Regelbasierte Service Level Agreements

Adrian Paschke
Roland Berger & O_2 Lehrstuhl für
Internetbasierte Geschäftssysteme
Institut für Informatik
TU München

Adrian.Paschke@in.tum.de
http://www.ibis.in.tum.de

Service Level Agreements

- Service Level Agreements (SLA)
  - Juristisch kein festehender Begriff
  - Beschreibt die vertraglichen Vereinbarungen zwischen einen IT-Dienstanbieter und einem Dienstnutzer (im IT-Dauerschuldverhältniss) insbesondere in Bezug auf Leistungs- und Qualitätsgarantien
  - Anwendungsgebiet:
    - Outsourcing (im speziellen ASP)
    - Netzbasierte Dienste: Hosted Services/Web Services
    - Grid Computing
  - Ziel: Klare Vereinbarungen hinsichtlich Leistungsumfang, Dienstqualität, Rechte und Pflichten, Kosten

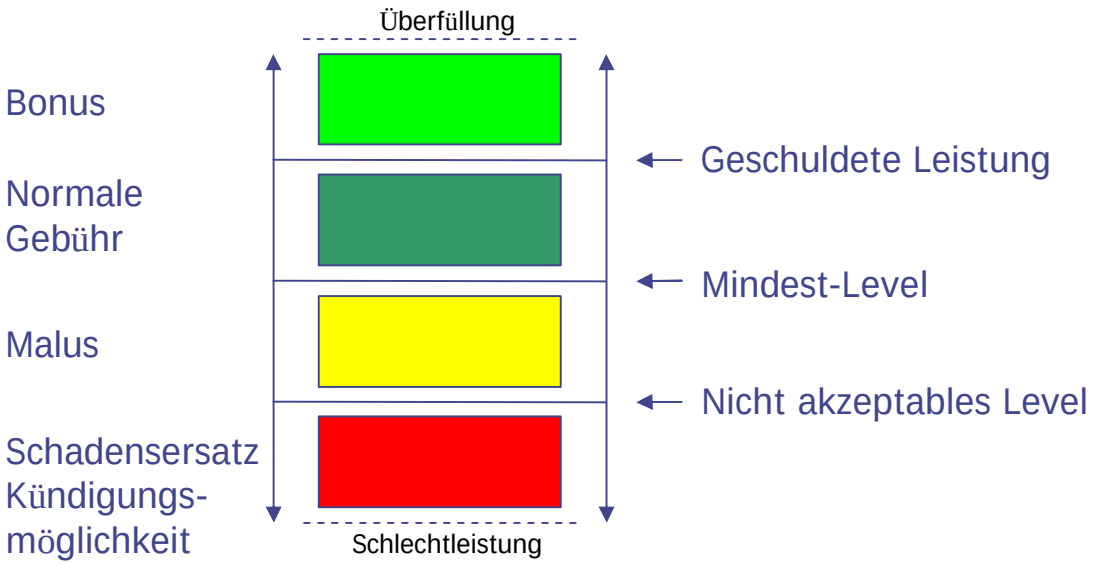
→ Vertrauensverhältniss zwischen Anbieter und Nachfrager (Rückversicherung)

Service Level Agreements

- Inhalte:
  - Hauptleistungspflichten / Gewährleistungspflichten
  - Vertragsbedingungen (Terms and Conditions) z.B.
    - Preispolitik (pricing policy)
    - Rabatte (discount policy)
    - Policies about creditworthiness, trustworthiness and authorization
  - Quantiative Standards
    - Leistungsumfang, Vereinbarte Quantität
  - Qualitative Standards:
    - Quality of Service Paramter (QoS)
      - Availability, Response time, Throughput
    - Messmethoden (Monitoring)
      - Measurement directives
    - Berichterstattung (Reporting)
    - Sanktionsregelungen
  - Gesetzliche Regelungen aus dem
    - Haftungsrecht
    - Urheberrecht
    - Datenschutzrecht
  - Event-Management
  - Change-Management
  - Exit-Management

Beispiel

- Pricing Policy



Service Level Agreements

- Inhalte sind in natürlicher Sprache definiert
- Enthalten Anweisungen in Form von meist impliziten Regeln (Business Rules)
- Regeln häufig nicht automatisiert, Einhaltung wird händisch überprüft.
- Im besten Fall: Regeln direkt in die Applikations-Logik codiert und automatisiert.
- Problem:
  - Modifikation und Wartung der Regeln

  → erfordert aufwändige und kostenintensiver Reimplementierung
  - Dynamische Regeln etc.
- Lösung:
  - Regeln extrahieren, formalisieren und separat verwalten
  - Regeln operationalisieren und automatisieren

Regelbeispiel

- Discount Policy

(1) The discount for a customer is 5.0 percent if the customer is low.
(2) The discount for a customer is 10 percent if the customer is medium.
(3) The discount for a customer is 15 percent if the customer is high.

