Martin Rost, Andreas Speck

Modellgestützte Validierung von WebService-Ketten

Lässt sich das Konzept der Daten verarbeitenden Stelle auf frei konfigurierbare WebServices noch anwenden? Der Beitrag stellt eine Idee vor, wonach dies dadurch gelingt, indem die die Verantwortung tragende Organisation ihre WebService-Kette nach einem automatisierten Test und Freigabeverfahren zusammenstellt und den Ablauf kontrolliert.

1 Hintergrund

Eine wesentliche Idee, die das Konzept von WebServices trägt, besteht darin, dass man sich als Datenverarbeiter bestimmte IT-Dienstleistungen („Services“) von unterschiedlichen Anbietern zeitlich, organisatorisch und technisch flexibel zusammenstellt, nach dem Motto: Outtasking on the fly. Vorausgesetzt, dass eine durchgängige automatisierte Verkettung von einzelnen IT-Leistungen technisch funktioniert und betriebswirtschaftlich attraktiv ist, stellt sich aus (datenschutz-)rechtlicher Sicht in diesen Fällen die Frage, ob eine Daten verarbeitende Stelle der bei ihr verbleibenden Verantwortung für die Verarbeitung personenbezogener Daten durch frei konfigurierbare WebService-Ketten noch gerecht werden kann.

Prof. Andreas Speck

ist Leiter der Arbeitsgruppe Angewandte Informatik (Wirtschafts-informatik) an der Christian-Albrechts-University zu Kiel. Forschungsschwerpunkte: Entwicklung und Betrieb vernetzter Systeme, insbesondere Modellierung und Validierung Service-orientierter Systeme.
E-Mail: andreas.speck@email.uni-kiel.de

Martin Rost

Mitarbeiter im Referat „Systemdatenschutz“ beim Unabhängigen Landeszentrum für Datenschutz Schleswig-Holstein.
E-Mail:
martin.rost@datenschutzzentrum.de

2 Das Problem

Datenschützer müssen einfordern, dass Systeme zur Datenverarbeitung beherrschbar sind. Voraussetzung für Beherrschbarkeit von Systemen ist deren Transparenz, sowohl für Betreiber als auch interne und externe Kontrolleure. In diesem Sinne schreiben Bizer et. al. in einer Studie: „Chancen und Risiken der neuen Service-orientierten Architekturen (SOA) liegen im Bereich der *Beherrschbarkeit* von SOA-Lösungen und des Nachvollzugs *verantwortlichen Handelns* mit SOA-Lösungen.“ [1]. Nachfolgend listet die Studie Faktoren, um Beherrschbarkeit und Verantwortlichkeit zu operationalisieren (vgl. [1], Seite III):

- Verallgemeinerung von Zusicherungen: Datenschutzpolicies, SLA, QoS, Dienstleistungsverträge;
- Protokolle: Revisionsfähigkeit, Auskunfterteilung an Bürger/Kunden, Überprüfung von Zusicherung für Bürger/Kunden sowie Verfahren durch Audits;
- Aggregation von Zusicherungen: verlangt eine standardisierte Sprache;
- Nutzungskompetenz: Anwender müssen mit Zusicherungen und Protokollen umgehen und zur Unterrichtung, Auskunftsersuchen und Überprüfung fähig sein.

Die Risiken von SOAs, die typischerweise als WebServices ausgelegt sind, sind dabei nicht zu unterschätzen [6]:

- *Transparenz-Erosion*: Betroffene können den „Zoo“ der Dienste und der Dienstanbieter nicht mehr überblicken;
- *Trennungs- und Zweckbindungsgebot-Erosion*: Die standardisierte Verkettung bzw. Verkettbarkeit separierter Datenbestände erleichtert eine Datenverarbeitung an jeder zuvor festgelegten Zweckbindung vorbei;
- *Verantwortungserosion*: Daten werden ohne Rechtsgrundlagen verarbeitet.

