Semantics to Energize the Full Services Spectrum: Ontological Approach to Better Exploit Services at Technical and Business Levels

Amit P. Sheth
Wright State University - Main Campus, amit.sheth@wright.edu

Follow this and additional works at: http://corescholar.libraries.wright.edu/knoesis

Part of the Bioinformatics Commons, Communication Technology and New Media Commons, Databases and Information Systems Commons, OS and Networks Commons, and the Science and Technology Studies Commons

Repository Citation
http://corescholar.libraries.wright.edu/knoesis/64

This Conference Proceeding is brought to you for free and open access by the The Ohio Center of Excellence in Knowledge-Enabled Computing (Kno.e.sis) at CORE Scholar. It has been accepted for inclusion in Kno.e.sis Publications by an authorized administrator of CORE Scholar. For more information, please contact corescholar@www.libraries.wright.edu.
Semantics to energize the full Services Spectrum:

Ontological approach to better exploit services at technical and business levels

Amit Sheth

LSDIS Lab, University of Georgia, Athens, Georgia, UGA

Special Thanks: Kunal Verma
Challenges

“Each enterprise will measure and aspire to its own unique level of dynamism based on its individual purpose. It is about being nimble and adaptable. A fully integrated business platform can respond faster, and completely, to change. Whether it involves fulfilling a new mandate or embracing a new market opportunity. Some organizations will push the envelope, automating event-triggered responses for highly integrated closed-loop processes, setting the stage for self-optimizing systems.”

Sandra Rogers, White Paper: Business Forces Driving Adoption of Service Oriented Architecture, Sponsored by: SAP AG
Semantic Services Sciences (3S Model)

- Based on IBM’s vision [1] of service sciences
  - Need to take a pervasive view of services
  - Modeling people and organizational aspects as well as technical aspects of services

- The 3S model [2]
  - Semantics for all types of services: Technical/Web Services to Knowledge Services

Using the 3S Model

• Consider global IT service provider developing a new multimedia service for UK telecom provider
  – Similar service already successfully provided in Japan

• To provide the new multimedia service
  – Business manager must leverage assets
  – Human assets
    • Teams in China (Telco Equipment), India (Telco SW, Back Office)
    • People who have domain expertise in the new market
    • Project Management, ...
  – Technical assets
    • Reuse SW assets and compose services to create technical platform
    • Use lightweight services for information aggregation and GUIs
Semantic Services Sciences (3S Model)

**Organization**
- Supports

**Services**
- Realized Using

**Assets**
- Humans
  - Managers, Developers, etc.,
  - Semantic Profiles
- Software
  - Applications, Databases etc.
  - Web Services
    - (SOAP, WSDL, UDDI)
  - Lightweight Web Services
    - (REST, AJAX)
  - Semantic Descriptions – WSDL-S
  - Semantic Descriptions – WSDL-S/XML Annotation

**Semantic Specifications**
- Ontologies (Enterprise, Domain Specific (SUMO Finance, General Purpose (SUMO, Cyc)
- Standards (RosettaNet, ebXML, SCOR, etc.)
- Taxonomies (NAICS etc.)
Autonomic Web Process*

- Self Healing
- Agile
- Self Optimizing
- Self Configuring

*it’s about the business, not just computing resources
Outline

- Semantics for Technical Services
  - Data Semantics *
  - Functional Semantics *
  - Non Functional Semantics *
  - Execution Semantics

- Semantics for Knowledge Services

- Conclusions

*Can be represented using ontologies
Semantics for Technical Services

Current and past focus of METEOR-S
Semantics for Technical Services

- **Data/Information Semantics**
  - **What:** (Semi-)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 Semantics**
  - (Semi-) 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**
  - (Semi-) 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.

