Anonym

Unterstützung der objektorientierten Softwareentwicklung durch die Visualisierung von Anforderungen anhand des UML - Modellierungswerkzeuges Rational Rose

Diplomarbeit

Bibliografische Information der Deutschen Nationalbibliothek:

Die Deutsche Bibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de/ abrufbar.

Impressum:

Copyright © 2004 GRIN Verlag
ISBN: 9783638331951

Anonym

Unterstützung der objektorientierten Softwareentwicklung durch die Visualisierung von Anforderungen anhand des UML - Modellierungswerkzeuges Rational Rose

GRIN Verlag

Unterstützung der objektorientierten Softwareentwicklung, durch die Visualisierung von Anforderungen, anhand des UML-Modellierungswerkzeuges Rational Rose

DIPLOMARBEIT

im Studiengang Berufsintegrierendes Studium Betriebswirtschaft

des Fachbereiches III: Wirtschaftswissenschaften

der Fachhochschule Mainz

Vorgelegt von: anonym

Eingereicht am: 05.10.2004

Inhaltsverzeichnis

Abkürzungsverzeichnis

AG	Aktiengesellschaft
ca.	circa
cm	Zentimeter
dpi	(engl.) dots per inch (Punkte pro Inch)
EAN-Code	Europäischer Artikelnummer-Code
EDV	Elektronische Datenverarbeitung
E-Mail	(engl.) Electronic Mail (Elektronische Nachricht)
GB	Gigabyte
GmbH	Gesellschaft mit beschränkter Haftung
ID	Identifikation
IT	Informationstechnologie
kg	Kilogramm
m	Meter
MB	Megabyte
MwSt.	Mehrwertsteuer
MS	Microsoft
NW	Netzwerk
OE	Organisationseinheit
OOA	Objektorientierte Analyse
OOD	Objektorientiertes Design
OOP	Objektorientierte Programmierung
PC	Personal Computer
PIN	Persönliche Identifikationsnummer
PLZ	Postleitzahl
S.	Seite
s/w	schwarz/weiß
UML	(engl.) Unified Modeling Language
vs.	(lat.) versus (gegen)

Abbildungsverzeichnis

Abbildungen **Seite**

Tabellenverzeichnis

Tabellen **Seite**

1. Einführung

Die Erfolgschance auf neue ertragreiche Softwareprojekte ist in den vergangenen Jahren gesunken. Dennoch scheitern, insbesondere durch das Nichteinhalten festgelegter Budgets und/oder Zeitrahmen, nach wie vor ein Großteil der beauftragten Projekte.(Vgl. Norris, 1995, zitiert bei: Thaller, 2002, S. 17)

Die Literatur sieht die Ursachen hierfür vor allem im schlechten Projektmanagement, im Fehlen zukunftsorientierter Gesamtkonzepte und im mangelnden methodischen Vorgehen. Der Programmcode entsteht oftmals zu früh, so dass die wahren Anforderungen der Anwender keine Berücksichtigung finden.(Vgl. Seidel, 2004, S. 1; vgl. o.V., 2004, S. 1)

Ein weiteres Problem stellt häufig die unterschiedliche Spezialisierung der Projektbeteiligten dar. Je spezialisierter Auftraggeber, Anwender, Analytiker und Entwickler in ihren Fachbereichen sind, umso schwieriger ist es die Grundlage für eine erfolgreiche Entwicklungsarbeit herzustellen bzw. eine gemeinsame Basis der Verständigung zu schaffen.(Vgl. Oestereich, 2001, S. 18).

Die objektorientierte Softwareentwicklung begegnet dieser Herausforderung, indem sie die Projekte von der Anforderungsanalyse bis zur Implementierung des Systems, durch eine einheitliche Notation begleitet. In der Praxis hat sich die Unified Modeling Language (UML) als standardisierte Modellierungssprache bewährt. Die Symbole und Diagramme der UML gelten als leicht verständlich und fördern damit den Informationsaustausch zwischen den Beteiligten.

