Ereignisorientiertes Testen Web-basierter Systeme – Verfeinerung des holistischen Ansatzes und eine Fallstudie

Fevzi Belli[+], Michael Linschulte[+], Ina Schieferdecker[*]

[+)]: Universität Paderborn, Fakultät für Elektrotechnik, Informatik und Mathematik,
Institut für Elektrotechnik und Informationstechnik, Fachgebiet „Angewandte Datentechnik“

[*)]: Technische Universität Berlin, Fakultät IV - Elektrotechnik und Informatik,
Lehrstuhl „Entwurf und Testen von Telekommunikationssystemen“

Kurzfassung

Auf der Grundlage eines ereignisbasierten Ansatzes zum Test von Web-basierten Systemen verfeinert dieser Beitrag die Beschreibung von Eingabedaten und modelliert charakteristische Merkmale von Eingaben anhand von Eingabeformularen. Eine Fallstudie validiert den vorgestellten Testansatz unter Berücksichtigung von ungültigen Eingabedaten.

1 Einleitung und verwandte Arbeiten

Im vergangenen Jahrzehnt ist die Anzahl an Web-basierten Systemen (auch: Web-Applikationen) enorm angestiegen. Dieses ist nicht zuletzt eine Folge des stark wachsenden E-Commerce, der durch groß angelegte Werbekampagnen wie zum Beispiel von IBM stark vorangetrieben wurde. Gerade weil jene Systeme entscheidend für diese Geschäftsprozesse sind, ist damit auch die Sorge um deren Qualität und Zuverlässigkeit verbunden.
Die Prüfung von Web-basierten Systemen stellt jedoch eine komplexe Aufgabe dar, zu dessen Hilfe Modelle und Techniken eingesetzt werden, die sowohl die Konzeption, Navigation als auch das Design berücksichtigen. Die meisten dieser Modelle sind graphisch; beispielsweise sei die Erweiterung von UML erwähnt, um Web-Applikationen darzustellen [1], [2]. [3] entwickelt eine Methode zur Testfallgenerierung auf Basis eines objektorientierten Modells. Hierbei wird das zu testende System aus drei verschiedenen Perspektiven betrachtet: der Objekt-, Struktur- und Verhaltensperspektive, die in unterschiedlichen Diagrammen modelliert werden. Darauf aufbauend werden Testbäume erstellt und anschließend daraus Testfälle generiert. Der Ansatz in [4] befasst sich mit dem Black-Box-Test von Web-Applikationen. Er berücksichtigt das zustandsabhängige Verhalten von Web-Applikationen und schlägt eine darauf gezielte Lösung des Zustandsexplosionsproblems vor.
In dieser Arbeit wird ein ereignisbasierter Ansatz [5] vorgestellt, der ähnlich zu [4] eine Strukturierung der Modelle ermöglicht und das System als Black-Box betrachtet. In [10] wurde dieser Ansatz um die Berücksichtigung von Daten durch Entscheidungstabellen (*ET*) erweitert, da die Eingabe von Daten ein häufig anzutreffendes, charakteristisches Merkmal von Web-basierten Systemen ist (s. auch [6]). Durch diese Eingaben können sogar dynamische Strukturänderungen innerhalb der Applikation hervorgerufen werden. Daher wird in der vorliegenden Arbeit der Aufbau der Entscheidungstabellen präzisiert, indem

- der Wertebereich der Eingabedaten näher definiert und
- die Abhängigkeit der Eingabedaten verdeutlicht

wird. Das Ziel dabei ist, das vorliegende Modell an die Applikation besser anzupassen und den Testaufwand zu reduzieren (Abschn. 2).
Weiterhin wird vermutet, dass viele Fehler durch Testfälle mit ungültigen Eingabedaten erkannt werden, weil zumeist nicht spezifiziert wird, wie der Prüfling in solchen Situationen reagieren soll. Anhand einer Fallstudie im Abschnitt 3 wird diese Vermutung validiert.

