

**Fujaba als Werkzeug  
zur anschaulichen Darstellung  
ausgewählter abstrakter Datentypen**

Schriftliche Hausarbeit  
Vorgelegt im Rahmen der Zweiten Staatsprüfung  
für das Lehramt der Sekundarstufen I und II  
am Studienseminar Sek II Paderborn  
gem. OVP v. 20.12.2001  
Paderborn, Mai 2004  
Erstgutachter: Thomas Freye  
von

Carsten Schulte

---

## Inhaltsverzeichnis

1 Fragestellung und Genese des Themas.....	2
2 Fujaba als Veranschaulichungshilfe.....	4
2.1 Aktivitätsdiagramme.....	4
2.2 Dobs.....	5
3 Zur Funktion von Abbildungen im Lernprozess.....	7
4 Veranschaulichung des Konzepts an ausgewählten Bausteinen der Unterrichtsreihe.....	10
4.1 Objektdiagramme: Einführung in den ADT Schlange.....	11
4.1.1 Erfahrungen und Rückmeldungen der Schülerinnen und Schüler.....	13
4.2 Der Zettel-Test.....	15
4.2.1 Erfahrungen und Rückmeldungen der Schülerinnen und Schüler.....	16
4.3 Dobs als Visualisierungshilfe: Knotenobjekte.....	17
4.3.1 Erfahrungen und Rückmeldungen der Schülerinnen und Schüler.....	20
4.4 Story-Driven-Modelling: Der ADT Liste.....	20
4.4.1 Fälle finden.....	22
4.4.2 Story-Boarding.....	22
4.4.3 Zusammenführen der Einzelteile zu einem Aktivitätsdiagramm.....	24
4.4.4 Erfahrungen und Rückmeldungen der Schülerinnen und Schüler.....	26
5 Fazit.....	28
6 Literatur.....	30
7 Versicherung.....	32
8 Anhang .....	33
8.1 Glossar.....	33
8.2 Weitere Diagramm-Beispiele.....	34
8.3 CD.....	36

# 1 Fragestellung und Genese des Themas

Fujaba (Fischer, Niere und Torunski, 1998) wird im Pelizaeus-Gymnasium im Jahrgang 11 zur Einführung der Objektorientierung eingesetzt. Den Einsatz des Werkzeugs im Anfangsunterricht mit Hilfe des von mir und der life<sup>3</sup>-Gruppe entworfenen life<sup>3</sup>-Konzepts habe ich im Rahmen meiner Promotion untersucht (Schulte 2004, life-Webseite). Die praktische Erprobungsphase fand unter anderem am Pelizaeus-Gymnasium im Schuljahr 2001/02 statt. Seitdem wird hier nach diesem Konzept in der Jahrgangsstufe 11 unterrichtet. Konzept und Werkzeug wurden mittlerweile auch in der Sekundarstufe I erprobt (Reinsch 2003). Zum Teil unabhängig entstandene Varianten davon werden auch mit anderen Werkzeugen in der Sekundarstufe II eingesetzt (Moll 2002, Johlen 2002).

Im dem Konzept für den Anfangsunterricht wird Fujaba aus verschiedenen Gründen eingesetzt:

- Aus UML-Klassendiagrammen wird im Hintergrund Quelltext generiert.
- Ein grafischer Debugger erlaubt das Inspizieren von Objekten zur Laufzeit. Zudem können Methoden aufgerufen und Beziehungen zu anderen Objekten angezeigt werden.
- Aktivitätsdiagramme ermöglichen es, mit UML-Diagrammen rein grafisch Methoden zu implementieren.

Damit kann mit Fujaba der Schwerpunkt des Unterrichtens auf die Modellierung gelegt werden, die syntaktischen und semantischen Strukturen für die im Anfangsunterricht behandelten Projekte lassen sich integriert in einen UML-/Modellierkurs vermitteln.

Eine Besonderheit sind jedoch die Aktivitätsdiagramme, die nicht allgemein gebräuchlich sind. Sie ersetzen UML-Sequenz- und Kollaborationsdiagramme. Während Klassendiagramme und der Umgang mit Objekten, Beziehungen, Attributen etc. auch im Jahrgang 12 benötigt wird, werden die Aktivitätsdiagramme nicht mehr unbedingt im Unterricht eingesetzt.

Für mich stellt sich daher die Frage, ob sich der Einsatz von Aktivitätsdiagrammen 'lohnt'? Im Anfangsunterricht wird so das Erlernen einer Programmiersprache umgangen: die grafischen Modelle werden sozusagen von Fujaba implementiert<sup>1</sup>. Die Frage ist, ob die relativ einfachen grafischen Bausteine ausreichen, um damit auch komplexere Probleme implementieren zu können.

Zudem sind die einfachen Aktivitätsdiagramme aus dem Anfangsunterricht sehr anschaulich. Den Schülerinnen und Schülern bereitet das 'Lesen' im Allgemeinen keine Schwierigkeiten. Unklar ist, ob mit Aktivitätsdiagramme auch etwas komplexere Methoden beschrieben werden können. Die zweite Frage lautet also, ob auch umfangreichere Methoden mit Fujaba anschaulich dargestellt werden können.

Drittens wird ein Verfahren benötigt, mit dem die Schülerinnen und Schüler diese Diagramme erstellen können. Dieses Verfahren muss im Unterricht thematisiert werden. Es gibt den Ansatz des Story-Driven-Modelling (Diethelm, Geiger und Zündorf 2002, 2004), der jedoch recht aufwändig wirkt. Wie könnten beispielsweise Methoden von Abstrakten Datentypen (ADTs) nach diesem Verfahren erstellt werden – und wie aufwändig ist es, dieses Verfahren bzw. Ausschnitte davon im Unterricht zu benutzen?

<sup>1</sup> Genau genommen muss das grafische Modell dieselbe syntaktische und semantische Eindeutigkeit wie eine Implementation besitzen, damit automatisiert Quelltext generiert werden kann. Im Grunde benutzen die Schülerinnen und Schüler eine grafische Programmiersprache, deren Syntax allerdings nach meinen Erfahrungen zumindest schneller als die Java-Syntax erlernt werden kann. (Gleichzeitig bereitet die grafische Programmierung den späteren Einsatz von Programmiersprachen vor.)

Daher möchte ich im Rahmen der Arbeit ein Konzept zum Einsatz von Fujaba in der Jahrgangsstufe 12 entwickeln und erproben, dass die visuellen Darstellungen von Fujaba für die Unterrichtung abstrakter Datentypen (ADT) einsetzt.

In der vorliegenden Arbeit soll also die Frage untersucht werden, ob Fujaba besondere didaktische Einsatzmöglichkeiten zur anschaulichen Darstellung und Implementation abstrakter Datentypen anbietet. Der Schwerpunkt der Arbeit liegt damit bei den Lehrerfunktionern Innovieren und damit verknüpft Unterrichten.

Der Einsatz von Fujaba bezieht sich zum einen auf den Lernprozess, der mit Hilfe von grafischen Notationen als Veranschaulichungen unterstützt werden soll. Hier sind Fragen zum Lernen mit Abbildungen angesprochen.

Andererseits dient die Veranschaulichung der Schwerpunktsetzung des Unterrichts: Die Implementation soll vor allem der Realisierung und Überprüfung einer (zuvor) gefundenen Lösung dienen (Richtlinien, S. 13): Auch „noch so attraktive Werkzeuge dürfen nicht dazu führen, dass deren Bedienung Richtschnur des Unterrichts wird“ (Richtlinien, S.13). Stattdessen sollen Sprachen, grafische Notationen und Entwicklungsumgebungen, hier also die *Diagramme von Fujaba als Werkzeuge zur Beschreibung der Lösung* eingesetzt werden - und nicht als Selbstzweck.

Des Weiteren steht im Hintergrund die Frage nach inhaltlichen und unterrichtsmethodischen *Zusammenhängen zwischen dem Anfangsunterricht und späteren Unterrichtseinheiten*. Sinnvoll wäre m. E. einzelne Aspekte des Unterrichts, aus dem Anfangsunterricht, spiralig wieder aufzugreifen. Damit könnte evtl. besser bestimmbar werden, bis zu welchem Lernschritt Fujaba eingesetzt werden soll: um Grundbegriffe zu vermitteln, zur Einführung in die UML, oder auch zur Einführung in die grafische Programmierung, die in der 12 (möglicherweise zum Thema ADT) wieder aufgegriffen wird.

Zudem soll untersucht werden, wie man den *Einsatz von Fujaba mit unterrichtsmethodischen Schritten verknüpfen* kann.

Das Thema ADT bietet sich an, da diese in der objektorientierten Sichtweise Beziehungen zwischen Objekten und Operationen auf entsprechenden Objektstrukturen definieren. Zur anschaulichen Darstellung werden üblicherweise Modelle oder grafische Veranschaulichungshilfen eingesetzt, die das Verständnis erleichtern sollen – doch zur Implementation wird dann meist auf eine textuelle Programmierung zurückgegriffen. Fujaba könnte evtl. zugleich Veranschaulichungshilfe und Werkzeug zur Implementation sein.

## 2 Fujaba als Veranschaulichungshilfe

Abstrakte Datentypen werden im Allgemeinen als Erweiterung der Datentypen der verwendeten Programmiersprache aufgefasst, die zur Lösung spezieller Aufgabentypen beitragen, etwa der ADT Schlange zur Lösung von Warteschlangenproblemen, indem er das FIFO-Prinzip umsetzt. Aus objektorientierter Sicht kann die Datenstruktur als Objektstruktur aufgefasst werden, bei der ein verwaltendes Objekt, eine Schlange, ein Stapel eine Liste, etc. mehrere Inhaltsobjekte verwaltet. Diese Objektstrukturen und die darauf angewendeten Operationen können mit der UML visualisiert werden.

Die grafische Notation UML hat sich in der Fachwissenschaft als Standard-Notation für die objektorientierte Modellierung durchgesetzt (Zündorf 2002, Oesterreich 1999, S. 203) und wird in allen Phasen des Entwicklungsprozesses eingesetzt (vgl. Jacobson, Booch und Rumbaugh 1999). Die grafische Notation erlaubt die Betrachtung des objektorientierten Systems auf einer abstrakteren Ebene als dies in einer Programmiersprache möglich wäre. Es ist eine grafische Sprache:

„not only to communicate with others but to provide a setting in which individual developers can think and analyze. [...] Basically, the UML enables the developers to visualize their work products in standardized blueprints or diagrams“ (Jacobson, Booch und Rumbaugh, 1999, S. 421).

Ziel des Konzepts ist es, die Diagramme so im Unterricht zu nutzen, dass sie als Veranschaulichungshilfe bei der Problemlösung und/oder Dokumentation sowie als Grundlage für die Implementation dienen können.

Da UML-Klassendiagramme allgemein bekannt sind, gehe ich im Folgenden nur auf die visuelle Implementation in Fujaba, die Aktivitätsdiagramme (Abschnitt 2.1), sowie die Darstellung von Objektstrukturen zur Laufzeit im grafischen Debugger Dobs (Abschnitt 2.2) ein<sup>2</sup>.

### 2.1 Aktivitätsdiagramme

In Fujaba werden Methoden als Aktivitätsdiagramme implementiert. In Abbildung 1 ist ein Fujaba-Aktivitätsdiagramm dargestellt. Den Methodenkopf sieht man in der Mitte oben (1), von dort beschreiben Pfeile, die Transitionen, den algorithmischen Ablauf. Dieser beginnt mit einem rechteckigen Kästchen, einem Story-Pattern (2).

Story-Pattern haben drei Aufgaben:

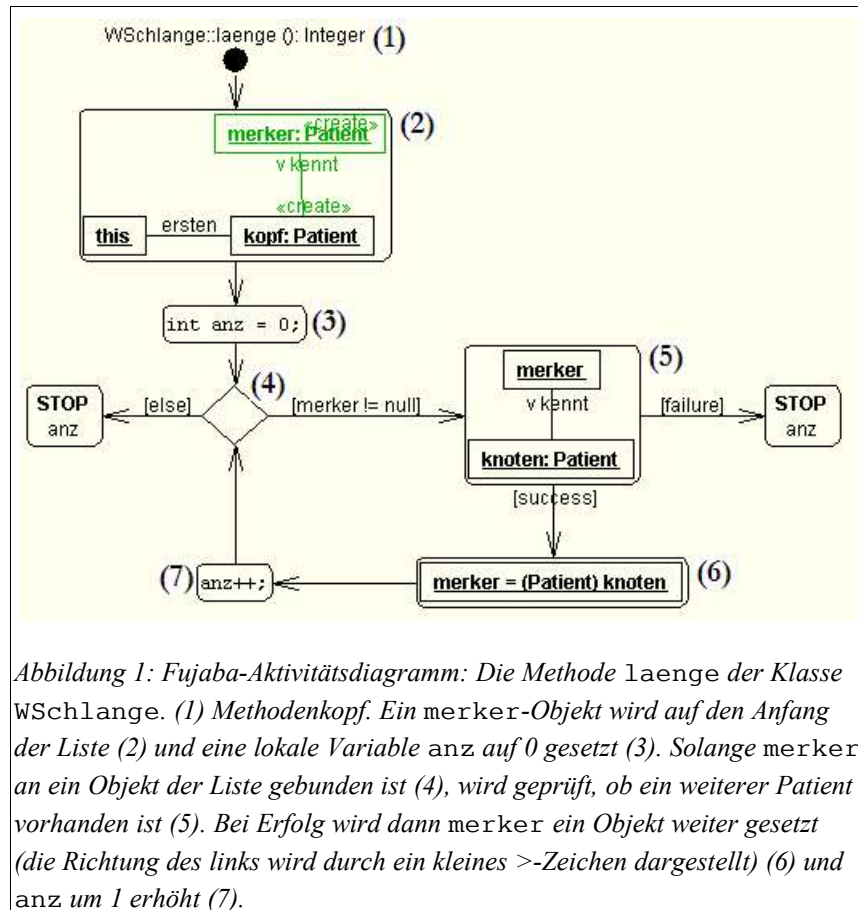
- 1) Aufgabe 1: Sie prüfen, ob die im Story-Pattern beschriebene Objektstruktur tatsächlich so existiert. Dieser Schritt ist Voraussetzung, um die Struktur verändern zu können. Dazu muss ein Story-Pattern ein bereits bekanntes Objekt haben, von dem aus das Laufzeitsystem prüfen kann, ob die gesuchte Struktur vorhanden ist. Diese bekannten Objekte sind gebundene Objekte: `bound`<sup>3</sup>. Im Beispiel wird geprüft, ob in das Schlangen-Objekt `this` ein Patient, das Objekt `kopf`, eingeschlangt ist. Die Verbindungen zwischen Objekten werden als links bezeichnet.
- 2) Aufgabe 2: Sie erweitern die Objektstruktur, indem neue Objekte, neue Beziehungen oder neue Attributwerte gesetzt werden. In dem ersten Story-Pattern der Abbildung 1 wird, falls die Schlange ein `kopf`-Element hat, ein `merker`-Objekt erzeugt. Dieses erstellt eine Be-

<sup>2</sup> Weitere Erläuterungen zu Fujaba und Dobs finden sich unter [life.upb.de](http://life.upb.de) und auf der beiliegenden CD. Unter anderem eine Flash-Animation, welche die Ausführung eines Story-Pattern darstellt.

<sup>3</sup> Erkennbar sind sie daran, dass sie nur einen Bezeichner ohne Klassennamen haben. In diesem Beispiel ist `this` ein `bound`-Objekt. Als reservierter Bezeichner bezieht `this` sich auf das Spieler-Objekt, welches die Methode ausführt.

ziehung kennt zum kopf-Element. Die neu zu erstellenden Elemente werden mit dem Wort `create` markiert und grün dargestellt.

- 3) Aufgabe 3: Sie können Elemente der Objektstruktur löschen, etwa Beziehungen oder Objekte. Diese Löschvorgänge werden durch das Wort `destroy` markiert und rot dargestellt.