Ziel dieser Diplomarbeit ist es, dem Leser nicht nur einen ausgewählten Extrakt der UML-Modellelemente vorzustellen, sondern vor allem deren Verwendbarkeit an praxisorientierten Beispielen aufzuzeigen. Dazu vermittelt der erste Teil dieser Arbeit die theoretischen Grundlagen, die für das Verständnis der fiktiven Fallstudie im zweiten Teil erforderlich sind.

Da in der Literatur einige Begriffe nicht eindeutig definiert oder in der Praxis synonym verwendet werden, erläutert das folgende Kapitel die notwendigen Begrifflichkeiten und grenzt sie voneinander ab.

Das dritte Kapitel beschreibt die wesentlichen Modell- und Diagrammtypen der UML sowie deren standardisierte Elemente. Die Auswahl beschränkt sich dabei, auf die in der Fallstudie modellierten Diagramme und Konzepte.

Der Leser erhält nicht nur grundlegende Informationen zur Verwendung der Modellelemente, sondern lernt gleichzeitig deren Notation, in der UML[1], kennen. Einige Elemente und Diagramme werden anhand einfacher Unternehmensprozesse des erfundenen Unternehmens „Computer GmbH“[2] beschrieben und grafisch dargestellt.

Im vierten Kapitel wurde die Fallstudie „Textil GmbH“ entwickelt, die zeigt, wie die UML in den Kontext eines komplexen Softwareprojektes integriert werden kann. Die Fallstudie setzt sich

- aus dem Pflichtenheft „Stammdatenverwaltung I“ der Textil GmbH (s. Anhang II),
- dem in Rational Rose angefertigten UML-Modell „Diplomarbeit 800410“ sowie
- dem generierbaren Pilotsystem „Ready 2004“

zusammen.

Das fünfte Kapitel zeigt ein Berechtigungskonzept für das neue Anwendungssystem auf.

Im sechsten Abschnitt der Arbeit gibt die Verfasserin einen Ausblick auf die Programmierung des Anwendungssystems „Ready 2004“ und komplettiert damit die Fallstudie „Textil GmbH“.

Neben dem Pflichtenheft umfasst der Anhang dieser Arbeit ein Glossar, das die im Dokument *kursiv* geschriebenen Begriffe erläutert (s. Anhang I). Ferner wurden die in Rational Rose verwendeten UML-Notationen in einer Übersicht zusammengestellt (s. Anhang III).

---

[1] In dieser Arbeit werden die Modellelemente in der UML-Version 1.3 notiert, da die aktuelle Version 2.0 nicht vom eingesetzten UML-Modellierungswerkzeug Rational Rose 2000e unterstützt wird.

[2] Die Computer GmbH verkauft EDV-Zubehör (Hard- und Software) an Endkunden.

2. Begrifflichkeiten

2.1 Objektorientierung

Die objektorientierte Systementwicklung stellt eine Alternative zur klassischen (strukturierten) Systementwicklung dar. Ein wesentlicher Unterschied besteht darin, dass die Daten und Funktionen nicht mehr voneinander getrennt, sondern vielmehr ganzheitlich betrachtet werden.

Dadurch lassen sich nun auch Beziehungen und Abhängigkeiten modellieren, die zwischen den Objekten der realen Welt bestehen.

Die Objektorientierung basiert auf den folgenden Grundprinzipien: (Vgl. Stahlknecht; Hasenkamp, 2002, S. 279)

- *Kapselung* von Daten,
- Klassenbildung und Vererbung (s. Kapitel 3.2) sowie
- Nachrichtenkommunikation (s. Kapitel 3.5) und *Polymorphismus*.

