2013

Vorhersagemodell für die Verfügbarkeit von IT-Services aus Anwendungssystemlandschaften

Sascha Bosse
Otto-von-Guericke-Universität, Magdeburger Research and Competence Cluster for Very Large Business Applications, Magdeburg, Germany, sascha.bosse@ovgu.de

Matthias Splieth
Otto-von-Guericke-Universität, Magdeburger Research and Competence Cluster for Very Large Business Applications, Magdeburg, Germany, matthias.splieth@ovgu.de

Klaus Turowski
Otto-von-Guericke-Universität, Magdeburger Research and Competence Cluster for Very Large Business Applications, Magdeburg, Germany, klaus.turowski@ovgu.de

Vorhersagemodell für die Verfügbarkeit von IT-Services aus Anwendungssystemlandschaften

Sascha Bosse, Matthias Splieth, und Klaus Turowski

Otto-von-Guericke-Universität, Magdeburger Research and Competence Cluster for Very Large Business Applications, Magdeburg, Germany
{sascha.bosse,matthias.splieth,klaus.turowski}@ovgu.de

Abstract. Serviceorientierung und Cloud Computing haben dazu geführt, dass sich inzwischen immer mehr Unternehmen darauf spezialisieren, IT-Dienstleistungen für Organisationen anzubieten. Erhöhter Wettbewerb in dieser Branche führt zu Kosten- und Qualitätsdruck, was eine bessere Beherrschung des damit einhergehenden Leistungserstellungsprozesses durch quantifizierbare Vorhersagen für Anpassungen an den Anwendungssystemlandschaften der Service-Anbieter wünschenswert macht. In diesem Beitrag wird ein neuer Ansatz entwickelt, um die Verfügbarkeit von IT-Services aus Anwendungssystemlandschaften vorherzusagen. Entgegen bisher bekannter Lösungen geht dieser Ansatz nicht davon aus, dass die einzelnen Systemkomponenten unabhängig voneinander ausfallen. Es wird gezeigt, dass der Lösungsansatz ohne diese Abhängigkeiten die Ergebnisse analytischer Methoden erreicht. Somit können Entscheidungsprozesse unterstützt werden.

Keywords: Anwendungssystemlandschaft, Verfügbarkeit, IT-Service

1 Einleitung

Die effektive und effiziente Nutzung von Informationstechnologie (IT) spielt eine immer entscheidendere Rolle für den Erfolg von Unternehmen. Neben der klassischen Strategie, Geschäftsprozesse durch eine betriebseigene IT-Abteilung zu unterstützen, ist auch das Outsourcing der IT zu einem externen Dienstleister in den Fokus gerückt [1]. Unternehmen, die sich auf das Anbieten solcher IT-Dienstleistungen bzw. IT-Services spezialisiert haben, können unter dem Begriff IT-Service-Industrie zusammengefasst werden.

Das IT-Outsourcing hat Vor- und Nachteile [1]: Wichtigste Motivation ist in der Regel die Vermeidung von Fixkosten und das Weiterreichen von Risiken. Größter Nachteil des IT-Outsourcing ist der Kontrollverlust, sei es bspw. bezüglich sensibler Daten, Ausfallsicherheit oder Performanz. Um diese Nachteile zu egalisieren, werden so genannte Service Level Agreements (SLAs) als Verträge zwischen Service-Anbieter und Kunde geschlossen [2]. Sie enthalten unter anderem Garantien für die Servicequalität (engl. Quality of Service – QoS).

11th International Conference on Wirtschaftsinformatik,
27th February – 01st March 2013, Leipzig, Germany

Um konkurrenzfähig zu sein und zu bleiben, muss ein Dienstanbieter zwischen den Kosten der Service-Bereitstellung und deren Qualität abwägen. Nicht zuletzt aufgrund der steigenden Energiepreise und der Nachhaltigkeitsdiskussion werden verschiedene Ansätze erforscht, die Diensterbringung durch Anwendungssystemlandschaften effizienter zu gestalten. Unter anderem sind dies Virtualisierungstechniken, intelligente Lastverteilung oder verbesserte Netzwerkstrukturen [3-5]. Für einen Service-Anbieter stellt sich dabei die Frage, ob diese Ansätze zu geringeren Kosten führen und die QoS gehalten oder sogar verbessert werden kann. Vorhersagemodelle könnten diese Entscheidungen unterstützen.

Während die Qualität von IT-Systemen traditionell eng an die Performanz der angebotenen Funktionen geknüpft ist, geraten nun auch andere nicht-funktionale Eigenschaften in den Fokus: Neben Wartbarkeit und Skalierbarkeit betrifft dies vor allem die Verfügbarkeit von Systemen [6]. Nach einer Untersuchung der *Aberdeen Group* von 134 Unternehmen mit Rechenzentren im Jahre 2012 erleiden diese durchschnittlich über eine Million US-Dollar an Umsatzeinbußen aufgrund von IT-Ausfällen [7]. Meldungen über Rechenzentrumsausfälle sind selbst bei spezialisierten Dienstleistern wie *Amazon.com* immer wieder aktuell [8-9].

1.1 Ziele und Methodologie

Ziel dieser Arbeit ist, ein Vorhersagemodell für die Verfügbarkeit von Services aus Anwendungssystemlandschaften auf Basis einer Diskreten-Ereignis-Simulation zu entwickeln. Dieses Modell würde es Betreibern ermöglichen, die Servicequalität bezüglich der Verfügbarkeit bei Änderungen oder dem Neuaufbau von Systemlandschaften zu bestimmen und damit die Entscheidungsfindung zu unterstützen. Diese Anwendungssystemlandschaften sind äußerst komplexe Gebilde, die sich durch eine hohe Dynamik auszeichnen [10]. Generelle Aussagen über Änderungseffekte sind in der Regel schwierig bis unmöglich. Dies führt unter anderem dazu, dass geplante Kosteneinsparungen geringer als erwartet ausfallen oder sogar nicht eintreten [11].

Die Entwicklung des Vorhersagemodells folgt einem gestaltungsorientierten Forschungsdesign, vgl. z. B. [12]. Das entstehende Artefakt besteht dabei aus einem Modell und Algorithmen zur Vorhersage. Diese erfolgt quantitativ durch die Simulation. Durch die hohe Komplexität der untersuchten Anwendungssystemlandschaften und die spätere Erweiterung auf andere Eigenschaften ist diese Methode geeignet, bei formal beschreibbaren, aber schwierig analytisch zu lösenden Problemen die nötige Flexibilität zu bieten [13]. Evaluiert wird das Artefakt durch einen Vergleich mit einer analytischen Methode an einem ausgewählten Anwendungsbeispiel.