(4) A customer is high if their spending has been more than 400 euro in the previous year.
(5) A customer is medium if their spending has been more than 200 euro in the previous year.
(6) A customer is low if their spending has been more than 100 euro in the previous year.

Fragestellung

Welche Art der Wissensrepräsentation ist geeignet, um Regeln in elektronischen Verträgen (SLAs) adäquat beschreiben zu können?

Prämisse: Ausführbare Regeln

Wissensrepräsentation

- Das Repräsentationsproblem
  - Wie soll das Wissen, die Problembeschreibung und die Lösung(en) dargestellt werden?
  - Die Mächtigkeit der Repräsentationsform beeinflusst die Effizienz und Leistungsfähigkeit des Systems.
  - Ohne adäquate Repräsentation keine adäquate Lösung.
- Das Suchproblem
  - Wie "manöveriere" ich von einer (Teil-)Lösung zu einer anderen?
  - Wie vermeide ich, Lösungen mehrfach zu "besuchen"?
- Das Inferenzproblem
  - Welches Wissen verwende ich zur Problemlösung?
  - Welches Wissen wende ich wie auf mein vorhandenes Wissen an?

Wissensrepräsentation

- Repräsentation:
  - Beziehung eines Begriffs zu einem von diesem Begriff erfaßten oder umfassten Objekt der Realität
- Repräsentationsproblem:
  - Adäquate Abbildung strukturell-operativer Zusammenhänge in eine (beschränkte) Sprache
- Lösungsansätze:
  - Sprachliche Darstellung von Wissen, bspw. in Expertensystemshells, KI-(Prolog, Lisp) oder Wissensrepräsentationssprachen (KL-One u.ä.)
- Bewertung:
  - mittels Adäquatheitskriterien

Adäquatheitskriterien

- Epistomologische Adäquatheit
  - Ausdrucksstärke: können alle relevanten Fakten und Regeln repräsentiert werden?
  - Welche sind abbildbar, welche nicht?

"A representation is called epistomologically adequate for a person or a machine if it can be used practically to express the facts that one actually has about the aspects of the world„

McCarthy/Hayes 1969

- Heuristische Adäqutheit
  - Sind die durchzuführenden Inferenzen überhaupt bzw. mit den zu Verfügung stehenden Ressourcen möglich?
  - Sind Zwischenschritte des Lösungswegs im Formalismus repräsentierbar?

- Algorithmische Adäquatheit
  - Gibt es Algorithmen mit angemessener bzw. tolerierbarer Komplexität?
  - Sind die sprachlichen Konstrukte effizient verarbeitbar?

- Logisch-formale Adäquatheit
  - Ist das System (bestehend aus Sprache und Verarbeiter) korrekt und vollständig?
  - Welche Klasse von Problemen ist entscheidbar, welche nicht?

Adäquatheitskriterien

- Psychologische Adäquatheit
  - Macht das System die selben Fehler, die auch ein Mensch in dieser Situation machen würde?
  - Erlaubt das System Widersprüche?
- Ergonomische Adäquatheit
  - Ist die Sprache leicht handhabbar?
  - Sind die Konstrukte verständlich?
  - Klarheit und Präzision, leichte Veränderbarkeit (Woods, 1987)
- Zusammenfassung

  „There is a tradeoff between the expressivness of a representational language and its computational tractability.“

  (Levesque, Brachman 1987)

  - Auswahl einer Wissensrepräsentationsform ist kritisch, da die Performanz des Systems dadurch beeinflusst wird.
  - Je nach Ziel sind unterschiedliche Repräsentationsformen angebracht.
  - Adäquatheitskriterien dienen der Beurteilung der Repräsentationsformen entsprechend des jeweiligen Ziels.
  - Tool-Unterstützung

