Electronic Communications of the EASST
Volume 17 (2009)

Workshops der Wissenschaftlichen Konferenz Kommunikation in Verteilten Systemen 2009 (WowKiVS 2009)

Selbstorganisierendes Service Level Management basierend auf Mechanismus-Design

Bernhard Jungk

12 pages

Guest Editors: M. Wagner, D. Hogrefe, K. Geihs, K. David
Managing Editors: Tiziana Margaria, Julia Padberg, Gabriele Taentzer
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

Selbstorganisierendes Service Level Management basierend auf Mechanismus-Design

Bernhard Jungk[1]

[1]Fachhochschule Wiesbaden - University of Applied Sciences,
Labor für Verteilte Systeme,
Kurt-Schumacher-Ring 18, D-65197 Wiesbaden
jungk@informatik.fh-wiesbaden.de,
http://wwwvs.informatik.fh-wiesbaden.de

Abstract: Immer komplexere IT-Infrastrukturen führen zu immer komplexeren IT-Managementsystemen. Selbstorganisierende Systeme stellen einen möglichen Ansatz zum Umgang mit der zunehmenden Komplexität dar. Solche Systeme konfigurieren sich selbst, optimieren automatisch die Leistung des Systems und reduzieren dadurch die sonst nötigen manuellen Eingriffe. In diesem Beitrag wird basierend auf Mechanismus-Design, eine Teildisziplin der Spieltheorie, ein selbstorganisierendes Managementsystem für Service-orientierte Systeme beschrieben. Das beschriebene, sogenannte SLO-Spiel, wird theoretisch sowie mit Hilfe von Simulationen untersucht und bewertet. Die Ergebnisse zeigen eine prinzipielle Eignung des Mechanismus für das Management, wobei allerdings weitere Verbesserungen für den realen Einsatz untersucht werden müssen.

Keywords: Service-Oriented Architecture, Self-Organisation, Mechanism Design

1 Einführung

Die Größe und Komplexität von IT-Systemen, sowie die Abhängigkeit vieler Unternehmen von diesen nimmt immer weiter zu. Deshalb wird es immer wichtiger ein durchgängiges IT-Management über System- und Unternehmensgrenzen hinweg einzuführen. Um dieses Ziel zu erreichen, wurde das Konzept des Service Level Managements (SLM) eingeführt. Ein zentrales Element des SLM ist ein sogenanntes Service Level Agreement (SLA), welches einen Vertrag zwischen Anbieter eines Services und dessen Nutzer darstellt. Dieser Vertrag definiert für den angebotenen Service bestimmte Leistungskriterien in Form von Service Level Objectivs (SLO), an deren Einhaltung sich der Service-Anbieter bindet (vgl. [Deb04]).

Um die Dienstgütekriterien eines SLAs einhalten zu können, muss auf Ereignisse in der IT-Infrastruktur durch geeignete Konfigurationsänderungen für die einzelnen Teilsysteme reagiert werden. Manuelles Management stößt hier durch die zunehmende Größe, Komplexität und erforderlichen kurzen Reaktionszeiten an Grenzen. Sogenanntes Selbstmanagement kann dieses Management in Teilbereichen automatisieren und löst damit das manuelle Management ab (vgl. [SK05]).

Die zunehmende Komplexität der Systeme steigert auch den zur Konfiguration des Managementsystems nötigen Aufwand, weshalb man aktuell die Entwicklung von selbstorganisierenden

Systemen vorantreibt (vgl. [MWJ+07]). Diese Systeme verändern nicht nur Konfigurationsparameter, sondern auch die Struktur des Managementsystems. Bisherige Managementsysteme sind im Gegensatz dazu allerdings meist zentraler und statischer Natur (z.B. [YBT05]) und besitzen deshalb eine Reihe von Nachteilen, z.B. Probleme bei der Skalierbarkeit oder geringe Fehlertoleranz bei Partitionierungen der Kommunikationswege.

Ein möglicher Ansatz für selbstorganisierende Managementsysteme basiert auf dem algorithmischen Mechanismus-Design, einem Teilgebiet der Spieltheorie (vgl. [NRTV07]). Modelliert werden sogenannte rationale Agenten, die stets versuchen den eigenen Nutzen zu maximieren. Dieses Verhalten ist beispielsweise zu beobachten, wenn das Management über administrative Domänengrenzen hinweg erfolgen soll. Jede Domäne versucht das Management üblicherweise so zu gestalten, dass ihr eigener Nutzen optimiert wird. Weiterhin wird durch das algorithmische Mechanismus-Design die algorithmische Komplexität der entworfenen Mechanismen untersucht, damit ein entworfener Mechanismus auch praktisch umsetzbar ist.

Eine parallel zur Entwicklung immer komplexerer Managementsysteme verlaufende Entwicklung, ist die zunehmende Verbreitung von Service-orientierten Systemen (vgl. [Vas07]). Ein Kernbestandteil solcher Systeme sind Workflows, die bestimmte Geschäftsprozesse eines Unternehmens modellieren. Ein Workflow besteht aus verschiedenen Aktivitäten, die in einem gerichteten Graphen angeordnet sind.

Dieser Beitrag beschreibt einen selbstorganisierenden Mechanismus für das SLM von solchen Service-orientierten Systemen auf Basis des algorithmischen Mechanismus-Designs. Dieser Mechanismus soll immer dann aktiv werden, wenn bestimmte Leistungskriterien eines Services nicht eingehalten werden. Prinzipiell eignet sich dieser Ansatz für viele unterschiedliche Leistungskriterien, allerdings betrachtet der Beitrag die Antwortzeit als einziges Leistungskriterium, um die prinzipielle Eignung zu zeigen.

