Specify QoS requirements?
  - Through service descriptions
    - Template are not sufficiently expressive to capture user desiderata
    - Service ranking is often subjective and needs to specify user-centric utility functions

A proposal

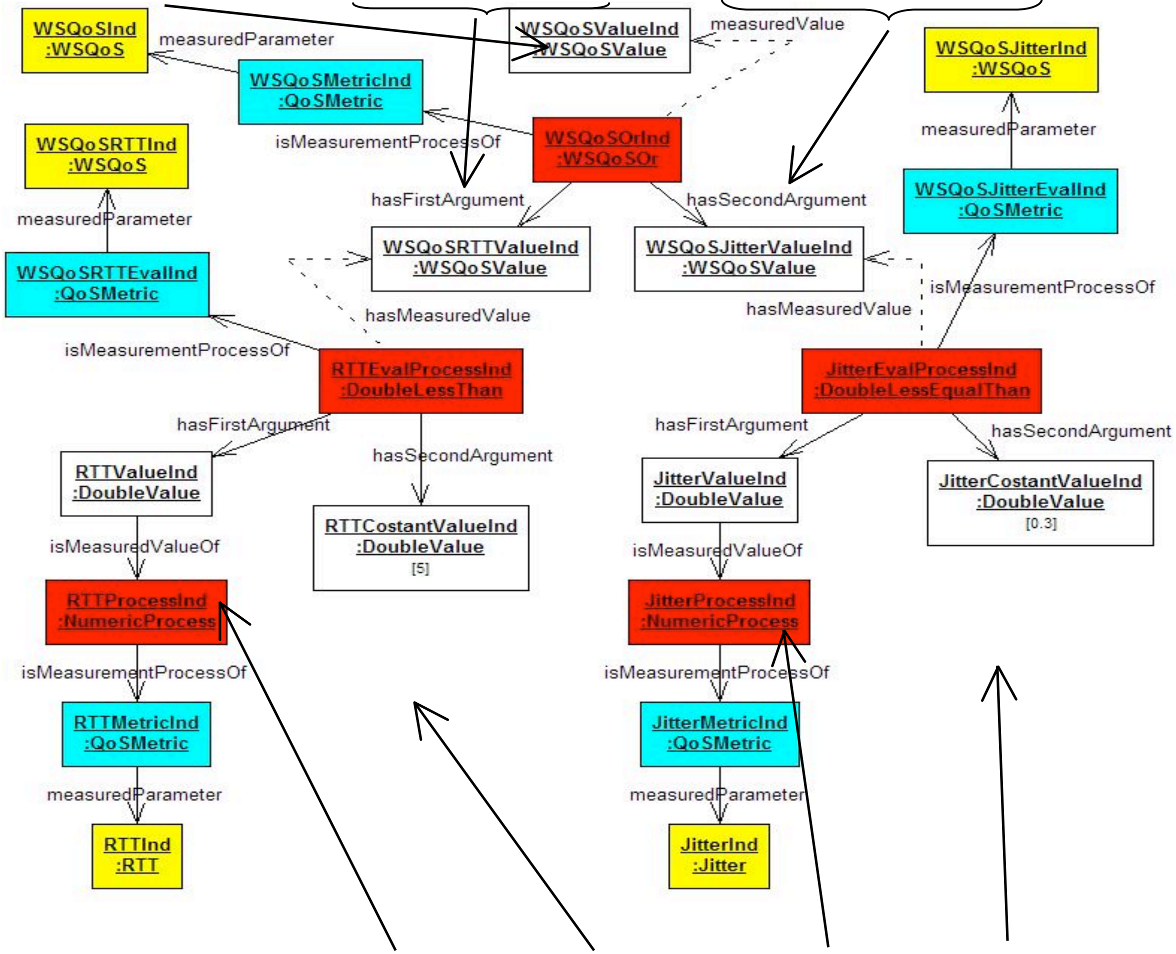
- onQoS-QL
  - To define effectively complex and expressive queries on QoS constraints
  - A way to formalize requestor real subjective QoS expectations and intentions so that the QoS discovery engine will be able to select automatically the “right” service reasoning not only on the QoS shared knowledge but also ranking the services according to the requestor criteria

onQoS-QL

- It is based on onQoS
  - The onQoS-QL elements are interpreted utilizing onQoS semantics and its own domain specializations
- WSQoSMetric is the main building block
  - It measures the degree of compatibility between two QoS descriptions

A Query in onQoS-QL

?WSQoSValue = ?RTTValue < 5 || ?JitterValue ≤ 0.3



?WSQoSValue = ?RTTValue < 5 || ?JitterValue ≤ 0.3

Constants

RTTConstantValue = 5

JitterConstantValue = 0.3

Elementary Metrics

RTTMetric = <RTT, RTTProcess, DoubleScale, ?RTTValue>

JitterMetric = <Jitter, JitterProcess, DoubleScale, ?JitterValue>

WSQoS Evaluation Metrics:

WSQoSRTTEval = <WSQoSRTT, RTTEvalProcess, WSQoSScale, ?WSQoSRTTValue>

WSQoSJitterEval = <WSQoSJitter, JitterEvalProcess, WSQoSScale , ?WSQoSJitterValue>

WSQoS Aggregation Metric:

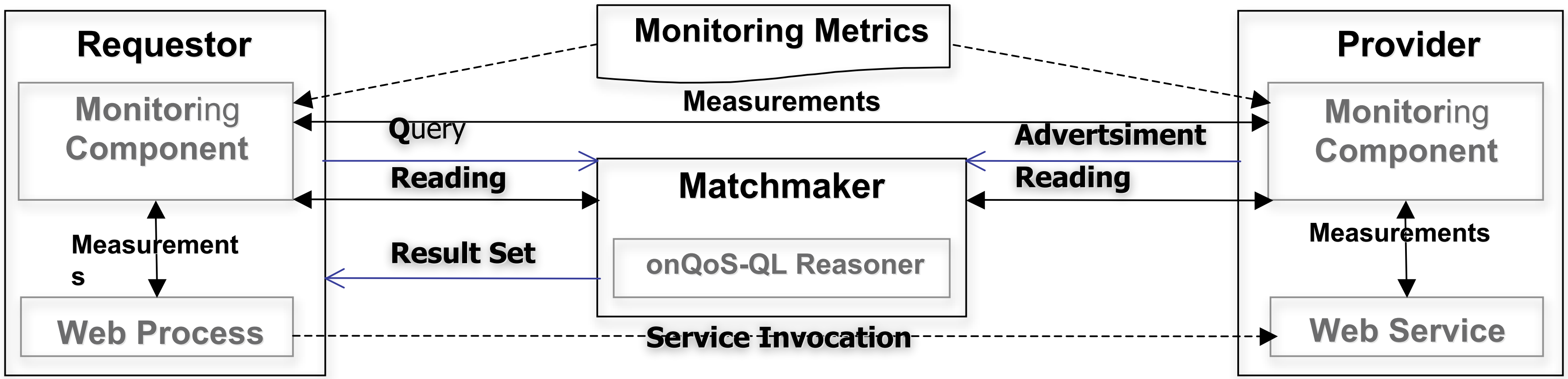
WSQoSMetric = <WSQoS, WSQoSOr, WSQoSScale, ?WSQoSValue>

Evaluating WSQoSmetric:

?WSQoSValue = WSQoSOr(
RTTEvalProcess(RTTProcess(),
RTTConstantValue),
JitterEvalProcess(JitterProcess(),
JitterConstantValue))

