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
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 Registry
  - Lookup
  - Publication
  - Service Specification
  - Bind & Invoke

- Service Consumer
- Service Provider
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

- UDDI
- SOAP
- WSDL
- RMI
- CORBA
- Jini

Enterprise Java Beans
Microsoft DCOM
Open Software Foundation DCE (Distributed Computing Environment)
Sun ONC/RPC (Open Network Computing)

IP, UDP, TCP
Web Services Technology

A very brief overview
Web Service Stack

The Conceptual Web Services Stack

WS-BPEL, XPDL, ...

Service Flow

UDDI

Service Discovery

UDDI

Service Publication

WSDL

Service Description

SOAP

XML-Based Messaging

HTTP, FTP, email, MQ, IIOP, etc.

Network

Our focus

Security

Management

Quality Of Service
Service description: WSDL

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

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

- Allows advertisement of service descriptions, dynamic discovery and binding of compatible services
  - Used in conjunction with UDDI registry
• 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 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 (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**
Web Processes: composition

Web Process Design

Web Processes

Web services

Jorge Cardoso

September 21, 2007
Eugenio Zimeo
FACS 07 - Sophia Antipolis
Web Processes: deployment

Web Processes

Organization A

$t_1$
Setup

$t_2$
Prepare Sample

$t_3$
Prepare Clones and Sequence

$t_4$
Assembly

$t_5$
Test Quality

$t_6$
Get Sequences

$t_7$
Sequence Processing

$t_8$
Process Report

Organization B

Organization C

Jorge Cardoso

Setup

Prepare Sample

Prepare Clones and Sequence

Assembly

Test Quality

Get Sequences

Sequence Processing

Process Report

September 21, 2007

Eugenio Zimeo

FACS 07 - Sophia Antipolis
Semantic Web Processes
RCOST Research Interests

Semantics

Web Processes

Web Process Composition

Web Process QoS

Web Services

Web Service Annotation

Web Service Discovery

Web Service QoS
Service binding: new requirements

Before

Now

Tasks

Workflow

Web Services

Web Process

QoS

September 21, 2007
Eugenio Zimeo
FACS 07 - Sophia Antipolis
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

Discovery API

Provider

Publish

Requestor

Discovery Engine

First step toward services search space reduction

Services subspace

Fine grained services subspace reduction

Matching Manager

Services matching pipe composed by a configurable sequence of filter

Services subspace

Registry Proxy

Registry (UDDI)

Services space
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

<table>
<thead>
<tr>
<th>Ontology</th>
<th>Goals</th>
<th>Upper QoS concepts</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAML QoS</td>
<td>DAML-S complement</td>
<td>QoS profile</td>
</tr>
<tr>
<td>WS QoS</td>
<td>Web service discovery</td>
<td>QoS property</td>
</tr>
<tr>
<td>QoS Ont</td>
<td>Service based system</td>
<td>And QoS metrics</td>
</tr>
<tr>
<td>QoS ontology</td>
<td>Agent based system</td>
<td>Base and unit, attribute and usage domain</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Upper, Middle and lower</td>
</tr>
</tbody>
</table>
onQoS

The Upper Ontology

The Middle Ontology

The Lower Ontology

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

The QoS Metric Middle Ontology

The QoS Metric Function Middle Ontology

The QoS Scale Vocabulary Ontology

QoS Network Ontology
Automotive ontology

September 21, 2007
Eugenio Zimeo
FACS 07 - Sophia Antipolis
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

QoSPParameterValue

DeclarativeProcess

AggregationProcess

EvaluationProcess

MonitoringProcess

ReadingProcess

QoS Parameter

isParameterOf

Participant

isPerformedBy

Measurement Process

isMeasurementProcessOf

hasMeasurementProcess

QoS Metric

hasParameter

QoS Metric

hasMeasuredValue
onQoS: Middle ontology

- QoS Parameters Vocabulary

The Middle Ontology

- QoS parameters vocabulary
- QoS Metrics vocabulary
- QoS Scales vocabulary
- ....

Available Parameters:
- Availability
- Capacity
- Cost
- Integrity
- Performance
- Reliability
- Robustness
- Scalability
- Security
- WSQoS

- MTTR
- Uptime
- Throughput
- Latency
- Response Time
- MTBF
- Recoverable
- Failover
- Disaster
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

- Specialization
  - Chen Zhou
  - Dobson
  - Cardoso
  - Bleal
  - Paolucci

- Implication
  - Taher
  - Oldham
  - Kim

- Composition
  - Cardoso
  - Zeng
  - Canfora
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 ....

