Handout zur informatik-didaktischen Fortbildung von Lehrenden

Dr. Nicole Weicker
Universität Stuttgart
weicker@informatik.uni-stuttgart.de

7. Juli 2005

1 UML

Die folgenden Anmerkungen zu UML zeichnen ein high-level Bild über die Zusammenhänge und erhebt keinen Anspruch auf Korrektheit oder Vollständigkeit.

Als Literatur wird empfohlen: Patrick Grässle, Henriette Baumann und Philippe Baumann „UML 2.0 projektorientiert“, 3. Auflage, 2004, Galileo Computing Verlag, ISBN 3-89842-547-9

Das Buch zeichnet sich vor allem dadurch aus, dass ein Projekt durchgängig beschrieben wird und UML auf die aus praktischer Sicht relevanten Teile beschränkt dargestellt wird.

1.1 Was ist UML

Die Unified Modeling Language (UML) ermöglicht es, Systeme in Worten und Bildern zu beschreiben. UML ist eine standardisierte Modellierungssprache mit einer formal definierten Bedeutung. UML eignet sich für die Darstellung verschiedenster Modellsichten eines IT-Systems. Dabei stellt jede Modellsicht nur einen Ausschnitt des IT-Systems dar. Bei jeder Modellsicht werden andere Informationen hervorgehoben bzw. weggelassen.

1.2 Diagrammarten

Geschäftsprozesse werden in Formularen ausformuliert beschrieben. Jegliche Aktion, die mit dem System ausgeführt werden soll, ist als Geschäftsmodell zu erfassen. Wichtig ist dabei, dass nur die Benutzeraktionen und nicht mögliche interne Realisierungen Geschäftsprozesse darstellen.

use-case Diagramme geben einen visuellen Überblick über die Geschäftsprozesse. Allerdings ist es oft nicht nötig, alle Geschäftsprozesse als use-case visuell aufzubereiten.

Klassendiagramme modellieren die Datensicht der Software. Welche Daten müssen verwaltet werden und wie lassen sich diese organisieren.

Zustandsdiagramme können verwendet werden, um den inneren Zustand von Klassen zu beschreiben. Im wesentlichen sind Zustandsdiagramme endliche Automaten.

Klassendiagramme können um *Methoden* erweitert werden, die die Schnittstellen zwischen den Klassen verdeutlichen.

Sequenzdiagramme eignen sich, um den zeitlichen Ablauf eines Szenario (sprich die Reihenfolge von externen und internen Aktionen) zu verdeutlichen. Sie können verwendet werden, um use-cases zeitlich festzulegen oder auch um einen Entwurf auf Tauglichkeit zu überprüfen.

Aktivitätsdiagramm wird für den Darstellung von Abläufen aus externer Sicht verwendet. Die Aktivitätsdiagramme sind eng mit „Flussdiagrammen“ verwandt. Durch Aktivitätsdiagramme wird deutlich, welche Geschäftsprozesse von welchen Akteuren (Klassen) gemeinsam oder unabhängig voneinander durchgeführt werden können.

1.3 Prozessmodell: Wasserfallmodell

Das einfachste Prozessmodell zur Entwicklung einer Software ist das Wasserfallmodell, bei dem in einem ersten Schritt die Anforderungsspezifikation (Pflichtenheft) erstellt wird und mit dem Kunden abgestimmt wird. Diese Vereinbarung zwischen Kunden und Software-Ersteller beinhaltet die Beschreibung aller Geschäftsprozesse (use-cases), die Datenanalyse (welche Daten müssen aus Anwendersicht in der Software enthalten sein → Klassendiagramm) sowie sonstige Anforderungen (Sicherheitsanforderungen, Wartbarkeit, Erweiterbarkeit, ...)