2 Idee und Lösungsansatz

Der Ansatz von Belli, Budnik und White [7] befasst sich mit dem Black-Box-Test von interaktiven Systemen. Mittels so genannter Ereignissequenzgraphen (kurz: ESG) (s. **Bild 1**) wird das System modelliert, anschließend werden anhand der Graphen Testfälle konstruiert, die am System ausgeführt werden.

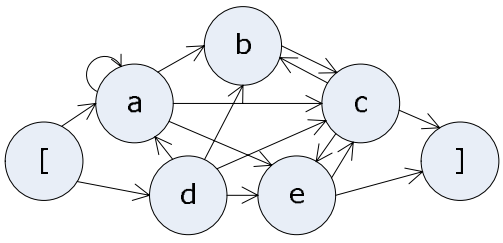

Bild 1 Beispiel eines ESG

Ein *Ereignis* stellt ein extern beobachtbares Phänomen dar, wie z.B. Nutzereingaben, Systemantworten etc. und wird als Knoten im ESG repräsentiert. Die Knoten werden durch gerichtete Kanten miteinander verbunden, um die mögliche Ausführungsreihenfolge darzustellen.
Weil jeder Knoten genau ein Ereignis des Systems darstellt, muss bei der Ausführung gleicher Interaktionen darauf geachtet werden, ob sich der Kontext geändert hat. Diese Eigenschaft eines interaktiven Systems, aus zwei unterschiedlichen Kontexten heraus die gleichen Ereignisse auszulösen, muss in einem ESG gesondert unterschieden werden. D. h. in einem solchen Fall darf nicht das gleiche Ereignis bzw. Knoten im ESG benutzt werden. Nur wenn Ereignisse in dem gleichen Kontext ausgelöst werden, dürfen diese durch ein und denselben Knoten dargestellt werden. Treten sie in unterschiedlichen Kontexten auf, sind die Ereignisse bzw. Knoten zu indizieren. In **Bild 2** ist ein solcher ESG skizziert, in dem zwei Knoten die gleichen Ereignisse bezeichnen aber durch einen Index voneinander unterschieden werden.

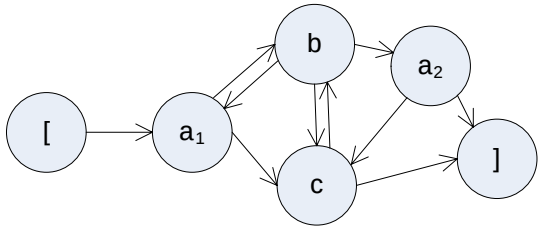

Bild 2 kontextsensitiver ESG

In diesem Fall steht das Ereignis a in zwei unterschiedlichen Kontexten. So kann beispielsweise das Ereignis a_2 nur dann ausgelöst werden, wenn zuvor Ereignis b ausgelöst wurde. Dies bedeutet, dass der Systemzustand über die gleiche Interaktion erfolgte, jedoch nicht mehr die alte Umgebung wieder zu finden war. Bei der Modellierung von Ereignissequenzgraphen muss streng auf den Kontext geachtet werden.

2.1 Erweiterung der ESG durch ET

Um nun die Dateneingabe im ESG zu modellieren, wurde das ESG-Modell [8] bei der Beschreibung der Testeingabedaten um Entscheidungstabellen (ET) erweitert (vgl. [6]). Der Knoten, der die ET repräsentiert, wird zusätzlich durch einen Doppelkreis im ESG kenntlich gemacht und wird *Dateneingabeknoten* genannt. Das Ereignis wäre somit die Eingabe mehrerer Daten. In Anlehnung an [9] gelten folgende Voraussetzungen für die Verwendung von ET:

- Die Ausführung des Nachfolgeereignisses darf nicht von früheren Ein- oder Ausgaben abhängen
- Die Ausführung des Nachfolgeereignisses darf nicht von der Eingabereihenfolge der Daten abhängen

Bild 3 in Verbindung mit **Tabelle 1** greift auf das Beispiel des Einloggens an einem System durch Benutzername und Passwort aus [6] zurück. Die Abbildung zeigt exemplarisch den ESG mit zugehöriger ET für den Knoten Login zur Generierung möglicher Testeingabesequenzen.