Eine weitere Neuerung besteht darin, dass die Entwicklungsphasen Analyse (OOA), Entwurf (OOD) und Realisierung (OOP) dieselben Konzepte (Klassen, Objekte, Beziehungen, etc.) verwenden. Diese methodische Durchgängigkeit legt den Grundstein für eine schnellere und bessere Verständigung zwischen allen Projektbeteiligten.(Vgl. Balzert, 1999, S. 2)

2.2 UML

Die UML hat sich als Standardnotation für die objektorientierte Modellierung durchgesetzt. Sie ermöglicht, objektorientierte Konzepte in Bildern und Worten darzustellen.(Vgl. Grässle; Baumann; Baumann, 2000, S. 13)

Mit geeigneten UML-Modellierungswerkzeugen, wie z.B. Rational Rose, können die Modelle zudem spezifiziert, konstruiert und dokumentiert werden, vollkommen unabhängig von einer Programmiersprache[3].(Vgl. Booch; Rumbaugh; Jacobson, 1999, S. 12). Da es sich bei der UML um eine standardisierte Modellierungssprache handelt, lässt sich aus einigen Diagrammtypen Quellcode generieren[4].(Vgl.

---

[3] Die Version Rose Enterprise unterstützt die folgenden Sprachen: C++, Java, Ada, CORBA, Visual Basic, COM, Oracle8 und XML (Vgl. Boggs; Boggs, 2003, S. 51).

[4] Die automatisierte Codegenerierung erfolgt hauptsächlich aus statischen Diagrammen heraus, z.B. aus Klassendiagrammen. Durch Reverse Engineering können auch Änderungen im Programmcode automatisch in das Modell zurück übernommen werden.

Oestereich, 2001, S. 22) Die erstellten Codeskelette können als Grundlage für die weitere Systemprogrammierung verwendet werden.

2.3 System

Der Systembegriff hat in der Literatur keine einheitliche Definition. In dieser Arbeit ist ein System als eine Kombination aus Hard- und Software[5] zu verstehen, die Probleme lösen und/oder Geschäftsprozesse optimieren soll. Der bereitgestellte Leistungsumfang von Systemen ist individuell. Er hängt maßgeblich von den spezifischen Anforderungen des Auftraggebers ab.

2.4 Diagramm vs. Modell

Die verschiedenen Beteiligten an einem System verfolgen meist voneinander abweichende Interessen. Demzufolge unterscheiden sich auch ihre Sichtweisen auf das System. So betrachtet beispielsweise ein Anwender, ein Computersystem aus einem anderen Blickwinkel als ein Komponentenhersteller oder ein Designer, der den PC entworfen hat.(Vgl. Schmuller, 2003, S. 34)

Nach Schmuller und Boggs repräsentieren UML-Diagramme diese unterschiedlichen Sichten auf das System (Vgl. Schmuller, 2003, S. 26; vgl. Boggs; Boggs, 2003, S. 76). Ausgewählte Aspekte eines Systems, insbesondere komplexe Problembereiche, können durch Diagramme visualisiert und grafisch dargestellt werden. Sie bilden damit für alle Beteiligten, die Grundlage für ein gemeinsames Verständnis der Anforderungen.

Da komplexe Systeme selten in einem Diagramm darstellbar sind, werden mehrere Diagramme modelliert, die zusammen das Modell des Systems bilden.

Vergleichbar mit den Bauplänen eines Hauses stellen Modelle die Baupläne der Softwareentwicklung dar. Sie zeigen auf, was ein System leisten soll und fördern die frühe Auseinandersetzung mit den Anforderungen, bevor die Programmierarbeit beginnt.(Vgl. Schmuller, 2003, S. 26) Fehler und konzeptionelle Lücken können auf diese Weise noch zu einem frühen Zeitpunkt aufgedeckt und meist mit geringerem Aufwand beseitigt werden als in nachfolgenden Projektschritten (Vgl. Boggs; Boggs, 2003, S. 49).

---

[5] Der Schwerpunkt dieser Arbeit, liegt auf der Betrachtung von Softwaresystemen.