Rule Based IT Service Level Management

30th April 2007

IBIS, TU München
+49 89 289 17504
http://ibis.in.tum.de

RBSLA: Rule Based Service Level Agreement

- IT Service Level Management and Service Level Agreements
- SLA Tools and Languages
- Rule Based Service Level Management
- ContractLog KR Framework
- RBSLA Markup Language
- RBSLM Tool
- Industry Study on SLAs and Evaluation
- Contributions

Adrian Paschke (TUM)

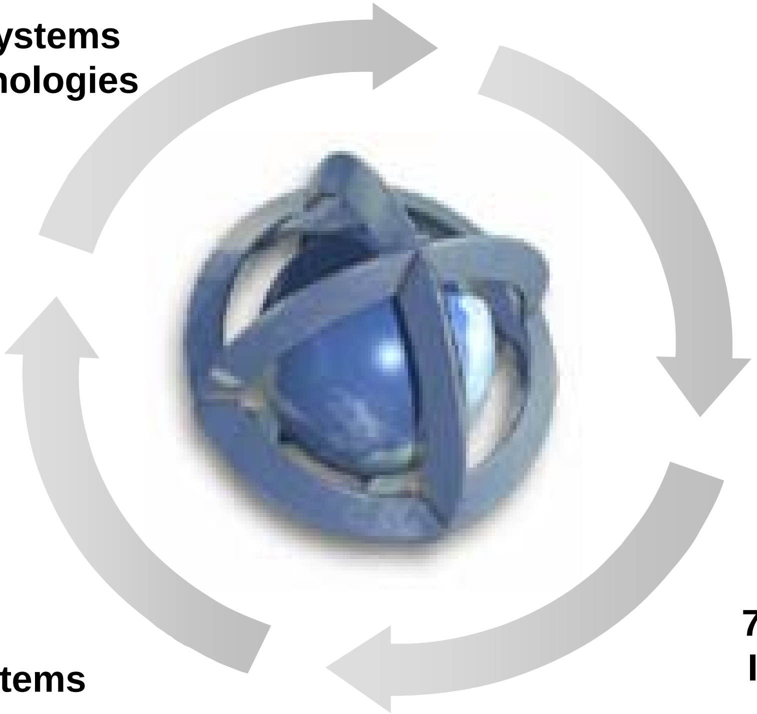
Colloquium at NRC-IIT, Fredericton, Canada, April 2007

Research Areas

4. Negotiation and Coordination Mechanisms with decentralized Decision Makers

3. Heterogenous Information Systems Modern XML and Internet Technologies

5. Semantic Web Technologies and Ontology based Content Management Systems



2. Supply Chain Management Supply Chain Tracking and Event Management

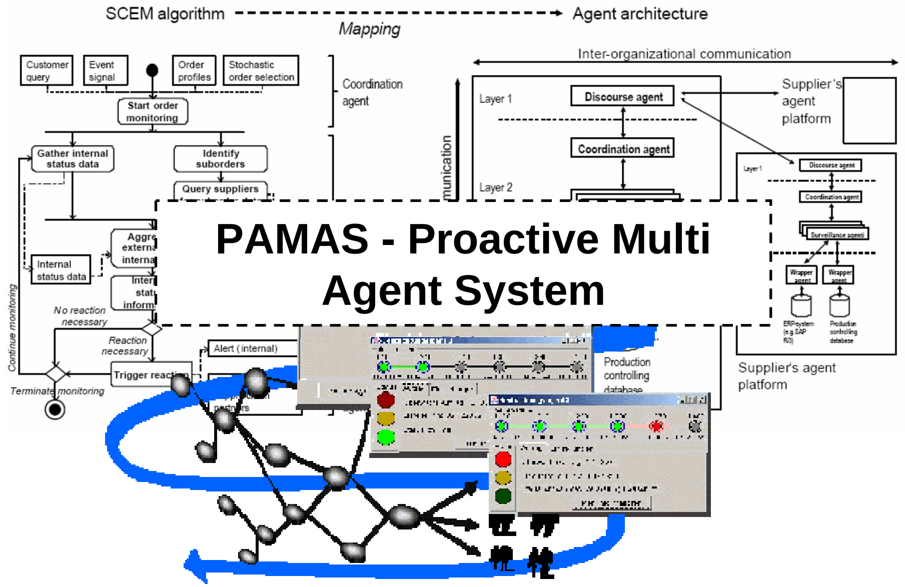
6. Web Services Computing

1. Multi-Agent Systems

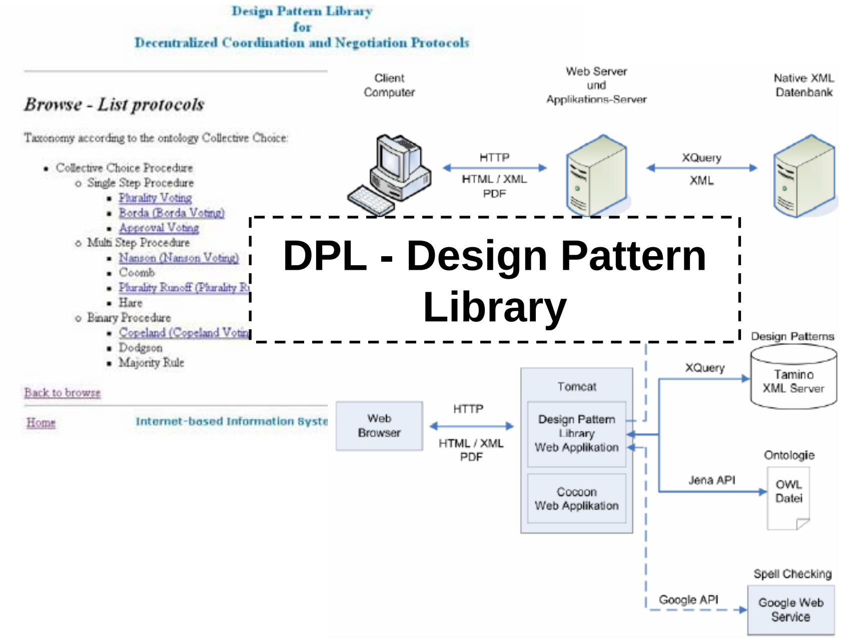
7. IT Service Management and IT Service Level Management

8. Rule Based Systems and Rule Standards

Selected Projects



PAMAS - Proactive Multi Agent System



DPL - Design Pattern Library

Design Pattern Library for Decentralized Coordination and Negotiation Protocols

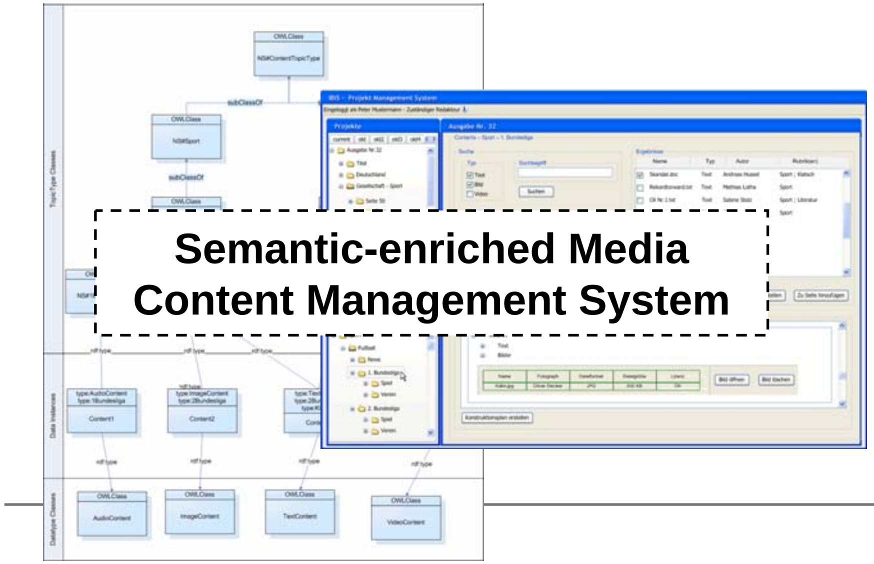
Browse - List protocols

Taxonomy according to the ontology Collective Choice

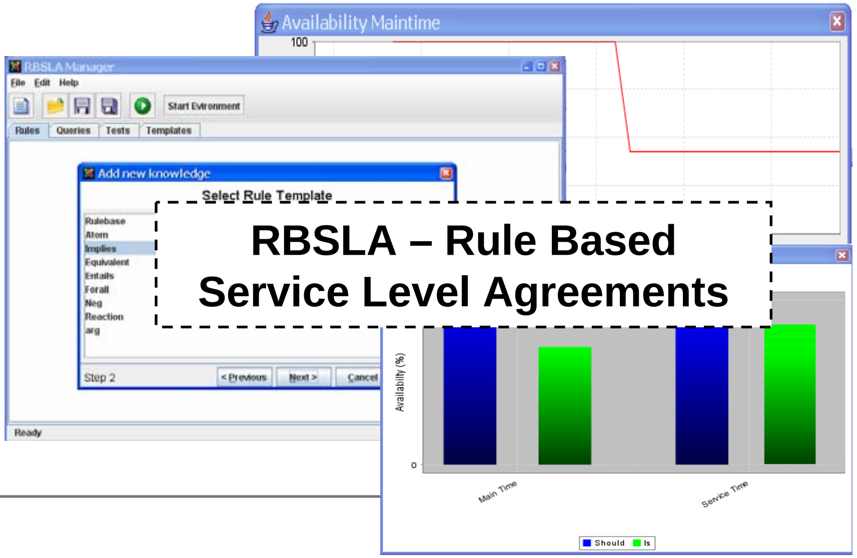
- Collective Choice Procedure
  - Single Step Procedure
    - Plurality Voting
    - Borda (Borda Voting)
    - Approval Voting
  - Multi Step Procedure
    - Nanson (Nanson Voting)
    - Coomb
    - Plurality Runoff (Plurality R
    - Hare
  - Binary Procedure
    - Copeland (Copeland Voting
    - Dodgson
    - Majority Rule

Back to browse

Home   Internet-based Information Syste



Semantic-enriched Media Content Management System



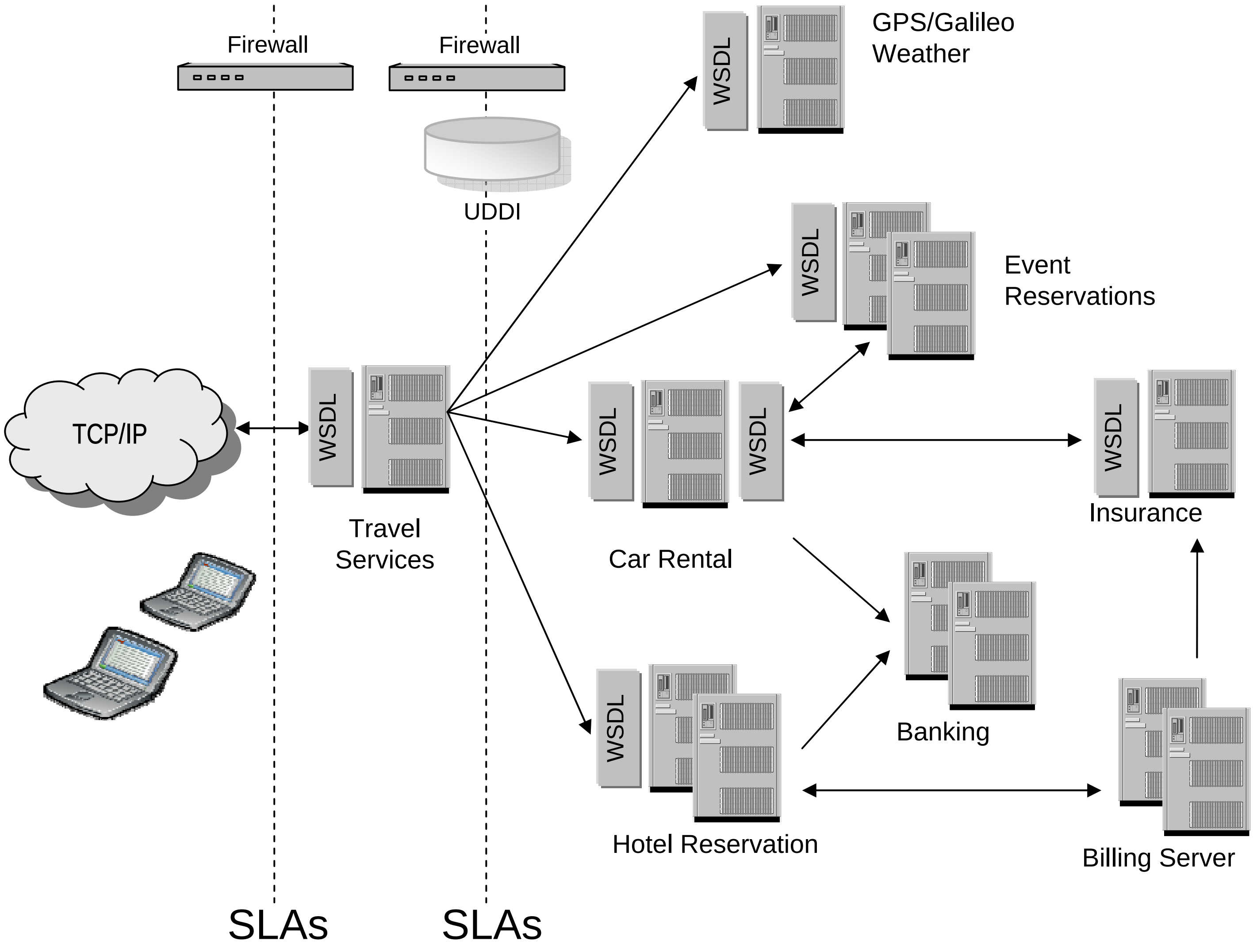
RBSLA – Rule Based Service Level Agreements

