12. STEV-Österreich-Fachtagung

Wien, 7. März 1997

Ein Vorgehensmodell für objektorientierte Software-Entwicklung

- ein Erfahrungsbericht

Zusammenfassung
Ausgehend vom im Unternehmen bekannten einfachen Vorgehensmodell für die klassische Software-Entwicklung (Wasserfallmodell) wird die Entwicklungsgeschichte eines Vorgehensmodells für objektorientierte Softwareentwicklung geschildert.
Im Detail wird auf die eingesetzten Methoden bei der Vorgehensmodellentwicklung selbst eingegangen. Die Unterschiede zwischen den beiden Vorgehensmodellen werden diskutiert. Die mit dem oo-Vorgehensmodell gemachten Erfahrungen in der konkreten Projektarbeit werden ebenso behandelt wie Weiterentwicklungspotentiale und Zukunftsperspektiven für den Einsatz von oo-Vorgehensmodellen in der qualitätsgesicherten Softwareentwicklung.

Autor
Dipl.-Ing. Angelika Mittelmann
VOEST-ALPINE STAHL LINZ GmbH
Tel: 0732-6585-9159
eMail: angelika.mittelmann@ps.stahl.voest.ada.at

Inhalt

Vorbemerkungen

Die VOEST-ALPINE STAHL LINZ GmbH ist eines der größten Industrieunternehmen Österreichs. Im integrierten Hüttenwerk werden kalt- und warmgewalzte, organisch beschichtete und verzinkte Flachwalzware sowie Schmiede- und Gießereiprodukte hergestellt. Alle Verarbeitungsvorgänge vom Roheisen bis zum fertigen veredelten Produkt werden am Betriebsstandort selbst durchgeführt. Die VOEST-ALPINE STAHL LINZ GmbH ist Zulieferer für die Automobil-, Hausgeräte- und Bauindustrie.

Projektinhalt

Das Projekt im Rahmen dessen das nachfolgend beschriebene Vorgehensmodell entwickelt wurde, beschäftigte sich mit der Entwicklung eines hüttenweiten Energieüberwachungs- und -prognosesystems (*EMS*). Dadurch wurde die Basis für eine effizientere Energiebewirtschaftung geschaffen. Da bei der Entwicklung dieses Informationssystems objektorientierte Analyse und Design sowie objektorientierte Implementierungstechniken auf der Basis der Programmiersprache C++ zur Anwendung kommen sollten und die Projektmitarbeiter einen Erfahrungszuwachs in objektorientierten Methoden, Werkzeugen und Sprachen erhalten sollten, wurde dieses Projekt bei der Kommission der Europäischen Gemeinschaften im Rahmen von ESSI (European Systems and Software Initiative) als "Application Experiment" (AE) eingereicht und akzeptiert (Projekt 10024/EMS).

Durch die Aufgabenstellung das EMS als Application Experiment zu führen, ergaben sich zwei getrennte Projektebenen mit unterschiedlichen Aufgabengebieten. Auf der einen Seite mußte die möglichst gute Implementierung eines Energieüberwachungs- und -prognosesystems gelöst werden. Um eine neue Technologie wie die Objektorientierung in den Softwareentwicklungsprozeß (SE-Prozeß) erfolgreich einzuführen, ist es auf der anderen Seite erforderlich, den SE-Prozeß an sich kritisch zu durchleuchten und identifizierte Problembereiche zu verbessern. Da die Verbesserung des SE-Prozesses eine komplexe Aufgabe darstellt, mußte also hier ein Vorgehensmodell gefunden werden, das dieser Aufgabenstellung gerecht wird. Das ami-Vorgehensmodell wurde aufgrund seiner spezifischen Eigenschaften ausgewählt. Im Gegensatz zu anderen Vorgehensmodellen weist es sowohl statische (Assessment, Metriken) als auch dynamische Elemente (Reifegrad-abhängige Zieldefinition, zyklische Wiederholung der Phasen) auf, die einen kontinuierlichen, gut steuerbaren Verbesserungsprozeß in der Softwareentwicklung adäquat unterstützen.

Parallel zur Vorprojektphase des Implementierungsteils des EMS-Projektes wurde daher mit die Umsetzung des ami-Vorgehensmodells (siehe [ami], [Basili/Rombach88], [Mittelmann96], [SEI93]) für die gesamte, mit der Entwicklung des EMS-Systems betrauten Organisationseinheit der VOEST-ALPINE STAHL LINZ GmbH beschlossen.