- **Non Functional Semantics (WS-*)**
  - (Semi-) formally represent qualitative and quantitative measures of Web process
  - Non- Quantitative includes security, transactions
  - Quantitative includes cost, time etc.
  - Business constraints and inter service dependencies (Domain and application ontologies)
Semantics for Technical Services

Development / Description / Annotation
- WSDL, WSDL-S, SAWSDL, WSMO, OWL-S
- METEOR-S (MWSAF)

Composition, Configuration and Negotiation
- BPEL, WS-Agreement, WS-Policy
- METEOR-S (MWSCF)

Execution, Adaptation and Mediation
- BPWS4J, activeBPEL, WSMX
- METEOR-S

Publication / Discovery
- (Semantic) UDDI
- METEOR-S (MWSDI)
Semantics for Technical Services

Execution, Adaptation and Mediation
- BPWS4J
- activeBPEL
- WSMX
- METEOR-S

Development / Description / Annotation
- WSDL, WSDL-S
- SAWSDL, WSMO
- OWL-S
- METEOR-S

Composition, Configuration and Negotiation
- BPEL, WS-Agreement
- WS-Policy
- METEOR-S

Data / Information Semantics

Publication / Discovery
- (Semantic) UDDI

METEOR-S
(MWSDI)

(MWSCF)
Semantics for Technical Services

Execution, Adaptation and Mediation
- BPWS4J, activeBPEL, WSMX, METEOR-S

Development / Description / Annotation
- WSDL, WSDL-S, SAWSDL, WSMO, OWL-S, METEOR-S (MWSCF)

Composition, Configuration and Negotiation
- BPEL, WS-Agreement, WS-Policy, METEOR-S (MWSCF)

Publication / Discovery
- (Semantic) UDDI, METEOR-S (MWSDI)

Functional Semantics
Semantics for Technical Services

Non Functional Semantics

Development / Description / Annotation

- BPWS4J, activeBPEL, WSMX, METEOR-S
- WSDL, WSDL-S, SAWSDL, WSMO, OWL-S, METEOR-S (MWSCF)
- (Semantic) UDDI, METEOR-S (MWSAF)

Execution, Adaptation and Mediation

Composition, Configuration and Negotiation

Publication / Discovery

BPEL, WS-Agreement, WS-Policy, METEOR-S (MWSCF)
Semantics for Technical Services

- **Execution, Adaptation and Mediation**
  - BPWS4J, activeBPEL, WSMX, METEOR-S

- **Composition, Configuration and Negotiation**
  - BPEL, WS-Agreement, WS-Policy, METEOR-S (MWSCF)

- **Development / Description / Annotation**
  - WSDL, WSDL-S, SAWSDL, WSMO, OWL-S, METEOR-S (MWSAF)

- **Publication / Discovery**
  - (Semantic) UDDI, METEOR-S (MWSDI)
Semantics for Technical Services

Execution, Adaptation and Mediation
- BPWS4J
- activeBPEL
- WSMX
- METEOR-S

Composition, Configuration and Negotiation
- BPEL
- WS-Agreement
- WS-Policy
- METEOR-S (MWSCF)

Development / Description / Annotation
- WSDL, WSDL-S, SAWSDL, WSMO, OWL-S
- METEOR-S (MWSAF)

Execution Semantics

QoS Semantics

Data / Information Semantics

Functional Semantics

Publication / Discovery
- (Semantic) UDDI
- METEOR-S (MWSDI)
DATA SEMANTICS
Data Semantics

Customer Process

- Locate Suppliers
- Send Quote Request
- Choose Supplier
- Negotiate Agreement
- Send Order

Supplier Process

- UDDI Registry
- UDDI Query
- Results

How does the supplier recognize Item Details?

- Receive Quote
- Check Inventory
- Negotiate Agreement
- Receive Order
Data Semantics - options

• Pre-defined agreement on all data fields
  - Limited flexibility, hard to integrate new suppliers in process

• Use a standard like Rosetta Net/ebXML
  - Greater flexibility, but limited to suppliers following standard
  - Standard may not be expressive enough for everyone's needs

• Annotate data fields with domain ontologies
  - Most flexible, semi-automatic transformation based on ontology mapping
  - Ontology can be based on domain standard, while providing more flexibility and extensibility
WSDL-S Specification
(Now the key input to W3C leading to Semantic Annotation of WSDL-
SAWSDL)
PurchaseOrder.wsdl

............