RBSLA

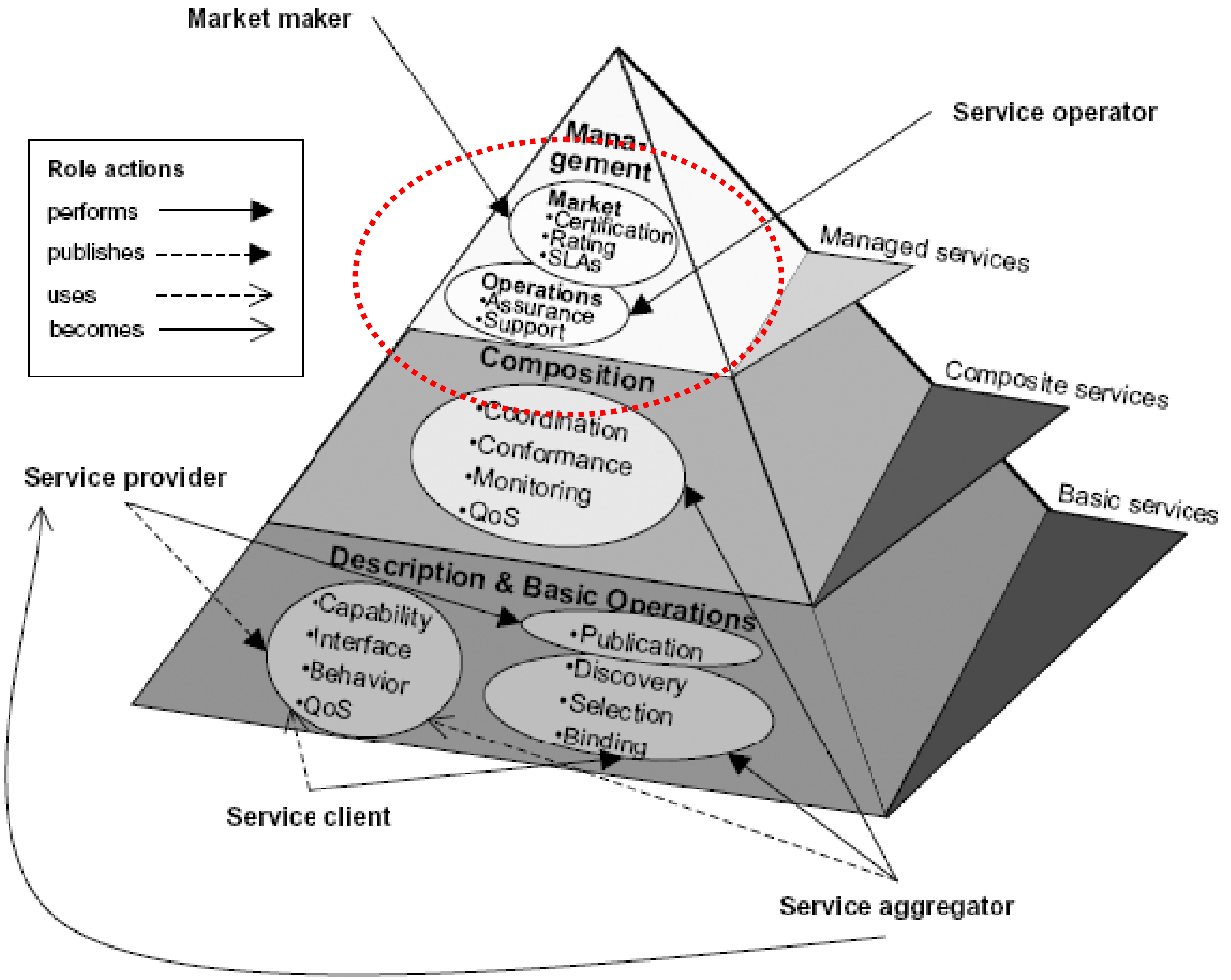
Rule Based Service Level Agreement

IT Service Management for electronic Contracts, Policies and SLAs

IT Service Supply Chain

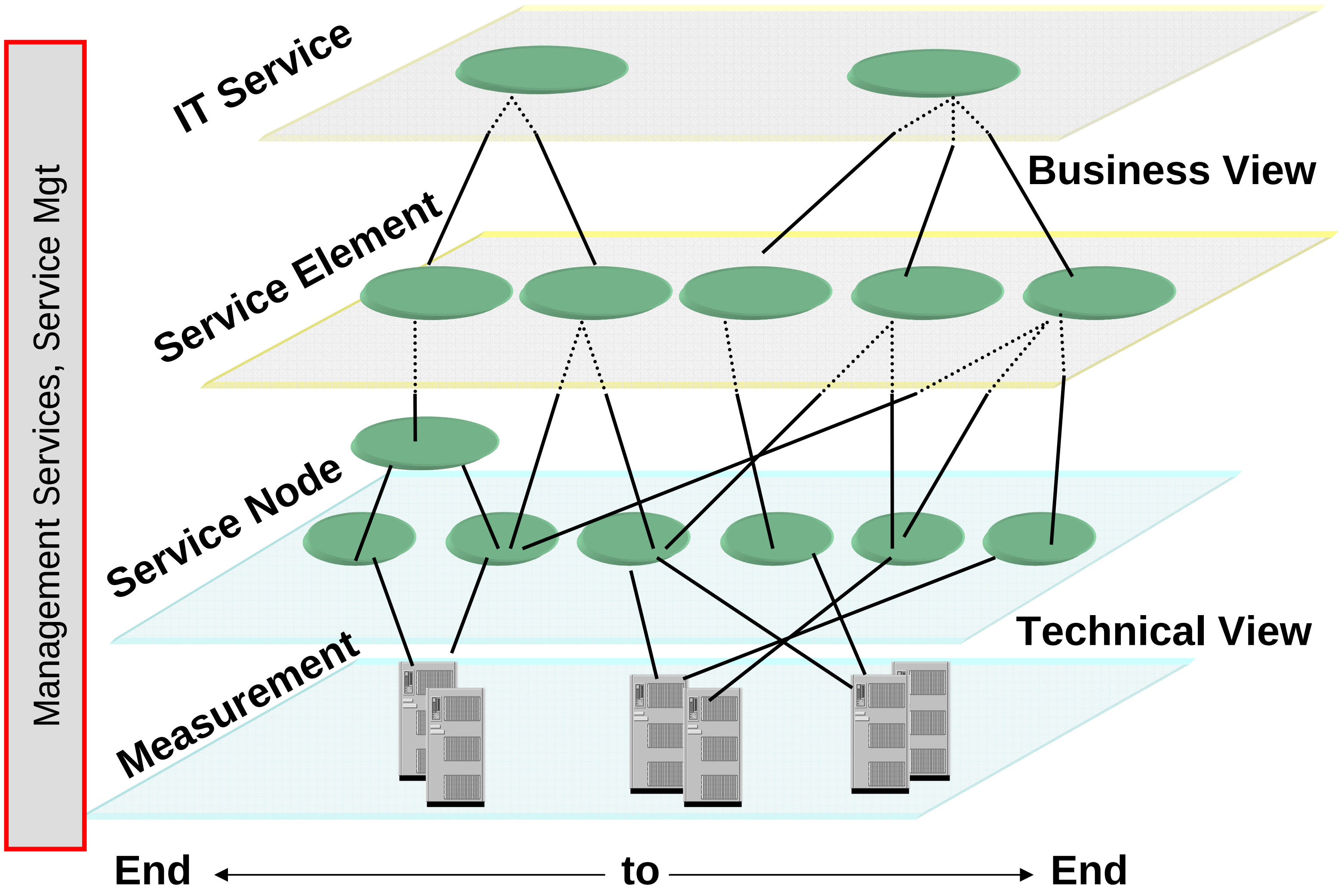


Service Oriented Architecture (SOA)



M. P. Papazoglou and D. Georgakopoulos. Service-oriented computing. Communications of the ACM, 46:2528, 2003.

IT Service Management (ITSM)

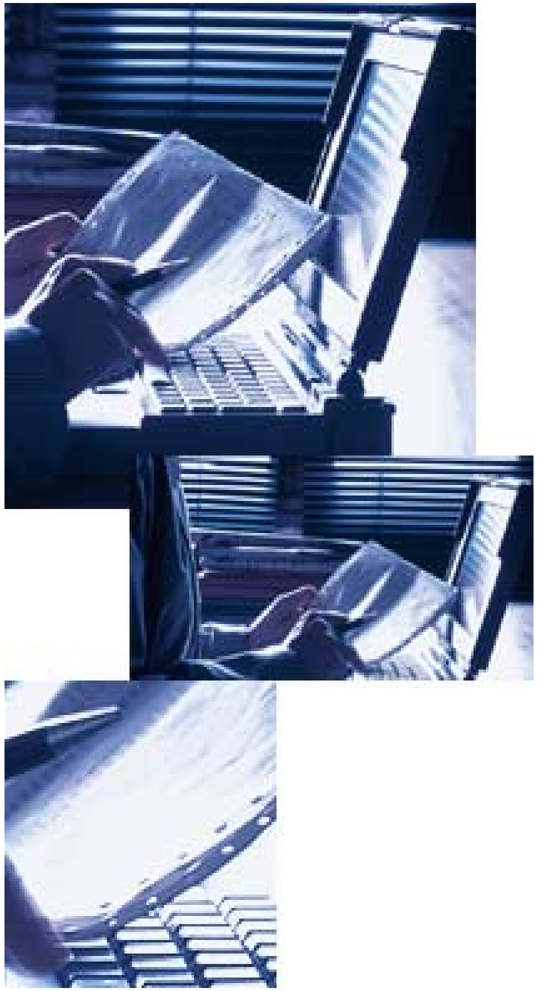


Service Level Agreement

A SLA contract is a document that describes the performance criteria a provider promises to meet while delivering a service.

It typically also sets out the rights and obligations each person has in a particular context or situation, the remedial actions to be taken and any penalties that will take effect if the performance falls below the promised standard.

