Exemplarische Betrachtungen zu lernförderlicher Software mit Entwurfsmustern für Informatiksystemverständnis

Peer Stechert

Didaktik der Informatik und E-Learning
Universität Siegen
Hölderlinstraße 3
57076 Siegen
stechert@die.informatik.uni-siegen.de

Abstract: Ziel dieses Artikels ist, Software zur Förderung des Informatiksystemverständnisses zu beschreiben, die vernetzte fundamentale Ideen und Schüleraktivitäten lernförderlich miteinander verknüpft. Anhand der theoretischen Fundierung im Rahmen eines Unterrichtsmodells und des exemplarischen Einsatzes einer Software zur Zugriffskontrolle mittels Proxymuster in zwei schulpraktischen Erkundungen werden Anforderung an lernförderliche Software abgeleitet und ihr Potential diskutiert, Problemstellen im Unterricht zu bewältigen.

1 Motivation

Informatiksystemverständnis ist ein wichtiges Ziel informatischer Bildung. Um Informatiksysteme in ihrer Einheit aus Hardware, Software und Vernetzung [CS06] zu verstehen, müssen Verhalten, innere Struktur und Implementierungsaspekte im Informatikunterricht auf die in ihnen vernetzten fundamentalen Ideen der Informatik [Sc93] zurückgeführt werden [SS07]. Als Zugang zu Informatiksystemverständnis wird das nach außen sichtbare Verhalten gewählt. Da vernetzte fundamentale Ideen nur in wenigen publizierten Unterrichtskonzeptionen thematisiert werden, ergibt sich, dass Lehrpersonen diese selbst vernetzen müssen. Zur Unterstützung entwickelte der Autor ein Unterrichtsmodell zur Förderung des Informatiksystemverständnisses, das Strukturmodellen wie Entwurfsmustern als Wissensrepräsentationen für vernetzte fundamentale Ideen der Informatik eine Schlüsselrolle zukommen lässt. Darüber hinaus ist für das Unterrichtsmodell, d.h. für den theoretischen Rahmen für unterschiedliche Unterrichtskonzepte und -reihen in der Sekundarstufe II, eine Vorgehensweise zur systematischen Erkundung von Informatiksystemen als Unterrichtsexperiment entwickelt worden [SS07].

In diesem Artikel wird Software zur Förderung des Informatiksystemverständnisses betrachtet, die eine Analyse von Verhalten, Struktur und Implementierungsaspekten im Rahmen des Unterrichtsmodells ermöglicht. Zur Wahrung des exemplarischen Charakters muss sie auf einem oder mehreren Strukturmodellen basieren, die gleichzeitig Wissensrepräsentationen für vernetzte fundamentale Ideen der Informatik sind. In zwei Un-

terrichterkundungen wurden mittels formativer Evaluation der konkreten Bildungssituation Widersprüche, offene Fragen und Potential des Unterrichtsmodells und der lernförderlichen Software aufgedeckt. In diesem Artikel werden anhand einer Beispielsoftware[1] zur Zugriffskontrolle und ihrer Evaluation im Unterricht Konsequenzen für die Gestaltung lernförderlicher Software im Rahmen des Unterrichtsmodells abgeleitet. Der Artikel ist folgendermaßen strukturiert: In Abschnitt 2 wird die Software theoretisch im Rahmen des Unterrichtsmodells begründet und ihre exemplarische Bedeutung erläutert. In Abschnitt 3 werden Problemstellen im Unterricht diskutiert, die durch die Kombination mit der systematischen Erkundung des Systemverhaltens anhand der Beispielsoftware bewältigt werden können. Diese sind 1) die Brücke zwischen dem nach außen sichtbaren Verhalten und der inneren Struktur, 2) fehlende Erfahrung der Schüler mit systematischen Erkundungen des Systemverhaltens und 3) die Vernetzung fundamentaler Ideen.

2 Rolle der Software zur Zugriffskontrolle im Unterrichtsmodell

2.1 Beitrag der Zugriffskontrolle zum Verstehen von Informatiksystemen

Zur Gestaltung von Lernsoftware zur Förderung des Informatiksystemverständnisses im Rahmen des Unterrichtsmodells ist die Frage zu stellen, welche Aspekte von Informatiksystemen und welche Sichten auf diese Systeme lernförderlich zu verknüpfen sind. Denning beschreibt sieben Kategorien der Informatik: computation, communication, coordination, automation, recollection, evaluation, design [De07]. Er stellt fest:

> „These categories cover the main functions of computing systems“ [De07, S. 15].

Diese Sichten („windows of computing mechanics“) können mit einem fachdidaktischen Sichtenkonzept kombiniert werden, wie Brinda es für Lernsoftware fordert [Br04, S. 52]. Schüler[2] verstehen die Konsequenzen ihres Handelns anhand des Wechsels zwischen den Perspektiven und der Analyse der durch Schülertätigkeiten hervorgerufenen Änderungen in den Sichten. So ist Zugriffskontrolle z.B. bezüglich zuverlässiger Datenübertragung (communication), ihres Entwurfs (design) sowie der Koordination von Anwendern und Maschinen (coordination) zu betrachten.