1.2 Struktur des Beitrags

Nachdem im ersten Abschnitt auf Motivation und Zielsetzung des Forschungsbeitrags eingegangen wurde, wird in Abschnitt 2 das Forschungsgebiet abgegrenzt. Dazu werden Definitionen des Begriffs Anwendungssystemlandschaft und der Größe Verfügbarkeit erörtert.

Anschließend wird in Abschnitt 3 der Stand der Technik bei der Vorhersage von Verfügbarkeit von Systemen beleuchtet, bevor in Abschnitt 4 ein eigener Lösungsansatz präsentiert wird.

Dieser Lösungsansatz wird in Abschnitt 5 auf das Beispielszenario eines international agierenden SAP *Application Service Provider* (ASP) angewendet. Auf Basis dieses Szenarios erfolgt dann eine Verifikation des vorgestellten Ansatzes. In Abschnitt 6 werden die Ergebnisse des Beitrags zusammengefasst und ein Ausblick auf weitere Forschungsarbeiten gegeben.

2 Definitionen

2.1 Anwendungssystemlandschaft

Eine Anwendungssystemlandschaft (auch IT-Systemlandschaft oder IT-Landschaft) ist ein System von Systemen, die miteinander integriert sind [14-15]. Sie ist in der Regel historisch gewachsen, komplex und besteht aus heterogenen Einzelteilen [10], [16]. Die Systeme der Landschaft werden von Menschen mit unterschiedlichen Zielen benutzt [15]. Neben Anwendungssystemen enthält sie alle benötigten Unterstützungssysteme [10].

Der Begriff *Enterprise Architecture* wird als Obermenge der Anwendungssystemlandschaft gesehen [14]. Diese bietet eine ganzheitliche Sicht auf ein Unternehmen, meistens Ebenen-orientiert. Diese reichen von der Ebene der IT-Infrastruktur über die betrieblichen Anwendungen, deren Integration untereinander bis zur Ebene der Geschäfts- bzw. Organisationssicht und der strategischen Ebene [15]. Auf jeder Ebene interagieren Menschen [17].

Eine Anwendungssystemlandschaft ist dann eine Teilmenge bzw. die Umsetzung einiger Teile dieser Enterprise Architecture. Neben den Anwendungen und deren Integration beschreibt die Anwendungssystemlandschaft alle nötigen Infrastruktursysteme. Auch enthalten ist die Nutzung der Systeme durch Menschen, um bestimmte Geschäftsprozesse zu unterstützen.

2.2 Nicht-funktionale Eigenschaften – Verfügbarkeit

Funktionale Eigenschaften eines Systems adressieren die Funktionen oder das Verhalten, die das System anbieten soll. Nicht-funktionale Eigenschaften beschreiben dabei alle qualitativen Aspekte eines Systems mit Ausnahme der funktionalen [18]. Leitfaden zur Aufstellung von qualitativen Eigenschaften wie z. B. die ISO 9126 (in der ISO 25000 2005 aufgegangen) basieren dabei auf dem so genannten *Factor Criteria Metrics (FCM) Model* [19]. Dabei werden zu Qualitätsfaktoren wie Effizienz oder Übertragbarkeit Qualitätskriterien definiert. Im Fall Übertragbarkeit können diese bspw. Anpassbarkeit oder Ersetzbarkeit bedeuten. Im letzten Schritt muss für jedes Kriterium eine Metrik definiert werden, um die Erfüllung quantitativ messen zu können.

Verfügbarkeit ist nach diesem Modell ein Qualitätskriterium, welches dem Faktor Zuverlässigkeit zugeordnet ist. Gemessen werden kann sie z. B. nach [20] mit der normierten Betriebsdauer, die sich aus dem Verhältnis der tatsächlichen Betriebsdauer zu der geplanten Betriebsdauer ergibt.

Eine probabilistische Definition aus [21] definiert Verfügbarkeit hingegen als die „Wahrscheinlichkeit, dass ein System zu einem bestimmten Zeitpunkt in Betrieb und in der Lage ist, die angeforderten Dienste zu liefern". Hierbei ist der Fokus statt auf die absolute Zeit eher auf die Nutzungsanfrage gesetzt. In [22] werden diese beiden Definitionen verbunden, um auf der einen Seite die Verfügbarkeit aus historischen Daten zu berechnen und sie auf der anderen Seite in Zukunft schätzen zu können. Alle Definitionen gehen dabei davon aus, dass die geplante Betriebszeit entscheidend ist, also geplante Wartungen aus der Berechnung heraus genommen werden. Auch in dieser Arbeit werden beide Definitionen berücksichtigt.

3 Stand der Technik

Vorhersagemodelle für Verfügbarkeit wurden bisher u. a. in der Softwareentwicklung eingesetzt. Bei der komponentenorientierten Softwareentwicklung hängt die Verfügbarkeit des Gesamtsystems von den einzelnen Komponenten ab. In [23] wird eine Literaturanalyse präsentiert, die den Stand der Technik bei der Vorhersage der Verfügbarkeit eines Softwareprodukts in einer frühen Entwurfsphase untersucht. Die gefundenen Ansätze werden in drei Kategorien klassifiziert: Zustandsbasiert- oder additiv-analytisch sowie pfadbasiert-quantitativ Modelle. Letztere erfordern zumindest eine experimentelle Datenerfassung, während die analytischen Ansätze ausnahmslos vom unabhängigen Ausfall der Komponenten ausgehen. Dies wird als signifikanter Schwachpunkt aller bekannten Ansätze identifiziert. Die vorgeschlagene Lösung dieses Problems ist die Erweiterung der Markov-Eigenschaft auf höhere Grade, was jedoch Modellierung und Parametrisierung schnell komplex werden lässt [24].