<xsl:element name="OrderConfirmation" type="xs:string
wssem:modelReference="rosetta#PurchaseOrderResponse"/>
</xsl:schema>
</types>
</interface name="PurchaseOrder">
<wssem:category name="Electronics" taxonomyURI=http://www.naics.com/
taxonomyCode="443112"/>

<operation name="order" pattern=wsdl:in-out
modelReference = "rosetta#RequestPurchaseOrder">
<input messageLabel = "processPurchaseOrderRequest"
element="tns:processPurchaseOrderRequest"/>
<output messageLabel ="processPurchaseOrderResponse"
element="processPurchaseOrderResponse"/>

<!—Precondition and effect are added as extensible elements on an operation>
<wssem:precondition name="ExistingAcctPrecond"
wssem:modelReference="POOntology#AccountExists">
<wssem:effect name="ItemReservedEffect"
wssem:modelReference="POOntology#ItemReserved"/>
</operation>
</interface>
Representing mappings

```xml
<complexType name="POAddress"
  <all>
    <element name="streetAddr1" type="string"/>
    <element name="streetAdd2" type="string"/>
    <element name="poBox" type="string"/>
    <element name="city" type="string"/>
    <element name="zipCode" type="string"/>
    <element name="state" type="string"/>
    <element name="country" type="string"/>
    <element name="recipientInstName" type="string"/>
  </all>
</complexType>
```

**WSDL complex type element**

**OWL ontology**

Mapping using XSLT

```xml
....
<xsl:template match="/"

<POOntology:Address rdf:ID="Address1"/>
<POOntology:has_StreetAddress rdf:datatype="xs:string">
  <xsl:value-of select="concat(POAddress/streetAddr1,POAddress/streetAddr2)"/>
</POOntology:has_StreetAddress>
</POOntology:has_StreetAddress>

<POOntology:has_City rdf:datatype="xs:string">
  <xsl:value-of select="POAddress/city"/>
</POOntology:has_City>

<POOntology:has_State rdf:datatype="xs:string">
  <xsl:value-of select="POAddress/state"/>
</POOntology:has_State>

....
```
FUNCTIONAL SEMANTICS
How to locate appropriate supplier?
Functional Semantics

• Keyword based search in UDDI
  – Needs human involvement
  – Low precision and high recall

• Port Type based search in UDDI
  – Requires service providers to agree on port types
  – Less flexible, requires total agreement on method names and data type names

• Template Based Semantic Discovery
  – Requires ontological commitment of data types and operations
  – Can search on any or many aspects of description+interface
  – Can have complex similarity measures and be used to provide ranked results based on similarity
Semantic Templates

- Semantic Templates capture the functionality of a Web service with the help of ontologies/other domain models
- Find a service that sells RAM in Athens, GA. It must allow the user to return and cancel, if needed
- The template can also have non-functional (QoS) requirements such as response time, security, etc.

**SEMANTIC TEMPLATE**

**Service Level Metadata (SLM)**
IndustryCategory = NAICS: Electronics
ProductCategory = DUNS: RAM
Location = Athens, GA

**Operation 1**
- Action = Rosetta#RequestPurchaseOrder
- Input = Rosetta#PurchaseOrderRequest
- Output = Rosetta#PurchaseConfirmation
- Policy = {Encryption = RSA, ResponseTime < 5 sec}

**Operation 2**
- Action = Rosetta#CancelOrder

---

**Data Semantics**
**Functional Semantics**
**Non-Functional Semantics**
Semantic Discovery

- Finds actual services matching semantic templates
- Implemented as a layer over UDDI [1]
- Current implementation based on ontological representation of operations, inputs and outputs
- Returns ranked of services for each semantic template
- Builds upon following previous discovery implementations
  - Extends matching presented in [2] to consider operations and service level metadata
  - Extends the approach presented “WSDL to UDDI Mapping” [3] to support operation level discovery


Non Functional Semantics

Business and Application constraints
Non Functional Semantics

Customer Process

Locate Suppliers

Send Quote Request

Choose Supplier

Negotiate Agreement

Send Order

UDDI Query

Results

Item Details

Quote Details

QoS Semantics

Receive Quote

Check Inventory

Negotiate Agreement

Receive Order

Supplier Process

UDDI Registry
Non Functional Semantics

• Does the supplier support customer’s business constraints
  – e.g. cost, supply time etc.
• Interaction should adhere to the entities’ policies
  – e.g. security, transactions
• In case of more suppliers, domain constraints should be satisfied
  – e.g. a certain supplier’s parts do not work with other supplier’s parts
Non Functional Semantics

- Used in lifecycle
  - Agreement Matching
    - Matching syntactically heterogeneous by semantically homogeneous agreements
  - Dynamic Process Configuration
    - Configuring process based on process constraint

We will demonstrate how ontology-driven semantic approach supports these capabilities.
SWAPS: Use of Semantics in Agreement Matching

