E-Learning Aufgaben für Ereignisgesteuerte Prozessketten mit automatischer Korrektur

Markus Siepermann, Chris Börgermann, Richard Lackes

Lehrstuhl für Wirtschaftsinformatik,
Technische Universität Dortmund

1 Motivation

Grundsätzlich lassen sich drei Aufgaben in der universitären Lehre unterscheiden: (1) Die Wissensvermittlung, (2) die Wissensanwendung und -vertiefung und (3) die Wissens- und Verständnisüberprüfung. Die Bereitstellung praxisbezogener E-Learning-Aufgaben ist dabei dem Punkt (2) und bedingt auch Punkt (3) zuzuordnen. Internetbasierte E-Learning-Systeme bieten Studierenden die Möglichkeit, zu jeder Zeit und an jedem Ort zu lernen und zu üben. Gerade ein solches selbstgesteuertes Lernen stellt die effizienteste Form des Lernens dar (vgl. Kerres und Jechle 2002, S. 272). Dabei sollte ein interaktives E-Learning-System anspruchsvolle und interaktive Übungsaufgaben bereitstellen (vgl. Haack 2002, S. 129), so dass die Studierenden die Lösungen für die Fragestellungen selbständig finden müssen, indem sie ihre erlerntes Wissen anwenden und somit ihre Problemlösungskompetenz unter Beweis stellen können. E-Learning-Aufgaben sollten deshalb nicht nur aus Multiple-Choice, Wahr-Falsch, Lückentext oder Würfeltext bestehen. Denn bei dieser Art von Aufgaben sind die Lösungen bereits in der Aufgabe enthalten, so dass Studierende weniger ihr erlerntes Wissen anwenden müssen, als vielmehr durch geschicktes Ausschließen die richtigen Antworten erraten können (vgl. König 2001, S. 112). Gute E-Learning-Aufgaben, die Studierenden dabei helfen, Vorlesungsinhalte zu verstehen und Themengebiete inhaltlich zu durchdringen, sollten demnach die richtige Lösung nicht mehr oder weniger offensichtlich enthalten. Obwohl solche auch Intelligent Tutoring Systems genannten E-Learning-Systeme bereits seit den 90er Jahren Gegenstand der Forschung sind (vgl. z. B. Patel und Kinshuk 1996), kommen sie bis heute aufgrund ihrer Komplexität und der Schwierigkeit, solche Aufgaben zu realisieren (vgl. Siepermann 2005, S. 1750), im praktischen Einsatz kaum vor. Vielmehr werden solche anspruchsvollen Aufgabenformen heutzutage immer noch von menschlichen Arbeitskräften korrigiert und bewertet (vgl. Schlageter und Feldmann 2002, S. 350 ff.; König 2001, S. 111). Es sollten also E-Learning-Systeme entwickelt werden, die die folgenden drei Eigenschaften erfüllen: (a) Die Übungsaufgaben sind interaktiv und anspruchsvoll (vgl. Haack 2002,

S. 129). (b) Bei der Bearbeitung der Aufgaben sollte das Problemlösungswissen abgefragt werden (vgl. Siepermann 2005, S. 1751). (c) Das System gibt den Studierenden möglichst direktes Feedback über die Qualität der Lösung (vgl. Bolliger und Martindale 2004, S. 62). E-Learning-Aufgaben sollten somit nicht den Problemlösungsprozess beschreiben. Stattdessen sollten den Studierenden möglichst viele Freiheitsgrade zur Lösung einer Aufgabe gelassen werden. Einschränkungen sollten nur dann in Betracht gezogen werden, wenn es die software-technische Implementierung nicht anders gestattet (vgl. Lackes und Siepermann 2007, S. 7198). Außerdem sollte das System die studentischen Lösungsversuche sofort automatisiert korrigieren und bewerten können, um den Studierenden ein unmittelbares Feedback geben zu können (vgl. Bolliger und Martindale 2004, S. 62). Aufgrund der vielen Freiheitsgrade bei der Aufgabenbearbeitung ist eine solche automatische Korrektur nicht leicht zu realisieren, da es zumeist nicht nur eine korrekte Antwort gibt, sondern vielmehr mehrere Antworten, die eine unterschiedliche Güte aufweisen (vgl. Siepermann und Lackes 2007, S. 13). Um also entscheiden zu können, ob eine Lösung richtig oder falsch ist bzw. zu wie viel Prozent eine Aufgabe korrekt bearbeitet wurde, müsste diese Aufgabe vom System verstanden werden. Ein solches Verständnis ist jedoch trotz des enormen Fortschritts in der Informatik in den letzten Jahren immer noch nicht realisierbar. Stattdessen ist es jedoch möglich, formale Anforderungen zu definieren, die die studentischen Lösungen erfüllen müssen. Mit Hilfe dieser formalen Anforderungen kann ein E-Learning-System dann die studentischen Abgaben auf ihre Eignung zur Lösung der Problemstellung hin untersuchen. Es wird also eine Methode benötigt, mit der es möglich ist, formale Anforderungen an die Lösungen einer Übungsaufgabe zu formulieren und die Einhaltung dieser Anforderungen durch eine studentische Lösung dann zu überprüfen, um diese Lösung automatisch korrigieren zu können.