A Query in on-QoS-QL

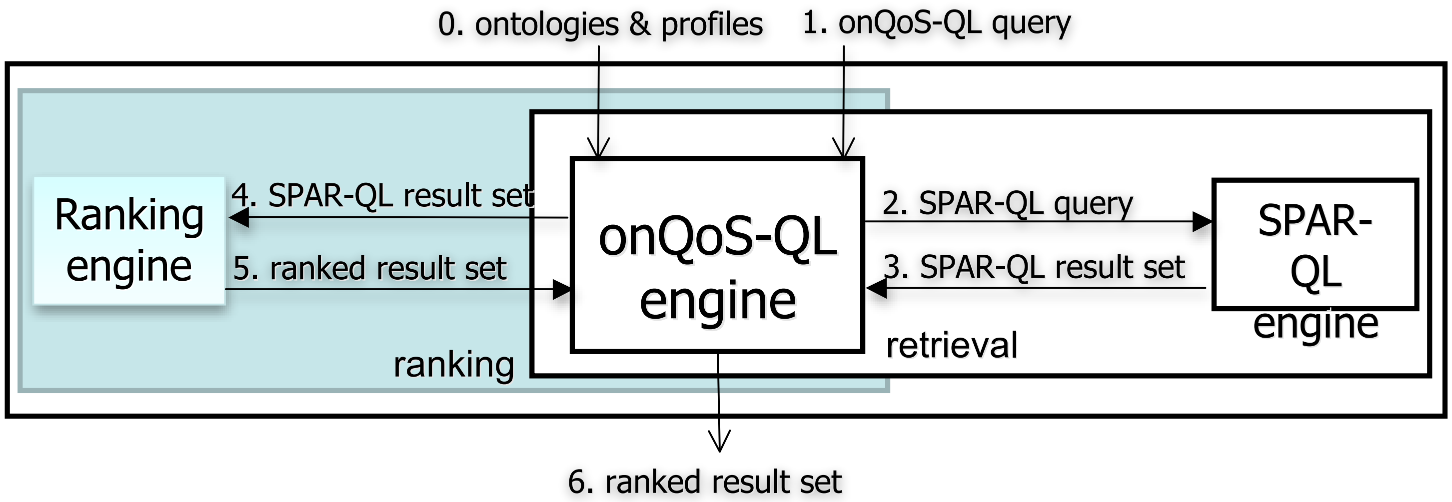
?WSQoSValue = ?RTTValue < 5 || ?JitterValue ≤ 0.3



Main predicates and aggregation functions

Vocabulary Term	Measurement Scale of Arguments	Retrieval Semantics	Ranking Semantics
Equal(x_i, x_j)	NominalScale $\{x_i\}_{i=1...N}$	$i = j$	1
NotEqual(x_i, x_j)		$i \neq j$	1
BetterEqualThan(x_i, x_j)	OrdinalScale $\{x_i \mid x_i < x_j \Leftrightarrow i < j\}_{i=1...N}$	$i \in \{j,\ldots, N\}$	$\frac{1}{N}(1+i-j)$
LessEqualThan(x_i, x_j)		$i \in \{1,\ldots, j\}$	$\frac{1}{N}(1+j-i)$
DoubleLessThan(x, y)	DoubleRatioScale $[X_{inf}, X_{sup}]$	$x < y$	$\frac{2}{1+e^{-\frac{y-x}{k}}}-1$
DoubleGreaterThan(x, y)		$x > y$	$\frac{2}{1+e^{-\frac{x-y}{k}}}-1$
WSQoSAnd(x, y)	WSQoSScale $\{p \equiv (x,w) \mid x \in [0,1] \wedge w \in [0, X_{sup}]\}$	$x \wedge y$	$min(x, y)$
WSQoSOr(x, y)		$x \vee y$	$max(x, y)$
WeightedMean$(p_i)_{i=1,\ldots,N}$		$\exists p_i$, $i = 1,\ldots, N$	$\sum_{i=1}^{N} w_i x_i \Big/ \sum_{i=1}^{N} w_i$

The ranking engine



computes a rank for each retrieved service according to the defined semantics.

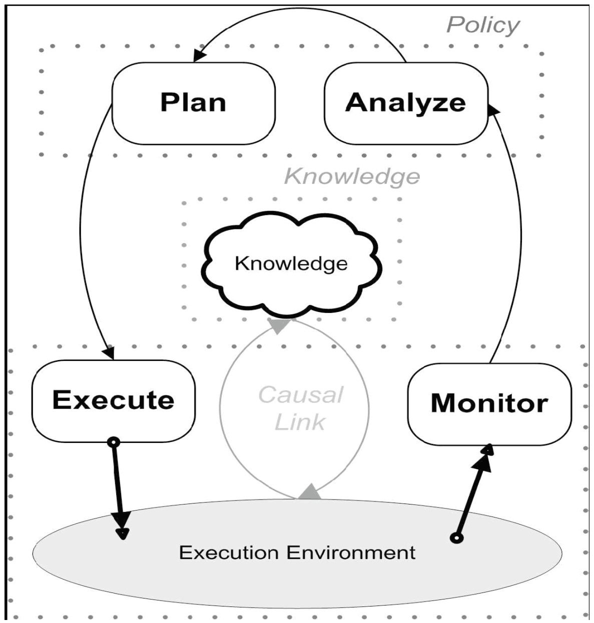
$$?WSQoSValue = max\left(\frac{2}{1+e^{-\frac{k_1 - ?RTTValue}{k}}} - 1, \frac{2}{1+e^{-\frac{k_2 - ?JitterValue}{k}}} - 1 \right)$$

Towards Self-evolution

Autonomic Computing

- The growing complexity of nowadays software platforms requires a lot of efforts for the system manager in order to maintain the systems in operation
- The autonomic computing is aimed to develop software systems that are able to manage themselves autonomously
- Autonomic systems must be able to provide four main functionalities: self-configuration, self-optimization, self-healing and self-protection
- These functionalities are identified as self* properties

Manager Control Cycle



1. ***Monitor***. **The manager retrieves data from the managed resources, by a push or pull policy**

2. ***Analyze***. **The collected data are analyzed in order to be contextualized to give them the right interpretation**

3. ***Plan***. **The data are processed for deciding whether there is the need for an intervention and which kind of action to perform**

4. ***Execute***. **The selected action is performed. This step is directly related to the interaction with the managed resource, using the effecting interface for altering the configuration of the autonomic element**

Autonomic Web Processes

- Natural evolution of autonomic computing from individual information technology resources to the business processes

- Take advantage of the autonomic computing sot that composed web services can benefit of self* properties

Proposal

- Increasing automatic management in:
  - Composition
  - Supervision
  - Evolution
- Using:
  - Autonomic self-aware manager
  - User-defined policies
  - Knowledge base and semantic descriptions