Partner

Das Projektteam rekrutierte sich im wesentlichen aus drei Quellen: der VOEST-ALPINE STAHL LINZ GmbH, dem VOEST-ALPINE Industrieanlagenbau und dem Christian-Doppler-Labor für Software-Engineering (CDL-SE). Eine der Hauptaufgaben des CDL-SE ist neben Grundlagenforschung der Know-How-Transfer auf dem Gebiet der objektorientierten Technologien zu allen Firmen, die Mitglieder der Christian-Doppler-Gesellschaft sind.

Projektorganisation

Die Projektleitung wurde von je einer Person der beiden erstgenannten Partner wahrgenommen. Der dritte Partner war vor allem in die Analyse-, Design- und Implementierungsphase des EMS eingebunden, um einen möglichst guten Know-How-Transfer in den objektorientierten Technologien zu gewährleisten. Für die Durchführung der ESSI-AE-spezifischen Aufgaben war eine Person ganztags beschäftigt.

Entwicklung unseres objektorientierten Vorgehensmodells

Kurzbeschreibung des Ausgangsmodells

Zum Verständnis des nachfolgend beschriebenen Vorgehensmodells ist es notwendig, einige Begriffsklärungen vorzunehmen.

Begriffsklärungen

Die erforderlichen Geschäftsprozesse in der Softwareentwicklung orientieren sich am *Lebenszyklus* eines Informationssystems. Dieser umfaßt die zwei Teilbereiche *Erstellung* und *Betrieb* des Informationssystems. Der Hauptprozeß, der die Erstellung eines Informationssystems zum Ziel hat, ist der *Software-Entwicklungsprozeß* (SE-Prozeß). Er umfaßt alle notwendigen Aktivitäten für die Produktion eines Informationssystems.

Der SE-Prozeß selbst wird durch den *Projektmanagement-Prozeß* (PM-Prozeß) gesteuert. Er umfaßt alle Aktivitäten, die für die reibungslose Abwicklung des SE-Prozesses im Unternehmen notwendig sind.

Der Betrieb eines Informationssystems wird neben dem eigentlichen Betriebsprozeß durch den *Konfigurationsmanagement-Prozeß* (KM-Prozeß) begleitet. Er umfaßt alle Aktivitäten, die sowohl der unmittelbaren Fehlerbehebung als auch der notwendigen Funktionserweiterungen des Informationssystems dienen.

Beide Teilbereiche des Lebenszyklus werden durch den *Qualitätssicherungsprozeß* (QS-Prozeß) begleitet. Er stellt sicher, daß die geforderten Qualitätsvorgaben bei der Erstellung des Informationssystems erreicht und beim Betrieb aufrechterhalten werden.

