von dr. rudolf hauber und dieter wagner

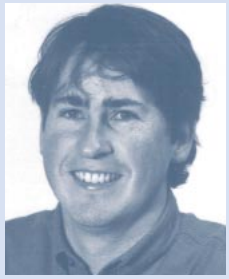
mehr zum thema:
www.syngo.com
www.siemensmedical.com

PROJEKTMANAGEMENT MIT REQUIREMENTS-ENGINEERING UND UML IM GRIFF

Die LFK GmbH entwickelt komplexe Hard/Software-Systeme, die aus mehreren eigenständigen Systemen bestehen. Der Artikel beschreibt, wie die Systeminformationen eines Requirements-Engineering und einer Systemmodellierung mit der UML für das Projektmanagement genutzt werden. Die Informationen des Requirements-Engineering, das UML-Modell und Zusatzartefakte werden zu einer übergreifenden Projektdatenbank verknüpft. Wie diese für die entwicklungsbegleitenden Aktivitäten – wie Angebotserstellung, Planung, Risikomanagement, Nachweisführung, Qualitätssicherung und Konfigurationsmanagement – genutzt werden kann, wird im Einzelnen aufgezeigt.

die autoren

Dr. Rudolf Hauber (E-Mail: RHauber@ka-muc.de) ist langjähriger Berater bei der Kölsch & Altmann Software & Management Consulting GmbH und spezialisiert auf OO-Methoden und Tooleinsatz.



Dieter Wagner (E-Mail: dieter.wagner@lfk.eads.net) ist Modulteamleiter bei der Firma LFK GmbH und arbeitet gemeinsam mit seinem Kollegen Thomas Rittel (thomas.rittel@lfk.eads.net) an der Entwicklung von Lenkflugkörpern

KORREKTURFAHNE

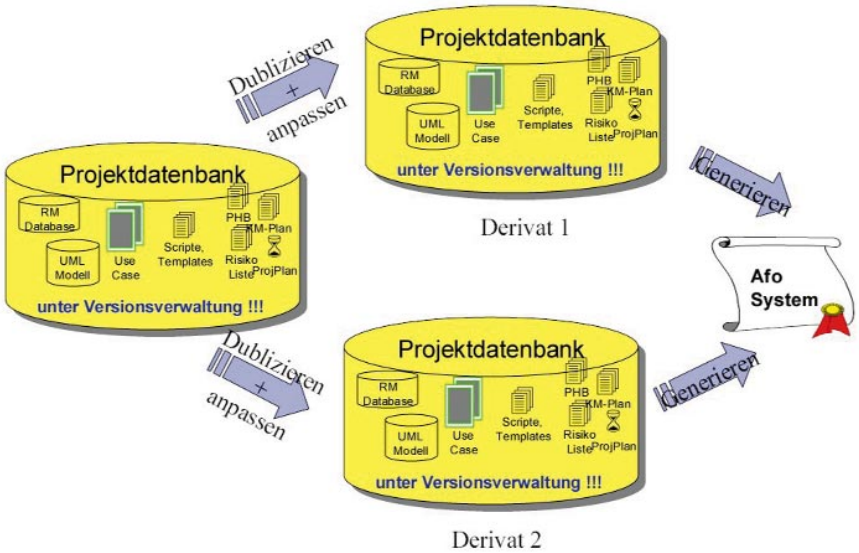
Motivation

Bei der Lenkflugkörper Systeme GmbH (LFK) – gehört zum EADS-Konzern – werden äußerst komplexe Systeme entwickelt, die in Realzeit große Datenmengen verarbeiten müssen,. Dies macht neuartige Entwicklungsansätze erforderlich. Sowohl Probleme bezüglich der Datenkonsistenz als auch externe Einflüsse, wie z. B. Änderungswünsche der Kunden, sind zu bewältigen. Dabei ist die Lösung vieler dieser Probleme nicht unbekannt, sie wird nur häufig nicht angewandt, weil Berührungsängste mit neuen Techniken und Vorgehensweisen bestehen. Bei der Entwicklung eines neuen Lenkflugkörpers wurde basierend auf einem angepassten Standardvorgehensmodell (V-Modell 97, vgl. [VM97]) in Zusammenarbeit mit zwei externen Firmen ein integriertes *Requirements-Engineering (RE)* und eine Systemmodellierung mit der *Unified Modeling Language (UML)* aufgesetzt. Zur Vermeidung einer dokumentenlastigen Vorgehensweise wurde eine integrierte Projektdatenbank aufgebaut, die alle Entwicklungsinformationen enthält und die Grundlage der Systementwicklung bildet.

