Grafische Validierungsregeln am Beispiel von EPKs

von Sven Feja, Daniel Fötsch und Sebastian Stein

Die Verwertungsrechte für diesen Beitrag liegen bei der Gesellschaft für Informatik (GI) e.V. Eine erneute Veröffentlichung ist nur mit Zustimmung der Autoren und unter Angabe des Verwertungsrechteinhabers möglich.

Bibtex Eintrag

```
@INProceedings{feja2008mseim,
   author="Feja, Sven and Fötsch, Daniel and Stein, Sebastian",
   title="Grafische Validierungsregeln am Beispiel von EPKs",
   year="2008",
   month="February",
   booktitle="Workshop Modellgetriebene Softwarearchitektur -- Evolution,
   Integration und Migration (MSEIM)",
   address="München, Germany",
   series="Lecture Notes in Informatics (LNI)"
}
```

Grafische Validierungsregeln am Beispiel von EPKs

Sven Feja[1], Daniel Fötsch[1], Sebastian Stein[2]

[1] Institut für Informatik, Christian-Albrechts-Universität zu Kiel
24098 Kiel
svfe, daf@informatik.uni-kiel.de

[2] IDS Scheer AG
Altenkesseler Str. 17, 66115 Saarbrücken
sebastian.stein@ids-scheer.com

Abstract: Als Grundlage für die modellgetriebene Softwareentwicklung werden korrekte und qualitativ hochwertige Modelle bereits auf abstrakter fachlicher Ebene vorausgesetzt. Um die Qualität der Modelle zu sichern, existieren verschiedene Ansätze wie Modellprüfen (*Model Checking*). Modellprüfen ist eine formale Methode, mit der Modelle gegen eine Spezifikation validiert werden. Das Modell des Modellprüfers ist in der Regel ein mit Eigenschaften beschrifteter Zustandsautomat. Die Spezifikation der Prüfregeln wird textuell mit temporaler Logik auf Ebene des Zustandsautomaten definiert. Diese formale Definition ist für Fachmodellierer wenig geeignet.

In diesem Beitrag wird deshalb eine grafische Notation für Validierungsregeln am Beispiel der Geschäftsprozessnotation Ereignisgesteuerte Prozesskette (EPK) entwickelt. Dies ermöglicht dem Fachmodellierer, Anforderungen an die EPK zu formulieren. Die Anwendung der grafischen Notation wird anhand eines Fallbeispiels aus der E-Government-Domäne kurz demonstriert.

1 Einleitung

Die modellgetriebene Softwareentwicklung im Rahmen einer Model Driven Architecture [MM03] ist ein viel versprechender Ansatz, um die dringend benötigte Effizienzsteigerung in der Softwareentwicklung zu erreichen. Grundlage bilden dabei Modelle, die über Transformationen und Anreicherungen mit weiteren Details schrittweise bis hin zur ausführbaren Software verfeinert werden. Im Unternehmenskontext kommt häufig Standardssoftware zum Einsatz, die für den speziellen Einsatz angepasst werden muss. Anstatt die Software grundlegend neu zu entwerfen, werden bestehende Komponenten miteinander kombiniert und lediglich fehlende Komponenten neu entwickelt.

Aktuell ist zu beobachten, dass bestehende Komponenten im Rahmen einer serviceorientierten Architektur (SOA) [MSJL06, z. B.] als Web Services gekapselt werden und die Verknüpfung der einzelnen Komponenten über eine Orchestrierungssprache wie die Business Process Execution Language (BPEL[1]) erfolgt. Verschiedene Autoren [KTS05, DSS08]

[1] http://www-128.ibm.com/developerworks/library/ws-bpel/

zeigen, dass es sinnvoll ist die Anforderungen zunächst auf Ebene der Geschäftsprozesse durch Fachmodellierer zu erfassen und aus dieser fachlichen Prozessbeschreibung mittels einer wie in [SI07] vorgestellten Transformation die Web Service Orchestrierung in BPEL automatisch abzuleiten.

Voraussetzung für eine solche modellgetriebene Softwareentwicklung sind korrekte und qualitativ hochwertige fachliche Modelle. Zur Prüfung müssen sowohl syntaktische als auch semantische Prüfregeln formuliert werden. Während syntaktische Prüfregeln und deren Validierung von vielen Modellierungswerkzeugen bereits unterstützt werden, wird in diesem Beitrag gezeigt, wie semantische Prüfregeln auf der Fachebene definiert werden können und wie eine Validierung der Geschäftsprozessmodelle anhand dieser Prüfregeln möglich ist. Dabei ist es essentiell, dass die Definition der fachlichen Prüfregeln auf für Fachmodellierer verständliche Weise erfolgt. Bisherige Verfahren haben diese Forderung weitestgehend ignoriert. Bei unserem Ansatz nimmt diese Forderung eine zentrale Stellung ein. Unsere Lösung wurde prototypisch im ARIS SOA Architect umgesetzt und anhand eines Geschäftsprozesses aus der E-Government-Domäne [LRK06] erprobt.

