Evolutionäre objektorientierte Systementwicklung und Projektmanagement

Wolfgang Hesse, Universität Marburg

Zusammenfassung:

Dieses Tutorium besteht aus drei Teilen: (a) wird über eine empirische Untersuchung zur Software-Entwicklung berichtet, (b) wird auf der Grundlage der Untersuchungen ein neues Entwicklungs- und Vorgehensmodell vorgestellt, (c) werden Konsequenzen für ein dynamisches, situationsangepaßtes Projektmanagement gezogen.

In einem kürzlich abgeschlossenen Forschungsprojekt zur wurden ca. 300 Software-Entwickler, -Manager und -Anwender über ihre Arbeitssituation befragt, um daraus Leitlinien für die künftige Arbeitsgestaltung und -organisation abzuleiten. Die Untersuchungsergebnisse zeigten unter anderem eine deutliche Diskrepanz zwischen den offiziell verkündeten und den tatsächlichen praktizierten Vorgehensweisen. Um diese besser miteinander in Einklang zu bringen und dabei neuere methodische Erkenntnisse zu berücksichtigen, wird ein Modell zur evolutionären, objektorientierten Software-Entwicklung (EOS) vorgeschlagen. Dieses Modell erlaubt eine systematische, an den Bausteinen der Software-Entwicklung orientierte Projektplanung und -verfolgung. Es stellt aber auch neue Herausforderungen an ein *dynamisches*, *situationsangepaßtes Projektmanagement*. Das EOS-Verfahren hat in der Software-Praxis bereits gute Resonanz gefunden und wird zur Zeit bei kleineren bis mittleren Software-Herstellern erprobt.

1 Einleitung

In einer 1993 abgeschlossenen interdisziplinären Untersuchung wurden ca. 300 Mitarbeiter in Software-Entwicklungsprojekten von Informatikern, Arbeitspsychologen und Soziologen über fachliche, psychologische und organisatorische Aspekte ihrer Arbeit befragt. Im Vordergrund stand dabei der Leitgedanke, auf empirischem Wege gefundene Erkenntnisse über die Arbeitssituation der Software-Entwickler in Leitlinien für die künftige Arbeitsgestaltung und -organisation umzusetzen.

In diesem Beitrag steht der Vorgehensaspekt im Mittelpunkt, aber auch eng damit verknüpfte Fragen, wie Entwicklungsmethodik, Projektmanagement und Qualitätsaspekte werden angesprochen. Die dazu dargestellen Ergebnisse stammen vorrangig aus den beiden von Informatikern und Soziologen durchgeführten Teilprojekten. Sie zeigten unter anderem eine deutliche Diskrepanz zwischen den offiziell verkündeten und den tatsächlichen praktizierten Vorgehensweisen bei der Software-Entwicklung.

Vorgehensmodelle für die Software-Entwicklung gehören zu den Standardthemen der Softwaretechnik und sind schon mehrfach klassifiziert und einander gegenübergestellt worden, z.B. von B. Boehm [Boe 88] und von Hesse et al. [HMF 92]. Danach lassen sich grundsätzlich *lineare* und *nicht-lineare (zyklische)* Vorgehensmodelle unterscheiden. Unter den linearen Modellen sind hervorzuheben: "Code and fix", Phasen- (oder "Wasserfall-") Modelle, Transformationsmodelle. Zu den nichtlinearen bzw. zyklischen Modellen zählen wir: Modelle zum *prototyping*, Spiralmodelle, inkrementelle und evolutionäre Modelle.

"Code and fix" steht für das ungeplante, oft chaotische, ohne verbindliche Richtlinien auskommende Vorgehen in den Frühzeiten der Software-Entwicklung. *Phasen- oder "Wasserfall"-Modelle* beschreiben die Software-Entwicklung als einen sequentiell von Phase zu Phase voranschreitenden Vorgang, wobei jede Phase auf den in der Vorphase erarbeitetetn Vorgaben aufsetzt und Ergebnisse zu erbringen hat, die gegenüber diesen Vorgaben auf Konsistenz und Vollständigkeit geprüft werden können.

