

# Repräsentation und automatisiertes Management von IT-Service-Verträgen

**Ein Ansatz zum Einsatz wissensbasierter Systeme für die Repräsentation, Automatisierung und das Management elektronischer IT-Dienstleistungsverträge**

Autor  
Dipl. Wirtsch.-Inf. Adrian Paschke  
Internetbasierte Geschäftssysteme (IBIS)  
Roland Berger & o2 Germany Stiftungslehrstuhl  
Institut für Informatik, I18, TU München  
Boltzmannstraße 3  
85748 Garching, Deutschland  
[Adrian.Paschke@in.tum.de](mailto:Adrian.Paschke@in.tum.de)

Betreuer  
Prof. Dr. Martin Bichler  
Internetbasierte Geschäftssysteme (IBIS)  
Roland Berger & o2 Germany Stiftungslehrstuhl  
Institut für Informatik, I18, TU München  
Boltzmannstraße 3  
85748 Garching, Deutschland  
[bichler@in.tum.de](mailto:bichler@in.tum.de)

## Kurzzusammenfassung

Vereinbarungen in IT-Dienstleistungsverträgen (engl. Service Level Agreement, SLA) werden heute zum Teil über Parametereinstellungen in Service-Level-Management-Werkzeugen überprüft. Die automatisierte Überwachung von SLA-Regeln stellt allerdings derzeit in der Praxis ein verbreitetes Problem dar. Komplexere Vereinbarungen mit Qualitäts- und Preisregeln, wie sie heute oft für verbrauchsabhängige IT-Dienstleistungen erforderlich sind (engl. utility computing), können derzeit nicht oder nur über eine aufwändige Zusatzimplementierung umgesetzt werden. In unserem Ansatz setzen wir fortgeschrittene Arten der Wissensrepräsentation ein, die es erlauben auch komplexe Vertragsregeln einfacher und flexibler umzusetzen. Neben klassischen Verfahren aus der logischen Programmierung (LP) ist zur Umsetzung eines flexiblen Vertragsmanagements die Erweiterung um ECA-Regeln, deontische Logik, temporale Erweiterungen sowie Ontologiebeschreibungssprachen nötig, wie sie in heutigen Inferenzmaschinen nicht angeboten wird. Die wichtigsten Vorteile des Ansatzes liegen in der automatisierten Konsistenzprüfung großer Regelmengen, der automatisierten Verknüpfung unterschiedlicher Regelmengen (z.B. in Gruppen- und Individualverträgen), sowie der Flexibilität bei der Erweiterung um neue Klauseln und Vertragsregeln. Das daraus entstehende logische **ContractLog** Framework wurde auf Basis der Open-Source Regelmaschine Mandarax implementiert und dient als Grundlage für unser „**Regel-basierte Service Level Management-Werkzeug**“ (**RBSLM**) und die von uns entwickelte deklarative Sprache, die „**Rule-based SLA**“ (**RBSLA**) zur generischen Beschreibung von SLAs. Zur Evaluierung wurden verschiedene Vertragsregeln in neueren SLA-Standardisierungsvorschlägen und mit RBSLA umgesetzt, um die Vor- und Nachteile des Konzeptes zu illustrieren.

Schlüsselwörter: IT-Service-Management, Contract-Management, Logische Programmierung, Wissensrepräsentation, Service-Level-Agreement

## 1. Ausgangssituation und Problemstellung

Service-Level-Agreements (SLAs) nehmen in der Beziehung zwischen IT-Diensteanbietern und Dienstnehmern eine zentrale Rolle zur bedarfsgerechten Sicherstellung geschäftlicher Kontinuität und Risikoabsicherung ein. Neben quantitativen Vereinbarungen über Art und Umfang der Nutzung eines oder mehrerer IT-Dienste finden sich in SLAs insbesondere auch qualitative Leistungsparameter (Verfügbarkeit, Antwortzeit, etc.), die um Regeln zur Überwachung, Vertragsstrafen und Art und Häufigkeit der Berichterstattung ergänzt werden. Utility-Computing beschreibt den Trend, dass IT-Dienstleistungen zunehmend nutzungsabhängig abgerechnet werden. Neben Qualitätsparametern werden dadurch zunehmend auch Preisregeln teil der SLAs. IT-Dienstleister sehen sich in einem solchen Szenario vor der Aufgabe, eine Vielzahl individuell oder auf Gruppenebene vereinbarter Verträge zu ihren Endkunden mit unterschiedlichsten Leistungs- und Qualitätzusicherungen verwalten und überwachen zu müssen.