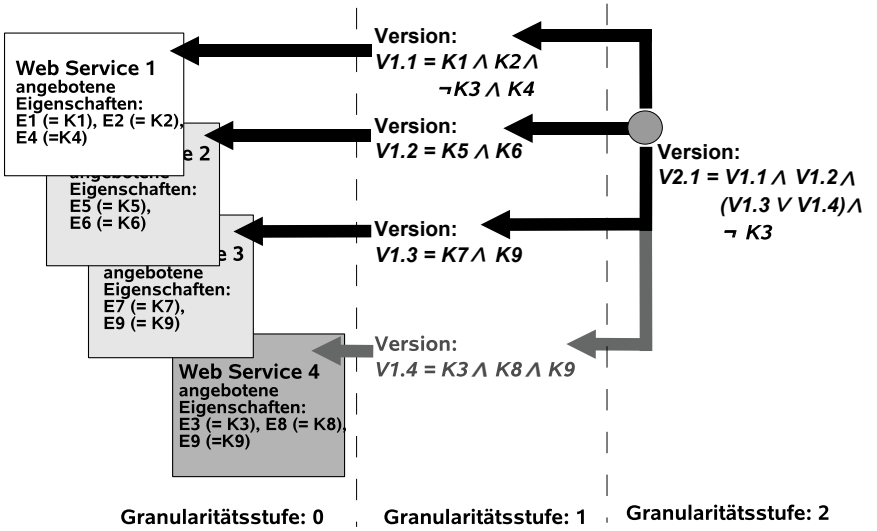
Die Chancen sind allerdings genauso herauszuheben:

- Die Offenlegung von Daten, Formaten und Prozessen sowie Protokolle, die das korrekte Funktionieren nachweisen [9], steigern die Transparenz von Datenverarbeitung in Organisationen;
- Durch das Kapseln von Aufgaben und einem engen Zuschnitt von Verantwortung lässt sich die Zweckbindung besser durchsetzen;
- Die Datenqualität kann steigen, weil Daten dezentral erfasst und gepflegt werden;
- Und das Datenschutzmanagement lässt sich auch zunehmend besser automatisieren.

3 Die Idee

Um diese Möglichkeiten real werden zu lassen und WebService-Ketten beherrschbar zu machen, sind automatisiert zugängliche Zusicherungen bezüglich der Eigenschaften von Services notwendig. Diese Zusicherungen finden in WebServices als WebService-Policy beziehungsweise WebService-Security-Policy ihren Ausdruck. Bei diesen Zusicherungen handelt es sich um Fokussierungen von Bemühungen, die je nach Kontext auch als „Designvorgaben“, „Schutzziele“ oder „Erfolgsfaktoren“ bezeichnet werden und zunehmend Bestandteile von Verträgen oder gesetzlichen Bestimmungen werden.

Abbildung 1 | Versionsmodell der statischen Hierarchie



Schutzziele müssen, im Hinblick auf Datensicherheit und Datenschutz, mindestens die folgenden sechs Aspekte ansprechen, aus denen sich dann gegebenenfalls weitere Schutzziele (Unbeobachtbarkeit, Abstreitbarkeit etc.) ableiten lassen:[1]

- Verfügbarkeit
- Integrität
- Vertraulichkeit
- Kontingenz
- Transparenz
- Nichtverkettbarkeit

Die Schutzziele werden dann über Schutzmaßnahmen (z. B. Redundanz, Verschlüsselung, Hashwertbildung, Konzeption und Protokollierung, Dokumentation, Identitätsmanagement) operationalisiert, die wiederum über Schutzbedarfsfeststellungen skalierbar sind.

Die für diesen Artikel zentrale Idee besteht nun darin, dass eine Organisation, bevor sie einen WebService bzw. eine WebService-Kette nutzen will, über eine Applikation die gewünschten Services triggert und auf deren Eigenschaften beziehungsweise Zusicherungen, sinnvollerweise bezüglich der oben aufgeführten elementaren Schutzziele hin automatisiert untersucht. Nach Auswertung der Antworten der Services stellt diese Applikation dann Empfehlungen für in Frage kommende Ketten zusammen, die vom Verfahrensverantwortlichen dann umgesetzt wird. Nach der Zusammenstellung erlaubt die Applikation zusätzlich, die Ausführungen der einzelnen Services zu kontrollieren.

Eine Voraussetzung für eine rechtlich angemessene, korrekte Zusammenstellung einer WebService-Kette besteht darin, dass die Applikation über eine Modellierung mit konsistenten Anforderungen verfügt, die diese Organisation aufgrund der für sie gültigen Rechtsdomäne stellen muss. Wünschenswert wäre dann, dass es für verschiedene Geschäftsbereiche jeweils standardisierte Modellierungen gäbe.

