

RuleML: Rule Markup and Modeling Language

25/26th Juni 2007

RuleML non-profit Inc.
<http://www.ruleml.com>

RuleML, Reaction RuleML and RBSLA

- RuleML
- Reaction RuleML
- RBSLA (Rule Based Service Level Agreements)
- Summary

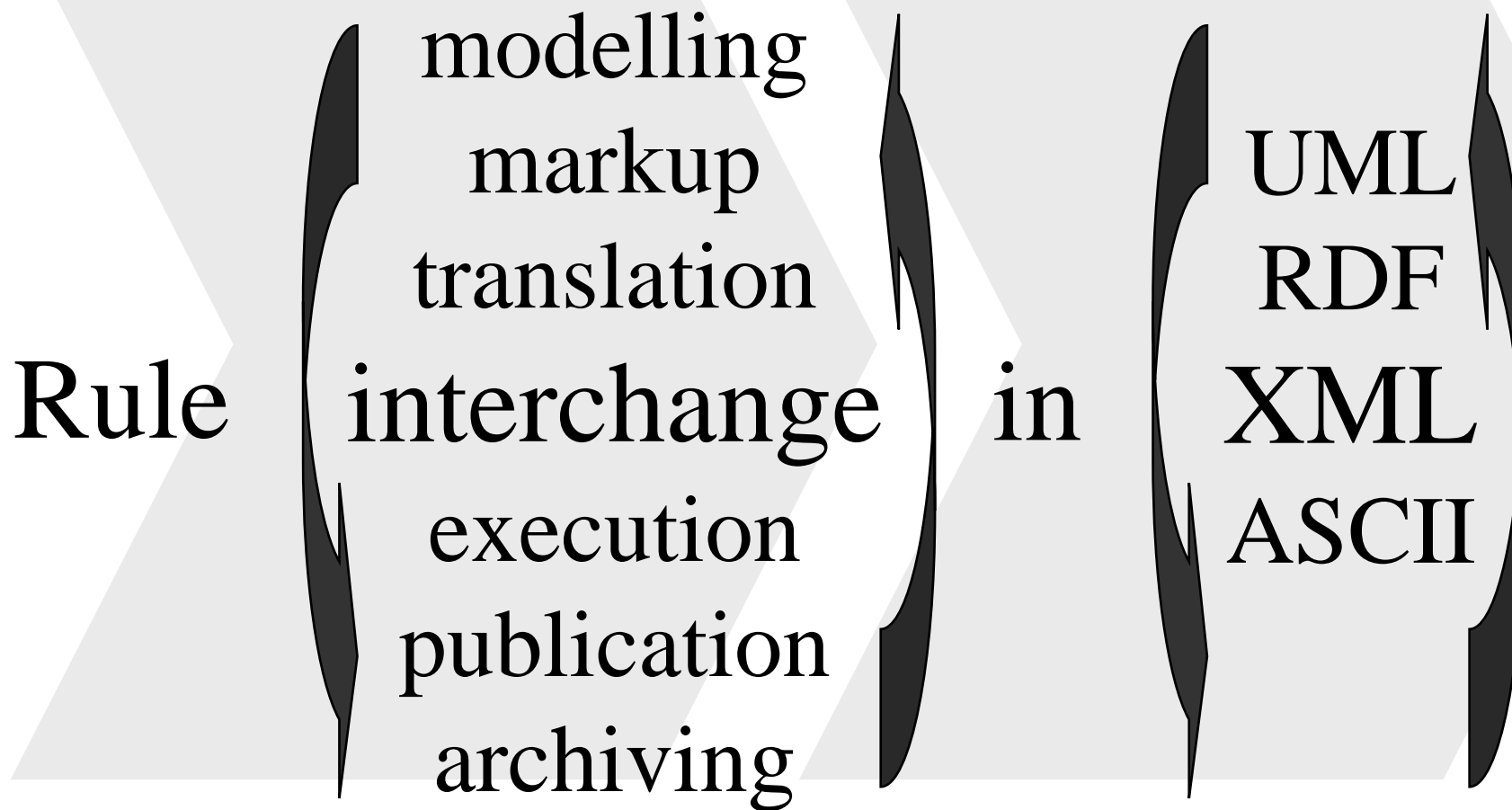
Adrian Paschke

5. Experten Gespräch „Domain Specific Reference Models for Event Patterns“, Regensburg, Germany, June 25/26, 2007

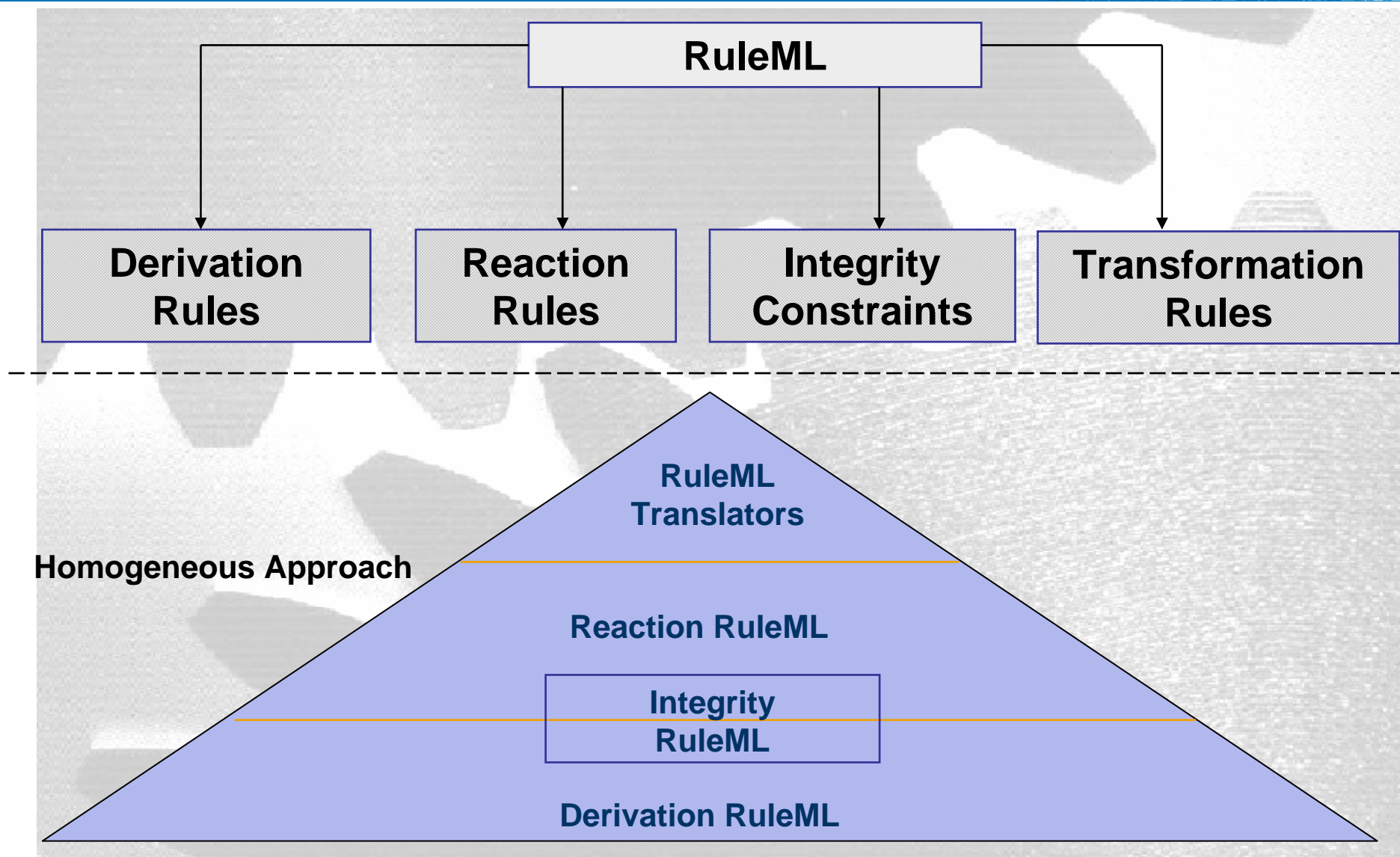


- RuleML non-profit Inc.
 - www.ruleml.com
 - Standardization of a rule markup and modelling language, tools and applications
 - Coordinates the work of the Rule Markup and Modeling initiative (www.ruleml.org)
- RuleML is the de facto open language standard for rule interchange/markup on the web
- General and open intermediary between various “specialized” vendors, applications, industrial and research working groups and standardization efforts
- Collaborating with W3C ([RIF](#)), OMG (PRR, SBVR), OASIS, DARPA-DAML and other standards/gov't bodies