Die in der Praxis vorherrschende, natürlichsprachige Vertragsbeschreibung und deren weitgehende manuelle Überwachung werden diesen Anforderungen nicht gerecht. Derzeitige Service-Management-Werkzeuge, wie z.B. der Tivoli SLA oder Remedy SLA bieten lediglich die Messung von Leistungsparametern wie Verfügbarkeit und Antwortzeit bestimmter IT-Ressourcen. Allerdings beschränkt sich ein derartiger Ansatz auf einfache, meist fest vorgegebene Parameter und erfordert eine aufwändige Neuimplementierung bei veränderten Anforderungen. Komplexere Regelzusammenhängen und differenzierte Geschäftspolitiken können nicht adäquat dargestellt werden.

## 2. Zielsetzung der Arbeit

Service-Level-Management ist einer der wichtigsten Bausteine von IT-Service-Management-Standards wie ITIL. Besonders durch die verstärkte Forderung nach individualisierten und differenzierten Vertragsmodelle in den aufkommenden On-Demand-/Utility-Computing Geschäftsmodellen werden vertragliche Vereinbarungen zwischen IT-Dienstleistern und Dienstnehmern zunehmend komplexer. Eine fehlende automatisierte Überwachung und Verrechnung solcher Vereinbarungen stellt in der Praxis zunehmend ein Problem dar. Regeln in verschiedenen Vertragsmodulen müssen miteinander verknüpft und automatisiert auf Konsistenz geprüft werden können. Das Service-Management-Werkzeug, das zur Überwachung der IT-Dienste eingesetzt wird, sollte in der Lage sein, Regeln in verschiedenen, auch individuell vereinbarten Verträgen umsetzen zu können. In derzeitigen IT-Service-Management-Werkzeugen ist das nur durch aufwändige Zusatzimplementierungen möglich.

Die grundlegende Frage in dieser Arbeit ist, ob fortgeschrittene Konzepte aus der Wissensrepräsentation durch den Einsatz generischer Inferenzmaschinen eine weitgehende Automatisierung des SLA-Managements ermöglichen. Wir schlagen dazu einen alternativen Ansatz vor, in dem Verträge durch eine deklarative Wissensrepräsentation basierend auf verschiedenen logischen Formalismen dargestellt und mittels Standardkomponenten aus der logischen Programmierung interpretiert und zur Ausführung gebracht werden. Die wesentlichen Kernpunkte unseres Ansatzes sind:

- Repräsentation von Vertragsregeln in formaler Logik, sowie Trennung der Vertragsregeln von der Anwendungslogik der IT-Service-Management-Werkzeuge. Verträge werden in modularen Vertragsbausteinen (allgemeine Geschäftsbedingungen, Verträge für einzelne oder Gruppen von Nutzern) gewartet und von generischen Inferenzmaschinen interpretiert und erlauben dadurch eine einfachere Pflege und Verwaltung von Verträgen.
- Vertragsregeln können automatisch verknüpft und durch Standardkomponenten aus der logischen Programmierung (Regelmaschinen) ausgeführt werden. Dies ermöglicht komplexe Geschäftspolitiken und individuelle, differenzierte Übereinkünfte zu modularisieren und deren Interpretation zu automatisieren.
- Automatisierte Verifikation von Regeln / Regelmodulen und Erkennen von Regelkonflikten.
- Automatisierte Erkennung von Vertragsverletzungen bei der Ausführung von IT-Dienstleistungen und automatische Reaktion durch ECA-Regeln.
- Sicherstellung der Vertragseinhaltung mittels vertraglichen Normen, die durch deontische Konzepte wie "Erlaubnis" (Permission), "Verbot" (Prohibition), "Verpflichtung" (Obligation) beschrieben und über entsprechende Erweiterungen der Inferenzmaschine überprüft werden können.
- Integration von semantisch definierten Domänenbeschreibungen, wie Vertragsontologien, Rollen- oder Systemmodellen durch Standards aus dem Semantic-Web. Dadurch können Vertragsregeln domänen-unabhängig gehalten und leichter über Anwendungs- und Geschäftsdomänen ausgetauscht werden. Zusätzlich wird die Transformation der geschäftsbezogenen Regelungen eines SLAs in untergeordnete operationale Regelmodule unterstützt.