Wie im Rahmen des RE und der Modellierung in UML komplexe Systeme effizient erfasst und bearbeitet werden können und eine Projektdatenbank entsteht, wurde in [Hau01] gezeigt. Dabei wurden die Tools der Rational-Suite („RequisitePro", „Rose", „TestManager", „SoDA") verwendet. Ergebnis dieser Methode ist, dass eine Projektdatenbank aufgebaut wird, die aus einer Vielzahl miteinander verknüpfter Informationen (RequisitePro-Datenbank für RE-Informationen, Rose-UML-Modell, Use-Case-Beschreibungen, Testfallbeschreibungen usw.) besteht. Im Folgenden wird erläutert, wie diese vorteilhaft für die weiteren Projektaktivitäten – Projektmanagement, Qualitätssicherung und Konfigurationsmanagement – genutzt werden kann.

Angebotserstellung/Derivate

Zu Beginn jedes Projekts kommt es zu einer Angebotserstellung. Für diese wird ein *Anwenderforderungsdokument* (im V-Modell Jargon *Afo* genannt) erstellt, das die vom Auftraggeber und Auftragnehmer akzeptierten Anforderungen enthält. Die Akzeptanz der Afo kann durch die natürlichsprachliche Formulierung der Anforderungen in Form von Anwendungsfällen (Use-Cases) erhöht werden, da Use-Cases die Anforderungen aus Benutzersicht beschreiben. Eine grafische Darstellung



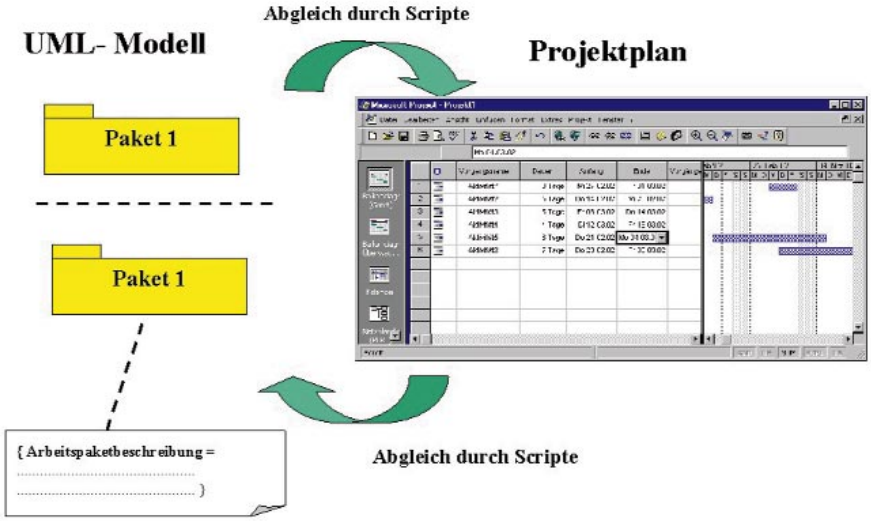


Abb. 2: Abgleich von Modell und Planung

von deren Zusammenhängen mittels Zustandsdiagrammen kann das gemeinsame Verständnis weiter verbessern. Leistungskenngrößen, Randbedingungen und Umweltbedingungen, unter denen das Produkt seine Funktionalität erbringt, sind ebenfalls eindeutig System-Use-Cases oder speziellen Systemteilen zuzuordnen.

Risk-Id	Beschreibung	Risikograd	Maßnahme	Status	Bearbeiter	...
4711	Es besteht die Gefahr, dass....	Hoch	Grenzwert Analyse	Erkannt	Maier	

Tabelle 1: Einfaches Risikomanagement

Dieses Vorgehen lässt sich auf alle Subsysteme des Produkts rekursiv anwenden. Dadurch wird ein durchgängiges *Tracing* der Anforderungen aufgebaut.

Da die Inhalte der Afo in der Projektdatenbank enthalten sind, lassen sich insbesondere für Produktfamilien leicht Varianten und Derivatprodukte ableiten. Durch einfache Duplizierung und Anpassung der Projektdatenbank können abgeleitete Anforderungsdokumente und andere Artefakte (V-Modell Produkte) für Produktvarianten leicht gewonnen werden (**siehe Abb. 1**).