An agreement is a collection of alternatives.
A={Alt1, Alt2, ..., AltN}

An alternative is a collection of guarantees.
Alt={G1, G2, ...GN}

"requirement(Alt, G)" returns true if G is a requirement of Alt
"capability(Alt, G)" returns true if G is an assurance of Alt
"scope(G)" returns the scope of G
"obligation(G)" returns the obligated party of G
"satisfies(Gj, Gi)" returns true if the SLO of Gj is equivalent to or stronger than the SLO of Gi

An alternative Alt1 is a suitable match for Alt2 if:
(\exists Gi such that Gi \in Alt1 \land requirement(Alt1, Gi) \land (\forall Gj such that Gj \in Alt2 \land capability(Alt2, Gj) \land scope(Gi) = scope(Gj) \land obligation(Gi) = obligation(Gj) \land satisfies(Gj, Gi)))
An agreement consists of a collection of Guarantee terms.

A guarantee term has a scope - e.g. operation of service.

There might be business values associated with each guarantee terms. Business values include importance, confidence, penalty, and reward.

A guarantee service level e.g. response time e.g. num

A guarantee condition e.g. num

OWL ontology

Agreement represented as an instance of ontology
SWAPS Ontologies

- **WS-Agreement:** individual agreements are instances of the WS-Agreement ontology
- **Temporal Concepts:** time.owl (OWL version of DAML time http://www.isi.edu/~pan/damltime/time.owl)
  - Concepts: seconds, dayOfWeek, ends
- **Quality of Service:** Max Maximilien’s QoS ontology (IBM) -> Ont-Qos
  - Concepts: responseTime, failurePerDay
- **Domain Ontology:** an ontology used to represent the domain
Using Semantic Agreements with WSDL-S

Adding Semantics to Agreements:
- Improves Monitoring and Negotiation
- Improves the accuracy of matching

Adding Semantics to Web Services:
- Enables more accurate discovery and composition.

Time
QoS

Domain Independent

Domain Dependent

Adding Semantics to Agreement Ontology:
- GetMoisture
- GetWeight
- GetPrice
- GetSplits

Agriculture Ontology:
- Crop
- Quality
- Price
- Split
- Moisture
- Weight

Merchant Service WSDL-S

WS-Agreement Ontology:
- Guarantee
- Scope
- Obligated
- SLO
- Predicate
- Less
- Greater

Input: Address
FarmerAdder

Adding Semantics to Agreements: Improves Monitoring and Negotiation
Improves the accuracy of matching.
## Evaluation

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>responseTime &lt; 5</td>
<td>responseTime &lt; 4</td>
<td>YES</td>
<td>YES</td>
<td>YES, but only if parameters are named similar syntactically</td>
<td>YES, but only if parameters are named similar syntactically</td>
</tr>
<tr>
<td></td>
<td>(duration1 + duration2) &lt; 4</td>
<td>YES</td>
<td>NO</td>
<td>YES, but only if the parameters are named similar syntactically to the rule criteria</td>
<td>NO</td>
</tr>
<tr>
<td>responseTime &lt; 5</td>
<td>rt &lt; 4</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>responseTime &lt; 5</td>
<td>networkTime &lt; 2 executionTime &lt; 1</td>
<td>YES</td>
<td>NO</td>
<td>YES, but only if the parameters are named similar syntactically to the rule criteria</td>
<td>NO</td>
</tr>
</tbody>
</table>
The Matching Process

Consumer

Provider1

Provider2

Obligated: Provider
responseTime < 14 s
QC: day of week = weekday
Penalty: 15 USD

Obligated: Provider
FailurePerWeek < 7
Penalty: 10 USD

Obligated: Provider
transmitTime < 4 s
QC: maxNumUsers < 1000
Penalty: 2 USD

Obligated: Provider
ProcessTime < 5 s
QC: numRequests < 500
Penalty: 1 USD

Obligated: Provider
99% of responseTimes < 14 s

Obligated: Provider
failurePerWeek < 10

Obligated: Provider
failurePerWeek < 7
Penalty: 10 USD

Obligated: Provider
failurePerWeek < 7
Penalty: 2 USD
The Matching Process

Knowledge from Domain Specific Rules:

\[
\text{if } (x \geq 96) \\
\quad \text{responseTime} < y \\
\text{else} \\
\quad \text{responseTime} > y
\]
The Matching Process