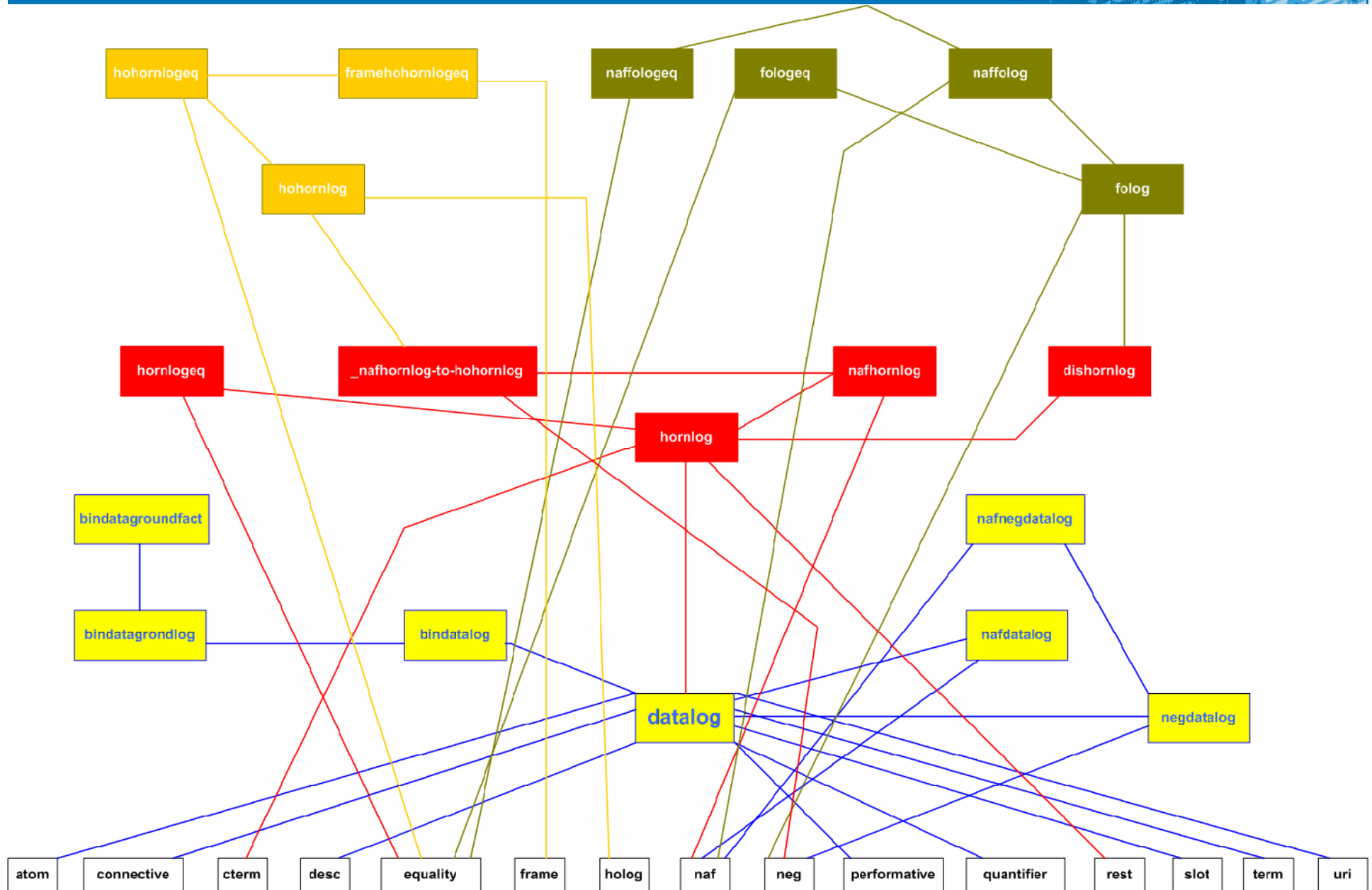
RuleML Enables ...



RuleML Language Family

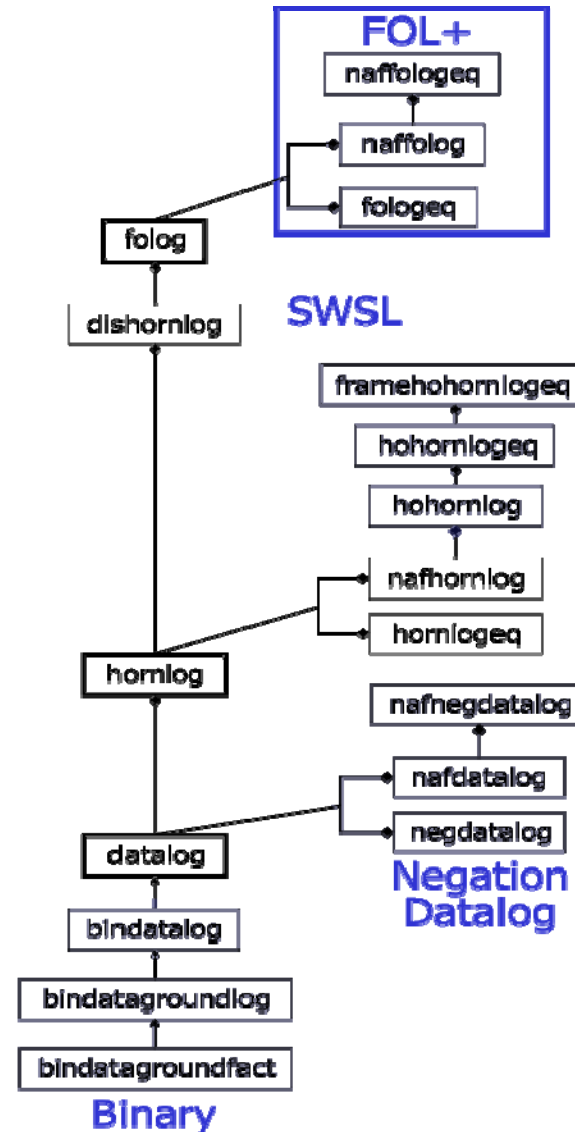


RuleML Language Family – Derivation RuleML



Schema Modularization

- XML Schema + EBNF Syntax
- Full RDF compatibility via type and role tags (akin to triple syntax);
- XML Schema Modularization: Layered and uniform design
 - The layers are organized around increasing expressiveness levels
 - Benefits:
 - easier to learn the language and to understand their relationships
 - facilitates reusability and complex language assemblies from modules
 - provides certain guidance to vendors who might be interested only in a particular subset of the features
 - easier to maintain, manage and extend in a distributed environment