Service Level Agreement (SLA)

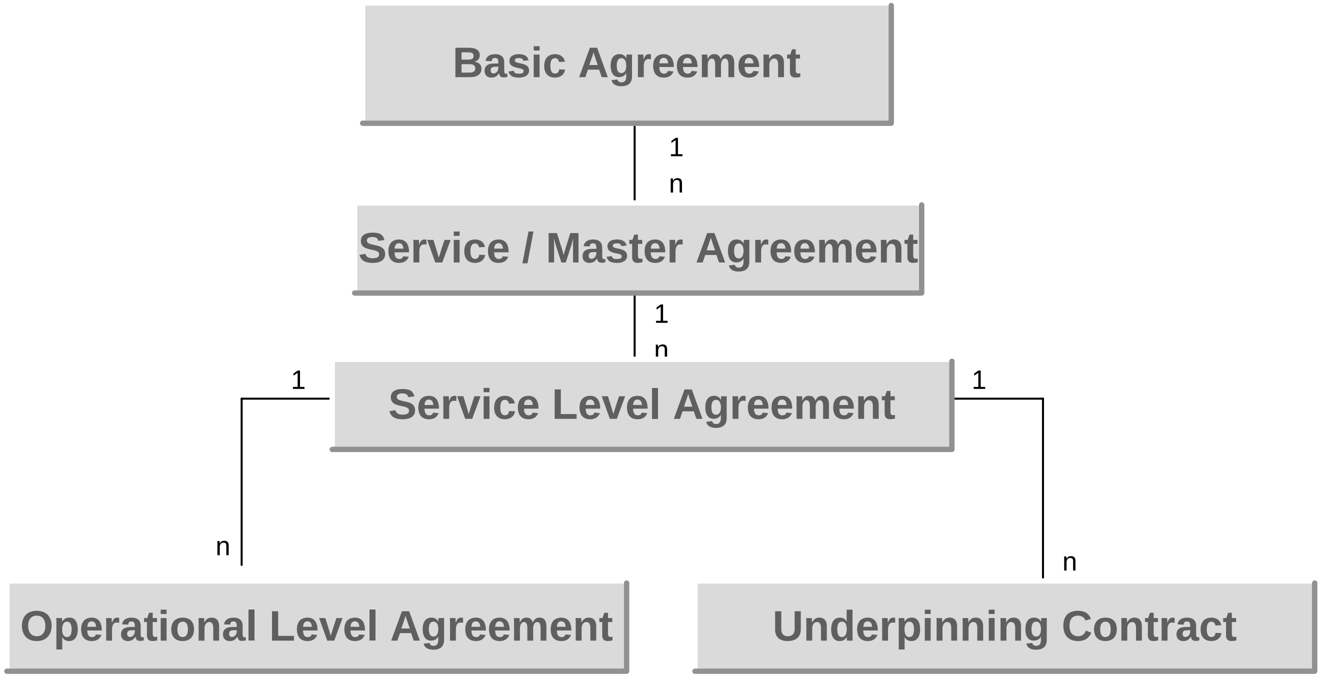


SLA Main Objectives:

- Verifiable, objective agreements
- Know risk distribution
- Trust and reduction of opportunistic behaviour
- Fixed rights and obligations
- Support of short and long term planning and further SLM processes
- Decision Support: Quality signal
  (e.g. assessment of new market participants)
- ...

➔ SLAs are an essential component of the legal contract between a service consumer and the provider.

Unitized, Modular Contract Structure



Basic Agreement

1

n

Service / Master Agreement

1

n

Service Level Agreement

1

n

Operational Level Agreement

1

n

Underpinning Contract

SLA Rule Classes

- Dependent Rules: (defining dependent Service Levels and SLOs)
  “*If the **average availability** falls below 98%*
  *then the **mean time to repair** must be less than 10 min.*”

- Graded Rules: (defining situational workflow-like rules)
  - Monitoring Schedules

Schedule	Time	Availability	Response Time
Prime	8 -18	99%	4 sec.
Standard	18-8	95%	10 sec.
Maintenance	0-4 *	30%	-

  - Escalation Levels with Role Model

Level	Role	Time-to-Repair	Rights / Obligations
1	Process Manager	10 Min.	Start / Stop Service
2	Chief Quality Manager	Max. Time-to-Repair	Change Service Levels
3	Control Committee	-	All rights

SLA Rule Classes (2)

- Dynamic Rules (exceptional rules which apply in special situations)

  “*There might be an **unscheduled period of time** which will be **triggered by the customer**.*
  *During this period **bandwidth must be doubled**.*”

- Normative Rules with Violations and Exceptions

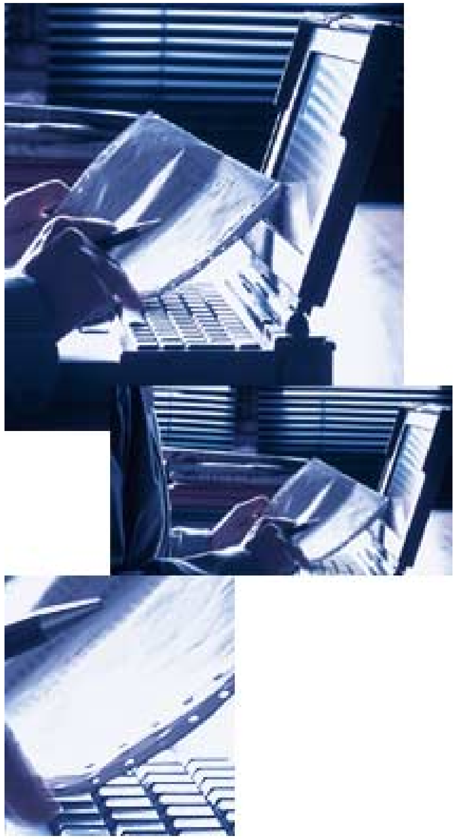
  “*The provider is **obliged** to repair an unavailable service in $t_{time\text{-}to\text{-}repair}$.*
  *If she fails to do so (**violation**) the customer is **permitted** to cancel the contract.*”

SLA Metrics

***SLA metrics** are used to measure and manage performance compliance to SLA commitments. They play a key role in metering, accounting and reporting, and provide data for further analysis and refinement of SLAs in the analysis phase.*

- Examples (taken from real contracts):
  - For purposes of this SLA, 100% **accessibility** shall mean that the DynDNS Nameserver Infrastructure shall not fail to respond to DNS queries for more than five (5) consecutive minutes.
  - Intermedia.NET guarantees 99.5% **Service Availability**, measured on a calendar-month basis. Service Availability is defined as the ability of a user within an organization to:
    - retrieve messages from the Exchange server and
    - send and receive messages via the Exchange server
  - In the event of Intermedia.NET server **hardware failure**, Intermedia.NET shall replace or repair hardware within eight hours of Intermedia.NET's determination that the hardware has failed. Such hardware failure, and repair and replacement of the hardware and the associated downtime shall not affect the Service Availability calculation.

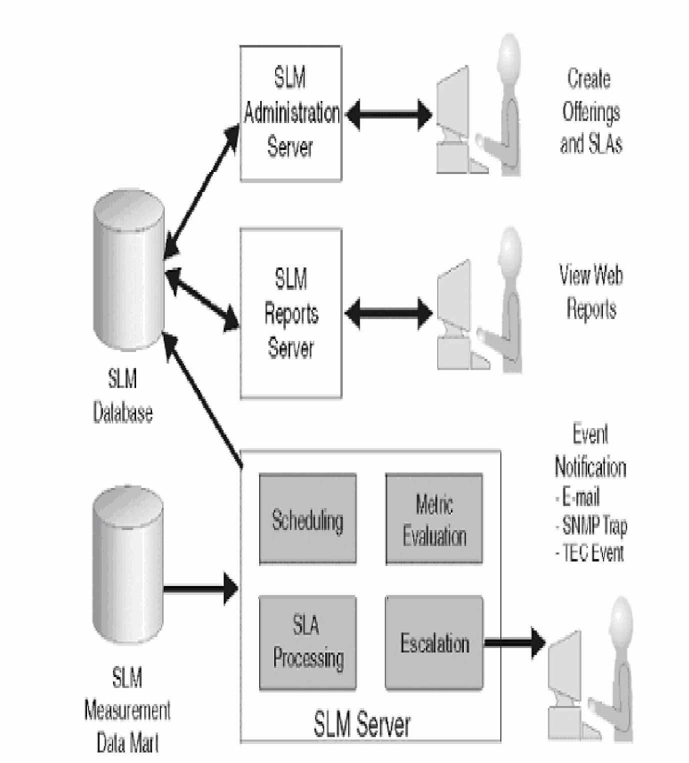
Problem Domain



- Many contracts
- Distributed environment
- Individual rules
- Dynamic changes and adaptations
- Different roles involved during Life Cycle

➔ **Service Level Management Tool**
- ➔ **Automated monitoring and enforcement of SLA contracts**
- ➔ **Flexible and adaptable**

SLA Management Tools



- IBM Tivoli SLM, SLM Express BMC, ICS SMC, Amberpoint, Oblicore …
- Commercial tools mainly focus on IT systems/resources
  - Often simple extensions to system and network management tools
  - Contract/Business logic is buried in the code or database tiers
  - Contract rules (logic) are adjusted by simple parameters
  - Control flow must be completely implemented
  - Missing link between technical view and SLA view

WS-Agreement

- XML based Specification Language for Web Service Agreements
- Only Syntactical Markup Representation
  - No Semantics
  - Non-standard procedural interpreter needed (e.g. Cremona)
- Example:

```
<wsag:ServiceLevelObjective xsi:type="sdtc:OpType">
  <Or> <SDT>numberOfCPUsHigh</SDT> <SDT>numberOfCPUsLow</SDT> </Or>
    </wsag:ServiceLevelObjective>
     <wsag:BusinessValueList>
      <wsag:Preference>
         <wsag:ServiceTermReference>numberOfCPUsHigh<wsag:ServiceTermReference>
         <wsag:Utility>0.8</wsag:Utility>
         <wsag:ServiceTermReference>numberOfCPUsLow </wsag:ServiceTermReference>
         <wsag:Utility>0.5</wsag:Utility>
      </wsag:Preference>
</wsag:BusinessValueList>

<wsag:ServiceDescriptionTerm wsag:Name="numberOfCPUsHigh"
  wsag:ServiceName="ComputeJob1">
   <job:numberOfCPUs>32</job:numberOfCPUs>
</wsag:ServiceDescriptionTerm>
```

No Semantic Interpretation

Only Syntax (XML Markup)

Web Service Level Agreement Language (WSLA)

IBM Web Service Level Agreement Sprache (WSLA) :

IF "TransactionRate < 10000" THEN "AverageResponseTime < 0.5"
IF "AverageResponseTime < 0.5" THEN …

```
<Expression>
   <And>
         <Implies>
               <Expression>
                              <Predicate xsi:type="Less">
                                             <SLAParameter>TransactionRate</SLAParameter>
                                             <Value>10000</Value>
                              </Predicate>
               </Expression>
               <Expression>
                              <Predicate xsi:type="Less">
                                             <SLAParameter>AverageResponseTime</SLAParameter>
                                             <Value>0.5</Value>
                              </Predicate>
               </Expression>
         </Implies>
          <Implies>
               <Expression>
               …
          </Implies>
      </And>
</Expression>
```

Interpretation with procedural interpreter

- No variables
- No global rules, no rule chaining
- Only nested material truth implication
- No dynamic language extensions; need reimplementation of procedural WSLA-Interpreter

➔ no declarative programing ; only syntactical specification

RBSLA: Rule-based Service Level Agreements

1. Representation of SLA rules with logic programming
   - Formalization of SLA specifications as logic programs
   - Automated execution by inference engine

2. Compact declarative representation of rules
   - Clear semantics
   - Global rules which might apply in several contexts
   - Separation of contract rules from the application code
   - Extensibility of the rule base (without changing the interpreter)

3. Efficient, generic interpreters for automated rule chaining

4. Automated conflict detection of rule conflicts
   - Traceable and verifiable results
   - Integrity constraints are possible
   - Automated conflict resolution by rule prioritization

Example

- Bonus-Malus System

Quality of Service (QoS)	Average Availability (quantitative)	Bonus/Malus Discount
High	100 %	+ 5%
Normal	98-100 %	+ 0%
Low	<98 %	- 5%
Below average	<95 %	1000 $ penalty

If average availability is 100 % then QoS is high.

Body		
Predicate	Complex Term	Constant
=	availability(Service)	100%

→

Head		
Predicate	Variable	Constant
qos	Service	high

If QoS is high then provide a bonus of 5% on the base price.

Body		
Predicate	Variable	Constant
qos	Service	high

→

Head		
Predicate	Variable	Constant
discount	Service	5%

Advantages of Logic Programming for SLM

1. Compact declarative knowledge representation of rules by
   - Global validity with a scope (module)
   - Separation of contract rules and application code
   - Simple extension of the rule base (without changing the interpreter)
2. Efficient, generic interpreter (LP inference engines) for automated rule changing and rule derivation
3. Automated conflict resolution
   - Traceable and verifiable rule sets
   - Integrity constraints are possible
   - Automated conflict resolution (e.g. by rule prioritization)

Compact declarative knowledge representation

Logic Programing

```
discount(Service, 5%) :- qos(Service,high).
discount(Service, -5%) :- qos(Service,low).
qos(Service,high):- availability(Service) = 1.
qos(Service,low):- availability(Service) < 0,98.
```

Queries

discount(Service,X)?	All discounts for all services
discount(s1,X)?	Discount for service „s1“
discount(s1,5%)?	service „s1“ → discount 5%?
discount(Service,5%)?	All services with discount 5%
qos(Service,Y)?	Service level of all services?
qos(s1,Y)?	Service of service “s1”?