In Abschnitt 2 wird das Modell eines Service-orientierten Systems beschrieben, wie es im weiteren Verlauf des Beitrags genutzt wird. Außerdem werden mögliche Managementaktionen innerhalb dieses Systems aufgezeigt. Abschnitt 3 definiert das verwendete Modell des Mechanismus-Designs, welches genutzt wird, um den Mechanismus, das sogenannte SLO-Spiel, zu entwerfen (Abschnitt 4). Im weiteren Verlauf werden theoretische Ergebnisse vorgestellt und das SLO-Spiel mittels verschiedener Simulationen bewertet (Abschnitt 5). Abschnitt 6 gibt eine Zusammenfassung und einen Ausblick über weitere ergänzende Arbeiten, damit der vorgestellte Mechanismus alle gängigen Anforderungen erfüllt.

2 System-Modell

Im Folgenden werden ein Service-orientiertes System und seine Eigenschaften beschrieben. Formal wird dieses Modell in [Jun08] definiert. Ein solches System wird in zwei Schichten beschrieben, einer logischen Sicht, bestehend aus Workflows und einer Ausführungsschicht, bestehend aus Services.

Die Workflow-Beschreibung ist an das in [Tex08] genutzte Workflow-Modell angelehnt. Jeder Workflow besteht aus einer Menge von Aktivitäten, die in einem gerichteten azyklischen Graph angeordnet sind. Ein Workflow-Graph wird aus den im folgenden beschriebenen Elementen gebildet. Die zusammengesetzten Elemente können dabei beliebig weiter geschachtelt werden.

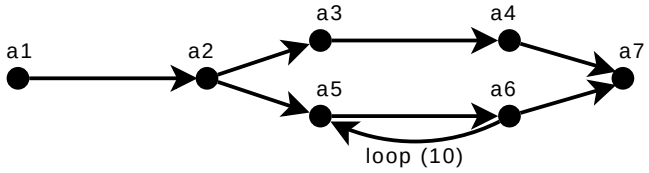


Abbildung 1: Ein Beispielworkflow mit mehreren Sequenzen, einer Parallelausführung und einer Schleife.

Die grundlegenden Elemente sind *atomare Aktivitäten*. Jede atomare Aktivität wird von genau einem Service ausgeführt. *Zusammengesetzte Aktivitäten* dienen der logischen Strukturierung eines Workflows. Möglich sind dabei *Sequenzen* und *Parallelausführungen*. Zudem sind *Schleifen* als abkürzende Schreibweise möglich, wenn eine atomare oder zusammengesetzte Aktivität endlich oft wiederholt ausgeführt wird. Dazu gibt man mit einem Schleifengewicht n an, wie oft eine Schleife maximal durchlaufen wird. Abbildung 1 zeigt einen Beispielworkflow, der alle möglichen Elemente nutzt.

Für ein effektives SLM muss weiterhin definiert werden, welche Eigenschaften des Systems im Rahmen des Managements betrachtet werden sollen. Relevant ist dabei der Inhalt von SLAs und SLOs. Zunächst wird das SLM auf die Betrachtung der Antwortzeit eingeschränkt, um ein einfaches Problem zu erhalten. Ein SLA-Zielwert beschreibt die maximale Gesamtantwortzeit eines Workflows, ein SLO-Zielwert die maximale Antwortzeit einer Aktivität.

Um von der maximalen Antwortzeit für den gesamten Workflow zu einem Zielwert für einzelne Aktivitäten zu kommen, wird der SLA-Zielwert auf die einzelnen Aktivitäten eines Workflows aufgeteilt, wodurch die Zielwerte für die einzelnen SLOs entstehen (vgl. [SK08], [Mik08]).

Das Ziel des Managements ist, die Antwortzeit für alle Aktivitäten immer geringer als das jeweils zugewiesene SLO zu halten. Ist dies nicht der Fall, so tritt eine SLO-Verletzung auf. Als Maßnahme müssen geeignete Änderungen am System durchgeführt werden, so dass diese Bedingung zu einem späteren Zeitpunkt wieder erfüllt wird.

Geeignete Managementaktionen müssen eines der folgenden beiden Resultate bewirken:

- Lockerung des SLOs
- Verkürzung der Antwortzeit

Die Lockerung des SLOs erhöht dessen Zielwert für die Antwortzeit, so dass das SLO wieder eingehalten wird, ohne dass die Antwortzeit verbessert wird. Da die Summe aller SLO-Zielwerte in einem Pfad durch den Workflow nicht größer als der aus dem SLA stammende Zielwert sein darf, muss außerdem der SLO-Zielwert für mindestens eine andere Aktivität gestrafft werden (vgl. [Sch07]). Der Betrag um den ein SLO-Zielwert gelockert, bzw. gestrafft wird, wird im weiteren Verlauf *SLO-Anteil* genannt.

Die zweite Möglichkeit sind Managementaktionen die zum Ziel haben, die Antwortzeit einer bestimmten Aktivität zu verkürzen. Dabei wird das SLO nicht verändert. Die generelle Vorgehensweise ist dabei, dass dem Service zur Ausführung der Aktivität mehr Ressourcen bereitgestellt werden. Da der Zusammenhang zwischen tatsächlicher Antwortzeit und Ressourcenzuordnung sehr komplex ist (vgl. [Mar08]), abstrahiert dieser Beitrag davon, indem davon ausge-

gangen wird, dass man Antwortzeitanteile zwischen Aktivitäten verschieben kann, sofern diese vom gleichen Service ausgeführt werden. D.h. Antwortzeitanteile werden als Ersatz für jegliche andere Ressource betrachtet.