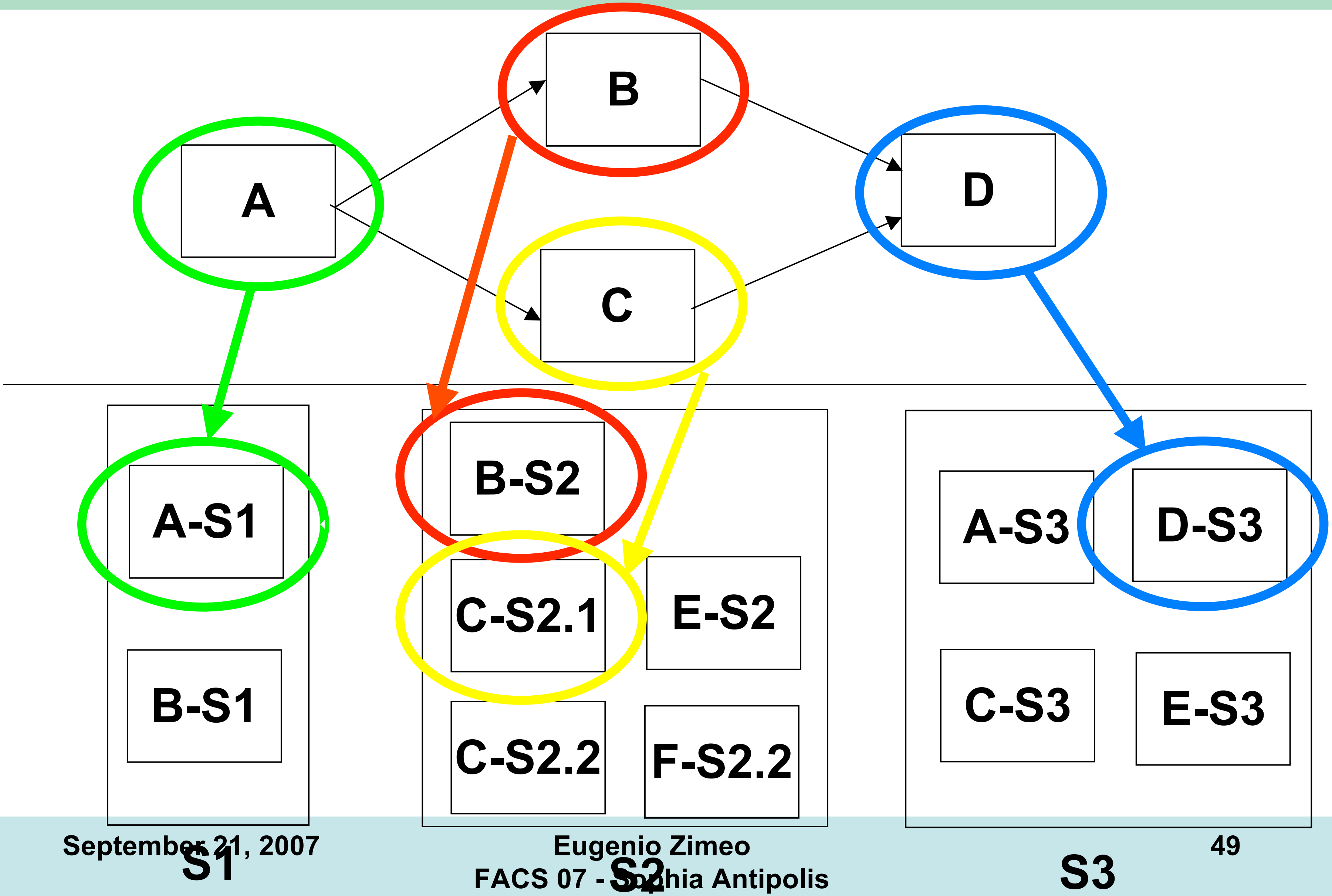
Definition

- An **Autonomic Workflow** may be defined as:

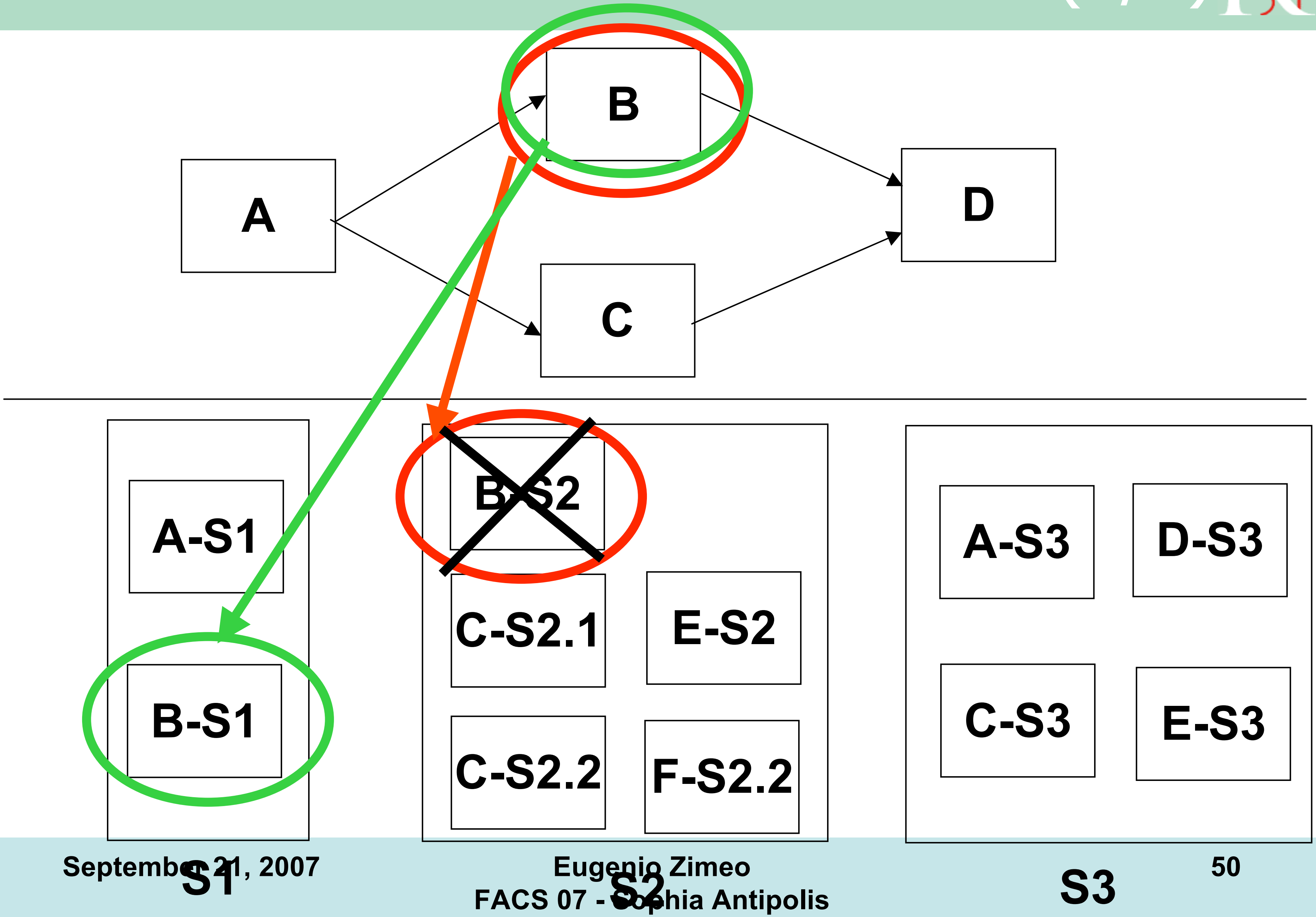
  a Workflow extended to contain semantic information about its objective and all the related data and constraints that may be useful for its definition, execution and evolution