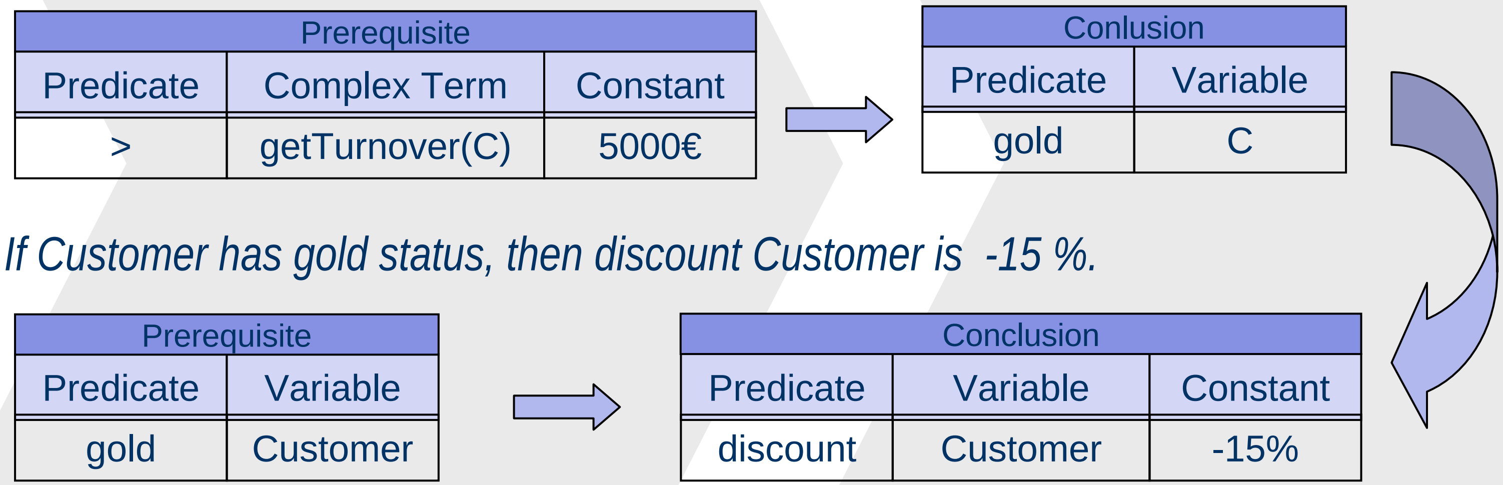
…

Procedural Programing

```
boolean getsDiscount(Service s, int value)  {
   if (getAvailability(s)==1) && (value==1) return true;
   else if (getAvailability(s)<0,98) && (value<0,98) return true;
   else return false;
}
…
Service getService(int value) {
    for (int i=0;i<getAllServices();i++) {
       Service s = getService(i);
       if (getAvailability(s)==1) && (value==1) return s;
       else if (getAvailability(s)<0,98) && (value < 0,98) return s;
      else return null;
  }
}
…
int getDiscount(Service s) {
      if (getAvailability(s)==1) return 5;
      else if (getAvailability(s<0,98) return -5;
      else return 0;
}
…
```

Simple Extension and Maintenance

- Adding / Changing new rules and rule sets (modules) to SLA specifications
- Without extensions of the inference engine (!)



If turnover Customer > 5000€, then Customer has gold status

Prerequisite		
Predicate	Complex Term	Constant
>	getTurnover(C)	5000€

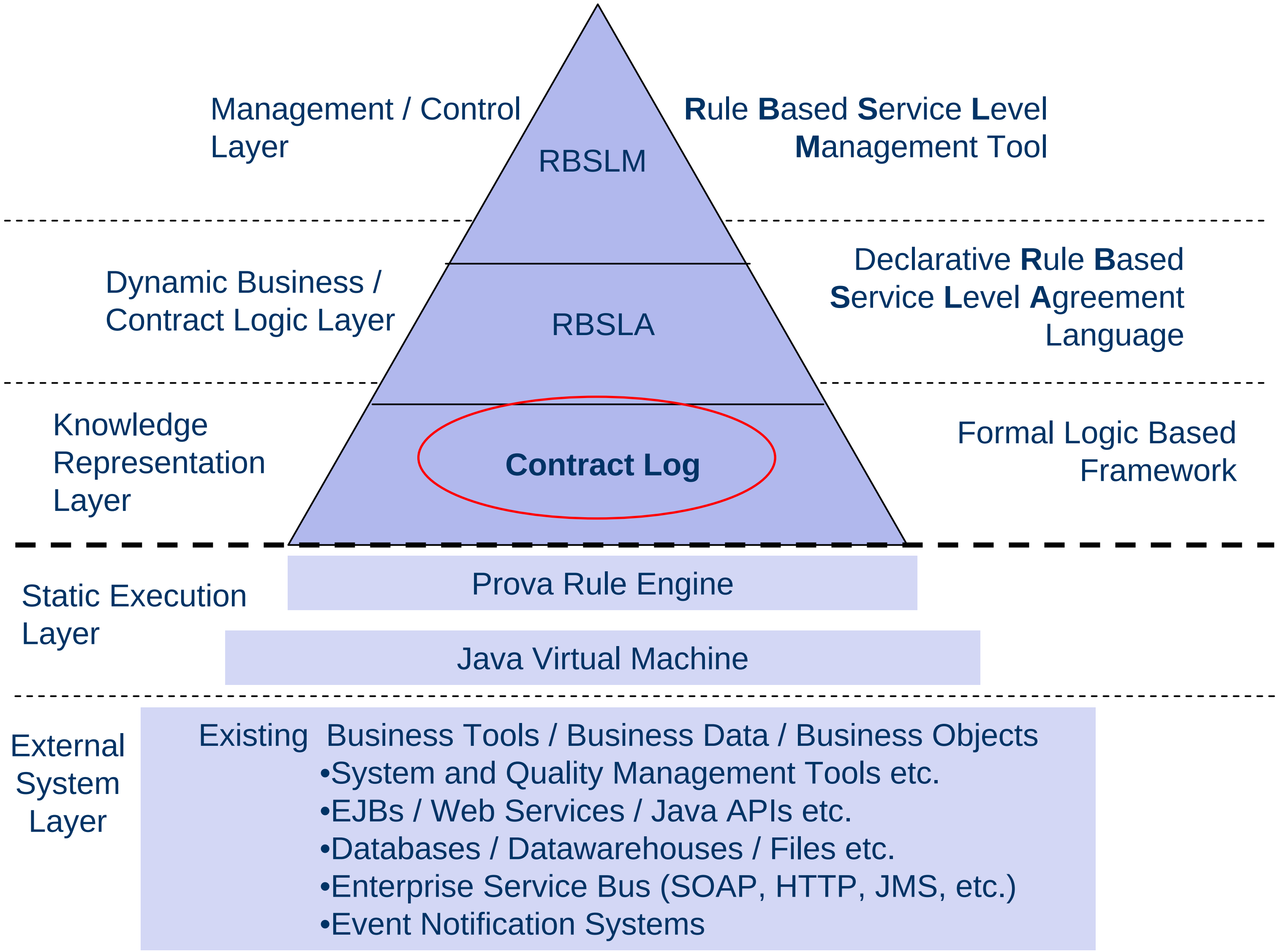
Conlusion	
Predicate	Variable
gold	C

If Customer has gold status, then discount Customer is -15 %.

Prerequisite	
Predicate	Variable
gold	Customer

Conclusion		
Predicate	Variable	Constant
discount	Customer	-15%

RBSLA Architecture



Management / Control Layer

RBSLM

Rule **B**ased **S**ervice **L**evel **M**anagement Tool

Dynamic Business / Contract Logic Layer

RBSLA

Declarative **R**ule **B**ased **S**ervice **L**evel **A**greement Language

Knowledge Representation Layer

Contract Log

Formal Logic Based Framework

Static Execution Layer

Prova Rule Engine

Java Virtual Machine

External System Layer

Existing Business Tools / Business Data / Business Objects

- •System and Quality Management Tools etc.
- •EJBs / Web Services / Java APIs etc.
- •Databases / Datawarehouses / Files etc.
- •Enterprise Service Bus (SOAP, HTTP, JMS, etc.)
- •Event Notification Systems

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

Typing, Import and Java Intergration Examples

```
import("http//../dl_typing/businessVocabulary1.owl").
import("http//../dl_typing/mathVocabulary.owl").
import("http//../dl_typing/currencyVocabulary.owl").

reasoner("dl"). % configure reasoner (OWL-DL=Pellet)

% Rule-based Discount Policy
discount(X:businessVoc1_Customer, math_Percentage:10) :-
        gold(X: businessVoc1_Customer).
discount(X: businessVoc2_Client, math_Percentage:5) :-
        silver(X: businessVoc1_Client).
discount(X: businessVoc2_Customer, math_Percentage:2) :-
        bronze(X: businessVoc1_Customer).
discount(X, 0) :-
        not(spending(X,lastYear)).   ...
```

```
% module imports
:-eval(consult(
  './ContractLog/math.prova')).

:-eval(consult(
  'http://ibis.in.tum.de/
  projects/rbsla/test.prova')).

% Java Integration

readfiles(Files) :-
  Dir=java.io.File("."),
  Files=Dir.list().

% Goal
:- eval(readfiles(Files)).
```

```
:-solve(discount(X:businessVoc2_Customer,Y:math_Percentage).
:-solve(discount(X:businessVoc1_Customer, math_Percentage:2).
:-solve(discount(X:rdfs_Resource, 5).
:-solve(discount(X,Y)).
:-solve(readfiles(Files)).
```

Event Calculus Example

Effects of Events/Actions on Fluents

- Rules for state transitions / derive actual contract State ~ Context
- Contract State Tracking
- Time-based / Context based complex events

EC Basic Axioms:

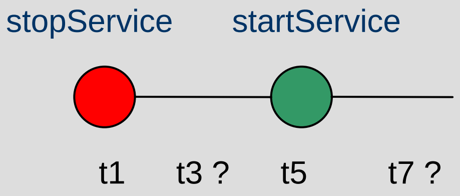
happens(E,T)	*event E happens at time point T*
initiates(E,F,T)	*event E initiates fluent F for all time>T*
terminates(E,F,T)	*event E terminates fluent F for all time>T*
holdsAt(F,T)	*fluent F holds at time point T*

EC Extensions:

valueAt(P,T,X)	*parameter P has changeable value X at time point T*
planned(E,T)	*event E is believed to happen at time point T*