Im folgenden Artikel wird ein E-Learning-System vorgestellt, das eine solche Anforderungsdefinition und Aufgabenkorrektur für graphische Modellierungstechniken, insbesondere Ereignisgesteuerte Prozessketten, ermöglicht. Das System basiert auf Temporallogik und Model Checking. Der Vorteil graphischer Modellierung ist dabei offenkundig: Ein graphisches Modell stellt ein formales Abbild der Realwelt dar. Dennoch ist die Bedeutung eines solchen formalen Modells im Gegensatz zu einem rein mathematischen Modell mit vordefinierten Variablen und fixen Berechnungsschemata nicht offensichtlich (vgl. Patel und Kinshuk 1996).

2 Graphische Modellierung und E-Learning

Der Hauptzweck graphischer Modellierung besteht darin, mittels graphischer Symbole mit festgelegter Bedeutung Informationen bzgl. Diskurswelt formal abzubilden und möglichst intuitiv zu visualisieren. Die Modellierung verfolgt dabei bestimmte Ziele. Einige Modellierungstechniken fokussieren auf die Beschreibung von Daten (z. B. das Entity-Relationship-Modell), andere auf die Beschreibung von

Prozessen (z. B. Ereignisgesteuerte Prozessketten), wieder andere auf die Beschreibung von Systemverhalten (z. B. Petri-Netze) etc. Wenn solche Modellierungstechniken gelehrt und geübt werden, müssen zwei Fragen beantwortet werden: (1) Wird die Modellierungstechnik korrekt genutzt? Das heißt, haben Studierende bei der Bearbeitung Modellierungsregeln verletzt? (2) Genügt die studentische Lösung allen inhaltlichen Anforderungen? Das heißt, beschreibt die Lösung alle Informationen, die in einer Aufgabe gegeben worden sind, oder werden einige Informationen vernachlässigt?

Die erste Frage befasst sich mit der Syntax der Modellierungstechnik, die auf einfachen Regeln basiert und graphentheoretisch über eine einfache Tiefensuche überprüft werden kann (vgl. Siepermann 2005, S. 1751). Die zweite Frage betrifft die Semantik, die sehr schwer zu überprüfen ist, da Rechner i. d. R. nicht die Bedeutung eines Modells verstehen können. Ein erster, intuitiver Versuch, die Semantik zu überprüfen, besteht darin, die studentische Lösung und Referenzlösung als Black-Box zu betrachten und beide Modelle dahingehend zu vergleichen, dass bei gleicher Eingabe die gleiche Ausgabe erfolgen muss.

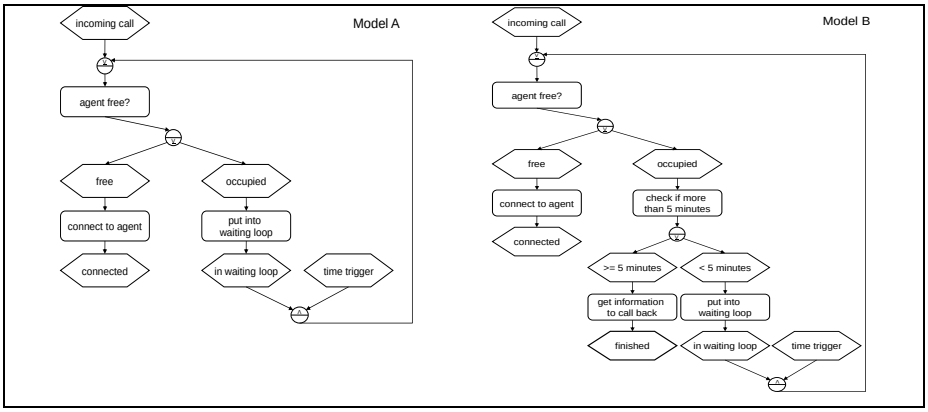

Abbildung 1: Zwei alternative Ereignisgesteuerte Prozessketten für die Anfragebearbeitung eines Call-Centers