Ungewöhnlich, aber praktisch, ist, dass die Aufgaben 2 und 3 eines Story-Pattern (Erzeugen und Löschen) nur dann ausgeführt werden, wenn die vorherige Prüfung (Aufgabe 1) erfolgreich war. Dadurch können oft explizite Prüfungen von Vorbedingungen entfallen. Beispielsweise die Frage, ob die Schlange leer ist, da diese Prüfung in die Semantik der Story-Pattern integriert ist. Daher liefert das obige Aktivitätsdiagramm auch für eine leere Schlange die richtige Länge als Ergebnis zurück.

## 2.2 Dobs

Der Einblick in das Laufzeitverhalten objektorientierter Software durch Programmieren und Modellieren kann mit den üblichen Entwicklungsumgebungen nicht oder nur indirekt gewonnen werden. Fujaba bietet mit dem grafischen Debugger Dobs (Dynamic Object Browsing System) die Möglichkeit, Objektstrukturen zur Laufzeit zu visualisieren und interaktiv zu verändern (Abbildung 2). In Dobs können Methoden auf Objekten aufgerufen werden, etwa einschlangen oder ausschlangen.

Der grafische Debugger Dobs soll eingesetzt werden, damit die Schülerinnen und Schüler das Verhalten der Objekte beobachten können und ersetzt so die sonst üblichen Darstellungen und Modelle, an denen die Funktion eines ADT erläutert wird.

In Dobs wird zunächst ein Objekt ausgewählt; anschließend eine Methode, die dann auf dem ausgewählten Objekt ausgeführt wird. Zur vollständigen Darstellung des Ergebnisses muss die Anzeige gegebenenfalls aktualisiert werden, damit neue Objekte und/oder Beziehungen angezeigt werden (im Kontextmenü 'expand' auswählen).

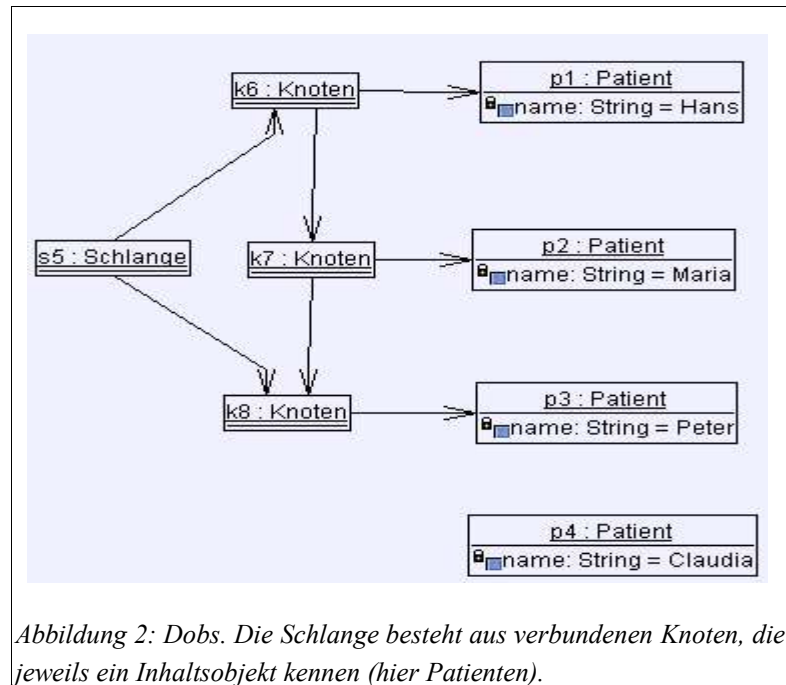


Abbildung 2: Dobs. Die Schlange besteht aus verbundenen Knoten, die jeweils ein Inhaltsobjekt kennen (hier Patienten).

Organisatorisch erfordert dies, entsprechende Beispiele bereitzustellen. Dazu sollten diese direkt in Dobs gestartet werden können, ohne den Weg über Fujaba gehen zu müssen<sup>4</sup>. Des Weiteren sollte bei den bereitgestellten Beispielen durch die Lehrkraft auf sprechende Attributnamen geachtet werden, denn diese werden in Dobs als Beschriftung der links verwendet (Abbildung 11, S. 19 zeigt einen mit dem Mauszeiger ausgewählten link).

Im Unterricht bieten sich Phasen des Erkundens sowie des Ausprobierens eigener Lösungen für den Einsatz von Dobs an. Die Schülerinnen und Schüler können dann interaktiv eine Datenstruktur und die darauf möglichen Operationen erkunden, analysieren, anwenden o.ä.

Dobs ermöglicht auch, auf die Implementation von grafischen Oberflächen zu verzichten, da es mit der Visualisierung von Objektstrukturen eine Art Benutzungsschnittstelle erzeugt.

<sup>4</sup> Eine einfache Möglichkeit unter Windows ist das Kopieren einer .bat-Datei in das Verzeichnis mit den Java-Klassen, die Dobs mit folgendem Aufruf startet: `java -cp C:\programme\fujaba\grafik.jar;C:\programme\fujaba\fujaba.jar; de.uni_paderborn.fujaba.dobs.DobsApp` (Der Pfad zu Fujaba muss natürlich an die lokale Installation angepasst werden). Auf diese Weise öffnet sich beim Erzeugen von Objekten in Dobs ein Dialog, der die im Startverzeichnis befindlichen Java-Klassen zur Auswahl anbietet.

### 3 Zur Funktion von Abbildungen im Lernprozess

Bezüglich des Informatikunterrichts gibt es bislang so gut wie keine empirischen Untersuchungen über die Qualität einzelner unterrichtsmethodischer Zugänge oder Unterrichtswerkzeuge. Anregungen zum Einsatz von Fujaba im Unterricht können jedoch aus den Kenntnissen der anderen Fächer desselben Aufgabenbereichs gewonnen werden. In TIMSS III<sup>5</sup> (Baumert, Bos und Lehmann 2000) wurden mathematische und physikalische Kompetenzen am Ende der gymnasialen Oberstufe untersucht:

„Zusammenfassend läßt sich festhalten, dass der Mathematikunterricht der gymnasialen Oberstufe aus Schülersicht bemerkenswert variationsarm ist. Vorherrschend sind zwei miteinander korrespondierende modale Muster: Sobald die Lehrkraft einen mathematischen Gedankengang entwickelt und vorgestellt hat, folgen in der Schülerarbeitsphase das Lösen von Gleichungen und die Übung von Rechenfertigkeiten.“ (Baumert und Köller, 2000, S. 283)

Physikunterricht wird hauptsächlich als Demonstrations-Unterricht durchgeführt, in dem der Lehrer anhand eines Experiments einen physikalischen Gedankengang entwickelt (Baumert und Köller, 2000, S.295f). Am lernwirksamsten ist dabei folgendes Muster: Die Lehrkraft legt Wert auf anspruchsvolle Aufgaben und theoretisches Verständnis; unterstützt Lernen durch gut vorbereitete Experimente, wobei Schüler- und Lehrerexperimente nicht der theoretischen Fragestellung (also nicht induktiv) vorgelagert sind. Die verfügbare Unterrichtszeit wird optimal genutzt (aaO., S.297). Dobs beispielsweise könnte dementsprechend eingesetzt werden, um in Experimenten die Funktionsweise eines ADT zu erkunden, nachdem die Theorie, nämlich die Idee des ADT oder der zu untersuchenden Methode, vorher geklärt wurde.

Allgemein, so die deutschen PISA-Autoren, könne eine Verbesserung der Unterrichtsqualität erreicht werden durch „mehr inner- und außermathematische Vernetzungen, weniger Verfahren und Kalküle, mehr Denkaktivitäten und Eigenkonstruktionen der Schüler, mehr Reflexion, flexiblere[n] Methodeneinsatz“ (Klieme, Neubrand und Lüdtke 2001, S. 186f, nach Blum 2001). Zudem wird vermutet, dass „das verständnisfördernde Potenzial von visuellen Darstellungsformaten noch lange nicht ausgenutzt ist“ (aaO., S.187). Die verschiedenen Darstellungsformen von Fujaba sollten demzufolge in verschiedenen methodischen Varianten zum Einsatz kommen. Dieses bezieht den Einsatz der Diagramme auf Papier oder an der Tafel ein.

In diesem Zusammenhang ist eine von Reinsch (2003, S. 24 und 33) wiedergegebene Kritik an den Aktivitätsdiagrammen in Fujaba zu bedenken: Diese würden das Programmieren unstrukturierter Sprünge erlauben und so das Ziel des Informatikunterrichts konterkarieren, 'strukturiertes Denken' zu vermitteln. In Bezug auf Flussdiagramme gibt es eine Reihe negativer empirischer Befunde über deren höhere Lernwirksamkeit gegenüber textuellen Darstellungen (Aktivitätsdiagramme können als eine Art von Flussdiagrammen gelten). So berichten Shneiderman u. a. (1977) über eine Serie einzelner Studien mit Studierenden, zum Teil in Anfängerkursen:

„Although our original intention was to ascertain under which conditions detailed flowcharts were most helpful, our repeated negative results have led us to a more skeptical opinion of the utility of detailed flowcharts under modern programming conditions.“ (Shneiderman u.a. 1977, S. 380)

Der Zusammenhang zwischen Flussdiagrammen und Struktogrammen (Nassi-Shneiderman-Diagrammen) ist nicht ganz eindeutig, die Diskussion scheint davon auszugehen, dass die Dar-

<sup>5</sup> In TIMSS werden verschiedene Untersuchungspopulationen unterschieden. TIMSS III bezieht sich auf Schülerinnen und Schüler der Sekundarstufe II.

stellungen vergleichbar sind<sup>6</sup>. Insgesamt bleibt die Forschungslage dazu etwas widersprüchlich, vermutlich weil zu viele Inferenzen hineinspielen (Scanlan 1988, S. 186).

Anschaulichkeit ist also eine schwer zu bestimmende Größe. Das zeigen auch die Ergebnisse von Kutar, Britton und Barker (2002). Sie haben die Verständlichkeit von Sequenz- und Kollaborations-Diagrammen vergleichend untersucht. Ihre Untersuchungsthese lautete, dass Kollaborationsdiagramme weniger gut verständlich seien. Beispielsweise ist der zeitliche Ablauf in Kollaborationsdiagrammen weniger gut ersichtbar, da die Transitionen zwischen Objekten frei wählbar sind, während in Sequenzdiagrammen der zeitliche Verlauf an der Sortierung der Transitionen von oben nach unten deutlich wird. In einer Studie mit 124 Studierenden des Grundstudiums ergab sich jedoch kein Unterschied in der Verständlichkeit der beiden Diagrammtypen. Die Autoren vermuten, dass möglicherweise die Vertrautheit mit den dargestellten Beispielen eine Rolle spiele, oder dass die 'unübersichtlicheren' Kollaborationsdiagramme sorgfältiger 'gelesen' werden.

Die Eignung von Darstellungen im Unterricht hängt sehr eng mit der der unterrichtsmethodischen Einbettung bzw. ihrer Nutzung durch die Lernenden zusammen. Das hier zu entwickelnde Konzept wird daher – gemäß den Lehrerfunktionen, den Schwerpunkt auf Möglichkeiten zur unterrichtlichen Nutzung der Diagramme legen.

Hier ist die Frage nach der genauen Funktion von Darstellungen im Lernprozess angesprochen. Der Ansatz 'Wissenserwerb mit Multimedia' nach Schnotz soll hier zur Erläuterung und als Grundlage herangezogen werden, da dieser Ansatz auf umfangreichen Vorarbeiten und empirischen Untersuchungen basiert (Schnotz, 2001).

Der Ansatz geht davon aus, dass bestimmte Informationsrepräsentationen eine von zwei möglichen kognitiven Verarbeitungsstrategien begünstigen: Über die visuelle Rezeption werden Informationen eher bildhaft-analog verarbeitet (dann wirken Darstellungen als depiktionale Repräsentation). Gesprochene Sprache wird symbolhaft-abstrakt (als deskriptionale Repräsentation) verarbeitet. Bilder, insbesondere Texte, können auch als deskriptionale Repräsentation wirken (siehe Abbildung 3).

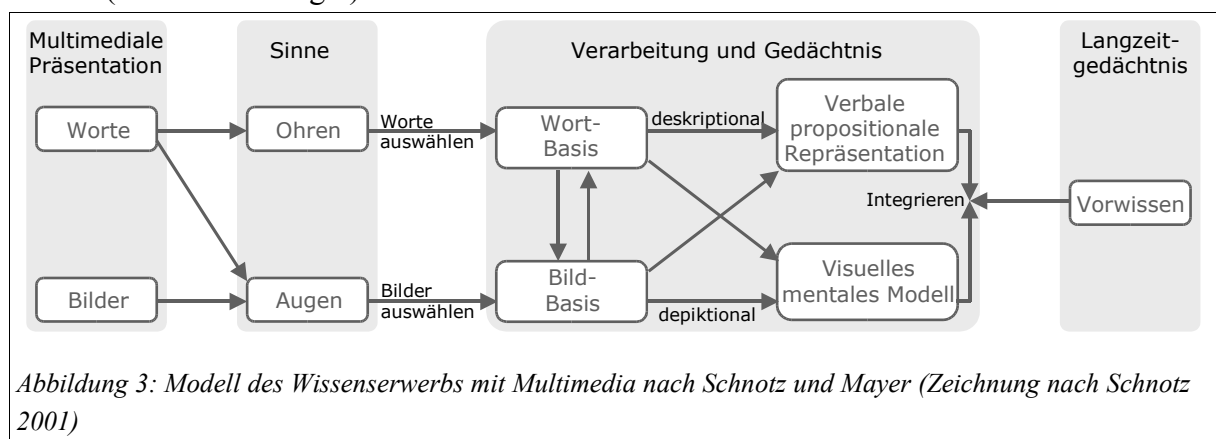


Abbildung 3: Modell des Wissenserwerbs mit Multimedia nach Schnotz und Mayer (Zeichnung nach Schnotz 2001)

Die gleichzeitige Präsentation von bildhaft-analoger und symbolhaft-abstrakter Information auf einem Sinneskanal (meist dem visuellen Quelltext plus Grafik zur Veranschaulichung) kann nach diesem Modell zu kognitiven Überlastungen führen ('cognitive load'). Günstiger wäre beispielsweise, bildhaft-analoge Informationen über den visuellen Kanal und gleichzeitig angebotene symbolhaft-abstrakte Informationen über den auditiven Kanal zu transportieren.

<sup>6</sup> Z.B. Scanlan 1988, S. 188: „Although this research deals with flowcharts, it is possible that with appropriate reservations the results could be generalized to other graphical techniques.“

Genau hier greift der Einsatz von Visualisierungshilfen im Unterricht: Die visuelle Darstellung der ADT's in UML-Notation wird durch die verbale Erläuterung des Lehrers oder der Schüler ergänzt. Dabei zielt die bildliche Darstellung auf die visuelle Verarbeitung als depiktionale Repräsentation, sie soll inhaltliche Zusammenhänge verdeutlichen und Analogiebildungen unterstützen. Ergänzt durch die verbalen Erläuterungen als deskriptionale Repräsentationen wird das symbolhafte, abstrakte Verstehen gefördert.

Eine Simultanpräsentation fördert das Integrieren der unterschiedlichen Repräsentationsformen, wobei auch das Vorwissen (Abbildung 3) eine Rolle spielt. Zudem wird durch die Wahl unterschiedlicher Sinneskanäle die Gefahr kognitiver Überlastung gemindert und die Informationsaufnahme optimiert.

## 4 Veranschaulichung des Konzepts an ausgewählten Bausteinen der Unterrichtsreihe

In diesem Kapitel wird das Konzept zum Einsatz der grafischen Notationen vorgestellt und an Bausteinen einer Unterrichtsreihe konkretisiert.