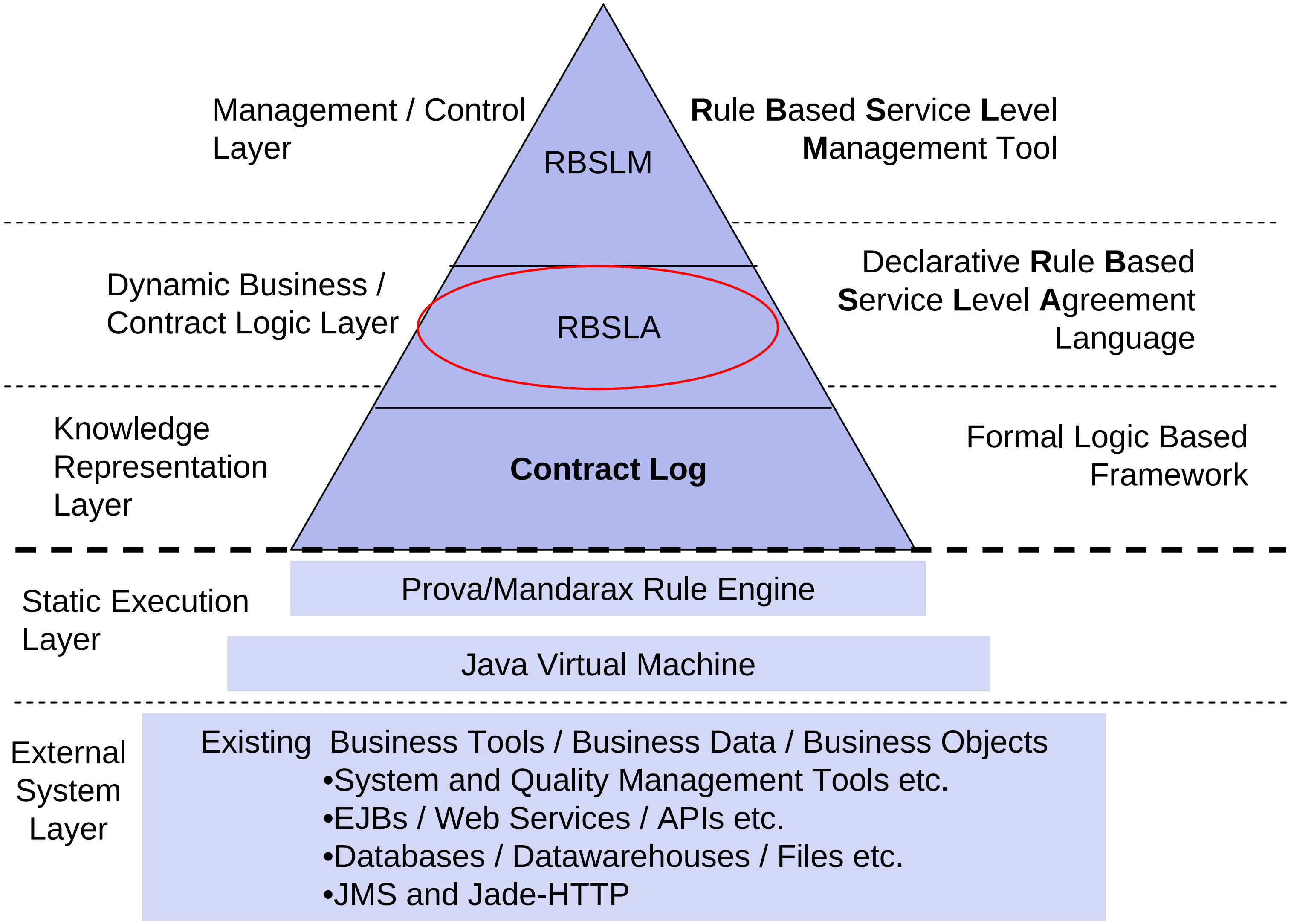
Example:

initiates*(stopService,serviceUnavailable,T)*
terminates*(startService,serviceUnavailable,T)*
happens*(stopService,t1);* ***happens****(startService,t5)*



holdsAt*(serviceUnavailable,t3)?* → ***true***
holdsAt*(serviceUnavailable,t7)?* → ***false***

RBSLA Architectur



Management / Control Layer

RBSLM

Rule **B**ased **S**ervice **L**evel **M**anagement Tool

Dynamic Business / Contract Logic Layer

RBSLA

Declarative **R**ule **B**ased **S**ervice **L**evel **A**greement Language

Knowledge Representation Layer

Contract Log

Formal Logic Based Framework

Static Execution Layer

Prova/Mandarax Rule Engine

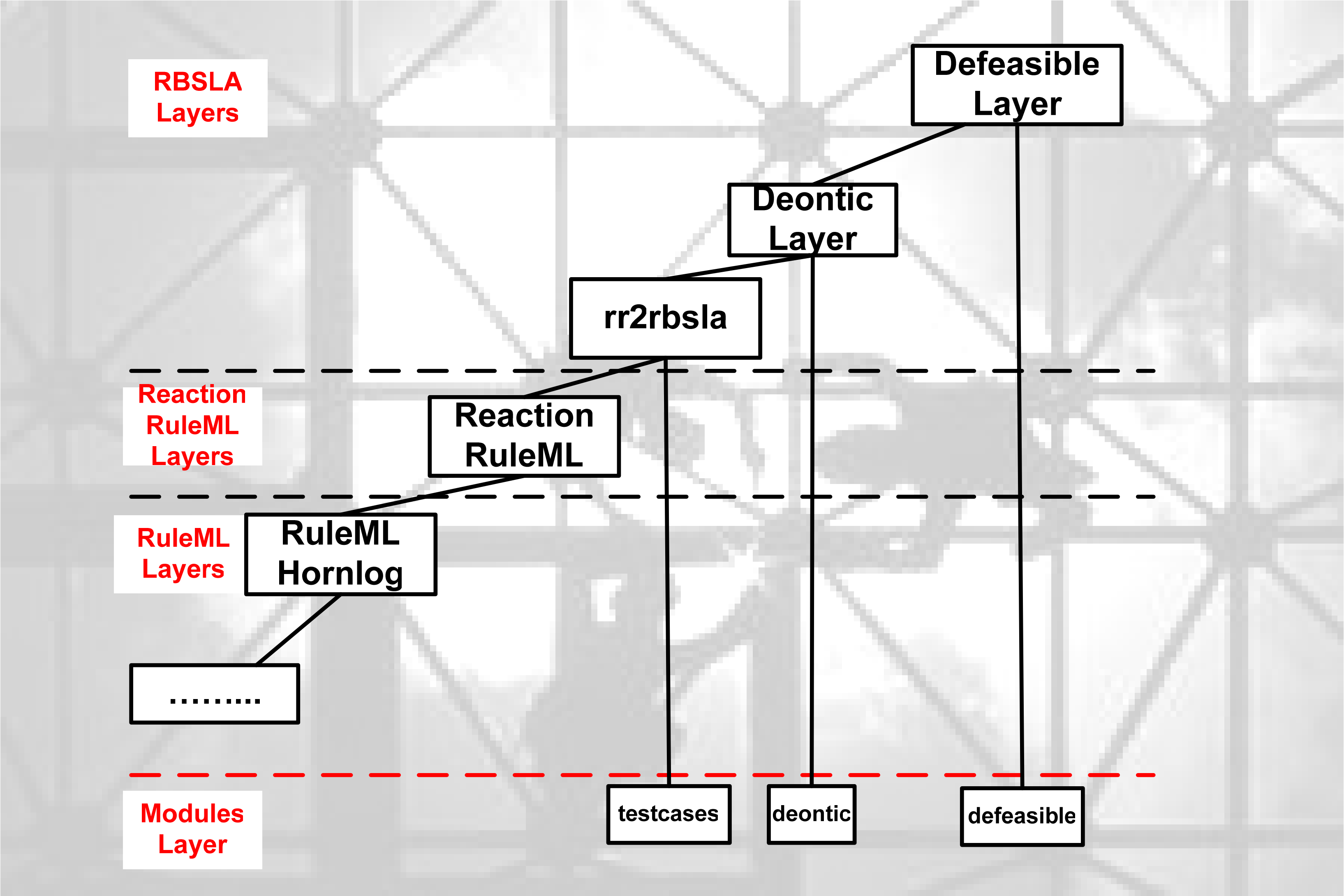
Java Virtual Machine

External System Layer

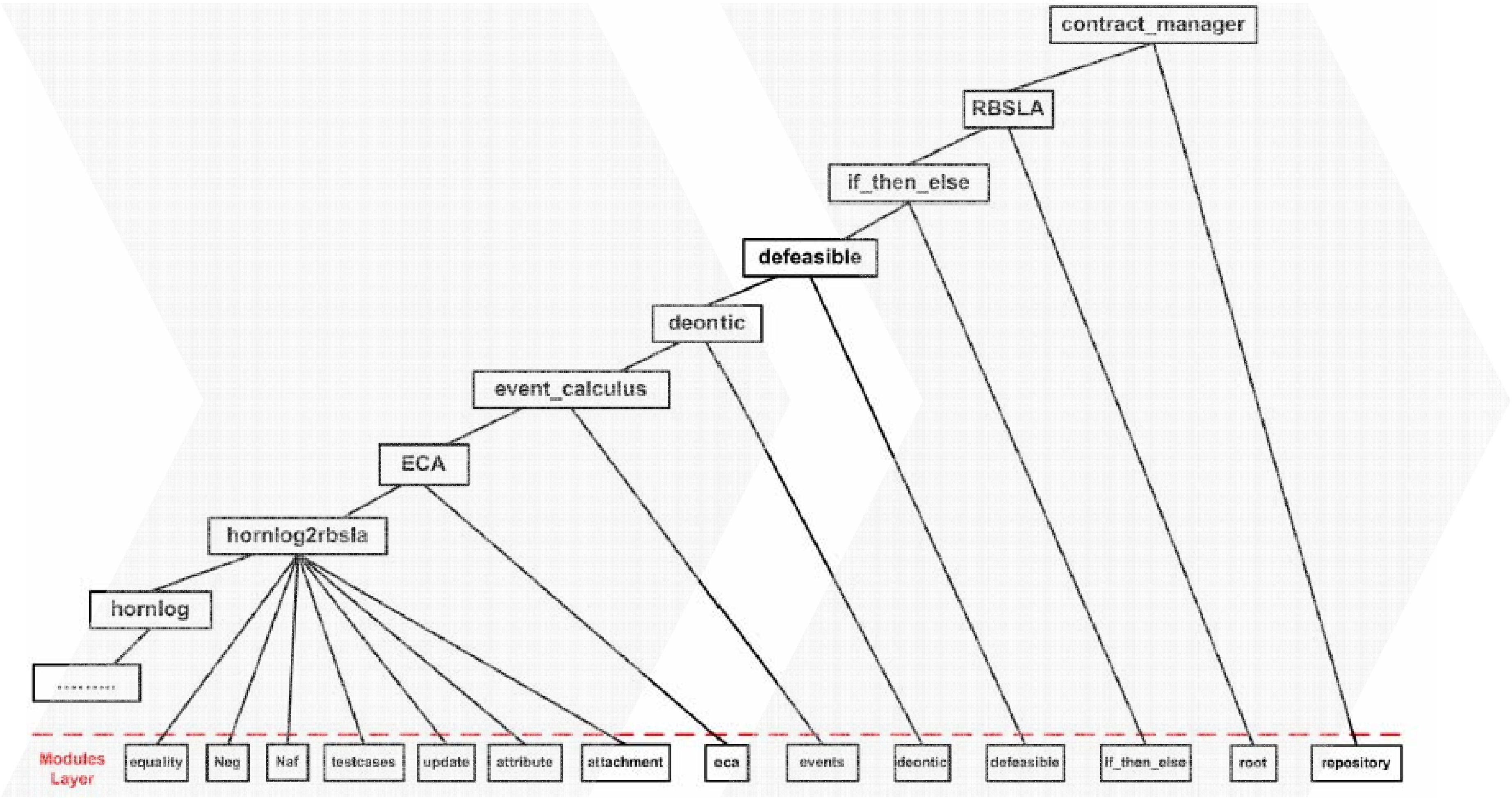
Existing Business Tools / Business Data / Business Objects

- •System and Quality Management Tools etc.
- •EJBs / Web Services / APIs etc.
- •Databases / Datawarehouses / Files etc.
- •JMS and Jade-HTTP

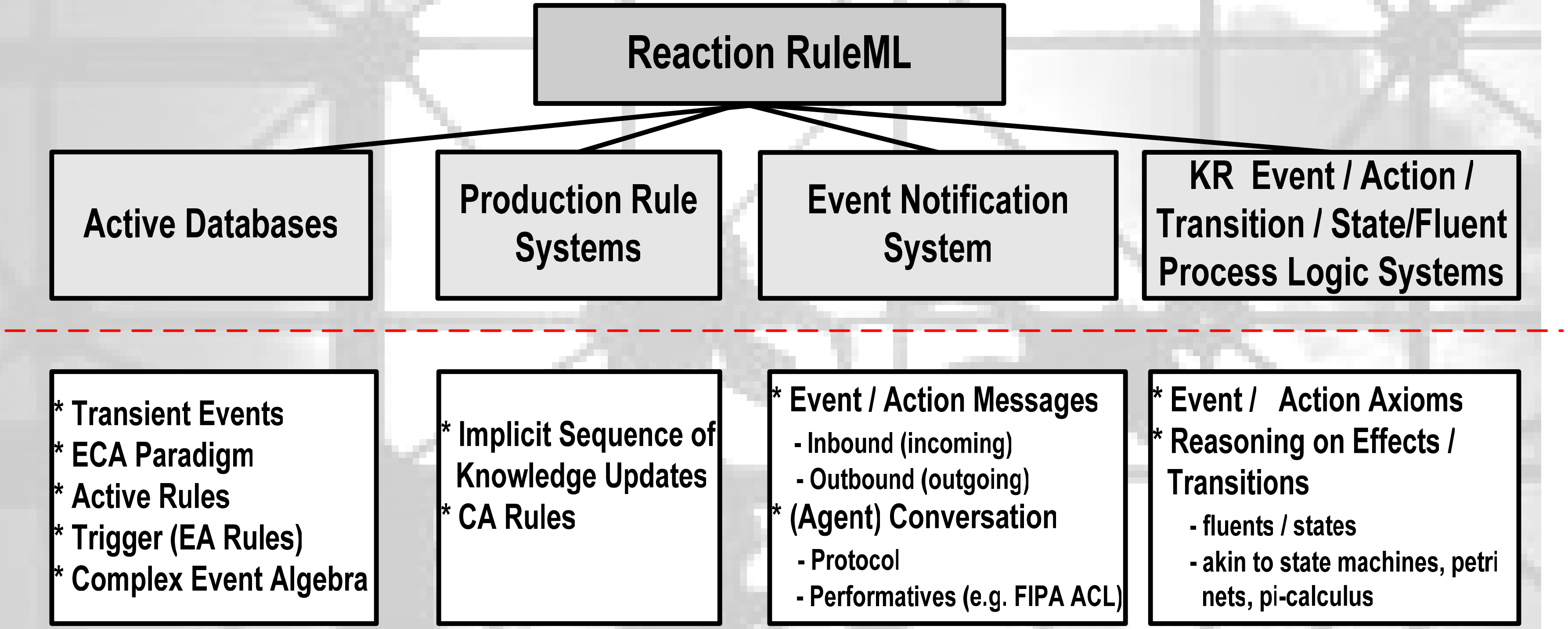
RBSLA: Rule Based Service Level Agreement Language



