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

© Internet-based Information Systems
Dept. of Informatics, TU München
Research Areas

4. Negotiation and Coordination Mechanisms with decentralized Decision Makers

3. Heterogeneous Information Systems
   Modern XML and Internet Technologies

2. Supply Chain Management
   Supply Chain Tracking and Event Management

1. Multi-Agent Systems

5. Semantic Web
   Technologies and Ontology based Content Management Systems

6. Web Services
   Computing

7. IT Service Management and
   IT Service Level Management

8. Rule Based Systems and Rule Standards
Selected Projects

- **PAMAS - Proactive Multi Agent System**
- **Semantic-enriched Media Content Management System**
- **DPL - Design Pattern Library**
- **RBSLA – Rule Based Service Level Agreements**
RBSLA

Rule Based Service Level Agreement

IT Service Management for electronic Contracts, Policies and SLAs
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

Service / Master Agreement

Service Level Agreement

Operational Level Agreement

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**

<table>
<thead>
<tr>
<th>Schedule</th>
<th>Time</th>
<th>Availability</th>
<th>Response Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prime</td>
<td>8 -18</td>
<td>99%</td>
<td>4 sec.</td>
</tr>
<tr>
<td>Standard</td>
<td>18-8</td>
<td>95%</td>
<td>10 sec.</td>
</tr>
<tr>
<td>Maintenance</td>
<td>0-4 *</td>
<td>30%</td>
<td>-</td>
</tr>
</tbody>
</table>

  **Escalation Levels with Role Model**

<table>
<thead>
<tr>
<th>Level</th>
<th>Role</th>
<th>Time-to-Repair</th>
<th>Rights / Obligations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Process Manager</td>
<td>10 Min.</td>
<td>Start / Stop Service</td>
</tr>
<tr>
<td>2</td>
<td>Chief Quality Manager</td>
<td>Max. Time-to-Repair</td>
<td>Change Service Levels</td>
</tr>
<tr>
<td>3</td>
<td>Control Committee</td>
<td>-</td>
<td>All rights</td>
</tr>
</tbody>
</table>
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 time-to-repair. If she fails to do so (violation) the customer is permitted to cancel the contract.”
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:
  