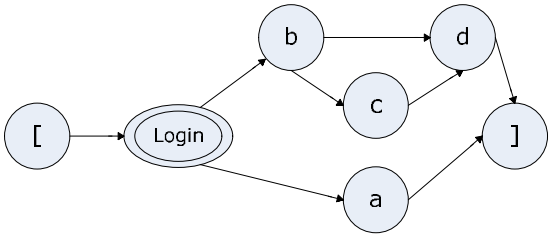

Bild 3 Login an einem System

Tabelle 1 Entscheidungstabelle

Login	R1	R2	R3	R4	R5	R6
Benutzername $\in \Sigma^*$	T	T	T	T	F	F
Passwort $\in \Sigma^*$	T	T	T	F	T	F
Benutzername $\wedge$ Passwort $\in$ Administrator-Login	T	F	F	-	-	-
Benutzername $\wedge$ Passwort $\in$ Benutzer-Login	F	T	F	-	-	-
a	X					
b		X				
Fehler			X	X	X	X

Im oberen Teil der ET werden die Eingabedaten mit Hilfe der Mengenlehre als *Bedingungen* modelliert, der untere Teil definiert mögliche Nachfolgeereignisse anhand der Erfüllung der unterschiedlichen Bedingungen. Eine Spalte in ET definiert eine *Regel*, in der die Erfüllung oder Nichterfüllung (true/false) der kombinierten Bedingungen und mögliche Nachfolgeereignisse festgelegt werden. Der obere Teil der Entscheidungstabelle wird nun verfeinert. Das Zeichen „Σ“ (=Sigma) der ET steht für das Eingabealphabet und wird definiert durch:

$$\Sigma=\{A,..,Z,a,..,z,0..9,@, '.', ',', '\}$$

„Σ^*“ bedeutet eine beliebig lange Kombination der in Σ enthaltenen Zeichen, also eine Zeichenfolge. „Administrator-Login“ und „Benutzer-Login“ stellen Mengen zulässiger Login-Daten dar. Die Bedingungen der ET bestehen nun aus zwei Teilen: Der eine Teil (nicht kursiv) bestimmt den Wertebereich der Daten, der andere Teil (kursiv) definiert eventuelle Abhängigkeiten unter Daten und/oder nötige Ausprägungen von Daten, um das Nachfolgeereignis zu bestimmen. Dieses geschieht unter zu Hilfenahme der Booleschen Algebra („$\wedge$“: „und“, „$\vee$“: „oder“, „$\neg$“ für „nicht“ (vgl. [9])). Für das Beispiel in Tabelle 1 heißt das: Sind Benutzername und Passwort jeweils Zei-

chenfolgen und gehören zu einem Administrator-Login oder Benutzer-Login, so kann das gültige Nachfolgeereignis bestimmt werden. Ist der Benutzername und/oder das Passwort keine Zeichenfolge (sie werden z.B. nicht ausgefüllt), so können sie nicht zu einem gültigen Login gehören und ein Login-Versuch müsste fehlschlagen. Für diesen Fall werden die letzten beiden Bedingungen der ET mit einem „Don't Care"-Term in Form eines „-" gekennzeichnet.

2.2 Modellierung charakteristischer Elemente

In Formularen gibt es unterschiedliche Möglichkeiten, Daten zu erfassen. Neben einfachen *Eingabefeldern*, in die direkt per Tastatur geschrieben werden kann, existieren noch *Auswahllisten*, *Radio-Buttons* und *Check-Boxen*. Beispiele für diese Elemente liefert **Bild 4**. Vor- und Nachname werden mit je einem Eingabefeld erfasst, das Geschlecht wählt man mittels der Radio-Buttons, den Beruf per Auswahlliste und die Sprachen über Check-Boxen.