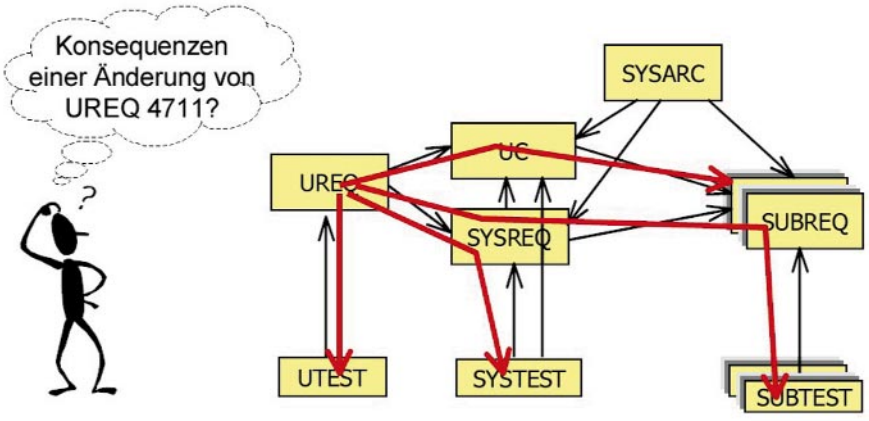
Darüber hinaus erlaubt das in der Projektdatenbank aufgebaute *Tracing* der Anforderungen die Auswirkung von Anforderungsdeltas genau zu bestimmen und besser zu bewerten.

Dieses Vorgehen wurde bereits erfolgreich bei der LFK für die Produktfamilie eines Lenkflugkörpers durchgeführt.

Arbeitspaketbeschreibung/ Planung

Bei der weiteren Projektbearbeitung werden durch Dekomposition und Verfeinerung die Anforderungen des Systems mit den Beschreibungsmitteln der UML (Paketen, Klassen, Interaktionen, Knoten, Prozesse etc.) heruntergebrochen. Die gefundenen Modellelemente dienen nicht nur als Grundlage für die Realisierung, sondern auch als Basis von Arbeitspaketbeschreibungen und dazugehörigen Planungen (insbesondere bei der Trennung zwischen Hard- und Software-Einheiten).

Bei der Erzeugung der Entwicklungsaktivitäten pro Dekompositionsstufe können die Arbeitspaketbeschreibungen in die Planung übernommen werden, wobei sich die Planungsgenauigkeit bei jeder Stufe verbessert.



Um dieses Vorgehen zu automatisieren, wurden für das Modellierungs-Tool „Rational Rose" Skripte erzeugt, die einen Abgleich mit dem Planungstool „MS Project" bereitstellen (**vgl. Abb. 2**).

Mit MS Project werden die Aktivitäten in zeitliche Beziehung zueinander gesetzt und mit Ressourcen hinterlegt. Die so aktualisierten Planungsdaten werden ebenfalls über Skripte in Rational Rose zurückgespielt und den Aktivitäten zugeordnet. Aus den UML-Beschreibungen der Aktivitäten und den Planungsdaten können somit automatisch über einen Dokumentationsgenerator („Rational SoDA") Arbeitspaketbeschreibungen erzeugt werden.

Risikomanagement

Aufgabe des Risikomanagements ist es Risiken zu identifizieren, zu bewerten und risikominimierende Maßnahmen einzuleiten. Auch hier kann aus der Systemmodellierung Nutzen gezogen werden. Dazu wird für jede funktionale und nichtfunktionale Anforderung im Rahmen des RE eine Kritikalität bestimmt. Über die vorhandenen Links können die betroffenen Use-Cases und die zu realisierenden Subsysteme bestimmt werden. Über Dekomposition und Verfeinerung lassen sich die einzelnen Risiken über die Hard/Software-Einheiten bis auf Hardware-Komponenten und Software-Klassen verfolgen. In einem ersten Schritt können die Kritikalitäten direkt im *Requirements-Management-Tool (RM-Tool)* – in diesem Fall „RequisitePro" – verwaltet werden.