### 3. Inhaltliche und methodische Vorgehensweise

Basierend auf der funktionalen und nicht-funktionalen Anforderungsanalyse sowie allgemeiner Adäquatheitskriterien (wie epistemologische, algorithmische, logisch-formale und ergonomische Adäquatheit) aus der Künstlichen Intelligenz werden geeignete Formen der Wissensrepräsentation für die Abbildung der vertraglichen Regelungen bewertet und zu einem adäquaten logischen Framework – dem **ContractLog** – sowie dem darauf aufsetzenden Sprachkonzept - der **RBSLA** – zur formalen Beschreibung von SLAs zusammengefasst, welche wiederum die Basis für das regel-basierte Service Level Management (**RBSLM**) Werkzeug ist.

Verschiedenste Formen der Logik (z.B. Boolesche Logik, Prädikatenlogik, Modale Logik, Temporale Logik, Beschreibungslogiken, etc.) stehen hier zur Verfügung um die Semantik von Geschäftsregeln in elektronischen Verträgen zu beschreiben. Dabei hat sich für uns die Horn Logik und die auf ihr basierenden Deduktionsregeln in mehrfacher Weise als geeignete Grundlage für die Repräsentation von Vertragsregeln erwiesen, u.a.:

- Horn Logik ist ein berechenbare, vollständige und weitgehend ausdrucksstarke Teilklasse der Prädikatenlogik mit Unterstützung von Quantoren, Variablen, Funktionen etc.
- Für die Deduktion von Horn Regeln existieren in der logischen Programmierung Standardkomponenten, so genannte Inferenzmaschinen (engl. Rule Engines), die unter anderem eine Regelverkettung unterstützen.
- Die Horn Logik erlaubt eine Vielzahl typischer Vertragsregeln, wie sie in SLAs vorkommen, einfach zu repräsentieren.

Allerdings genügt Horn Logik nicht als alleiniges Mittel zur vollständigen Ableitung formaler Vertragsspezifikationen, sondern muss um weitere Logikkonzepte, insbesondere nicht-monotone Logiken, ergänzt werden. Zu den wesentlichen Erweiterungen, die wir in unserem ContractLog-Konzept einsetzen, gehören:

Logik	Nutzung
Horn Logik (engl. derivation rules)	Basis für deduktives Schließen in Geschäftsregeln
Event-Condition-Action-Regeln (ECA)	Aktive Ereigniserkennung und aktives Verhalten durch ausführbare Aktionen
Temporale Logik und Event-Calculus	Ermöglicht temporale Schlussfolgerung über dynamische Systeme, wie z. B. Effekte von Ereignissen/Aktionen auf den Vertragszustand
Defeasible Logik (Priorisierung)	Default-Regeln und Präzedenzordnungen von Regeln/Regelsets dienen als Basis für die Vertragsmodularisierung, Regelvererbung und Regelkonfliktbehandlung
Deontische Logik	Ermöglicht normative Konzepte wie „Permission“, „Prohibiton“, „Obligation“ zur Steuerung des erlaubten Verhaltens auf Basis von Vertragsnormen sowie Verletzungen (Violations: contrary-to-duty obligations) und Ausnahmen (Exceptions: defeasible prima facie deontic norms).
Beschreibungslogik (engl. description logic)	Semantische Domänenbeschreibung zur Integration in domänen-unabhängige Vertragsregeln mit Hilfe von Semantic-Web-Standards
Prozedurale Logik: Prozedurale Anhänge (Java) und Type Logik	Prozedurale Anhänge (engl. procedural attachments) erlauben die Integration von objekt-orientierten Programmiersprachen für optimierte Berechnungen und nicht in deklarativer Logik darstellbarer Funktionen. Dies erlaubt die Vorteile der deklarativen Programmierung mit den Vorteilen der prozeduralen Programmierung zu vereinen.

Tabelle 1: Logisches Konzepte zur Repräsentation von IT-Verträgen zusammengefasst in ContractLog

Tabelle 2 zeigt zur Verdeutlichung unseres Ansatzes typische Regelbeispiele, wie sie in SLAs vorkommen, zusammen mit einer entsprechenden Formalisierung in unserem ContractLog Konzept.