2 Hintergrund

Für die Umsetzung unseres Validierungsansatzes wird die Modellprüfung verwendet. Die Modellprüfungstechnik hat ihre Ursprünge in der Hardware-Prüfung. In der aktuellen Forschung finden Modellprüfer immer stärker auch bei der Validierung von Software ihren Einsatz. Deutlich wird dies bspw. in der steigenden Anzahl von Beiträgen zum Thema Software-Modellprüfung in wissenschaftlichen Veranstaltungen wie der ECOOP Workshop Serie zum Thema Software-Modellierung und Validierung [PSD+01, SPC+02, VDSSP+03].

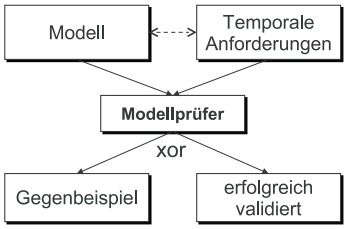

Abbildung 1: Grundprinzip des Modellprüfens.

Durch Modellprüfer können Abläufe modelliert und gegenüber Spezifikationen verglichen werden, die in temporaler Logik formuliert sind. Es wird die Erfüllbarkeit bestimmter temporaler Aussagen durch das vorliegende Modell bewiesen oder durch ein Gegenbeispiel widerlegt (siehe Abbildung 1). Damit lässt sich eine höhere Genauigkeit in Bezug auf die jeweilige Aussage erzielen als gegenüber einem manuellen Test. Trotzdem wird das modellierte System an sich nicht bewiesen, da nicht alle Anforderungen vollständig überprüft werden, bzw. es kann auch eine Prüfung über eine unvollständige Menge von Systemanforderungen erfolgen.

Überprüfbare Prozessmodelle oder Workflows sind bspw. UML Modelle oder Ereignisgesteuerte Prozessketten [KNS92]. Für UML existiert mit der Object Constraints Language (OCL) die Möglichkeit statische Abhängigkeiten in UML Diagrammen zu überprüfen. Dynamische Abhängikeiten lassen sich mit OCL nicht direkt prüfen. Dies wird erst durch die in [FM02], [DKR00] und [RM99] vorgestellten Erweiterungen ermöglicht. Für Geschäftsprozessmodelle sind bisher nur syntaktische Prüfungen durch die verwendeten Modellierungswerkzeuge möglich. Daher soll im Folgenden ein Verfahren der semantischen Überprüfung von Geschäftsprozessmodellen am Beispiel der EPK vorgestellt werden. Dennoch haben die folgenden Aussagen auch für andere Prozessmodelle wie UML ihre Gültigkeit.

Um die Modellprüfung für Geschäftsprozessmodelle nutzen zu können, müssen diese in das Modell eines Modellprüfers transformiert werden. Das Inputmodell ist typischerweise ein Zustandsautomat, dem je nach Modellprüfer ein linear diskretes (z. B. SPIN [Hol97]) oder verzweigt diskretes Zeitmodell (z. B. SMV [McM93]) zugrunde liegt. In diesem Beitrag wird das letztere Zeitmodell betrachtet, da es Aussagen über mehrere Pfade gleichzeitig ermöglicht. Eine Spezifikationssprache mit der sich temporale Aussage über verzweigte diskrete Zeitmodelle treffen lassen, ist *Computational Tree Logic* (CTL).

CTL ist eine temporale Logik, die von Clarke und Emerson entwickelt wurde [CE81]. CTL erweitert die Boolesche Logik, um verschiedene temporale Operatoren, die zeitliche Aussagen über Zustände erlauben. Dazu bietet CTL die Operatoren X (next), F (future) und G (globally), die spezifizieren, dass etwas im nächsten/irgendeinem/allen folgenden Zuständen gilt. Mit dem binäre Operator U kann ausgedrückt werden, dass etwas solange gilt, bis etwas anderes gilt. Diese temporalen Operatoren werden immer paarweise mit den Pfadquantoren A (auf allen Pfaden) und E (für mindestens ein Pfad) verwendet.