Consumer

Provider1

Obligated: Provider
responseTime < 14 s

Obligated: Provider
responseTime < 14 s
QC: day of week = weekday
Penalty: 15 USD

Obligated: Provider
failurePerWeek <10

Obligated: Provider
FailurePerWeek < 7
Penalty 10USD

Knowledge from Semantics of Predicate Rules
The Matching Process

Knowledge from Semantics of Predicate Rules
The Matching Process

Domain Specific Rule
responseTime = transmitTime + processTime
The Matching Process

Consumer

Obligated: Provider
responseTime < 14 s

Obligated: Provider
failurePerWeek < 10

Provider

Obligated: Provider
responseTime < 9 s

QC: maxNumUsers < 1000 AND
numRequests < 500

Penalty: 2USD

Obligated: Provider
failurePerWeek < 7

Penalty: 1 USD

Penalty: 2USD
The Matching Process

Consumer

Obligated: Provider responseTime < 14 s

Obligated: Provider failurePerWeek < 10

isStronger

Provider

Obligated: Provider responseTime < 9 s
QC: maxNumUsers < 1000 AND numRequests < 500
Penalty: 2 USD

Obligated: Provider failurePerWeek < 7
Penalty: 1 USD

Steps #5-6: Comparison Rules
The Matching Process

User Preference Rule:
\(\text{dayofWeek} = \text{weekday}\) notSuitable
The Matching Process

Consumer
- Obligated: Provider responseTime < 14 s
- Obligated: Provider failurePerWeek < 10

Provider1
- Obligated: Provider responseTime < 14 s
- Obligated: Provider FailurePerWeek < 7
- Penalty: 15 USD

Provider2
- Obligated: Provider responseTime < 9 s
- QC: day of week = weekday
- Penalty: 15 USD
- Obligated: Provider failurePerWeek < 7
- Penalty: 2 USD

Penalty: 1 USD
- Obligated: Provider responseTime < 9 s
- QC: maxNumUsers < 1000 AND numRequests < 500
- Penalty: 1 USD
Dynamic Process Configuration

- Operations Research has been used in industry for business process optimization

- There is often a lot of domain knowledge in business process optimization
  - Minds of analysts/experts
  - Hidden in databases/texts

- We try to explicitly capture domain knowledge and link with IT systems
Dynamic Process Configuration

Find optimal partners for the process based on process constraints – cost, supply time, etc.

**Conceptual Approach**

1. Create framework to capture represent domain knowledge
2. Represent constraints on the domain knowledge
3. Ability to reason on the constraints and configure the process
Dynamic Process Configuration

Research Challenges

- Capturing functional and non-functional requirements of the Web process (Abstract process specification)
- Discovering service partners based on functional requirements (Semantic Web service discovery)
- Choosing optimal partners that satisfy non-functional requirements (Constraint Analysis)

Abstract Process Specification

1. Specify process control flow by using virtual partners

2. Specify Process Constraints

3. Capture Functional Requirements of Services using Semantic Templates
Process Constraints

• Constraints can be specified on a partner, an activity or the process as a whole.
• An objective function can also be specified e.g., minimize cost and supply-time, etc.
• Two types of constraints:
  – Quantitative (Q) (Time < 5 sec)
  – Logical (L) (preferredPartner, Security, etc.)
# Process Constraints

<table>
<thead>
<tr>
<th>Feature</th>
<th>Scope</th>
<th>Goal</th>
<th>Value</th>
<th>Unit</th>
<th>Aggregation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cost (Quantitative)</td>
<td>Process</td>
<td>Minimize</td>
<td>Dollars</td>
<td>Σ</td>
<td></td>
</tr>
<tr>
<td>Supplytime (Quantitative)</td>
<td>Process</td>
<td>Satisfy</td>
<td>&lt; 7</td>
<td>Days</td>
<td>MAX</td>
</tr>
<tr>
<td>Cost (Quantitative)</td>
<td>Activity</td>
<td>Satisfy</td>
<td>&lt;200000</td>
<td>Dollars</td>
<td>Σ</td>
</tr>
<tr>
<td>PreferredSupplier(P1) (Logical)</td>
<td>Partner 1</td>
<td>Satisfy</td>
<td>True</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Compatible (P1, P2) (Logical)</td>
<td>Process</td>
<td>Satisfy</td>
<td>True</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Constraint Analysis