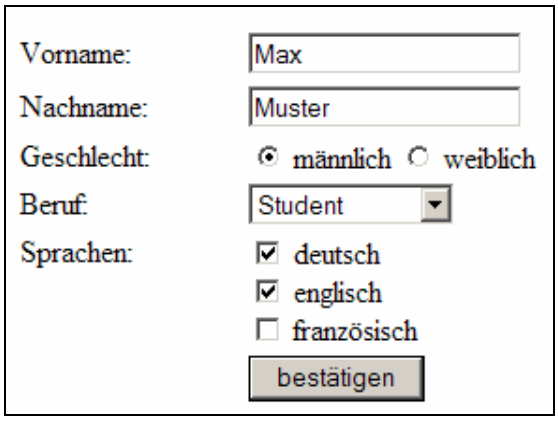

Bild 4 Formular-Elemente

Tabelle 2 ET zu Bild 4

Dateneingabe	R1
Vorname $\in \Sigma^*$	T
Nachname $\in \Sigma^*$	T
Geschlecht $\in$ {männlich, weiblich}	-
Beruf $\in$ {Schüler, Student, Angestellter, Selbständiger}	-
Sprachen $\subseteq$ {deutsch, englisch, französisch} $\vee\varnothing$	-
Beruf=Schüler	F
Beruf=Student	T
…	
„bestätigen" klicken	X

Die Erfassung von Eingabefeldern in ET wurde bereits dargestellt. Mittels des Eingabealphabets Σ kann, wie in Tabelle 1 geschehen, eine Bestimmung des Wertebereichs von Eingabedaten vorgenommen werden. Anders sieht dies bei Auswahllisten aus. Auswahllisten ermöglichen es dem Benutzer, aus einer vorgegebenen Liste von Werten genau einen Wert zu selektieren. Deswegen kann die Eingabe nur genau ein Element („$\in$") der vorgegebenen Menge sein. Genau so sieht dies bei Radio-Buttons aus. Aus einer vorgegebenen Liste von Werten wird ebenfalls genau ein Wert ausgewählt. Eine Mehrfachauswahl ist nicht möglich. Eine Mehrfachauswahl bei vorgegebenen Werten wird durch Check-Boxen realisiert. Hier ist die Auswahl einer Teilmenge („$\subseteq$") aus einer vorgegebenen Menge möglich. Die Erfassung von Auswahllisten, Radio-Buttons und Check-Boxen in einer ET zeigt **Tabelle 2**. Weil Radio-Buttons, Auswahllisten und Check-Boxen bereits gültige Werte in der Eingabemaske vorgeben, aus denen der Benutzer nur auswählen aber nichts verändern oder hinzufügen kann, werden sie in der ET mit einem „Don't Care"-Term belegt, um die Testfallmenge zu begrenzen. Ist die Ausprägung einer Select-Box, Auswahlliste oder Check-Box relevant für das Nachfolge-Ereignis, so können in der ET weitere Bedingungen hinzugefügt werden, z.B. „Beruf=Schüler" oder „Beruf=Student", die dann mit Wahr und Falsch entsprechend der ET-Notation behandelt werden können. Zusätzlich kann es sein, das ein bestimmtes Datum nicht ausgefüllt werden muss und/oder leer bleiben kann. Solche Fälle können in der ET durch das Zeichen für die leere Menge „$\varnothing$" Berücksichtigung finden.

Eine der beiden Voraussetzungen für ET ist, das die Ausführung eines Nachfolgeereignisses nicht von der Eingabereihenfolge der Daten abhängen darf. Es kann jedoch vorkommen, dass sich das Eingabeformular bei der Auswahl eines Wertes (z.B. einer Select-Box) verändert oder neu aufbaut. In diesem Fall muss die Dateneingabe durch zwei (oder mehr) ET modelliert werden. Im ESG werden die ET entsprechend durch zwei (oder mehr) Dateneingabeknoten beschrieben, die durch eine gerichtete Kante miteinander verbunden sind.