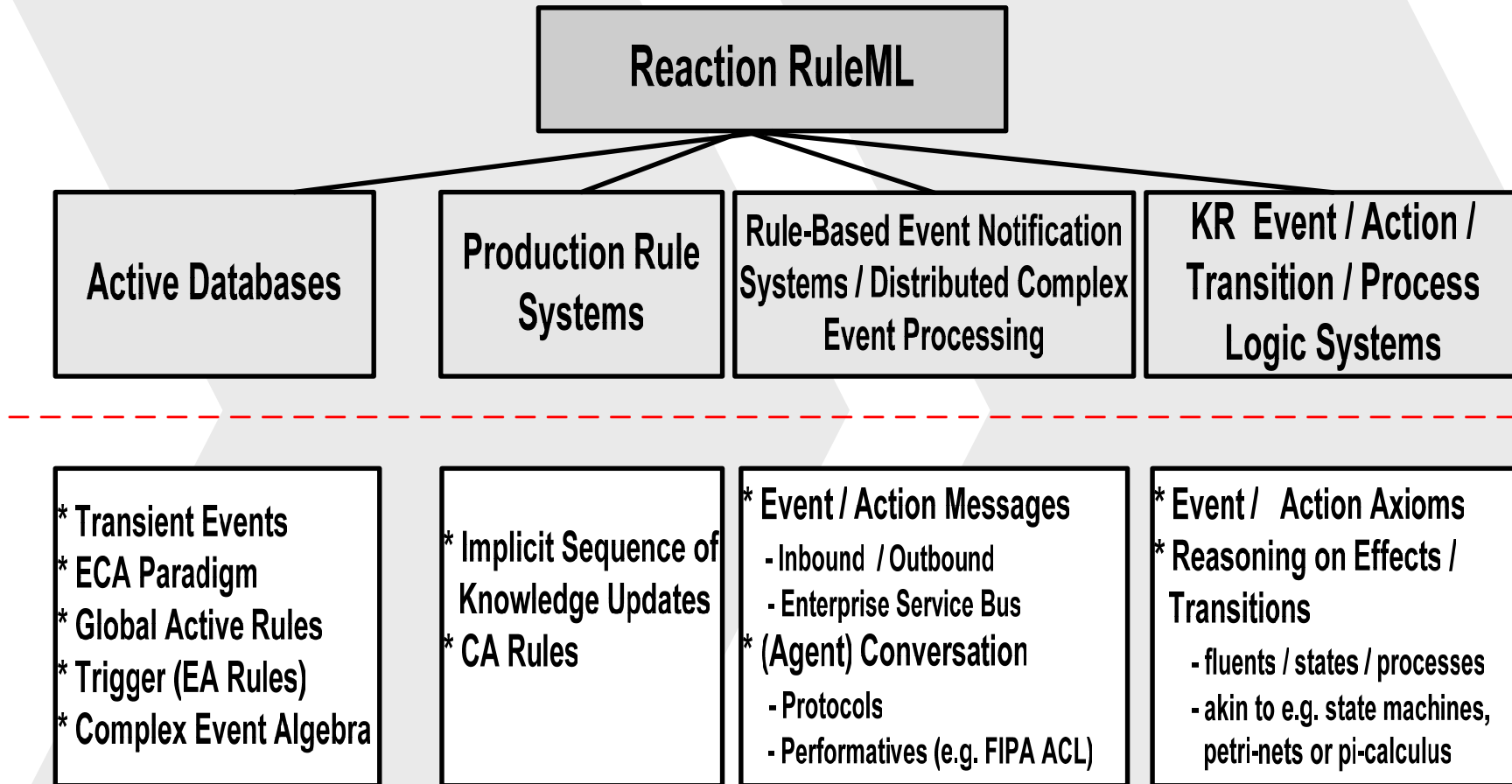




Reaction RuleML

<http://ibis.in.tum.de/research/ReactionRuleML/>

Scope of Reaction RuleML



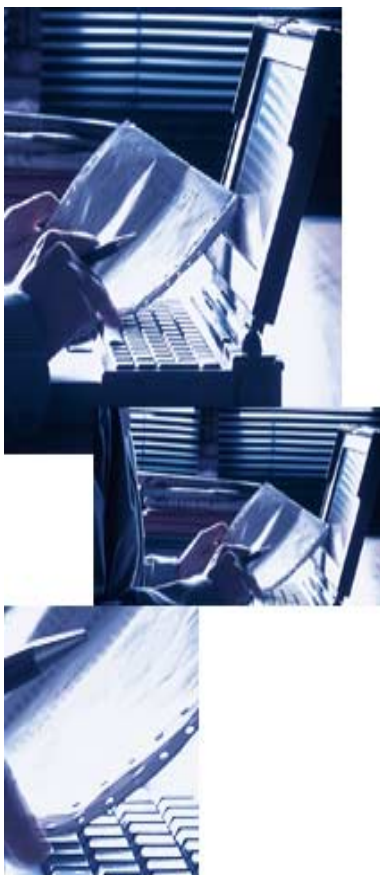
Reaction RuleML is intended for ...

- Enable interoperation between various domains of reaction rules, event/action definition and processing such as:
 - Event Processing Networks
 - Event Driven Architectures (EDAs)
 - Reactive, rule-based Service-Oriented Architectures (SOAs)
 - Active Semantic Web Applications
 - Real-Time Enterprise (RTE)
 - Business Activity Management (BAM)
 - Business Performance Management (BPM)
 - Service Level Management (SLM) with active monitoring and enforcing of Service Level Agreements (SLAs) or e-Contracts
 - Supply Chain Event Management
 - Policies
 - ...

General Concepts (1)

- General (reaction) rule form that can be specialized as needed
- Three general execution styles:
 - **Active:** 'actively' polls/detects occurred events in global ECA style, e.g. by a ping on a service/system or a query on an internal or external event database
 - **Messaging:** Waits for incoming complex event message
 - **Reasoning:** KR event/action logic reasoning and transitions (as e.g. in Event Calculus, Situation Calculus, TAL formalizations)
- Appearance
 - **Global:** 'globally' defined reaction rule
 - **Local:** 'locally' defined (inline) reaction rule nested in an outer rule

General Syntax for Reaction Rules (Reaction RuleML 0.2)



```
<Rule execution="active" kind="ecapa" eval="strong" >  
  
  <on>  
    <!-- event -->  
  </on>  
  
  <if>  
    <!-- condition -->  
  </if>  
  
  <do>  
    <!-- action -->  
  </do>  
  
  <ifPost>  
    <!-- postcondition -->  
  </ifPost>  
  
  <doAlternative>  
    <!-- alternative/else action -->  
  </doAlternative>  
  
</Rule>
```

Complex Event Processing (1)

■ Event Definition

- Definition of event pattern by event algebra

■ Event Selection

- Defines selection function to select one event from several occurred events (stored in an event instance sequence) of a particular type, e.g. “*first*”, “*last*”
- Crucial for the outcome of a reaction rule, since the events may contain different (context) information, e.g. different message payloads or sensing information

■ Event Consumption

- Defines which events are consumed after the detection of a complex event
- An event may contribute to the detection of several complex events, if it is not consumed
- Distinction in event messaging between “multiple receive” and “single receive”
- Events which can no longer contribute, e.g. are outdated, should be removed

- Separation of these phases is crucial for the outcome of a reaction rule base in the context of complex events

- Declarative configuration of different selection and consumption policies

Example: Atomic Event (Common Base Event)

```
<on>
  <Message mode="outbound" directive="ACL:inform">
    <oid><Ind>conversation123</Ind></oid>
    <protocol><Ind>esb</Ind></protocol>
    <sender><Ind>Organization@lapbichler32</Ind></sender>
    <content>
      <And>
        <Atom>
          <oid><Ind type="cbe:CommonBaseEvent">i000000</Ind></oid>
          <Rel use="value" uri="cbe:creationTime"/>
          <Ind type="owlTime:Year">2007</Ind>
          <Ind type="owlTime:Mont">6</Ind>
          ...
        </Atom>
        <Atom>
          <oid><Ind type="cbe:CommonBaseEvent">i000000</Ind></oid>
          <Rel use="value" uri="cbe:msg"/>
          <Ind type="xsd:String">Hello World</Ind>
        </Atom>
        ...
      </And>
    </content>
  </Message>
</on>
```

Complex Event Algebra



Complex Event Algebra Operators:

Sequence | Disjunction | Xor |
Conjunction | Concurrent | Not | Any |
Aperiodic | Periodic

and

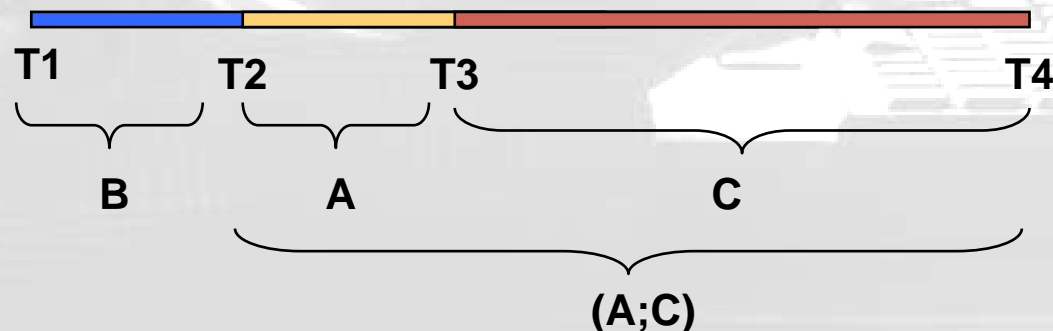
Complex Event Message Patterns

Semantics:

- Interval-based KR Event Calculus semantics (model-theory + proof theory) based on time intervals modeled as fluents

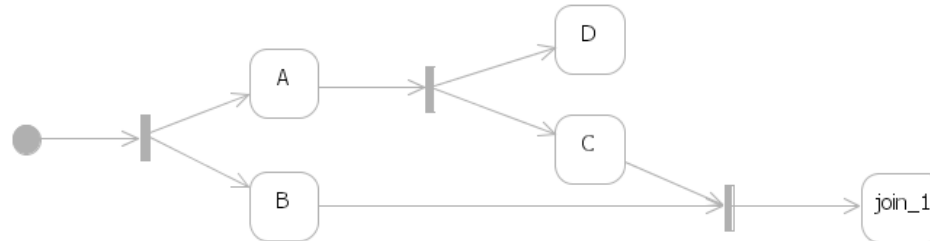
$$I : \overline{T_i} \times \overline{Fl} \mapsto \{true, false\}$$

- Example: $B;(A;C)$ (Sequence)



Complex Event Messaging and Processing

- Semantics a la Petri nets and pi-calculus
- Workflow patterns



```
process_join() :-
    iam(Me),
    init_join(XID,join_1,[c(_),b(_)]),
    fork_a_b(Me,XID).
fork_a_b(Me,XID) :-
    rcvMsg(XID,self,Me,reply,a(1)),
    fork_c_d(Me,XID).
fork_a_b(Me,XID) :-
    rcvMsg(XID,self,Me,reply,b(1)),
    join(Me,XID,join_1,b(1)).
fork_c_d(Me,XID) :-
    rcvMsg(XID,self,Me,reply,c(1)),
    % Tell the join join_1 that a new pattern is ready
    join(Me,XID,join_1,c(1)).

% The following rule is invoked by join once all the inputs are assembled.
join_1(Me,XID,Inputs) :-
    println(["Joined for XID=",XID," with inputs: ",Inputs]).

% Prints
% Joined for XID=agent@hostname001 with inputs [[b,1],[c,1]]
```



RBSLA

Rule Based Service Level Agreement

IT Service Management for electronic Contracts, Policies and SLAs

RBSLA: Rule Based Service Level Agreement

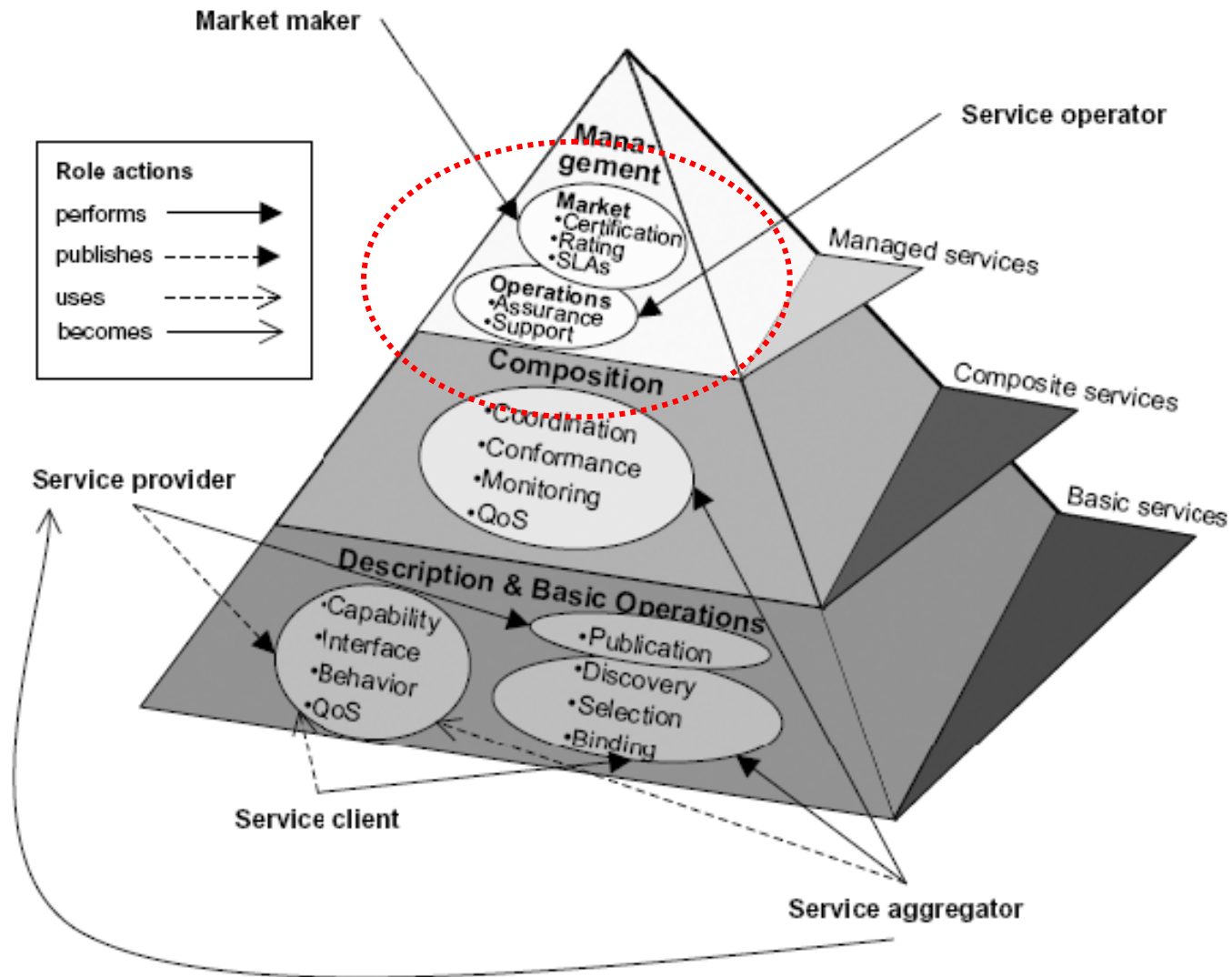
IT Service Management for electronic Contracts, Policies and SLAs