Der *Transformations-Ansatz* basiert auf einer ähnlichen Grundidee. Hier werden jedoch als Vorgaben formale Spezifikationen gefordert, daher eignet sich der Ansatz vorwiegend für kleinere Programmsysteme oder -teile. Aus den Spezifikationen lassen sich mit Hilfe von als korrekt nachgewiesenen Transformationsregeln automatisch korrekte Programme ableiten, wobei man in der Regel schrittweise interaktiv vorgeht. .

Prototyping wurde als Gegenkonzept zum Phasenmodell propagiert: Starre Vorgaben und festgeschriebene Anforderungen erweisen sich oft als unrealistisch und werden im Laufe der Entwicklung überholt - mit der Gefahr von teuer erkauften Fehlentwicklungen oder Revisionen. Abhilfe sollen hier schnell gefertigte Prototypen schaffen, anhand derer man z.B. Anforderungen mit den Anwendern diskutieren, überprüfen und damit eine stabile Basis für die Entwicklung des endgültigen Systems legen kann [BKM 84]. B. Boehm hat in seinem *Spiralmodell* exemplarisch einen 3-fachen Prototyp-Entwicklungszyklus vorgeführt [Boe 88].

Als *inkrementell* bezeichnet man eine Systementwicklung, die von einem überschaubaren Teil- oder Kernsystem ausgeht und dieses schrittweise durch sog. *Inkremente* erweitert, bis die gesamte geforderte Funktionalität erreicht ist. Der Begriff der *evolutionären* Software-Entwicklung wurde von M.M. Lehman geprägt. Dieser unterscheidet mehrere Typen von Systementwicklungen und dabei als anspruchsvollsten Typ sogenannte *sozial eingebettete Systeme* (embedded systems). Diese sind eng mit der verwendenden Institution und den dort arbeitetenden Menschen (Nutzern) verflochten und haben Rückkopplungseffekte auf deren Organisation und Arbeitsweise. Sie müssen sich diesen Veränderungen dynamisch anpassen, was nach M.M. Lehman zu einer *Evolution* von fortschreitend an die Bedürfnisse der Anwender angepaßten Systemen führt [Leh 80].

Die IPAS-Untersuchungen haben ein deutliches Überwiegen der Phasen-/Waserfall-Modelle erbracht - zumindest was die offiziell verkündeten Richtlinien betrifft. Die tatsächlich ausgeübte Praxis zeigt allerdings erhebliche Abweichungen davon. Im ersten Teil dieses Beitrags wird näher darüber berichtet. Die offenkundige Diskrepanz zwischen verkündeten und praktizierten Vorgehensweisen zu überwinden und dabei moderne methodische Erkenntnisse zu berücksichtigen, wie sie sich z.B. aus den Prinzipien der Objektorientierung ergeben, waren die wesentlichen Ziele bei der Entwicklung des EOS-Modells. Dieses wird im zweiten Teil dieses Beitrags vorgestellt. Im dritten Teil stehen Fragen des Projektmanagements im Mittelpunkt: Welches sind die heute hervorstchenden Probleme bei der Projektabwicklung, welche Anforderungen stellen sich für das Management künftiger Projekte und welche Konsequenzen wird ein Vorgehen nach EOS für das Management nach sich ziehen?

2 Das IPAS-Projekt: Bericht über eine empirische Untersuchung zur Software-Entwicklung

In diesem Abschnitt wird über zwei ausgedehnte empirische Untersuchungen berichtet, die im Rahmen von IPAS durchgeführt wurden. Für die erste Untersuchung war ein gemischtes Team von Arbeitspsychologen und Informatikern zuständig. Dabei wurden insgesamt 29 Projekte in 19 Firmen untersucht und ca. 200 Entwickler, Projektleiter und Anwender mit Hilfe von Fragebögen und Interviews zu ihrer Arbeitssituation befragt (vgl. [F-H 93] und [H-F 94]). Einige wichtige Ergebnisse dieser Untersuchung seinen hier kurz zusammengefaßt:

- Kommunikation und Kooperation spielen in der Arbeit der Software-Entwickler eine größere Rolle als gemeinhin angenommen wird. Die sogenannten "frühen" (analysierenden) Tätigkeiten der Software-Entwicklung, in denen es wesentlich auf Kooperation mit Anwendervertretern sowie auf Kommunikation mit Fachspezialisten und Kollegen ankommt, werden immer wichtiger gegenüber den "klassischen" (synthetisierenden) Tätigkeiten des technischen Entwurfs, der Programmierung und des Tests. Beim Einsatz objekt-orientierter Techniken wird dieser Zwang zur Kooperation und Kommunikation (vor allem innerhalb der Teams) noch verstärkt.