Eine zusätzliche Möglichkeit, die im weiteren Verlauf allerdings nicht betrachtet wird, ist die gezielte Kombination der beiden Managementaktionen. Die Lockerung von SLOs kann nur innerhalb eines Workflows erfolgen, die Verkürzung der Antwortzeit nur zwischen zwei Aktivitäten, die durch den gleichen Service ausgeführt werden. Eine Kombination dieser Möglichkeiten kann nun genutzt werden, um zwischen Aktivitäten unterschiedlicher Workflows SLO- bzw. Antwortzeitanteile zu verschieben, welche nicht durch den gleichen Service ausgeführt werden. Dazu kann eine Aktivität als Händler auftreten, welche beispielsweise das eigene SLO lockert, um anschließend Antwortzeit-Anteile an eine Aktivität zu vergeben, welche durch den gleichen Service ausgeführt wird.

Führt keine mögliche Aktion zur Behebung einer SLO-Verletzung, muss diese an eine höhere Managementebene eskaliert werden.

3 Mechanismus-Design

Nach der Beschreibung des Systems werden nun die Grundlagen für den Management-Mechanismus beschrieben, welcher auf der Basis des Mechanismus-Designs entworfen wurde.

Das dem Mechanismus-Design zugrunde liegende Modell wird unter anderem in [NR01, NRTV07, Ste08] detailliert beschrieben.

In der Spieltheorie sind folgende Schreibweisen üblich (vgl. [NRTV07, Ste08]):

- Ein Vektor $v_{-i} = (v_1, \cdots, v_{i-1}, v_{i+1}, \cdots, v_n)$ enthält jedes Element des ursprünglichen Vektors v, ausgenommen des Eintrags i.
- Ein Vektor v kann auch als $v = (v_i, v_{-i})$ geschrieben werden.

Definition 1 (Mechanismus-Design Problem) Ein Mechanismus-Design Problem wird durch ein gewünschtes Ergebnis beschrieben, das von einer Anzahl rationaler Agenten erreicht werden soll. Dabei gelten folgende Annahmen:

- Ein *Mechanismus* ist ein Tupel $M =_{def} (A, \mathfrak{S}, O, f, p_1, \cdots, p_n)$, mit einer endlichen Menge *Agenten* $A \subset \mathbb{N}$, einer Menge möglicher *Strategien* $\mathfrak{S}$, einer Menge möglicher *Ergebnisse* O, der *Ergebnisfunktion* $f : \mathfrak{S}^n \rightarrow O$ und *Zahlungsfunktionen* $p_i : \mathfrak{S}^n \rightarrow \mathbb{R}$ für alle Agenten $i \in A$.
- Jeder Agent i besitzt private Informationen, die im sogenannten *Typ* $t_i \in T_i$ wiedergespiegelt werden. Die Menge T_i ist dabei für den Agenten i eine Teilmenge aller möglichen Typen $\mathfrak{T}$.
- Jeder Agent i hat mehrere mögliche Strategien $S_i \subseteq \mathfrak{S}$. Eine Strategie j ist definiert als eine Funktion $s_i^j : T \times \mathfrak{S}^{n-1} \rightarrow S_i$. Anschaulich ist die Strategie j des Agenten i vom eigenen Typ und von den Strategien aller anderen Agenten abhängig.

- Für jeden Strategievektor $s = (s_1, \cdots, s_n)$ existiert ein Ergebnis $o \in O$, das durch die Ergebnisfunktion f berechnet wird.
- Die Ergebnisfunktion wird auch *soziale Entscheidungsfunktion* genannt. Eine soziale Entscheidungsfunktion heißt *effizient*, wenn sie ein bestimmtes Kriterium maximiert.
- Für jeden Agenten hat ein Ergebnis einen bestimmten Wert, der durch eine *Wertfunktion* $v_i : T_i \times O \rightarrow \mathbb{R}$ beschrieben wird. Zusammen mit einer vom Mechanismus zugeteilten Bezahlung p_i ergibt sich der tatsächliche Nutzen für einen Agenten i durch die *lineare Nutzenfunktion* u_i:

  $$u_i(o, p_i, t_i) =_{def} v_i(t_i, o) - p_i$$

  Ein rationaler Agent versucht, diese Nutzenfunktion zu maximieren (vgl. [NRTV07]).

Ein Beispiel für einen Mechanismus ist die verallgemeinerte Vickrey-Auktion (vgl. [AC04]), welche zu einer wichtigen Klasse von Mechanismen, den sogenannten Vickrey-Clarke-Groves-Mechanismen (VCG) gehört (vgl. [NRTV07]).

Definition 2 Die verallgemeinerte Vickrey-Auktion (GVA) ist eine Auktionsform und damit ein Mechanismus M, durch die eine Menge von diskreten Objekten $\mathfrak{O}$ versteigert wird. Jeder Bieter i gibt für jedes Objekt $\mathfrak{o} \in \mathfrak{O}$ ein Gebot b_i ab. Weiterhin legt der Verkäufer der Objekte einen Reservierungspreis p_r fest, den jeder Bieter für jedes Objekt mindestens bieten muss.
Die Ermittlung des Ergebnisses erfolgt so, dass der soziale Nutzen maximiert wird:

$$f(v_1, \cdots, v_n) \in max_{o \in O} \sum_{i=1}^{n} v_i(o)$$

Die Ermittlung von Preisen wird mit der sogenannten Clarke-Pivot-Regel durchgeführt (vgl. [NRTV07]). Der Reservierungspreis kann dabei als Gebot für alle Objekte betrachtet werden. Der zu zahlende Preis p_i lässt sich wie folgt berechnen:

$$p_i = \max_{c \in O} \sum_{j \neq i} v_j(c, t_j) - \sum_{j \neq i} v_j(o, t_j)$$