RBSLA Schema Structure



Scope of Reaction RuleML



Reaction RuleML

Active Databases

Production Rule Systems

Event Notification System

KR Event / Action / Transition / State/Fluent Process Logic Systems

- * Transient Events
- * ECA Paradigm
- * Active Rules
- * Trigger (EA Rules)
- * Complex Event Algebra

- * Implicit Sequence of Knowledge Updates
- * CA Rules

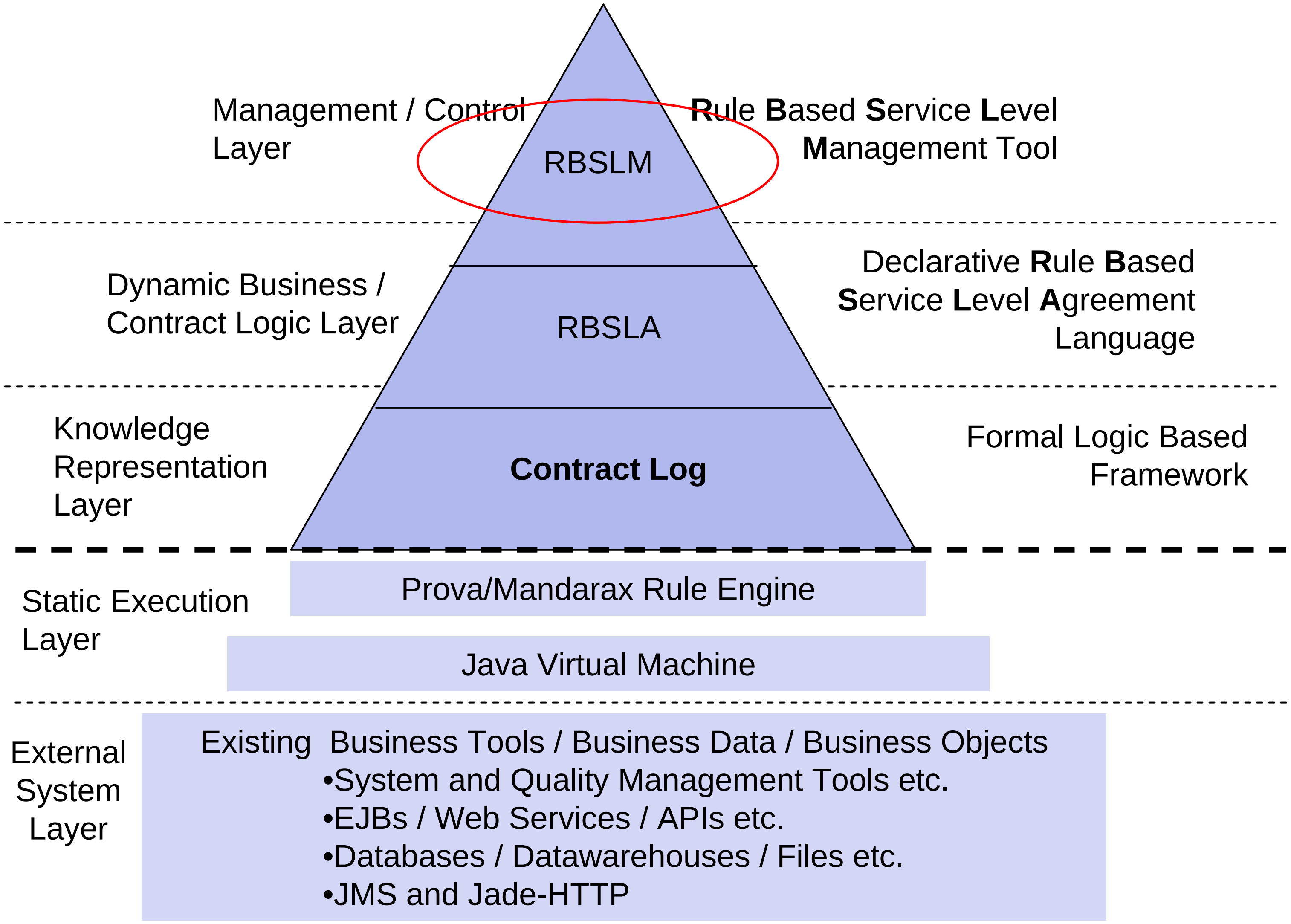
- * Event / Action Messages
  - Inbound (incoming)
  - Outbound (outgoing)
- * (Agent) Conversation
  - Protocol
  - Performatives (e.g. FIPA ACL)

- * Event / Action Axioms
- * Reasoning on Effects / Transitions
  - fluents / states
  - akin to state machines, petri nets, pi-calculus

RBSLA Features

- Rule Interchange and Serialization
- Layered structure (unitized in modules)
- Main Features
  - Module Concept
  - Import and Include
  - External Type Systems (vocabularies)
  - Procedural Attachments and external functions
  - External Data Integration
  - Event Condition Action Rules with Sensing, Monitoring and Effecting
  - Derivation Rules
  - (Situated) Update Primitives
  - Complex Event Processing and State Changes (Fluents)
  - Deontic Norms and Norm Violations and Exceptions
  - Defeasible Rules and Rule Priorities
  - Built-Ins, Aggregate and Compare Operators, Lists …
- Declartive Rule Programming Language
  - Based on logical semantics (ContractLog KR)
  - Syntactically extensible by external vocabularies (e.g. WSMO, OWL WS-Policy, OWL time etc.)

RBSLA Architecture



Management / Control Layer

RBSLM

Rule **B**ased **S**ervice **L**evel **M**anagement Tool

Dynamic Business / Contract Logic Layer

RBSLA

Declarative **R**ule **B**ased **S**ervice **L**evel **A**greement Language

Knowledge Representation Layer

Contract Log

Formal Logic Based Framework

Static Execution Layer

Prova/Mandarax Rule Engine

Java Virtual Machine

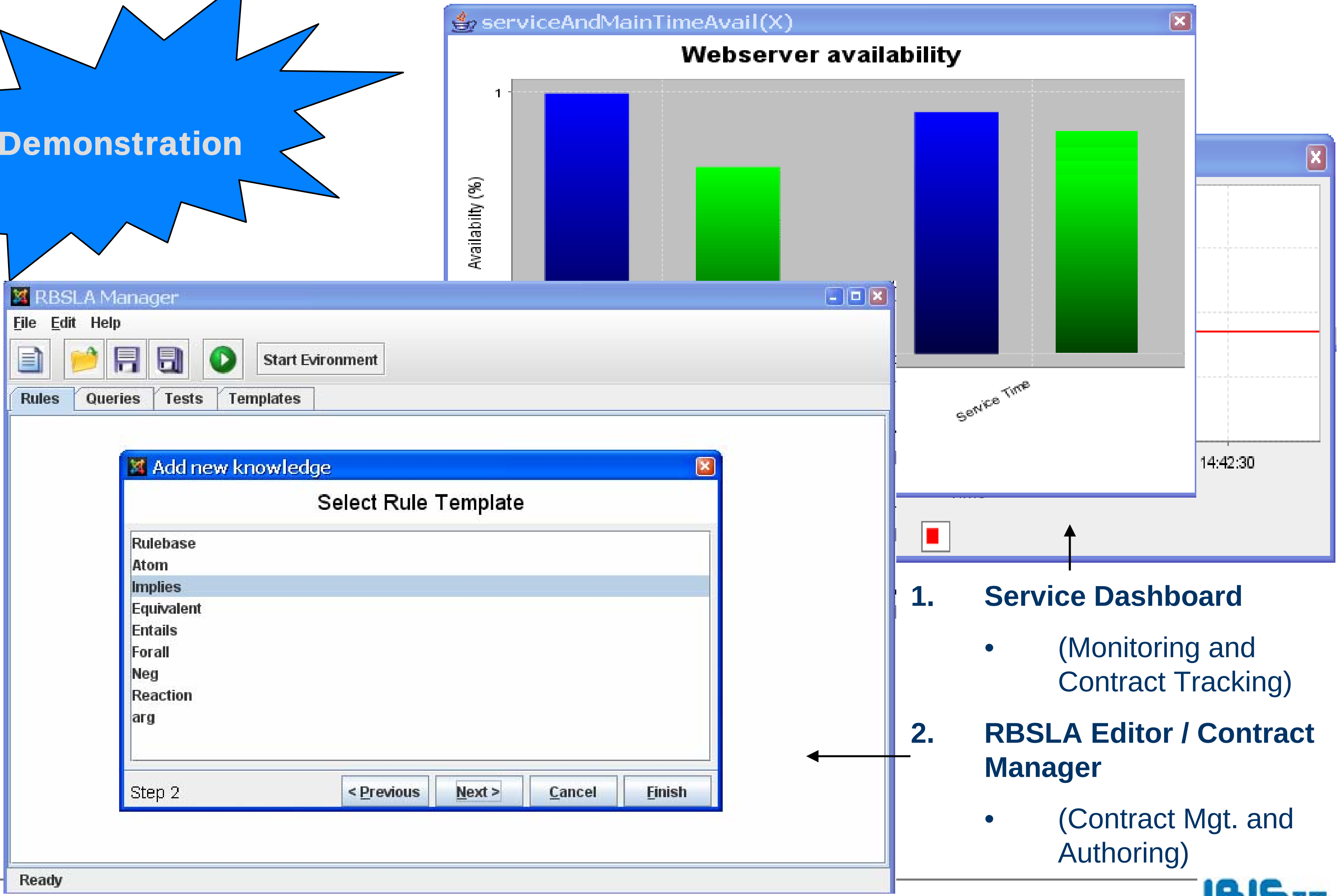
External System Layer

Existing Business Tools / Business Data / Business Objects

- System and Quality Management Tools etc.
- EJBs / Web Services / APIs etc.
- Databases / Datawarehouses / Files etc.
- JMS and Jade-HTTP