Centralized Self-Evolution

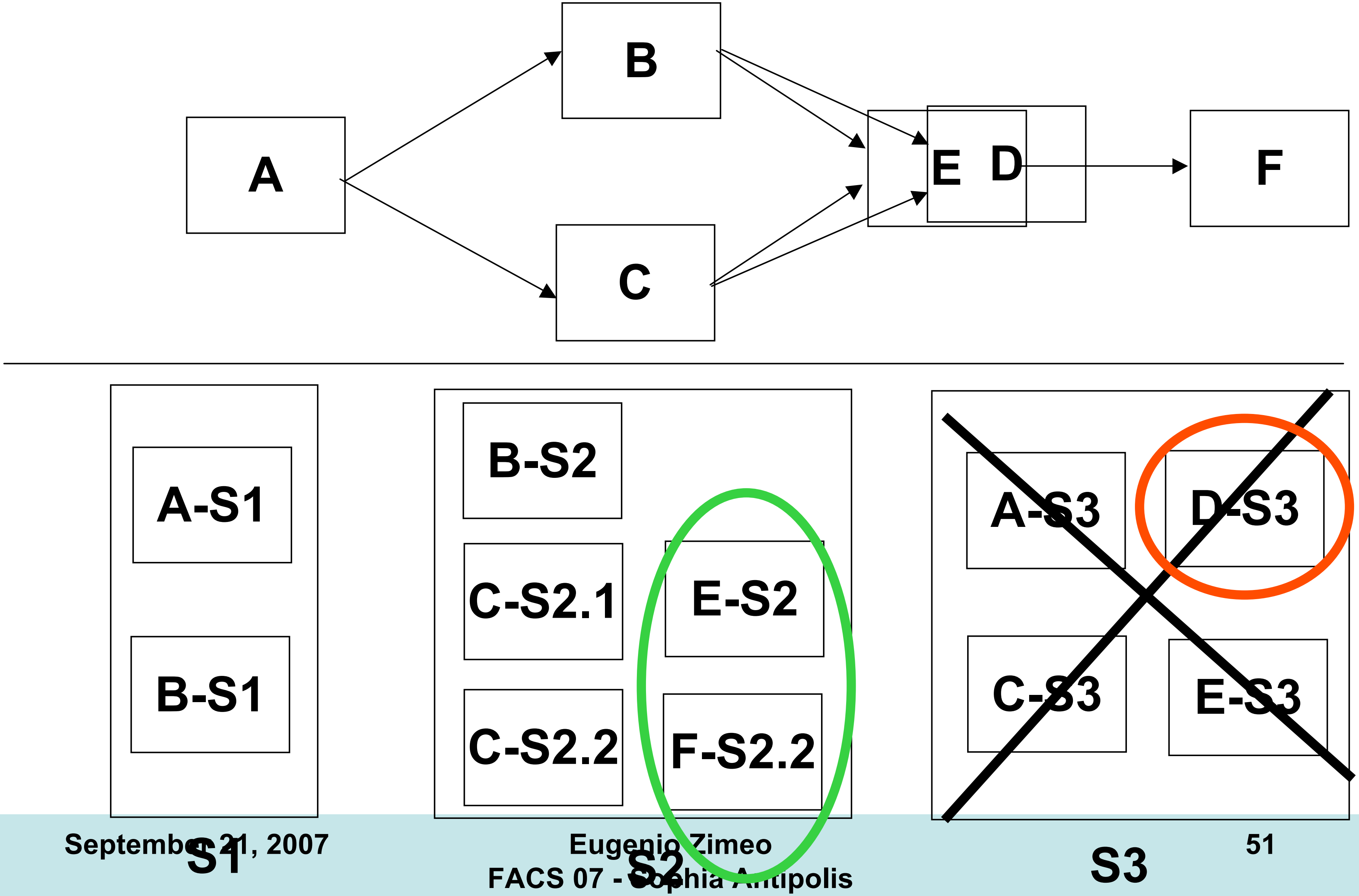
Autonomic Workflow Execution (1/3)



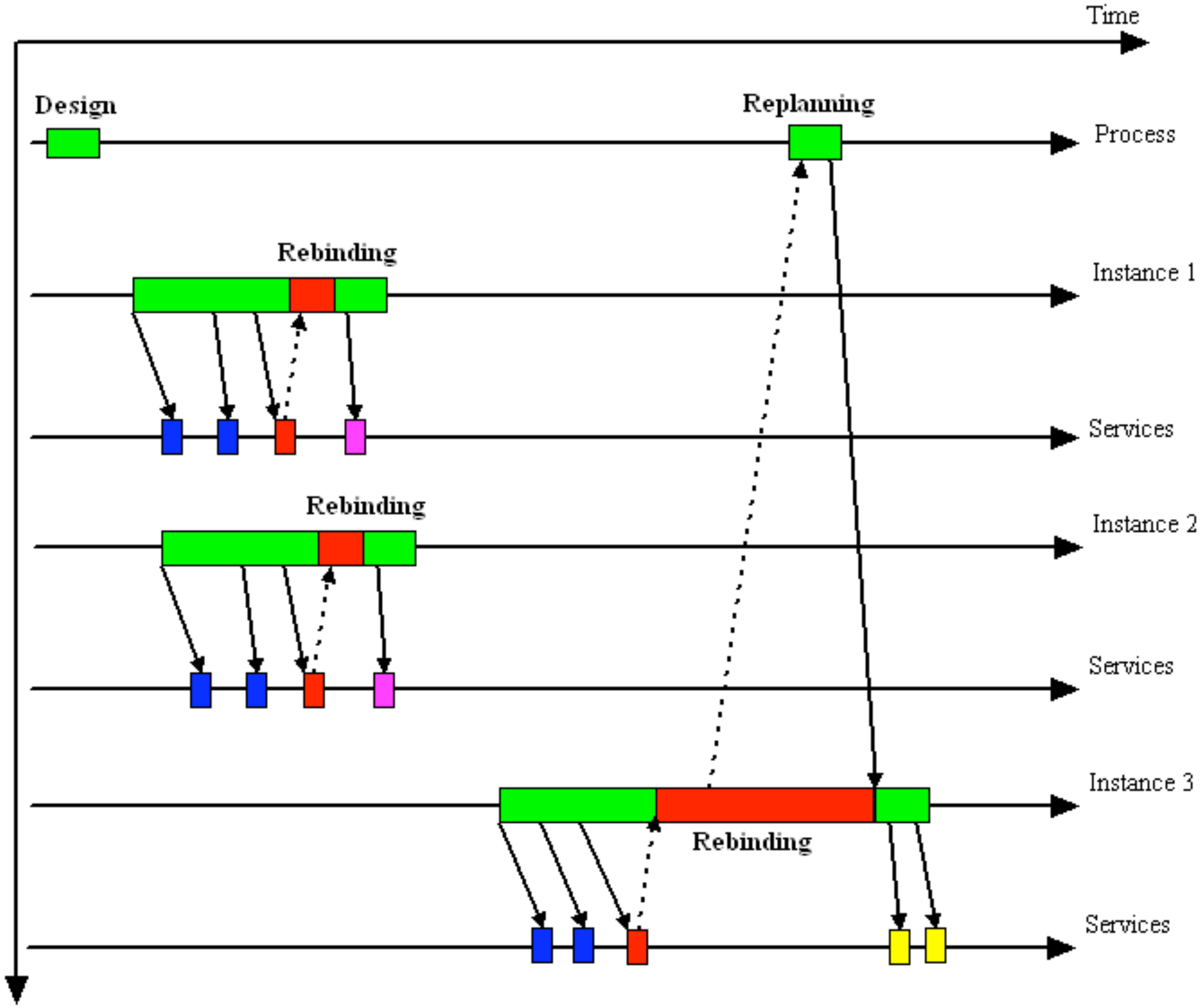
Autonomic Workflow Execution (2/3)