Auf diesen Erkenntnissen lässt sich ein Risikomanagement (**vgl. Tabelle 1**) aufsetzen, um die kritischen Komponenten des Systems zu bestimmen und dementsprechend zu behandeln. Um eine sinnvolle Risikobewertung durchführen zu können, ist darauf zu achten, dass der Risiko-

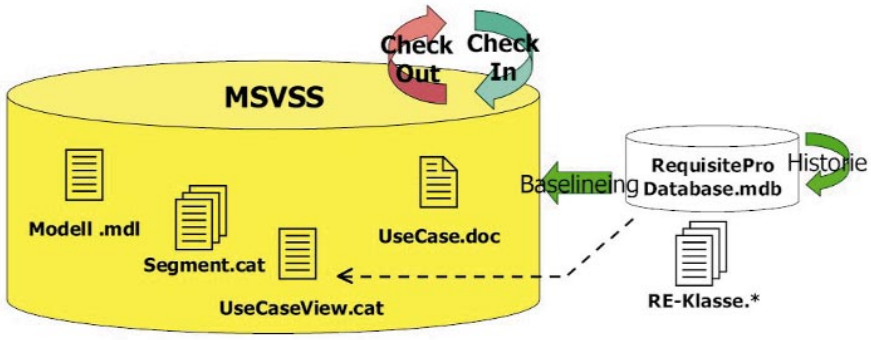


Abb. 4: Konfigurationsmanagement im Projektdatenbank

manager seine Aufgaben unabhängig vom Projekt- und Entwicklungsteam durchführen kann.

Durch die *Tracebility* lassen sich Änderungen der Anforderungen direkt auf eventuelle Entwicklungsrisiken hin untersuchen und ihre Auswirkung auf einzelne Systemteile automatisch ermitteln (**vgl. Abb. 3**).

Die Ergebnisse der Risikoanalyse unterstützen einen inkrementellen Entwicklungsprozess, wobei das Produkt in Inkremente aufgeteilt wird, um nach der Regel „das Schwierigste zuerst" bei der Entwicklung vorzugehen.

Test

Alle Anforderungen an das System, die bereits im RM-Tool RequisitePro und mittels Use-Cases erfasst sind, müssen durch geeignete Tests nachgewiesen werden (**siehe Abb. 3**). Dieser Vorgang muss über alle Stufen der Dekomposition und Verfeinerung durchgeführt werden. Die Nachweise können ebenfalls im RM-Tool verwaltet (**Abb. 4**) und mit den Anforderungen und Use-Cases verknüpft werden. Das ermöglicht, später eine Testabdeckung sicherzustellen. Wichtig ist hierbei, dass pro Stufe zeitgleich zu den Anforderungsdefinitionen die zugehörigen Akzeptanzkriterien definiert werden. Die Testverfahren müssen wie die Anforderungen mittels Engineering-Methoden erstellt werden, damit für das Produkt ein durchgängiges, vollständiges und in sich geschlossenes Testkonzept entsteht (Kundenakzeptanztest gegen Systemanforderungen, Systemtest gegen Use-Cases, Subsystemtest gegen Subsystem-Use-Cases, Komponententest, Klassentest). Die Verwaltung dieser Informationen kann im RM-Tool oder einem eigenen Testmanagement-Werkzeug (z. B. „Rational Test-Manager") erfolgen, sollte jedoch in jedem Fall ein entsprechendes *Tracing* unterstützen (**vgl. Tabelle 2**). Aus diesen Daten können durch einen Dokumentationsgenerator (wie z. B. Rational SoDA) Prüfpläne und Prüfspezifikationen automatisch generiert werden. Zudem kann die Abdeckung der Anforderungen durch die Nachweise automatisch überprüft werden und gegebenenfalls Diskrepanzen aufgezeigt werden (**vgl. Abb. 5**).

Test-Id	Kurzbeschreibung	Beschreibung	Hinweise	Status	Tester	...
4711	Missionsanlegung	Es sind 3 Missionen anzulegen, die ...	Beachten, dass nur ...	korrekt	Maier	

Tabelle 2: Einfaches Testfallmanagement

Qualitätssicherung

Vollständigkeit gewährleistet

Durch die automatische Generierung der Dokumente ist gewährleistet, dass immer die aktuellen Informationen der Datenbanken – z. B. Randbedingungen – in den unterschiedlichsten Dokumenten vorhanden sind. D. h. alle projektspezifischen Informationen sind nur an einem Ort abgelegt, wodurch die Datenkonsistenz automatisch gewahrt wird. Ebenso wird die Vollständigkeit durch Skripte im Rational Rose und RequisitePro überprüft. Nur bei einer syntaktisch korrekten Modellierung sollten die zugehörigen Dokumente erzeugt werden.