Die bislang geforderte Dokumentation von IT-Systemen und deren Sicherheit, die eine Konzeptionsphase, ein Monitoring des aktuellen Systemzustands sowie eine revisionsfeste Protokollierung vorsieht, würde deswegen erweitert und teilweise ersetzt werden durch Aktivitäten des Policy-Making, Policy-Matching und (internen oder externen) Policy-Controlling.

Weil die Daten verarbeitende Stelle über die Applikation zur Auswahl der WebServices Einfluss auf eine gesetzeskonform angelegte Datenverarbeitung nimmt, kann sie ihrer Verantwortung für die Datenverarbeitung tatsächlich gerecht werden. Und kann Entscheidungen für einen „Qualitätsdienst" auch organisationsintern belegen, wenn sich ein Systemverantwortlicher beispielsweise dafür rechtfertigen muss, dass er nicht den billigsten Anbieter irgendwo auf der Welt genommen hat, weil dieser nachgewiesenermaßen nicht den abgestimmten und entsprechend modellierten Schutzziel-Anforderungen genügt. Und organisationsextern kann ein Auftragnehmer (beispielsweise ein Rechenzentrum) gegenüber einem Auftraggeber (beispielsweise einem Ministerium) oder Kontrollbehörden nachweisen, dass Daten gesetzeskonform, d. h. durch kontrollierte und prüffähige Prozesse, verarbeitet wurden.

4 Validierung von Web-Service-Ketten im Detail

Die Qualitätssicherung von Systemen ist ein zentrales Anliegen der planmäßigen beziehungsweise ingenieurmäßigen Entwicklung von Systemen. Die Mechanismen der Qualitätssicherung sind für den vorgestellten Ansatz mit Bezug zu Web-Services wesentlich und sollen nachfolgend im Vordergrund stehen.

Es stehen zwei Vorgehensweisen zur Prüfung von Software-Systemen und den entsprechenden Schutzzielen zur Verfügungen: manuelle und automatisierte Prüfung. Manuelle Prüfverfahren sind dann von Vorteil, wenn keine formalen (oder nur ungenaue) Informationen über das zu validierende System vorliegen. Allerdings kann auf diesem Weg nur eine relativ kleine Menge von Informationen überprüft werden. Automatisierte Verfahren hingegen ermöglichen die Prüfung von großen Informationseinheiten. Entsprechende Algorithmen können große Informationsmengen durchsuchen und analysieren. Allerdings müssen die Informationen dann formal, d. h. maschinenverständlich, vorliegen.

Damit ergibt sich eine Aufgabenteilung: Die Überprüfung des Codes gegenüber den in der Entwurfsphase erstellten Modellen erfolgt von Hand. Die Prüfung auf Modellebene erfolgt automatisiert und auf der Basis einer erfolgreichen manuellen Code-Prüfung.

Im Weiteren sollen die Möglichkeiten der automatisierten Prüfung kurz betrachtet werden. Eine vollautomatische Verifikation soll nicht erfolgen, da diese eine vollständige Spezifikation erforderte. Das System wird in der Terminologie der Prüfungssysteme gegenüber einer Spezifikation, die die entsprechenden Regeln enthält geprüft.

Grundsätzlich gibt es zwei Aspekte, die automatisiert geprüft werden können: die statischen Hierarchien der einzelnen Elemente und die dynamischen Interaktionen zwischen diesen Elementen. Diese Aufspaltung ergibt sich aus der üblichen Einteilung der Modelltypen (statische und dynamische) der verwendeten einschlägigen Modellierungskonzepte (z. B. UML oder ARIS). Ebenso sind die Validierungswerkzeuge eher für den einen oder den anderen Aspekt optimiert.

1 Zur Systematik von Schutzzielen siehe den Beitrag von Rost / Pfitzmann, ebenfalls in diesem Heft.

4.1 Statische Abhängigkeiten in Hierarchien

WebService Systeme sind wie andere Systeme der Informationstechnik meist hierarchisch gegliedert. Hierarchien ergeben sich dadurch, dass sich einzelne Dienste aus anderen Diensten zusammensetzen. In der Informatik sind seit langem Möglichkeiten bekannt, statische Abhängigkeiten zu beschreiben und zu validieren. Das von uns vorgeschlagene Konzept ist die Versionierung [3].