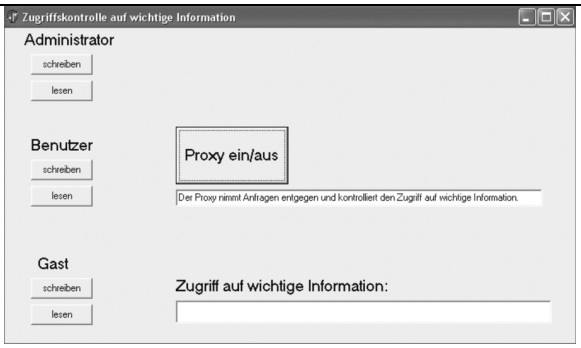
Zugriffskontrolle erfüllt die Kriterien für fundamentale Ideen der Informatik [Sc93]. Dabei ist zu betonen, dass der Ideenkatalog nach Schwill keinen Anspruch auf Vollständigkeit erhebt. Für die Arbeit mit Informatiksystemen ist meist eine Authentifizierung notwendig und den Schülern vom Schulnetz vertraut. Auf Zugriffskontrolle mittels Stellvertreter treffen Lernende in Form von Platzhaltern für Grafiken in Textdokumenten und beim Navigieren im Internet (Sinnkriterium). Zugriffskontrolle ist auf unterschiedlichen intellektuellen Niveau erklär- und verstehbar, da bereits Kinder Zugriffskontrolle durch ihre Eltern erfahren (Vertikalkriterium). Seit der Einführung von Mehrbenutzersystemen, z.B. UNIX, ist Zugriffskontrolle in der Informatik relevant und wird durch die zunehmende Verbreitung von Informatiksystemen längerfristig relevant bleiben (Zeitkri-

[1] Lernmittel sind über die Webseite des Autors verfügbar: www.die.informatik.uni-siegen.de/lehrstuhl/stechert/
[2] Alle Personenbezeichnungen gelten gleichermaßen für die männliche und weibliche Form.

terium). Zugriffskontrolle kommt in der Informatik sowohl bei Mensch-Maschine-Interaktion bei Betriebs- und Anwendungssystemen als auch bei deren Entwurf vor, wenn Zugriff auf unterschiedliche Schichten kontrolliert wird (Horizontalkriterium).

2.2 Kriterien zur Auswahl von Entwurfsmustern für Informatiksystemverständnis

Zur Vernetzung der Lerninhalte wurde Potential in ausgewählten Entwurfsmustern identifiziert [St06]. Bei der Gestaltung von Lernmitteln für eine ersten Erkundung des Unterrichtsmodells spielte die konkrete Lernsituation in Jahrgangsstufe 12 der Kooperationsschule eine große Rolle. Die Schüler besaßen Vorwissen zur objektorientierten Modellierung und zur Datenstruktur Liste. Um daran anzuknüpfen und die Vernetzung der Lerninhalte herzustellen, wurde erst das Iteratormuster als Darstellungsmittel der Informatik herangezogen, und anschließend das Proxymuster zur Zugriffskontrolle. Beide können in einem Szenario, in dem Zugriff auf eine Liste stattfindet, kombiniert werden.



1) Wie lautet der Name des Systems? (Erinnern; Faktenwissen)
2) Beschreiben Sie die Benutzungsoberfläche. (Verstehen; Faktenwissen)
3) Welche Funktionalitäten vermuten Sie unter Berücksichtigung der vorherigen Beobachtungen? (Verstehen; Konzeptwissen)
4) Welche Beziehungen bestehen zwischen Elementen? (Verstehen; Konzeptwissen)
5) Welche informatischen Konzepte wurden eingesetzt, um die Funktionalität zu erreichen? (Analysieren; Konzeptwissen)
6) Analysieren Sie das Programm, indem Sie Sonderfälle ermitteln und mit ihnen experimentieren. (Analysieren; Konzeptwissen)
7) Werten Sie Fehler und unerwartetes Verhalten aus – auch mit Blick auf die Sonderfälle der Konzepte. (Evaluieren; Konzeptwissen)
8) Hypothetischer Einsatz des Systems: „Wie würde sich das Programm in anderen (realen, komplexeren) Situationen verhalten?“ (Erzeugen; Konzeptwissen)

Abbildung 1: Benutzungsoberfläche und Aufgabe zur Erkundung der lernförderlichen Software zur Zugriffskontrolle mit kognitivem Prozess und Wissensart gemäß Lernzieltaxonomie [AK01]

Abbildung 1 zeigt eine Aufgabe aus der ersten Unterrichtserprobung und die Benutzungsoberfläche der Software, die auf dem Entwurfsmuster Proxy basiert. Es lassen sich die Rollen Administrator, Benutzer und Gast identifizieren. Falls ein Proxy eingeschaltet ist, unterscheiden sich die Schreib- und Leserechte. Bei der Erkundung des Systemverhaltens müssen die Schüler Hypothesen darüber bilden, ob Zugriffskontrolle vorliegt und wie diese umgesetzt ist, um Fehlvorstellungen zu überwinden.