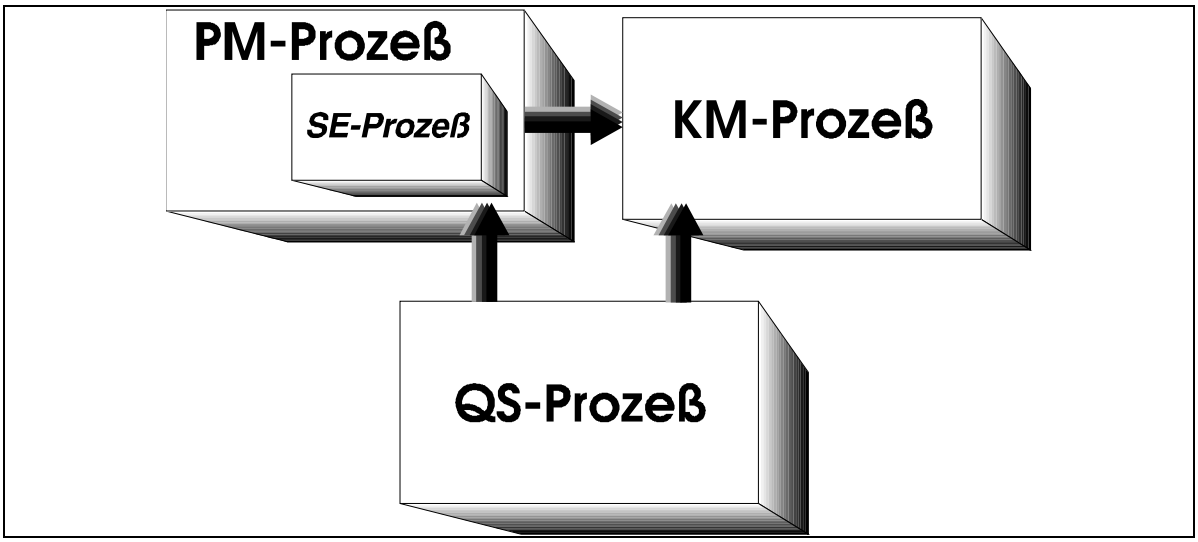

Abbildung 1. Geschäftsprozesse in der Softwareentwicklung

Das klassische SE-Prozeßmodell

Die nachfolgende Grafik beschreibt das klassische "*Wasserfallmodell*" des SE-Prozesses. In der Praxis wird der Prozeß nicht so idealtypisch ablaufen, sondern manche Phasen können ineinander verschachtelt sein bzw. der Rücksprung kann auch über mehrere Phasen erfolgen, z.B. wenn sich in der Fertigung herausstellen sollte, daß ein Analysefehler gemacht wurde.

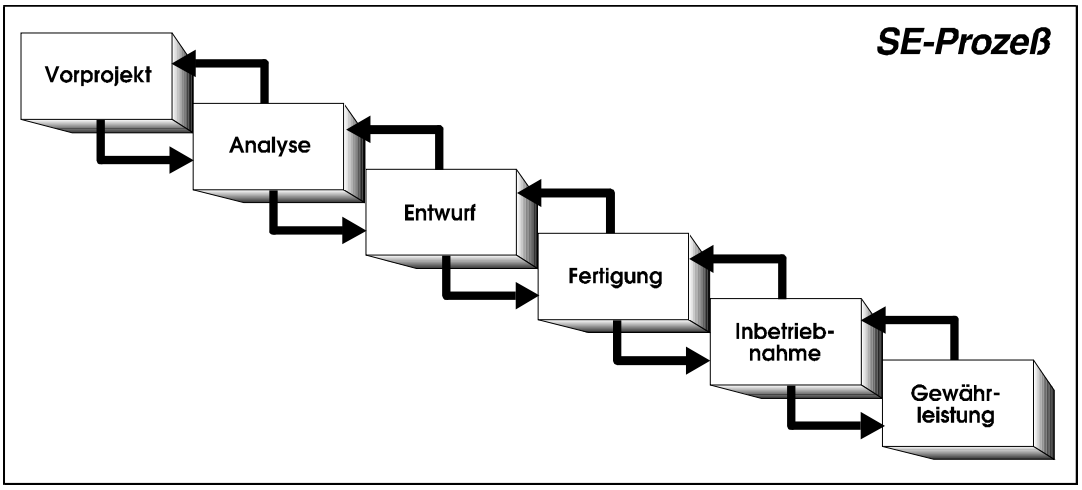

Abbildung 2. Phasen des klassischen SE-Prozesses

Im Rahmen des SE-Prozesses kann auch Prototyping zum Einsatz kommen. *Prototyping* ist eine Methode des Software-Engineering, die es erlaubt, bereits in einer sehr frühen Phase des SE-Prozesse, Benutzeranforderungen rasch in lauffähigen Code abzubilden. Verständnisprobleme zwischen Benutzern und Entwicklern können durch Einsatz dieser Methode oft verhindert oder rasch ausgeräumt bzw. Systemarchitekturprobleme in der Analyse- und vor allem in der Entwurf-Phase einfacher gelöst werden.

Abhängig davon, welches Ziel bei der Prototyp-Erstellung verfolgt wird, entsteht ein *wiederverwendbarer Prototyp*, der in den Nachfolgephasen entsprechend verfeinert

wird, oder ein *Wegwerf-Prototyp*, der nach Klärung der strittigen Punkte nicht mehr weiterverwendet wird.

Kurzbeschreibung des objekt- und prototyping-orientierten Vorgehensmodells

Da das objektorientierte Paradigma ein prototypisches Vorgehen bei der Software-Entwicklung geradezu aufdrängt, wird im folgenden nur auf das objekt- *und* prototyping-orientierte Prozeßmodell eingegangen.