<http://ibis.in.tum.de/projects/rbsla/index.php>



<https://sourceforge.net/projects/rbsla>

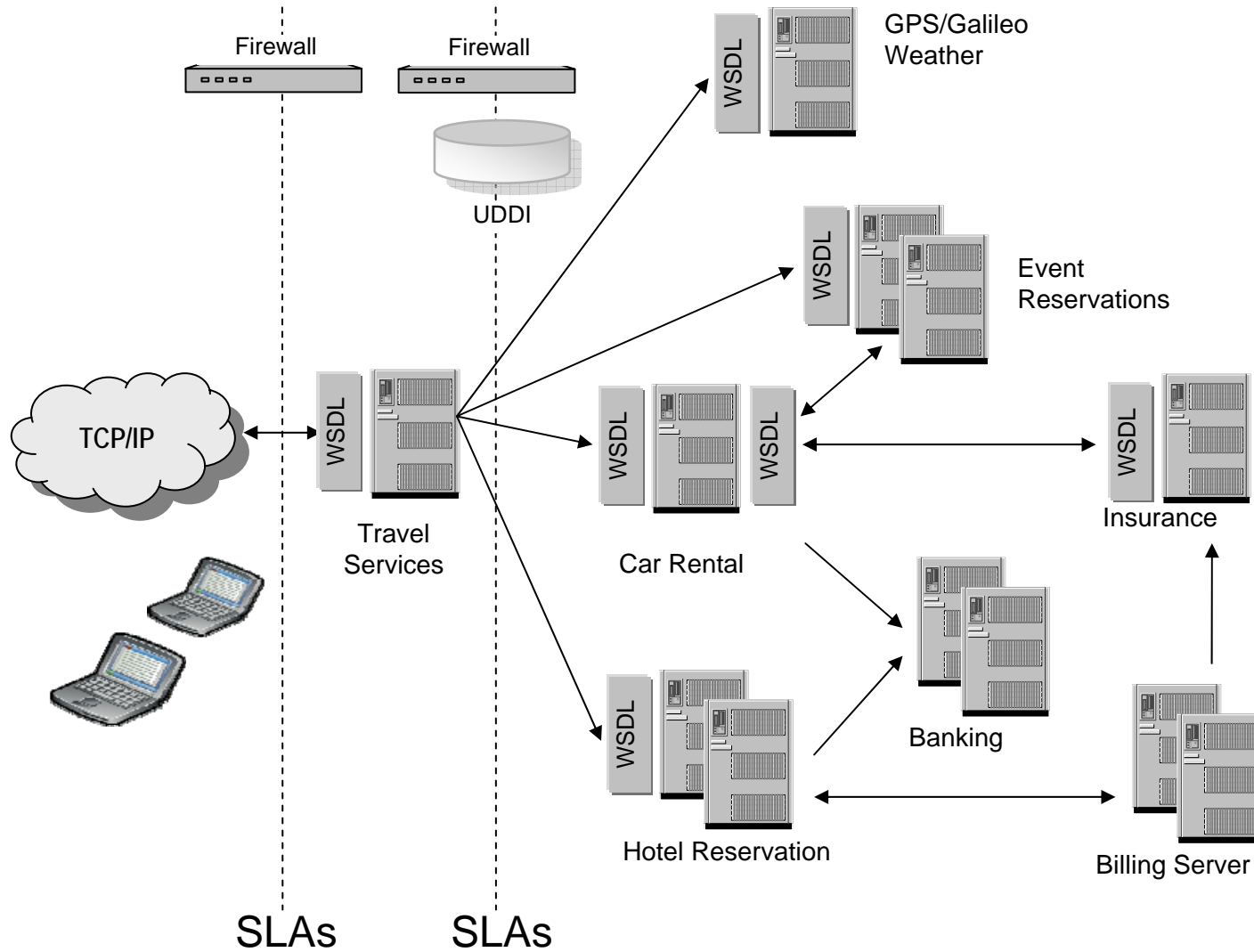
Service Oriented Architecture (SOA)



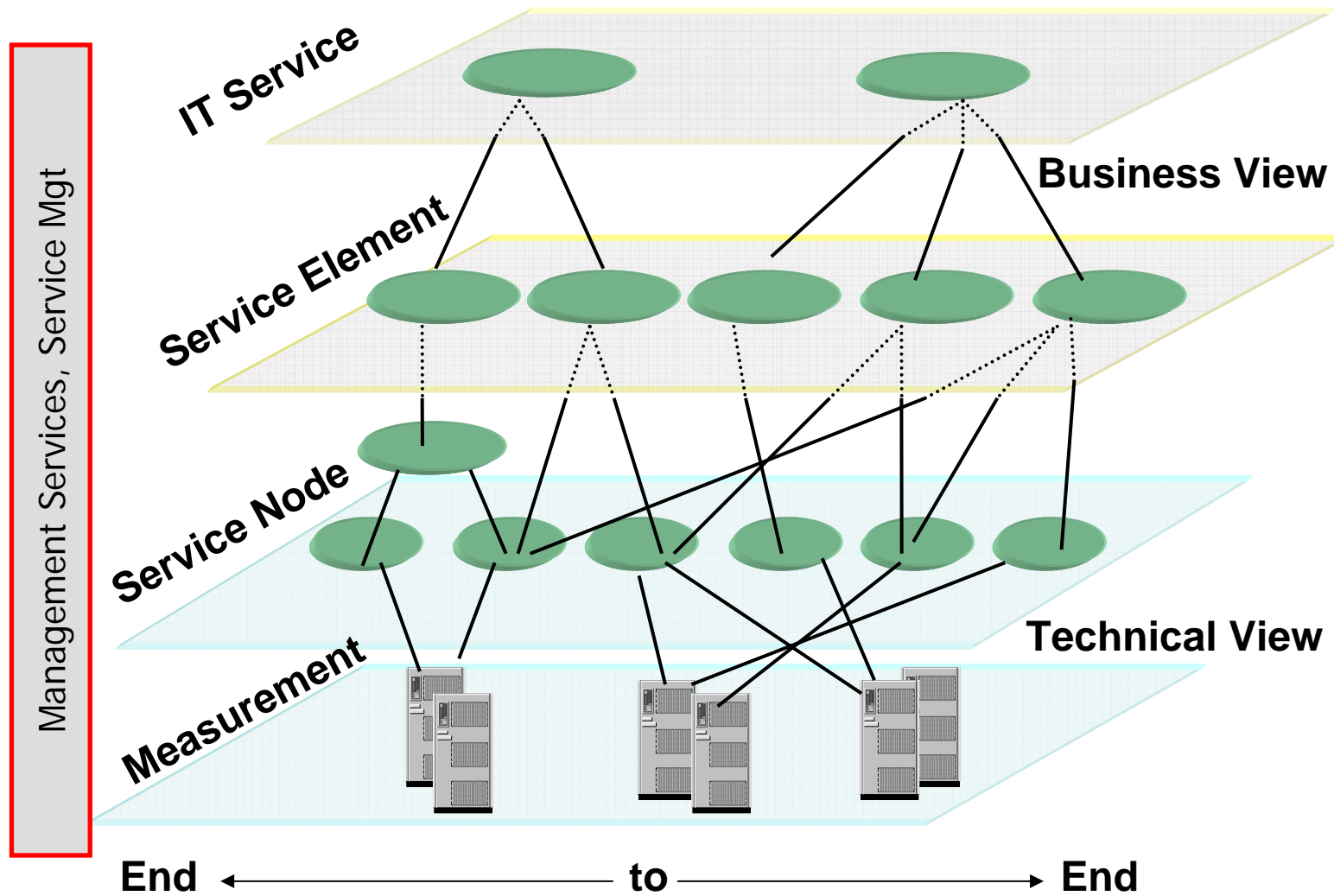
M. P. Papazoglou and D. Georgakopoulos. Service-oriented computing. Communications of the ACM,

46:2528, 2003.