Stechert und Ufer [St06], [Uf07] entwickelten fünf Kriterien, nach denen Entwurfsmuster klassifiziert werden können, um den Bildungsprozess zum Verstehen von Informatiksystemen positiv zu beeinflussen. Durch sie können Entwurfsmuster ausgewählt werden, die Informatiksystemverständnis fördern und als Wissensrepräsentationen bezeichnet seien. Ziel hinsichtlich der lernförderlichen Software ist, dass die fachdidaktisch durch die Kriterien begründeten Eigenschaften der ausgewählten Entwurfsmuster dadurch für den Unterricht nutzbar gemacht werden, dass sie die innere Struktur der Software bilden. Die Begründung der Kriterien stützt sich im Wesentlichen darauf, dass Strukturmodelle das Verstehen von Informatiksystemen fördern [SS07] und dass in Informatiksystemen fundamentale Ideen erkennbar sind, die sich beeinflussen. Nach der positiven Klassifikation des Proxymusters wurde anschließend die Beispielsoftware zur Zugriffskontrolle entwickelt. Das erste Kriterium dient dementsprechend dazu, die Vernetzung fundamentaler Ideen in Entwurfsmustern zur fachdidaktischen Kommunikation explizit zu machen (K1: Vernetzte Ideen). Das zweite Kriterium, Zusammenhänge mit weiteren Entwurfsmustern [GHJV95], ermöglicht Anknüpfungspunkte für weiteren Unterricht (K2: Mustersprache). Die Analyse der Komplexität (K3: Komplexität) basiert auf einer Arbeit von Harrer und Steinert [HS02] zu Softwaremustern in der Hochschullehre. Deren Klassifikation anhand des Abstraktionsgrads und der für die Softwaretechnik wichtigen Konzepte, z.B. Interaktionsregelung und Management, wurde für allgemein bildenden Unterricht zu Informatiksystemverständnis verändert und erweitert. Zusätzlich wird das Einsatzgebiet (K4: Einsatzgebiet) betrachtet, in dem die Verallgemeinerbarkeit des durch Anwendung des Musters lösbaren Problems analysiert wird. Der Lebensweltbezug (K5: Lebenswelt) knüpft an den Alltag der Schüler an. In Kombination mit Anforderungen aus konkreten Lehr-Lernprozessen lassen sich Konsequenzen für lernförderliche Software ableiten. In Abschnitt 3 werden die Kriterien aufgegriffen, z.B. die Vernetzung der Zugriffskontrolle mit den fundamentalen Ideen Vererbung und Schnittstelle in der lernförderlichen Software, um Problemstellen im Unterricht zu bewältigen.

2.3 Exemplarische Betrachtung zur Zugriffskontrolle mittels Proxymuster

Kognitive Barrieren und Fehlvorstellungen treten auf, wenn Erwartungen der Lernenden nicht mit der Realität in Einklang zu bringen sind und sollen mit Hilfe der lernförderlichen Software abgebaut bzw. korrigiert werden. In dem Kontext ist das Lernen aus Fehlern, z.B. ausgelöst durch unerwartetes Verhalten von Informatiksystemen, ein Ansatzpunkt. Deshalb wurde die in der Aufgabe angegebene Schrittfolge zur systematischen Erkundung des Verhaltens von Informatiksystemen entwickelt (Abbildung 1). Sie basiert auf dem Ablauf von Unterrichtsexperimenten [Me06] und spricht sukzessive die sechs Stufen der kognitiven Prozesse aus der überarbeiteten Lernzieltaxonomie nach Bloom [AK01] an: Erinnern, Verstehen, Anwenden, Analysieren, Bewerten und Erzeugen. Grundlage ist die für Experimente typische Hypothesenbildung, die zur Erklärung des nach außen sichtbaren Verhaltens genutzt wird, sowie Wiederholbarkeit. Eine Bereicherung des Lehr-Lernprozesses wird durch die Kombination von Wissensrepräsentationen (gemäß der Kriterien K1 bis K5) mit der Vorgehensweise zur systematischen Erkundung erzielt (Abbildung 1). Durch Schritt 3 der Vorgehensweise erfolgt ein Bezug zu den Hauptfunktionen von Informatiksystemen nach Denning [De07]. Der Bezug zu informatischen Konzepten (Schritt 5) und deren Sonderfälle (Schritt 6) ermöglicht die Klassifi-