Als Methode wird die "*Object Modelling Technique*"-Methode von *Rumbaugh* [Rumbaugh93] mit der Erweiterung der Methode der *use cases* (Anwendungsfälle), die von *Jacobson* [Jacobson94] entwickelt und angewandt wurde, verwendet. Als *Entwicklungswerkzeug* wird derzeit *Select OMT* benutzt.

Da Erfahrungswerte noch weitgehend fehlen, wird das bereits definierte Prozeßmodell zugrunde gelegt. Die oben definierten Phasen des SE-Prozesses bleiben aufrecht. Die Aktivitäten und Ergebnisse je Phase werden neu definiert.

Vorgehensweise bei der Modellentwicklung

Da das objektorientierte Vorgehensmodell für die Software-Entwicklung bei der Anwendung des ami-Vorgehensmodells entstanden ist, wird im folgenden kurz darauf eingegangen.

Umsetzungschritte des ami-Vorgehensmodells

Das ami-Vorgehensmodell beruht auf einem rein quantitativen Ansatz und ermöglicht wertneutral die Erreichung der relevanten Unternehmensziele und die ständige Verbesserung der Geschäftsprozesse im Rahmen der Software-Entwicklung.

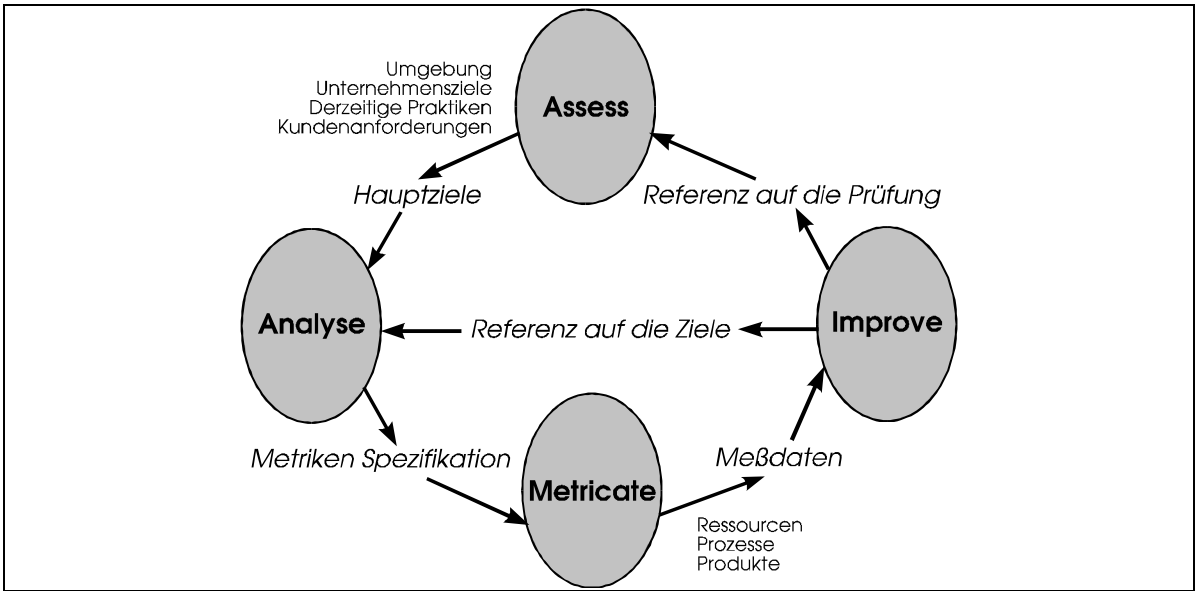

Abbildung 1. **Das ami-Vorgehensmodell**

Gestartet wurde mit der Informationsveranstaltung "*Was ist ami und warum ami?*". Anschließend wurden die Erwartungen der Teammitglieder in das Projekt erhoben. Ebenso wurde auch notiert, was jeder einzelne zum *Scheitern* des Projekts beitragen könnte, um im Team eine realistische Erwartungshaltung zu erreichen und das Problembewußtsein zu steigern. Diese Liste wurde im Verlauf des Projekts bei Bedarf herangezogen.

Das Selbst-Assessment der Organisation mit Hilfe des SEI-Fragebogens [SEI93] führte zur Identifikation der organisationsspezifischen Problembereiche. Große Unterschiede in Erfahrung und im Wissen der einzelnen Mitarbeiter im Software-Engineering und keine detaillierten Aufzeichnungen über den Verlauf abgeschlossener Projekte stellten sich als die vorrangigsten Probembereiche heraus.