Neben Software sind auch verteilte Systeme Gegenstand von Vorhersagemodellen für die Verfügbarkeit. In [25] wird z. B. eine Vorhersage für ein OSCAR[1]-Cluster vorgenommen. Hierbei werden drei Ebenen modelliert, die für die Verfügbarkeit entscheidend sind. Neben der Serverebene, auf der Anfragen angenommen werden, gibt es die Netzwerkebene für die Kommunikation und die Clientebene für die angefragte Berechnung. Jede Ebene enthält redundante fehlerbehaftete Komponenten. Bei einem Ausfall der primären Komponenten übernehmen nach kurzer Zeit Standby-Komponenten den Betrieb. Zur Modellierung des Systemverhaltens werden für jede Ebene so genannte *Stochastic Reward Nets* (SRN) verwendet.

Dieses Modell wird in [26] eingeführt. Zwar erlauben kombinatorische Modelle wie Fehlerbäume eine schnelle Auswertung, aber die Abbildung von Abhängigkeiten gestaltet sich schwierig. Zustandsbasierte Modelle wie Markov-Ketten auf der anderen Seite leiden häufig unter der so genannten Zustandsexplosion, welche die Hand-

[1] Open Source Cluster Application Resources

habbarkeit dieser Modelle stark verringert. Eine Möglichkeit, diesen Nachteil zu verringern, bieten Stochastische Petri-Netze, die eine automatische Erzeugung des Zustandsgraphen ermöglichen. Der Zustand des Systems wird dabei über die Kombination der markierten Stellen repräsentiert, während zeitbehaftete Transitionen die Markierung des Netzes verändern.

Eine Erweiterung dieser Netze stellen die so genannten *Generalisierten Stochastischen Petri-Netze* (GSPN) dar. Hierbei wird das SPN um Konstrukte erweitert, die einzelne Transitionen bei bestimmten Systemzuständen deaktivieren können. Weiterhin sind auch zeitlose Transitionen möglich. Verschiedene Farben von Marken ermöglichen die Zusammenfassung strukturell gleicher Netze [27].

Ein SRN ist dann ein GSPN mit *Reward Functions*. Diese werden unterteilt in *Rate Rewards* und *Impulse Rewards*. Erstere verändern Variablen, wenn eine bestimmte Stelle markiert ist, während Letztere Variablen bei Ereignissen verändern. Das Systemverhalten kann dann in einem GSPN modelliert werden, während die Veränderungen der Verfügbarkeit durch Rewards berechnet wird [26].

Die Vorhersage der Verfügbarkeit von IT-Diensten wird unter anderem in [28] adressiert. Dort werden existierende Verfahren in analytische, qualitative und quantitative Methoden eingeteilt. Petri-Netze und ihre Erweiterungen werden als analytische Verfahren klassifiziert, auf die Lösung dieser Netze durch Simulation wird nicht eingegangen. Quantitative Ansätze setzen hier ebenfalls eine Datenerfassung aus laufenden Systemen voraus. Sie sind daher auf hypothetische Systeme nicht übertragbar.

Für den eigenen, quantitativen Lösungsansatz wird ebenfalls angenommen, dass die Verfügbarkeit eines zusammengesetzten Systems aus den einzelnen Komponenten berechnet werden kann. Die zunächst vorgestellten Ansätze aus der Softwareentwicklung gehen dabei ohne Ausnahme davon aus, dass diese Komponenten unabhängig ausfallen [23]. In den Ansätzen für die Verfügbarkeit verteilter Systeme werden unter anderem Stochastic Reward Nets verwendet. Die entwickelten Modelle sind nicht auf beliebige Systeme übertragbar, die Modellierungsmethode jedoch geeignet, auch Abhängigkeiten zwischen den einzelnen Komponenten abzubilden [27]. Die analytische Lösung dieser Netze wird jedoch durch das Problem der Zustandsexplosion erschwert [28]. Aus diesem Grund werden die Netze in diesem Beitrag nicht analytisch, sondern durch eine Diskrete-Ereignis-Simulation mit anschließender Datenanalyse gelöst.

4 Lösungsvorschlag

In einem Experteninterview bei einem international agierenden SAP ASP wurde festgestellt, dass vier Ereignisse zu einer Nichtverfügbarkeit eines Services führen können: Fehler in einem Rechenzentrum bei der Bearbeitung des Dienstes, Fehler in der Infrastruktur um das Rechenzentrum, menschliche Fehler bei der Administration sowie Wartungsarbeiten. Da letztere zu geplanten Downtimes zählen, die bei der Berechnung der Verfügbarkeit ausgeblendet werden [20], müssen nur die ersten drei Ereignisse betrachtet werden. In einer Datensammlung über Serverausfälle von 1996 bis 2005, bereitgestellt vom Los Alamos National Lab, werden analog zu diesem

Interview Hardware-, Software- und Infrastrukturfehler sowie menschliche Fehler unterschieden [29]. Bei der Beschreibung von einzelnen Ressourcen können weiterhin transiente, intermittierende und permanente Fehler unterschieden werden [28]. Jede dieser Fehlerarten zeichnet sich dabei durch unterschiedliche Ausfall- oder Reparaturzeiten aus.

In Abbildung 1 sind die relevanten Schichten der Enterprise Architecture für das Problem, die Verfügbarkeit von Services aus Anwendungssystemlandschaften zu bestimmen, auf Basis des Schichtenmodells aus [17] dargestellt. Die Bereitstellung eines IT-Dienstes erfordert dabei, dass bestimmte Komponenten einer Anwendungssystemlandschaft intakt sind. Die Verfügbarkeit des angebotenen Service hängt dann von diesen Komponenten ab, denen jeweils für jede Fehlerart ein Wert *Mean Time to Failure* (Mittlere Zeit bis zum Fehler - MTTF) und *Mean Time to Recover* (Mittlere Zeit zur Reparatur - MTTR) zugeordnet ist. Die Annahme ist dabei, dass Ausfall- und Reparaturzeiten exponentialverteilt sind. Formal ist eine Komponente c dann ein 6-Tupel.

$$c = (MTTF_t, MTTR_t, MTTF_i, MTTR_i, MTTF_p, MTTR_p) \quad (1)$$

Menschen administrieren diese Komponenten und können dabei Fehler erzeugen. Dies wird hier der Vollständigkeit halber erwähnt, aber im Modell aufgrund schwieriger Abstraktion zunächst nicht aufgenommen.