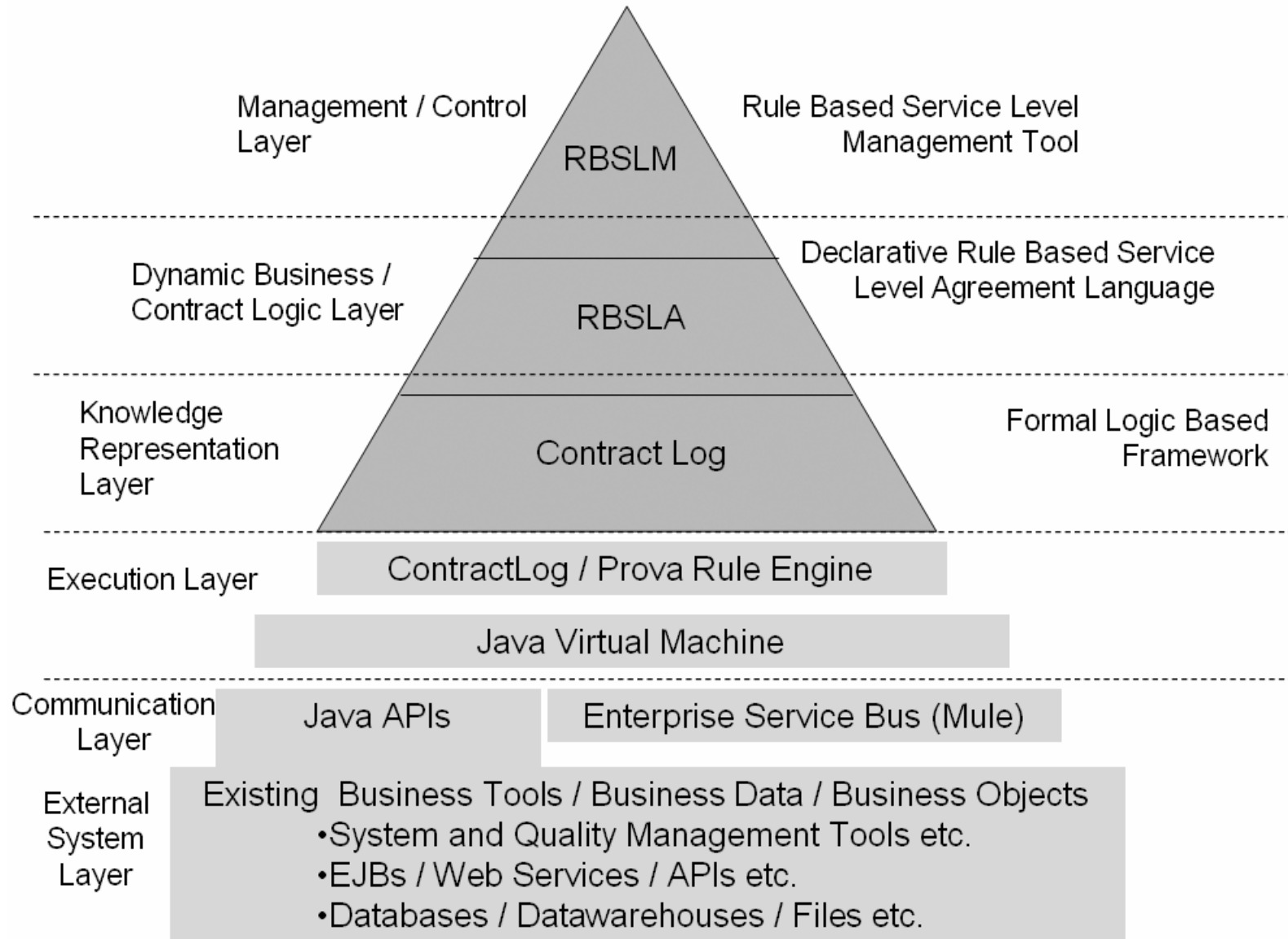
IT Service Supply Chain



IT Service Management (ITSM)



RBSLA Architecture

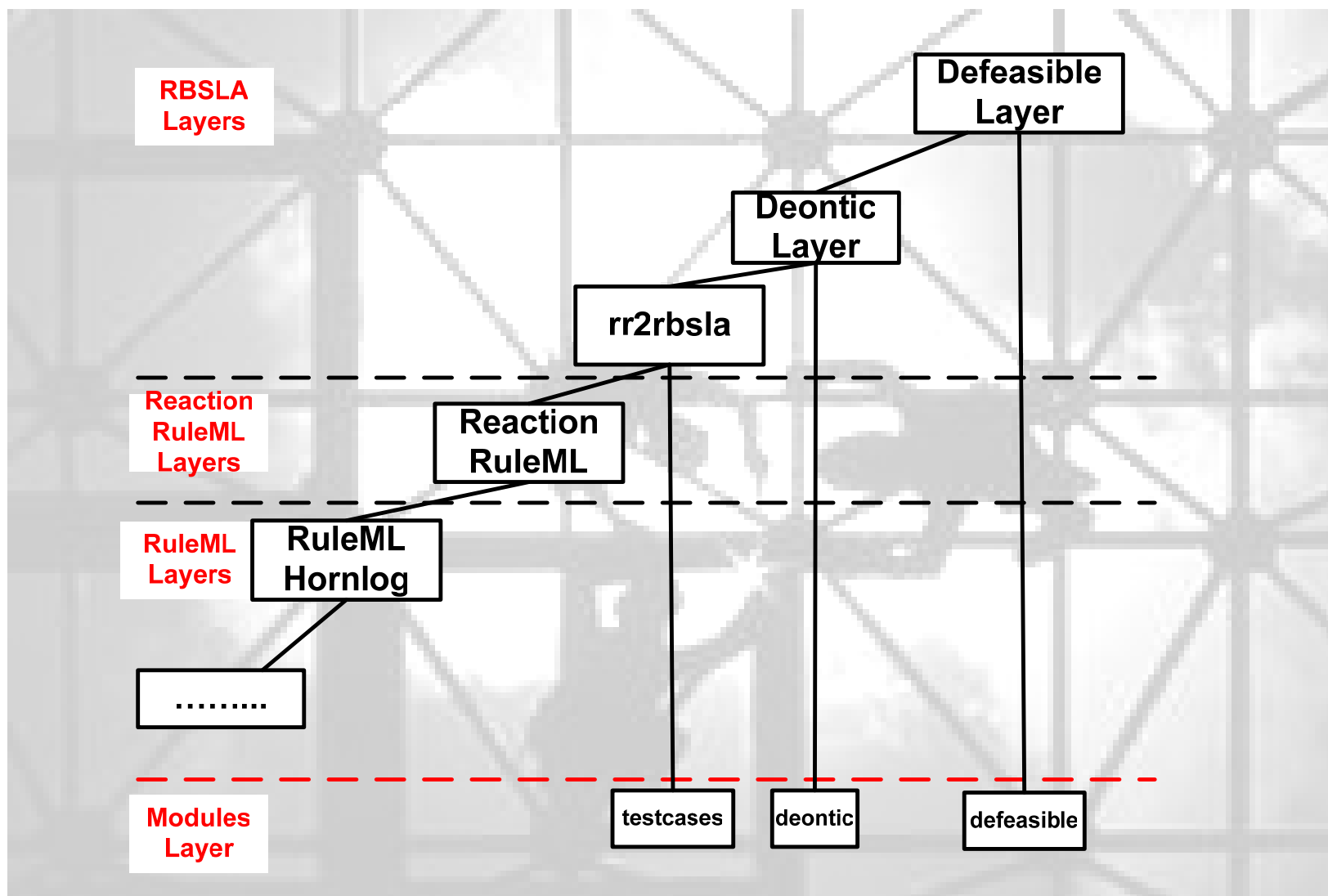


ContractLog KR

Selection of adequate formalisms:

Logic	Application
Extended Logic Programs + Extensions (Hybrid LPs)	Derivation rules, negation, integration of object-oriented code (Java), external databases (SQL)
Event-Condition-Action rules (ECA)	Reaction rules, complex event/action processing
Event Calculus	Temporal reasoning about effects of events / actions
Defeasible logic and Integrity Constraints	Integrity constraints, default rules, exceptions, priorities between rules and rule sets
Deontic logic	Rights and obligations, contract norms
Description logic	Integration of Semantic Web contract vocabularies (RDFS, OWL) and domain-specific meta data
Metadata Annotated Ordered Logic Programs	Metadata annotation of rules, modularization of rule sets, scoped reasoning

RBSLA: Rule Based Service Level Agreement Language



Rule Based Service Level Management (RBSLM)

The screenshot displays the RBSLA Manager application interface. At the top, a browser window shows the URL `http://ibis.in.tum.de`. Below it, there are three main monitoring windows:

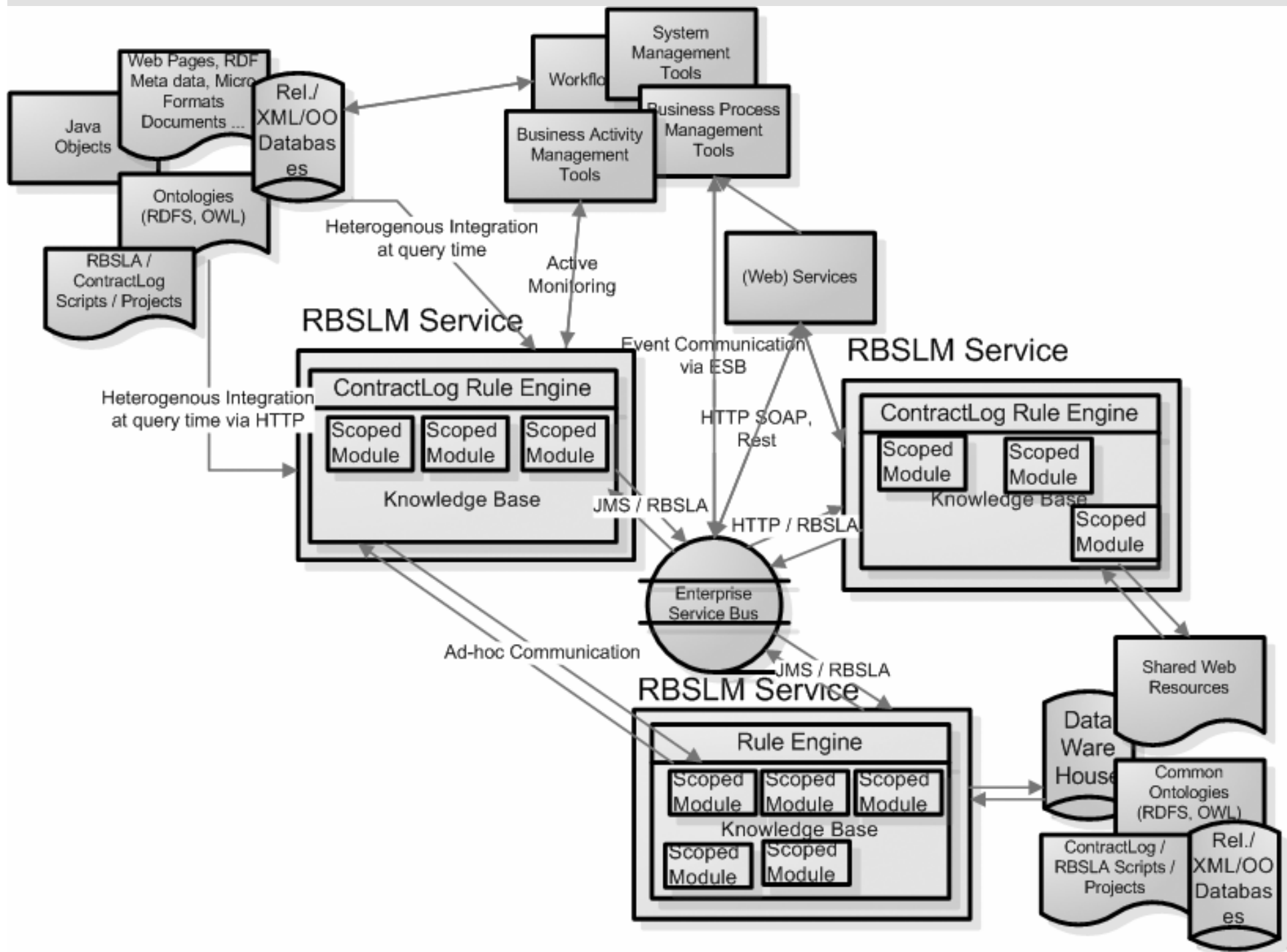
- Response Time:** A line graph showing response time over time, with a green shaded area indicating a target range.
- Webserver avail:** A bar chart showing availability percentages for different servers.
- Traffic Messages in Thousands:** A pie chart showing the distribution of traffic across three servers: Server FRA = 100 (green), Server MUC = 130 (red), and Server PIT = 10 (blue).

The main RBSLA Manager window has a menu bar (File, Edit, Help) and a toolbar with a 'Start Environment' button. Below the toolbar are tabs for 'Rules', 'Queries', 'Tests', and 'Templates'. A 'Add new knowledge' dialog is open, showing a 'Select Rule Template' list with options: Rulebase, Atom, Implies, Equivalent, Entails, Forall, Neg, Reaction, and arg. The dialog is on 'Step 2' and has navigation buttons: '< Previous', 'Next >', 'Cancel', and 'Finish'.

On the right side, a numbered list describes the components:

- 1. Service Dashboard**
 - (Monitoring and Contract Tracking)
- 2. RBSLA Editor / Contract Manager**
 - (Contract Mgt. and Authoring)

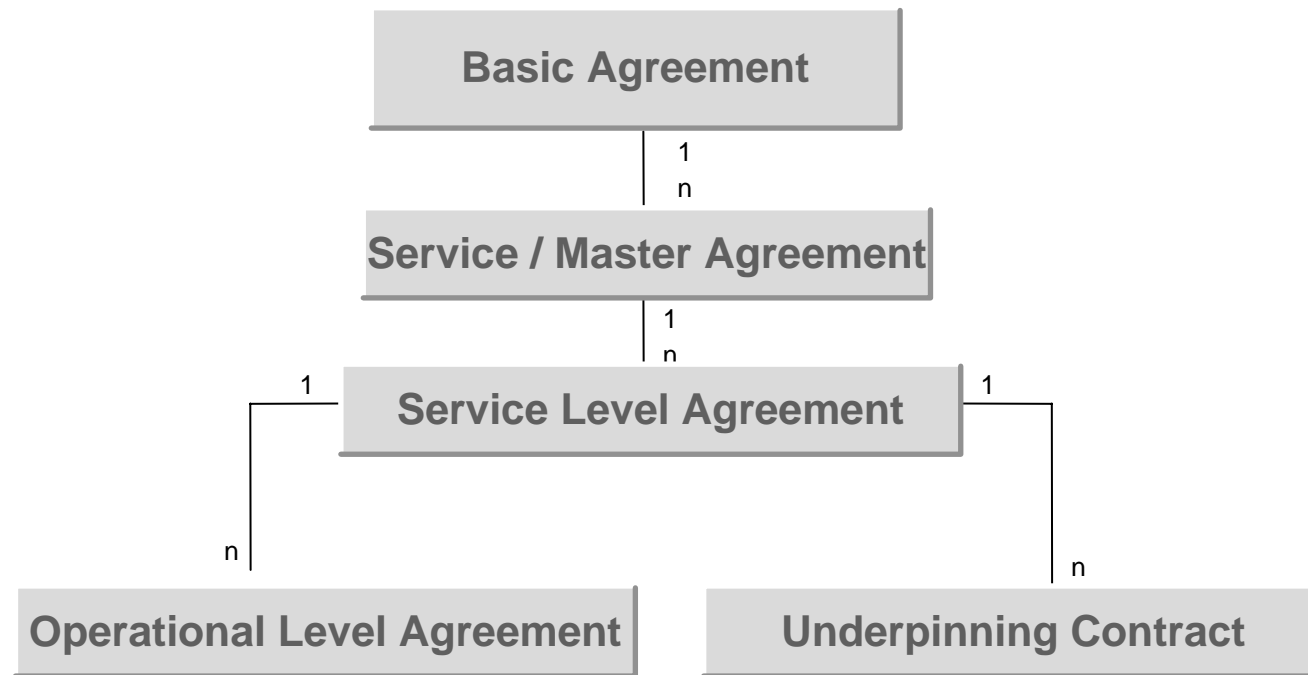
Arrows point from the list items to the corresponding windows in the screenshot: an arrow from item 1 points to the traffic pie chart, and an arrow from item 2 points to the RBSLA Manager window.