Um die Kenntnisse der einzelen Teammitglieder über die Geschäftsprozesse in der Softwareentwicklung auf ein gleiches und möglichst hohes Niveau zu bringen, wurde sofort mit periodischen *Tutorien* zu den Themen "Softwareentwicklungsprozeß", "Projekt-Management", "Konfigurationsmanagement" und "Qualitätssicherung" begonnen. In diesen Turorien wurde der Diskussion der vorgestellten Konzepte viel Zeit eingeräumt. Die Auswirkungen der objektorientierten Technologie fanden dabei besondere Berücksichtigung.

Parallel dazu wurden die Hauptziele, die sich aus den identifizierten Problembereichen ergaben, operationalisiert und Metriken zu jedem Teilziel spezifiziert sowie erste Messungen durchgeführt. Die Ergebnisse der Messungen wurden zu den definierten Zielen in Beziehung gesetzt und Maßnahmen zur Umsetzung beschlossen.

Als eine der ersten und wichtigsten Maßnahmen wurde die Erstellung von erforderlichen Verfahrenshandbücher für den Softwareentwicklungsprozeß umgesetzt, d.h. vor allem die Entwicklung eines Vorgehensmodells für die objektorientierte Software-Entwicklung (oo-SE).

Entwicklungsschritte des Vorgehensmodells für die objekt-orientierte Software-Entwicklung

Ein im Rahmen des EMS-Projekts vereinbartes Grundprinzip war die strikte Teamorientierung. Daher wurde auch das Vorgehensmodell unter Beteiligung aller Projektmitarbeiter erarbeitet.

Erste Teambesprechung

In der ersten Teambesprechung zu diesem Thema wurde in einer regen Diskussion erhoben, welche Informationen zu diesem Thema dem Team zur Verfügung stehen und die weitere Vorgangsweise mit Terminplan verbindlich festgelegt. Es wurde beschlossen als Ausgangspunkt, das im zentralen IS-Bereich verwendete Vorgehensmodell für die klassische Software-Entwicklung als Basis zu verwenden.

Wissensakquisition

Die Dokumentation des oben erwähnten Vorgehensmodells wurde jedem Teammitglied zur Verfügung gestellt mit der Bitte, es im Detail durchzuarbeiten und dabei Bemerkungen, Ergänzungen, aber auch Fragen zu notieren.

Interviewphase

Nach dieser Einarbeitungsphase wurde mit jedem Teamitglied ein Einzelinterview geführt, um seine Ergänzungen zu besprechen und seine Fragen zu beantworten. Alle Interviewergebnisse wurden zu einem Ergebnisbericht zusammengeführt und in der nächsten Teambesprechung zum Thema präsentiert.

Erstellung der Erstfassung

Auf Basis des Ergebnisberichtes wurde eine Erstfassung des Vorgehensmodells erstellt. Diese wurde von allen Teammitgliedern Korrektur gelesen und in einer weiteren Teambesprechung zur allgemeinen Benutzung freigegeben. Um die Benutzung möglichst zu erleichtern, wurde aus dem Dokument eine Online-Version (Windows-Helpfile) generiert und auf dem Projektserver verfügbar gemacht.

Es wurde auch vereinbart, das Vorgehensmodell entsprechend dem Projektfortschritt zu ergänzen und weiterzuentwickeln entsprechend dem Wissenszuwachs in der objektorientierten Technologie. D. h. die Erstfassung war in großen Teilen mit dem Ausgangsmodell ident. Unterschiede waren nur dort festzustellen, wo entweder unser Projektpartner (CDL-SE) spezielle Ergänzungen und/oder oo-spezifische Erweiterungen einbrachte oder aufgrund praktischer Erfahrungen einzelner Teammitglieder Änderungen als sinnvoll erachtet wurden.

Ergänzung von Dokumentvorlagen

Da im Vorgehensmodell in verschiedenen Phasen bestimmte Dokumentarten als Ergebnis oder zur Prozeß-Unterstützung beschrieben sind, wurden Dokumentvorlagen erstellt und ebenfalls auf dem Projektserver verfügbar gemacht.