Die Unterrichtsreihe dient dazu, das Konzept im Unterricht zu evaluieren. Daher schließt sich an die Vorstellung der Bausteine jeweils eine kurze Evaluation an, die auf kurze schriftliche und mündliche Befragungen der Schülerinnen und Schüler zu den einzelnen Bausteinen zurückgreift, in denen nach der Anschaulichkeit und dem Nutzen der Diagramme sowie des unterrichtsmethodischen Vorgehens gefragt wurde. Im Folgenden steht nicht die Unterrichtsreihe selbst, sondern der Einsatz von Fujaba zur anschaulichen Darstellung ausgewählter abstrakter Datentypen im Mittelpunkt.

Die Unterrichtsreihe selbst orientiert sich an einer Vorlage, die auf dem NRW-Bildungsserver [learnline](#) ([learnline](#)) zu finden ist. Sie folgt den allgemeinen Ansätzen zur Gestaltung von Unterrichtsreihen: Lernprozesse sollten im „Kontext der Anwendung“ erfolgen und die Anwendung informatischer Inhalte erfordern oder durch Systematisierung, Abstraktion und Verallgemeinerung zu Fachkenntnissen zu gelangen (Richtlinien, S.16). Als Anwendungskontext dient eine Arztpraxis, in der Patienten im Wartezimmer auf ihre Untersuchung warten. Durch zusätzliche Anforderungen an die Verwaltung der Patienten wird aus einer Verkettung über eine Warteschlange, den ADT Schlange schließlich der ADT Liste eingeführt. Die Konkretisierung erfolgt im Rahmen des Wartezimmer-Projekts, indem exemplarisch einzelne Bausteine vorgestellt werden.

Der grafische Debugger Dobs wird eingesetzt, um interaktiv Daten- bzw. Objektstrukturen aufzubauen und zu verändern.

Die grafische Programmiersprache von Fujaba soll zur Implementation und als Veranschaulichungshilfe abstrakter Datentypen eingesetzt werden, um die Lücke zwischen dem modellhaften, anschaulichen Verständnis (vgl. Abbildungen 4, 5) und der Implementation zu verringern.

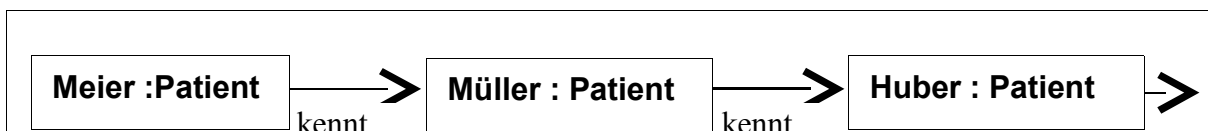


Abbildung 4: ADT als Verkettungsstruktur (Schlange oder Liste). Darstellung aus dem Wartezimmer-Projekt auf ([learnline](#)).

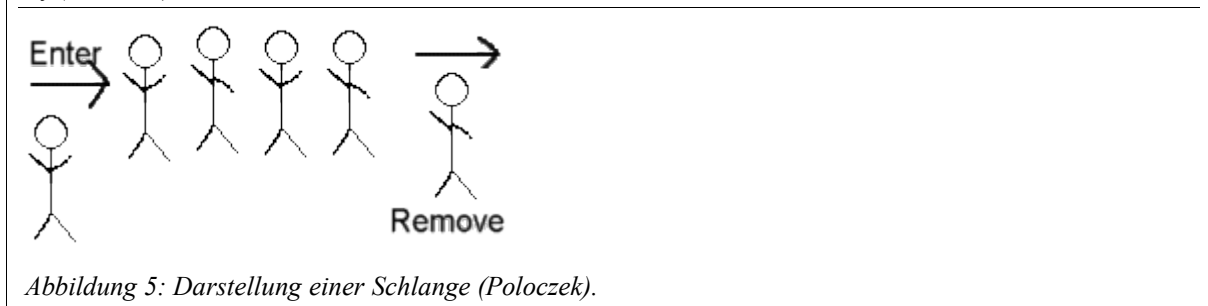


Abbildung 5: Darstellung einer Schlange (Poloczek).

Der unterrichtliche Weg zur Implementation basiert auf der Idee des Story-Driven-Modelling (SDM). Es ist eine noch in der Entwicklung befindliche Methodik zur Softwareentwicklung

mit Fujaba (Diethelm, Geiger, Zündorf 2004). Diese wurde hier im Hinblick auf die Funktion von Abbildungen zur anschaulichen Darstellung im Unterricht eingesetzt und dazu konkretisiert und angepasst. Die Idee des Zettel-Tests kommt von Ira Diethelm. Ich habe die genannten Autoren im Rahmen der Arbeit und der Lehrerfunktion Innovieren besucht, um dort Möglichkeiten der Nutzung und Anpassung des SDM im Unterricht zu besprechen. Ideen für eine formālere Umsetzung des SDM als die hier vorgeschlagene gibt es ebenfalls (Diethelm, Geiger, Zündorf 2002).

Im Einzelnen werden die folgenden Bausteine der von mir durchgeführten Unterrichtsreihe vorgestellt:

1. Objektdiagramme: Ein informeller Einstieg in die Fujaba-Diagramme
2. Zettel-Test: Ein spielerischer Weg zum Prüfen von Fujaba-Implementationen sowie als Hilfe, um die Semantik von Aktivitätsdiagrammen zu verstehen oder auch um Aktivitätsdiagramme einzuführen.
3. Dobs: Die interaktive Erkundung von ADT's
4. SDM und Story-Boarding: SDM beschreibt eine Vorgehensweise zur Implementation mit Fujaba. Greift auf Story-Boarding und Zettel-Test zurück.  
Beim Story-Boarding werden aus Objektspiel und Objektdiagrammen als einzeln durchgespielten Stories Aktivitätsdiagramme abgeleitet.

#### **4.1 Objektdiagramme: Einführung in den ADT Schlange**

Bereits in der Einstiegsstunde wird ein Objektdiagramm als Vorläufer für ein Aktivitätsdiagramm verwendet. Zunächst erarbeiten die Schülerinnen und Schüler, z.B. in Gruppen, anhand verschiedener Probleme die Idee der Warteschlange (Abbildung 6).

***Untersuchen Sie die folgenden Probleme und abstrahieren sie Ihre Lösungen.***

1. Mehrere reparaturbedürftige Autos stehen im Hof der Autowerkstatt
2. Mehrere Leute wollen an der Kinokasse Karten kaufen
3. Bezahlen an der Supermarktkasse
4. In der Postfiliale: Peter will einen Brief aufgeben, Paul möchte Briefmarken kaufen, Mary will Geld von ihrem Konto abbuchen.
5. Patienten im Wartezimmer

*Abbildung 6 Arbeitsauftrag zum Einstieg in das Thema Warteschlangen*

Im anschließenden Unterrichtsgespräch werden die Merkmale einer (Warte-)Schlange gesammelt, zusammengefasst und mit Hilfe der Lehrperson abstrahiert (Abbildung 7).

**Abstrakter Datentyp: (Warte-) Schlange**

1. Es entsteht eine Warteschlange
2. Reihenfolge muss eingehalten werden
3. Neue stellen sich hinten an (einschlängen)
4. der Erste wird bedient (ausschlängen)
5. die Anzahl der Wartenden ist nicht bekannt
6. Methoden einschlängen, ausschlängen

Abbildung 7 Mögliches Tafelbild als Ergebnis des Unterrichtsgesprächs zur Auswertung der Einstiegsaufgabe (Abbildung). Ergebnis aus dem Unterricht.

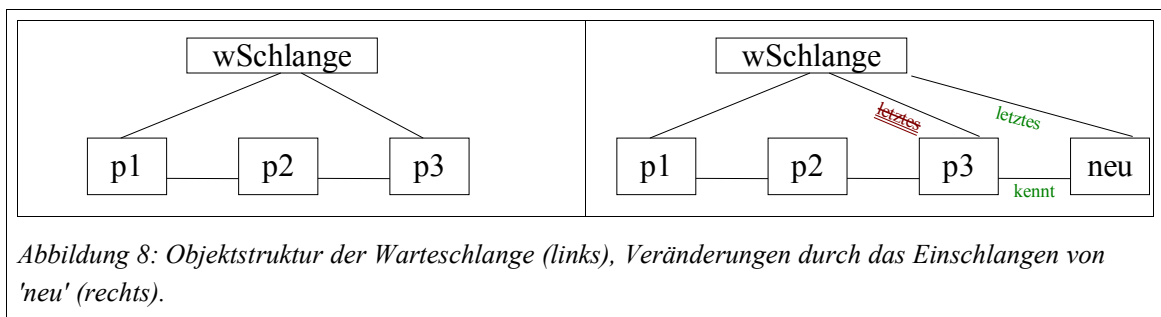
Die Schülerinnen und Schüler sollten die ersten vier Punkte (Abbildung 7) eigenständig in der Gruppenarbeit erarbeiten können. Die Überschrift, die Methodennamen und die weiteren Punkte werden im Auswertungsgespräch ergänzt.

In diesem ersten Schritt entsteht die Schlange durch eine einfache Objektverkettung (vgl. Abbildung 4 und 5).

Datenstruktur und Operationen werden nun im Objektspiel verdeutlicht. Dazu spielt ein Schüler das Objekt Schlange, welches die Methoden ausführt und den ein- bzw. ausgeschlangten Objekten sagt, was diese tun sollen. Ein weiterer Schüler übernimmt die Rolle des Benutzers, der die Schlange auffordert, etwas zu tun, also beispielsweise 'Peter' einzuschlangen.

Um die Ergebnisse des Objektspiels zu sichern, wird ein Objektdiagramm gezeichnet: Zunächst wird von einem Schüler auf einer Folie oder an der Tafel die entstandene Objektstruktur gezeichnet. Im nächsten Schritt werden die Veränderungen durch den Aufruf der Methode `einschlängen` eingezeichnet. Dazu soll ein Schüler zu entfernende Elemente rot und neu hinzukommende Elemente grün einzeichnen.

Im Anschluss an diese erste Visualisierung (Abbildung 8, linke Seite) können die Beziehungen benannt werden (Abbildung 8, rechte Seite), damit der Übergang bzw. die Abstraktion zum Klassendiagramm deutlicher wird. Als Hausaufgabe bietet es sich an, aus der Objektstruktur das zugehörige Klassendiagramm abzuleiten und/oder den Sonderfall der Methode `einschlängen` zu zeichnen: Wie sieht das `einschlängen` aus, wenn ein Objekt in eine leere Schlange hinzugefügt wird?

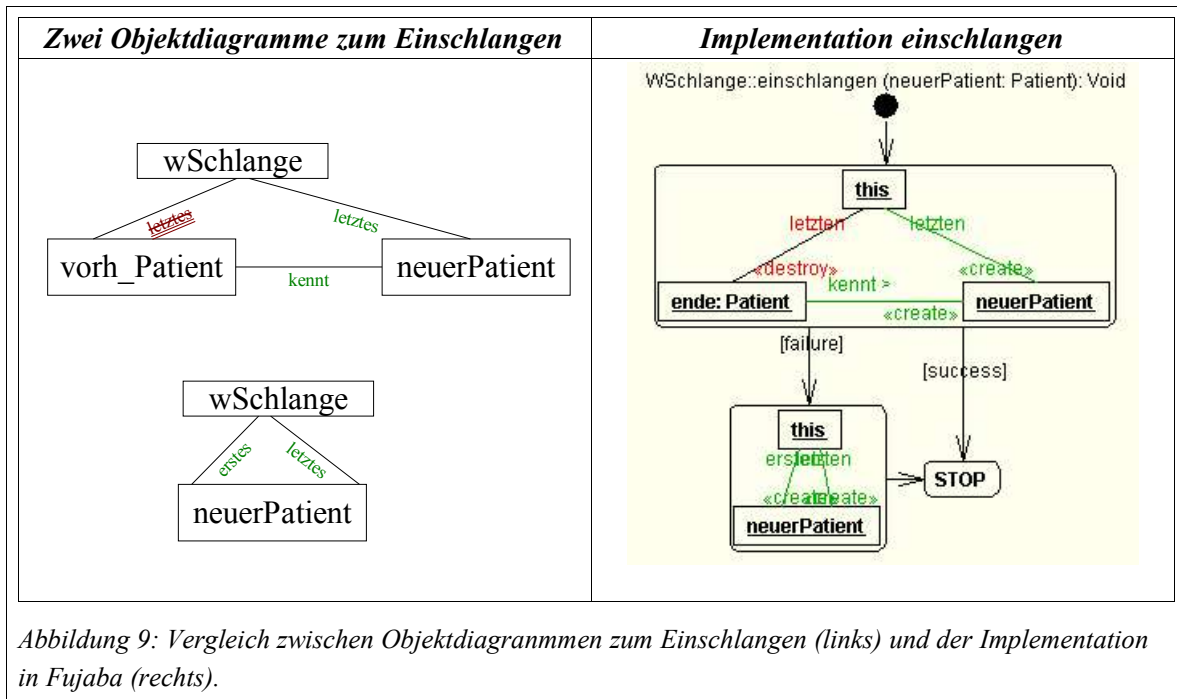


Anhand dieser beiden Informationen kann bereits in der folgenden Stunde / Doppelstunde die Implementation erfolgen.

Dazu werden nur die notwendigen Objekte gezeichnet, wobei in diesem Fall die Lehrperson auf die richtige Reihenfolge der Objektdiagramme achten sollte, damit aus der Zeichnung möglichst einfach die Implementation abgeleitet werden kann (Abbildung 9).

Dies kann im Unterrichtsgespräch geschehen, wobei dann eine Wiederholung der Fujaba-Semantik erfolgt (siehe Abschnitt 2.1, S. 4).

Die in Abbildung 9 dargestellte Implementation der Methode nutzt aus, dass Story-Pattern zunächst die Objektstruktur prüfen, bevor etwas geändert wird. Anhand dieser Darstellung kann daher gut die grundsätzliche Semantik erläutert werden.



Es bietet sich an, nun den Ablauf der Methode ausführlich mit dem Zettel-Test (Abschnitt 4.2, S. 15) durchzuspielen. Anschließend an das Einschlangen kann analog das Ausschlangen thematisiert werden, wobei hier die Eigenaktivität der Schülerinnen und Schüler erhöht werden kann.

Dass der Zettel-Test in der durchgeführten Reihe an dieser Stelle nicht durchgespielt wurde, führte zu den im Folgenden diskutierten Problemen bei der Implementation durch die Schülerinnen und Schüler, die ja eigentlich nur ein 'Abtippen' der fertigen Methode darstellen sollte.

### 4.1.1 Erfahrungen und Rückmeldungen der Schülerinnen und Schüler

Die Methode Objektspiel und die anschließende Sicherung mit Objektdiagrammen hat fast allen Schülerinnen und Schülern gut gefallen. Ihrer Meinung nach war der Unterricht motivierend und gut verständlich. Einige Schüler mit besserem Vorkenntnissen in Fujaba fanden das Vorgehen allerdings etwas zu ausführlich und kleinschrittig: „Weniger wäre mehr gewesen“. Ein skeptischer Schüler meinte: „Das Objektspiel hat Probleme verdeutlicht, die man vorher nicht bemerkt hatte (kennt Vorder- oder Hintermann). Das hatte ich nicht erwartet und habe es daraufhin doch als sinnvoll erachtet“.

Mehrere Schüler meinten, dass die Benutzung unterschiedlicher Farben an der Tafel sehr hilfreich gewesen sei.

Das Objektspiel zum ADT Schlange wurde auch auf dem NRW-Informatiktag als Workshop zu Fujaba durchgeführt. Auch hier funktionierte die Vorgehensweise gut, allerdings blieb bei den Lehrern die Skepsis, ob auch komplexere Probleme so bewältigt werden können – dazu braucht man vermutlich ein aufwändigeres Vorgehen, welches hier im folgenden Kapitel vorgestellt werden soll. Die Darstellungsweise war so eingängig, dass zu einem großen Teil mit Hilfe der Abbildungen über den ADT und dessen Implementation diskutiert wurde – die dazu notwendige Kenntnis der Syntax und Semantik wurde nebenbei geklärt.