3 Grafische Modellierung von Prüfregeln

Zur Validierung von Geschäftsprozessen durch Modellprüfen bedarf es Validerungsregeln in einer geeigneten grafischen Notation auf der Abstraktionsebene des Geschäftsprozesses, wie z. B. die Ereignisgesteuerte Prozessketten. Die bisher notwendige textuelle Definition der temporalen Logik auf der Ebene des Modellprüfer-Modells ist für Fachmodellierer unbrauchbar, da es zum einen erheblichen Einarbeitungsaufwand erfordert und zum anderen die Kenntnis über das vom EPK überführte Modell des Modellprüfers verlangt. Daher muss eine grafische Notation an die im Geschäftsprozessmodell verwendete Notation angelehnt sein, um temporale Aussagen über die dort verwendeten Elemente zu möglichen. Die grafische Notation der von uns verwendeten temporalen Logik CTL, wird als *Graphical Computation Tree Logic* (kurz G-CTL) bezeichnet. Die Elemente der Notation sind in Abbildung 2 dargestellt.

Die G-CTL-Operatoren bestehen neben den Booleschen Operatoren aus zwei Komponenten, zum einen aus dem temporalen Operator und zum anderen aus einem Platzhalter. Die temporalen Operatoren entsprechen den in Abschnitt 2 vorgestellten. Die Platzhalter dienen zur Anlehnung an das Geschäftsprozessmodell. Dazu können die Felder a und b in

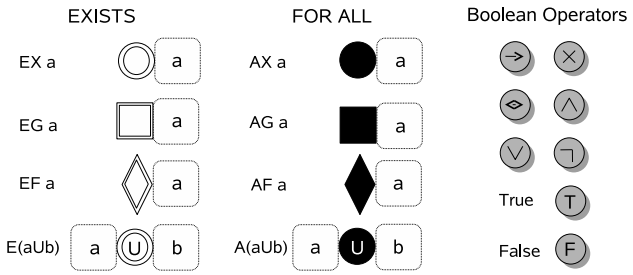

Abbildung 2: Die G-CTL-Operatoren.

Abbildung 2 mit den Elementen der EPK belegt werden. Der Name dieser Notation lautet daher EPK-G-CTL. Die Erstellung von grafischen Validierungsregeln mit den in Abbildung 2 vorgestellten Elementen wird im nächsten Abschnitt demonstriert.

4 Vorgehen und Anwendung

In Abbildung 3 ist das Gesamtvorgehen als vereinfachte EPK in fünf Schritten dargestellt. Neben den Einzelschritten ist weiterhin angegeben, in welchem Werkzeug im Rahmen unserer prototypischen Umsetzung jeder Schritt bearbeitet wird.

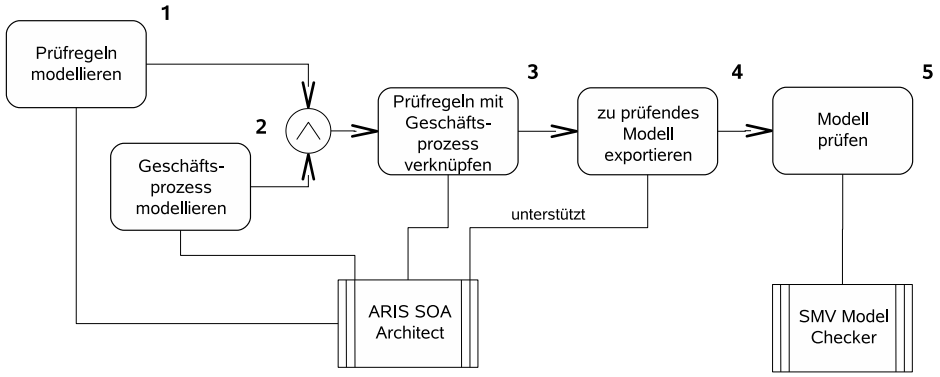

Abbildung 3: Übersicht des Gesamtvorgehens.

Schritt 1 und 2 des Vorgehens können dabei gleichzeitig ausgeführt werden, da sie zunächst voneinander unabhängig sind. In Schritt 1 erstellt der Fachmodellierer die Prüfregeln. Idealerweise handelt es sich dabei um allgemeingültige Regeln, die für alle Geschäftsprozesse des Unternehmens gelten, etwa dass neue Verträge immer von zwei unabhängigen Mitarbeitern autorisiert werden müssen. Parallel erfolgt in Schritt 2 die Modellierung der Geschäftsprozesse. In Schritt 3 wird angegeben, welche Prüfregel für welchen

Geschäftsprozess gilt. Dazu werden die grafischen Objekte, die die Regeln repräsentieren, in das Geschäftsprozessmodell eingebettet. In Schritt 4 wird das Modell und die Prüfregeln exportiert und mit Hilfe eines Transformators [FP07] in das für den Modellprüfer notwendige Format automatisch überführt. Im abschließenden Schritt 5 findet die Validierung statt. Im Folgenden wird das Vorgehen anhand eines Beispiels aus dem E-Government demonstriert.