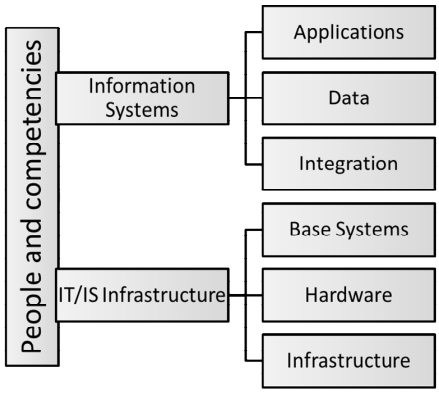

Abb. 1. Relevante Schichten für die Vorhersage der Verfügbarkeit von Services aus Anwendungssystemlandschaften basierend auf [17]

Eine Komponente kann dabei den zwei Schichten Informationssysteme und IT/IS Infrastruktur zugeordnet werden, wobei diese noch weiter unterteilt werden können: Anwendungen (z. B. ein ERP-System) mit Daten- und Integrationskomponenten, Unterstützungs- (z. B. Betriebssysteme), Hardware- (z. B. eine Festplatte) und Infrastruktursysteme (z. B. Stromversorgung).

Dabei wird davon ausgegangen, dass nur Komponentenklassen betrachtet werden, die für die Bereitstellung des Diensts notwendig sind. Eine Komponentenklasse kann jedoch mehrfach realisiert werden, so dass sie dann redundant vorhanden ist. Im

nächsten Schritt können nun die Komponenten und deren Zusammenspiel modelliert werden.

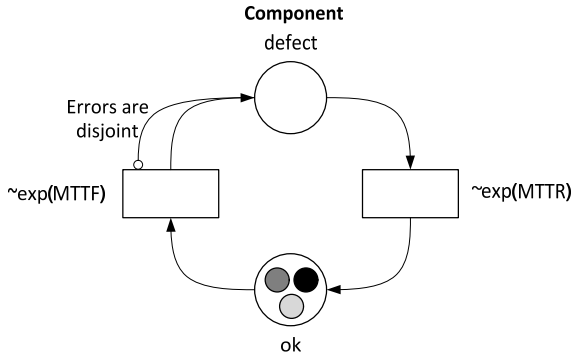

Abb. 2. Farbiges GSPN einer einzelnen Komponente

Einzelne Komponenten werden als farbiges GSPN modelliert (Abbildung 2). Dieses besteht aus zwei Stellen, die den jeweiligen Zustand „defekt“ oder „arbeitet“ widerspiegeln. Dieser Kreislauf enthält drei farbige Marken, die den jeweiligen Fehlerarten zugeordnet sind. Die beiden Transitionen zwischen diesen Stellen schalten nach exponentialverteilter Dauer mit Erwartungswert MTTF beziehungsweise MTTR. Dabei wird angenommen, dass das Auftreten von Fehlerarten disjunkt ist, daher wird die Transition nach „defekt“ durch eine Inhibitor-Kante für die anderen Fehlerarten deaktiviert. Der Vereinfachung halber wird in den folgenden Abbildungen auf die Berücksichtigung von Farben verzichtet.

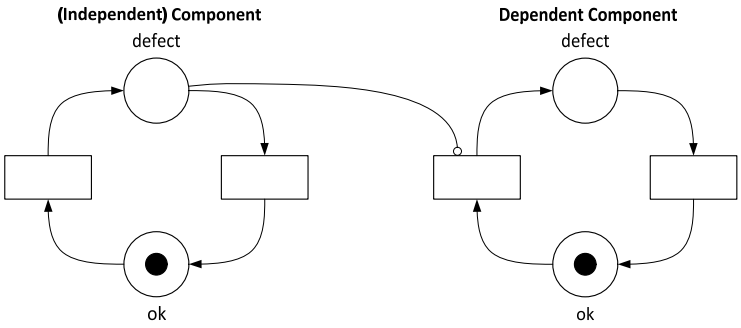

Abb. 3. Zwei Komponenten mit Abhängigkeitsbeziehung

Ist der Ausfall einer Komponente nur möglich, wenn eine andere Komponente arbeitet (z. B. Softwarefehler bei Hardwareausfall), wird dies ebenfalls über eine Inhibitor-Kante modelliert. Zwei Komponenten und eine solche Ausfallabhängigkeitsbeziehung sind in Abbildung 3 dargestellt. Die Transition zu „defect“ in der abhängigen Kom-

ponente ist genau dann deaktiviert, wenn die Stelle „defect“ in der unabhängigen Komponente markiert ist.

Eine weitere mögliche Abhängigkeit zwischen zwei Komponenten ist die so genannte Standby-Redundanz, bei der die redundante Komponente erst aktiv wird, wenn die Primärkomponente ausfällt. Dieser Vorgang nimmt eine bestimmte Übernahmezeit in Anspruch [25]. Erst nach dieser Zeit, wenn sich eine Marke in „Tookover“ befindet, ist die Sekundärkomponente in Betrieb und kann ausfallen. Ein solches Paar von Komponenten wird dann wie in Abbildung 4 modelliert.

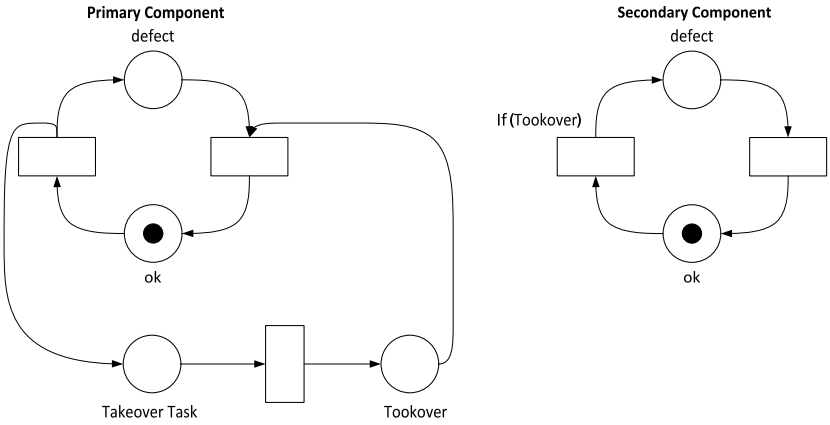

Abb. 4. Modell für Standby-Redundanz zweier Komponenten

Mit diesen Möglichkeiten können nun einzelne Komponenten sowie deren Abhängigkeiten beim Ausfall modelliert werden. Diese Modelle sowie die Beziehungen dieser Komponenten zur Bereitstellung des Services werden dann in einem erweiterten Zuverlässigkeitsblockschaltbild (engl. Reliability Block Diagram - RBD) abgebildet.