Da die Ableitung der beiden Story-Pattern aus den Objektdiagrammen (Abbildung 9) im Unterrichtsgespräch flüssig und (scheinbar) ohne Verständnisprobleme gelang, sollten die Schülerinnen und Schüler nach dem Abtippen und Testen der Lösung in Fujaba eigenständig die Methode `ausschlangen` entwickeln und implementieren. Bereits bei der ersten Aufgabe, dem Abtippen, traten überraschenderweise viele Probleme auf:

- Mehrere Gruppen wollten das Parameter-Objekt (oben: `knoten`) mit der Begründung erzeugen, dass ja das Objekt neu in die Schlange aufgenommen werden und daher erzeugt werden müsse.
- Mehrere haben die Methoden irgendwo an Klassen angehängt bzw. wussten nicht, zu welcher Klasse die Methoden zugeordnet werden sollen.
- Viele haben die Bedeutung des `this`-Objekts nicht verstanden.
- Viele Gruppen wollten die `links`, deren Benennung eigentlich im Klassendiagramm festgelegt werden sollte (etwa: `kenntNachfolger`) in den einzelnen Story-Pattern verändern, um damit eine bestimmte Semantik auszudrücken.

Die Schülerinnen und Schüler waren an dieser Stelle offenbar überfordert. Es fehlte ein Verständnis des Zusammenhangs von Klassen und Objekten. Vor allem war unklar, wie überhaupt ein objektorientiertes Programm ausgeführt wird. Die Schülerinnen und Schüler betrachteten nur die Methode ohne deren Kontext: Klassendiagramm und Programmablauf und wussten daher nicht, wie genau sie die Methode aufbauen sollten.

Daher wurde die Arbeit am PC in der Stunde abgebrochen und stattdessen der Ablauf der Methode `einschlangen` ausführlich am Beispiel durchgespielt (siehe oben) und die Methode in Fujaba am Beamer gemeinsam erstellt. Anschließend konnten sie die ähnlich Methode `ausschlangen` eigenständig in ca. 15 Minuten erstellen.

Die Implementierung der Warteschlange wurde daraufhin auch von den Schülerinnen und Schülern als „sehr einfach“ empfunden, allerdings seien Fehler im Fujaba-Programm nur schwer zu finden. Mehrere äußerten sich ähnlich wie der folgende Schüler: „Die theoretische Arbeit macht mir mehr Spaß als die Arbeit am Rechner, da dort so viele Misserfolge lauern, jedoch ist ein gutes Ergebnis am Rechner auch fein“. Andererseits bemängelten zwei Schüler ein Zuviel an „trockener Theorie“ und möchten mehr am PC arbeiten.

Die Umsetzung am Rechner scheint überraschenderweise nur für einige Schüler zusätzliche Motivation zu bedeuten, die meisten erwarten eine reibungslose Implementation als Beweis für die Richtigkeit der vorher erarbeiteten Idee und sind frustriert bei Fehlern. Diese allerdings treten oft schon durch ungenaues 'Abtippen' der vorher erarbeiteten Lösungen auf (siehe oben etwa die Art der Benutzung von Parametern).

## 4.2 Der Zettel-Test

Dieser Test stellt schrittweise das Laufzeitverhalten der Methode dar. In Zukunft werde ich diesen Test als eigene Methode einführen und mit den Schülerinnen und Schülern üben, bevor sie selbst Methoden in Fujaba implementieren sollen.

Vermutlich war das Durchspielen auf Papier bzw. an der Tafel eine wesentliche Ursache dafür, dass die Schülerinnen und Schüler die zweite Methode viel schneller selbst als Aktivitätsdiagramm implementieren konnten als die erste.

Der Test ermöglicht außerdem das schnelle und einfache Aufspüren von Fehlern.

Während der Entwicklungsphase einer Methode können so bereits angefertigte Teile ausprobiert und (evtl. schrittweise) ergänzt werden. Der Zettel-Test ist damit ein Teil des Story-Driven-Modelling.

Aktivitätsdiagramme werden getestet, indem ihr Laufzeitverhalten schrittweise an einem oder mehreren Beispielen durchgespielt wird. Die Feinheiten im Ablauf sind nicht exakt festgelegt. Im Folgenden beschreibe ich eine Möglichkeit, ihn einzusetzen. Zunächst wird das Verfahren beschrieben, dann einige Überlegungen zum Einsatz und möglichen Varianten.

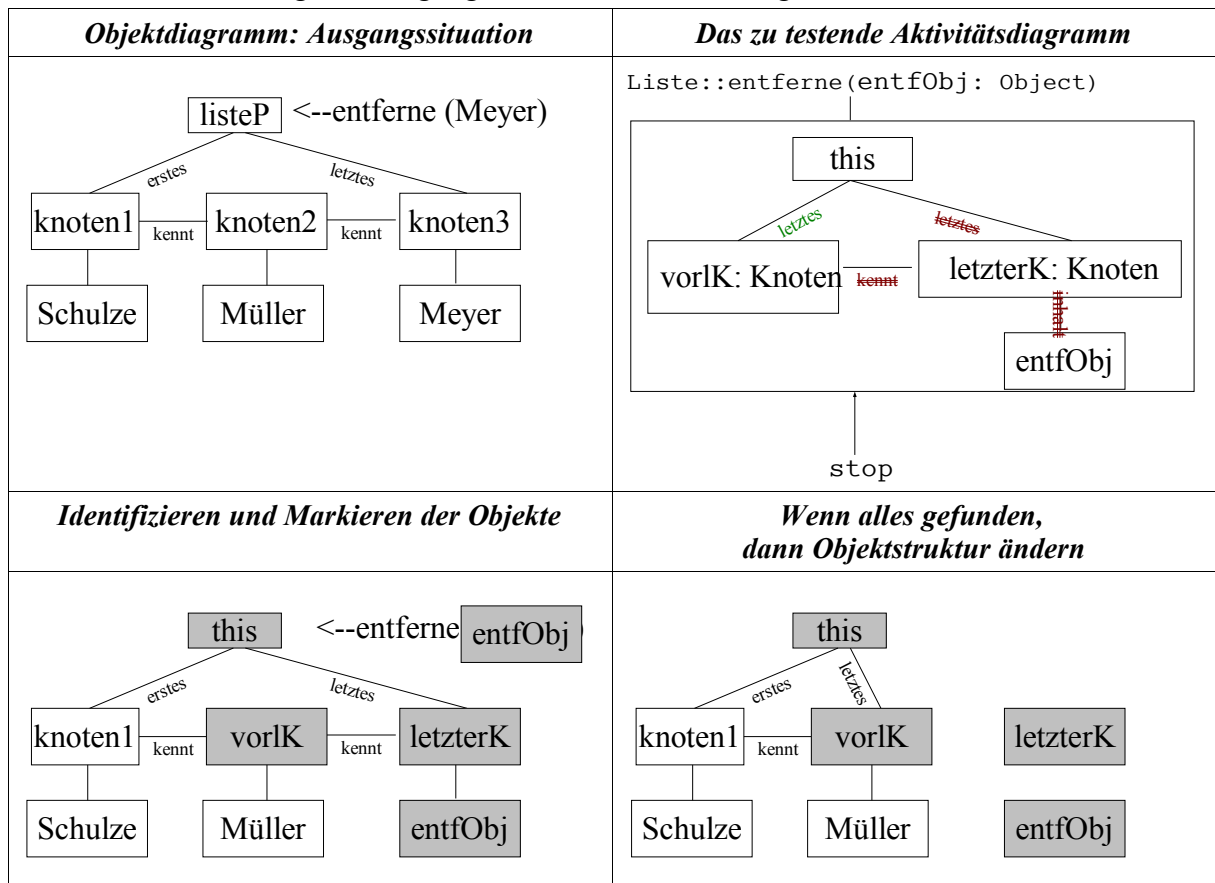


Tabelle 1: Ablauf des Zettel-Tests im Überblick (von oben links nach unten rechts): 1) Beispielszenario als Objektdiagramm darstellen. Anhand der Darstellung 2) der Methode 3) im Beispielszenario alle Objekte finden, die im zu prüfenden Story-Pattern angegeben sind: this, vorlK, letzterK, entfObj. In die Abbildung entsprechende Zettel kleben (Post-it o. ä. - hier als graue Kästchen). Bei Erfolg werden 4) die im Aktivitätsdiagramm vorgeschriebenen Änderungen im Beispiel durchgeführt. Man beachte, dass der Parameter (entfObj bzw. Meyer) berücksichtigt werden muss. Weitere Erläuterungen siehe Text.

Zunächst wird als Beispiel eine Objektstruktur aufgezeichnet (Tabelle 1, oben links) und die zu testende Methode daneben gelegt (Tabelle 1, oben rechts). Das kann beispielsweise an der Tafel geschehen. Dann wird im Objektdiagramm ein konkreter Methodenaufruf eingezeichnet (Tabelle 1, oben links): Auf dem Objekt `listeP` wird die Methode `entferne` mit dem aktuellen Parameter `Meyer` aufgerufen.

Dann werden anhand des Aktivitätsdiagramms die Objekte gebunden. An den formalen Parameter `entfObj` wird der aktuelle Parameter `Meyer` gebunden. Um diesen Prozess anschaulich darzustellen, werden die Zettel eingesetzt: Im Objektdiagramm wird im Methodenaufruf der Parameter `Meyer` mit dem formalen Parameter identifiziert, indem ein Zettel mit der Aufschrift `entfObj` darüber geheftet wird<sup>7</sup>. Da `listeP` das ausführende Objekt ist, wird es mit einem `this`-Zettel verdeckt.

Nach diesen beiden Schritten, Parameter und `this`-Objekt identifizieren, wird nun geprüft, ob im Beispiel die im Story-Pattern geforderte Objektstruktur gefunden werden kann: Ausgehend von einem gebundenen Objekt, etwa `this`, wird die im Aktivitätsdiagramm geforderte `erstes`-Beziehung zu einem Knoten-Objekt gesucht: im Objektdiagramm wird so das Objekt `knoten1` gefunden und mit dem entsprechenden Zettel beklebt: `vor1K`. Nach und nach sollten von bereits identifizierten (=den beklebten) Objekten ausgehend alle im Aktivitätsdiagramm angegebenen Objekte gebunden werden können. Wenn alle Objekte und Beziehungen identifiziert sind (Tabelle 1, unten links), dann werden die im Aktivitätsdiagramm beschriebenen Änderungen in das Objektdiagramm (Tabelle 1, unten rechts) eingezeichnet. An der Tafel kann das einfach mit Schwamm und bunter Kreide erfolgen.

Konnte der Zettel-Test erfolgreich durchgespielt werden, bietet es sich evtl. an, die geänderte Struktur für einen weiteren Durchlauf zu nutzen. Im obigen Beispiel könnte nun verdeutlicht werden, wieso das erste Objekt (`entferne(Schulze)`) mit dem Aktivitätsdiagramm nicht entfernt werden kann und was passiert, wenn die Liste nur aus einem Objekt besteht.

Der Zettel-Test zeigt, dass in Aktivitätsdiagrammen Variablen benutzt werden, die zur Laufzeit an bestehende Objekte gebunden werden. Dieser wichtige Aspekt kann anhand des Zettel-Tests angesprochen werden. Um die Semantik der Aktivitätsdiagramme zu verdeutlichen, dürfen etwaige Änderungen an der Objektstruktur erst nach dem vollständigen Binden aller Objekte und Beziehungen erfolgen.

Der Test kann an der Tafel oder mit den bekannten gelben Klebe-Zetteln auch auf Papier, etwa in Gruppenarbeit, durchgeführt werden. Er wird später bei der Entwicklung eigener Methoden mittels Story-Driven-Modelling benötigt (dazu mehr weiter unten, Abschnitt 4.4.4, S. 26).

### 4.2.1 Erfahrungen und Rückmeldungen der Schülerinnen und Schüler

Die Schülerinnen und Schüler, die Fujaba nicht kannten, waren der Meinung, dass dies eine gute Hilfe gewesen sei, um die Aktivitätsdiagramme besser zu verstehen. Diejenigen mit Fujaba-Erfahrung fanden gut, dass so die „Richtigkeit der Ergebnisse geprüft“ werden können bzw. ein sinnvoller Weg von der Idee zur Implementation deutlich wurde.

---

<sup>7</sup> Falls keine Magnettafel vorhanden ist, kann dazu eine spezielle Folie aus dem Schreibwarenhandel verwendet werden, die an der Tafel klebt, rückstandsfrei entfernt werden kann und mit Boardmarkern wiederbeschreibbar ist.

Ein Schüler meinte, dieses Vorgehen sei zwar „im Nachhinein verständlich, besser wäre jedoch gewesen, vorher die wichtigsten Schritte beim Implementieren zu erklären und dann zu zeigen“.

Bei der Durchführung ist es sinnvoll, den Unterschied zwischen der linken Seite als Beispiel bzw. Laufzeit-Szenario und der rechten Seite als grafischer Quelltext durch entsprechende Benennung zu betonen (siehe oben).

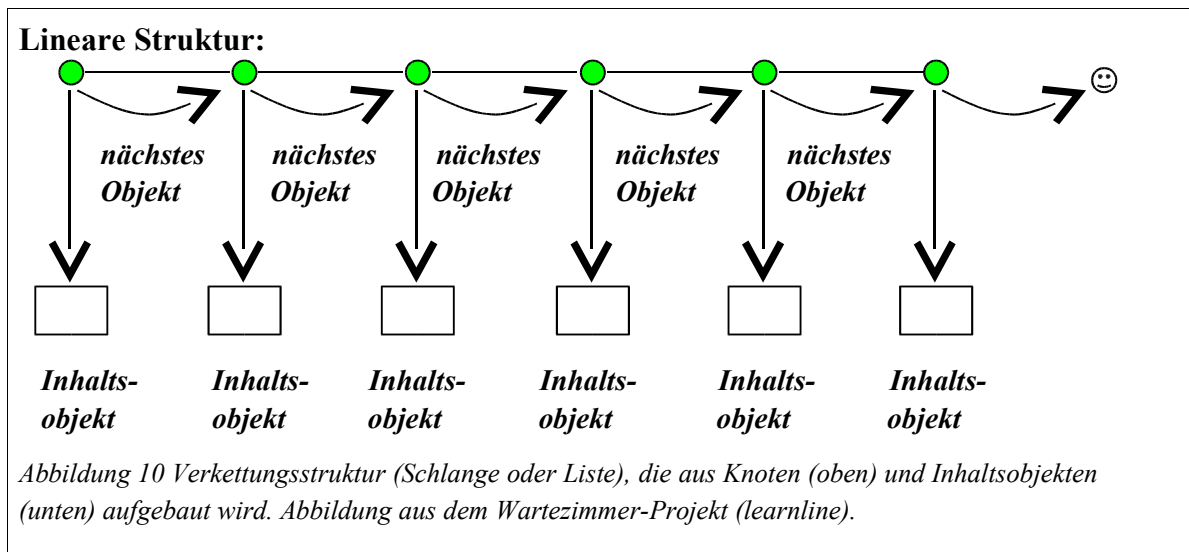
Der Einsatz des Zettel-Tests kann unterschiedlichen Zwecken dienen: Um das Laufzeitverhalten bzw. die Semantik von Story-Pattern zu untersuchen oder die Richtigkeit der Implementation zu prüfen. Neben der beschriebenen Vorgehensweise kann der Test auch als Gruppen-, Partner- oder in Einzelarbeit durchgeführt werden.

Wenn die Methode bekannt ist, kann der Test auch informal anhand eines schnell skizzierten Objektdiagramms ohne den Einsatz der Zettel mit Kreide und Schwamm durchgeführt werden. Im Unterricht beispielsweise konnte so die Tauglichkeit umstrittener Lösungsideen, die in Gruppenarbeit entstanden sind, schnell und einfach im Unterrichtsgespräch geklärt werden.