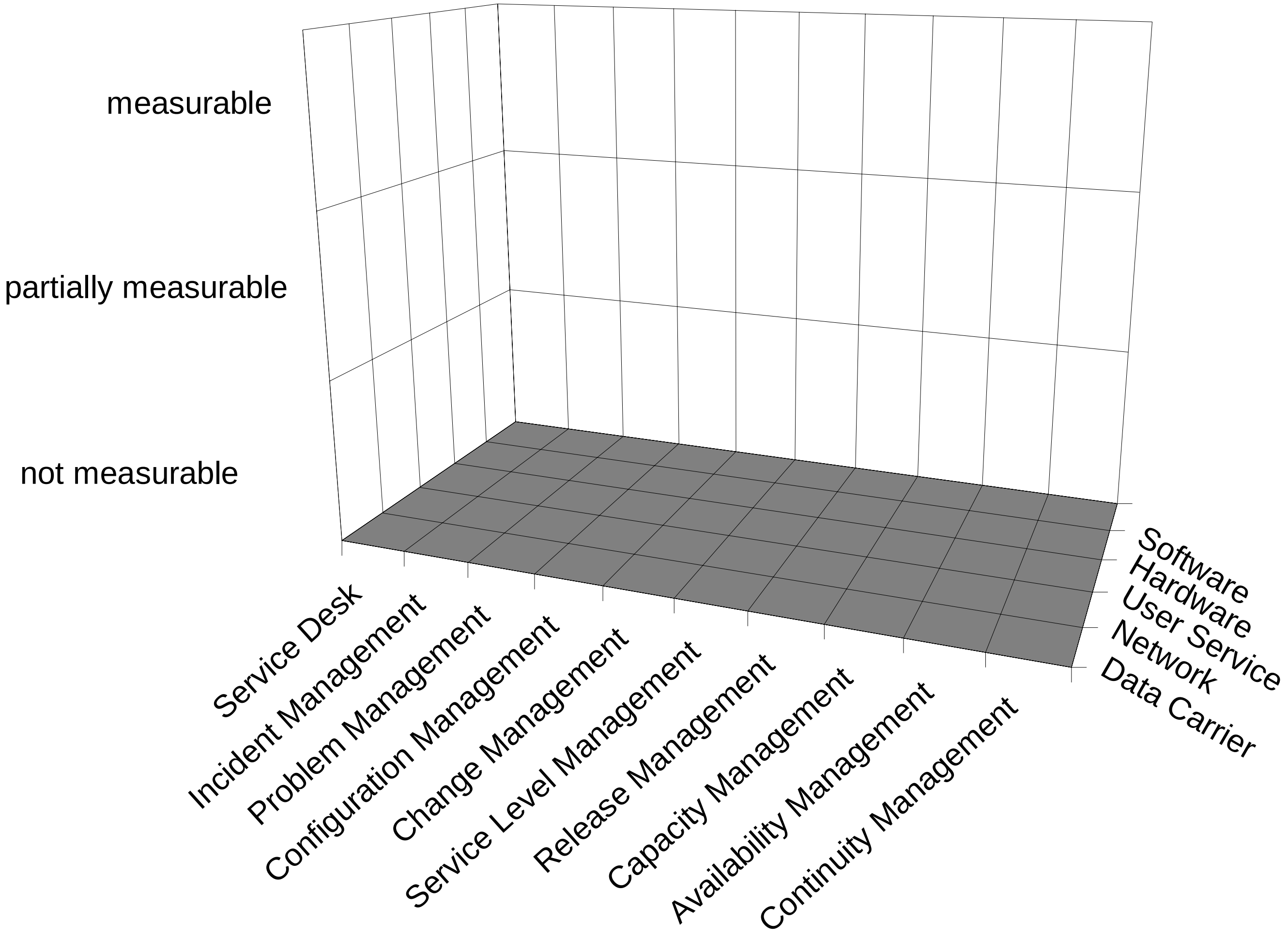
Rule Based Service Level Management (RBSLM)

Demonstration

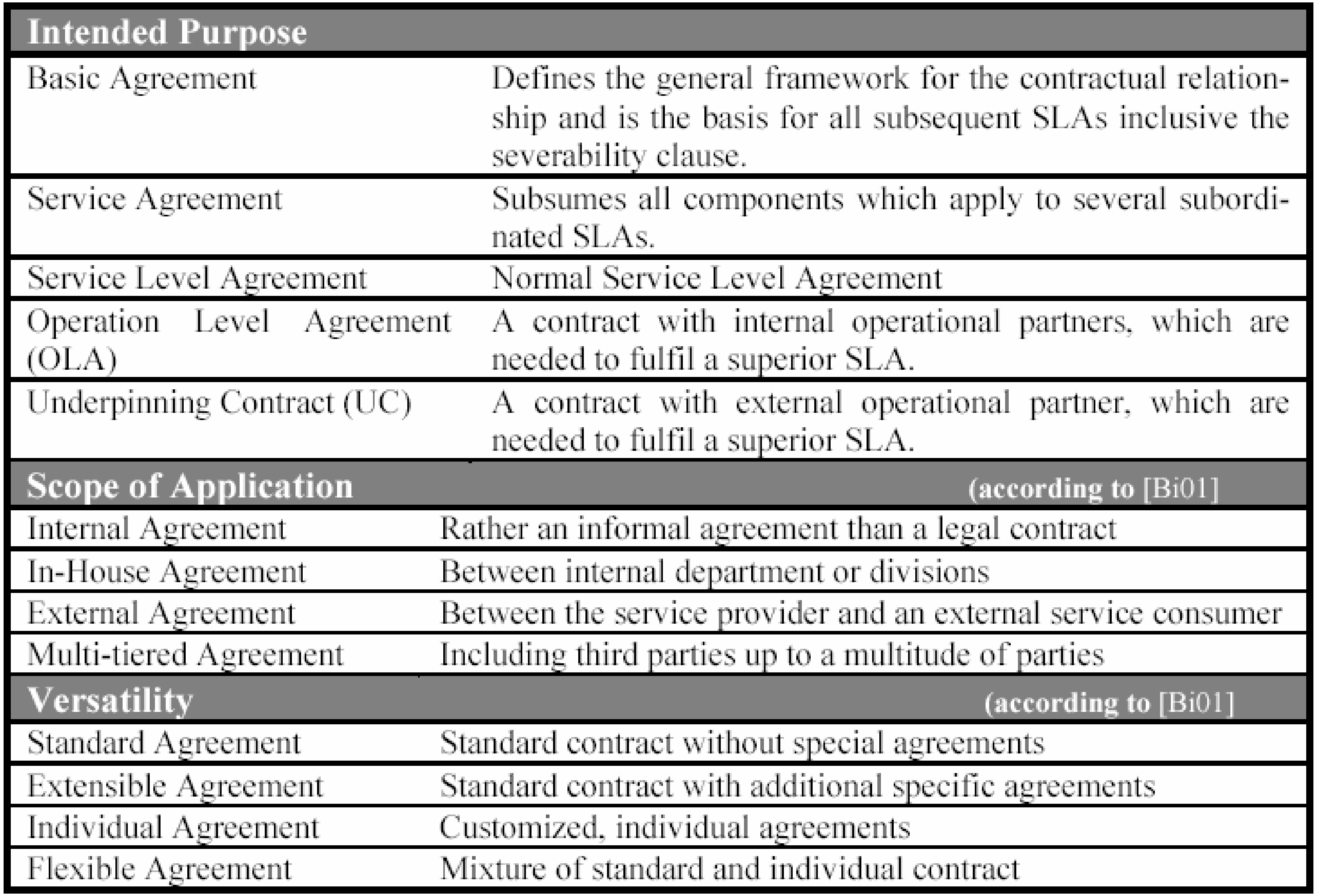


1. **Service Dashboard**
   - (Monitoring and Contract Tracking)
2. **RBSLA Editor / Contract Manager**
   - (Contract Mgt. and Authoring)

Multi-layered Categorization of Contracts / Metrics

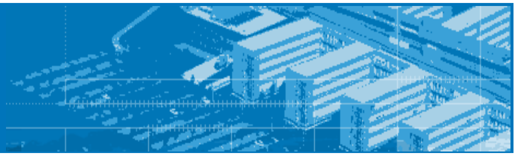


Categorization of SLAs



Intended Purpose	
Basic Agreement	Defines the general framework for the contractual relationship and is the basis for all subsequent SLAs inclusive the severability clause.
Service Agreement	Subsumes all components which apply to several subordinated SLAs.
Service Level Agreement	Normal Service Level Agreement
Operation Level Agreement (OLA)	A contract with internal operational partners, which are needed to fulfil a superior SLA.
Underpinning Contract (UC)	A contract with external operational partner, which are needed to fulfil a superior SLA.
Scope of Application	**(according to** [Bi01]
Internal Agreement	Rather an informal agreement than a legal contract
In-House Agreement	Between internal department or divisions
External Agreement	Between the service provider and an external service consumer
Multi-tiered Agreement	Including third parties up to a multitude of parties
Versatility	**(according to** [Bi01]
Standard Agreement	Standard contract without special agreements
Extensible Agreement	Standard contract with additional specific agreements
Individual Agreement	Customized, individual agreements
Flexible Agreement	Mixture of standard and individual contract

Categorization of SLA Metrics



No	Description	Object	Unit
1	Availability	Hardware	Time hour, percent
2	Maximum down-time	Hardware	Hours or percent
3	Failure frequency	Hardware	Number
4	Response time	Hardware	Duration in minutes/seconds
5	Periods of operation	Hardware	Time
6	Service times	Hardware	Time
7	Accessibility in case of problems	Hardware	Yes/no
8	Backup	Hardware	Time
9	Processor time	Hardware	Seconds
10	Instructions per second	Hardware	Number per second
11	Number of workstations	Hardware	Number

No	Description	Object	Unit
1	Service times	Software	Time
2	Response times	Software	Minutes
3	Availability	Software	Time
4	Solution times	Software	Minutes
5	Number of licences	Software	Number

Categorization of SLA Metrics

No	Description	Object	Unit
1	WAN period of operation	Network	Time
2	WAN Service times	Network	Time
3	LAN period of operation	Network	Time
4	LAN Service times	Network	Time
5	Solution times	Network	Minutes
6	Availability WAN	Network	Percent
7	Availability LAN	Network	Percent
8	Access Internet across Firewall	Network	Yes/no
9	Access RAS	Network	Yes/no
10	Latency times	Network	Ms

No	Description	Object	Unit
1	Availability	Storage	Time hour, percent
2	Maximum down-time	Storage	Hours or percent
3	Failure frequency	Storage	Number
4	Response time	Storage	Duration in minutes/seconds
5	Periods of operation	Storage	Time
6	Service times	Storage	Time
7	Accessibility in the case of problem	Storage	Yes/no
8	Backup	Storage	Time
9	Bytes per second	Storage	Number per second
10	Memory size	Storage	Number in bytes

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

ITIL Service Metrics

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

Performance and Scalability Evaluation

- Theoretical Worst Case Complexity
  - Only isolated analysis of ContractLog KR formalisms possible (most of them polynomial)

- Performance Test Theories (Experimental Evaluation)
  - Heuristic, algorithmic Adequacy

- Examples
  - Rule Chaining

    In chains(n) a_0 is at the end of a chain of n rules $a_i():\text{-}\ a_{i-1}()$. chains(n) starts with fact a_0 and continues with a chain of n (strict) rules of the form $a_i():\text{-}\ a_{i-1}()$. A goal a_n will use all of n rules and the fact.

$$\text{chain(n)=} \begin{cases} a_n :\text{-}\ a_{n-1}. \\ \ldots \\ a_2 :\text{-} a_1. \\ a_1 :\text{-} a_0. \end{cases}$$

$$a_0. \qquad \text{\% fact}$$

  - Directed Acyclic Graph

    In dag(n,k) a_0 is at the root of a k-branching tree of depth n in which every literal occurs k times.

$$\text{dag(n,k) =} \begin{cases} a_0 :\text{-}\ a_1, a_2, \ldots, a_k. \\ a_1 :\text{-}\ a_2, a_3, \ldots, a_{k+1}. \\ \ldots \\ a_{nk} :\text{-}\ a_{nk+1}, a_{nk+2}, \ldots, a_{nk+k}. \ \text{\% rules} \end{cases}$$

$$a_{nk+1}.\ a_{nk+2}.\ \ldots\ a_{nk+k}. \quad \text{\% facts}$$

Performance and Scalability Evaluation

Test Theory Examples

Loop Checking (logic-formal adequacy)

circles(n) consists of n rules $a_i()$:- $a_{i-1}()$ and a circular rule $a_0()$:- $a_n()$

$$\text{circles(n)=} \begin{cases} a_n \text{ :- } a_{n-1}. \\ \dots \\ a_2\text{:-}a_1. \\ a_1\text{:-}a_0. \\ a_0\text{:-}a_n. \end{cases}$$

a_0. % fact

Recursion

In tree(n,k) a_0 is at the root of a k-branching tree of depth n in which every literal occurs once.

tree(n,k) = rule(a_0,n,k) where, if p is a literal, n>0, r is a new unique label, and $a_1,a_2,\dots,a_k$ are new unique literals:

$$\text{rules(p,n,k) =} \begin{cases} p \text{ :- } a_1,a_2,\dots,a_k. \\ \text{rules}(a_1\text{,n-1,k}) \\ \text{rules}(a_2\text{,n-1,k}) \\ \dots \\ \text{rules}(a_k\text{,n-1,k}) \end{cases}$$

and: rules(p,0,k) = p

Reactive Rules

reactive(n,t) constists of n ECA rules eca_n(t,e,c,a) which fire every interval t.