Da die GVA ein VCG-Mechanismus mit Clarke-Pivot-Regel ist, besitzt dieser Mechanismus verschiedene nützliche Eigenschaften, darunter *individuelle Rationalität*, *Anreizkompatibilität* und *Effizienz* (vgl. [AC04, NRTV07]):

4 Das SLO-Spiel

Bisher wurde beschrieben, aus welchen Bestandteilen ein System besteht, welche Ziele der SLM-Prozess erreichen soll und welche Managementaktionen möglich sind, um diese Ziele zu erreichen. Weiterhin wurde das abstrakte Konzept eines Mechanismus definiert und mit dem Beispiel der GVA veranschaulicht.

Das SLO-Spiel kombiniert nun alle beschriebenen Teile in einem Management-Mechanismus, der mittels Auktionen SLO- oder Antwortzeitanteile zwischen verschiedenen Aktivitäten umverteilt.

Im weiteren Verlauf werden folgende Varianten des SLO-Spiels untersucht.

- Das *einfache SLO-Spiel* lockert SLOs bzw. strafft diese innerhalb eines Workflows.
- Das *verallgemeinerte SLO-Spiel* implementiert außerdem den Transfer von Antwortzeitanteilen innerhalb eines Services zwischen verschiedenen Aktivitäten.

Am SLO-Spiel nehmen alle Aktivität des Systems separat teil, d.h. die Menge der Agenten entspricht der Menge aller Aktivitäten im System. Es wird also für jede Aktivität unabhängig eine Entscheidung getroffen, welche Managementaktion durchgeführt werden soll.

Da der Mechanismus mittels Auktionen funktionieren soll, wird ein Agent zu einem Bieter, sofern er SLO- oder Antwortzeitanteile kaufen muss, d.h. wenn die Antwortzeit den SLO-Zielwert übersteigt. Ansonsten wird der Agent zu einem Verkäufer. Jeder Verkäufer führt anschließend auf Basis der GVA eine Auktion durch. Parallel dazu sucht jeder Käufer nach aktiven Auktionen, wählt eine davon aus, berechnet sein mögliches Gebot und gibt das berechnete Gebot in der ausgewählten Auktion ab. Anschließend berechnet jeder Verkäufer das Ergebnis der eigenen Auktion, d.h. eine Zuordnung von SLO- und Antwortzeitanteilen an die Agenten, welche ein Gebot abgegeben haben. Dieses Ergebnis wird den einzelnen Bietern anschließend mitgeteilt.

Eine wichtige Frage bleibt, wie findet ein Käufer passende Auktionen? Aus der Analyse der möglichen Managementaktionen stellen sich folgende Möglichkeiten dar (vgl. [Jun08]):

- SLO-Anteile können innerhalb von Sequenzen umverteilt werden.
- Antwortzeitanteile können nur innerhalb eines Services umverteilt werden.

Die Umverteilung von Antwortzeitanteilen geschieht lokal innerhalb eines Services, deshalb kann auf alle Informationen immer zugegriffen werden. Somit ist das Auffinden von Auktionen innerhalb dieses Services trivial über eine globale Datenstruktur innerhalb des Services implementierbar.

Für die Umverteilung von SLO-Anteilen wird allerdings eine kompliziertere Kommunikationsstruktur benötigt. Es wird dazu ein Gruppenkommunikationssystem genutzt (vgl. [CKV01]). Jeweils eine Gruppe wird pro zusammengesetzter Aktivität gebildet, diese Gruppe bildet sich selbständig aus allen enthaltenen Aktivitäten über einen gemeinsamen Gruppennamen, der aus der ursprünglichen Workflowbeschreibung abgeleitet wird. In jeder dieser Gruppen wird anschließend ein Vertreter ausgewählt, der diese Gruppe in der übergeordneten Gruppe vertritt.

Um über dieses Gruppenkommunikationssystem Auktionen zu finden, sendet jeder Käufer in seiner Gruppe eine Multicast-Nachricht an alle Gruppenmitglieder. Alle Verkäufer in dieser Gruppe werden auf diese Anfrage antworten.

Den Vertretern innerhalb der zusammengesetzten Aktivität kommt eine Sonderrolle zu, sie sammeln alle Performance-Daten der in der zusammengesetzten Aktivität enthaltenen Aktivitäten. Falls der Vertreter an einer übergeordneten Sequenz beteiligt ist, kann er auf Basis der gesammelten Performance-Daten SLO-Anteile für die zusammengesetzte Aktivität kaufen oder verkaufen. Die jeweiligen Anteile werden anschließend geeignet unter den beteiligten Aktivitäten aufgeteilt (vgl. [Jun08]).

5 Bewertung

Um die Qualität des SLO-Spiels aus theoretischer Sicht beurteilen zu können, muss zunächst ein Effizienzkriterium definiert werden, welches das SLO-Spiel erreichen soll. Grundsätzlich lassen sich zwei Effizenzkriterien unterscheiden.