Das schrittweise Vorgehen des Zettel-Tests klingt hier in der Beschreibung vermutlich langwierig und wenig motivierend, aber die Anwendung im Unterricht erzeugte jedes Mal eine aufmerksame Spannung, weil alle den Ablauf mitverfolgten und überlegten, ob es nun klappt bzw. an welcher Stelle ein Fehler auftreten könnte. Falls tatsächlich Fehler im Aktivitätsdiagramm erkennbar wurden, meldeten sich oft die Schülerinnen und Schüler sofort mit einem Verbesserungsvorschlag. Das liegt vermutlich auch daran, dass der Test jeweils dann durchgeführt wurde, wenn eine Lösung entworfen und geprüft werden sollte oder wenn Vorschläge (der Schülerinnen und Schüler oder des Lehrers) unklar waren.

### ***4.3 Dobs als Visualisierungshilfe: Knotenobjekte***

Im vorangegangenen Baustein besteht die Schlange aus einer Verkettung von Patienten. Diese Verkettungsstruktur soll nun weiter abstrahiert werden, sodass der ADT Schlange aus Knotenobjekten besteht, die Inhaltsobjekte verwalten. Diese Inhaltsobjekte werden in die Schlange eingefügt, ohne selbst etwas über die Datenstruktur zu wissen, in der sie sich befinden.

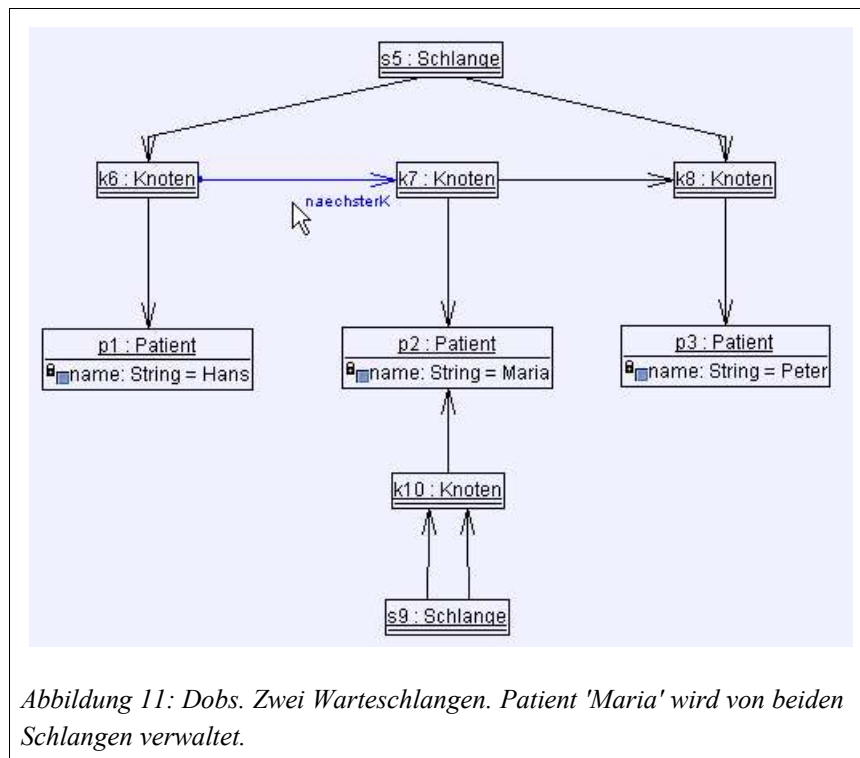


Die anschauliche Erklärung dieser Darstellung (Abbildung 10) ist die der Wäscheleine, an „deren Klammern die Verbindungen zu beliebigen Objekten – hier: Patienten – festgemacht werden bzw. wieder entfernt werden können“ (learnline).

In der Unterrichtsreihe zum Wartezimmer-Projekt (learnline) wird dieser Schritt wie folgt begründet:

„In einer komplexeren Arztpraxis ist ein Patient aber nicht nur in der Wartezimmer-Schlange, sondern zum Beispiel auch in der Schlange, die alle Patienten erfasst, deren Blutdruck noch vor der Arztbehandlung gemessen werden soll oder in der Schlange, in die alle EKG-Patienten der Reihe nach eingefügt werden. Da es wenig sinnvoll ist, denselben Patienten mehrfach anzulegen, werden lineare Strukturen entwickelt, die lediglich den Verweis auf ein beliebiges Objekt in der Verkettungsstruktur ablegen, so dass ein und dasselbe Objekt in mehreren Strukturen Berücksichtigung finden kann.“

Aus fachlicher Sicht ist ein weiterer Vorteil hinzuzufügen: Bei der vorangegangenen Verkettung von Patienten sind die in der Liste abgelegten Inhaltsobjekte (die Patienten) für die Verwaltung der Reihenfolge zuständig. Zusätzlich zu den Aufgaben eines Patienten übernehmen sie Aufgaben zur Verwaltung der Datenstruktur – sobald die Datenstruktur im Programm geändert wird, müsste auch die Klasse Patient geändert werden. Damit werden Aufgabenbereiche vermischt. Durch die Einführung von Knotenobjekten werden diese Bereiche getrennt.



Die entsprechende Implementation der Schlange soll im Unterricht mit Hilfe der Veranschaulichung in Dobs eingeführt werden, indem die Schülerinnen und Schüler die Schlange erkunden. Sie sollen nur mit der gegebenen Implementation (in Dobs) die Aufgabe lösen, einige Patienten nur auf den Arzt, andere aber auf den Arzt und die Blutdruckuntersuchung warten zu lassen. Anschließend sollen sie die Unterschiede zwischen dieser Implementation der Schlange und der vorangegangenen Verkettung der Patienten (die im Unterricht WSchlange genannt wurde) herausstellen.

Abbildung 12 stellt das Ergebnis der Arbeitsphase dar:

1. Patienten wissen nicht (mehr), ob und wo in der Schlange sie sich befinden.
  2. Keine Vermischung der Aufgabenbereiche: Objekte, welche die Datenstruktur verwalten (Schlange, Knoten), sind klar getrennt von den Objekten, die verwaltet werden (Patienten).
  3. Als Folge kann man Patienten in mehreren Schlangen verwalten. Man kann z.B. das Wartezimmer-Programm um weitere Warteschlangen erweitern, ohne die Klasse Patient ändern zu müssen.
  4. (Optional) Man kann diese Implementation der Schlange einfach verwenden, ohne sie ändern zu müssen, da beliebige Objekte verwaltet werden können.
  5. Zusammenfassung: Ein abstrakter Datentyp stellt Methoden zur Verwaltung von Daten bzw. Objekten bereit. Eine Schlange hat zwei Methoden: `einschlangen` und `ausschlangen`. Sie garantiert das FIFO-Prinzip: First-in, first-out. Mögliche weitere Methoden sind: `Schlange leer` und `laenge`.
- Abbildung 12 Auswertung und Sicherung der Erkundungsphase in Dobs

### 4.3.1 Erfahrungen und Rückmeldungen der Schülerinnen und Schüler

Die meisten Schülergruppen wunderten sich zunächst über die zusätzlichen Knotenobjekte, die für das bekannte Ein- und Ausschlagen bislang nicht benötigt wurden. Durch den Versuch, einen Patienten in zwei Schlangen zu verwalten, wurde deren Funktion deutlich.

Bei der Bearbeitung fiel auf, dass nicht alle Gruppen die Objektstrukturen in Dobs grafisch so angeordnet hatten, dass die Struktur anschaulich wurde. Hier wurde einigen ein entsprechender Tipp gegeben. Im anschließenden Unterrichtsgespräch konnten die Schüler ihre Beobachtungen detailliert beschreiben: „Jede Warteschlange benutzt Knoten, die als Stellvertreter für die Patienten die Reihenfolge festlegen. Die Patienten selbst kennen ihre Nachfolger nicht.“ Ein weiterer Schüler ergänzte: „Jede Schlange hat ihre eigenen Knoten, die Patienten kennen. Aber die Patienten kennen die Knoten nicht. Daher kann ein Patient zugleich von zwei Schlangen verwaltet werden.“ Die Trennung von Datenstruktur und Inhaltsobjekten musste im Unterrichtsgespräch dann noch verdeutlicht werden.

Zur Bedienung in Dobs: Insgesamt ist Dobs hilfreich und konnte so eingesetzt werden, dass die Schülergruppen selbst Konzepte entdecken. Allerdings waren die Schülerinnen und Schüler ohne Vorkenntnisse nicht zufrieden, da Dobs nicht immer so reagiert, wie man es erwarten würde. Notwendig sind Bedienungshilfen für die folgenden Probleme: Nach Methodenaufrufen sollte man die beteiligten Objekte neu expandieren, damit tatsächlich alle geänderten Beziehungspfeile auch dargestellt werden. Ein weiterer Nachteil ist, dass nicht benötigte Knotenobjekte nicht automatisch entfernt werden, diese sollten in Dobs per Hand gelöscht werden. Gegebenenfalls kann eine kurze Erklärung erfolgen<sup>8</sup>.

### 4.4 Story-Driven-Modelling: Der ADT Liste

Im Unterschied zum ADT Schlange können in eine Liste Objekte an beliebiger Stelle eingefügt oder entfernt werden. Ausgehend vom Projekt Wartezimmer wird im Unterricht der ADT Liste als Erweiterung der Schlange eingeführt<sup>9</sup>. Die Erweiterung wird motiviert durch den Fall, dass jemand das Wartezimmer vorzeitig verlässt; sei es, weil der Patient nicht länger warten möchte und die Praxis verlässt oder weil ein Notfall vorliegt. Aufgabe ist die Erweiterung der Datenstruktur, sodass ein Patient entfernt werden kann ohne dabei die verbleibende Struktur zu verändern.

Die Schlange soll mit dem Verfahren Story-Driven-Modelling erarbeitet und implementiert werden: Die verschiedenen Fälle werden, ggf. mittels Objektspiel, bestimmt. Mit dem Story-Boarding werden Story-Pattern entwickelt, die dann schrittweise in die Implementation überführt werden.

Am Beispiel 'Entfernen eines Objekts an beliebiger Stelle' werden die einzelnen Schritte dargestellt.

---

<sup>8</sup> In Java löscht das Laufzeitsystem (der so genannte 'garbage collector') eigenständig Objekte, wenn diese nicht mehr benötigt werden und(!) der Speicherplatz knapp wird. Daher ist der Zeitpunkt des Löschens nicht vorhersagbar – vor allem erfolgt er sehr wahrscheinlich nicht direkt nach dem Methodenaufruf. Zum anderen werden nur Objekte gelöscht, die nicht mehr benötigt werden: Das sind diejenigen, die vom Programm nicht mehr erreichbar sind, also keine Beziehung zu anderen Objekten mehr haben (vgl. auch: Ullenboom, S. 281f). Da aber in Dobs alle Objekte von Dobs gekannt werden, werden nach Meinung des garbage collectors alle Objekte weiterhin benötigt – und die Fujaba-Entwickler haben bislang keinen eigenen garbage collector implementiert, der auch die Objekte freigibt, die nur noch von Dobs gekannt werden.

Daher werden immer nur die Beziehungen zwischen den Objekten, aber nicht die Objekte selbst gelöscht.

<sup>9</sup> Vgl. das Vorgehen aus dem Wartezimmer-Projekt (learnline).

<b><i>Ablauf des Story-Driven-Modelling: Implementation einer Methode</i></b>	<b><i>Vereinfachte Variante</i></b>
1) Ein Objektdiagramm beschreibt die Ausgangssituation	1) Fälle stichwortartig sammeln
2) In einer Kopie dieses Diagramms wird der Methodenaufruf eingezeichnet	
3) In weiteren Diagrammen werden wesentliche Schritte des Ablaufs dokumentiert	
4) Ein abschließendes Diagramm stellt die jeweilige Endsituation dar	
-> Klassendiagramm ergänzen	
5) Aus den Objektdiagrammen werden die entsprechenden Aktivitätsdiagramme abgeleitet. (Story-Boarding)	2) Story-Boarding nur für unbekanntes Fall 2a) evtl.: Klassendiagramm ergänzen 2b) evtl. Zettel-Test
-> Schritte 1-5 für alle verschiedenen Situationen wiederholen	
6) Die verschiedenen Aktivitätsdiagramme zu einer Methode zusammenführen	3) Zusammenfassen der Fälle zu einem Aktivitätsdiagramm
7) Das Ergebnis wird anhand der einzelnen Ausgangssituationen getestet. (evtl. Zettel-Test)	4) Zettel-Test

*Tabelle 2: Story-Driven-Modelling im Überblick (links). Rechte Seite: vereinfachte Variante aus dem Unterricht.*

Erläuterungen zu den einzelnen Schritten der ausführlichen Version Story-Driven-Modelling (vgl. Diethlem, Geiger, Zündorf 2002 sowie 2004), Tabelle 2, linke Seite:

1. Zusätzlich zum Objektdiagramm werden Bedingungen beschrieben: z.B.: Das letzte Objekt der Liste (Peter) wird gelöscht; es wird versucht ein Element zu löschen, das nicht in der Liste einhalten ist, ...
2. Durch das Einzeichnen des Aufrufs wird das Objekt bestimmt, das für die Methode zuständig ist (es wird später das this-Objekt werden)
3. Mehrere Objektdiagramme: Manchmal kann der Ablauf in einem Schritt erfolgen und dieser Teil der Methode besteht aus nur einem Story-Pattern, manchmal ist der Vorgang zu komplex, sodass man ihn besser in Teilschritte aufteilt.
4. Dieses Diagramm wird benötigt, um die spätere Methodenausführung mit der Intention vergleichen zu können.
5. Beim Erstellen der Aktivitätsdiagramme können Namensgebungen / Neuerungen notwendig werden, die das Klassendiagramm betreffen, das entsprechend erstellt bzw. ergänzt werden sollte.
6. Beim Zusammenführen der einzelnen Aktivitätsdiagramme muss auf den korrekten Kontrollfluss geachtet werden.

7. Zum Testen werden die Objektdiagramme aus Schritt 1 herangezogen und anhand des Aktivitätsdiagramms aus Schritt 6 durchgespielt. Das Ergebnis wird mit den Diagrammen aus Schritt 4 verglichen.

Im Unterricht wurde das Verfahren etwas vereinfacht. Denn da vorher der ADT Schlange im Unterricht behandelt wurde, waren Klassendiagramm sowie die zu beachtenden Fälle beim Entwickeln der Methoden im Grund bekannt und konnten übertragen werden. Im Folgenden wird dieses Verfahren an zwei Fällen vorgestellt.

#### 4.4.1 Fälle finden

Für die Methode `entfernen` sind insgesamt vier Fälle zu unterscheiden (vgl. die entsprechenden 'Zeilen' in Abbildung 14):

1. Die Liste ist leer -> keine Aktion erforderlich. (Erste Zeile)
2. Der erste Patient wird entfernt -> Implementation bereits bekannt: `entfernenKopf` (Zweite Zeile)
3. Der letzte Patient wird entfernt -> Implementation direkt umsetzbar (Dritte Zeile)
4. Der Patient befindet sich in der Mitte der Liste -> Implementation nicht direkt umsetzbar, da eine Schleifenstruktur erforderlich ist. (vierte Zeile)

Das mögliche implementierte Ergebnis, welches die vier Fälle deutlich macht, ist in Abbildung 14 dargestellt.

#### 4.4.2 Story-Boarding

Im Story-Boarding werden verschiedene Beispiele schrittweise (als Objektspiel) durchgespielt und mit Objektdiagrammen dargestellt.

Story-Boarding funktioniert für einfache Objektstrukturen sehr gut, da diese fast direkt in Fujaba-Aktivitätsdiagramme übernommen werden können (vgl. oben Abbildung 9, S. 13). Falls jedoch Suchprozesse auf der Objektstruktur erforderlich sind, die zu Schleifenstrukturen in der Implementation führen, ist diese direkte Umsetzung nicht mehr gegeben. Im Folgenden werden beide Varianten erläutert.