- Die Reaktionen auf Fragen zur Benutzerbeteiligung sind zwiespältig. Einerseits wird anerkannt, wie wichtig sie für die System- und Problemanalyse ist, andererseits kann sie die Arbeit der Entwickler empfindlich "stören": durch zusätzliche Diskussionen, schwer zu befriedigende Änderungs- und Erweiterungswünsche, kurz durch zusätzliche Unsicherheit im Projekt.

- Das lineare Vorgehen bei der Software-Entwicklung ist meist eine Fiktion. Von den 29 untersuchten Projekten wurde über insgesamt 34 Rückkehrschleifen in frühere Projektphasen berichtet, davon führten 11 über zwei oder mehr Phasen hinweg.

- Bei der Betrachtung von Methoden und Werkzeugen steht das durchgängige methodische Konzept im Vordergrund. Werkzeuge werden umso schlechter akzeptiert, je "komplizierter" sie sind, je begrenzter ihr Anwendungsbereich ist, je weniger sie ins Gesamtkonzept passen, komplizierte Bedienungsregeln haben und ständige (Wieder-) Einarbeitung erfordern.

- Die Vielfalt neuer Entwicklungskonzepte, Methoden, Werkzeuge und technischer Plattformen sowie die sich verkürzenden Innovationszyklen stellen Entwickler und vor allem Projektmanager vor ständig wachsende Herausforderungen. Daraus resultiert ein starker Wunsch nach technischer Standardisierung und methodischer Harmonisierung.

Parallel dazu lief eine soziologische Untersuchung, die im Hinblick auf Vorgehensweisen und Projektmanagement u.a. die folgenden Ergebnisse erbrachte (vgl. [W-O 92] und [H-W 94]):

- In über 60 % der untersuchten Projekte war ein lineares Vorgehensmodell mit fest definierten Abschnitten oder Phasen vorgegeben. Eine inkrementelle Vorgehensweise bzw. Prototyping wurden in jeweils etwa einem Drittel der Projekte genannt (Mehrfachnennungen waren möglich).

- Der tatsächliche und der geplante Projektablauf wichen nicht selten erheblich voneinander ab. Nur in einem Viertel der Projekte wurde das Phasenschema, nach dem man vorging, auch eingehalten. In 41 % kam es zu Phasen-Überlappungen, in 13 % der Projekte wurde der Projektablauf als Echternacher Springprozession charakterisiert (zwei Schritte vorwärts, einen zurück), in weiteren 13 % als anarchisch.

- Formale Planung und Projekterfolg sind kaum miteinander korreliert: Als wichtig für den Projekterfolg erwies sich vielmehr die Fähigkeit, sich schnell an neue oder veränderte Erfordernisse anzupassen. Am erfolgreichsten waren solche Projekte, in denen der Entwicklungsprozeß als ständiger Lernprozeß verstanden wurde, und die die Entwicklungsziele dynamisch den aktuellen Erfordernissen anpassen konnten.

- Erfolg oder Mißerfolg von Projekten hängen mindestens ebenso stark von informellen Prozessen der Selbststeuerung (etwa bei der Einarbeitung in ein neues Projekt oder bei Kontakten zum Anwenderbereich) und von "stillen" Leistungen der Softwareentwickler ab als vom Einsatz formalisierter Verfahren und Management-Instrumente.

- Zwischen offizieller Planung und rigiden Regelungen einerseits und flexiblen Reaktionen und Selbststeuerungsprozessen der Entwickler andererseits herrscht eine deutliche Diskrepanz, die teilweise zu verdeckten Konflikten, Reibungs- und Effizienzverlusten führten. So litten z.B. Kostenkontrolle, Nachevaluierung und der projektübergreifende Wissenstransfer unter dieser Diskrepanz.