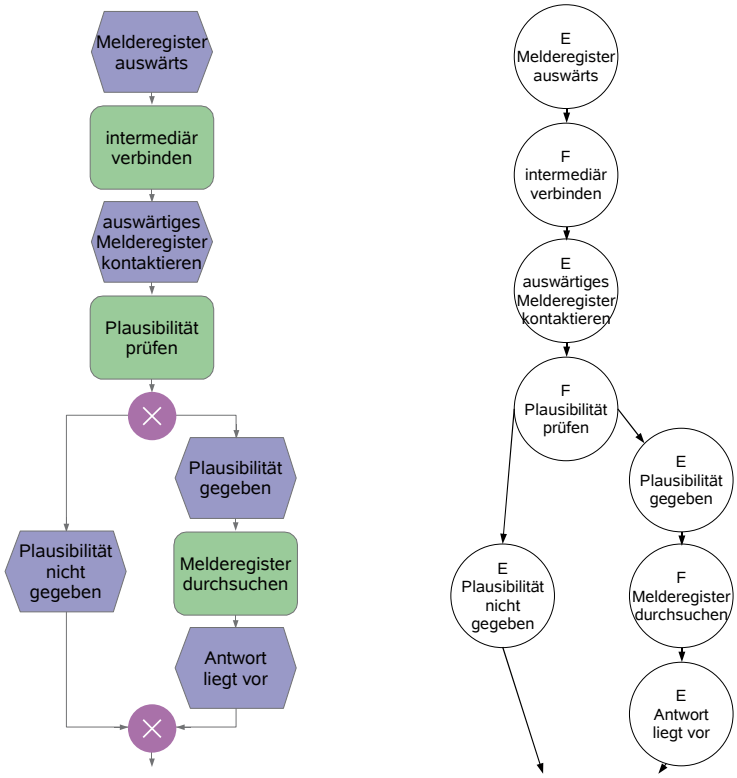
Die einzelnen Bausteine der WebServices werden als Versionen betrachtet, aus denen dann übergeordnete Versionen erstellt werden können. Die Gültigkeit, d. h. die Korrektheit des Vorhandenseins einer Version wird durch Konditionen geregelt (siehe Abb. 1). Damit kann das Versionssystem durch Boolesche Gleichungen dargestellt werden, die sich wiederum in Beziehung setzen lassen und somit die Korrektheit des gesamten Gleichungssystems ermittelbar machen [7].

Die Regeln über die Korrektheit des Systems sind den einzelnen Versionen als Konditionen zugeordnet. Im Wesentlichen sind diese Konditionen so formuliert, dass entweder weitere Versionen zur Gültigkeit benötigt werden (requires) oder ausgeschlossen werden (mutual exclude). Zusätzlich bestehen Alternativen und Optionen. Damit können praktisch alle Formen der statischen Abhängigkeiten ausgedrückt werden. Als Werkzeuge zur Überprüfung können Theorembeweiser oder Constraint Solving Systeme (zur Lösung der Booleschen Gleichungssysteme) verwendet werden. Es existieren hierzu auch kommerzielle Werkzeuge (wie z. B. das ILOG Business Rule Managament System).

4.2 Dynamische Interaktionen

Im Unterschied zu den statischen Abhängigkeiten sind dynamische Interaktionen schwieriger zu prüfen. Sie stellen zum einen den Prozessablauf dar, bei dem temporale Abhängigkeiten gelten, d. h. die Abhängigkeiten sind nicht permanent, sondern treten nur zu bestimmten Zeitpunkten auf. Zum anderen ist es nur bedingt sinnvoll, einzelne Abläufe in Unterversionen zu gruppieren und diese nicht mehr zu betrachten. Bei der statischen Betrachtung erhöht eine derartige Abstraktion die Übersicht, bei einer Prüfung dynamischer Abläufe können dadurch jedoch Fehler versteckt und nicht erkannt

Abbildung 2 | Umwandlung eines WebService Modells in ein formales Automatenmodell (Auszug aus Beispiel in Abbildung 4)