Für das erste Effizienzkriterium wird jedem Workflow ein Wert zugeordnet, der sich aus dem Nutzen des Workflows für den Service-Anbieter ableitet. Für einen bestimmten Zeitpunkt ist der Gesamtwert des Systems die Summe der Werte aller Workflows, deren SLA seit dem letzten Zeitpunkt der Wertermittlung eingehalten wurde. Effizient wäre das Ergebnis des SLO-Spiels, wenn es den Gesamtwert des Systems maximiert. Da der vorliegende Mechanismus dieses Effizienzkriterium nicht erreichen kann (vgl. [Jun08]), wird im folgenden ein zweites, schwächeres Kriterium vorgestellt:

Definition 3 Ein Ergebnis $o \in O$ ist dann in Bezug auf einen Workflow w effizient, wenn alle SLOs aller an w beteiligten Aktivitäten a eingehalten werden. Dies ist dann möglich, wenn alle Pfade durch w eine Gesamtausführungszeit kleiner dem Zielwert des SLAs besitzen.

Bei der Betrachtung des einfachen und des verallgemeinerten SLO-Spiels, stellt man die beiden folgenden Eigenschaften fest. Die vollständigen Beweise finden sich in [Jun08].

Theorem 1 *Das* einfache SLO-Spiel *erreicht nach einer endlichen Anzahl von Auktionen immer ein nach Definition 3 effizientes Ergebnis, sofern alle Pfade p durch den Workflow w eine Gesamtausführungszeit kleiner als die vorgegebene Ausführungszeit des Workflow-SLAs haben.*

Theorem 2 *Das* verallgemeinerte SLO-Spiel *erreicht* nicht *garantiert die Effizienz des einfachen SLO-Spiels.*

Die Beweisidee zu Theorem 1 basiert auf einer geschickten Wahl der Wertfunktion für Bieter, so dass es für die Bieter immer möglich ist, in endlicher Zeit die benötigten Anteile zu kaufen, sofern irgendeine Aktivität in der gleichen Sequenz genügend Anteile zur Verfügung hat.

Der Beweis von Theorem 2 setzt voraus, dass zwei Workflows mindestens einen Service gemeinsam Nutzen. Dadurch kann es passieren, dass ein Workflow dem anderen Workflow benötigte Antwortzeitanteile abkauft. Hinterher können beide Workflows schlechter dastehen, falls die abgekauften Antwortzeitanteile für den kaufenden Workflow nicht ausreichen und der verkaufende Workflow nun ebenfalls nicht mehr genug Antwortzeitanteile besitzt.

Um außerdem eine praktische Bewertung zu ermöglichen, wurde eine Simulation erstellt, welche das SLO-Spiel umsetzt. Die prinzipielle Architektur der Simulation folgt Abbildung 2.

Die Kernkomponente ist der Service-Manager, welchem jeweils genau ein Service und eine Strategie zugeordnet wird. Von außerhalb der Simulation wird dem Service-Manager mitgeteilt welche Aktivitäten eines Workflows der Service ausführt und jeder Aktivität jeweils ein SLO zugeordnet, welches der Service-Manager verwaltet. Der Service-Manager überwacht zur Laufzeit der Simulation den ihm zugeteilten Service und initiiert gegebenfalls Managementaktionen, welche durch die Strategie umgesetzt werden.

Die in der Simulation eingesetzte Strategie implementiert das beschriebene SLO-Spiel und nimmt somit Einfluss auf den Service oder die vom Service-Manager verwalteten SLOs. Die

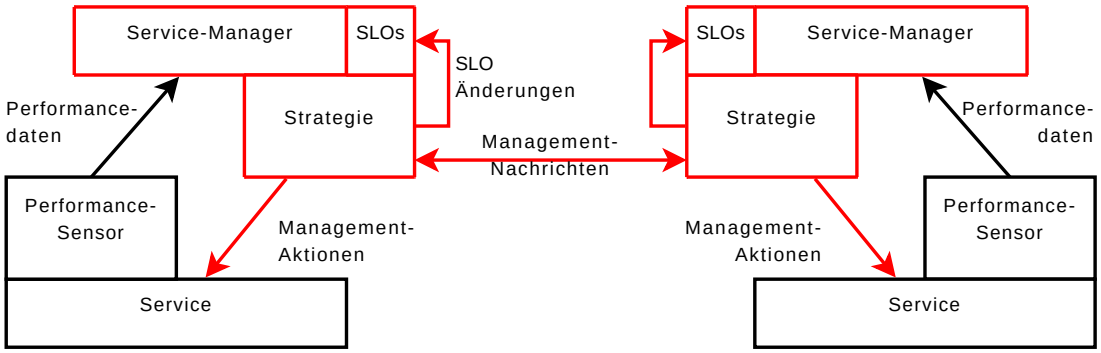


Abbildung 2: Die Architektur der Simulation.

Strategie kommuniziert bei der Umsetzung der Managementaktionen mit den Strategien anderer Service-Manager auf Basis eines Gruppenkommunikationssystems (vgl. [CKV01]), welches gemäß der Beschreibung in Abschnitt 4 implementiert ist.

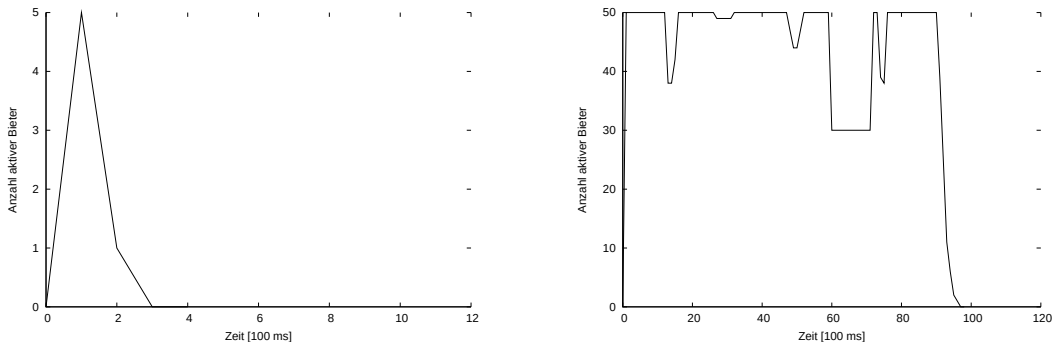
Ein Service innerhalb der Simulation generiert aus einem vorgegebenen Performance-Modell Performance-Daten, im vorliegenden Fall simulierte Antwortzeiten. Die Performance-Daten werden mittels eines Performance-Sensors an den Service-Manager übertragen, welcher diese Daten nutzt, um Managementaktionen über die Strategie anzustoßen.