  ```xml
  <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:Preference>
    <wsag:Preference>
      <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)
IBM Web Service Level Agreement Sprache (WSLA):

\[
\text{IF } "\text{TransactionRate} < 10000" \text{ THEN } "\text{AverageResponseTime} < 0.5"
\]

\[
\text{IF } "\text{AverageResponseTime} < 0.5" \text{ THEN } \ldots
\]

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

-\( \rightarrow \) no declarative programming; 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
## Bonus-Malus System

<table>
<thead>
<tr>
<th>Quality of Service (QoS)</th>
<th>Average Availability (quantitative)</th>
<th>Bonus/Malus Discount</th>
</tr>
</thead>
<tbody>
<tr>
<td>High</td>
<td>100 %</td>
<td>+ 5%</td>
</tr>
<tr>
<td>Normal</td>
<td>98-100 %</td>
<td>+ 0%</td>
</tr>
<tr>
<td>Low</td>
<td>&lt;98 %</td>
<td>- 5%</td>
</tr>
<tr>
<td>Below average</td>
<td>&lt;95 %</td>
<td>1000 $ penalty</td>
</tr>
</tbody>
</table>

**If average availability is 100 % then QoS is high.**

<table>
<thead>
<tr>
<th>Body</th>
<th>Head</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predicate</td>
<td>Complex Term</td>
</tr>
<tr>
<td>=</td>
<td>availability(Service)</td>
</tr>
<tr>
<td>Predicate</td>
<td>Variable</td>
</tr>
<tr>
<td>qos</td>
<td>Service</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Body</th>
<th>Head</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predicate</td>
<td>Variable</td>
</tr>
<tr>
<td>qos</td>
<td>Service</td>
</tr>
<tr>
<td>Predicate</td>
<td>Variable</td>
</tr>
<tr>
<td>discount</td>
<td>Service</td>
</tr>
</tbody>
</table>
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 Programming

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%)? Discount „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 Programming

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**

<table>
<thead>
<tr>
<th>Prerequisite</th>
<th>Complex Term</th>
<th>Constant</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt;</td>
<td>getTurnover(C)</td>
<td>5000€</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Prerequisite</th>
<th>Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>gold</td>
<td>Customer</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Conclusion</th>
<th>Predicate</th>
<th>Variable</th>
<th>Constant</th>
</tr>
</thead>
<tbody>
<tr>
<td>gold</td>
<td>C</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
RBSLA Architecture

- **Rule Based Service Level Management Tool (RBSLM)**
- **Declarative Rule Based Service Level Agreement Language**
- **Contract Log**
- **Formal Logic Based Framework**
- **Prova Rule Engine**
- **Java Virtual Machine**
- **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:

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

:-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)).

% module imports
:-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)).
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)? \rightarrow true`
`holdsAt(serviceUnavailable, t7)? \rightarrow false`
RBSLA Architectur

- **Rule Based Service Level Agreement (RBSLA)**
  - **RBSLM (Rule Based Service Level Management Tool)**
  - **Contract Log**
  - **Knowledge Representation Layer**
  - **Dynamic Business / Contract Logic Layer**
  - **Management / Control Layer**

**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

**Static Execution Layer**
- Prova/Mandarax Rule Engine
- Java Virtual Machine

**Formal Logic Based Framework**
Scope of Reaction RuleML

Reaction RuleML

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

Production Rule Systems
* Implicit Sequence of Knowledge Updates
* CA Rules

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

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

- Declarative 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

RBSLM

Dynamic Business / Contract Logic Layer

Knowledge Representation Layer

Static Execution Layer

External System Layer

Prova/Mandarax Rule Engine

Java Virtual Machine

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 Tool

Declarative Rule Based Service Level Agreement Language

Formal Logic Based Framework

RBSLA

Contract Log

Management / Control Layer
Rule Based Service Level Management (RBLSM)

Demonstration

1. Service Dashboard
   - (Monitoring and Contract Tracking)
2. RBLSA Editor / Contract Manager
   - (Contract Mgt. and Authoring)
Multi-layered Categorization of Contracts / Metrics

Service Desk
Incident Management
Problem Management
Configuration Management
Change Management