<p>Beispiel: ECA</p> <p>“If the service is unavailable then send a notification to the service provider. The service availability will be measured every minute by a service ping.”</p> <p><math>timer(minute, T) \leftarrow time(T) \wedge second(T) \bmod 60 = 0</math>  <math>event(unavailable) \leftarrow \neg ping(service)</math>  <math>action(notify) \leftarrow sendMessage(SP, \text{“service unavailable”})</math>  <math>eca(check) \leftarrow timer(minute, T) \wedge event(unavailable) \wedge action(notify)</math></p>	<p>Beispiel: Horn Logik / Regelverkettung</p> <p>“A customer gets a discount of <math>p_{gold}</math> if he is a gold customer”</p> <p><math>discount(Customer, p_{gold}) \leftarrow gold(Customer)</math>  <math>discount(Customer, p_{silver}) \leftarrow silver(Customer)</math>  <math>gold(Customer) \leftarrow spending(Customer, &gt;5000\text{€}, last\ year)</math>  <math>silver(Customer) \leftarrow spending(Customer, &gt;3000\text{€}, last\ year)</math></p>												
<p>Beispiel: Deontische Logik</p> <p>“The service consumer (SC) is permitted to use the service (Service) operations read, store, delete.”</p> <p><math>permit(SC, Service, store())</math>  <math>permit(SC, Service, read())</math>  <math>permit(SC, Service, delete())</math></p>	<p>Beispiel: Event Calculus</p> <table border="0"> <tr> <td><math>initiates(ei, f, T)</math></td> <td>Event <math>ei</math> initiates Fluent <math>f</math></td> </tr> <tr> <td><math>terminates(et, f, T)</math></td> <td>Event <math>et</math> terminates Fluent <math>f</math></td> </tr> <tr> <td><math>happens(ei, t0)</math></td> <td>Event <math>ei</math> happens at <math>t0</math></td> </tr> <tr> <td><math>happens(et, t3)</math></td> <td>Event <math>et</math> happens at <math>t3</math></td> </tr> </table> <table border="0"> <tr> <td>Query: <math>holdsAt(f, t1)?</math></td> <td>Fluent <math>f</math> is true at timepoint <math>t1</math></td> </tr> <tr> <td>Query: <math>holdsAt(f, t4)?</math></td> <td>Fluent <math>f</math> is false at timepoint <math>t4</math></td> </tr> </table>	$initiates(ei, f, T)$	Event $ei$ initiates Fluent $f$	$terminates(et, f, T)$	Event $et$ terminates Fluent $f$	$happens(ei, t0)$	Event $ei$ happens at $t0$	$happens(et, t3)$	Event $et$ happens at $t3$	Query: $holdsAt(f, t1)?$	Fluent $f$ is true at timepoint $t1$	Query: $holdsAt(f, t4)?$	Fluent $f$ is false at timepoint $t4$
$initiates(ei, f, T)$	Event $ei$ initiates Fluent $f$												
$terminates(et, f, T)$	Event $et$ terminates Fluent $f$												
$happens(ei, t0)$	Event $ei$ happens at $t0$												
$happens(et, t3)$	Event $et$ happens at $t3$												
Query: $holdsAt(f, t1)?$	Fluent $f$ is true at timepoint $t1$												
Query: $holdsAt(f, t4)?$	Fluent $f$ is false at timepoint $t4$												

**Tabelle 2: Typische Regelbeispiele und entsprechende logische Formalisierung**

Im Rahmen des Projektes wurden die oben beschriebenen Konzepte auf Basis der Open-Source-Inferenzmaschine Mandarax implementiert und zu einer prototypischen Vertrags- und Service-Level-Managementanwendungen, der RBSLM) erweitert. Diese Implementierung dient uns als Proof-of-Concept-Implementierung mit der wir zum einen die Adäquatheit unseres ContractLog Frameworks und der RBSLA zur Repräsentation typischer Realwelt-SLAs, wie wir sie mit Praxispartnern erarbeitet haben, zeigen und zum anderen die Vor- und Nachteile unseres Ansatzes gegenüber den in der Praxis verwendeten IT-Service-Management-Werkzeugen aufzeigen.

### Wissenschaftstheoretischer Ansatz

Im Gegensatz zu zahlreichen empirisch orientierten Arbeiten in der Wirtschaftsinformatik der letzten Jahre folgt diese Arbeit einer konstruktivistischen, ingenieurwissenschaftlichen Vorgehensweise. Im Rahmen von Experimenten soll die Qualität des vorgeschlagenen Konzeptes in Bezug auf Flexibilität und Performanz der Implementierung evaluiert werden. Daneben wurden bereits detaillierte Vergleiche mit alternativen Ansätzen, die derzeit in einigen Forschungslabors (z.B. IBM WSLA) sowie an verschiedenen universitären Einrichtungen entwickelt werden, angestellt.