EPK des Web Service *auswärtiges Melderegister durchsuchen*

Zustandsmodell für das EPK des Web Service *auswärtiges Melderegister durchsuchen*

Abbildung 3 | Erfüllung temporaler Ausdrücke (CTL)

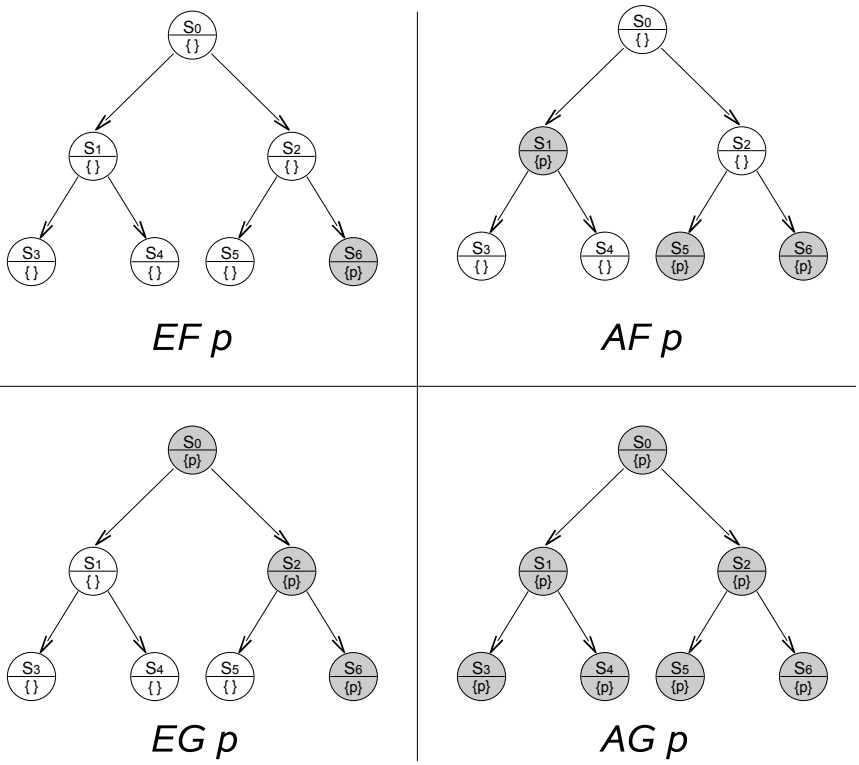
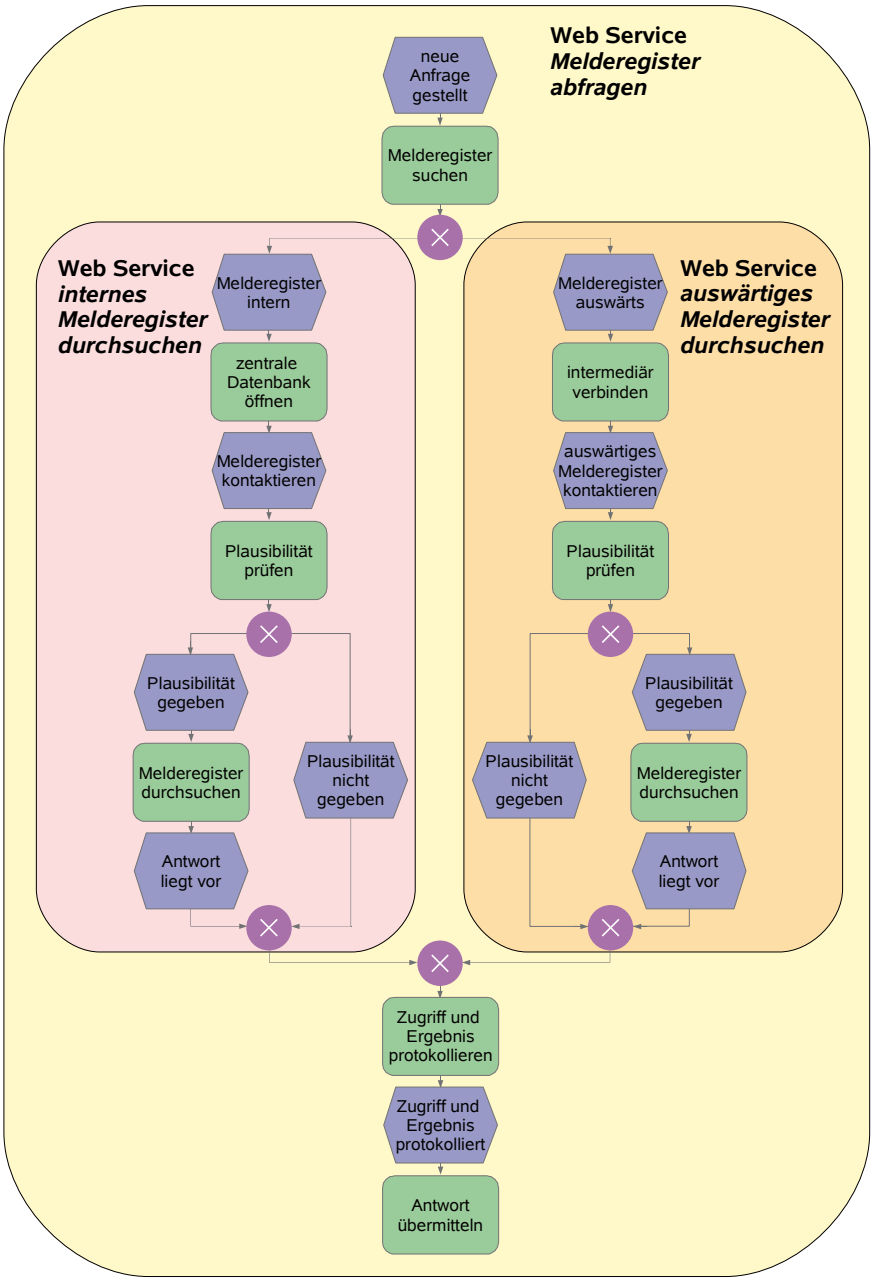