Es wurden insgesamt drei Szenarien untersucht, die hier der Reihe nach mit den jeweiligen Ergebnissen vorgestellt werden. Das erste Szenario stellt sich folgendermaßen dar:

- Es wird der Einschwingvorgang des Systems mit 10 bzw. 100 Knoten simuliert. Dabei verletzten jeweils 50% der Knoten initial das SLO und 50% nicht. Die Gesamtantwortzeit ist dabei immer geringer als der SLA-Zielwert für den gesamten Workflow.

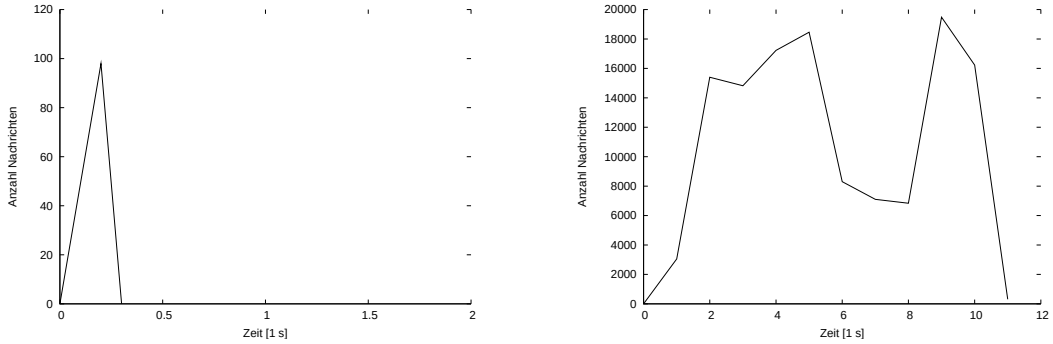
Das Ziel dieses Szenarios ist zu untersuchen, ob das SLO-Spiel praktisch immer eine Lösung findet, d.h. ob das theoretische Effizienzkriterium tatsächlich eingehalten wird.

Gemessen werden kann dies durch die Anzahl der Bieter, da jede einzelne SLO-Verletzung da-



(a) 10 (b) 100

Abbildung 3: Einschwingverhalten aus Sicht der Bieter mit 10 bzw. 100 Agenten.



(a) 10 (b) 100

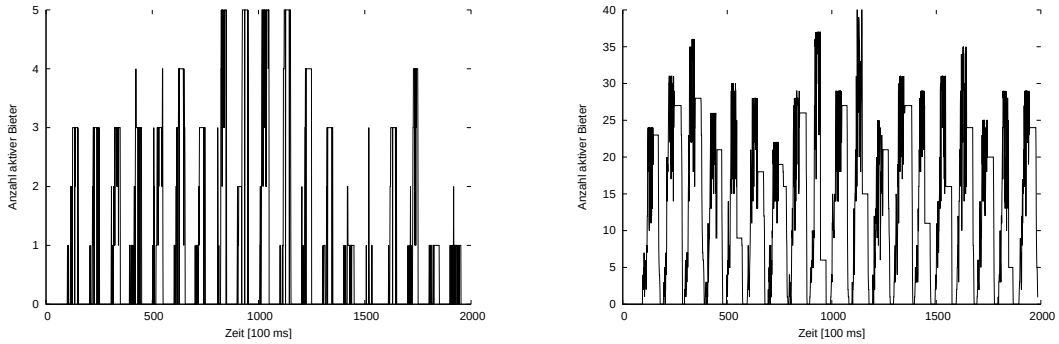
Abbildung 4: Nachrichten-Overhead beim Einschwingverhalten mit 10 bzw. 100 Agenten.

zu führt, dass genau ein Bieter existiert. Der Umkehrschluss gilt auch, d.h. gibt es keine Bieter, ist auch keine SLO-Verletzung vorhanden. Deshalb wird hier die Anzahl der Bieter gegenüber dem Zeitverlauf der Simulation dargestellt. Die Ergebnisse der Simulation in Abbildung 3 zeigen, dass das SLO-Spiel tatsächlich immer eine Lösung findet.

Anhand des Zeitverlaufs lässt sich außerdem eine Grenze der Skalierbarkeit erkennen. In diesem Szenario erkennt man, dass das SLO-Spiel mit der 10-fachen Menge an Agenten deutlich mehr Zeit benötigt. Dies lässt sich mit einer ebenfalls stark zunehmenden Anzahl zu versendender Nachrichten erklärten, denn jeder Käufer muss an alle anderen Agenten Nachrichten verschicken (vgl. Abbildung 4).