Abbildung 1 zeigt eine studentische und die Referenzlösung für den Anfragebearbeitungsprozess eines Call Centers. Zur Überprüfung können nun einige Testereignisse (z. B. „Incoming Call“ und die abhängigen Ereignisse „free“ und „occupied“) erzeugt werden und auf beide Lösungen angewandt werden. Die resultierenden Ergebnis-Ereignisse der beiden Lösungen werden dann verglichen. Wenn die erzeugten Testereignisse repräsentativ sind und die Ergebnisse gleich, kann angenommen werden, dass die studentische Lösung die Diskurswelt abbildet und zu den richtigen Ergebnissen kommt. Ein solcher Black-Box-Test reicht oftmals beim Testen von Computer-Programmen aus, im Bereich der Lehre spielt jedoch noch eine weitere Frage eine große Rolle: (3) Ist die studentische Lösung elegant?

Auch wenn Syntax und Semantik einer studentischen Lösung korrekt sind und zum korrekten Ergebnis führen, kann die Lösung trotzdem Fehler enthalten. Diese Fehler betreffen meist die Flexibilität und den praktischen Nutzen einer Lösung: Auch wenn zwei Lösungen zum gleichen Ergebnis führen, kann die eine Lösung flexibler und genereller anwendbar sein als eine andere Lösung. Die flexiblere Lösung sollte deshalb bevorzugt werden. Wenn z. B. bei Ereignisgesteuerten Prozessketten zwei Funktionen unabhängig voneinander sind, sollten diese über parallele Stränge modelliert werden anstatt in einem sequentiellen Strang, auch wenn die Prozessergebnisse im Endeffekt gleich sind. Durch eine Taylorisierung der Arbeitsschritte könnte der parallel verlaufende Prozess z. B. schneller durchgeführt werden als der sequentiell modellierte. Wenn die graphischen Lösungen also in Teilgraphen unterteilt werden, kann eine Abstufung in der Korrektheit festgestellt werden: Falsch modellierte Teilgraphen, halbkorrekt modellierte Teilgraphen und korrekt modellierte Teilgraphen. Bezüglich der (halb)korrekt modellierten Teilgraphen können bei der Korrektur verschiedene Möglichkeiten unterschieden werden: (a) Die Teilgraphen sind äquivalent und gleichwertig. (b) Die Teilgraphen sind äquivalent, aber nicht gleichwertig. (c) Die äquivalenten Teilgraphen können ohne Einschränkung untereinander ausgetauscht werden. (d) Die äquivalenten Teilgraphen können nicht ausgetauscht werden. (e) Manche Teilgraphen bedingen die Existenz anderer Teilgraphen. (f) Manche Teilgraphen schließen einander gegenseitig aus.

Wenn also graphische Modellierung bei E-Learning-Aufgaben eingesetzt werden soll, stellen sich die folgenden vier Probleme: (1) Wie können die inhaltlichen Anforderungen formuliert werden? (Anforderungsdefinition) (2) Wie wird überprüft, ob die Anforderungen eingehalten werden? (Anforderungsüberprüfung) (3) Wie werden die studentischen Fehler behandelt? (Korrektur und Bewertung) (4) Wie wird mit wachsenden Referenzlösungen umgegangen, d.h. mit unvollständigen Anforderungen? (Lernendes System)

3 Der Einsatz von Temporallogik und Model Checking

Um das oben charakterisierte Problem zu lösen, wird im weiteren Temporallogik und Model Checking betrachtet, ein aus der Software-Verifikation bekanntes und erprobtes Verfahren, mit dem Anforderungen an Software formalisiert und überprüft werden können. Die Analogie zwischen graphischer Modellierung und Computerprogrammen besteht darin, dass jedes Computerprogramm als Graph dargestellt werden kann. Jedes Computerprogramm muss bestimmte Anforderungen erfüllen die vor dem operativen Einsatz besonders in sicherheitsrelevanten Bereichen überprüft und getestet werden müssen. Ohne Test und Verifikation würde z. B. eine Aufzugsteuerung nie zum Einsatz kommen können. Temporallogik und Model Checking ermöglichen nun eine Überprüfung, ob ein Programm stets zum gewünschten Ergebnis kommt, z. B.: „Jede Person wird irgendwann vom Aufzug