Die äußere Form der Ergebnisdokumente für z.B. Analyse- und Design-Dokumente wurde in einem eigenen Standard beschrieben und als allgemeine Dokumentvorlage bereitgestellt. Inhaltliche Festlegungen sind entweder direkt im Vorgehensmodell enthalten oder in einem weiteren Richtlinien-Dokument festgehalten (z.B. Dokumentationsrichtlinien für Klassen).

Für Prozeß-unterstützende Dokumente wie Änderungsanforderungen, Problemberichte oder Metriken-Erfassung wurden Dokumentvorlagen mit entsprechenden Dateneingabemasken entwickelt.

Weiterentwicklung des Vorgehensmodells

Entsprechend dem Projektfortschritt in der Software-Entwicklung wurden periodische Teambesprechungen durchgeführt mit dem Ziel,

Modellvergleich

Phase "Voruntersuchung"

Im Rahmen eines Vorprojekts wird das zu lösende Problem grob analysiert und Lösungsalternativen werden aufgezeigt. Die Anforderungen der Benutzer werden struktu-

riert und die Zielsetzungen des zu fertigenden Software-Produktes festgelegt. Anhand der Ergebnisse wird über die Weiterführung bzw. den Abbruch des Projekts entschieden.

Der Hauptunterschied zum "klassischen" Vorgehensmodell ist in der Beschreibung des fachlichen Lösungsansatzes zu finden. Dieser enthält statt Hauptfunktionen und Entitätstypen die elementaren Anwendungsfälle und ein erstes, einfaches Objektmodell.

Phase "Analyse"

In der Analyse-Phase wird in sehr detaillierter Form festgelegt, **WAS** das Software-Produkt leisten soll. Die fachlich erforderlichen Klassen samt Methoden, die sie umgebende Ablauforganisation und die Hard- und Software der Zielumgebung werden festgelegt. Die Darstellung erfolgt in Modell- und Prototypform.

Der zunächst ins Auge fallende Unterschied in der Ergebnisdarstellung ist die Forderung nach einem Benutzeroberflächen-Prototypen ergänzend zum Dokument *Leistungsbeschreibung*. Diese Forderung ist an sich kein objektorientiertes Spezifikum, da Benutzeroberflächen-Prototypen auch in einem klassischen Vorgehensmodell gefordert und mit Hilfe von klassischen Mitteln erzeugt werden können. Im vorliegenden Fall wird von einem mit objektorientierten Mitteln produzierten Prototypen ausgegangen.

Die Hauptunterschiede finden sich allerdings im Inhalt der Leistungsbeschreibung. Statt eines detaillierten Funktionen- und Datenmodells wird ein Objektmodell (Objektmodelldiagramm + Data Dictionary), ein Dynamisches Modell (Zustandsdiagramme + globales Ereignisflußdiagramm) und ein Funktionales Modell (Datenflußdiagramme + Einschränkungen) gefordert.

Phase "Entwurf"

In der Entwurfsphase wird festgelegt, **WIE** die DV-technische Umsetzung de Software-Produktes entsprechend der Leistungsbeschreibung erfolgen soll. Alle Klassen und Methoden, die Ablauforganisation und die Hard- und Softwareumgebung sind zu entwerfen.

Der Entwurf kann in einen zielsystemneutralen und zielsystemspezifischen unterteilt werden. Die drei Ebenen System, Modul und Klasse sind zu berücksichtigen. Besonderes Augenmerk ist dabei auf modulares Design und Wiederverwendbarkeit der Klassen zu legen.

Der Testplan ist um die DV-spezifischen Testfälle (z.B. Division durch Null) zu erweitern, wobei die in der Phase *Voruntersuchung* definierten und in der Phase *Analyse* weiter verfeinerten Anwendungsfälle zugrunde gelegt werden.

Der Hauptunterschied liegt hier in der Art und Weise der Darstellung der Systemarchitektur. Das Entwurfsdokument des klassischen Vorgehensmodells enthält dazu die erforderlichen Programme, Module sowie die Beschreibung des logischen und physischen Datenbankdesigns, das des objektorientierten die zu Komponenten zusammengefaßten Klassen samt Methoden und die Objekt-Interaktionsdiagramme je

Komponente. Eine Beschreibung des logischen und physischen Datenbankdesigns kann ganz fehlen bzw. nur sehr rudimentär vorhanden sein, wenn auf eine Speicherung der Klassenattribute in einer Datenbank aus problemspezifischen Gründen ganz oder teilweise verzichtet werden kann.