Das *Reliability Block Diagram* ist ein Modell, das entwickelt wurde, um militärischen Missionserfolg in Teilerfolge zu zerlegen [30]. Später entwickelte es sich zu einem Standardmodell bei der Verfügbarkeitsanalyse von zusammengesetzten Systemen [26]. Komponenten werden dabei in einem Graphen seriell oder parallel verbunden. Dabei ist das System online, wenn auf einem möglichen Pfad vom Wurzelknoten zu einem Blattknoten kein Fehler vorhanden ist. Das *Extended Reliability Block Diagram* (ERBD) enthält zusätzlich Ausfallabhängigkeiten zwischen den einzelnen Komponenten sowie standby-redundante Komponenten. Ein Beispiel für ein solches Diagramm ist in Abbildung 5 dargestellt.

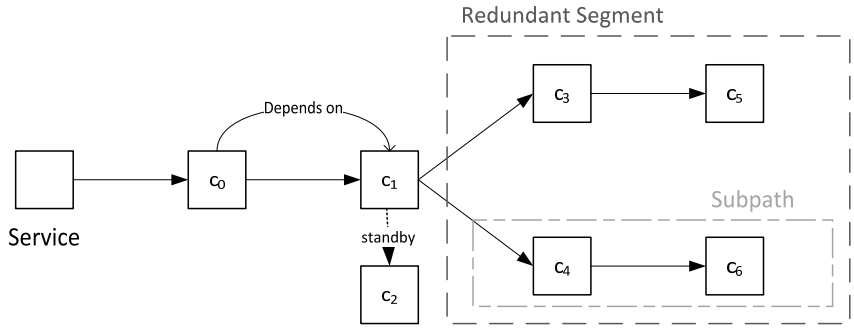

Abb. 5. Beispiel eines Extended Reliablity Block Diagrams

Komponenten, die bei alleinigem Ausfall den Erfolg der Bereitstellung verhindern, werden als kritisch bezeichnet (c_0 und c_1). Gibt es zwei oder mehr aktive Komponenten, die die gleiche Funktion erfüllen, werden diese als „(parallel-)redundant“ bezeichnet (c_3 und c_4 sowie c_5 und c_6). Zwischen den Komponenten c_0 und c_1 ist eine Ausfallabhängigkeitsbeziehung dargestellt. Weiterhin übernimmt die standby-redundante Komponente c_2 bei einem Ausfall von c_1.

Ein Teilgraph, der mehrere seriell verbundene Knoten enthält, wird als Teilpfad (Subpath) bezeichnet. Zwei oder mehrere Teilpfade, die jeweils die gleichen Komponententypen enthalten und unabhängig sind, werden insgesamt als redundantes Segment bezeichnet. Jedes redundante Segment ist somit kritisch.

Ist das ERBD aufgestellt, kann daraus ein Petri-Netz für alle Komponenten und deren Beziehungen erzeugt werden. Sobald sich der Zustand einer Komponente ändert, wird mithilfe des ERBD der Zustand des Gesamtsystems bestimmt. Das System ist ausgefallen, genau dann wenn

- Eine kritische Komponente ausgefallen ist und (noch) keine standby-redundante Komponente übernommen hat oder
- In einem redundanten Segment alle Teilpfade ausgefallen sind.

Ein Teilpfad kann dabei als eigenes System aufgefasst werden. Diese Rekursion endet, sobald ein Teilpfad nur aus (auf ihn bezogen) kritischen Komponenten besteht.

Nachdem der Zustand des Systems nun in einem Modell erfasst ist, muss noch die Nutzung des Systems modelliert werden. Das Nutzungsmodell definiert die Verwendung des im ERBD modellierten Systems als SRN. Es ist in Abbildung 6 dargestellt. Die Zeit zwischen zwei Anfragen wird als exponentialverteilt mit Mittelwert *Mean Time to Request* (MTTReq) angenommen, die Anfragen dann in der Stelle „Service Request“ gesammelt. Die Stellen „SystemUp“ und „SystemDown“ geben den aktuellen Zustand des Systems wider, wobei die zeitlose Transition „t_{down}“ genau dann aktiviert ist, wenn das System im ERBD nicht verfügbar ist. Andernfalls ist „t_{up}“ aktiviert.

Ist ein Service Request vorhanden und das System verfügbar, wird die Marke nach einer zufälligen Bearbeitungszeit vernichtet und eine Marke in der Stelle „Request

Served“ erzeugt. Fällt das System während dieser Bearbeitung aus, so werden alle Marken in „Service Request“ vernichtet und entsprechend Marken in der Stelle „Request not Served“ erzeugt.

Verschiedene Rewards können genutzt werden, um unterschiedliche Metriken für die Verfügbarkeit zu berechnen. Wenn die Verfügbarkeit anfrageorientiert geschätzt werden soll, kann ein Impulse Reward bei Eintritt einer Marke in „Request Served“ oder „Request not Served“ wie folgt berechnet werden:

$$\text{Availability}_1 = \#\text{RequestServed} / (\#\text{RequestServed} + \#\text{RequestNotServed}) \,. \qquad (2)$$

Dieser Wert kann dann als Schätzer für die Wahrscheinlichkeit, dass eine Anfrage nicht vom System beantwortet werden kann, angenommen werden.

Eine weitere Möglichkeit ist, die Verfügbarkeit mittels der normierten Betriebsdauer zu berechnen. Dies kann durch einen Rate Reward geschehen:

$$\text{Availability}_2 = t_{\text{SystemUp}} / t_{\text{total}} \qquad (3)$$

Dabei wird die Variable t_{SystemUp} mit einer Rate von 1 erhöht, falls „SystemUp“ markiert ist. Die Variable t_{total} gibt die Simulationszeit an.