bedient." Um ein Programm testen zu können, muss die graphische Repräsentation des Programms in eine sogenannte Kripke-Struktur überführt werden (vgl. Clarke, Grumberg und Peled 2001, S. 35). Die Anforderungen an das Programm bzw. das graphische Modell können dann als temporallogische Ausdrücke formuliert werden, die mit Hilfe von Model Checking an der Kripke-Struktur überprüft werden können. Wenn kein Ausdruck verletzt wird, kann sichergestellt werden, dass das Programm oder das graphische Modell allen formulierten Anforderungen genügt. Bei der Kripke-Struktur handelt es sich um eine spezielle Form eines gerichteten Graphen, der aus Zuständen und Zustandsübergängen besteht. Die Zustände repräsentieren die Eigenschaften eines Programms oder graphischen Modells, die Zustandsübergänge die Beziehungen zwischen diesen. Für die Definition der Anforderungen können neben den bekannten booleschen Operatoren weitere temporallogische Operatoren verwendet werden. Von den verschiedenen Arten von Temporallogik eignet sich für die Überprüfung graphischer Modellierung am besten die sogenannte Computation Tree Logic (CTL). Diese verfügt i.A. über die folgenden Ausdrücke:

Operatoren

Globally:	G y	Bedingung y gilt von der aktuellen Position an in jeder folgenden Position.
neXt:	X y	Bedingung y gilt von der aktuellen Position an in der nächsten Position.
Future:	F y	Bedingung y gilt von der aktuellen Position an in mindestens einer der folgenden Positionen.
Until:	y U z	Bedingung y gilt von der aktuellen Position an in jeder folgenden Position bis Bedingung z gilt.

Quantifier

Always	A y	Bedingung y gilt auf jedem Pfad.
Exist	E y	Bedingung y gilt auf mindestens einem Pfad.

Im Beispiel aus Abbildung 1 wäre eine typische Anforderung: „Jede Person wird innerhalb von fünf Minuten bedient." In eine korrekte temporallogische Formel übersetzt, würde die Anforderung heißen: „Auf jedem Pfad gilt die Bedingung, dass ein Kunde mit einem Call-Center-Mitarbeiter verbunden wird oder seine Kontaktdaten übermittelt hat, wenn fünf Minuten vergangen sind." Bei Überprüfung der Modelle A und B wird eine Verletzung dieser Anforderung durch Modell A festgestellt, weil es einen unendlichen Pfad hinsichtlich der „Warteschlange" gibt, so dass ein Kunde dort unendlich lange warten kann.

Temporallogik und Model Checking lösen die oben formulierten vier Probleme bzgl. graphischer Modellierung bei E-Learning-Aufgaben wie folgt:

1. Anforderungsdefinition
   Die Anforderungen werden über die temporallogischen Ausdrücke formuliert. Eine Referenzlösung muss nicht erstellt werden.

2. Anforderungsüberprüfung
   Die studentischen Lösungen müssen in eine Kripke-Struktur überführt werden, die dann via Model Checking auf die Einhaltung der Anforderungen hin überprüft wird.
3. Korrektur und Bewertung
   Wenn Anforderungen nicht eingehalten werden, enthalten die Teile der studentischen Lösung, die die Anforderungen verletzen, die Fehler und können als nicht korrekt markiert werden. Da das Standard Model Checking den Teil der Kripke-Struktur nicht markiert, der die Anforderungen verletzt, muss das Verfahren entsprechen angepasst werden.
4. Lernendes System
   Wenn die Anforderungen nicht komplett definiert sind, könnte es sein, dass eine fehlerhafte Lösung als korrekt akzeptiert wird. Das tritt z. B. auf, wenn Studierende überflüssige und fehlerhafte Teile modellieren, die keine Anforderung verbietet. Deshalb müssen solche „unbekannten" Teile einer studentischen Lösung über geeignete Anforderungen entdeckt werden. Diese unbekannten Teile werden dem Dozenten übermittelt, der die Anforderungsdefinition dann geeignet anpassen kann.

Der Vorteil des Einsatzes von Temporallogik und Model Checking liegt darin, dass Anforderungen allgemeiner formuliert werden können als z. B. spezielle Referenzlösungen. Der große Nachteil liegt jedoch darin, dass der Gebrauch von Temporallogik und Model Checking nicht intuitiv ist. Dozenten müssen nun zusätzlich zur Modellierungstechnik auch noch Temporallogik beherrschen. Darüber hinaus muss der Model Checking Algorithmus angepasst werden, um fehlerhafte Teile studentischer Lösungen markieren zu können.