2.3 Testdurchführung

Die Testdurchführung lässt sich in drei Schritte gliedern: Der erste Schritt des Testvorgangs stellt die hierarchische Zerlegung des Systems in logisch zusammenhängende Teilsysteme dar. In der Praxis hat sich für die Strukturierung von Web-Applikationen eine bestimmte Vorgehensweise als nützlich erwiesen. So liefert die Navigation bereits eine grobe Strukturierung der Applikation, denn durch die Navigation wurde bereits eine funktionale Einteilung des Systems durch den System-Designer vorgenommen. Jeder Menüpunkt kann daher als eigenes Teilsystem betrachtet werden. Sofern sich hinter einem Menüpunkt mehrere miteinander verbundene Webseiten verbergen, so kann jede einzelne Seite wiederum als Teilsystem verstanden werden. Ist eine Seite sehr umfangreich in ihren Aktionsmöglichkeiten und hat zusätzlich verschiedene Bereiche mit Listen und Formularen, so sollten diese Seiten nochmals unter-

teilt werden. Ziel der Strukturierung ist es, eine möglichst feine Granularität des Systems für die folgende Modellierung im ESG zu erhalten.
Die Aufstellung der ESG stellt damit den zweiten Schritt der Testdurchführung dar. Ausgehend vom Start des Systems werden nun alle ESG anhand der gebildeten, hierarchischen Struktur aufgestellt und verfeinert. Sind alle ESG aufgestellt, werden im dritten und letzten Schritt die Testfälle generiert und am System ausgeführt. Die Testfallgenerierung basiert auf dem Ansatz von [8], welcher die minimale Überdeckung von Ereignissen der ESG durch vollständige Ereignissequenzen vorsieht. Hierbei werden die entsprechenden Ereignissequenzen generiert und anschließend die Knoten, die Entscheidungstabellen beinhalten, durch gültige und ungültige Eingabedaten ersetzt.

3 Fallstudie

Anhand einer Fallstudie wurde das im vorigen Kapitel beschriebene Verfahren evaluiert. Grundlage ist das Buchungs- und Verwaltungssystem ISELTA, das Hotelverwaltung und Zimmerbuchung über das Internet ermöglicht.
ISELTA wurde mittels der Ereignissequenzgraphen und den dazugehörigen Entscheidungstabellen modelliert. Im Anschluss daran wurden die Ereignissequenzen mit gültigen und ungültigen Eingabedaten aufgestellt und am System ausgeführt.
Um nach der Durchführung der Testfälle am System den Erfolg des erstellten Testkonzepts messen zu können, ist die Klassifizierung der bei der Testdurchführung gefundenen Fehler sehr wichtig. Zu diesem Zweck wurden die aufgetretenen Fehler nach vier Fehlerklassen kategorisiert. Die Fehlerklassen zeigt **Bild 5**.

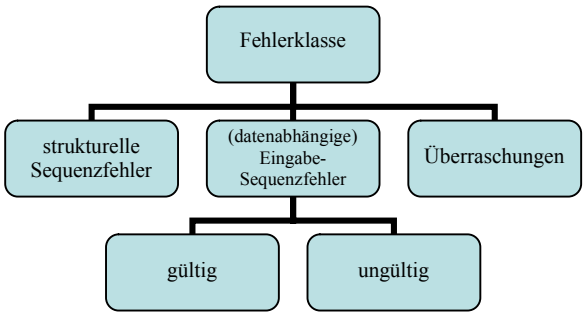

Bild 5 Fehlerklassen

Strukturelle Sequenzfehler stellen Fehler dar, die von Testfällen ohne Eingabedaten gefunden werden sollten und damit Fehler in der Struktur der Web-Applikation aufdecken. Testfälle mit Eingaben haben das Ziel, *Eingabe-Sequenzfehler* aufzudecken. Eingabe-Sequenzfehler werden unterschieden in *gültig* und *ungültig* analog zu Testfällen mit gültigen und ungültigen Eingabedaten. *Überraschungen* stellen Fehler dar, mit denen man nicht unbedingt gerechnet hat und die per Zufall durch den Tester gefunden wurden. Ein Beispiel könnte ein fehlerhaftes Design oder die fehlerhafte Darstellung von Daten sein. Das Testorakel ist in diesem Falle der Tester.
Für die Modellierung des Systems und die Testfallerstellung wurden insgesamt 15 Tage benötigt. 5 Tage hiervon wurden für die Testfallerstellung aus den Modellen verwendet. Die anschließende Eingabe in ein Testwerkzeug für die automatisierte Ausführung nahm nochmals 20 Tage in Anspruch. Nachdem die Testfälle erstellt und eingegeben wurden, ist ein zusammenhängender Testlauf gestartet worden, der 248 Minuten gedauert hat.
Die Ergebnisse der Fallstudie zeigt **Bild 6**. Eine Sequenz wurde als negativ gekennzeichnet, wenn sie das erwartete Ergebnis bzgl. der Ausführung bestätigte, andernfalls positiv. Insgesamt wurden durch den Testlauf 27 Fehler aufgedeckt.