Release Management
Service Level Management
Capacity Management
Availability Management
Continuity Management

Software
Hardware
User Service
Network
Data Carrier

measurable
partially measurable
not measurable

Data Carrier
Network
User Service
Hardware
Software
## Categorization of SLAs

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

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

<table>
<thead>
<tr>
<th>No</th>
<th>Description</th>
<th>Object</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Service times</td>
<td>Software</td>
<td>Time</td>
</tr>
<tr>
<td>2</td>
<td>Response times</td>
<td>Software</td>
<td>Minutes</td>
</tr>
<tr>
<td>3</td>
<td>Availability</td>
<td>Software</td>
<td>Time</td>
</tr>
<tr>
<td>4</td>
<td>Solution times</td>
<td>Software</td>
<td>Minutes</td>
</tr>
<tr>
<td>5</td>
<td>Number of licences</td>
<td>Software</td>
<td>Number</td>
</tr>
</tbody>
</table>
# Categorization of SLA Metrics

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

<table>
<thead>
<tr>
<th>No</th>
<th>Description</th>
<th>Object</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Availability</td>
<td>Storage</td>
<td>Time hour, percent</td>
</tr>
<tr>
<td>2</td>
<td>Maximum down-time</td>
<td>Storage</td>
<td>Hours or percent</td>
</tr>
<tr>
<td>3</td>
<td>Failure frequency</td>
<td>Storage</td>
<td>Number</td>
</tr>
<tr>
<td>4</td>
<td>Response time</td>
<td>Storage</td>
<td>Duration in minutes/seconds</td>
</tr>
<tr>
<td>5</td>
<td>Periods of operation</td>
<td>Storage</td>
<td>Time</td>
</tr>
<tr>
<td>6</td>
<td>Service times</td>
<td>Storage</td>
<td>Time</td>
</tr>
<tr>
<td>7</td>
<td>Accessibility in the case of problem</td>
<td>Storage</td>
<td>Yes/no</td>
</tr>
<tr>
<td>8</td>
<td>Backup</td>
<td>Storage</td>
<td>Time</td>
</tr>
<tr>
<td>9</td>
<td>Bytes per second</td>
<td>Storage</td>
<td>Number per second</td>
</tr>
<tr>
<td>10</td>
<td>Memory size</td>
<td>Storage</td>
<td>Number in bytes</td>
</tr>
<tr>
<td>Description</td>
<td>Position</td>
<td>Task</td>
<td></td>
</tr>
<tr>
<td>------------------------------</td>
<td>-----------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>Service Desk</td>
<td>Function</td>
<td>Group of specialists, inquiry -, treatment of disturbances</td>
<td></td>
</tr>
<tr>
<td>Incident Management</td>
<td>Process</td>
<td>Support user, problem acceptance, assistance, monitoring service level</td>
<td></td>
</tr>
<tr>
<td>Problem Management</td>
<td>Process</td>
<td>Treatment of losses, cause identifying, recommendations at Change Mgmt., improvement of productive resources use</td>
<td></td>
</tr>
<tr>
<td>Configuration Management</td>
<td>Process</td>
<td>Process control of the inventory (components hard -, software....)</td>
<td></td>
</tr>
<tr>
<td>Change Management</td>
<td>Process</td>
<td>Change process</td>
<td></td>
</tr>
<tr>
<td>SLM</td>
<td>Process</td>
<td>Formulate SLA</td>
<td></td>
</tr>
<tr>
<td>Release Management</td>
<td>Process</td>
<td>Storage of authorized software, release in productive environment, distribution to remote bases, implementation to start-up</td>
<td></td>
</tr>
<tr>
<td>Capacity Management</td>
<td>Process</td>
<td>Correct and cost-related-justifiable IT capacity provision analysis, prognosis; Capacity plans</td>
<td></td>
</tr>
<tr>
<td>Availability Management</td>
<td>Process</td>
<td>Optimization IT resources use, foreseeing and calculation of losses, safety guidelines monitoring SLAs, Security, Serviceability, Reliability, Maintainability, Resilience</td>
<td></td>
</tr>
<tr>
<td>Service-Continuity-Management</td>
<td>Process</td>
<td>Re-establishment of services, replacement in case of failure</td>
<td></td>
</tr>
<tr>
<td>Financial Management</td>
<td>Process</td>
<td>Process investment strategy, definition that-achievement-aims, those-brought achievement to measurement</td>
<td></td>
</tr>
</tbody>
</table>
## ITIL Service Metrics

<table>
<thead>
<tr>
<th>ITIL Process</th>
<th>Service Metrics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service Desk</td>
<td>Customer satisfaction with the Help Desk</td>
</tr>
<tr>
<td>Incident Management</td>
<td>Time between loss and replacement</td>
</tr>
<tr>
<td>Problem Management</td>
<td>Number of repeated disturbances</td>
</tr>
<tr>
<td>Configuration Management</td>
<td>Time between adding configuration items to Configuration Management Data Base (CMDB)</td>
</tr>
<tr>
<td>Change Management</td>
<td>Number of untreated changes</td>
</tr>
<tr>
<td>Service-Level Management</td>
<td>Number of SLAs</td>
</tr>