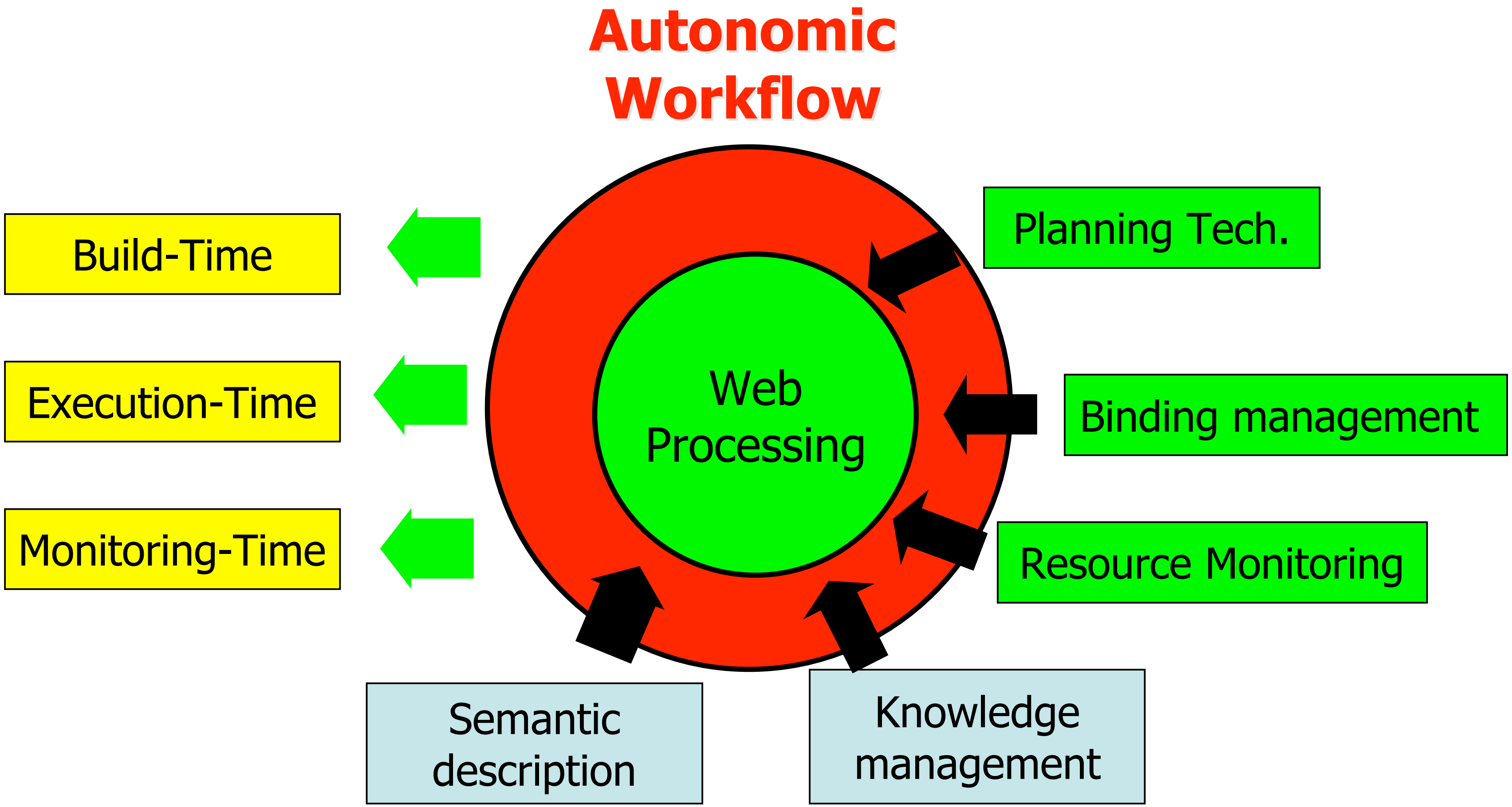
Autonomic Workflow Execution (3/3)



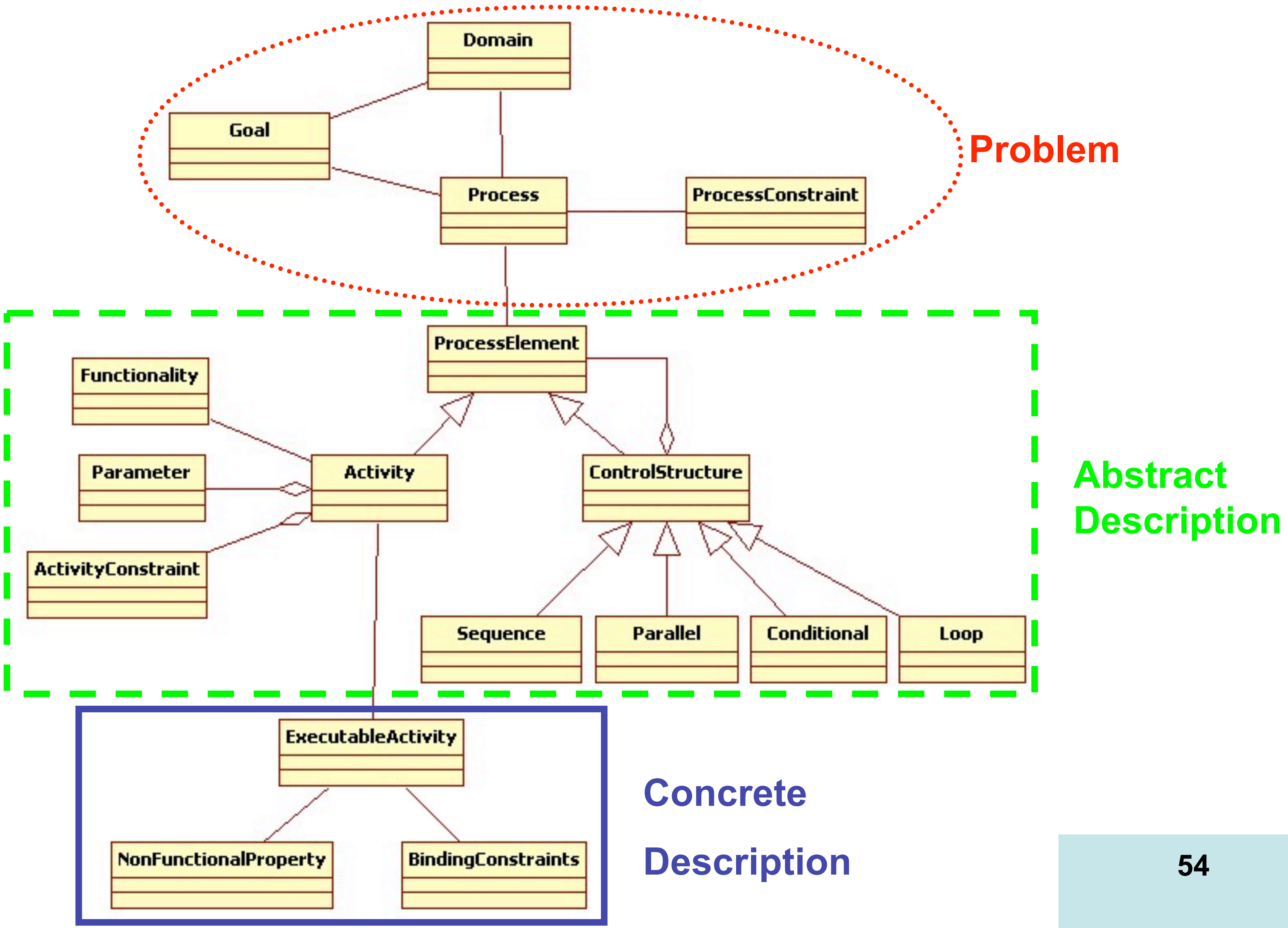
Execution


Time
Design
Replanning
Process
Rebinding
Instance 1
Services
Rebinding
Instance 2
Services
Instance 3
Rebinding
Services