- Insgesamt überwog bei vielen untersuchten Projekten der Eindruck, daß die Schwierigkeiten (und damit die Lösungsmöglichkeiten) weniger auf software-technischer Ebene lagen als im Bereich des Projektmanagements. Das heißt der Schlüssel zum Erfolg liegt oft weniger in neuen Sprachen, Methoden, "Software-Architekturen" oder CASE-Tools als in einem richtigen Verständnis des Projektmanagements. Kurz: Die sogennate "Software-Krise" stellt sich (zumindest zu einem erheblichen Teil) als eine *Krise des Projektmanagements* heraus.

3 EOS: Ein Modell zur evolutionären, objektorientierten Software-Entwicklung

Als Antwort auf die dargestellten Befunde wird ein Modell für *evolutionäre objektorientierte Software-Entwicklung* (EOS) vorgeschlagen. Dieses orientiert sich an sechs Leitlinien, die im folgenden kurz erläutert werden:

- *Objektorientierung als durchgängige Entwicklungsmethodik*

  Die Software-Entwicklung folgt einem einheitlichen, durchgängigen Konzept: Dies beginnt bei der Auffassung und Modellierung eines Gegenstandsbereichs als Geflecht eigenständiger, miteinander agierender Objekte (objektorientierte Analyse, OOA) und führt über deren Umsetzung in weitgehend autarke Software-Bausteine (objektorientierter Entwurf, OOD) zu Systemen miteinander interagierender Programmeinheiten (objektorientierte Programmierung, OOP). Damit werden Strukturbrüche zwischen einzelnen Phasen vermieden und die Software-Bausteine lassen sich besser weiterentwickeln und wiederverwenden.

- Hierarchischer Systemaufbau

  Es wird eine (statische) Aufbaustruktur und eine (dynamische) Ablauf- und Interaktionsstruktur unterschieden. Statisch setzt sich ein System aus *Klassen* zusammen, die Mengen gleichartiger Objekte repräsentieren. Mehrere Klassen, die eine logische Einheit bilden oder ansonsten in enger Verbindung miteinander stehen, werden zu *Komponenten* zusammengefaßt. Komponenten (und die durch sie zusammengefaßten Klassen) sind zueinander disjunkt und bilden eine hierarchische (Baum-) Struktur.

  Dadurch ergibt sich eine dreistufige Hierarchie von Software-Bausteinen, bestehend aus (S.) System, (X.) Komponenten (oder Klassenkomplexen) (K.) Klassen. Komponenten können beliebig tief gestaffelt sein. Klassen (und damit Objekte) stehen untereinander in vielfältigen (semantischen) Beziehungen, z.B. der (baum- oder netzartigen) Vererbungsstruktur und der netzartigen (i.a. nicht-hierarchischen) "benutzt"-Struktur.

- Zyklische Entwicklung

  Jeder Software-Baustein (gleich welcher Entwurfsebene er angehört) durchläuft (ein- oder mehrfach) einen Entwicklungs- und Lebenszyklus, der durch die folgenden vier wesentlichen *Tätigkeiten* charakterisiert ist: (.A) Analyse, (.E) Entwurf, (.I) Implementierung, (.O) Operationeller Einsatz -Erprobung, Nutzung und Revision (vgl. Abb.1).