kation des Systemverhaltens. Gleichzeitig bilden sie den Anknüpfungspunkt zu vernetzten fundamentalen Ideen der Informatik (K1: Vernetzte Ideen). Da diese per Definition in der Wissensrepräsentation und damit in der Software vorhanden sind, unterstützen sie Schüler und Lehrperson, unterschiedliche Fälle im Systemverhalten zu klassifizieren. So können beim Proxymuster unterschiedliche Zugriffsrechte identifiziert werden. Aufgrund des Lebensweltbezugs der Entwurfsmuster (K5: Lebenswelt) ist es für Schüler über Analogien leichter möglich, Hypothesen zu bilden, z.B. zu unterschiedlichen Zugriffsrechten, die überprüft werden. Darauf aufbauend sind Hypothesen zur inneren Struktur zu prüfen. Zusammenhänge mit anderen Entwurfsmustern (K2: Mustersprache) zeigen Anknüpfungspunkte zur Erweiterung der lernförderlichen Software, so dass Ergebnisse von Experimenten vor und nach der Erweiterung durch Schüler verglichen werden können. Anhand der Hauptfunktionen von Informatiksystemen ist außerdem eine Diskussion über Informatiksysteme und Gesellschaft strukturierbar (Schritt 8).

3 Problemstellen im Unterricht und Rolle der Software

3.1 Brücke zwischen dem Verhalten und der inneren Struktur von Systemen

Eine erste Erkundung des Unterrichtsmodells im unterrichtlichen Geschehen wurde vom Autor im Herbst 2006 mit 23 Schülern in einem Grundkurs der Klassenstufe 12 durchgeführt. Exemplarische Lernziele waren: Die Schüler verstehen ...

S_A: Zugriffskontrolle mittels Proxymuster anhand des sichtbaren Verhaltens,
S_B: statische und dynamische Aspekte der Zugriffskontrolle mittels Proxymuster durch Analyse der Komponenten, z.B. via Klassen- und Sequenzdiagramm,
S_C: eine auf Vererbung beruhende Zuweisung von Zugriffsrechten anhand der Modifikation des Quelltextes des Proxymusters.

Dabei bezeichnet S_i die Strukturierung von Basiskompetenzen für Informatiksystemverständnis als Weiterentwicklung der Wissensstrukturen nach Brinda [Br04]. Sie sind nach dem nach außen sichtbaren Verhalten (S_A), der inneren Struktur (S_B) und Implementierungsaspekten (S_C) von Informatiksystemen unterteilt. Im Unterricht und in der benoteten schriftlichen Übung wurden Problemstellen im Unterricht und kognitive Hürden ausgemacht. Bei der in Abbildung 1 dargestellten Vorgehensweise zur Erkundung des Informatiksystems wurde die für Unterrichtsexperimente wichtige Dokumentation der Schritte anfangs von vielen Schülern vernachlässigt. Geeignete und den Schülern bekannte formale informatische Modelle wurden genutzt, um Quelltext zu veranschaulichen, z.B. Klassendiagramm, aber nicht, um einen Sachverhalt oder eine Hypothese hinsichtlich des Systemverhaltens darzustellen. Die Auswertung ergab, dass eine Verfeinerung der Strukturierung der Basiskompetenzen zur Überwindung der gedanklichen Brüche zwischen den Perspektiven vorzunehmen ist. Damit wird es möglich, Überleitungen wie „... und nun kommen wir zur inneren Struktur des Informatiksystems!“ zu vermeiden. Unter der Prämisse, dass ein Zugang zum Informatiksystemverständnis über das nach außen sichtbare Verhalten zu wählen ist, ergibt sich folgende Verfeinerung der Strukturierung, die in der zweiten schulpraktischen Umsetzung eingesetzt wurde: nach außen sichtbares Verhalten von Informatiksystemen (S_A), innere Struktur von Informatiksystemen (S_B), Kombination des nach außen sichtbaren Verhaltens mit der inneren

Struktur (S_{AB}), Kombination des sichtbaren Verhaltens mit Implementierungsaspekten (S_{AC}). Implementierungsaspekte (S_C) und die Zwischenebene zur Kombination der Implementierungsaspekte mit der inneren Struktur (S_{BC}) werden für Basiskompetenzen nicht betrachtet [SS07].