Abbildung 4 | Beispiel für ein einfaches WebService System



werden. Daher ist ein solches Gruppieren nicht sinnvoll. In der Praxis kommt erschwerend hinzu, dass die Forschung in der Validierung der dynamischen Interaktionen noch nicht sehr fortgeschritten ist und zudem keine kommerziellen Werkzeuge vorliegen.

Eine Technologie zur Lösung stellt die Modellprüfung dar. Hierbei werden die zu prüfenden dynamischen Systeme als Automaten modelliert. Automatenmodelle (Zustands-Übergangsmodelle) sind formale Modelle von dynamischen Abläufen und können daher von automatischen Systemen wie beispielsweise den Modellprüfern gelesen und interpretiert werden. Abbildung 2 zeigt eine Überführung einer Ereignis-Prozessketten-Darstellung, als eine Modellierungsform für WebServices [4], in ein Automatenmodell. Diese Automatenmodelle werden dann gegenüber den Spezifikationen mit temporalen Regeln geprüft. Die temporalen Regeln werden in temporaler Logik dargestellt, die eine Erweiterung der Booleschen Logik um temporale Operatoren ist.

Die Abbildung 3 zeigt temporale Operatoren, die der CTL („Computational Tree Logic" [2]) entsprechen, und deren Erfüllung in Automatenmodellen:

- *EF (exists in the future)*: Es muss auf einem Pfad mindestens einmal in der Zukunft vorkommen.
- *AF (always in the future)*: Es muss auf jedem Pfad mindestens einmal in der Zukunft vorkommen.
- *EG (exists globally)*: Es muss auf einem Pfad immer in der Zukunft vorkommen.
- *AG (always globally)*: Es muss auf jedem Pfad immer in der Zukunft vorkommen.

Dazu gibt es noch die Operatoren neXt und Until:

- *EX (exists next)*: Eine Eigenschaft muss ein einem direkt nachfolgenden Zustand gelten.
- *EU (exists until)*: Es muss ab einem bestimmten Zustand immer in einem möglichen Pfad gelten.

Die WebServices können nun in derartige Automatenmodelle überführt und dann gegenüber den temporalen Regeln, formuliert in temporaler Logik, geprüft werden

Ein mögliches Werkzeug zur Prüfung ist ein Modellprüfer (SMV „Symbolic Model Verifier", vgl. [8]). Eine weitere Verbesserung stellt ein Checking-Framework dar, das die Umwandlung der WebServices in Automaten optimiert und den Einsatz von weiteren Prüftechnologien unterstützt (vgl. [5]).