Phase "Fertigung"

In der Fertigungsphase werden die Programme und Datenbanken erstellt oder generiert, die Anwendungskomponenten getestet und die erforderliche Dokumentation erstellt.

Die Unterschiede beschränken sich auf die Verwendung unterschiedlicher Programmier- und Dokumentationsrichtlinien. Die Testprotokolle enthalten insbesonders auch Testfälle, die die Funktionstüchtigkeit vererbter bzw. geerbter Eigenschaften von Klassen berücksichtigt. Die Dokumentation des produzierten Softwareproduktes enthält eine detaillierte Beschreibung aller Klassen samt Attribute und Methoden, um eine Wiederverwendung von Klassen zu unterstützen.

Phase "Inbetriebnahme"

In der Phase Inbetriebnahme wird das Software-Produkt samt den vereinbarten Dokumenten (z.B. Benutzerhandbuch, Betriebshandbuch) an den Auftraggeber übergeben und in Betrieb genommen. Die Einschulung der Benutzer wird anschließend durchgeführt.

Für den Benutzer ist nicht ersichtlich, daß das in Betrieb genommene Software-Produkt objektorientiert entwickelt wurde.

Phase "Gewährleistung"

Während der Gewährleistungsphase (ca. 3 - 6 Monate) erfolgt die kostenlose Korrektur von aufgetretenen Mängeln gemäß der vereinbarten Gewährleistungsbedingungen. Es werden aber keine Zusatzfunktionen realisiert.

Auch in dieser Phase ist äußerlich kein Unterschied zu klassisch entwickelten Software-Produkten festzustellen. Etwaige Fehlerbehebungen erfolgen selbstverständlich in der benutzten objektorientierten Sprache und mit objektorientierten Mitteln.

Erfahrungen bei der Modellanwendung

Das vorliegende Vorgehensmodell für die objektorientierte Software-Entwicklung ist mit dem Erfahrungszuwachs des Projektteams in dieser Technologie "mitgewachsen". Nachfolgend werden die wesentlichsten Erkenntnisse aus diesem Lernprozeß zusammengefaßt.

- Es hat sich als sehr motivierend herausgestellt, nicht irgendein (z.B. aus der Fachliteratur) bekanntes Vorgehensmodell herzunehmen und dem Projektteam zur Anwendung in die Hand zu geben, sondern das von einigen Projektmitarbeitern intuitiv verwendete Vorgehensmodell als Ausgangspunkt zu nehmen. Die Bereitschaft dieses Modell anzuwenden und vor allem auch in Richtung objektorientierter Technologie zu verbessern, war nach anfänglichen Startschwierigkeiten während der gesamten Projektlaufzeit gegeben.
- Die Unterstützung durch in objektorientierter Technologie gut ausgebildete Personen hat sich besonders bewährt, weil diese während der gesamten Projektlaufzeit direkt am Entwicklungsgeschehen direkt beteiligt waren und den Software-Qualitätssicherungsgedanken mitgetragen haben. Die Teammitglieder konnten dadurch unmittelbarer die Umsetzung des Vorgehensmodells "erfahren" und mitgestalten.
- Die in regelmäßigen Abständen durchgeführten Besprechungen mit dem Schwerpunkt der Software-Entwicklungsprozeßverbesserung haben ebenfalls einen wesentlichen Beitrag dazu geleistet, das Modell ständig zu verbessern und seine Anwendung praxisgerecht zu gestalten.
- Die Diskussionen in diesen Besprechungen haben immer wieder gezeigt, daß ein Vorgehensmodell, so "gut" es immer sein mag, nur einen Rahmen für die durchzuführenden Arbeiten bzw. geforderten Ergebnisse angeben kann. Bei seiner Anwendung ist es unbedingt notwendig mit Augenmaß vorzugehen. Es z.B. nicht sinnvoll, für jede kleine Berechnungfunktion ein Datenflußdiagramm zu zeichnen. Ebenso sollte man ein dynamisches Modell nur für diese System-Komponenten erstellen, die einen hohen Komplexitätsgrad aufweisen. Für solche mit mittlerem Komplexitätsgrad sind Objektinteraktionsdiagramme völlig ausreichend. Ein Speicherobjektmodell ist natürlich nur dann erforderlich, wenn eine relationale Datenbank zur Speicherung wichtiger Objektattribute benutzt wird.