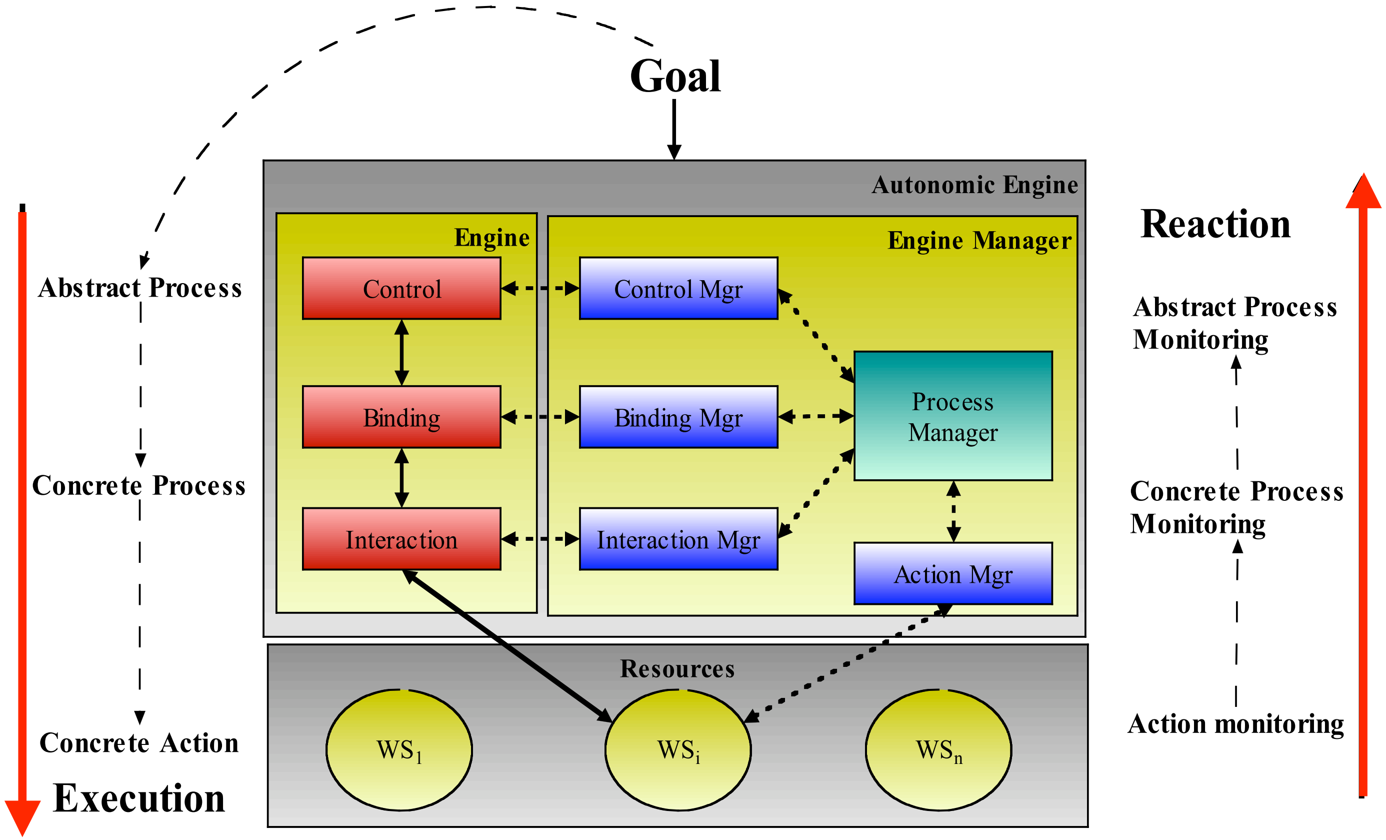
Features


Autonomic
Workflow
Build-Time
Execution-Time
Monitoring-Time
Web
Processing
Planning Tech.
Binding management
Resource Monitoring
Semantic
description
Knowledge
management

Process Entities and Relationships



Operative flows



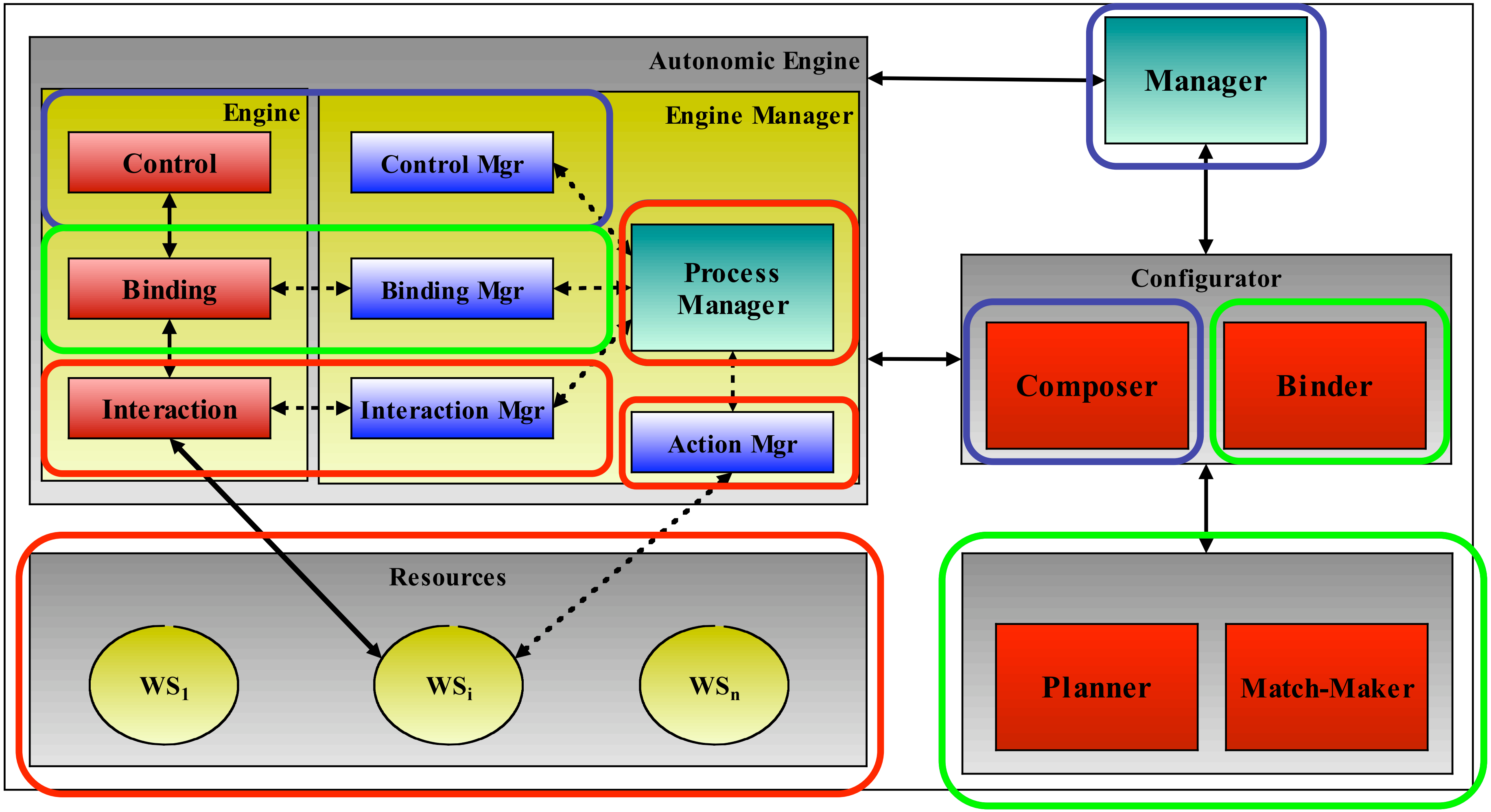
Execution Flow

- Starts with goal submission
- Details are added going down through the execution steps until concrete service are invoked

Reaction Flow

- Detects events and other information about the state of managed elements
- Data collected are analyzed with management policies
- If the event can not be handled by the manager, the Process Manager is involved in