\[ L = RTT + t_e + e_t \]
<table>
<thead>
<tr>
<th>D1</th>
<th><strong>QoS requirements</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Authentication Authorization</strong></td>
</tr>
<tr>
<td></td>
<td>Cost $\leq 100\€$</td>
</tr>
<tr>
<td></td>
<td>EncStand: RSA, PKI, OpenPGP, Triple-DES</td>
</tr>
<tr>
<td></td>
<td><strong>ExecutionTime</strong> $\leq 0.5\ ms$</td>
</tr>
<tr>
<td></td>
<td>FaultRate $\leq 50%$</td>
</tr>
<tr>
<td></td>
<td>Jitter $\leq 0.3\ ms$</td>
</tr>
<tr>
<td></td>
<td>NetThroughput $\geq 200\ kbps$</td>
</tr>
<tr>
<td></td>
<td><strong>RTT</strong> $\leq 14\ ms$</td>
</tr>
<tr>
<td></td>
<td>Scalability $\geq 78%$</td>
</tr>
<tr>
<td></td>
<td><strong>TransmissionTime</strong> $\leq 7\ ms$</td>
</tr>
<tr>
<td></td>
<td>UpTime $\geq 90%$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>D2</th>
<th><strong>QoS requirements</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Cost $\leq 121\ \€$</td>
</tr>
<tr>
<td></td>
<td>EncStand: RSA, PKI, OpenPGP</td>
</tr>
<tr>
<td></td>
<td>ExecutionTime $\leq 0.6\ ms$</td>
</tr>
<tr>
<td></td>
<td>FaultRate $\leq 50%$</td>
</tr>
<tr>
<td></td>
<td>Jitter $\leq 1.5\ ms$</td>
</tr>
<tr>
<td></td>
<td>Privacy</td>
</tr>
<tr>
<td></td>
<td>RTT $\leq 17\ ms$</td>
</tr>
<tr>
<td></td>
<td>Scalability $\geq 43%$</td>
</tr>
<tr>
<td></td>
<td>ThrLatRatio $\geq 3.5\text{Mbps/s}$</td>
</tr>
<tr>
<td></td>
<td>UpTime $\geq 86%$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>D3</th>
<th><strong>QoS requirements</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Cost $\leq 140\ \€$</td>
</tr>
<tr>
<td></td>
<td>EncStand: RSA, PKI, OpenPGP</td>
</tr>
<tr>
<td></td>
<td>Jitter $\leq 0.3\ ms$</td>
</tr>
<tr>
<td></td>
<td><strong>NetLatency</strong> $\leq 24.9\ ms$</td>
</tr>
<tr>
<td></td>
<td><strong>Privacy</strong></td>
</tr>
<tr>
<td></td>
<td>UpTime $\geq 65%$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>D4</th>
<th><strong>QoS requirements</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td></td>
<td>Authorization</td>
</tr>
<tr>
<td></td>
<td>NetLatency $\leq 26\ ms$</td>
</tr>
<tr>
<td></td>
<td>ThrLatRatio $\geq 3.2\ \text{Mbps/s}$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>D5</th>
<th><strong>QoS requirements</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td></td>
<td>Authorization</td>
</tr>
<tr>
<td></td>
<td>EncStand: RSA, PKI, OpenPGP</td>
</tr>
<tr>
<td></td>
<td>ExecutionTime $\leq 0.8\ ms$</td>
</tr>
<tr>
<td></td>
<td>Jitter $\leq 2.6\ ms$</td>
</tr>
<tr>
<td></td>
<td>NetThroughput $\geq 10\ kbps$</td>
</tr>
<tr>
<td></td>
<td>RTT $\leq 5\ ms$</td>
</tr>
<tr>
<td></td>
<td>TransmissionTime $\leq 6\ ms$</td>
</tr>
<tr>
<td></td>
<td>UpTime $\geq 65%$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>D6</th>
<th><strong>QoS requirements</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NetLatency $\leq 22\ ms$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>D7</th>
<th><strong>QoS requirements</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td></td>
<td>Authorization</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>D8</th>
<th><strong>QoS requirements</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RTT $\leq 25\ ms$</td>
</tr>
</tbody>
</table>
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

\[ ?\text{WSQoSValue} = ?\text{RTTValue} < 5 \ || \ ?\text{JitterValue} \leq 0.3 \]

**Constants**
- \( \text{RTTConstantValue} = 5 \)
- \( \text{JitterConstantValue} = 0.3 \)

**Elementary Metrics**
- \( \text{RTTMetric} = \langle \text{RTT}, \text{RTTProcess}, \text{DoubleScale}, ?\text{RTTValue} \rangle \)
- \( \text{JitterMetric} = \langle \text{Jitter}, \text{JitterProcess}, \text{DoubleScale}, ?\text{JitterValue} \rangle \)

**WSQoS Evaluation Metrics:**
- \( \text{WSQoSRTTEval} = \langle \text{WSQoSRTT}, \text{RTTEvalProcess}, \text{WSQoSScale}, ?\text{WSQoSRTTValue} \rangle \)
- \( \text{WSQoSJitterEval} = \langle \text{WSQoSJitter}, \text{JitterEvalProcess}, \text{WSQoSScale}, ?\text{WSQoSJitterValue} \rangle \)

**WSQoS Aggregation Metric:**
- \( \text{WSQoSMetric} = \langle \text{WSQoS}, \text{WSQoSOr}, \text{WSQoSScale}, ?\text{WSQoSValue} \rangle \)