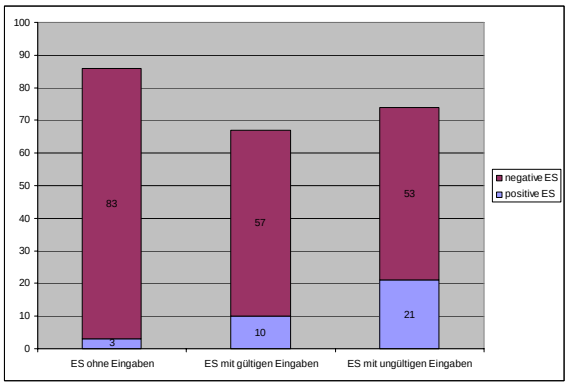

Bild 6 Graphische Übersicht über die Ergebnisse der Fallstudie

Für die Fallstudie wurden insgesamt 227 Testfälle erstellt, von denen 74 Testfälle ungültige Dateneingaben enthielten und damit das Systemverhalten bei fehlerhaften Eingaben abtesten sollten. 21 oder 28,4% von diesen 74 Testfällen waren positiv. Die 21 positiven Testfälle haben 24 der 27 Fehler aufgedeckt. Von den 67 Testfällen mit gültigen Eingaben waren 10 positiv. Diese 10 Testfälle haben 3 der 27 Fehler aufgedeckt. Die 86 Testfälle ohne Eingaben haben 3 Fehler durch 3 positive Testfälle aufgedeckt.

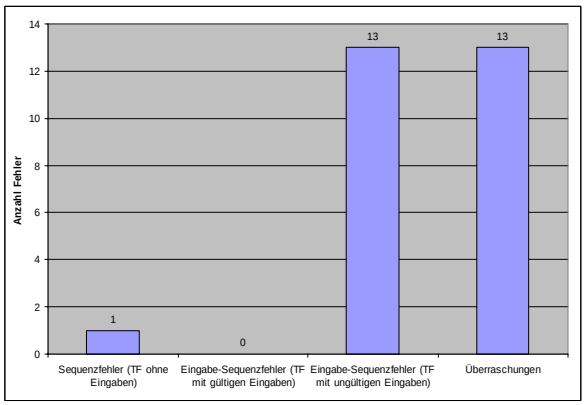

Bild 7 Fehlerkategorisierung

Werden die Fehler kategorisiert (s. **Bild 7**), so fällt auf, dass 13 der 27 Fehler als Überraschung klassifiziert werden. Diese Fehler wurden während der Ausführung der Testfälle durch die Beobachtung des Testers gefunden und nicht durch die Sequenzen selber aufgeworfen. Sie sind daher eingeschränkt zu betrachten. Weiterhin fällt auf, dass 13 der 14 restlichen Fehler Eingabe-Sequenzfehler waren und durch Testfälle mit ungültigen Eingaben entdeckt wurden.
Im Hinblick auf die Ergebnisse kommt eines sehr deutlich heraus: Die meisten der gefundenen Fehler wurden durch Testfälle mit ungültigen Eingabedaten aufgedeckt. Eine Quote von 0,18 Fehlern pro Testfall mit ungültigen Eingaben bestätigt die im Vorfeld aufgestellte Hypothese, dass viele Fehler durch diese Testfälle gefunden werden. Dieser Sachverhalt lässt sich sehr leicht begründen: Der Programmierer erhält oftmals nur eine Spezifikation darüber, was ein Programm oder eine Funktion zu leisten imstande sein soll. Was jedoch passiert, wenn eine Funktion mit falschen Parametern aufgerufen wird, ist oft nicht Teil dieser Spezifikation. In dieses Schema passt dementsprechend auch, dass durch die Testfälle mit gültigen Eingabedaten keine Eingabe-Sequenzfehler aufgeworfen werden konnten. Gültige Eingaben wurden also vom System korrekt akzeptiert.
Die erzielten Ergebnisse zeigen, wie immens wichtig die systematische Generierung von Testfällen mit ungültigen Eingabedaten für den Test des Systemverhaltens ist.