Components



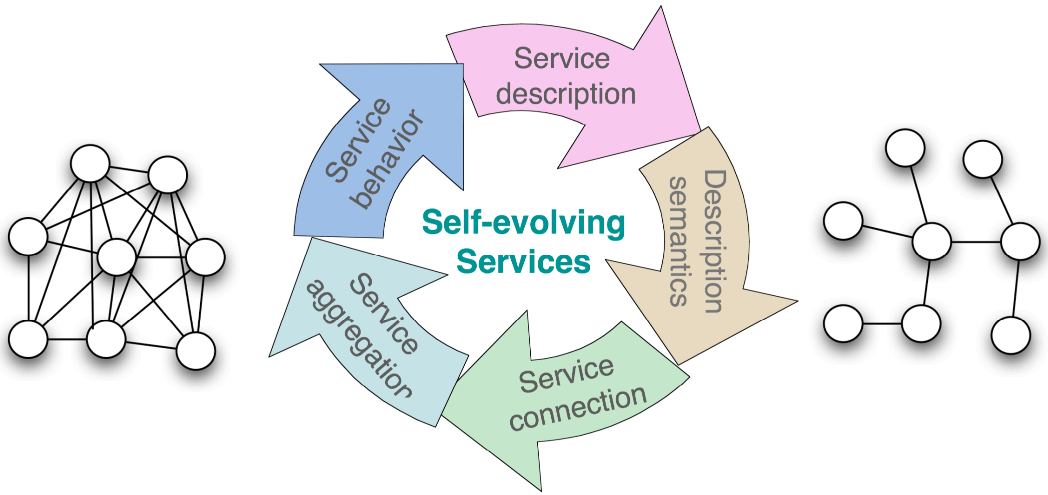
Autonomic Engine
Engine
Engine Manager
Control
Control Mgr
Binding
Binding Mgr
Process Manager
Interaction
Interaction Mgr
Action Mgr
Manager
Configurator
Composer
Binder
Resources
WS1
WSi
WSn
Planner
Match-Maker

Decentralized Self-Evolution

research centre on software technology

What in the future ?

- Extension of the current vision of SOA to support self-evolving, service oriented systems where
  - services are discovered and composed using a collaborative approach, and
  - service descriptions are automatically extracted from source code and monitoring data



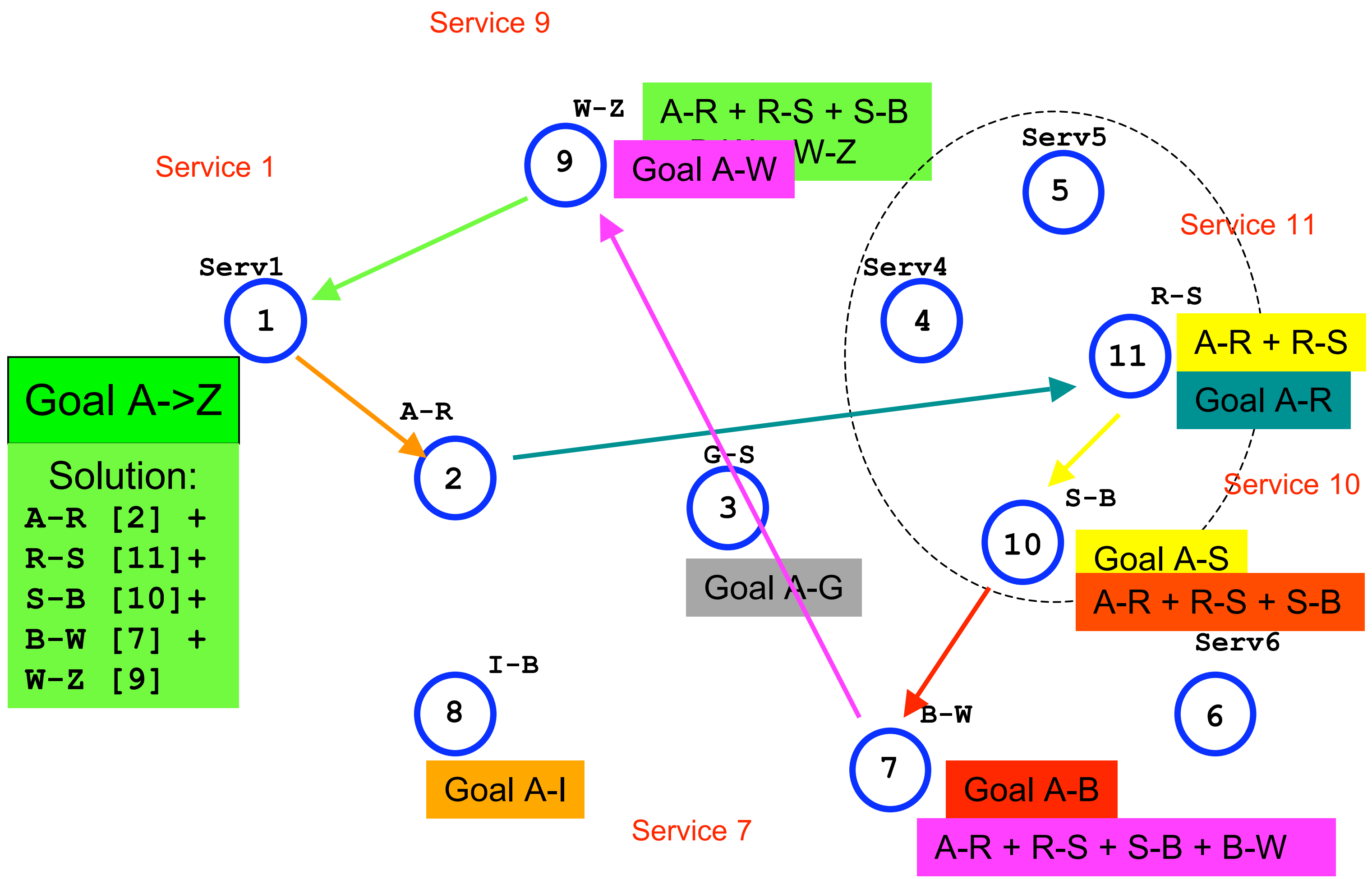
Autonomic Adaptation

Self-composing services

- Why? SOA is becoming a pervasive paradigm for heterogeneous distributed applications
  - Centralized and supervised approaches for discovery and composition represent bottlenecks for scalability (for both performance and functionalities)
  - Applications are limited to only coarse grained distributed interactions
    - Lack of flexibility, heterogeneous composition and cooperation