Als fachlicher Hintergrund für die folgende Regel dient die *Einfache Melderegisterauskunft* (eMRA) [LRK06]. Diese ermöglicht es einem Benutzer über ein Portal des Einwohnermeldeamtes die Daten eines Bürgers zu überprüfen. Für die eMRA gilt in jedem Bundesland das Landesmeldegesetz (LMG), in welchem bspw. eine Protokollierung von versendeten Nachrichten an den anfragenden Benutzer festgelegt ist. Im konkreten Fall muss nach einer erfolgreichen Auskunftsberechtigungsprüfung das Absenden einer Antwort im Verlauf des modellierten Geschäftsprozesses protokolliert werden. In Form der temporalen Logik CTL würde dies bedeuten:

$$AG(Auskunftsberechtigung\_liegt\_vor \rightarrow AF(Protokollierung)). \quad (1)$$

Wobei `Auskunftsberechtigung_liegt_vor` dem Ereignis *Auskunftberechtigung liegt vor* und `Protokollierung` der Funktion *Protokollierung* in dem E-Government Geschäftsprozess entsprechen.

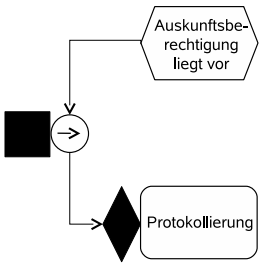

Abbildung 4: Beispiel einer EPK-G-CTL-Regel.

Die Modellierung mit EPK-G-CTL ist in Abbildung 4 dargestellt. Die Regel besteht aus einer Wenn-Dann-Beziehung, die man in der vorgegebenen Pfeilrichtung liest. Die den Implikationspfeil und der Funktion *Protokollierung* voran gestellten temporalen Operatoren gelten jeweils für das entsprechende Element. Das heißt konkret, immer wenn das Ereignis *Auskunftberechtigung liegt vor* im Prozess eintritt (`AG`), dann muss in der Zukunft auf allen Pfaden mindestens einmal (`AF`) die Funktion *Protokollierung* ausgeführt werden. Unter Anwendung dieser Regel kann nun ein vorhandenes Geschäftsprozessmodell auf die Einhaltung dieser Anforderung überprüft werden.

5 Zusammenfassung

Die syntaktische und semantische Validierung von fachlichen Anforderungsmodellen ist eine Grundvoraussetzung für die modellgetriebene Softwareentwicklung. Es existieren eine Reihe von Ansätzen auf Basis von Modellprüfen, die aber wenig praktikabel sind, da im Gegensatz zu den grafischen Fachmodellen die Formulierung der Prüfregeln lediglich textuell in Temporaler Logik erfolgt. Dieser Beitrag zeigt, wie Fachmodellierer sowohl Geschäftsprozessmodelle als auch die zugehörigen Prüfregeln modellieren können. Für die Darstellung der Geschäftsprozessmodelle werden Ereignisgesteuerte Prozessketten (EPK) verwendet und für die Darstellung der Prüfregeln wurde eine an die EPK angelehnte Notation entwickelt.

Eine erste Evaluierung unseres Vorgehens fand am Beispiel der *Einfachen Melderegisterauskunft* aus dem E-Government statt. Weiterhin existiert eine prototypische Werkzeugunterstützung auf Basis des ARIS SOA Architect. Unsere momentane Arbeit konzentriert sich auf der Evaluierung mit weiteren Beispielszenarien und einer Überarbeitung der Notation. Hier ist speziell von Interesse, wie die Modellierung der Prüfregeln sauber in die Gesamtmethodik zur Geschäftsprozessmodellierung integriert werden kann.

Literatur

[CE81] E. M. Clarke und E. A. Emerson. Synthesis of synchronization skeletons for branching time temporal logic. In D. Kozen, Hrsg., *Proceedings of Logic of Programs Workshop*, LNCS 131, Seiten 52 – 71. Springer Verlag, 1981.