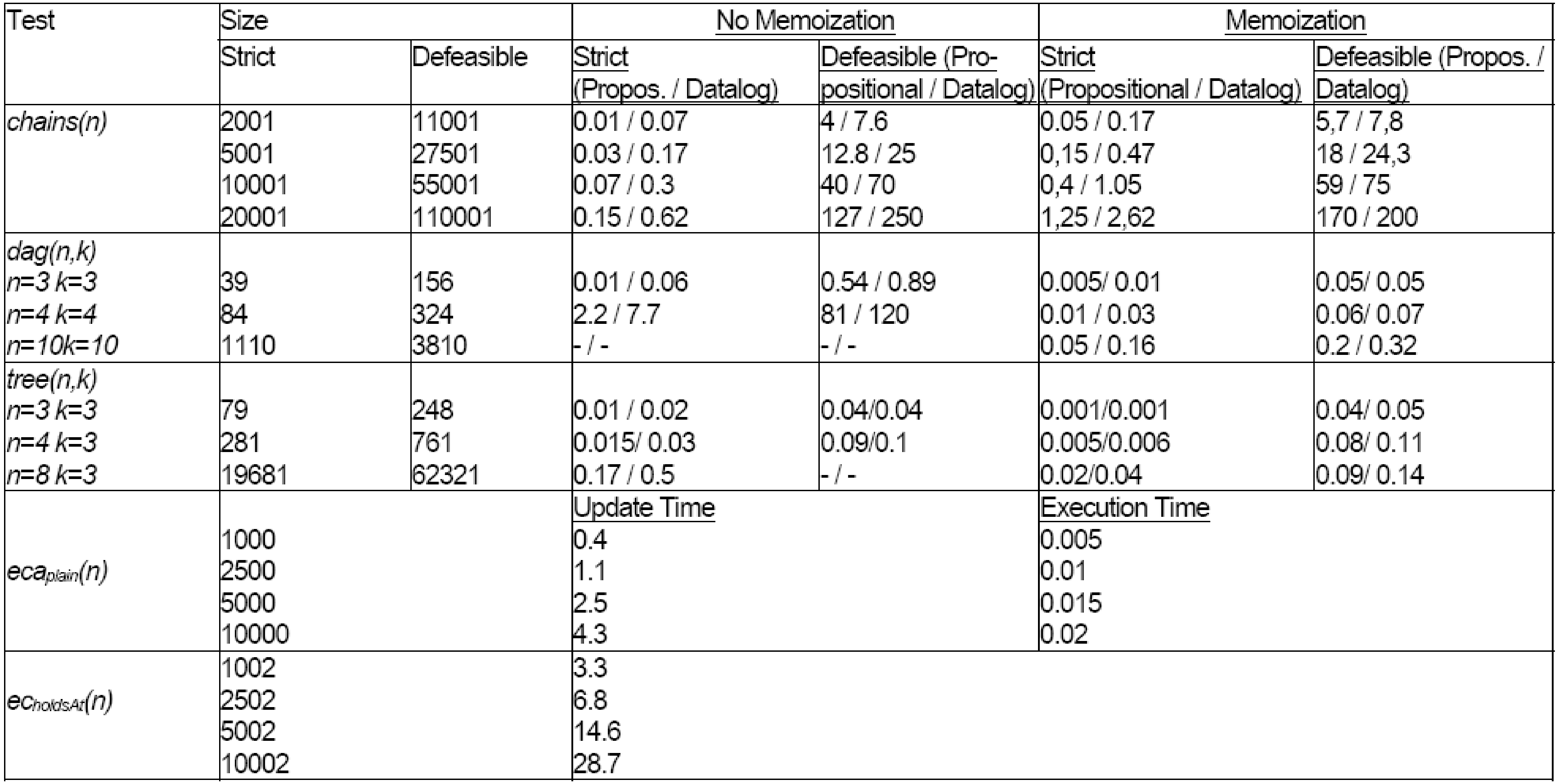
Event Calculus

occurrence(n,m) consists of a m pairs of initiates / terminates Event Calculus rules and n event occurrences (happens facts) which initiate resp. terminate a fluent.

…

Experimental Performance Evaluation

- ContractLog Hybrid Logic: Combination of declarative logic programming and object-oriented programming
- Performance Tests



Test	Size		No Memoization		Memoization	
	Strict	Defeasible	Strict (Propos. / Datalog)	Defeasible (Propositional / Datalog)	Strict (Propositional / Datalog)	Defeasible (Propos. / Datalog)
chains(n)	2001	11001	0.01 / 0.07	4 / 7.6	0.05 / 0.17	5,7 / 7,8
	5001	27501	0.03 / 0.17	12.8 / 25	0,15 / 0.47	18 / 24,3
	10001	55001	0.07 / 0.3	40 / 70	0,4 / 1.05	59 / 75
	20001	110001	0.15 / 0.62	127 / 250	1,25 / 2,62	170 / 200
dag(n,k)						
n=3 k=3	39	156	0.01 / 0.06	0.54 / 0.89	0.005/ 0.01	0.05/ 0.05
n=4 k=4	84	324	2.2 / 7.7	81 / 120	0.01 / 0.03	0.06/ 0.07
n=10k=10	1110	3810	- / -	- / -	0.05 / 0.16	0.2 / 0.32
tree(n,k)						
n=3 k=3	79	248	0.01 / 0.02	0.04/0.04	0.001/0.001	0.04/ 0.05
n=4 k=3	281	761	0.015/ 0.03	0.09/0.1	0.005/0.006	0.08/ 0.11
n=8 k=3	19681	62321	0.17 / 0.5	- / -	0.02/0.04	0.09/ 0.14
$eca_{plain}(n)$			Update Time		Execution Time	
	1000		0.4		0.005	
	2500		1.1		0.01	
	5000		2.5		0.015	
	10000		4.3		0.02	
$ec_{holdsAt}(n)$	1002		3.3			
	2502		6.8			
	5002		14.6			
	10002		28.7			

Contributions to IT Service Level Management (1)

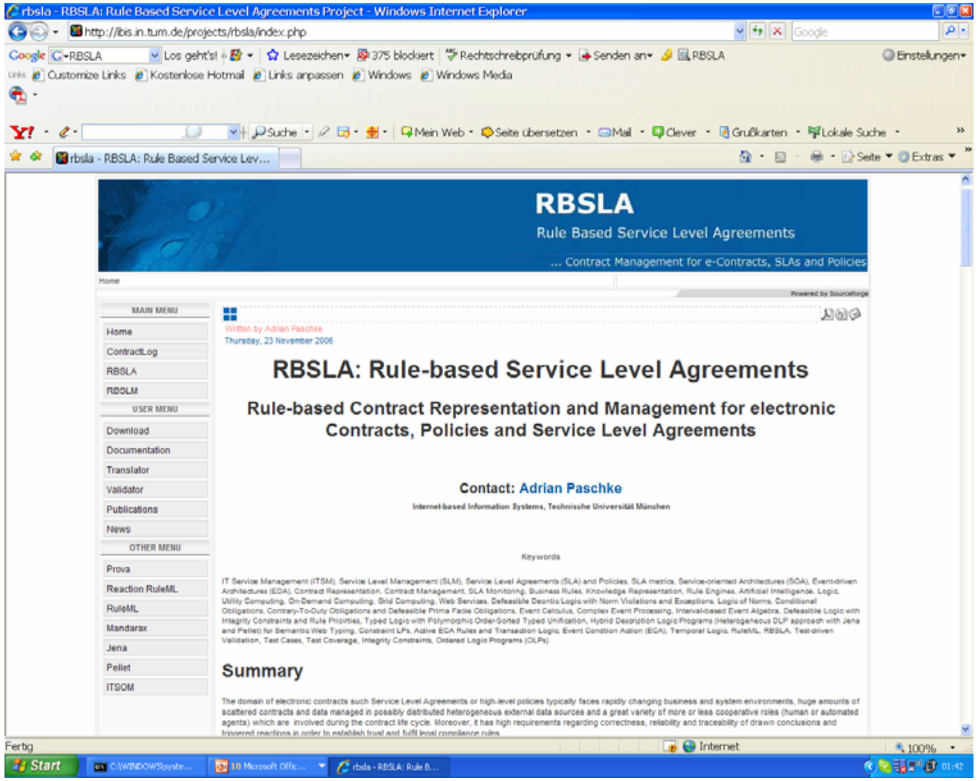
- Application of logic programming in SLM
  - Flexible and dynamic rule management
  - Complex SLA rules (not just parameters with thresholds)
  - Reduced Costs for Modification of contract logic
  - Shorter development cycles
  - Simplified decentralized management and reuse of SLA rules for different 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 for representations of SLAs

Contributions to IT Service Level Management (2)

- Integration and extension of different advanced logic concepts in ContractLog KR
  - Semantics, Expressiveness, Scalability, Flexibility
- Integration into standardization initiatives (Reaction RuleML, RIF, PRR)
- Integration and Interoperation with Semantic Web Standards
  - RDFS/OWL: Contract ontologies
  - Rule Based Service Level Agreement (RBSLA) Language
  - Straightforward Integration into Internet Technologies
    (e.g. WSDL extension with reference to RBSLA)
- Declarative Rule Based Programming
  - Clear logic based semantics
  - Syntactic extensions via integration of Semantic Web Vocabularies

Homepage

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


rbsla - RBSLA: Rule Based Service Level Agreements Project - Windows Internet Explorer
http://ibis.in.tum.de/projects/rbsla/index.php
Google
Google G-RBSLA Los geht's! Lesezeichen 375 blockiert Rechtschreibprüfung Senden an RBSLA Einstellungen
Links Customize Links Kostenlose Hotmail Links anpassen Windows Windows Media
Y! Suche Mein Web Seite übersetzen Mail Clever Grußkarten Lokale Suche
rbsla - RBSLA: Rule Based Service Lev...
Seite Extras
Home
Powered by Sourceforge
MAIN MENU
Home
ContractLog
RBSLA
RBSLM
USER MENU
Download
Documentation
Translator
Validator
Publications
News
OTHER MENU
Prova
Reaction RuleML
RuleML
Mandarax
Jena
Pellet
ITSOM
Fertig
Internet
100%
Start
C:\WINDOWS\syste...
10 Microsoft Offic...
rbsla - RBSLA: Rule B...
01:42

RBSLA

Rule Based Service Level Agreements

... Contract Management for e-Contracts, SLAs and Policies

Written by Adrian Paschke

Thursday, 23 November 2006

RBSLA: Rule-based Service Level Agreements

Rule-based Contract Representation and Management for electronic Contracts, Policies and Service Level Agreements

Contact: Adrian Paschke

Internet-based Information Systems, Technische Universität München

Keywords

IT Service Management (ITSM), Service Level Management (SLM), Service Level Agreements (SLA) and Policies, SLA metrics, Service-oriented Architectures (SOA), Event-driven Architectures (EDA), Contract Representation, Contract Management, SLA Monitoring, Business Rules, Knowledge Representation, Rule Engines, Artificial Intelligence, Logic, Utility Computing, On-Demand Computing, Grid Computing, Web Services, Defeasible Deontic Logic with Norm Violations and Exceptions, Logic of Norms, Conditional Obligations, Contrary-To-Duty Obligations and Defeasible Prima Facie Obligations, Event Calculus, Complex Event Processing, Interval-based Event Algebra, Defeasible Logic with Integrity Constraints and Rule Priorities, Typed Logic with Polymorphic Order-Sorted Typed Unification, Hybrid Description Logic Programs (Heterogeneous DLP approach with Jena and Pellet) for Semantic Web Typing, Constraint LPs, Active ECA Rules and Transaction Logic, Event Condition Action (ECA), Temporal Logic, RuleML, RBSLA, Test-driven Validation, Test Cases, Test Coverage, Integrity Constraints, Ordered Logic Programs (OLPs).

Summary

The domain of electronic contracts such Service Level Agreements or high-level policies typically faces rapidly changing business and system environments, huge amounts of scattered contracts and data managed in possibly distributed heterogeneous external data sources and a great variety of more or less cooperative roles (human or automated agents) which are involved during the contract life cycle. Moreover, it has high requirements regarding correctness, reliability and traceability of drawn conclusions and triggered reactions in order to establish trust and fulfil legal compliance rules.

https://sourceforge.net/projects/rbsla