In einem zweiten Schritt wird ein Entwurf für die Software erstellt. Dabei wird eine Software-Architektur konzipiert, die die Datenverwaltung und die Interaktionen zwischen den Klassen unterstützt. Dazu werden die Klassendiagramme um Methoden ergänzt, weitere Hilfsklassen, Oberflächenklassen, u.ä. hinzugefügt. Durch weitere Verfeinerungen und Anpassungen an die gewählte Sprache entsteht daraus der Feinentwurf. Über Sequenzdiagramme können die einzelnen Geschäftsprozesse für den Entwurf durchgespielt und damit die Tauglichkeit des Entwurfs überprüft werden.

Die Grenzen der Anwendungsbereiche der einzelnen Diagrammarten sind dabei fließend.

1.4 Vorgehensweise im Informatikunterricht

Eine Möglichkeit, wie UML und Prozessmodelle im Informatikunterricht motivierend und anwendungsnah behandelt werden können, besteht darin, in einem ersten Schritt

eine bekannte Software (z.B. ein aktuelles Spiel wie „Warcraft“ oder „Sims“) hinsichtlich der Daten, die vom System benötigt werden und der möglichen Benutzeraktionen analysiert wird. Diese Analyse braucht nicht vollständig zu sein, motiviert jedoch sich über die innere Verwaltung und Darstellung von Daten, sowie deren Manipulationsmöglichkeiten Gedanken zu machen. Klassendiagramme sowie Geschäftsprozesse können anhand dieses Beispiels einfach und nachvollziehbar eingeführt werden. Die anderen Diagrammarten können nach Bedarf hinzugefügt werden.

In einem zweiten Schritt kann die Klasse die Entstehung einer Software von der Idee bis zur Realisierung gemeinsam erarbeiten. Die Idee kann über Brainstorming entwickelt werden. Eine kleine Gruppe kann die Rolle des Kunden übernehmen. Sprich, diese SchülerInnen haben die Aufgabe, sich über die Idee hinaus konkrete Szenarien zu überlegen und diese festzulegen. Die restliche Klasse hat die Aufgabe, aus den Kunden die Anforderungen möglichst vollständig „herauszukitzeln“. Sehr gut hat sich dabei bewährt, Zeitgutscheine für die Gespräche mit den Kunden auszugeben. Ein Kunde kann nicht beliebig oft und lange befragt werden. Die Ergebnisse der Anforderungsanalyse werden in Form der schon bekannten UML-Diagramme notiert. Wenn diese vollständig scheinen, ist der nächste Schritt, die interne Sicht der Daten und ihrer Interaktionen über Diagramme zu erschließen. Möglicherweise wird dabei deutlich, dass Experten für verschiedene Teilaufgaben benötigt werden (Oberfläche, Versionsverwaltung, Tests,...). Neben der Rolle des Projektleiters, der die Teamtreffen vorzubereiten und zu leiten hat, gibt es beispielsweise auch die Rolle des Kundenbeauftragten (der alle Kundenkontakte abwickelt), des Schriftführers (der nicht etwa alle Schriftstücke erstellt, sondern vor allem verwaltet) und des Kontrolleurs (der den Prozessablauf im Auge behält). Inwieweit die Software tatsächlich implementiert wird, hängt von der Zeit und der Größe des Projekts ab. Für die Motivation ist es immer förderlicher, wenn ein Produkt erstellt wird!

2 Lernstile

R.M. Felder and L.K. Silverman, Learning and Teaching Styles in Engineering Education,Ëngr. Education, 78(7), 674-681 (1988). The article that originally defined the Felder-Silverman model and identified teaching practices that should meet the needs of students with the full spectrum of styles. The paper is preceded by a 2002 preface that states and explains changes in the model that have been made since 1988.

Literatur

Rüdiger Baumann. *Didaktik der Informatik*. Klett, Stuttgart, 1996.

Franz Eberle. *Didaktik der Informatik bzw. einer informationstechnologischen und kommunikationstechnologischen Bildung auf der Sekunda*. Sauerländer GmbH Verlag, Aarau, 1996.

Peter Hubwieser. *Didaktik der Informatik: Grundlagen, Konzepte, Beispiele*. Springer, Berlin, 2000.

Sigrid Schubert and Andreas Schwill. *Didaktik der Informatik*. Spektrum Akademischer Verlag, Heidelberg, Berlin, 2004.