[DKR00] Dino Distefano, Joost-Pieter Katoen und Arend Rensink. On a Temporal Logic for Object-Based Systems. In Scott F. Smith und Carolyn L. Talcott, Hrsg., *Formal Methods for Open Object-Based Distributed Systems IV - Proc. FMOODS'2000, September, 2000, Stanford, California, USA*. Kluwer Academic Publishers, 2000.

[DSS08] J. Drawehn, S. Stein und P. Schneider. Ein Verfahren zur Orchestrierung von IT Services. In *Modellgetriebene Softwarearchitektur – Evolution, Integration und Migration*, München, Februar 2008.

[FM02] Stephan Flake und Wolfgang Mueller. An OCL Extension for Real-Time Constraints. In *Object Modeling with the OCL*, Seiten 150–171, 2002.

[FP07] D. Fötsch und E. Pulvermüller. Constructing higher-level Transformation Languages based on XML. In H. Fujita und D. Pisanelli, Hrsg., *New Trends in Software Methodologies, Tools and Techniques*, Jgg. 161 of *Frontiers in Artificial Intelligence and Applications*, Seiten 269–284, Rome, Italy, November 2007. IOS Press.

[Hol97] G. J. Holzmann. The Model Checker Spin. *IEEE Transactions on Software Engineering*, 23(5):279 – 295, May 1997.

[KNS92] G. Keller, M. Nüttgens und A.-W. Scheer. Semantische Prozeßmodellierung auf der Grundlage „Ereignisgesteuerter Prozeßketten (EPK)". Arbeitsbericht Heft 89, Institut für Wirtschaftsinformatik Universität Saarbrücken, 1992.

[KTS05] S. Kühne, M. Thränert und A. Speck. Towards a methodology for orchestration and validation of cooperative e-business components. In M. J. Rutherford, Hrsg., *7th GPCE Young Researcher Workshop*, Seiten 29–34, 2005.

[LRK06] J. Lehmann, W. Rotzoll und S. Kühne. Analyse der E-Government-Domäne Meldewesen am Beispiel des Dienstes „einfache Melderegisterauskunft“. In K.-P. Fähnrich, S. Kühne, A. Speck und J. Wagner, Hrsg., *Integration betrieblicher Informationssysteme: Problemanalysen und Lösungsansätze des Model-Driven Integration Engineering*, Jgg. IV of *Leipziger Beiträge zur Informatik*, Seiten 20–30. Eigenverlag Leipziger Informatik-Verbund (LIV), Leipzig, September 2006.

[McM93] K. McMillan. *Symbolic Model Checking*. Kluwer Academic Publishers, 1993.

[MM03] J. Miller und J. Mukerji. MDA Guide. Bericht omg/2003-06-01, Object Management Group (OMG), Juni 2003. Version 1.0.1.

[MSJL06] J. McGovern, O. Sims, A. Jain und M. Little. *Enterprise Service Oriented Architectures*. Springer, Dordrecht, The Netherlands, 2006.

[PSD+01] E. Pulvermüller, A. Speck, M. D'Hondt, W.D. De Meuter und J.O. Coplien. Proccedings of Workshop on Feature Interaction in Composed Systems, ECOOP 2001, Budapest, Hungary. Bericht No. 2001-14, Universität Karlsruhe, 2001.

[RM99] Sita Ramakrishnan und John McGregor. Extending OCL to Support Temporal Operators. In *Proceedings of the 21st International Conference on Software Engineering (ICSE99) Workshop on Testing Distributed Component-Based Systems, LA, May 16 - 22, 1999*, 1999.

[SI07] S. Stein und K. Ivanov. EPK nach BPEL Transformation als Voraussetzung für praktische Umsetzung einer SOA. In Wolf-Gideon Bleek, Jörg Raasch und Heinz Züllighoven, Hrsg., *Software Engineering 2007*, Jgg. 105 of *Lecture Notes in Informatics (LNI)*, Seiten 75–80, Hamburg, Germany, März 2007. Gesellschaft für Informatik (GI).

[SPC+02] A. Speck, E. Pulvermüller, M. Clauß, R. Van Der Straeten und R. Reussner. Proccedings of Workshop on Model-based Software Reuse ECOOP 2002, Malaga, Spain. Bericht No. 2002-4, Universität Karlsruhe, 2002.

[VDSSP+03] R. Van Der Straeten, A. Speck, E. Pulvermüller, M. Clauß und A. Pleuß. Proccedings of Workshop on Correctness of Model-based Software Composition (CMC) ECOOP 2003, Darmstadt. Bericht No. 2003-13, Universität Karlsruhe, 2003.