Aufgrund der Flexibilität von Temporallogik und Model Checking wurde ein E-Learning-System für Ereignisgesteuerte Prozessketten entwickelt, das automatisch studentische Lösungen korrigiert und bewertet. Dozenten müssen nur dann eingreifen, wenn die Anforderungsdefinition unvollständig war. Dieses System wird im Folgenden weiter vorgestellt.

4 E-Learning-System

4.1 Überblick

Das auf Temporallogik und Model Checking basierende E-Learning-System für ereignisgesteuerte Prozessketten soll die folgenden Anforderungen erfüllen:

1. Universalität
   Auch wenn zunächst der Fokus auf ereignisgesteuerten Prozessketten liegt, soll es möglich sein auch andere Diagrammtypen auf einfache Weise zu integrieren

2. Flexibilität
   Die notwendige Intervention des Tutors im Korrektur- und Bewertungsprozess soll soweit wie möglich reduziert werden
3. Selbstlernendes System
   Wenn einen Intervention im Korrektur- und Bewertungsprozess notwendig ist, so sollen die dabei neu definierten Anforderungen so generell wie möglich sein und für den weiteren Prozess dem System zur Verfügung stehen
4. Ergonomie
   Die unintuitive Methode Anforderungen über temporallogische Formeln zu definieren sollte so einfach wie möglich gestaltet werden.

Anforderung 1 wird erreicht, indem die visuellen Komponenten in spezielle Klassen aufgeteilt und die Syntax der Diagramme über temporallogische Formeln definiert werden. Daher ist keine zusätzliche Programmierung erforderlich, um neue graphische Modelle zu implementieren. Die Anforderungen 2 und 4 sind das Resultat der Benutzung von Temporallogik, da dieser Ansatz eine Reduzierung der Intervention durch den Tutor verspricht. Um bei der Anforderungsdefinition mit Temporallogik eine gewisse Usability zu erreichen (Anforderung 4) wurde eine neue graphische Repräsentation für temporallogische Ausdrücke entwickelt, mit der ereignisgesteuerte Prozessketten gekapselt werden können. Das entwickelte E-Learning-System ist ein Client-Server basiertes System, mit einer speziellen Clientsoftware, die es dem Tutor erlaubt Aufgaben zu definieren und Studenten das Üben damit ermöglicht. Die Definition einer Aufgabe wird durch die Verwendung der Client Software erreicht. Zunächst wird eine Beschreibung der Aufgabe benötigt. Danach definiert der Tutor die Anforderungen, welche jede der Lösungen zu erfüllen hat. Die Prüfung der Eigenschaften kann der Tutor prüfen, indem er selbst die Aufgabe löst. Dieser iterative Prozess stellt sicher, dass der Tutor durch die Bearbeitung der Aufgabe und die anschließende Korrektur durch das System die meisten Anforderungen definiert werden. Nach Abschluss der Aufgabendefinition wird diese zum Server geschickt und dort gespeichert. Der Server stellt nur die Aufgaben zur Verfügung und schickt diese zur Clientsoftware der Studierenden. Jede Aufgabe enthält die textuelle Beschreibung der Aufgabe und eine Menge von in Temporallogik definierten Anforderungen, welche nicht durch die Studierenden eingesehen werden können. Wenn Studierende die Bearbeitung der Aufgabe beendet haben, transformiert das System die Lösung in eine Kripke Struktur und benutzt die Menge der Anforderungen, um die studentische Lösung automatisch zu korrigieren und zu bewerten. In der Regel wird die Korrektur erfolgreich durchgeführt und das Resultat der Korrektur den Studierenden präsentiert. Sofern es Probleme bei der Korrektur gab, wird eine Aufgabe zurück zum Server gesendet und der Tutor darüber informiert. Der Tutor überprüft dann die Lösung und definiert neue Anforderungen, mit denen das System die Lösung in Zukunft automatisch korrigieren kann. Im Anschluss werden die Studierenden über die Durchsicht informiert, um die Korrektur einsehen zu können.

4.2 Benutzung

Wie in Abbildung 2 gezeigt wird besteht die Benutzeroberfläche generell aus drei Teilen: Dem Projektbereich (oben links), dem Funktionsbereich (unten links) und dem Modellierungsbereich (rechts).