Die systematische Erkundung des nach außen sichtbaren Verhaltens der lernförderlichen Software erhält hinsichtlich der Strukturierung der Basiskompetenzen die Einordnung S_A, wobei das Informatiksystem zunächst als Black Box betrachtet und systematisch analysiert wird. Anschließend werden Bezüge zur inneren Struktur und zu ausgewählten Implementierungsaspekten betrachtet, um das zugrunde liegende Strukturmodell zu verstehen. Zur Kombination des nach außen sichtbaren Verhaltens mit der inneren Struktur (S_{AB}) bieten sich Klassen-, Objekt- und Sequenzdiagramme als formale Darstellung an, die z.B. durch die Lernsoftware Pattern Park [St06] im Kontext von vernetzten fundamentalen Ideen und Entwurfsmustern erlernt werden können. Mit ihnen können Objekte, die aus dem Verhalten des Systems abgeleitet und auf der Benutzungsoberfläche identifiziert wurden, beschrieben werden. Das Analysieren der Systemkomponenten kann beispielsweise zum Testen und Modifizieren (S_{AC}) einzelner Komponenten führen. Ufer klassifizierte 2007 in seiner vom Autor betreuten Diplomarbeit Softwaremuster und erstellte auf Basis ausgewählter Muster lernförderliche Software [Uf07]. Dabei bewährte sich die Weiterentwicklung im Bereich der Wissensstrukturen (S_i) nach Brinda, indem sie mit Aufgabe und Aufgabentyp sowie mit einer Einordnung in die Kompetenzstrukturierung, Lernziel, und Lernzielebene [AK01] verknüpft wurden. Ein Aufgabentyp war z.B. „Beobachtung mit vorgegebener Vorgehensweise“.

3.2 Fehlende Erfahrung mit systematischen Erkundungen von Informatiksystemen

Im Zusammenhang mit der Kombination der Perspektiven des nach außen sichtbaren Verhaltens und der inneren Struktur steht, dass den Schülern Erfahrung mit systematischen Erkundungen des Verhaltens von Informatiksystemen fehlte. Da die Schüler programmieren konnten, verfügten sie implizit über eine Vorgehensweise zur Analyse von Quelltext. In der ersten Unterrichtserprobung mussten sie nun die in Abbildung 1 gegebene Vorgehensweise zur Erkundung der lernförderlichen Software zur Zugriffskontrolle durchführen (K5: Lebenswelt). Eine erwartete Fehlvorstellung zur Zugriffskontrolle war, dass ein Objekt den Zugriff auf sich selbst kontrolliert. Sei es aus Sicherheits- oder Performancegründen wird jedoch meist, z.B. beim Proxymuster, ein Stellvertreterobjekt mit der gleichen Schnittstelle wie der des Originalobjekts eingesetzt. Daher wurde die Benutzungsoberfläche der Software so gestaltet, dass die Schüler leichter erkannten, dass ein Stellvertreterobjekt eingesetzt wird (Abbildung 1). Im weiteren Verlauf der systematischen Erkundung des Systemverhaltens waren die Schüler nicht in der Lage, die Frage, wie ein Stellvertreter das selbe Verhalten wie das Original haben kann, hypothetisch zu beantworten. Erst bei der Analyse des Quelltextes erkannten die Schüler, dass Zugriffskontrolle mit Vererbung (K1: Vernetzte Ideen) realisiert wurde, indem Stellvertreter und Original von der gleichen (abstrakten) Klasse erben. Dadurch ergibt sich für beide die gleiche Schnittstelle bei durchaus unterschiedlicher Spezialisierung. Eine gleichnamige Operation „lesen()“ bzw. „schreiben()“ wird beim Original ein bestimmtes Verhalten auslösen, während beim Stellvertreter nur die Prüfung der Zugriffsrechte und ggf. die Weiterleitung an das Original durchgeführt wird. Zur Dokumentation der Erkundungs-

schritte sollten die Schüler ein Klassendiagramm erstellen, das der Struktur des Proxymusters entsprach. In diesem Zusammenhang ist es wichtig, Variationen des Entwurfsmusters (K3: Komplexität), z.B. die Erweiterung um die Benutzungsoberfläche, explizit im Unterricht aufzugreifen. So fragten Schüler bei der Diskussion des Klassendiagramms des Proxymusters, wie der Zugriff auf das Muster umgesetzt werden könne. Dies ist durch die Benutzungsoberfläche, oder allgemeiner, durch einen Klienten möglich, der in das Klassendiagramm integrierbar ist. Die Dokumentation des Systemverhaltens durch Formalisierung von Abläufen stellte sich als weitere Hürde dar.

Die systematische Erkundung von Informatiksystemen wurde im Vorfeld eines zweiten Unterrichtsprojektes verfeinert, um die Akzeptanz bezüglich der gewählten Fachsprache zu prüfen. Außerdem wurden die Erkundungsschritte hinsichtlich S_A, S_{AB}, S_{AC} und S_B angepasst. Zwei Studierende führten dazu die Methode des Laut-Denkens als Laborstudie im Rahmen eines Hauptseminars mit vier studentischen Probanden durch. Da die gewählten Probanden zum Teil eine geringere informatische Vorbildung als Schüler eines Informatikkurses hatten, eigneten sie sich besonders, um diese Aspekte zu verbessern. Die zweite Erkundung des Unterrichtsmodells wurde von zwei Lehramtsstudierenden im Herbst 2007 mit 13 Schülern eines Grundkurses der Klassenstufe 12 durchgeführt. Die handlungsorientierte Vorgehensweise bildete in dieser Unterrichtserprobung den ersten Schwerpunkt, um den vorher festgestellten Mangel an Vorerfahrungen mit systematischen Erkundungen von Informatiksystemen zu beheben. Dafür wurde zu Beginn die Wichtigkeit der Hypothesenbildung der Schüler zum Verstehen des nach außen sichtbaren Verhaltens thematisiert. Es wurde die Schrittfolge der systematischen Erkundung von Schülern und Lehrperson gemeinsam im Plenum durchgeführt. Anschließend erkundeten die Schüler wie im ersten Unterrichtsprojekt die lernförderliche Software zur Zugriffskontrolle mit den Rollen Administrator, Benutzer und Gast (Abbildung 1).