• Multi-paradigm proposed:
  – Integer Linear Programming for quantitative constraints
  – Semantic Web Rule Language and OWL for domain constraints

• Discovered Services first given to ILP solver
  – It returns ranked sets of services

• Then each set is checked for logical constraints using a SWRL reasoner
  – Sets not satisfying the criteria are rejected
Rules

- **Supplier 1 should be a preferred supplier.**
  - “if S1 is a supplier and its supplier status is preferred then the S1 is a preferred supplier”.

  \[
  \text{Supplier} \; (?S1) \text{ and } \text{partnerStatus} \; (?S1, \text{"preferred"}) \Rightarrow \text{preferredSupplier} \; (?S1)
  \]

- **Supplier 1 and supplier 2 should be compatible.**
  - if S1 and S2 are suppliers and they supply parts P1 and P2, respectively, and the parts work with each other, then suppliers S1 and S2 are compatible for parts P1 and P2.

  \[
  \text{Supplier} \; (?S1) \text{ and } \text{supplies} \; (?S1, \; ?P1) \text{ and Supplier} \; (?S2) \text{ and } \text{supplies} \; (?S2, \; ?P2) \text{ and } \text{worksWith} \; (?P1, \; ?P2) \Rightarrow \text{compatible} \; (?S1, \; ?S2, \; ?P1, \; ?P2)
  \]

  \[
  \text{RAM} \; (?P1) \text{ and MB} \; (?P2) \text{ and } \text{worksWithMB} \; (?P1, \; ?P2) \Rightarrow \text{worksWith} \; (?P1, \; ?P2)
  \]
Configuration Step 1: Semantic Discovery

DISCOVERY RESULTS – List of candidate service for each template

- **MB Candidate Service 1 (M1)**
  - Q: Cost = $110000
  - Q: SupplyTime < 7 Days
- **MB Candidate Service 2 (M2)**
  - Q: Cost = $145000
  - Q: SupplyTime < 7 Days
- **MB Candidate Service 4 (M4)**
  - Q: Cost = $185000
  - Q: SupplyTime < 6 Days

- **RAM Candidate Service 1 (R1)**
  - Q: Cost = $45000
  - Q: SupplyTime < 5 Days
- **RAM Candidate Service 3 (R3)**
  - Q: Cost = $40000
  - Q: SupplyTime < 8 Days
- **RAM Candidate Service 4 (R4)**
  - Q: Cost = $41000
  - Q: SupplyTime < 5 Days

- **Processor Candidate Service 1 (P1)**
  - Q: Cost = $210000
  - Q: SupplyTime < 5 Days
- **Processor Candidate Service 3 (P3)**
  - Q: Cost = $255000
  - Q: SupplyTime < 8 Days
- **Processor Candidate Service 4 (P4)**
  - Q: Cost = $228000
  - Q: SupplyTime < 5 Days

PROCESS CONSTRAINTS
- Q: Cost <= $600000
- Q: SupplyTime < 7 Days
- L: Compat (RAM, MB) = True
- L: Compat (PROC, MB) = True
- L: preferredSupplier(S1) = True
- Min: Cost

CONTRAINT ANALYSIS MODULE

- **ILP Solver**
  - ILP SOLVER RESULTS - Service Sets that satisfy all quantitative constraints in increasing Cost order
    1. R1, M2, P1
      - Cost = $400000
    2. R4, M1, P3
      - Cost = $410000
    3. R4, M2, P3
      - Cost = $441000

- **SWRL Reasoner**
  - SWRL REASONER RESULTS - Service sets that satisfy both quantitative and non-quantitative constraints
    1. R1, M2, P1
      - Cost = $400000
    2. R4, M1, P3
      - Cost = $410000

(REJECTED SET 3 as R4 not compatible with M2 and P3 not compatible with M2)
Configuration Step 2: Quantitative Constraint Analysis

**PROCESS CONSTRAINTS**

<table>
<thead>
<tr>
<th>Constraint</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q. Cost</td>
<td>&lt;= $600000</td>
</tr>
<tr>
<td>Q. SupplyTime</td>
<td>&lt; 7 Days</td>
</tr>
<tr>
<td>L. Compat (RAM, MB)</td>
<td>True</td>
</tr>
<tr>
<td>L. Compat (PROC, MB)</td>
<td>True</td>
</tr>
<tr>
<td>L. preferredSupplier(S1)</td>
<td>True</td>
</tr>
</tbody>
</table>