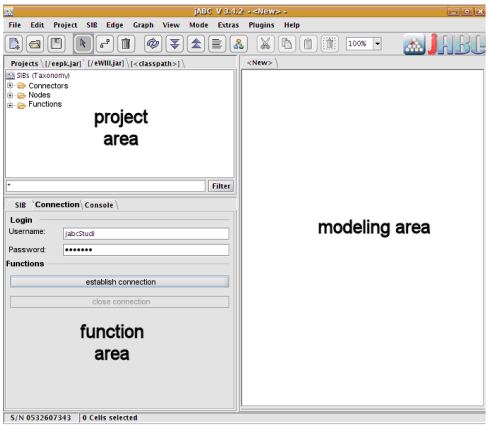

Abbildung 2: Aufteilung der Benutzeroberfläche

Um eine Aufgabe anbieten zu können, muss zunächst der entsprechende Graphentyp vorhanden sein. Sofern der gewünschte Graphentyp noch nicht im System existiert, muss dieser erstellt werden, indem die Syntaxregeln definiert werden, welche automatisch mit jeder Aufgabe dieses Graphentyps überprüft werden. Diese werden durch die graphische Repräsentation der Temporallogik auf die gleiche Weise definiert wie später die aufgabenspezifischen Regeln.

Nach Eingabe der Grunddaten für eine Aufgabe können im Modellierungsbereich die aufgabenspezifischen Regeln modellieren. Die Ereignis- und Funktionsknoten, welche für eine ereignisgesteuerte Prozesskette notwendig sind, sind ebenso im Projektbereich zu finden, wie die grafischen Repräsentationen der temporallogischen Operatoren, mit denen die Eigenschaften spezifiziert werden können. Ereignis- und Funktionselemente können im Funktionsbereich mit Bezeichnungen versehen werden. Diese Bezeichnungen stehen auch den Studierenden bei der Bearbeitung der Aufgabe zur Verfügung, um Fehler durch eine falsche Schreibweise oder die Verwendung von Synonymen zu verhindern, da diese nur schwer durch ein automatisiertes System erkannt werden können. Die daraus resultierende Abnahme des Schwierigkeitsgrades lässt sich kompensieren, indem Dummy-Bezeichnungen angelegt werden, die nicht im Graphen verwendet werden können, die aber den Studierenden ebenfalls zur Verfügung stehen. Dadurch muss eine Auswahl der zu verwendenden Bezeichnungen durch die Studierenden erfolgen. Ist eine Eigenschaft spezifiziert, wird sie in einem sogenannten Eigenschaftscontainer gekapselt. Für jeden dieser Eigenschaftscontainer kann der Tutor die erreichbaren Punkte bei Einhaltung der Eigenschaft sowie die Fehlermeldung festle-

gen, welche angezeigt wird, wenn eine Regel verletzt oder gar nicht beachtet wurde. Zwei mögliche Regeln, welche dem vorangegangenen Beispiel folgen, sind in Abbildung 3 gezeigt. Nach Abschluss der Regelspezifikation wird die graphische Repräsentation automatisch in zwei temporallogische Formeln übersetzt.

Nach der Bearbeitung einer Aufgabe wird diese durch das System korrigiert. Die Korrekturprozedur besteht aus zwei Teilen: (1) Überprüfung, ob eine Regel an einer Stelle verletzt wurde. (2) Überprüfung, ob eine Regel überhaupt berücksichtigt wurde.

Der zweite Schritt ist notwendig, da eine nicht modellierte Regel beim Model Checking nicht verletzt wird, so dass eine leere Abgabe zur vollen Punktzahl führen würde. Wenn eine Regel verletzt wurde, kann das System feststellen, wo in der Lösung die Verletzung stattgefunden hat, und diese Stelle als falsch markieren. Alle anderen Elemente werden als korrekt markiert. Im Falle einer Nichtmodellierung kann das E-Learning-System nicht entscheiden, an welcher Stelle die Eigenschaft hätte modelliert werden müssen. Es kann jedoch erkennen, dass die Modellierung dieser Eigenschaft fehlt. Mit der maximalen Anzahl an Punkten und der Anzahl an Punkten, die durch Verletzung oder Nichtmodellierung von Eigenschaften nicht vergeben wurden, berechnet sich das Ergebnis der studentischen Lösung.

4.3 Beispiel für die Eigenschaftsmodellierung

Gegeben sei die oben dargestellte Call Center Lösung, die auf die Erfüllung der folgenden Eigenschaften geprüft werden soll: (1) Nach dem Ereignis „occupied“ muss eine Funktion „check if more than 5 minutes“ folgen. (2) Nach der Funktion „check if more than 5 minutes“ gibt es mindestens einen Pfad, der zum Ereignis „finish“ führt.