Der ausführliche Ablauf des Story-Boarding wird in der folgenden Tabelle für den Fall 'entferne das letzte Objekt' vorgestellt.

<b>Story-Boarding: Vom Objekt- zum Aktivitätsdiagramm</b>	
<b>Diagramme</b>	<b>Kommentar</b>
<p>Ausgangssituation</p>	<p>Zunächst wird ein Beispiel angelegt und als Objektdiagramm gezeichnet. Falls das Klassendiagramm noch nicht eindeutig ist, wird es anhand der Objektbeziehungen und Objekttypen bestimmt. Beispielsweise: <u>Patienten kennen sich untereinander</u>, ...</p>
<p>Konkreter Methodenaufruf</p>	<p>Im zweiten Schritt wird durch das Einzeichnen des Methodenaufrufs festgelegt, welches Objekt für die Ausführung verantwortlich sein soll: hier <code>listeP</code>. Im Unterricht wird man hier ggf. den Namen des entsprechenden Schülers aus dem Objektspiel notieren.</p>
<p>Schritt 1: Ausführung (ggf. weitere Schritte)</p>	<p>Über die Kante <code>letztes</code> kann <code>listeP</code> direkt den Knoten finden, der das Objekt <code>Meyer</code> verwaltet und dementsprechend die Änderungen an der Objektstruktur vornehmen. Diese werden eingezeichnet. (Bei komplexeren Methoden sollten weitere Schritte als Objektdiagramm gesichert werden) Eventuell kann der Endzustand zur Sicherung als ein weiteres Diagramm dargestellt werden. Im Unterricht werden die Änderungen in der Regel farblich markiert und an der Tafel schrittweise erweitert – und daher mit nur einem Diagramm gearbeitet.</p>
<p>Das resultierende Aktivitätsdiagramm:</p> <pre>Liste::entferne(entfObj: Object):Object</pre>	<p>Nun kann das Aktivitätsdiagramm abgeleitet werden, indem der Aufruf übernommen, der minimale Kontext bestimmt und jedem Objekt ein sinnvoller Name zugeordnet wird. Zum minimalen Kontext gehören die Objekte, die verändert werden oder deren links geändert werden. Anschließend kann dann der Zettel-Test durchgeführt werden.</p>

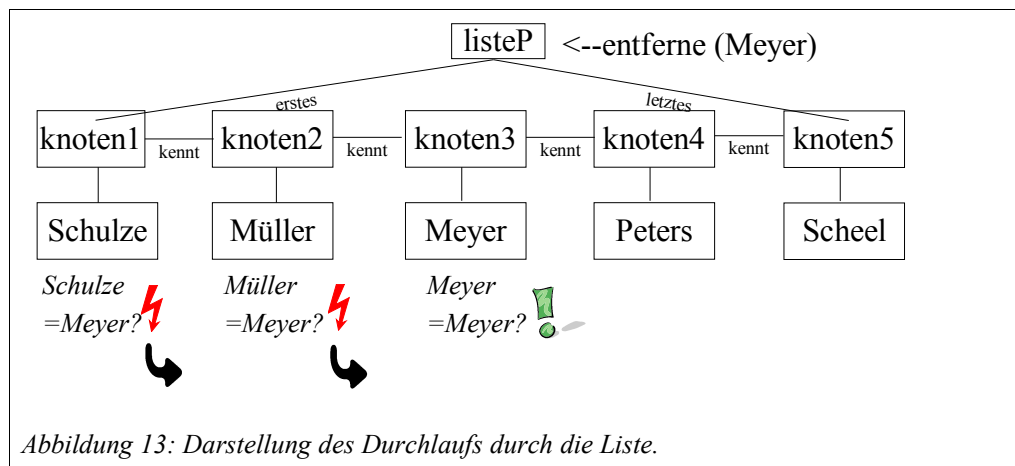
Tabelle 3: Entwicklung eines Teils der Methode `entferne` aus den Darstellungen eines Beispielablaufs, in dem das letzte Objekt der Liste entfernt wird. Vgl. dazu die Implementation in Abbildung 14, insbesondere die dritte Zeile.

Mit dem in Tabelle 3 geschilderten Ablauf ist der Kern des Story-Boarding beschrieben. Für die vollständige Entwicklung der Methode sind jedoch weitere Schritte notwendig. Zunächst wird das entstandene Aktivitätsdiagramm anhand des Ausgangsszenarios getestet. Falls das funktioniert und die gewünschte Endsituation erreicht wird, folgt der Test mit weiteren Szenarien. Für die weiteren Szenarien / Fälle werden weitere Aktivitätsdiagramme erzeugt, die anschließend zu einem gemeinsamen Diagramm zusammengefasst werden, welches die Methode für alle Sonderfälle implementiert. Diese Arbeit kann bei entsprechender Erfahrung der Lerngruppe arbeitsteilig geschehen.

Im Unterricht kann das SDM zunächst informal eingesetzt werden – etwa mit dem Ziel, anhand der Darstellungen die Semantik von Methodenabläufen zu erarbeiten (siehe Abschnitt 4.1). Später wird dann auf dieser Einsatz (sozusagen spiralförmig) an einem komplexeren Beispiel aufgegriffen, wobei den Schülerinnen und Schülern verdeutlicht wird, dass sie hier ein Verfahren kennen lernen, welches sie später (in Gruppen) eigenständig anwenden sollen.

Im Folgenden wird der letzte Fall, das Löschen eines Objekts aus der Mitte der Liste, an einem weiteren Szenario (vgl. Abbildung 13) vorgestellt, um das Ableiten einer Schleifenstruktur kurz zu erläutern.

Das Löschen von Müller, Meyer oder Peters scheint zunächst ebenso einfach zu sein: das zu löschende Objekt entfernen und die Lücke in der Verkettungsstruktur schließen. Doch das setzt voraus, dass die Position des Objekts bekannt ist. Daher ist jeder der drei Fälle unterschiedlich: das zweite, dritte oder das vierte Element der Liste soll gelöscht werden. Für eine allgemeine Fassung des Ablaufs wird ein schrittweiser Durchlauf durch die Liste benötigt. Dazu wird im Objektdiagramm dieser schrittweise Durchlauf eingezeichnet (Abbildung 13) und in eine Schleifenstruktur im Aktivitätsdiagramm überführt. Aus diesem Ablauf wird dann das Aktivitätsdiagramm für den entsprechenden Fall abgeleitet (vgl. dazu die untere 'Zeile' in Abbildung 14, S. 25).



Anschließend werden die einzelnen Aktivitätsdiagramme zu einer Methode zusammengeführt und anschließend das Vorgehen reflektiert und gesichert, indem es analog zur Tabelle 2 (S.21) notiert wird.

#### 4.4.3 Zusammenführen der Einzelteile zu einem Aktivitätsdiagramm

Um aus den einzelnen Aktivitätsdiagrammen für die einzelnen Fälle ein gemeinsames Diagramm zu erstellen, werden die einzelnen Story-Pattern in eine geeignete Reihenfolge gebracht

und mit Transitionen verbunden. Dazu müssen gegebenenfalls die Namen der Objekte in den einzelnen Story-Pattern angepasst sowie weitere kleinere Änderungen gemacht werden.

Ansonsten können die Aktivitätsdiagramme einzelner Fälle meist einfach untereinander angeordnet werden. Gegebenenfalls werden Story-Pattern nötig, um die Fälle zu unterscheiden.

Im Unterricht kann das Zusammenführen der einzelnen Aktivitätsdiagramme in Form einer Puzzle-Aufgabe von den Schülerinnen und Schülern bearbeitet werden. Dazu bekommen sie auf einem Blatt Papier die (fast) fertige Methode als Puzzle (Abbildung 15, S.34) mit der Aufgabe, daraus die Methode zu erstellen, indem sie die Puzzleteile mit Hilfe von Verbindungspfeilen sowie eventueller success/failure-Beschriftung in eine sinnvolle Reihenfolge zu bringen.

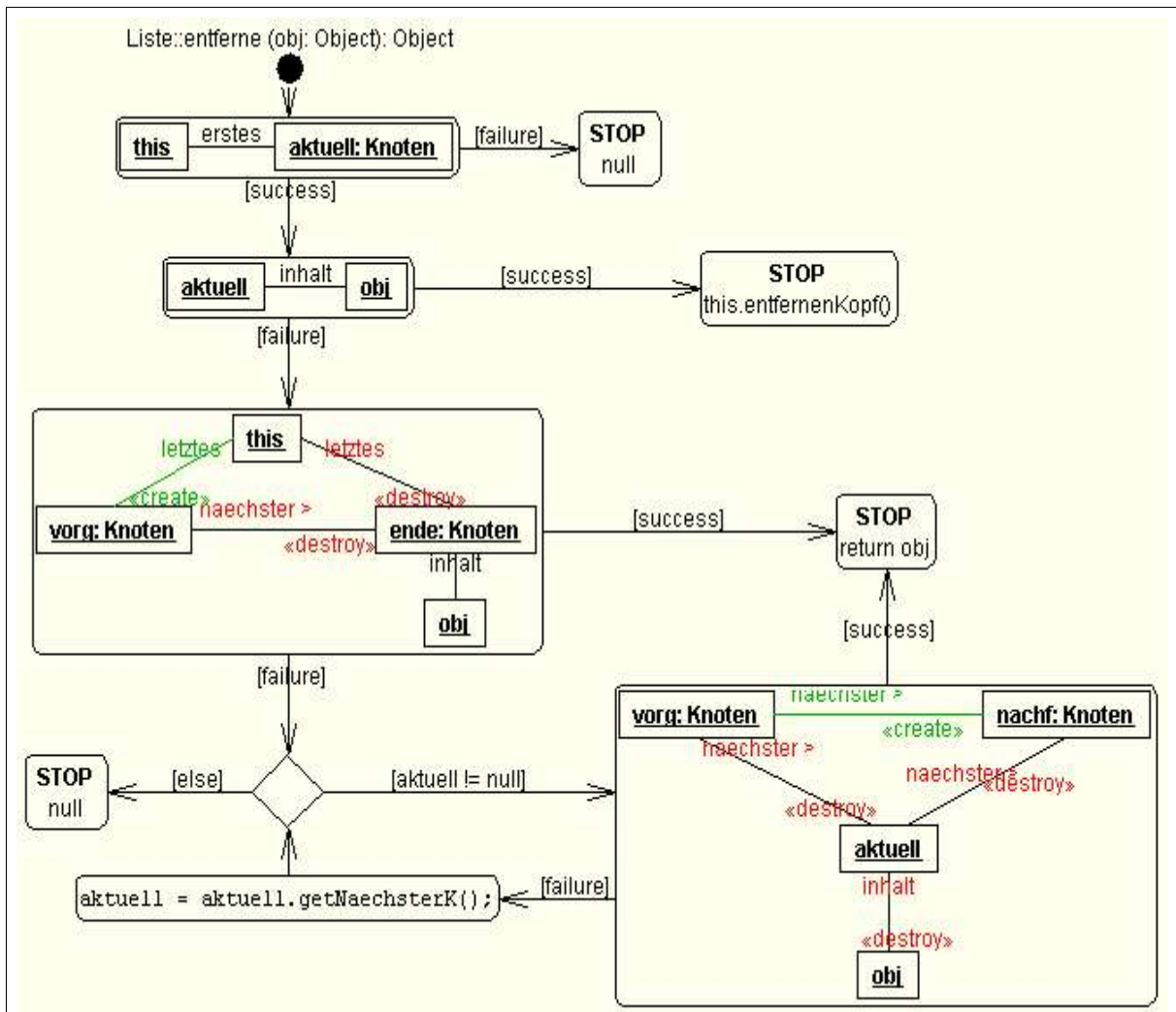


Abbildung 14: Entfernen aus einer Liste. Die erste 'Zeile' prüft, ob die Liste leer ist. In den beiden folgenden 'Zeilen' werden die Fälle am Anfang und am Ende löschen behandelt. Im unteren Abschnitt wird die Position des zu löschenden Objekts in einer Schleife gesucht.

Alternativ kann das direkt in Fujaba geschehen, indem aus der Implementation die Kanten entfernt und die Story-Pattern verschoben werden (vgl. Abbildung 15, S. 34); es bietet sich allerdings auch die Arbeit mit Papier und Schere an. Sei es als einfacher Methodenwechsel,

um eine zu schnelle Konzentration auf das Werkzeug zu vermeiden oder einfach als methodische Variation.

In Abbildung 14 ist das zusammengeführte Aktivitätsdiagramm dargestellt. Im ersten Story-Pattern wird geprüft, ob die Liste leer ist. Falls nicht, wird in der zweiten Zeile geprüft, ob das im vorangegangenen Story-Pattern erfolgreich gebundene Knoten-Objekt das zu löschende Objekt verwaltet. Das heißt, diese beiden Fälle sind beim Zusammenführen verändert worden. Der Name `aktuell` ist gewählt worden, damit das Objekt in der unteren Schleifenstruktur benutzt werden kann.

Als Anwendung bzw. Übung kann anschließend z.B. in Gruppenarbeit die Methode `einfügenVor` und/oder `einfügenNach` mit diesem Verfahren entworfen und implementiert werden. Dabei zeigt sich, ob die Methode und die Vorgehensweise des SDM verstanden werden.

#### **4.4.4 Erfahrungen und Rückmeldungen der Schülerinnen und Schüler**

Nach einer kurzen Einführung des ADT Liste über die Notwendigkeit, an beliebigen Stellen Patienten entfernen zu müssen, wurde eine angepasste Version des Story-Driven-Modelling im restlichen Verlauf einer Doppelstunde eingeführt: Für die Methode entfernen haben die Schülerinnen und Schüler zunächst die verschiedenen Fälle bestimmt (vgl. Abschnitt 4.4.1, S. 22). Anschließend wurde im Unterrichtsgespräch der Fall vier 'Patient befindet sich in der Mitte der Liste' mit dem Story-Boarding in ein Aktivitätsdiagramm überführt und anschließend mit dem Zettel-Test überprüft. Die weiteren Fälle konnten dann die Schülerinnen und Schüler, zum Teil als Hausaufgabe, in Gruppenarbeit lösen.

Die Schülerinnen und Schüler fanden die Doppelstunde aus verschiedenen Gründen motivierend: Zum einen weil einige die (in ihren Augen 'theoretische') Arbeit ohne den Computer bevorzugen. Zudem empfanden sie die Möglichkeit gut, einen schrittweise entstehenden Tafelanschrieb mitzuschreiben, da die Mitschrift 'informativer' sei und man daran besser wisse, was man selbst machen müsse. Einige haben auch an ihre Mitschrift zusätzliche Kommentare angefügt, etwa: 'wird später zum this-Objekt'.

Gut fanden die Schülerinnen und Schüler auch, dass die Ableitung des Aktivitätsdiagramms sofort und ohne die Eingabe in Fujaba geprüft werden kann. Man könne so Fehler schneller und einfacher finden. Ansonsten sitze man vor Fujaba und wisse nicht, wo der Fehler liege.

Das Entwickeln des Aktivitätsdiagramms zu diesem Fall hat zusammen mit dem Zettel-Test insgesamt 45 Minuten gedauert. Eine ganze Stunde Unterrichtsgespräch ist m.E. zu lang und war auch augenscheinlich für die Schülerinnen und Schüler anstrengend. Auch wenn diese Gruppe die ganze Zeit über aufgepasst hat, bietet es sich an, zwischendurch einen Methoden- und Sozialformwechsel einzuplanen: Demnächst würde ich, wie in der Gliederung der Arbeit angedeutet, den Zettel-Test bereits vorher als eine eigenständige Test-Methode einführen. Diese kann dann in dieser Stunde von den Schülerinnen und Schülern eigenständig angewendet werden. Evtl. in Gruppen, die parallel verschiedene Szenarien testen.