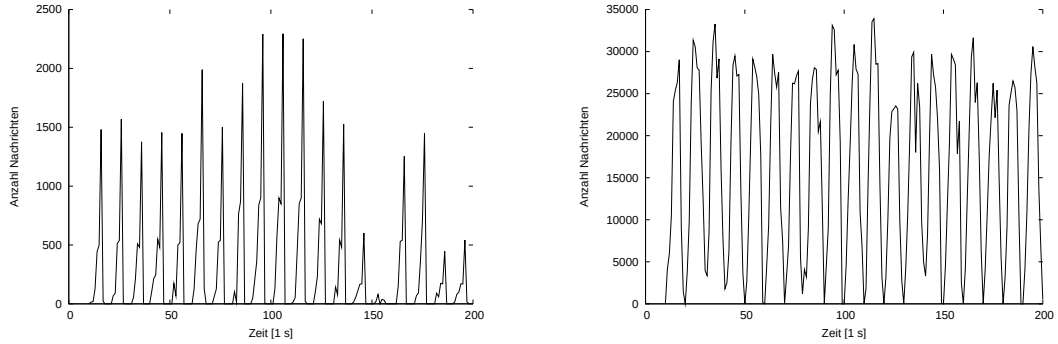
Das zweite Szenario soll überprüfen, wie sich das SLO-Spiel bei hohen Lasten verhält, d.h. wenn es keine garantierte Lösung innerhalb eines Workflows gibt:

- Das Verhalten bei dynamischen Änderungen und hoher Last wird simuliert, d.h. es wird häufig das SLA verletzt. Es wird wieder mit 10 und 100 Knoten gemessen. Die Änderung der Antwortzeiten der Services geschieht periodisch mit leicht zufälliger Verzögerung.



(a) 10 (b) 100

Abbildung 5: Verhalten bei dynamischen Änderungen und 10 bzw. 100 Agenten.



(a) 10 (b) 100

Abbildung 6: Nachrichten-Overhead bei dynamischen Änderungen und 10 bzw. 100 Agenten.

Dieses Szenario soll Aufschluss über das Verhalten des Systems im Grenzfall geben, wenn das System sehr stark ausgelastet ist. Wünschenswert ist in diesem Fall, dass das Managementsystem die Performance auf keinen Fall verschlechtert.

Da keine Lösung für alle SLO-Verletzungen gefunden werden kann, ist die Frage natürlich ob und wie das Managementsystem das eigentliche System beeinflusst. Eine wesentliche Kenngröße dabei ist der erzeugte Nachrichten-Overhead, welcher möglichst gering ausfallen sollte, um das System nicht weiter zu belasten.

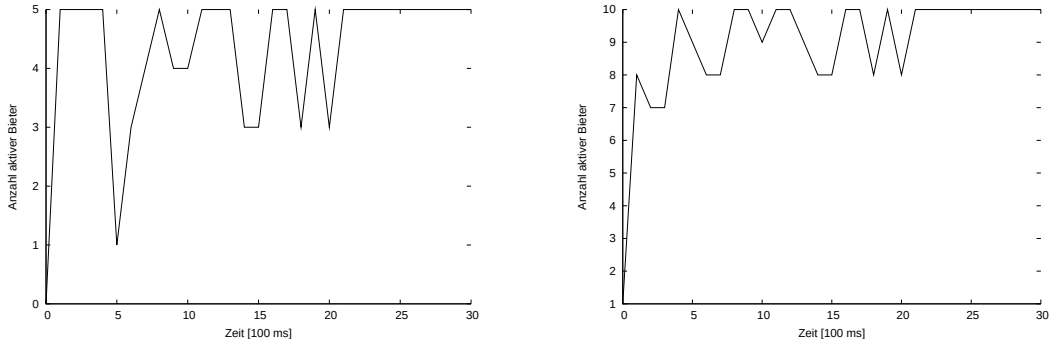
In den Auswertungen ist zu sehen (Abbildung 5), dass sehr häufig keine Lösung gefunden werden kann. Dies zeigt sich deutlich an den Plateaus in Abbildung 5(b), jeweils nach einer kurzen Verbesserungsphase zuvor.

Der Nachrichten-Overhead im Falle der dynamischen Änderungen bei sehr hoher Last explodiert allerdings regelrecht (Abbildung 6). Die bei 100 Agenten erzeugte sehr hohe Spitzenlast würde in der Realität die Dienstgüte vermutlich weiter deutlich verschlechtern. Erklären lässt sich dieses Phänomen durch die Art und Weise, wie die Käufer nach Auktionen suchen. Jeder Bieter versendet dazu an alle anderen Agenten eine Nachricht. Der Nachrichtenaufwand wächst also quadratisch mit der Anzahl der Agenten. Dieses Verfahren lässt sich vermutlich abändern, so dass der Nachrichtenaufwand wesentlich geringer ausfällt. Die prinzipielle Idee ist dabei, dass sich der Nachrichtenfluss auch umkehren lässt, d.h. statt der Suchanfrage der Bieter, senden die Verkäufer ein Broadcast an alle Agenten. Da im schlechtesten Fall keine Agenten mehr Verkäufer sind, tendiert der zusätzliche Netzwerk-Overhead gegen 0 (vgl. [Jun08]).

Als drittes Szenario soll das zweite Ergebnis der Effizienzbetrachtung praktisch untersucht werden, d.h. es sind zwei Workflows vorhanden, deren Aktivitäten sich Services teilen:

- Es existiert ein Workflow mit sehr hoher Priorität, sowie einer mit sehr niedriger Priorität. Beide Workflows teilen sich mehrere Services. Es gibt es nur ein effizientes Ergebnis, wenn der Workflow mit niedrigerer Priorität alle SLOs und damit sein SLA einhält.