Nr.	graphische Repräsentation
1	
2	

Abbildung 3: Beispiele für die graphische Modellierung temporallogischer Eigenschaften

5 Kritische Betrachtung

Auf den ersten Blick nachteilig erscheint, dass für die Modellierung der ereignisgesteuerten Prozessketten sowohl die graphischen Modellierungselemente wie auch mögliche Bezeichner bereits vorgegeben werden und somit ein wichtiger Abstrahierungsschritt entfällt, den Studierende im Rahmen einer Veranstaltung lernen sollen.

Die Vorgabe der Modellierungselemente stellt aber nur eine geringe Erleichterung für Studierende dar, da neben den korrekten Elementen auch falsche Elemente zur Verfügung gestellt werden können und so die Bestandteile dennoch erlernt werden müssen. Darüber hinaus treffen die vorgegebenen Elemente keine Aussage über die korrekte Anwendung und die weitere Syntax des Modells.

Die Vorgabe möglicher Bezeichner ist eine Erleichterung, die durchaus bei der Modellierung unterstützen kann. Beim Erlernen graphischer Modellierungstechniken stehen jedoch verschiedene Zielsetzungen im Vordergrund. Als erstes soll der Umgang mit der entsprechenden Modellierungstechnik geübt werden. Diese Zielsetzung wird durch das System unterstützt. Weiterhin ist vielfach die genaue Abbildung eines bestimmten Sachverhaltes gewünscht, wie z. B. im Rahmen einer Isterhebung üblich. Zu diesem Zweck werden Aufgaben erstellt, die eine klar umrissene Diskurswelt vorgeben, die möglichst genau als Modell abgebildet werden soll. Hierbei wird einerseits der Umgang mit der Modellierungstechnik geübt und andererseits die Übertragung geschilderter Sachverhalte in ein formales Modell. Die Vorgabe von Bezeichnern stellt hier eine sehr geringe Erleichterung dar, da diese Bezeichner i. d. R. so oder in ähnlicher Form im Aufgabentext bereits vorhanden sind. Eine weitere Zielsetzung besteht darin, dass Studierende ein möglichst optimales Modell erstellen. Hierbei steht die Struktur des Modells und weniger die Bezeichner im Vordergrund. Die Vorgabe der Bezeichner ist wie bei der vorherigen Zielsetzung lediglich eine kleine Erleichterung, aber kein Ersatz eines Abstrahierungsschrittes. Als letzte Zielsetzung ist der freie Umgang mit einer Modellierungstechnik zu nennen, bei dem Studierende eine Problemstellung selbständig erarbeiten und ein entsprechendes Modell erstellen sollen. Bei dieser Zielsetzung steht weniger die Anwendung der Modellierungstechnik als vielmehr der Abstraktionsschritt im Vordergrund, von der realen Welt zu einem formalen Modell zu gelangen. Nicht nur die Vorgabe bestimmter Bezeichner, sondern bereits die Aufgabenbeschreibung selbst stellt dabei je nach Detaillierungsgrad eine Einschränkung hinsichtlich der Zielsetzung dar. Je geringer detailliert die Aufgabenstellung ausfällt, desto größere Freiheitsgrade bieten sich jedoch für die Modellierung. Da Modellierung ein kreativer Prozess ist, der zu unterschiedlichen, aber gleichwertigen Lösungen gelangen kann, bzw. dessen Lösungen unterschiedliche Aspekte besser oder schlechter lösen können, erscheint die Schaffung eines Systems, das diesen großen Freiheitsgraden gerecht wird und eine studentische Lösung noch adäquat beurteilen kann, nicht erfolgversprechend.

Bei der Nutzung des Systems wurde festgestellt, dass eine komplette Anforderungsspezifikation mit Temporallogik anspruchsvoll und zeitaufwendig ist. Die temporallogischen Regeln, die von Tutoren erstellt werden, sind vielfach sehr einschränkend (z. B. auf ein bestimmtes Ereignis muss eine bestimmte Funktion folgen), da der Umgang mit Temporallogik zunächst nicht intuitiv ist. Allgemeine Regeln, die sich weniger mit bestimmten Abfolgen in der Lösung als vielmehr mit Anforderungen an das Modell befassen, werden nur selten verwendet. Darüber hinaus sind die Anforderungen/Regeln für eine Aufgabe bei der Definition der Referenzlösung vielfach unvollständig und müssen im Laufe der Veranstaltung verfeinert werden.