Logikarten

- Klassische Logik (Monotone Logiken), z.B.
  - Aussagenlogik (Propositional Logic)
  - Prädikatenlogik (First Order Logic)
  - Horn Logik
- Nicht-klassische Logik (Nicht-monotone Logiken)
  - Mehrwertige Logik, z.B.
    - Modale Logik ("notwendig\,"möglich\, . . .)
    - Temporale Logik (zusätzlich:"vorher\,"nachher\,"nicht gleichzeitig\,"während\)
    - Deontische Logik ("erlaubt\,"verboten\, . . .)
    - Epistemische Logik („glauben“ \ „wissen“)
- Taxonomien/Ontologien
  - RDFS/OWL
  - Description Logic
  - F-Logic

Regeln

- Regeln sind Erweiterungen der logischen Formalismen
- Regeln können logisch interpretiert werden
- Regeltypen
  - Reaction Rules (ECA-rules)
  - Derivation Rules
  - Integrity Constraints
- Forward vs. Backward Reasoning
- Regelsprachen
  - Datalog
  - Prolog
  - Weitere, z.B. RuleML (Mandarax RuleML und Mandarax XKB/ZKB)
- Rule Engines
  - Clips/Jess Forward (Rete)
  - Mandrax Backward

Regelsprache Anforderungen

- Anforderung:
  - Expressiveness <-> Efficiency/ Computational Tractability and Tool Support
  - Rule Chaining / Composing
  - Modularity, Rule inheritance
  - Rule Conflict Handling / Prioritization/ Default rules
    - Conflict resolution, assigning priorities to the rules
  - Support of large sets of facts (a.k.a Clause Sets/Autofacts)
  - Allow facts to be retrieved from secondary data storage sources
  - Support complete and incomplete information (negation-as-failure, strong negation)
  - Support complex structures (Object, Functions etc.)
  - Support typed clauses (by Ontologies / Programming Languages)
  - Allow events /actions and procedural attachments
  - Complex event processing
  - Support deontic norms which can be violated and conflicting (exceptions)

RuleML

- RuleML
  - http://www.dfki.uni-kl.de/ruleml/
  - Offenes Netzwerk von Leuten und Gruppen aus Industrie und Forschung
  - Das Ziel der RuleMarkup Initiative ist es RuleML als die Web-Sprache zur Beschreibung von Regeln zu entwickeln.
  - RuleML basiert auf XML bzw. RDF
  - RuleML soll in Zukunft das gesamte Regelspektrum umfassen (jeweils mit eigener DTD)

RuleML Beispiel

```
<!--  A customer is medium if their turnover has
      been more than 100 euro in the previous year.
-->

<imp>
  <_head>
    <atom>
      <_opr><rel>medium</rel></_opr>
      <var>customer</var>
    </atom>
  </_head>
  <_body>
    <atom>
      <_opr><rel>more than</rel></_opr>
      <var>turnover(customer,previous year)</var>
      <ind>100 euro</ind>
    </atom>
  </_body>
</imp>
```

```
<!--  The discount for a customer is 5.0
percent if the customer is medium.
-->

<imp>
  <_head>
    <atom>
      <_opr><rel>gets</rel></_opr>
      <var>customer</var>
      <ind>discount[5.0 percent]</ind>
    </atom>
  </_head>
  <_body>
    <atom>
      <_opr><rel>silver</rel></_opr>
      <var>customer</var>
    </atom>
  </_body>
</imp>
```

Mandarax

- Java based rule engine
- Derivation rules (backward reasoning)
- Terms:
  - Constant terms
  - Variable terms
  - Complex terms
- Terms sind typisiert
- Unterstüzt Semantiken und Rule Priorities
- Unterstützt RuleML, aber
  - Typing
  - Complex terms and functions
  - Clause sets
  - Integration von (SQL) Datenquellen
  - Regeln mit OR-Verknüpfung

RBSLA: Rule-Based SLA Framework

- Verknüfpung von Ontologien (Description Logic) und RuleML (Logic Programming / Horn Logic).
  - Tools: Mandarax und Jena2
- ECA-Regeln durch Event Listeners / Thread based Daemon
  - Tools: Mandarax ECA / Meta Interpreter Simulation
- Temporale und Event Erweiterungen (Event Processing)
  - Event Calculus
- Deontische Erweiterungen
  - Defeasible Deontic Logic mit Violations (Contrary-to-Duty) und Exceptions (Defeasible Prima Facie)
- Regelkonflikte und Präzedenzordnungen für Regeln
  - Defeasible Logic und Courteous Logic Programs mit Mutex