Zudem kann die Schleifenstruktur vorher erarbeitet werden oder anhand eines Beispiels einfach vorgegeben werden. Ich hatte sie an der Tafel schrittweise entwickelt, das war etwas umständlich. Man könnte bspw. den Schülerinnen und Schülern ein Blatt vorgeben, auf dem die Grundstruktur der Schleife zu sehen ist, die dann für den konkreten Fall angepasst wird.

Andererseits ergeben sich beim Arbeiten an der Tafel viele Gelegenheiten, unklare Aspekte der Syntax und Semantik am Beispiel zu besprechen.

Nun zu den weiteren Schritten des SDM, das ja mehr als das Story-Boarding umfasst:

Für die anschließende Arbeit in Gruppen, die die weiteren Fälle eigenständig lösen und später eine eigene Methode eigenständig entwickeln, ist es sinnvoll, den Methodenkopf, insbesondere die Parameter, vorher festzulegen. Dieses sollte im Klassendiagramm erfolgen, das an geeigneter Stelle erarbeitet werden sollte. Ein möglicher Zeitpunkt dafür wäre die Stelle, an der das Problem verdeutlicht ist und bevor die Gruppen zu arbeiten beginnen. Hier kann ein erster Entwurf erfolgen und die Methodennamen bestimmt werden. Während der Gruppenarbeit sollten dann aber Änderungen am Klassendiagramm erlaubt sein, etwa um zusätzlich benötigte Beziehungen oder Hilfsmethoden einzufügen.

Das spätere Zusammenfügen der einzelnen Fälle zu einer Methode gelingt dann recht flüssig. Voraussetzung ist aber eine recht genaue Kenntnis der Funktionsweise der einzelnen Teile. Dazu ist ein Überblick über die Schritte des SDM nützlich. Dieser kann als eine Art Methodenreflexion nach der einführenden Stunde gemeinsam mit den Schülerinnen und Schülern erarbeitet werden. Eine weitere Reflexion der Vorgehensweise bietet sich für die Stunde an, in der die von den Gruppen eigenständig erarbeitete Methode fertiggestellt wurde.

Die Ableitung der Gesamt-Methode aus den einzelnen Diagrammen wird durch bidirektionale Objektstrukturen stark vereinfacht: In der Welt von Fujaba kennen Objekte sich immer gegenseitig. Das Werkzeug übernimmt die Implementation des entsprechenden Verhaltens, insbesondere die Konsistenzsicherung. Aus der Verwendung bidirektionaler Assoziationen resultiert jedoch nicht zwangsläufig, dass die Liste als eine doppelt verkettete implementiert wird. Denn abstrakte Datentypen abstrahieren ja ihre nach außen sichtbare Schnittstelle von der Implementation. Ihre Entwicklung geht auf drei Strömungen zurück (vgl. Quibeldey-Cirkel, 1994, S. 91f): a) vom Benutzer definierte Datentypen; b) das Klassenkonzept in Simula und c) Parnas' Prinzip des information hiding. Insbesondere das letzte Prinzip ist wichtig:

„Datenstrukturen sollen die Details ihrer Implementierung und ihrer Manipulation in einem gemeinsamen Modul verbergen. Der Anwender *abstrahiert* (Hervorhebung im Original, C.S.) von der Realisierung eines Datentyps“ (Quibeldey-Cirkel, 1994, S. 92)

Die Verwendung bidirektionaler Assoziationen ist das Mittel zur Implementation, nicht zur Modellierung des ADT. Denn so können Objektstrukturen benutzt werden, ohne darauf achten zu müssen, ob die jeweils betrachtete Verbindung auch in der gewünschten Richtung verfügbar ist.

## 5 Fazit

Wie sind nun die mit dem Einsatz von Fujaba als Werkzeug zur Darstellung abstrakter Datentypen verbundenen Fragen (vgl. Abschnitt 1, S. 2) zu beantworten?

Zunächst zur ersten Frage: Die relativ einfachen grafischen Bausteine der Aktivitätsdiagramme reichen aus, um damit Methoden abstrakter Datentypen implementieren zu können.

Aber, zweite Frage, ist die Implementation auch anschaulich?

Die Schülerinnen und Schüler konnten mit den Darstellungen gut umgehen und fanden sie auch einfach zu verstehen. Hier könnten ähnliche Effekte wie bei der Untersuchung von Kutar, Britton und Barker (2002) eine Rolle gespielt haben (vgl. Abschnitt 3, dort S. 8), da die verwendeten Methoden wie Puzzle oder Zettel-Test zu gründlichem Lesen zwingen, sodass dadurch das Diagramm gründlich verstanden wird.

Insgesamt hatten die Schülerinnen und Schüler keine Schwierigkeiten, die Diagramme zu lesen. Allerdings muss die grundlegende Semantik (Die drei Aufgaben von Story-Pattern (vgl. Abschnitt 2.1, S. 4) bewusst sein. Dazu scheint es sinnvoll zu sein, im Unterricht eine bestimmte Wenn-Dann-Sprechweise zur Erklärung einzuführen: „WENN die Objekte ... und Beziehungen und Zusicherungen ... gefunden werden können, DANN werden folgende Änderungen durchgeführt...“. Diese Sprechweise hilft, den grundlegenden Prozess immer wieder deutlich zu machen.

Zur dritten Frage: Wie können die entsprechenden Diagramme im Unterricht entwickelt werden?

Die Idee des Story-Driven-Modelling (SDM) und die vorgestellten Bausteine stellen eine Reihe von praktikablen Möglichkeiten bereit, mit denen die Diagramme im Unterricht eingesetzt und erstellt werden können.

Besonders vorteilhaft finde ich, dass das Verfahren mit den verschiedenen Sozialformen (EA, PA, GA und UG) kombinierbar ist. Dagegen kann man die Implementation in einer textuellen Programmiersprache nur schlecht als GA oder UG durchführen. Gerade diese beiden Sozialformen sind aber mit dem SDM sehr gut durchführbar. Auch methodisch sind viele Variationsmöglichkeiten denkbar: Zu Beginn des SDM (Fälle finden) können Objektspiele durchgeführt, direkt Objektdiagramme gezeichnet oder einfach Fälle schriftlich notiert werden. Der Unterricht zur Erstellung von Aktivitätsdiagrammen kann variiert werden, indem Teillösungen und/oder Puzzles vorgegeben werden oder das Funktionsprinzip mit Dobs erkundet wird.

Zur Sicherung der Arbeitsschritte und -ergebnisse sind nach den Erfahrungen und Aussagen der Schülerinnen und Schüler Mitschriften gut geeignet. Die Diagramme sind schnell abzuzeichnen, allerdings sollten sie mit den entsprechenden Farben (grün, schwarz, rot) gezeichnet werden. Bei Bedarf können den Diagrammen dann Kommentare in einer weiteren Farbe hinzugefügt werden.

Interessant ist auch, dass viele Unterrichtsphasen ohne die Nutzung des Rechners auskommen. Der Rechner wird im Grunde nur noch zum Prüfen vorher entwickelter Lösungen eingesetzt (und das auch mit 'dynamischen' Diagrammen in Dobs) – und auch das praktisch als eine weitere Alternative zum Zettel-Test. Nun ist Informatikunterricht nicht deswegen gut, weil kein Rechner eingesetzt wird und dessen Einsatz sollte m.E. auch weiterhin eine Rolle spielen. Aber wenn es gelingt, Phasen des eher planlosen Ausprobierens am Rechner zu verkürzen, dann ist das ein Vorteil. Das gelingt hier durch die direkte Unterstützung des Problemlöseprozesses, da mit dem SDM Probleme schrittweise (meiner Meinung nach am effektivsten in Gruppenarbeit)

in Teilprobleme zerlegt, bearbeitet, überprüft und zu einer Gesamtlösung zusammengesetzt werden können.

Ein wesentliches Element dabei ist die anschauliche Darstellung mit den verschiedenen Diagrammarten. Zudem erleichtert die grafische Darstellung auch das Präsentieren von Lösungen, da eine Grafik häufiger und einfacher durch die Schülerinnen und Schüler mit eigenen Worten erläutert wird als ein Quelltext, den viele Schüler oft nur vorlesen oder direkt den Quelltext ins Deutsche übersetzen (vgl. dazu auch das Modell des Wissenserwerbs mit Multimedia, Abbildung 3, S. 8).

Mit der vorgestellten Vorgehensweise ist Fujaba ein sehr gut geeignetes Werkzeug zur anschaulichen Darstellung. In einer Unterrichtsreihe zum Thema ADT würde ich, im Unterschied zur durchgeführten Reihe, die Schülerinnen und Schüler nur ein bis zwei Methoden eines ADT exemplarisch implementieren lassen, um mehr Zeit für die Anwendung der Datentypen zu haben. Dazu könnte gegebenenfalls das Wartezimmer-Projekt von den Schülerinnen und Schülern erstellt und erweitert werden. Auf diese Weise wird vielleicht deutlicher, dass ein ADT von seiner Implementierung abstrahiert und eine Implementation nur in Kenntnis der Funktionen benutzt werden kann (und soll).

Das Konzept selbst jedenfalls wird erneut unterrichtet werden, unter den Kolleginnen und Kollegen wurde vereinbart, dass ich dazu eine schulinterne Fortbildung abhalten werde.

Durch die Art der Diagramm-Nutzung liegt der größte Vorteil des Konzepts in den verwendeten Diagrammen, welche die Lücke zwischen dem ersten, groben Verständnis und dem algorithmischen bzw. informatischem Verständnis verkleinern. Zur Erläuterung können die einzelnen Methoden/Diagramme den Stufen enaktiv, ikonisch und symbolisch zugeordnet werden. Objektspiel oder die interaktive Erkundung mit Dobs, zum Teil der Zettel-Test, liegen auf der enaktiven Ebene, bei der sich die Lernenden spielerisch einem ersten Verstehen nähern: Sie spielen Beispielsituationen nach. Objektdiagramme sind die ikonische Darstellung der Erkundung, sie stellen eine unmittelbare Veranschaulichung der durchgespielten Situation dar: Etwa ein Ausdruck eines Objektdiagramms aus Dobs, der als Sicherung einer interaktiven Erkundung angefertigt wird, oder ein Objektdiagramm, das ein Objektspiel sichert.

Ein Aktivitätsdiagramm zeigt den Ablauf einer Methode dann auf der verallgemeinerten und abstrahierten, symbolischen Ebene (so wie das Klassendiagramm als abstrahierte, symbolische Darstellung der möglichen Objektstrukturen aufgefasst werden kann).

Die ikonische Hilfsebene ist sozusagen größer, als dies bei den sonst üblichen grafischen Darstellungen und dem textuellen Programmcode der Fall ist. Denn die Objektdiagramme können schrittweise (anhand der einzelnen Darstellungen!) in Aktivitätsdiagramme überführt werden. Insbesondere das Story-Boarding (Tabelle 3, S. 23) unterstützt diesen Wechsel von der ikonischen zur symbolischen Ebene. Der Zettel-Test (Abschnitt 4.2, S. 15) ermöglicht dann wiederum einen einfachen Zugang zum genauen Verstehen dieser Ebene.

## 6 Literatur

- Baumert und Köller 2000: Baumert, J.; Köller, O.: Unterrichtsgestaltung, verständnisvolles Lernen und multiple Zielerreichung im Mathematik- und Physikunterricht der gymnasialen Oberstufe. In: Baumert, Bos und Lehmann 2000. S. 271-316.
- Baumert, Bos und Lehmann 2000: Baumert, J.; Bos, W.; Lehmann, R. (Hrsg.): TIMSS/III. Dritte Internationale Mathematik- und Naturwissenschaftsstudie. Mathematische und naturwissenschaftliche Bildung am Ende der Schullaufbahn. Band 2: Mathematische und physikalische Kompetenzen am Ende der gymnasialen Oberstufe. Leske u. Budrich 2000.
- Deutsches PISA-Konsortium 2001: Baumert, J.; u.a.: PISA 2000. Basiskompetenzen von Schülerinnen und Schülern im internationalen Vergleich. Leske u. Budrich 2001.
- Diethelm, Geiger und Zündorf 2002: Diethelm, I.; Geiger, L.; Zündorf, A.: UML im Unterricht: Systematische objektorientierte Problemlösung mit Hilfe von Szenarien am Beispiel der Türme von Hanoi. In: Schubert, Magenheimer, Hubwieser und Brinda 2002. S.33-42.
- Diethelm, Geiger und Zündorf 2004: Diethelm, I.; Geiger, L.; Zündorf, A.: Systematic Story Driven Modeling. Technical Report, University of Kassel. [www.se.eecs.uni-kassel.de/se/fileadmin/se/publications/SDM04.pdf](http://www.se.eecs.uni-kassel.de/se/fileadmin/se/publications/SDM04.pdf) (zuletzt besucht am 21.05.2004)
- Fischer, Niere und Torunski 1998: Fischer, T.; Niere, J.; Torunski, L.: Konzeption und Realisierung einer integrierten Entwicklungsumgebung für UML, Java und Story-Driven-Modeling. Diplomarbeit, Universität Paderborn 1998.
- Holland, Griffiths und Woodmann 1997: Holland, S.; Griffiths, R.; Woodmann, M.: Avoiding Object Misconceptions. ACM SIGCSE Bulletin 29 (1997) Nr. 1. S. 131-134.
- Hubwieser 2001: Hubwieser, Peter: Didaktik der Informatik. Grundlagen, Konzepte, Beispiele. 2. korr. Auflage. Springer 2001.
- Jacobson, Booch und Rumbaugh 1999: Jacobson, I.; Booch, G.; Rumbaugh, J.: The Unified Software Development Process. Addison-Wesley 1999.
- Johlen 2002: Johlen, D.: Methodik der OOSE für Fachinformatiker nach dem Lernfeldansatz unter Einbeziehung der Lehrerfortbildung. In: Schubert, Magenheimer, Hubwieser und Brinda 2002. S.55-64.
- Klieme, Neubrand und Lüdtke 2001: Klieme, E.; Neubrand, M.; Lüdtke, O.: Mathematische Grundbildung: Testkonzeption und Ergebnisse. In: Deutsches PISA-Konsortium 2001. S. 141-191.
- Kölling und Rosenberg 2001: Kölling, M.; Rosenberg, J.: Guidelines for Teaching Object Orientation with Java. In: ItiCSE 2001. Canterbury 2001. S. 33-36.
- Kutar, Britton und Barker 2002: Kutar, M.; Britton, C.; Barker, T.: A Comparison of Empirical Study and Cognitive Dimension Analysis in the Evaluation of UML Diagrams. In: 14th Workshop of the Psychology of Programming Interest Group. PPIG 2002.
- life-Webseite: <http://life.upb.de/> (auch erreichbar über learnline. Zuletzt besucht am 21.05.2004)
- learnline: <http://www.learn-line.nrw.de/angebote/oop/> (zuletzt besucht am 21.05.2004). Das Projekt Wartezimmer befindet sich in einem nur für registrierte Lehrerinnen und Lehrer zugänglichen Bereich: <http://www.learn-line.nrw.de/angebote/oop/medio/kurs/einfuehrung/wartezimmer/index.html>
- Moll 2002: Moll, S.: Objektorientierte Modellierung unter Einsatz eines CASE-Tools im Infor-

- matikunterricht der Jahrgangsstufe 11. In: Schubert, Magenheimer, Hubwieser und Brinda 2002. S. 43-52.
- Oesterreich 1999: Oesterreich, B.: Objektorientierte Softwareentwicklung. Analyse und Design mit der Unified Modeling Language. Oldenbourg 1999.
- Poloczek: Webangebot von Dr. Jürgen Poloczek auf dem hessischen Bildungsserver: <http://www.bildung.hessen.de/abereich/inform/skii/material/delphi/listen/warteschlange.htm> (zul. Bes. am 21.5.04).
- Quibeldey-Cirkel 1994: Quibeldey-Cirkel, Klaus: Das Objekt-Paradigma in der Informatik. Teubner 1994.
- Reinsch 2003: Reinsch, T.: Darstellung und Analyse eines objektorientierten Einstiegs im Anfangsunterricht der Sekundarstufe I mit Hilfe von UML und Fujaba. Schriftliche Hausarbeit vorgelegt im Rahmen der Zweiten Staatsprüfung für das Lehramt für die Sekundarstufe I/II in Informatik. Studienseminar Bonn 2003.
- Richtlinien: Ministerium für Schule, Wissenschaft und Forschung (Hrsg.): Richtlinien und Lehrpläne für die Sekundarstufe II – Gymnasien/Gesamtschule in Nordrhein-Westfalen. Informatik. Ritterbach Verlag 1999.
- Scanlan 1988: Scanlan, D.: Should short, relatively complex algorithms be taught using both graphical and verbal methods? Six replications. In: ACM SIGCSE Bulletin 20 (1988) Nr. 1. S. 185-189.
- Schnotz 2001: Schnotz, W.: Wissenserwerb mit Multimedia. In: Zeitschrift für Unterrichtswissenschaft 29 (2001). S. 293-318.
- Schubert, Magenheimer, Hubwieser und Brinda 2002: Schubert, S.; Hubwieser, P.; Magenheimer, J.; Brinda, T.: Forschungsbeiträge zur 'Didaktik der Informatik' – Theorie, Praxis, Evaluation. 1. GI-Workshop DDI '02 (Schwerpunkt: Modellierung in der informatischen Bildung). Gesellschaft für Informatik 2002.
- Schulte 2004: Schulte, C.: Lehr- Lernprozesse im Informatik-Anfangsunterricht. Theoriegeleitete Entwicklung und Evaluation eines Unterrichtskonzepts zur Objektorientierung in der Sekundarstufe II. Dissertation, Universität Paderborn, 2004.
- Shneidermann u.a. 1977: Shneiderman, B.; Mayer, R.; McKay, D.; Heller, P.: Experimental investigations of the utility of detailed flowcharts in programming. In: Communications of the ACM 20 (1977) Nr. 6. S. 373-381.
- Ullenboom, Christian: Java ist auch eine Insel. Programmieren für die Java 2-Plattform in der Version 1.4. (3. Auflage). Galileo Computing 2003.
- Zündorf 2002: Zündorf, A.: Rigorous Object Oriented Software Development. Draft. Version 0.3. (5.3.2002). Manuskript, Universität Paderborn 2002.