Wie man in Abbildung 7 erkennen kann, verletzen beide Workflows das SLA. Das theoretische Ergebnis wird also bestätigt. Es zeigt sich, dass eine rein lokale Optimierung des eigenen Nutzen nicht ausreicht, um in jedem Fall eine Verbesserung zu erzielen.



(a) Workflow 1 (b) Workflow 2

Abbildung 7: Verhalten der Bieter zweier Workflows mit gemeinsam genutzten Services im schlechtesten Fall.

Diese Problematik kann eventuell durch zusätzliche Kommunikation zwischen den Agenten verringert werden. Ein effektives Kommunikationsschema muss noch entwickelt werden.

6 Zusammenfassung und Ausblick

Im vorliegeden Beitrag wurde ausgehend von der Motivation für selbstorganisierende Systeme das sogenannte SLO-Spiel, ein einfacher Mechanismus auf der Basis von Mechanismus-Design, entwickelt. Dieser ist prinzipiell dazu geeignet einige Arten von SLO-Verletzungen zu beheben.

Als Grundlage für die Entwicklung wurde ein Service-orientiertes System, ein Ziel für das Management und die Grundlagen des Mechanismus-Designs beschrieben. Auf dieser Basis wurde das SLO-Spiel als spieltheoretischer Mechanismus entwickelt, welcher mit Hilfe von verteilten Auktionen Anteile an logischen Ressourcen umverteilen kann. Das SLO-Spiel nutzt dafür eine verteilte verallgemeinerte Vickrey-Auktion, da diese einige gewünschte spieltheoretische Eigenschaften aufweist.

Für das entwickelte SLO-Spiel wurde anschließend gezeigt, dass es ein effizientes Ergebnis erreichen kann, sofern ein Ergebnis existiert. Allerdings erreicht das SLO-Spiel in verallgemeinerter Form dieses Ergebnis nicht, sondern verschlechtert die Situation sogar.

Die durchgeführte Simulation bestätigt die theoretischen Ergebnisse und zeigt auch den Weg für weitere Verbesserungen auf. In stark ausgelasteten Systemen erzeugt das bisherige Spiel einen sehr hohen Netzwerk-Overhead auf Grund der genutzten Kommunikationsstruktur zur Selbstorganisation. In weiteren Arbeiten können diese Schwächen sowohl theoretisch als auch praktisch durch Änderungen an der bestehenden Simulation weiter untersucht werden.

Literatur

[AC04] L. M. Ausubel, P. Cramton. Vickrey Auctions with Reserve Pricing. Papers of peter cramton 99wpvic, University of Maryland, Department of Economics - Peter Cram-

ton, 2004.

[CKV01] G. V. Chockler, I. Keidar, R. Vitenberg. Group communication specifications: a comprehensive study. *ACM Comput. Surv.* 33(4):427–469, Dezember 2001.

[Deb04] M. Debusmann. *Modellbasiertes Service Level Management verteilter Anwendungssysteme*. PhD thesis, Universität Kassel, Dezember 2004.

[Jun08] B. Jungk. Bewertung von Selbstorganisationsmechanismen für Managementkomponenten auf Basis von Simulationen und Leistungsmessungen. Master's thesis, FH Wiesbaden, FB Design Informatik Medien, September 2008.

[Mar08] D. Marinescu. Design and Evaluation of Self-Management Approaches for Virtual Machine-Based Environments. Master's thesis, FH Wiesbaden, FB Design Informatik Medien, Februar 2008.

[Mik08] A. Mikula. Automatisierte Überwachung von Dienstgütekriterien in SOA Workflows. Diplomarbeit, FH Wiesbaden, FB Design Informatik Medien, August 2008.

[MWJ+07] G. Mühl, M. Werner, M. A. Jaeger, K. Herrmann, H. Parzyjegla. On the Definitions of Self-Managing and Self-Organizing Systems. In *KiVS 2007 Workshop: Selbstorganisierende, Adaptive, Kontextsensitive verteilte Systeme*. Springer, 2007.

[NR01] N. Nisan, A. Ronen. Algorithmic Mechanism Design. *Games and Economic Behavior* 35(1-2):166–196, April 2001.

[NRTV07] N. Nisan, T. Roughgarden, E. Tardos, V. V. Vazirani. *Algorithmic Game Theory*. Cambridge University Press, New York, NY, USA, 2007.

[Sch07] M. Schmid. Ein Ansatz für das Service Level Management in dynamischen Architekturen. In *KiVS 2007 - Kommunikation in Verteilten Systemen - Industriebeiträge, Kurzbeiträge und Workshops*. Pp. 255–266. VDE Verlag, March 2007.

[SK05] M. Schmid, R. Kröger. Selbstmanagement-Ansätze im eBusiness-Umfeld. *PIK - Praxis der Informationsverarbeitung und Kommunikation* 28 / 4:211–216, 2005.

[SK08] M. Schmid, R. Kröger. Decentralised QoS-Management in Service Oriented Architectures. In Meier and Terzis (eds.), *DAIS*. Lecture Notes in Computer Science 5053, pp. 44–57. 2008.

[Ste08] J. Steimle. *Algorithmic Mechanism Design*. Springer-Verlag, 2008.

[Tex08] A. Textor. Monitoring unternehmenskritischer Anwendungen unter Verwendung modellbasierter Performance Constraints. Bachelor's thesis, FH Wiesbaden, FB Design Informatik Medien, September 2008.

[Vas07] Y. Vasiliev. *SOA and WS-BPEL*. Packt Publishing, August 2007.

[YBT05] J. Yu, R. Buyya, C.-K. Tham. QoS-based Scheduling of Workflow Applications on Service Grids. Technical report, Melbourne, Australia, 2005.