<tr>
<td>Release Management</td>
<td>Time between releases</td>
</tr>
<tr>
<td>Capacity Management</td>
<td>Completion of the capacity plan at a fixed time</td>
</tr>
<tr>
<td>Availability Management</td>
<td>Completion of the availability plan at a fixed time</td>
</tr>
<tr>
<td>IT-Service-Continuity-Management</td>
<td>Completion of the contingency plan at a fixed time</td>
</tr>
<tr>
<td>Financial Management</td>
<td>Cost overview to the deadline</td>
</tr>
</tbody>
</table>
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 \(a_i() \rightarrow a_{i-1}()\). chains(n) starts with fact \(a_0\) and continues with a chain of \(n\) (strict) rules of the form \(a_i() \rightarrow a_{i-1}()\). A goal \(a_n\) will use all of \(n\) rules and the fact.
    
    \[
    \text{chain}(n) = \begin{cases} 
    a_n \leftarrow a_{n-1} \\
    \vdots \\
    a_2 \leftarrow a_1 \\
    a_1 \leftarrow a_0 \\
    a_0 \quad \% \text{ fact}
    \end{cases}
    \]

  - **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.
    
    \[
    \text{dag}(n,k) = \begin{cases} 
    a_0 \leftarrow a_1, a_2, ..., a_k \\
    a_1 \leftarrow a_2, a_3, ..., a_{k+1} \\
    \vdots \\
    a_{nk} \leftarrow a_{nk+1}, a_{nk+2}, ..., a_{nk+k} \% \text{ rules} \\
    a_{nk+1}, a_{nk+2}, ..., a_{nk+k} \% \text{ facts}
    \end{cases}
    \]
Test Theory Examples

- Loop Checking (logic-formal adequacy)

\[
circles(n) \text{ consists of } n \text{ rules } a_i(): \leftarrow a_{i-1}() \text{ and a circular rule } \begin{align*} a_n & : \leftarrow a_{n-1}. \\
& \vdots \\
\end{align*} \\
circles(n) = \begin{align*} a_2 & : \leftarrow a_1. \\
a_1 & : \leftarrow a_0. \\
a_0 & : \leftarrow a_n. \\
a_0 & \text{ % fact} \\
\end{align*}
\]

- 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) = \text{rule}(a_0,n,k) \text{ where, if } p \text{ is a literal, } n>0, r \text{ is a new unique label, and } a_1,a_2,...,a_k \text{ are new unique literals:} \\
\[
\begin{align*}
p & \leftarrow a_1,a_2,...,a_k. \\
rules(a_1,n-1,k) \\
rules(p,n-1,k) = \\
& \vdots \\
rules(a_k,n-1,k) \\
\text{and: } & \text{rules(p,0,k) = p}
\end{align*}
\]

- Reactive Rules

\[
\text{reactive(n,t) consists of } n \text{ ECA rules } eca_n(t,e,c,a) \text{ which fire every interval } t.
\]

- Event Calculus

\[
\text{occurrence(n,m) consists of } m \text{ pairs of initiates / terminates Event Calculus rules and } n \text{ event occurrences (happens facts) which initiate resp. terminate a fluent.}
\]
## Experimental Performance Evaluation

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

### Performance Tests

<table>
<thead>
<tr>
<th>Test</th>
<th>Size</th>
<th>No Memoization</th>
<th>Memoization</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Strict (Propos. / Datalog)</td>
<td>Defeasible (Propos. / Datalog)</td>
</tr>
<tr>
<td>chains(n)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2001</td>
<td>11001</td>
<td>0.01 / 0.07</td>
<td>4 / 7.6</td>
</tr>
<tr>
<td>5001</td>
<td>27501</td>
<td>0.03 / 0.17</td>
<td>12.8 / 25</td>
</tr>
<tr>
<td>10001</td>
<td>55001</td>
<td>0.07 / 0.3</td>
<td>40 / 70</td>
</tr>
<tr>
<td>20001</td>
<td>110001</td>
<td>0.15 / 0.62</td>
<td>127 / 250</td>
</tr>
<tr>
<td>dag(n,k)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>n=3 k=3</td>
<td>39</td>
<td>0.01 / 0.06</td>
<td>0.54 / 0.89</td>
</tr>
<tr>
<td>n=4 k=4</td>
<td>84</td>
<td>2.2 / 7.7</td>
<td>81 / 120</td>
</tr>
<tr>
<td>n=10 k=10</td>
<td>1100</td>
<td>3810</td>
<td>- / -</td>
</tr>
<tr>
<td>tree(n,k)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>n=3 k=3</td>
<td>79</td>
<td>0.01 / 0.02</td>
<td>0.04/0.04</td>
</tr>
<tr>
<td>n=4 k=3</td>
<td>281</td>
<td>0.015/0.03</td>
<td>0.09/0.1</td>
</tr>
<tr>
<td>n=8 k=3</td>
<td>19681</td>
<td>0.17 / 0.5</td>
<td>- / -</td>
</tr>
<tr>
<td>@access(n)</td>
<td>1000</td>
<td>Update Time</td>
<td>Execution Time</td>
</tr>
<tr>
<td></td>
<td>2500</td>
<td>0.4</td>
<td>0.005</td>
</tr>
<tr>
<td></td>
<td>5000</td>
<td>1.1</td>
<td>0.01</td>
</tr>
<tr>
<td></td>
<td>10000</td>
<td>2.5</td>
<td>0.015</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4.3</td>
<td>0.02</td>
</tr>
<tr>
<td>@getuid(n)</td>
<td>1002</td>
<td>3.3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2502</td>
<td>6.8</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5002</td>
<td>14.6</td>
<td></td>
</tr>
<tr>
<td></td>
<td>10002</td>
<td>28.7</td>
<td></td>
</tr>
</tbody>
</table>
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
http://ibis.in.tum.de/projects/rbsla/index.php

https://sourceforge.net/projects/rbsla