Zusätzlich werden Reibungsverluste, die durch den Abgleich von Anforderungen zwischen den Dekompositionsstufen entstehen, wie sie in der Vergangenheit üblich waren, komplett vermieden.

Produktkonformität

Durch die durchgängige Verwendung der standardisierten Beschreibungssprache UML und den Einsatz von Dokumenten-Templates werden die Lesbarkeit und dadurch das Verständnis des Systemmodells und damit der Dokumente deutlich verbessert. Es entsteht eine einheitliche Form von Dokumenten innerhalb des Projekts, ja sogar innerhalb der Firma.

Das vereinfacht den Aufwand für die Qualitätssicherung erheblich, da die Dokumente allein schon auf Grund ihrer formalen Korrektheit einfacher zu lesen und zu verstehen sind. Dieses Vorgehen verhindert, dass – wie in der Vergangenheit – jeder Entwickler die Dokumentenstruktur anders interpretiert.

Ferner erhält die Projektdatenbank (und damit Dokumentation) eine deutlich bessere Qualität, wenn durch automatische, skript-basierte Konsistenzprüfungen die Vollständigkeit der Modelle und Einhaltung der Modellierungsrichtlinien überprüft wird. Typische Prüfungen sind:

- Entspricht die Subsystemkommunikation den geplanten Kommunikationskanälen?
- Sind die Anforderungen vollständig Subsystemen zugewiesen?
- Sind alle Klassen/Pakete etc. dokumentiert?
- Existieren Tests für jede Anforderung?

Eine Konsistenz der Projektinformationen ist bei der Komplexität der besprochenen Systeme anders nicht mehr sicherzustellen.

Auch sollte rechtzeitig daran gedacht werden, Entwicklungsmaßzahlen für das höhere Management bereitzustellen. Die Aussagen über die Anforderungsstabilität lassen sich leicht gewinnen, indem beispielsweise die Anzahl der Anforderungen und Use-Cases und ihrer Änderungen über die Zeit mitverfolgt wird. Genauso sollten als Metriken für die Designstabilität, die Anzahl der Pakete und Klassen und ihre Änderungen über die Zeit festgehalten werden. Die Tools bieten diese Informationen auf Knopfdruck. Über solche Metriken lassen sich wichtige

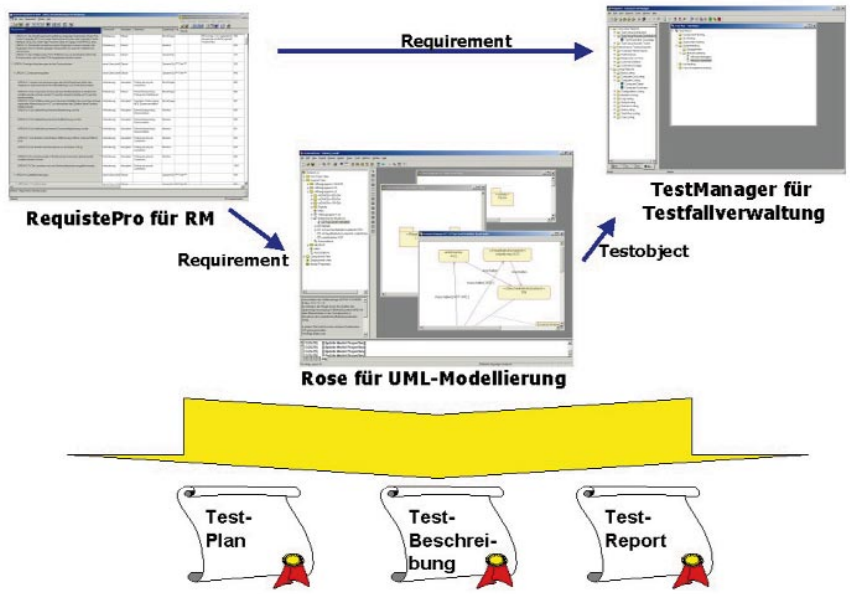


Abb. 5: Testkonzept mit tool-übergreifenendem Informationsfluss

Aussagen über den Stand von Projekten gewinnen.