4.3 Beispiel für Validierung von WebService-Ketten

An einem einfachen Beispiel soll nun gezeigt werden, wie die beschriebenen Techniken zur Prüfung von WebService Systemen eingesetzt werden können.

Abbildung 4 zeigt ein einfaches WebService System für die Abfrage von Meldedaten. Es besteht aus dem eigentlichen WebService *Melderegister abfragen*. Dieser nutzt zwei eingelagerte WebServices (*internes Melderegister durchsuchen* und *auswärtiges Melderegister durchsuchen*). Ein Aufruf eines externen WebService durch *auswärtiges Melderegister durchsuchen* wird hier nicht weiter verfolgt.

Die Prüfung der statischen Abhängigkeiten ist in diesem Beispiel trivial. Es muss lediglich geprüft werden ob gilt:

Melderegister abfragen = internes Melderegister durchsuchen ^ auswärtiges Melderegister durchsuchen.

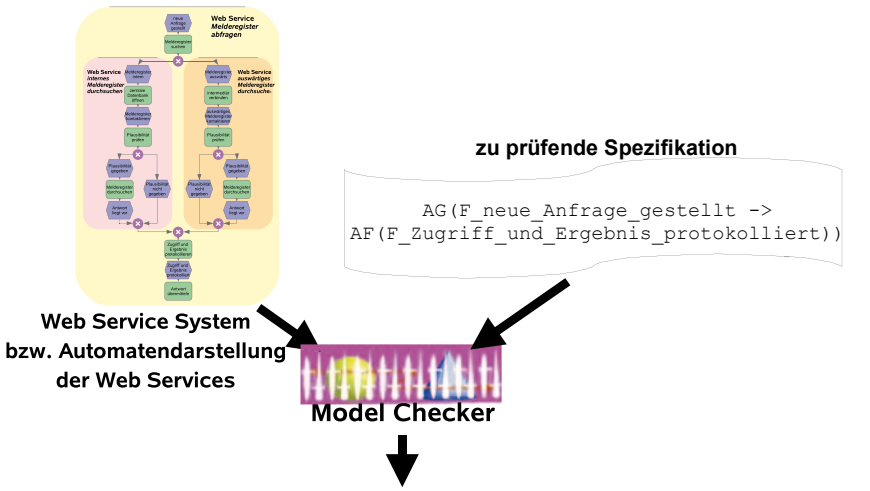
Dieser Ablauf wird gegenüber Spezifikationen, die als CTL-Formeln ausgedrückt sind, verifiziert. Exemplarisch sei hier geprüft, ob in jedem Fall das Ereignis *Zugriff und Ergebnis protokolliert* eintritt und dieses nicht umgangen werden kann und damit das entsprechende Schutzziel erreicht wird. Abbildung 5 zeigt die entsprechende CTL Formel und das Ergebnis,

das der Modellprüfer SMV bei einer derartigen Abfrage liefert.

In diesem Beispiel wird also eine Kombination von mehreren WebServices geprüft. Zwar kann ein derartig kleines Beispiel noch leicht von Hand validiert werden. Allerdings geht diese Übersichtlichkeit verloren, sobald weitere WebServices hinzukommen. Die dann sehr schnell unübersichtlich werdenden Modelle können von den Prüfwerkzeugen zuverlässig durchsucht werden. Schutzziele, die als formale Regeln vorliegen, können damit automatisiert geprüft werden.

Liegen die entsprechenden Automatenmodelle der WebServices vor, dann kann eine Prüfung einer WebService-Kette auf Basis der Konkatenation dieser (Teil-)Automatenmodelle vor der eigentlichen Ausführung erfolgen. Die Konformität einer WebService-Kette gegenüber Schutzzielen kann damit sichergestellt werden.