Bei der Beschreibung der Benutzungsoberfläche sind zwei Vorgehensweisen der Schüler offensichtlich geworden: Eine Schülergruppe hat die graphische Benutzungsoberfläche in Blöcke mit Sinnzusammenhang hinsichtlich unterschiedlicher Teilnehmer (-objekte) mit unterschiedlichen Zugriffsrechten unterteilt und diese in Leserichtung von links nach rechts beschrieben. Eine zweite Gruppe hat die Oberflächenelemente nach bekannten Programmiersprachenkonstrukten sortiert, nämlich nach Labels, Buttons und Textfeldern. Beim Experimentieren mit Sonderfällen vermutete eine Schülergruppe allein durch das nach außen sichtbare Verhalten des Programms eine Realisierung der Zugriffskontrolle über Case-Anweisungen. Diese nicht mit der objektorientierten Denkweise konforme Vorstellung konnte analog zur ersten Unterrichtserprobung durch Identifizieren von Klassen überwunden werden, die zeigen, wie ein Stellvertreter durch Vererbung das gleiche Erscheinungsbild bzw. die gleiche Schnittstelle wie das Original haben kann. Durch die aufeinander aufbauenden Schritte zur systematischen Erkundung des Informatiksystems stellte die begleitende Anfertigung eines Klassendiagramms zur Dokumentation und Formalisierung keine unüberwindbare Hürde dar. Interessant ist, dass eine Schülerin bei der Beschreibung der Beziehungen zwischen den Oberflächenelementen nachfragte: „Ab wann dürfen wir ausprobieren?“. Wenngleich die Frage darauf abzielte, eine Versuch-Irrtum-Vorgehensweise anzuwenden, so zeigte der Vergleich der Schülerlösungen durch Nennung unterschiedlicher, auch verworfener Hypothesen, dass Hypothesenbildung erfolgte. Die Wiederholbarkeit als Kriterium für Experimente zeigte sich

dadurch, dass in beiden Erprobungen ähnliche Ergebnisse beobachtet wurden, die in der zweiten durch gezielte Vorbereitung verständiger diskutiert wurde.

Um der fehlenden Erfahrung der Schüler mit der systematischen Erkundung von Informatiksystemen zu begegnen, konnte eine weitere, auf dem Entwurfsmuster Proxy als Wissensrepräsentation basierende lernförderliche Software eingesetzt werden. Dabei unterstützte der Ansatz der auf dem Proxymuster basierenden Software die Erstellung einer in der Struktur unveränderten Software mit einem anderen Kontext. Es handelte sich um eine Arztpraxissoftware, die von den Schülern systematisch erkundet, und in der Zugriffskontrolle auf Patienten durch eine Arzthelferin realisiert wurde (K4: Einsatzgebiet). Das in der Klassifikation der Entwurfsmuster genannte Kriterium der Komplexität (K3: Komplexität) liefert dabei Hinweise auf Variationen des Proxymusters, so dass z.B. Schwerpunktverlagerungen auf weitere fundamentale Ideen im Proxymuster (K1: Vernetzte Ideen) vorgenommen werden können [St06] oder die Anzahl der Zugriffe durch den Stellvertreter gezählt werden kann.

3.3 Vernetzung von fundamentalen Ideen und Entwurfsmustern

Ein wichtiger Vorteil von Entwurfsmustern wurde bislang nicht aufgegriffen: ihre Kombinierbarkeit mit weiteren Entwurfsmustern (K2: Mustersprache). Da dies kein Selbstzweck ist, wurde für eine Fortsetzung im Unterricht ein Beitrag zur Erklärung des nach außen sichtbaren Verhaltens durch das Zustandskonzept analysiert:

> „[. . .] das System als solches bleibt gleich, es behält seine Identität und seinen Namen. Der Zustand besteht aus den jeweils vorhandenen Komponenten, ihren Eigenschaften und ihren Beziehungen. Die Folge der Zustände bezeichnet man als das **Verhalten des Systems**“ [GZ06, S. 18; Hervorh. i. Orig.].