Konfigurationsmanagement

Bei modellgestütztem Arbeiten reicht es nicht aus, die Dokumente unter Versionsverwaltung zu stellen, sondern es sind alle Inhalte der Gesamtprojektdatenbank zu versionieren. Im vorliegenden Fall wurde die Entwicklung mit den Tools der Rational-Suite durchgeführt. Um einen konsistenten Stand der Gesamtprojektdatenbank zu sichern, muss eine Baseline gezogen werden. Die Baseline für eine Afo-Erstellung umfasst dabei beispielsweise die RequisitePro-Datenbank, alle Rose-Modell-Dateien, die Use-Case-Beschreibungen inklusive Bildern und Anhängen, die Prüfskripte, das SoDA-Dokumenten-Template und die zugehörige Modellierungsrichtlinie.

Dadurch können Änderungen genau zurückverfolgt werden. Ferner kann durch geeignete Rollendefinitionen im Projekt mit entsprechenden Zugriffsberechtigungen auch eine größere Sicherheit über die Systeminformationen gewonnen werden.

Darüber hinaus ist zu erwähnen, dass bei modell-basiertem Arbeiten Wiederverwendung auf einer viel höheren Stufenleiter einsetzen kann. Ganze Subsysteme oder Architekturkonzepte lassen sich mittels Copy&Paste in andere Systeme übernehmen und im Design anpassen. Die Implementierungssprache kann dabei unterschiedlich sein. Hier kann das Konfigurationsmanagement durch Modellkataloge existierende Systemmodelle anderen Projekten verfügbar machen.

Zusammenfassung

Ein konsequentes RE und eine Systemmodellierung mit der UML erlaubt heutzutage dank leistungsfähiger Werkzeuge ein modell-basiertes Entwickeln, das auch sehr komplexe Systeme beherrschbar macht. Die dabei gewonnenen Informationen können zu einer Projektdatenbank verknüpft werden, die der zentrale Informationsspeicher des Projekts ist, was gerade bei langlebigen Systemen (mehr als 10 Jahre) von zentraler Bedeutung ist.

Eine solche Projektdatenbank kann jedoch neben dem immensen Vorteil für die Systementwicklung auch noch erheblichen „Kollateral"-Nutzen für das Projektmanagement, die Qualitätssicherung und das Konfigurationsmanagement bieten. Wer darauf verzichtet, verschenkt viel. Jedoch muss darauf hingewiesen werden, dass dieser Ansatz insbesondere dann sinnvoll ist, wenn ähnliche Systeme über lange Zeit in derselben Domäne entwickelt werden. Für ein Softwarehaus, das kurzlebige E-Business-Projekte entwickelt, ist ein architekturzentrierter Ansatz (vgl. [Sta02]) interessanter, da dieser domänen-zentrierte Ansatz sicherlich einen zu hohen Initialaufwand erfordert. Für die meisten größeren Firmen ist diese Domänenzentrierung jedoch gegeben (siehe z.B. [Dor02]) und der Initialaufwand auf die Dauer gesehen gering.

Ferner ist entscheidend, dass der Informationsbestand dauerhaft gepflegt wird. System-Engineering bedeutet auch, die RE-Datenbank, das UML-Modell, die sonstigen Informationsquellen sowie ihre Verknüpfungen konsequent fortzuschreiben, da sonst Inkonsistenten auftreten können und ein gewonnenes Artefakt keinen zuverlässigen Stand besitzt. Aus Aufwandsgründen ist es wichtig, die Tools nicht mit Zusatzinformationen zu überfrachten, damit eine Chance besteht, die Daten auch bis zum Ende der Entwicklung zu pflegen. ■

Literatur

[Can98] M. Cantor, Object-Oriented Project Management with UML, Wiley, 1998

[Dor02] K. Dorn, Vom Programmieren zum Konfigurieren von Software, in: OBJEKTspektrum 1/02

[Hau01] R. Hauber, M. Reinhold, T. Rittel, D. Wagner, Große Systeme in den Griff bekommen, in: OBJEKTspektrum 5/01

[Oes01] B. Oestereich, P. Hruschka, M. Reinhold, N. Josuttis, H. Kocher, H. Krasemann, Erfolgreich mit Objektorientierung: Vorgehensmodelle und Managementpraktiken für die objektorientierte Softwareentwicklung, Oldenbourg-Verlag, 2001

[Rei97] M. Reinhold, UML und das standardisierte Prozessmodell „V-Modell '97": Warum reicht eine Modellierungssprache alleine nicht aus?, in: OBJEKTspektrum 5/97

[Sta02] T. Stahl, Generative Softwareentwicklung in der Praxis, in: OBJEKTspektrum 1/02