**ILP SOLVER RESULTS** - Service Sets that satisfy all quantitative constraints in increasing Cost order

1. **R1, M2, P1**
   - Cost = $400000
2. **R4, M1, P3**
   - Cost = $410000
3. **R4, M2, P3**
   - Cost = $441000

**SWRL REASONER RESULTS** - Service sets that satisfy both quantitative and non-quantitative constraints

1. **R1, M2, P1**
   - Cost = $400000
2. **R4, M1, P3**
   - Cost = $410000

(REJECTED SET 3 as R4 not compatible with M2 and P3 not compatible with M2)
Configuration Step 3: Logical Constraint Analysis

DISCOVERY RESULTS – List of candidate service for each template

**MB Candidate Service 1 (M1)**
- Q: Cost = $110000
- Q: Supply Time < 7 Days

**MB Candidate Service 2 (M2)**
- Q: Cost = $145000
- Q: Supply Time < 7 Days

**MB Candidate Service 4 (M4)**
- Q: Cost = $185000
- Q: Supply Time < 6 Days

**RAM Candidate Service 1 (R1)**
- Q: Cost = $45000
- Q: Supply Time < 5 Days

**RAM Candidate Service 3 (R3)**
- Q: Cost = $40000
- Q: Supply Time < 8 Days

**RAM Candidate Service 4 (R4)**
- Q: Cost = $41000
- Q: Supply Time < 8 Days

**Processor Candidate Service 1 (P1)**
- Q: Cost = $210000
- Q: Supply Time < 5 Days

**Processor Candidate Service 3 (P3)**
- Q: Cost = $255000
- Q: Supply Time < 8 Days

**Processor Candidate Service 4 (P4)**
- Q: Cost = $228000
- Q: Supply Time < 5 Days

**Constraint Analysis Module**

**ILP Solver**

**ILP Solver Results**
Service Sets that satisfy all quantitative constraints in increasing Cost order
1. R1, M2, P1
   Cost = $400000
2. R4, M1, P3
   Cost = $410000
3. R4, M2, P3
   Cost = $441000

**SWRL Reasoner**

**SWRL Reasoner Results**
Service sets that satisfy both quantitative and non-quantitative constraints
1. R1, M2, P1
   Cost = $400000
2. R4, M1, P3
   Cost = $410000
(REJECTED SET 3 as R4 not compatible with M2 and P3 not compatible with M2)

**Semantic Templates (ST1, ST2 and ST3) from Abstract Process Specification**

**UDDI Registry with Semantic Layer**
EXECUTION SEMANTICS
Execution Semantics

1. How to recover from physical/ logical errors (e.g. delays in goods)
Process Adaptation

• Ability to adapt the processes from failures, unexpected events

• Two kinds of failures
  – Failures of physical components like services, processes, network
    • Can replace services using dynamic configuration
  – Logical failures like violation of SLA constraints/Agreements such as Delay in delivery, partial fulfillment of order
    • Need additional decision making capabilities
Process Adaptation

Adaptation Problem

Optimally react to events like delays in ordered goods

Conceptual Approach

1. Maintain states of the process – normal states, error states, goal states
2. Capture costs while transitioning from error states to goal state
3. Ability to decide optimal actions on the basis of state

K. Verma, A. Sheth, Autonomic Web Processes, ICSOC 2005
Process Adaptation

- **Research Challenges**
  - Creating a model to recover from failures and handle future events
  - Model must deal with two important factors
    - Uncertainty about when a failure occurs
    - Cost based recovery

- **Scenario**
  - After order for MB and RAM are placed, they may get delayed
  - The manufacturer may have severe costs if assembly is halted
  - It must evaluate whether it is cheaper to cancel/return and reorder or take the penalty of delay
  - Caveat: possible that reordered goods may be delayed too

- **Proposed Solution**
  - Modeling decision making capabilities of Service Managers as Markov Decision Processes (MDPs)
Generating States using preconditions and effects

Actions

- **Operation: Order**
  - Pre: Ordered = False
  - Post: Ordered = True

- **Operation: Cancel**
  - Pre: Ordered = True & Received = false
  - Post: Canceled=True & Ordered = false