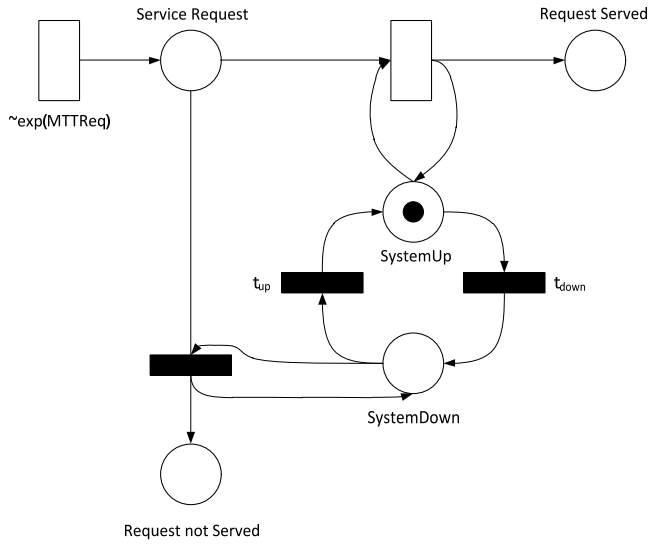

Abb. 6. Nutzungsmodell

5 Verifikation am Beispielszenario eines SAP ASP

Der in Abschnitt 4 vorgestellte Lösungsansatz wird in zwei Schritten verifiziert. Zunächst wird am Beispiel eines international agierenden SAP ASP gezeigt, dass eine reale Systemlandschaft im entwickelten Modell abbildbar ist. Das zugehörige RBD ist in Abbildung 7 dargestellt.

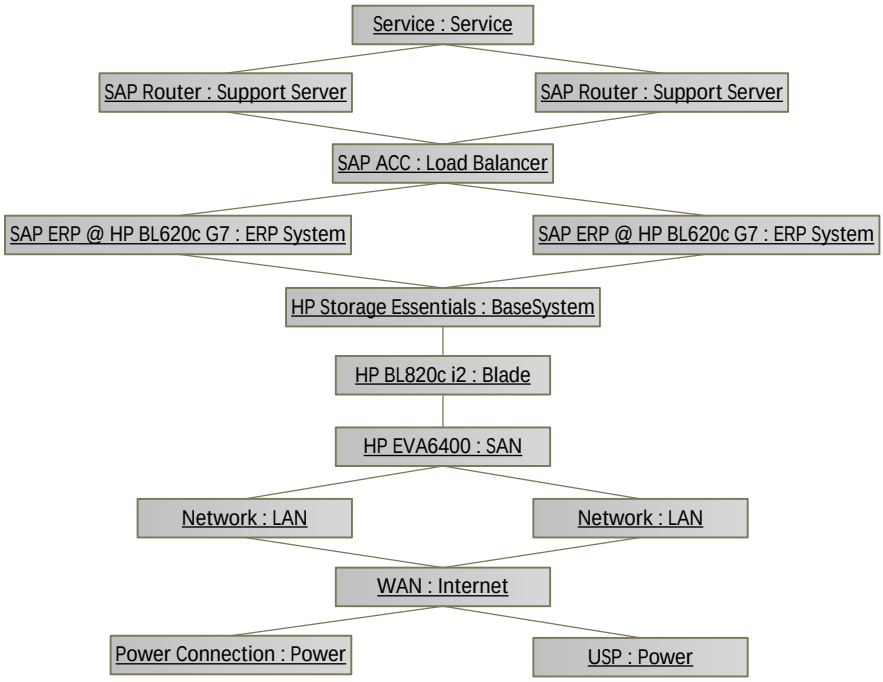

Abb. 7. Beispielszenario als Reliability Block Diagram

Den Kern des Beispielszenarios bildet ein IT-Service, der durch zwei Anwendungssysteme (SAP ERP 6.0) bereitgestellt wird. Eine Anfrage an einen Service wird zunächst von zwei redundanten Routern verarbeitet, welche diese an eines der Anwendungssysteme weiterleitet. Beide ERP-Systeme werden auf je einem Blade betrieben. Beide greifen auf das gleiche Datenbanksystem zu, welches auf einem weiteren Blade betrieben wird und die Datenhaltung auf einem Storage Area Network (SAN) übernimmt.

Neben diesen Hard- und Software-Entitäten sind weitere Komponenten Bestandteil des Beispielszenarios: Die Anwendungen HP Storage Essentials und SAP Adaptive Computing Controller, betrieben auf je einem Blade, verwalten bestimmte Teile der Infrastruktur. Weiterhin sind ein redundantes Netzwerk sowie eine Anbindung an das Internet (nicht redundant) und eine Stromversorgung Teil des Beispielszenarios. Zur Sicherung des Betriebs im Falle eines Stromausfalls ist zudem eine unterbrechungsfreie Stromversorgung (USP) vorhanden.

Im zweiten Schritt wird gezeigt, dass der Lösungsansatz plausible Ergebnisse produziert. Da zum Zeitpunkt der Erstellung dieser Arbeit keine Daten zu den Ausfallzeiten seitens des SAP ASP vorlagen, wurde für die Bestimmung der Größen MTTF und MTTR der jeweiligen Komponenten auf den Datenbestand des Los Alamos National

Lab [29] zugegriffen. Softwarefehler wurden für dieses Beispielszenario nicht berücksichtigt, da Daten über Ausfälle der ERP-Systeme nicht vorlagen.

Um den entwickelten Lösungsvorschlag zu verifizieren, wurde das Beispielszenario zunächst nur als klassisches RBD ohne Ausfallabhängigkeiten oder Standby-Redundanzen modelliert. Weiterhin wurde nur eine Fehlerart betrachtet. In diesem Fall kann die Verfügbarkeit des Gesamtsystems analytisch aus den einzelnen Verfügbarkeiten der Komponenten berechnet werden, welche wiederrum aus MTTF und MTTR berechnet werden können [31]:

$$A_c = MTTF / (MTTF + MTTR)$$

$$A_{seriell} = A_1 * A_2 \quad (4)$$

$$A_{parallel} = 1 – (1 – A_1)(1 - A_2)$$

Nachdem diese Formeln auf das Beispielszenario angewendet wurden, wurde der ermittelte Wert mit den Simulationsergebnissen, basierend auf dem Komponentennetz und dem RBD, verglichen. In Abbildung 8 sind die ermittelten Konfidenzintervalle und Mittelwerte für die Verfügbarkeit nach Formel 3 in Abhängigkeit von der Anzahl der Replikationen dargestellt. Dabei wurden 100.000 Stunden simuliert und ein Konfidenzwert von 0,99 angenommen. Der errechnete Wert 0,99311 ist zur Orientierung als gestrichelte Linie eingezeichnet.