4 Zusammenfassung und Ausblick

Web-basierte Systeme zeichnen sich im Gegensatz zu anderen Systemen durch die beim Ausfüllen von Formularen typischerweise getätigten Eingaben seitens des Benutzers aus. Durch die hierbei gewählten Eingabedaten wird häufig eine dynamische Strukturänderung hervorgerufen. Web-Applikationen stellen daher neue Anforderungen an die Prüfung des funktionalen Verhaltens dar. Verbunden mit der Schnelllebigkeit von Web-Applikationen am Markt bleibt in der Praxis jedoch meist nur noch Zeit für den Entwurf eines Papier-basierten Prototypen. Diese können aber dazu genutzt werden, um die entsprechenden Modelle zum Test aufzustellen.
In diesem Beitrag wurde auf die Modellierung und den Test dieser Systeme anhand eines ereignisbasierten Ansatzes eingegangen, der um die Berücksichtigung von Eingabedaten durch ET erweitert wurde. Der Aufbau der ET wurde weiter präzisiert und die Modellierung charakteristischer Merkmale von Eingabeformularen beschrieben. In einer Fallstudie wurde anschließend gezeigt, dass gerade der Test ungültiger Eingabedaten enorm wichtig ist und nicht vernachlässigt werden sollte. Gerade hier wurden die meisten Fehler gefunden.
Laufende Arbeiten beschäftigen sich derzeit mit dem Negativ-Test Web-basierter Systeme, um neben der Eingabe falscher Daten zusätzlich auch den Test unerwarteter und damit unerwünschter Ereignisfolgen zu berücksichtigen.

5 Literatur

[1] Conallen, J.: Building Web Applications with UML, Addison-Wesley, 1999.

[2] Hennicker, R.; Koch, N.: Modeling the User Interface of Web Applications with UML. UML 2001. LNI, vol. 7, 2001; S.158-172.

[3] Kung, D.C.; Liu, C.-H.; Hsia, C.-T.: An Object-Oriented Web Test Model for Testing Web Applications. In Proc. of Asia-Pacific Conf. on Quality Software. IEEE Comp. Press, 2000; S. 111.

[4] Andrews, A.A.; Offutt, J.; Alexander, R.T.: Testing Web applications by modelling with FSMs. Software and System Modeling, vol. 4(3), 2005; S. 326-345.

[5] Belli, F.: Finite-State Testing and Analysis of Graphical User Interfaces. In Proc. of Int. Symp. on Softw. Reliability and Eng. IEEE Comp. Press, 2001; S. 34-43.

[6] Belli, F.; Budnik, C.J.; Linschulte, M.; Schieferdecker, I.: Testen Web-basierter Systeme mittels strukturierter, graphischer Modelle – Vergleich anhand einer Fallstudie. Model-based Testing 2006, LNI, Vol. 94, 2006; S. 266-273.

[7] Belli, F.; Budnik, C. J.; White, L.: Event-based Modeling, Analysis and Testing of User Interactions: Approach and Case Study. The Journal of Software Testing, Verification and Reliability. John Wiley & Sons, vol. 16(3), 2006; S. 3-32.

[8] Belli, F.; Budnik CJ.: Minimal Spanning Set for Coverage Testing of Interactive Systems. Proceedings of Int. Colloquium on Theoretical Aspects and Computing (Lecture Notes in Computer Science, vol.201). Springer Verlag, 2004; S. 220-234.

[9] Binder, R.V.: Testing Object Oriented Systems - Models, Patterns and Tools. Addison-Wesley Professional, 1999.

[10] Linschulte, M.: Testkonzeptionierung Web-basierter Systeme anhand einer kommerziellen Anwendung, Diplomarbeit, Universität Paderborn, 2006.