- **Operation: Return**
  - Pre: Ordered = True & Received = True
  - Post: Returned = True & Ordered = false and Received = false

Events

- **Event: Delayed**
  - Pre: Ordered = True & Received = false
  - Post: Delayed=True & Ordered = True

- **Event: Received**
  - Pre: Ordered = True & Received = false
  - Post: Received = True

Flags

- Ordered
- Received
- Delayed
- Cancelled
- Returned

Use an algorithm similar to reachability analysis to generate states
## Generated State Transition Diagram

<table>
<thead>
<tr>
<th>State No.</th>
<th>Values of Boolean variables</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>&lt;O C R Del Rec &gt;</td>
<td>Ordered</td>
</tr>
<tr>
<td>2</td>
<td>&lt;O C R Del Rec &gt;</td>
<td>Ordered and Canceled</td>
</tr>
<tr>
<td>3</td>
<td>&lt;O C R Del Rec &gt;</td>
<td>Ordered and Delayed</td>
</tr>
<tr>
<td>4</td>
<td>&lt;O C R Del Rec &gt;</td>
<td>Ordered, Received and Returned</td>
</tr>
<tr>
<td>5</td>
<td>&lt;O C R Del Rec &gt;</td>
<td>Ordered, Delayed and Cancelled</td>
</tr>
<tr>
<td>6</td>
<td>&lt;O C R Del Rec &gt;</td>
<td>Ordered, Delayed, Received and Returned</td>
</tr>
<tr>
<td>7</td>
<td>&lt;O C R Del Rec &gt;</td>
<td>Ordered, Delayed and Received</td>
</tr>
<tr>
<td>8</td>
<td>&lt;O C R Del Rec &gt;</td>
<td>Ordered and Received</td>
</tr>
</tbody>
</table>
Costs and Probabilities

- Costs of ordering taken from configuration module
  - From first two service sets
    - Optimal supplier and alternate supplier
- Probability of delay and cost of returning and canceling taken from supplier policy
  - Can be represented using WS-Policy or WS-Agreement
Semantics for Lightweight Services
Lightweight services and Mashups

- REST based implementation becoming popular
  - SOAP -> Web service
  - REST -> Lightweight Web service

- REST services exposed as API’s
  - Eg. Google Maps API, Flickr API

- Mashups combine information from different services on the Web to create services with additional value

- Asynchronous Javascript And XML (AJAX) is primarily used by mashups to display the results to the user
Current limitations and Role of semantics

- Current Mashups tightly coupled (lack dynamism)
  - E.g. HousingMaps.com uses craigslist and Google maps.

- Tight binding limits effectiveness
  - Better information may be available for a specific area
  - E.g. for Atlanta area, realtor1.com might be a better service than craigslist.

- Can annotate XML for automated integration
An example

• Consider a mashup: mybook.com
  – Allows users to search and buy used and new books
  – Gets data from various vendors on the web

• Can customize vendors based on requests
  – E.g., discover two vendors, ubn.com and yaos.com on the fly

• Use conceptual model/ontology based annotation of XML data for integration
  – mybook.com can interpret the XML documents from vendors with help of annotations
An Example of Smashup (Semantic mashup)
Semantics for Knowledge Services

Current and past focus of METEOR-S
Semantics for Knowledge Services

• Work in last two decades on knowledge modeling not so successful
  – Focus on capturing knowledge
  – However most businesses use people to solve problems not expert systems

• Knowledge service try to create semantic profiles of human expertise
  – Focus on “who can” not “how to”
  – Use of ontologies for shared descriptions
High Level Model for Knowledge Services
Using Model for Knowledge Services

• Such a model can be used to answer questions
  – Find managers who have led project worth at least a million dollars
  – Find developers who have created multimedia services using Java
  – Find consultants who have some expertise in Law
Autonomic Web Processes

• The goal (Albatross)
  – Self Configuring, Self Healing, Self Optimizing, Self Protecting Business Processes

• Realization
  – Comprehensive modeling of business processes using 3S model

• Advantages
  – Alignment of technology with business goals
  – Dynamic processes that adapt with the changing environment
Conclusions

• Businesses trying perceive IT as an extension of business strategy
  – 3S Model uses semantics to provide a comprehensive model of human and technical assets
  – Modeling and exploitation of four types of semantics

• CS Researchers must take a more pervasive view of services