Dabei ist jedoch darauf hinzuweisen, dass es sich um eine theoretische Betrachtung des Systemverhaltens handelt, die nicht ohne Schwierigkeiten mit dem nach außen sichtbaren Verhalten eines Systems gleichzusetzen ist. Dadurch ist eine Bildungsherausforderung beschrieben, die für Schüler das Potential bietet, mittels Zustandsautomaten eine Brücke vom nach außen sichtbaren Verhalten zur inneren Struktur des Informatiksystems zu bauen. Zustandsautomaten stellen im Unterricht neben den Sequenzdiagrammen eine weitere Sicht auf Abläufe dar, um die kognitive Hürde der Formalisierung zur Dokumentation zu überwinden. Deshalb wurde eine Erweiterung der Arztpraxissoftware um das Zustandsmuster vorgenommen und im Unterricht eingesetzt. Abbildung 2 zeigt die Kombination von Proxy- und Zustandsmuster in der Arztpraxis. Zustände des Programms sind abhängig von Gesundheitszuständen der Patienten. Streng genommen muss zwischen Systemzuständen und Zuständen der Patienten unterschieden werden. Für Schüler ist die im Informatiksystem vorhandene Zugriffskontrolle dahingehend ein Unterscheidungsmerkmal. Sie erstellten anhand des Verhaltens des Programms Zustandsdiagramme in Abhängigkeit der Zugriffskontrolle (K1: Vernetzte Ideen). Somit erfolgte eine weitere Vernetzung. Bei der systematischen Erkundung erleichtert der Lebensweltbezug die Hypothesenbildung (K5: Lebenswelt). Die Schüler mussten in der Rolle des Arztes dem Patienten die richtige Medikamentendosis in Abhängigkeit von deren Gesundheitszustand geben. Nachfolgend sind exemplarische Aufgaben angegeben:

S_A: Beschreiben Sie die Benutzungsoberfläche in eigenen Worten. Gehen Sie dabei nur auf Veränderungen zur ersten Arztpraxissoftware mit Zugriffskontrolle ein (Verstehen; Faktenwissen),

S_{AB}: Heilen Sie einen Patienten, der im Zustand „krank“ ist. Hinweis: Eine Krankenschwester ist nicht anwesend. Begründen Sie Ihre Vermutung. Wie viele Schritte sind mindestens bis zur Genesung des Patienten nötig? Überprüfen Sie Ihre Vermutung anhand eines gegebenen Zustandsdiagramms und dokumentieren Sie Ihre Schritte sowie die Zustände des Patienten (Anwenden; Konzeptwissen).

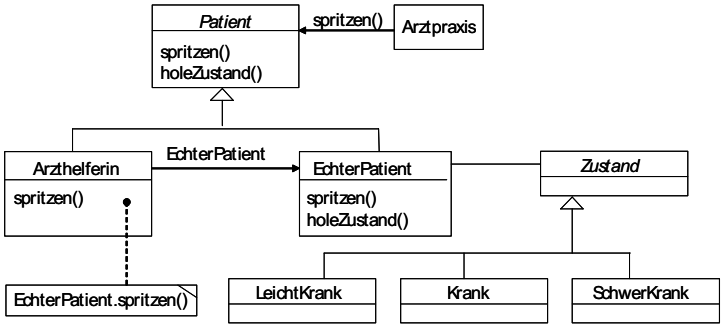


Abbildung 2: Klassendiagramm der Arztpraxis mit den Entwurfsmustern Proxy und Zustand

Interessant hinsichtlich Informatiksysteme und Gesellschaft war die Diskussion der Schüler, in der sie hinterfragten, inwieweit eine durch ein Programm vorgegebene Vereinfachung der Lebenswelt hinzunehmen sei. Nächster Schritt war eine Modellierung der inneren Struktur durch Zustandsdiagramme (S_B), um Quelltext mit dem Zustandsdiagramm in Verbindung zu bringen (S_{AC}). Bewusst wurde dazu ein Zugang über die Analyse des Systemverhaltens gewählt, statt anhand der Klassen, Operationen und Attribute.

4 Zusammenfassung und Fazit

In dem vorgestellten Unterrichtsmodell zur Förderung des Informatiksystemverständnisses werden Verhalten, innere Struktur und Implementierungsaspekte als Perspektiven auf Informatiksysteme genutzt [St06], um die Systeme in Form lernförderlicher Software systematisch und handlungsorientiert zu erkunden. Eine Schlüsselrolle nehmen Strukturmodelle, z.B. Entwurfsmuster, ein [SS07]. Die lernförderliche Software zur Zugriffskontrolle hat diesbezüglich exemplarischen Charakter, da sie auf dem Proxymuster basiert, und die Dualität von Verhalten und Struktur eine systematische Erkundung unterstützt. In den Unterrichtsprojekten wurde die lernförderliche Software genutzt, um Problemstellen zu bewältigen. Diese waren 1) die Brücke zwischen der Perspektive des nach außen sichtbaren Verhaltens und der inneren Struktur, 2) fehlende Erfahrung der Schüler mit systematischen Erkundungen des Systemverhaltens und 3) die Vernetzung fundamentaler Ideen und Entwurfsmuster. Zu 1) erwies sich die verfeinerte Strukturierung der Basiskompetenzen für Informatiksystemverständnis als geeignet, um bei der systematischen Erkundung der Software das Systemverhalten mit Aspekten ihrer inneren Struktur zu verbinden. Der fehlenden Erfahrung der Schüler mit einer systematischen Vorge-