Insgesamt betrachtet ist das Verständnis für und das Wissen über das wirksame Zusammenspiel der Geschäftsprozesse in der Softwareentwicklung deutlich gestiegen. Die Bereitschaft der Teammitglieder, einen geringen organisatorischen Mehraufwand auf sich zu nehmen, um auf einer qualitativ fundierten Basis Software zu entwickeln und Fehler möglichst zu vermeiden, hat deutlich zugenommen.

Sämtliche Verfahrenshandbücher stehen wie oben ausgeführt in Form von Online-Hilfetexten auf dem Netzwerkserver zur Verfügung. Eine ISO-konforme Lenkung der Dokumente ist auf diese Art und Weise einfacher zu realisieren. Bereits während des Projektverlaufs wurde durch die klare Struktur der Dokumente die Kommunikation innerhalb des Projektes vereinfacht, da jedes Projektmitglied jederzeit Zugriff auf die ent-

sprechenden einheitlich gestalteten Projekt-Dokumente auf dem Server hatte, was eine gezielte Suche bei offenen Fragen erleichterte. Die anfängliche Angst vor immensem Dokumentationsaufwand konnte durch die Verwendung der Standardvorlagen für die Dokumentation sehr schnell beseitigt werden. Dadurch konnte viel effizienter und vor allem organisierter dokumentiert werden. Lücken wurden schnell entdeckt und jeder Teil der Dokumentation fügte sich als wesentlicher Bestandteil in das Gesamtdokument ein, was die Motivation für diese zuvor gerne vernachlässigte Arbeit bedeutend erhöhte.

Die Verbesserungen des SE-Prozesses haben die Projektmitarbeiter überzeugt. Projektmitarbeiter, die in der Zwischenzeit an anderen Projekten arbeiten, sind nach eigenen Aussagen bestrebt, die verbesserte Arbeitsweise auch in diesen Projekten fortzusetzen, da sie von ihrer Sinnhaftigkeit überzeugt sind. Die erarbeiteten Standards werden in einigen Nachfolgeprojekten mit objektorientierter Analyse, Design und Implementierung verwendet und weiter verfeinert. Dadurch konnte schon in einer frühen Phase dieser Projekte der Aufwand für Qualitätssicherung deutlich reduziert werden.

Literatur

***[AMI]* ESPRIT project 5494**

AMI *ESPRIT project 5494: ami application of metrics in industry, Metric Users' Handbook*. Eigenverlag ami User Group, o.O. o.J.

***[Booch94]* Object-oriented analysis and design : with applications**

Booch, Grady: *Object-oriented analysis and design : with applications*. - 2. ed. - Redwood City, Calif. u.a. : Benjamin, Cummings, 1994

***[Basili/Rombach88]* The TAME project. Towards improvement-orientated software environments**

Basili, V.R.; Rombach, D.: *The TAME project. Towards improvement-orientated software environments.* In: IEEE Trans Soft Eng, 14 (1988) 6, S. 758 - 773

***[Jacobson94]* Object-Oriented Software Engineering: A Use Case Driven Approach**

Jacobson, Ivar; Christerson, Magnus; et al.: *Object-Oriented Software Engineering : A Use Case Driven Approach*. - revised 5. print. - Addison-Wesley, Wokingham/ Reading/ u.a. 1994

***[Mittelmann96]* The ami Approach Applied to an Object-Oriented Energy Management System**

Mittelmann, Angelika; Blümelhuber, Franz: *The ami Approach Applied to an Object-Oriented Energy Management System*. In Hoffmann, Detlef (Hrsg.): Final Symposium Eagle 1996 Proceedings. SISZ, Dortmund 1996, S 147-157

***[Rumbaugh93]* Objektorientiertes Modellieren und Entwerfen**

Rumbaugh, James: *Objektorientiertes Modellieren und Entwerfen*. Deutsche Ausgabe: Doris Märtin. - München ; Wien u.a. : Hanser u.a , 1993

***[SEI93]* Capability Maturity Model for Software**

Paulk, Mark C.; et al.: *Capability Maturity Model for Software (Version 1.1)*, Technical Report, CMU/SEI-93-TR-24. Software Engineering Institute Carnegie Mellon University, Pittsburgh 1993