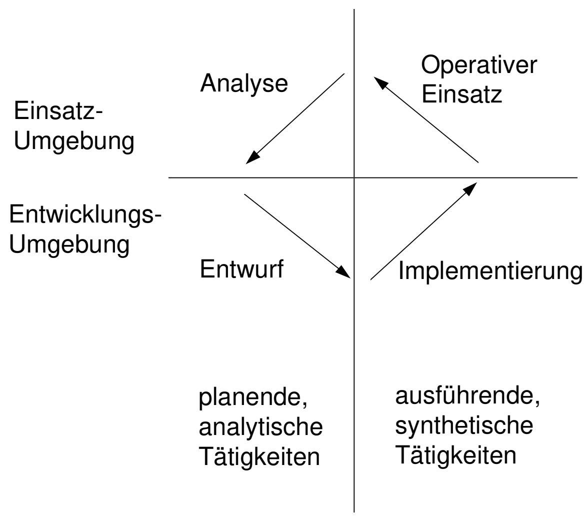

Abb. 1: Tätigkeiten in einem Entwicklungszyklus

- Einbezug von Erprobung, Nutzung und Revision in die Entwicklungszyklen

Ein wichtiger Unterschied zu den herkömmlichen Vorgehensmodellen besteht darin, daß Software-Entwicklung nicht länger als bloßes Konstruktions-Projekt mit einer nicht näher definierten "Wartungsphase" am Ende betrachtet wird, sondern als ein kontinuierlicher zyklischer Prozeß, in dem auf analysierende und konstruierende Entwicklungsschritte solche der Erprobung, Nutzung und (möglichen) Revision folgen (vgl. oben). Das gilt nicht nur (wie bei den Prototyping-Modellen) für das Gesamtsystem sondern *für Bausteine aller Entwicklungsebenen*. (vgl. Abb. 2)

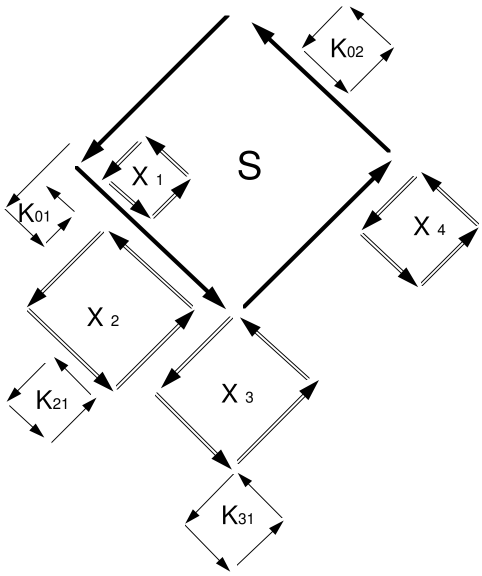

Abb. 2: Tätigkeitsyklen für Software-Bausteine

- Weiter- und Wiederverwendung von Software-Bausteinen

Wichtigstes Ziel aller OO-Vorgehensweisen ist die Förderung der *Wiederverwendbarkeit* von Software-Bausteinen. Dazu ist eine *Bausteinbibliothek* (vgl. oben) aufzubauen, die die bisher definierten und erstellten Bausteine enthält und die für die Ableitung neuer Bausteine (etwa durch Spezialisierung, Modifikation, Zusammenfassung) herangezogen wird. Die Arbeit mit der Bausteinbibliothek ist besonders wichtig in den beginnenden und abschließenden Tätigkeiten eines Zyklus: um einerseits (während der Analyse) möglicherweise wiederverwendbare Bausteine aufzuspüren und andererseits (nach den ersten erfolgreichen Nutzungsläufen) geeignete Klassen in die Bibliothek einzubringen.

- Mit der Entwicklungsmethodik abgestimmte Managementverfahren

Projektmanagement-Verfahren können nur dann effektiv sein, wenn sie in enger Abstimmung mit den technischen Entwicklungsverfahren entwickelt and angewendet werden. Deshalb gehört zum EOS-Vorgehensmodell ein darauf zugeschnittenes Management-Modell. An die Stelle der herkömmlichen Phasen treten die entsprechenden EOS-Strukturen: Zyklen und Tätigkeitsschritte. Daraus ergeben sich weitere Anforderungen und Konsequenzen für das Management.

4 Konsequenzen für das Projektmanagement

Warum sind Phasenmodelle trotz der geschilderten Unzulänglichkeiten immer noch so verbreitet und werden vor allem vom Management nach wie vor bevorzugt? Wichtige Gründe dafür sind:

- Der Entwicklungsprozeß wird auf einfache und überschaubare Weise strukturiert.
- Das Projekt als Ganzes und die einzelnen Arbeitsschritte lassen sich leicht planen, komplexe Prozesse lassen sich phasenweise arbeitsteilig organisieren.
- Das Phasenschema liefert ein Raster zur Überprüfung der (Zwischen-) Ergebnisse.