hensweise zur Erkundung des Systemverhaltens (2) konnte durch explizite Thematisierung der Hypothesenbildung und -prüfung im Unterricht begegnet werden. Dabei unterstützte der Ansatz der auf dem Entwurfsmuster Proxy aufbauenden lernförderlichen Software den Wissenstransfer der Schüler: Eine strukturell unveränderte Software mit einem anderen Kontext wie der Arztpraxis konnte im Unterricht ohne großen Aufwand für die Lehrperson eingesetzt werden. Die Vernetzung fundamentaler Ideen (3) konnte einerseits durch den Einsatz solcher Variationen der im Kern unveränderten Software mit anderer Schwerpunktsetzung im Unterricht erfolgen. Andererseits ist es möglich, durch Kombination mit weiteren Entwurfsmustern, z.B. Zustandsmuster, eine für Informatiksystemverständnis förderliche Software zu gestalten. Bei einer Erkundung des Verhaltens des erweiterten Systems können die Schüler auf vorherige Erkenntnisse aufbauen. Fazit ist, dass sich die Software auf Grundlage des Proxymusters in Kombination mit einer systematischen Vorgehensweise zur Erkundung von Informatiksystemen im Informatikunterricht bewährt hat. Über das Proxymuster kann die lernförderliche Software sowohl auf ihr nach außen sichtbares Verhalten als auch auf ihre für die informatische Bildung angemessene innere Struktur überprüft werden. Die Unterrichtsprojekte zeigten, dass Schüler einen experimentierenden Zugang über die lernförderliche Software annehmen und die darin enthaltene Vernetzung fundamentaler Ideen der Informatik entdecken.

Literaturverzeichnis

[AK01] Anderson, L.; Krathwohl, D. (Hrsg.): A taxonomy for learning, teaching and assessing: A revision of Bloom's Taxonomy of educational objectives. Addison Wesley, NY, 2001.

[Br04] Brinda, T.: Didaktisches System für objektorientiertes Modellieren im Informatikunterricht der Sekundarstufe II, Dissertation, Universität Siegen, 2004.

[CS06] Claus, V. ; Schwill, A.: Duden Informatik A-Z. Fachlexikon für Studium, Ausbildung und Beruf. Duden Verlag, Mannheim, 2006.

[De07] Denning, P.: Computing is a natural science. In: Commun. ACM 50, (7), 2007 S. 13-18.

[GHJV95] Gamma, E.; Helm, R.; Johnson, R.; Vlissides, J.: Elements of Reusable Object-Oriented Software. Addison-Wesley, Reading, MA, 1995.

[GZ06] Goos, G.; Zimmermann, W.: Vorlesungen über Informatik. Band 1: Grundlagen und funktionales Programmieren. Springer, Berlin, 2006.

[HS02] Harrer, A.; Schneider, M.: Didaktische Betrachtung zur Unterrichtung von Software-Mustern im Hochschulbereich. In: Schubert, S.; Magenheim, J.; Hubwieser, P.; Brinda, T. (Hrsg.): Forschungsbeiträge zur Didaktik der Informatik – Theorie, Praxis, Evaluation. 1. Workshop der GI-Fachgruppe DDI, Bd. 22 (LNI), 2002, S. 67-76.

[Me06] Meyer, H.: Unterrichtsmethoden. Bd. II: Praxisband. Cornelsen Scriptor, Berlin, 2006.

[Sc93] Schwill, A.: Fundamentale Ideen der Informatik. In: Zentralblatt für Didaktik der Mathematik 1, 1993, S. 20-31.

[SS07] Stechert, P.; Schubert, S.: A Strategy to Structure the Learning Process Towards Understanding of Informatics Systems. In Proc. of Working / Joint IFIP-Conference Informatics, Mathematics and ICT: A golden triangle. Boston, 2007.

[St06] Stechert, P.: Unterrichtsmodellentwicklung zur Förderung des Informatiksystemverständnisses mit Entwurfsmustern. In: Schwill, A.; Schulte, C.; Thomas, M. (Hrsg.): Didaktik der Informatik. 3. Workshop der GI-Fachgruppe DDI, Bd. 99 (LNI), 2006. S. 89-98.

[Uf07] Ufer, J.: Architekturmuster als Beitrag zum Informatiksystemverständnis. Diplomarbeit. Lehrstuhl Didaktik der Informatik und E-Learning. Universität Siegen, 2007.