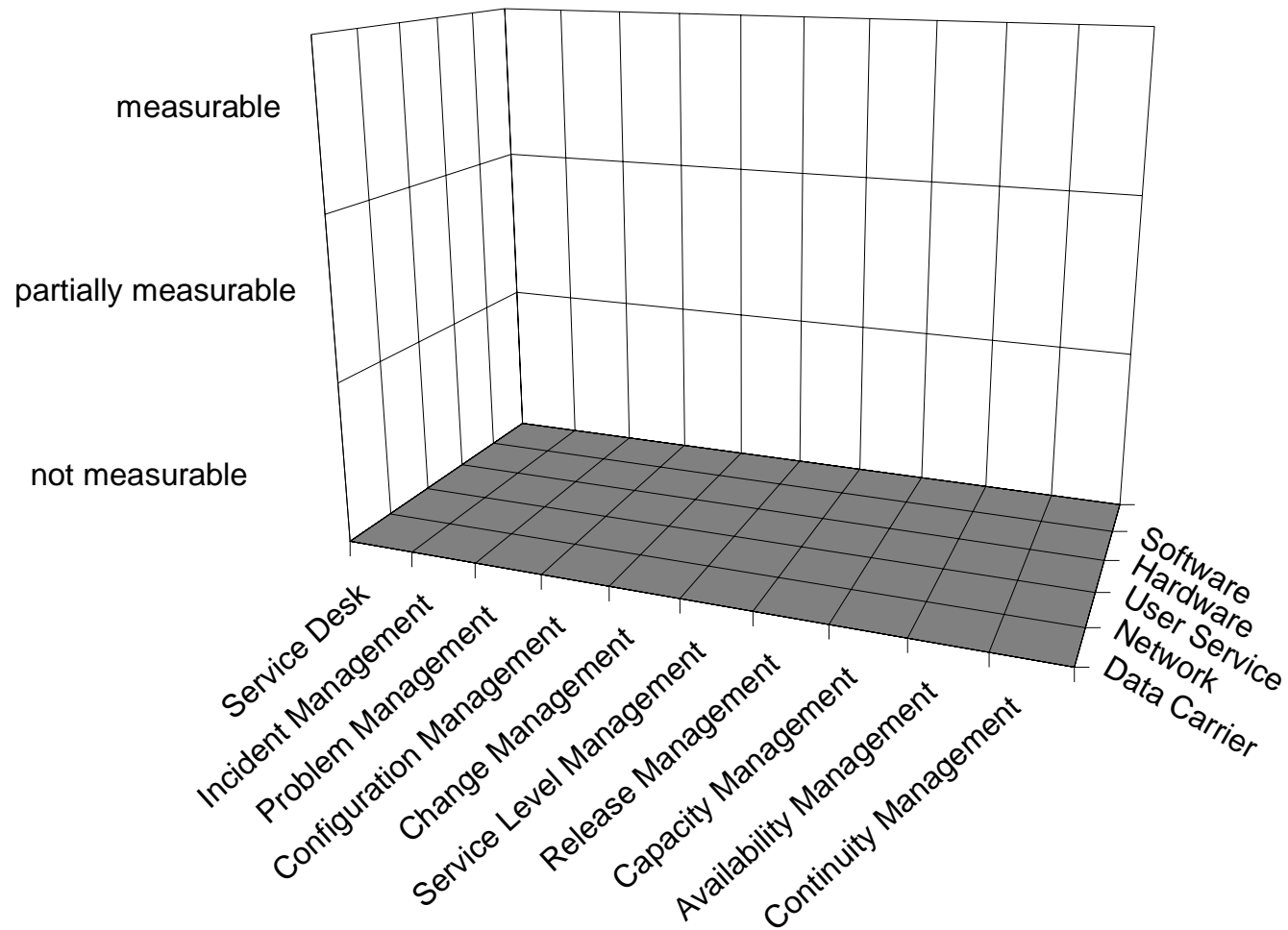
RBSLA Industry Studies, Use Cases, Projects

Industry Studies:

- Analysis of IT Service Management standards (e.g. ITIL, BS15000)
- Three dozen IT service providers from small-and medium-sized enterprises to big companies
- Analysis of IT Service Technologies (e.g. Server Virt., Grid Comp., Autonomous Comp.)
- Analysis of commercial ITSM and BAM tools (e.g. IBM Tivoli, HP Open View, Oracle 10, BMC Patrol)
- Analysis of representation languages (WSLA, WS-Agreement, WS-Policy, OWL-S, CBE)
- 50 state-of-the-art SLAs currently used throughout the industry in the areas of IT outsourcing, Application Service Provisioning (ASP), Hardware Hosting, Service Suppliers and many other



Multi-layered Categorization Metrics / SLOs / KPIs



Example: ITIL (IT Infrastructure Library)

Description	Position	Task
Service Desk	Function	Group of specialists, inquiry -, treatment of disturbances
Incident Management	Process	Support user, problem acceptance, assistance, monitoring service level
Problem Management	Process	Treatment of losses, cause identifying, recommendations at Change Mgmt., improvement of productive resources use
Configuration Management	Process	Process control of the inventory (components hard -, software....)
Change Management	Process	Change process
SLM	Process	Formulate SLA
Release Management	Process	Storage of authorized software, release in productive environment, distribution to remote bases, implementation to start-up
Capacity Management	Process	Correct and cost-related-justifiable IT capacity provision analysis, prognosis; Capacity plans
Availability Management	Process	Optimization IT resources use, foreseeing and calculation of losses, safety guidelines monitoring SLAs, Security, Serviceability, Reliability, Maintainability, Resilience
Service-Continuity-Management	Process	Re-establishment of services, replacement in case of failure
Financial Management	Process	Process investment strategy, definition that-achievement-aims, those-brought achievement to measurement

Example: ITIL Service Metrics / KPIs

ITIL Process	Service Metrics
Service Desk	Customer satisfaction with the Help Desk
Incident Management	Time between loss and replacement
Problem Management	Number of repeated disturbances
Configuration Management	Time between adding configuration items to Configuration Management Data Base (CMDB)
Change Management	Number of untreated changes
Service-Level Management	Number of SLAs
Release Management	Time between releases
Capacity Management	Completion of the capacity plan at a fixed time
Availability Management	Completion of the availability plan at a fixed time
IT-Service-Continuity-Management	Completion of the contingency plan at a fixed time
Financial Management	Cost overview to the deadline

Contributions to IT Service Management (1)

- Application of declarative rules and rule-based events in SLM and BAM
 - Flexible and dynamic rule management
 - Complex SLA rules (not just parameters with thresholds)
 - Reduced Costs for Modification of contract logic
 - Shorter development cycles
 - Simplified decentralized management and reuse of SLA rules for different service offerings in distributed environments
- Applications in:
 - IT Service Monitoring and Enforcement Phase
 - Discovery and Negotiation Phase (e.g. Semantic Web Services)
 - Analysis Phase (based on SLA QoS data)
- Selection of adequate KR concepts

Contributions to IT Service Management (2)

- Integration and extension of different advanced logic concepts
 - Semantics, Expressiveness, Scalability, Flexibility
- Integration into standardization initiatives
- Integration and Interoperation with Semantic Web Standards
 - RDFS/OWL: Contract ontologies
 - Rule Based Service Level Agreement (RBSLA) Language
 - XML Representation Languages, e.g. Common Base Event
 - Straightforward Integration into Internet Technologies
(e.g. WSDL extension with reference to RBSLA; BPEL message flow)
- Declarative Rule Based Programming
 - Clear logic based semantics
 - Syntactic extensions via integration of Semantic Web Vocabularies, Enterprise Beans
- Efficient, scalable and stable tools:
 - Enterprise Service Bus, Rule Inference Services, flexible Engineering UIs and Service Dashboard UIs