Es ist zu erkennen, dass sich die Simulationsergebnisse bei steigender Replikationsanzahl zusehends dem errechneten Wert annähern. Damit kann mit hoher Wahrscheinlichkeit die Aussage getroffen werden, dass das entwickelte Simulationsmodell die Verfügbarkeit wie analytische Methoden schätzen kann.

Im nächsten Schritt wird für das Beispielszenario bestimmt, welchen Effekt die Einführung von Ausfallabhängigkeiten hat. Dazu wurden verschiedene Abhängigkeiten, z. B. die aller Hardwarekomponenten von der Stromversorgung oder die der Softwarekomponenten von der ausführenden Hardware, modelliert.

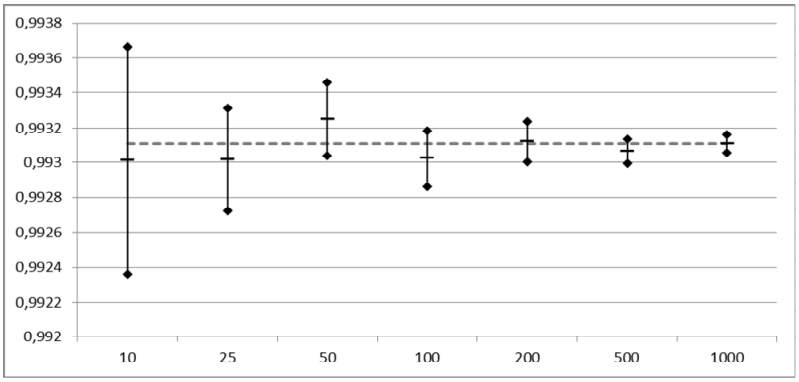

Abb. 8. Konfidenzintervalle der Verfügbarkeit in Relation zur Anzahl der Replikationen

Bei 1000 Replikationen mit 100.000 Stunden Simulationszeit ergibt sich dabei ein 0,99-Konfidenzintervall von [0,99303;0,99313] mit Mittelwert 0,99308. Dies bedeutet, dass die mittlere Verfügbarkeit hier minimal kleiner ist als die mit der Annahme unabhängiger Komponenten. Die Tatsache, dass dieser Unterschied erst in der vierten Stelle nach dem Komma auftritt, zeigt, dass der Effekt von Ausfallabhängigkeiten gering ist. In der Anwendungsdomäne IT-Services mit hohen Verfügbarkeitsgarantien kann jedoch auch ein solch kleiner Unterschied zur Verletzung eines Service Level Agreements führen.

6 Schlussfolgerungen

In diesem Artikel wurde ein Lösungsansatz zur Vorhersage der Verfügbarkeit von IT-Service mittels Diskreter-Ereignis-Simulation vorgestellt. Dafür wurde die Nutzung des IT-Services, dessen Bereitstellungsabhängigkeiten von Komponenten der Anwendungssystemlandschaft sowie der Ausfall einzelner Komponenten modelliert und in eine hierarchische Struktur eingeordnet. Der entwickelte Ansatz ermöglicht die Vorhersage der Verfügbarkeit für Systeme, in denen die Komponenten unabhängig voneinander ausfallen. Diese Fähigkeit wurde mittels eines Vergleiches mit analytischen Verfahren verifiziert. Weiterhin ermöglicht der Lösungsansatz so genannte Ausfallabhängigkeiten zwischen einzelnen Komponenten zu modellieren. Dies bedeutet, dass eine abhängige Komponente nur dann ausfallen kann, wenn die Komponente, von der sie abhängig ist, in Betrieb ist. Auch Standby-Redundanz und verschiedene Fehlerarten sind im Modell abbildbar.

Der Vergleich zwischen dem unabhängigen Ansatz und dem mit Ausfallabhängigkeiten am Beispiel eines Ausschnittes der Anwendungssystemlandschaft des SAP ASP ergab, dass die Unterschiede in der vorhergesagten Verfügbarkeit vorhanden, aber sehr gering sind. Dennoch sind gerade bei der Verfügbarkeit auch hintere Nachkommastellen wichtig, da die garantierten Niveaus oft sehr nah an 100 % liegen.

Damit stellt diese Arbeit ein generisches Modell zur Verfügung, um die Verfügbarkeit von IT-Services vorherzusagen. Durch Modularisierung und durch die freie Wahl des Abstraktionsniveaus können beliebige Anwendungssystemlandschaften abgebildet werden. Die Möglichkeit, dabei Abhängigkeiten beim Ausfall zwischen den einzelnen Komponenten abzubilden, löst das signifikante Problem bisheriger Ansätze, die unabhängige Ausfälle annehmen [23]. Durch die Lösung mit Simulation wird das Problem der Zustandsexplosion vermieden, das die Handhabbarkeit zustandsbasierter Modelle erschwert hat. Weiterhin kann das Modell beliebige Verteilungen für die Ausfallzeiten der Komponenten abbilden. Damit ist eine weitere Einschränkung der zustandsbasierten Verfügbarkeitsmodelle, die in der Regel die Markov-Eigenschaft annehmen, egalisiert. Ein Nachteil des Ansatzes ist, dass wie bei analytischen Verfahren notwendige Kenntnisse über die Struktur des untersuchten Systems vorliegen müssen. Des Weiteren kann die Lösung durch Simulation viel Zeit beanspruchen und liefert nur approximative Ergebnisse.

Die nächsten Forschungsarbeiten auf diesem Gebiet werden die Erweiterung des vorgeschlagenen Lösungsansatzes beinhalten. Bspw. könnten die Abhängigkeiten

zwischen den Komponenten um zusätzliche Relationen erweitert werden. Neben der Ausfallabhängigkeit ist z. B. denkbar, dass Komponenten bei der Wiederinbetriebnahme einer anderen Komponente eine erhöhte Fehlerwahrscheinlichkeit aufweisen (z. B. Stromspitzen nach einem Stromausfall). Auch könnten redundante Komponenten, abhängig von ihrer Auslastung, eine geringe Ausfallwahrscheinlichkeit besitzen.

Parallel dazu wird eine Datensammlung beim SAP ASP durchgeführt. Diese hat das Ziel, das entwickelte Modell und dessen mögliche Erweiterungen an einem realen Anwendungsbeispiel zu validieren und somit die Vorteile gegenüber Verfahren mit der Annahme des unabhängigen Ausfalls von Komponenten zu belegen.