## 7 Versicherung

### Versicherung

Ich versichere, dass ich die Arbeit selbstständig verfasst, keine anderen Quellen als die angegebenen benutzt und die Stellen der Arbeit, die anderen Werken dem Wortlaut oder Sinn nach entnommen sind, in jedem einzelnen Fall unter Angabe der Quelle kenntlich gemacht habe. Das Gleiche gilt auch für beigegebene Zeichnungen, Kartenskizzen und Darstellungen.

---

(Datum)

---

(Unterschrift)

## 8 Anhang

### 8.1 Glossar

- Aktivitätsdiagramm:** Grafische Darstellung einer Methodenimplementation in Fujaba. Besteht hauptsächlich aus Story-Pattern, die durch Transitionen verbunden werden. Neben Story-Pattern können auch Quelltext-Statements eingefügt werden.
- Assertion:** (Zusicherung) Ein Objekt im Story-Pattern, für das ein bestimmter Attributwert angegeben wird, z.B.: Name des Patienten soll gleich „Peter“ sein.
- Bound:** Gebundene Variable. Die in einem Story-Pattern dargestellte Objekte sind genau genommen Variablen für Objekte. Wenn eine Variable nicht nur deklariert, sondern auch initialisiert ist, dann ist sie bound: an ein bestehendes Objekt gebunden. Das Objekt this ist immer bound.
- Collaborations-Statement:** Methodenaufruf innerhalb eines Story-Patterns. Wird nur im success-Fall ausgeführt. Beispiele in Abbildung 16, S. 35.
- Dobs:** (Dynamic Object Browsing System, auch Mr.Dobs) Grafischer Debugger, der Objekte als Objektdiagramm anzeigt und das Ausführen von Methoden ermöglicht.
- Failure:** Ergebnis eines Story-Patterns bei erfolgreicher Ausführung ist success, ansonsten failure. Wird zur Steuerung des Kontrollfluss im Aktivitätsdiagramm benutzt (siehe Transition).
- Fujaba:** (From UML to Java And Back Again). UML-Implementationswerkzeug, das aus UML-artigen Diagrammen ausführbaren Java-Quelltext erzeugt.
- Link:** Beziehung zwischen Objekten. Wird in Story-Pattern verwendet und hat dort drei Aufgaben: 1) Prüfen, ob eine Beziehung besteht oder eine Zusicherung erfüllt wird, 2) Anlegen oder 3) Löschen einer Beziehung oder einer Zusicherung. Ein link bezieht sich auf eine Assoziation, Aggregation oder Komposition, die aus dem Klassendiagramm übernommen wird.
- NOP:** (No-Operation). Wird im Aktivitätsdiagramm als Raute dargestellt, welche je zwei Ein- und Ausgänge für Transitionen hat. Dient zur Darstellung von Verzweigungen und Schleifen.
- Objektdiagramm:** Darstellung von Objekten als Kästchen. Beziehungen als Pfeile (gerichtet) oder Linien (ungerichtet, bzw. bidirektional). Attributwerte können ebenfalls dargestellt werden. Diese Darstellungsart wird von Dobs verwendet.
- Objektstruktur:** Durch Beziehungen (Assoziation, Aggregation oder Komposition) zusammenhängendes Netz von Objekten. Darstellbar mit Objektdiagrammen.
- Story-Boarding:** Bezeichnet ein Verfahren zur Ableitung von Aktivitätsdiagrammen mit einem Schwerpunkt auf der Ableitung von Story-Pattern. Grundlage sind meist Objektdiagramme, die schrittweise überführt werden. Einzelne Schritte sind: Vergeben von Namen, evtl. Bestimmen eines this-Objekts, Bestimmen der Beziehungen und Klassen, evtl. Einfügen von Collaborations-Statements.
- Story-Driven-Modelling:** (SDM) Verfahren zur systematischen Implementation in Fujaba. Benutzt die Darstellung einzelner Fälle, aus denen mit dem Story-Boarding Aktivitätsdiagramme erzeugt werden, die dann zur vollständigen Methodenimplementation zusammengeführt und mit dem Zettel-Test geprüft werden.
- Story-Pattern:** Wichtigster Baustein eines Aktivitätsdiagramms. Ist ein Objektdiagramm, in dem eine Objektstruktur beschrieben wird. Wenn zur Laufzeit die beschriebene Struktur mit einer bestehenden Objektstruktur identifiziert werden kann, dann werden die im Sto-

ry-Pattern angegebenen Veränderungen (Erzeugen/Löschen von Objekten oder Beziehungen, Veränderungen von Attributwerten, Methodenaufrufe durch Collaborations-Statements) durchgeführt. Die interne Zustandsvariable hat dann den Wert success, ansonsten den Wert failure. Damit der Such- bzw. Identifikationsprozess durchgeführt werden kann, benötigt jedes Story-Pattern ein bound-Objekt, von dem aus die Suche beginnen kann.

In folgenden Story-Pattern können die identifizierten Objekte als bound genutzt werden. Dargestellte Objekte der Form 'name: Typ' entsprechen Variablendeklarationen, gebundene Objekte haben die Form 'name'.

**Success:** Ergebnis eines Story-Patterns bei erfolgreicher Ausführung ist success, ansonsten failure. Wird zur Steuerung des Kontrollfluss im Aktivitätsdiagramm benutzt (siehe Transition).

**Transition:** Verbindet die Bausteine eines Aktivitätsdiagramms. Eine Transition wird als Pfeil dargestellt. Einfache Pfeile werden immer durchlaufen, beschriftete nur, wenn die dargestellte Bedingung erfüllt ist. Mögliche Bedingungen: Boolesche Ausdrücke, success, failure, else. Siehe auch NOP.

**Zettel-Test:** Schrittweise Ausführung eines Aktivitätsdiagramms per Hand, indem mit Zetteln der Bindungsprozess der Variablen eines Story-Patterns an einem Objektdiagramm durchgeführt wird. Bei erfolgreicher Bindung werden dann die beschriebenen Änderungen am Beispiel durchgeführt und geprüft, ob die Änderungen korrekt sind.

**Zusicherung:** siehe Assertion.

## 8.2 Weitere Diagramm-Beispiele

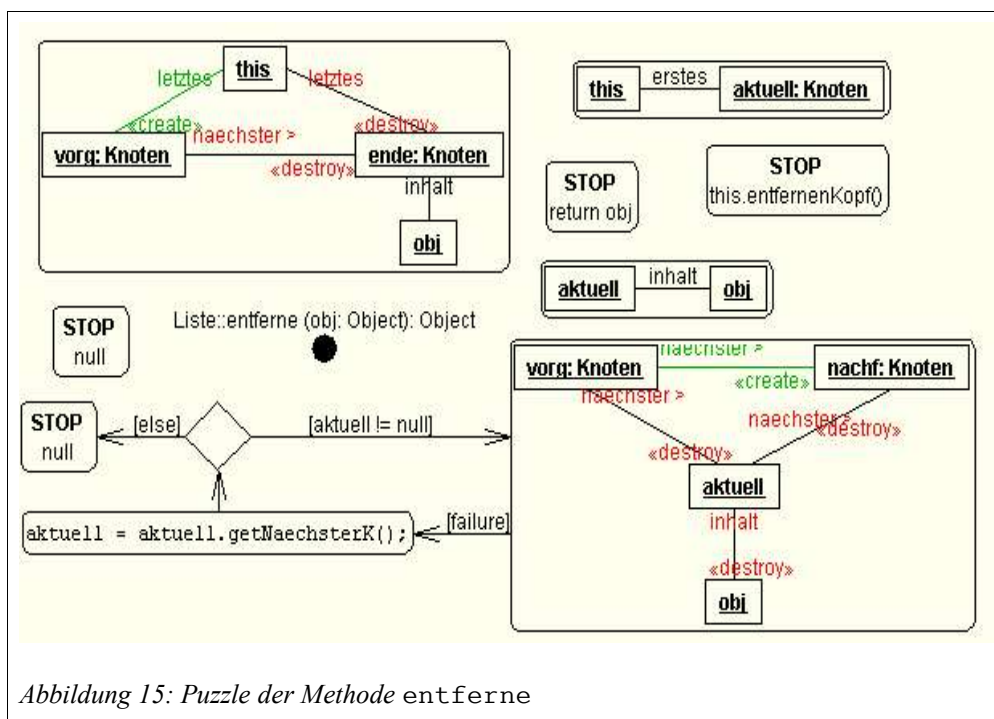


Abbildung 15: Puzzle der Methode entferne

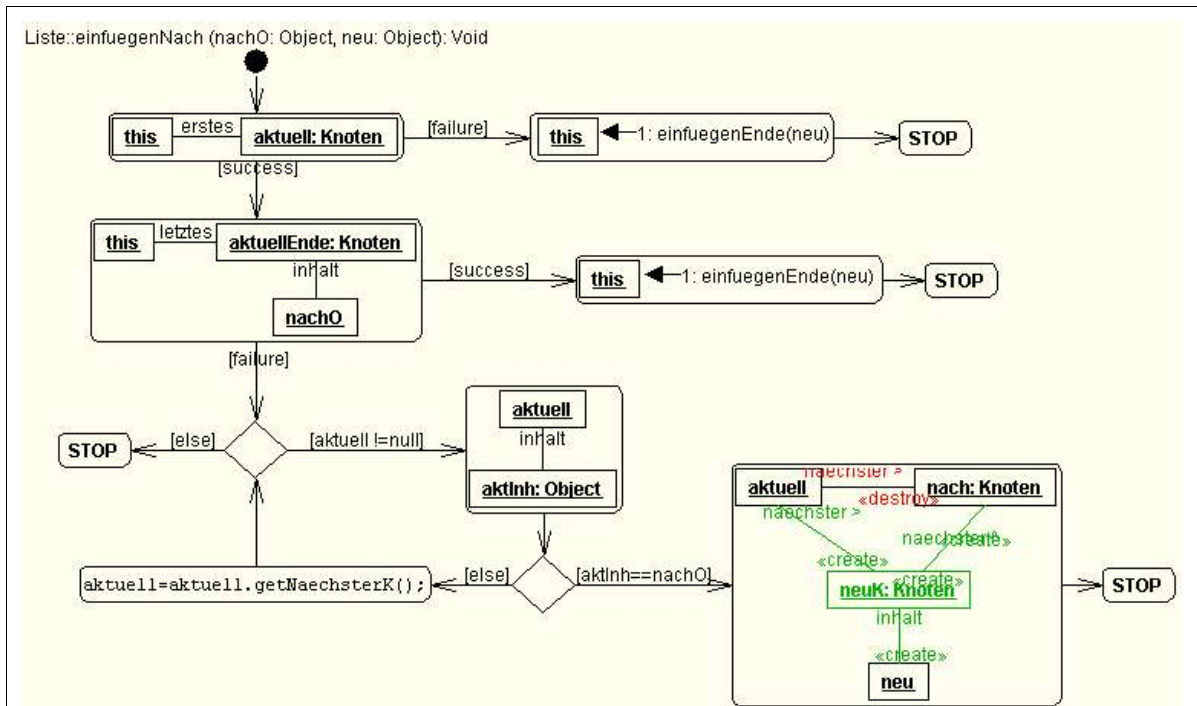


Abbildung 16: Methode `einfuegenNach` des ADT `Liste`. Zurückgeführt auf drei Fälle: Liste leer, einfügen nach dem letzten Objekt, einfügen in der Mitte.

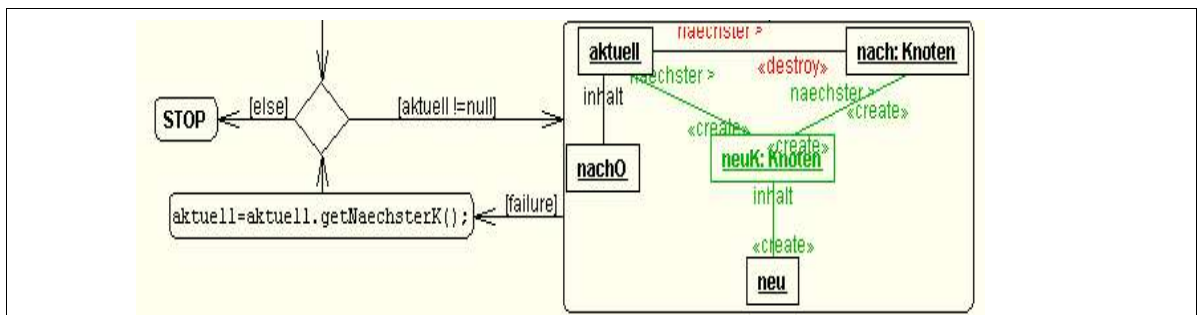


Abbildung 17: Vereinfachte Schleife der Methode `einfuegenNach` aus Abbildung 16. Dort wurde mit einer zusätzlichen Verzweigung geprüft, ob der aktuelle Knoten auch derjenige ist, der das gesuchte Objekt `nachO` verwaltet. Diese Prüfung kann ja das Story-Pattern, in dem die Objektstruktur verändert wird, selbst vornehmen. Ist ein typischer 'Fehler' bei der Nutzung von Fujaba, wenn man vom Quelltext ausgeht: dort wird ja zunächst in einer Verzweigung die Bedingung geprüft bevor die Änderung beschrieben wird.

### **8.3 CD**

Auf der beiliegenden CD finden Sie:

- Die im Unterricht verwendete Fujaba-Version
- Fujaba-Projekte mit Grafiken (Aktivitätsdiagramme, Screenshot Dobs-Beispiel ...)
- Hilfen zu Fujaba. U.a.: Dokumentation und eine Flash-Animation, welche die Semantik von Story-Pattern erklärt.
- Literatur zu Fujaba