Abbildung 5 | Validierung des WebService System (aus Abb. 4) mittels Model Checker



5 WebServices Controlling Application

Wenn dieser Ansatz trägt, dann gilt es, eine integrierte Applikation zu entwickeln, die ein Management von WebServices in WebService-Ketten erlaubt. Dazu zählenden folgende Funktionalitäten:

- Modellierung einzelner WebServices sowie WebService-Ketten: Dazu ist es nötig, den Zweck der Datenverarbeitung in funktionale Teile sowie die entsprechenden Schutzziele zu zerlegen.
- Zerlegen einer Gesamtanforderung in einzelne Services und automatisierte Anfragen, ob Services mit den gewünschten Eigenschaften, beziehungsweise bezüglich der Aspekte der Verfügbarkeit, Integrität, Vertraulichkeit, Transparenz und Nichtverkettbarkeit, zur Verfügung stehen.
- Validierung der modellierten sowie der per Anfrage von WebServices zugesicherten Eigenschaften durch automatisierte Tests.
- Controlling der Ausführungen der einzelnen Services in der WebService-Kette und Protokollierung der einzelnen Aktivitäten.
- Controlling der Ausführung mehrerer oder aller WebServices in einer Kette: Damit ergibt sich die Möglichkeit Schutzziele in Teilketten oder gesamten WebService-Ketten zu validieren.
- Bewertung der Aktivitäten der einzelnen Services nach Abschluss anhand von KPIs zwecks zukünftig möglicher Reaktivierung sowie zu Revisionszwecken.

6 Fazit

Durch den Einsatz automatisierter Prüftechniken können technisch verfügbare Zusicherung effizient angefragt, getestet und verifiziert werden.
Schutzziele, die lediglich das Vorhandensein von Eigenschaften und deren Eingliederung in eine statische Hierarchie betreffen, können auf Basis Boolescher Logik mit einem Versionsmodell dargestellt und verifiziert werden. Schutzziele, die den dynamischen Ablauf von Systemen betreffen, können mit einer einfachen Erweiterung der Booleschen Logik um Operatoren der zeitlichen Abhängigkeiten formuliert werden. Wenn WebService-Systeme in Automatenmodellen zur Verfügung stehen können auch diese mittels Modellprüfern validiert werden.

Tools zur Entwicklung und zum Testen von Modell-WebServices sind vorhanden. Tools zum Testen einzelner WebServices, zum automatisierten Auswerten und zum Zusammenstellen sowie zur Überprüfung von WebServices in WebService-Ketten sind zu entwickeln.

Literatur

[1] Bizer, J.; Grimm, R.; Staab, S.: *SOAinVO – Chancen und Risiken von Service-orientierten Architekturen in Virtuellen Organisationen*, Version 1.0, 2007 https://www.datenschutzzentrum.de/soa/SOAinVO-Analyse.pdf

[2] Clark, E.M.; Grumberg, O.; Peled, D.A.: *Model Checking*, The MIT Press, 1999.

[3] Conradi, R.; Westfechtel, B.: *Version Models for Software Configuration Management*. ACM Surveys, 30, 1998; S. 232 – 282.

[4] Feja, S.: *Service-orientierte Integration im E-commerce*. In: Informatiktage 2007: Fachwissenschaftlicher Informatik-Kongress, B-IT Bonn-Aachen, International Center for Information Technology: Bonn, 2007.

[5] Fötsch, D.; Pulvermüller; E.; Rossak, W.: *Modeling and Verifying Workflow-based Regulations*, In: Proceedings of International Workshop on Regulations Modelling and their Validation and Verification (REMO2V '06) in conjunction with CAiSE '06, Luxemburg, 2006, S. 825-830.

[6] Meyer-Spasche, G.; Störing, M.: *Dienstbar – Rechtliche Risiken bei SOA-Projekten managen*, in: iX, Mai 2009, S. 134-135.

[7] Pulvermüller, E.; Speck, A.; Coplien, J. O.: *A Version Model for Aspect Dependency Management*. In: Proceedings of 3rd International Symposium of Generative and Component-based Software Engineering (GCSE 2001), LNCS 1241, Erfurt, Springer 2001, S. 70-79.

[8] McMillan: *Symbolic Model Checking*. Kluwer, Academic Publishers, 1993.

[9] Ringelstein, Chr.: *Protokollierung in service-orientierten Architekturen*. In: Datenschutz und Datensicherheit (DuD), Vieweg-Verlag, 10/2007, S. 736-739.

[10] Wolf, G.; Pfitzmann, A.: *Charakteristika von Schutzzielen und Konsequenzen für Benutzungsschnittstellen*. In: Informatik Spektrum, 06/2000, S. 173-191.