Ist das Vorhersagemodell für die Verfügbarkeit validiert, könnte das Modell auch für die Vorhersage anderer nicht-funktionaler Merkmale eingesetzt werden, bspw. um die Performanz eines Systems abhängig von Ausfällen zu bestimmen. Diese Modelle können dann einen Service-Anbieter bei Entscheidungen betreffend seiner Anwendungssystemlandschaft unterstützen.

Literatur

1. Stahlknecht, P., Hasenkamp, U.: Einführung in die Wirtschaftsinformatik. Springer, Berlin Heidelberg (2002)
2. Lewis, L.: Service Level Management Definition, Architecture and Research Challenges. In: IEEE Global Telecommunications Conference, pp. 1974–1978. IEEE Computing Society (1999)
3. Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R.H., Kowinski, A., Lee, G., Patterson, D.A., Rabkin, A., Stoica, I., Zaharia, M.: Above the Clouds: A Berkeley View of Cloud Computing. Technical report, Electrical Engineering and Computer Sciences, University of California at Berkeley (2009)
4. Caron, E., Rodero-Merino, L., Desprez, F., Mresan, A.: Auto-Scaling, Load Balancing and Monitoring in Commercial and Open-Source-Clouds. Technical report, Laboratoire de l'Informatique du Parallélisme, University of Lyon (2012)
5. Zhang, Q., Cheng, L., Boutaba, R.: Cloud Computing: state-of-the-art and research challenges. Journal of Internet Services and Applications 1, 7–18 (2010)
6. Hennessy, J.L.: The Future of Systems Research. Computer 32, 27–33 (1999)
7. Aberdeen Group: Datacenter Downtime - How Much Does It Really Cost?. Technical report, www.stratus.com/~/media/Stratus/Files/Library/AnalystReports/AberdeenDatacenterDowntimeCost.pdf
8. Büst, R.: CloudUser.de-Artikel über Ausfälle von Amazon Web Services, http://clouduser.de/news/aws-erneut-mit-ausfall-wieder-ein-stromausfall-wieder-in-north-virginia-schwere-sturme-sind-die-ursache-12547
9. Günther, R.: Blog-Beitrag über Stromausfall bei Amazon und Microsoft, http://www.rgblog.de/stromausfall-rechenzentren-amazon-microsoft-cloud-computing-europaweit-lahmgelegt/
10. Grabski, B., Günther, S., Herden, S., Krüger, L., Rautenstrauch, C., Zwanziger, A.: Very Large Business Applications. Informatik-Spektrum 30, 259–263 (2007)
11. Hamel, G.: Disadvantages of Going Green, eHow Money Online-Magazine www.ehow.com/list_6182126_disadvantages-going-green.html
12. Hevner, A.R., March, S.T., Park, J., Ram, S.: Design Science in Information Systems Research. MIS Quarterly 28, 75–106 (2004)

13. Wilde, T., Hess, T.: Methodenspektrum der Wirtschaftsinformatik – Überblick und Portfoliobildung. Technical report, Institut für Wirtschaftsinformatik und Neue Medien, Ludwig-Maximilians-Universität München (2006)
14. Addicks, J.S., Steffens, U.: Supporting Landscape Dependent Evaluation of Enterprise Applications. In: Multikonferenz Wirtschaftsinformatik, pp. 1815–1825. GITO, Berlin (2008)
15. Matthes, F.: Softwarekartographie. Informatik-Spektrum 31, 527–536 (2008)
16. Siedersleben, J.: SOA revisited. Wirtschaftsinformatik 49, 110–117 (2007)
17. Ahlemann, F., Stettiner, E., Messerschmidt, M., Legner, C.: Strategic Enterprise Architecture Management – Challenges, Best Practises, and Future Developments. Springer, Berlin Heidelberg (2012)
18. Balzert, H.: Lehrbuch der Softwaretechnik – Entwurf, Implementierung, Installation und Betrieb. 3rd Edition, Spektrum, Heidelberg (2012)
19. Ackermann, J., Brinkop, F., Conrad, S., Fettke, P., Frick, A., Glistau, E., Jaekel, H., Kotlar, O., Loos, P., Mrech, H., Ortner, E., Raape, U., Overhage, S., Sahm, S., Schmietendorf, A., Teschke, T., Turowski, K.: Vereinheitlichte Spezifikation von Fachkomponenten. Memorandum des Arbeitskreises 5.10.3 der GI (2002)
20. Berger, T.G.: Konzeption und Management von Service-Level-Agreements für IT-Dienstleistungen. Dissertation, ULB Darmstadt (2005)
21. Sommerville, I.: Software Engineering. 8th Edition, Pearson Education München (2007)
22. Kaiser, T.: Methodik zur Bestimmung der Verfügbarkeit von verteilten anwendungsorientierten Diensten. Herbert Utz, München (1999)
23. Goseva-Popstojanova, K., Trivedi, K.S.: Architecture-based Approach to Reliability Assessment of Software Systems. Performance Evaluation 45, 179–204 (2001)
24. Raftery, A.E.: A Model for High-order Markov Chains. Journal of the Royal Statistical Society B 47, 528–539 (1985)
25. Leangsuksun, C., Shen, L., Liu, T., Song, H., Scott, S.L.: Availability Prediction and Modeling of High Availability OSCAR Cluster. In: 5th IEEE International Conference on Cluster Computing, pp. 380–386. IEEE Computer Society (2003)
26. Muppala, J.K., Ciado, G., Trivedi, K.S.: Stochastic Reward Nets for Reliability Prediction. Communications in Reliability, Maintainability and Serviceability 1, 9–20 (1994)
27. Ciardo, G., Muppala, J.K., Trivedi, K.S.: SPNP: Stochastic Petri Net Package. In: Proceedings of the 3rd International Workshop PNPM, pp. 142–151. IEEE Computer Society (1989)
28. Malek, M., Hoffmann, G.A., Milanovic, N., Brüning, S., Meyer, R., Milic, B.: Methoden und Werkzeuge zur Verfügbarkeitsermittlung. Informatik-Berichte, Humboldt-Universität Berlin (2007)
29. Los Alamos National Lab: Records of cluster node outages, workload logs and error logs, http://cfdr.usenix.org/data.html (2012)
30. Military Standard: Reliability Modeling and Prediction (MIL-STD-756B). U.S. Department of Defence (1981)
31. Shooman, M.L.: Reliability of Computer Systems and Networks – Fault Tolerance, Analysis, and Design. John Wiley & Sons, New York (2002)