Aus diesen Gründen erscheint eine Kombination mit dem Ansatz der Graphenüberdeckung vielversprechend (vgl. Siepermann 2005), bei dem studentische Lösungen mit zuvor definierten Teillösungen überdeckt werden, die korrekt oder falsch sein können. Die Modellierung einer Referenzlösung und bestimmter typischer Fehler ist für Tutoren intuitiver als die Handhabung der Temporallogik. Mit Hilfe der richtigen und falschen Muster lassen sich temporallogische Formeln extrahieren, die dann im Rahmen des Model Checking genutzt werden können. Es besteht dabei jedoch die Gefahr, dass mit der intuitiven Herangehensweise eine Abnahme der Flexibilität bei der Erkennung von Fehlern erkauft wird.

6 Ergebnisse

In diesem Artikel wurde ein neuer Ansatz vorgestellt, welcher ermöglicht, Aufgaben zur graphischen Modellierung von Sachverhalten zu erstellen und automatisch durch ein E-Learning-System korrigieren und bewerten zu lassen. Die Hauptvorteile dieses E-Learning-Systems sind, dass Studenten zu jeder Zeit an jedem Ort üben können und damit ein selbstgesteuertes Lernen erreicht wird, welches eines der effizientesten Wege überhaupt im Bereich des Lernens ist (vgl. Kerres und Jechle 2002, S. 272). Als zweiter Vorteil erweist sich, dass die studentischen Lösungen ohne Beteiligung des Tutors korrigiert und bewertet werden können. Dadurch wird Zeit eingespart, die die Lehrenden zweckbestimmter einsetzen können. Außerdem kann es für klassische Prüfungsaufgaben verwendet werden und somit wieder Zeit sparen und Fehler bei der Korrektur minimieren. Zu guter Letzt enthalten die Aufgaben nicht die Lösungen (wie etwa bei Multiple Choice), so dass Studenten tatsächlich eigenes Wissen zur Anwendung bringen müssen, anstatt das Ergebnis zu raten oder abzuschätzen.

Literatur

Bolliger, D., Martindale, T. (2004). Key Factors for Determining Student Satisfaction in Online Courses. International Journal on E-Learning, (3), 61-67.

Clarke, E.M., Grumberg, O. Peled, D.A. (2001). Model Checking. Cambridge London: The MIT Press.

Haack, J. (2002). Interaktivität als Zeichen von Multimedia und Hypermedia. In: Issing, L.J, Klimsa, P. (Hrsg.). Information und Lernen mit Multimedia und Internet. Weinheim: BeltzPVU, 127-136.

Huth, M., Ryan, M. (2000). Logic in Computer Science: Modelling and Reasoning about Systems. Cambridge: Cambridge University Press.

Kerres, M., Jechle, T.(2002). Didaktische Konzeption des Telelernens. In: Issing, L.J, Klimsa, P. (Hrsg.). Information und Lernen mit Multimedia und Internet. Weinheim: BeltzPVU, 267-281.

König, M. (2001). E-Learning und Management von technischem Wissen in einer webbasierten Informationsumgebung. Duisburg: Druckerei Duennbier.

Lackes, R., Siepermann, M. (2007). Bru-N-O'Mat – Automatically Generating and Marking Net Requirements Calculation Exercises. In: Richards, G. (Hrsg.). Proceedings of E-Learn 2007. Chesapeake: Association for the Advancement of Computing in Education, 7198-7204.

Patel, A., Kinshuk (1996). Intelligent Tutoring Tools – A problem solving framework for learning and assessment. In: M. F. Iskander et al. (Hrsg.): Proceedings of 1996 Frontiers in Education Conference – Technology-Based Re-Engineering Engineering Education. 140-144.

Schlageter, G., Feldmann, B.(2002). E-Learning im Hochschulbereich: der Weg zu lernzentrierten Bildungssystemen. In: Issing, L.J, Klimsa, P. (Hrsg.). Information und Lernen mit Multimedia und Internet. Weinheim: BeltzPVU, 347-357.

Siepermann, M. (2005). Lecture Accompanying E-Learning Exercises with Automatic Marking. In: Richards, G. (Hrsg.). Proceedings of E-Learn 2005. Chesapeake: Association for the Advancement of Computing in Education, 1750-1755.

Siepermann, M. Lackes, R. (2007). Self-Generating and Automatic Marking of Exercises in Production Planning. In: Isaias, P., Nunes, M.B., & Barroso, J. (Hrsg.). Proceedings of the IADIS International Conference WWW/Internet 2007. Volume II, International Association for Development of the Information Society, 13-17.

Strzebkowski, R., Kleeberg, N. (2002). Interaktivität und Präsentation als Komponenten multimedialer Lernanwendungen. In: Issing, L.J, Klimsa, P. (Hrsg.). Information und Lernen mit Multimedia und Internet. Weinheim: BeltzPVU, 229-245.