**Evaluating WSQoS metric:**
\[ ?\text{WSQoSValue} = \text{WSQoSOr}(\text{RTTEvalProcess}(\text{RTTProcess}(), \text{RTTConstantValue}), \text{JitterEvalProcess}(\text{JitterProcess}(), \text{JitterConstantValue})) \]
A Query in on-QoS-QL

\[ ?\text{WSQoSValue} = ?\text{RTTValue} < 5 \quad \text{||} \quad ?\text{JitterValue} \leq 0.3 \]
# Main predicates and aggregation functions

<table>
<thead>
<tr>
<th>Vocabulary Term</th>
<th>Measurement Scale of Arguments</th>
<th>Retrieval Semantics</th>
<th>Ranking Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equal($x_i$, $x_j$)</td>
<td>Nominal Scale</td>
<td>$i = j$</td>
<td>1</td>
</tr>
<tr>
<td>NotEqual($x_i$, $x_j$)</td>
<td></td>
<td>$i \neq j$</td>
<td>1</td>
</tr>
<tr>
<td>BetterEqualThan($x_i$, $x_j$)</td>
<td>Ordinal Scale</td>
<td>$i \in {j, \ldots, N}$</td>
<td>$\frac{1}{N}(1+i-j)$</td>
</tr>
<tr>
<td>LessEqualThan($x_i$, $x_j$)</td>
<td></td>
<td>$i \in {1, \ldots, j}$</td>
<td>$\frac{1}{N}(1+j-i)$</td>
</tr>
<tr>
<td>DoubleLessThan($x$, $y$)</td>
<td>DoubleRatio Scale</td>
<td>$x &lt; y$</td>
<td>$\frac{2}{1 + e^{\frac{y-x}{k}}} - 1$</td>
</tr>
<tr>
<td>DoubleGreaterThan($x$, $y$)</td>
<td></td>
<td>$x &gt; y$</td>
<td>$\frac{2}{1 + e^{\frac{x-y}{k}}} - 1$</td>
</tr>
<tr>
<td>WSQoSAnd($x$, $y$)</td>
<td></td>
<td>$x \land y$</td>
<td>$\min(x, y)$</td>
</tr>
<tr>
<td>WSQoSOr($x$, $y$)</td>
<td></td>
<td>$x \lor y$</td>
<td>$\max(x, y)$</td>
</tr>
<tr>
<td>WeightedMean($p_i$)</td>
<td>WSQoS Scale</td>
<td>$\exists p_i, i = 1, \ldots, N$</td>
<td>$\frac{\sum_{i=1}^{N} w_i x_i}{\sum_{i=1}^{N} w_i}$</td>
</tr>
</tbody>
</table>
The ranking engine computes a rank for each retrieved service according to the defined semantics.

\[ ?WSQoSValue = \max \left\{ \frac{2}{k_1 - ?RTTValue} - 1, \frac{2}{k_2 - ?JitterValue} - 1 \right\} \]
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 (2/3)
Autonomic Workflow Execution (3/3)

A

B

C

D

E

F

A-S1

B-S1

B-S2

C-S2.1

C-S2.2

C-S3

D-S3

E-S2

E-S3

F-S2.2

S1

S2

S3

September 21, 2007

Eugenio Zimeo

FACS 07 - Sophia Antipolis
Features

Autonomic Workflow

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

September 21, 2007  Eugenio Zimeo  FACS 07 - Sophia Antipolis
Process Entities and Relationships

- **Domain**
  - **Goal**
  - **Process**
  - **ProcessConstraint**

**Problem**

**Abstract Description**

- **ProcessElement**
  - **Functionality**
  - **Parameter**
  - **Activity**
    - **ActivityConstraint**
  - **ControlStructure**
    - **Sequence**
    - **Parallel**
    - **Conditional**
    - **Loop**

**Concrete Description**

- **ExecutableActivity**
  - **NonFunctionalProperty**
  - **BindingConstraints**
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 information about the state of managed elements
- Details are added and analyzed wrt management policies
- If the event can not be handled by the manager, the Process Manager is involved in...
Components

- Autonomic Engine
  - Engine Manager
    - Control
    - Binding
    - Interaction
  - Process Manager
  - Interaction Mgr
  - Binding Mgr

- Manager
  - Configurator
    - Composer
    - Binder
  - Action Mgr
  - Binding Mgr

- Resources
  - WS₁
  - WSᵢ
  - WSₙ

- Planner
- Match-Maker
Decentralized Self-Evolution
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 ....

Solution:
A-R [2] +
R-S [11] +
S-B [10] +
B-W [7] +
W-Z [9]
... to network re-factoring

Goal A->Z

Solution:
A-R [2] +
R-S [11] +
S-B [10] +
B-W [7] +
W-Z [9]
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 run-time verification needs sophisticated monitoring of functional and QoS properties
Thanks

zimeo@unisannio.it