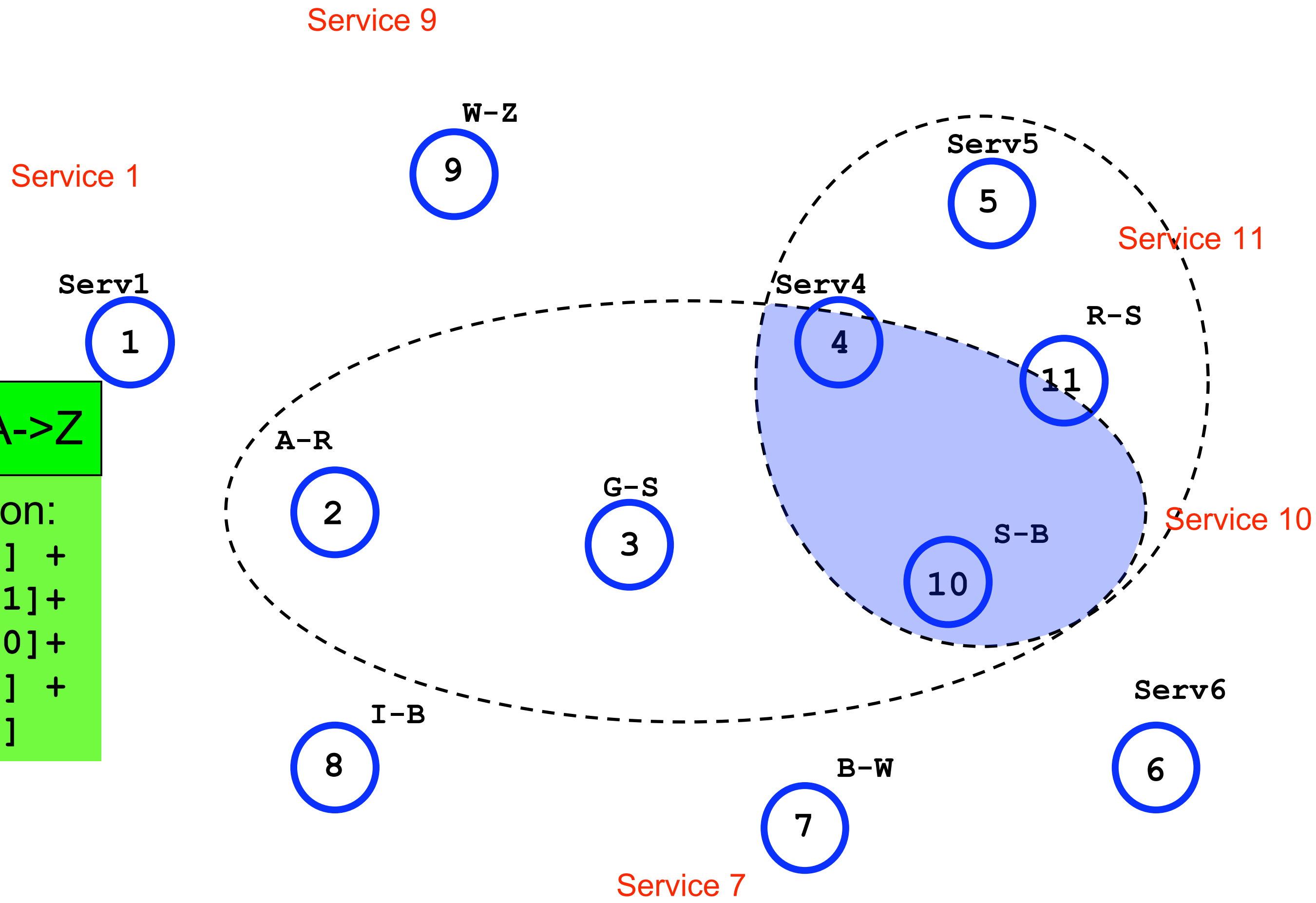
- Objective: extending SOA towards *a network of cooperative services*
  - Fully distributed discovery and composition
  - Composition and execution without orchestration
    - Cooperative, peer-to-peer approach
  - Dynamic P2P hybrid topology with semantic multiple overlays

From a goal ….



Service 9
Service 1
W-Z
A-R + R-S + S-B
W-Z
9
Goal A-W
Serv5
5
Service 11
Serv1
1
Serv4
4
R-S
11
A-R + R-S
Goal A-R
Goal A->Z
Solution:
A-R [2] +
R-S [11]+
S-B [10]+
B-W [7] +
W-Z [9]
A-R
2
G-S
3
Goal A-G
S-B
Service 10
10
Goal A-S
A-R + R-S + S-B
Serv6
I-B
8
6
B-W
7
Goal A-I
Goal A-B
Service 7
A-R + R-S + S-B + B-W

… to network re-factoring



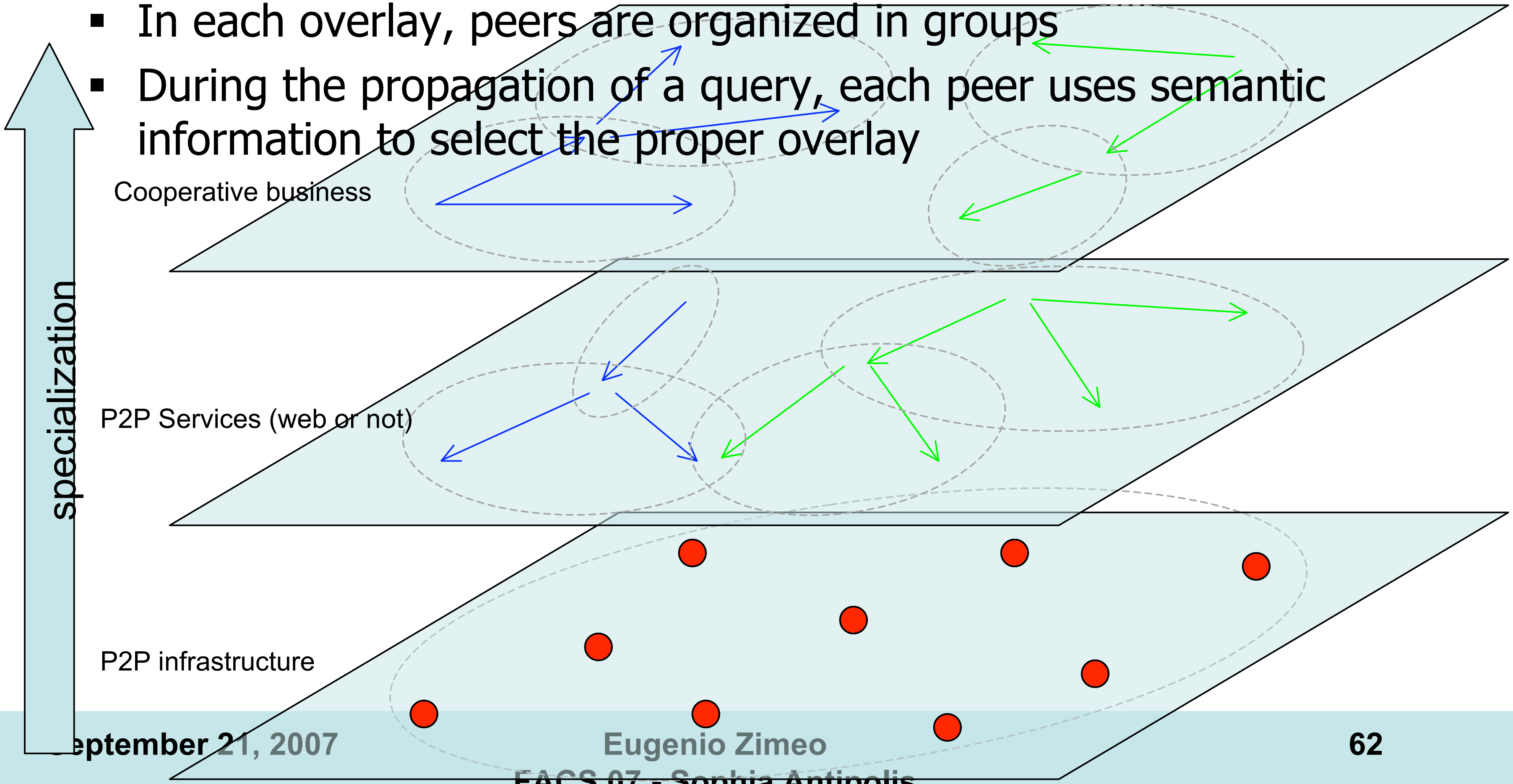
Goal A->Z

Solution:

```
A-R [2] +
R-S [11]+
S-B [10]+
B-W [7] +
W-Z [9]
```

... to multiple overlay networks

- Many overlay networks at different abstraction levels, each one able to solve a kind of problem
  - In each overlay, peers are organized in groups
  - During the propagation of a query, each peer uses semantic information to select the proper overlay



Where do services and components meet ?

Services and components

- Services and components should be used together for large scale applications
  - Services tackle the problems of the open world
  - Components support reusable software in closed environments
- Research activity on semantic service binding could be applied to other kinds of components
- Verification is useful in composite web services at design-, deployment- and run-time
- At runt-time verification needs sophisticated monitoring of functional and QoS properties

Thanks

zimeo@unisannio.it

research centre on software technology