Die Attraktivität des Phasenmodells beruht nicht zuletzt in seiner Einfachheit und Eindeutigkeit. Das Phasenkonzept stellt an den Projektbeginn, also zu einem Zeitpunkt, wo Durchsetzungsaspekte besonders wichtig sind, feste Aussagen über Zeiten, Kosten und Ablauf des Projekts. Die IPAS-Untersuchung hat diese Tendenz bestätigt: Die Mehrheit der untersuchten Projekte wurde - zumindest offizell - nach einem linearen Phasenmodell durchgeführt und dafür waren in erster Linie die genannten Gründe ausschlaggebend.

Der Wunsch nach leicht berechenbaren Planungsdaten (und der Zwang, solche bereits zu Projektbeginn zu präsentieren) sind offenbar so groß, daß man dafür deren Ungenauigkeit und spätere oft unangenehme Nachkalkulationen und Nachverhandlungen in Kauf nimmt.

Projekte *evolutionär* abzuwickeln (d.h. nicht mit einer vollständigen Planung zu beginnen, sondern diese schrittweise den Bedürfnissen anzupassen) gestaltet sich demgegenüber wesentlich komplizierter: Ziele und Vorgaben müssen immer wieder neu überprüft und definiert werden. Dies stellt neue Anforderungen an die Projektmanager und deren Fähigkeiten, die Entscheidungs- und Konsensbildung im Projekt zu fördern. Dies erfordert u.a. eine laufende Neubestimmung von Vorgaben und eine kontinuierliche Abstimmung zwischen Entwicklungs- und Anwendungsbereich.

Daraus lassen sich Forderungen für das Management evolutionärer Projekte ableiten. An die Stelle "planlastigen Handelns" im Phasenmodell muß ein neues Verhältnis von miteinander verschränktem Planen und Handeln treten:

- Planerische Vorgaben müssen elastischer werden.
- Die strenge Abfolge (und Trennung) von Planen und Ausführen muß überwunden werden.
- Zielsetzungen müssen im Laufe der evolutionären Entwicklung stets neu überdacht und angepaßt werden.

Wie kann lassen sich diese Forderungen für das Management von Projekten nach dem EOS-Modell umsetzen?

- Die drei EOS-Hierarchieebenen bilden den Ansatzpunkt für Planungsschritte unterschiedlicher Detaillierung: Die projektweite Grobplanung des S-Zyklus wird durch feinere Planungsschritte auf Komponentenebene (X-Zyklen) und für die Tätigkeiten auf Klassenebene (K-Zyklen) ergänzt. Für die Detailplanung stehen dem Management die einzelnen Zyklusschritte als Planungseinheiten zur Verfügung. Mit seinen unterschiedlichen Differenzierungsstufen liefert das EOS-Modell die geeignete Handhabe, sich in der jeweiligen Projektsituation flexibel an die unterschiedlichen Bedürfnisse von Auftraggebern, Anwendern und Entwicklern anzupassen.
- Im EOS-Modell ist die klassische Trennung von "fachlichen" und "DV-technischen" Phasen aufgehoben. An deren Stelle treten die Entwicklungszyklen, die beide Aspekte umfassen und auf allen Ebenen für ein Wechselspiel von Entwicklung und Anwendung

und für einen kontinuierlichen Abgleich zwischen anwendungsbezogenen Anforderungen und technischen Gestaltungsmöglichkeiten sorgen.

- Das EOS-Modell sieht Baustein-orientierte Arbeitsteilung und Verantwortlichkeiten vor. Das heißt, einzelne Arbeitsschritte sind jeweils an den zu bearbeitenden Software-Baustein gekoppelt. Mehrere, am gleichen Bausteine durchzuführende Tätigkeiten (wie Klasse XYZ analysieren, entwerfen, implementieren) sind *einer* Person oder einem Team zugeordnet. Im Falle anwendungsnaher Komponenten oder des Gesamtsystems können dies auch (aus Entwicklern und Anwendern) gemischte *Betreuungs-Teams* sein.

  Daraus ergeben sich einerseits klare Verantwortlichkeiten, die in der Regel auch den operativen Einsatz und damit die gesamte Betreuung des betreffenden Bausteins umfassen. Andererseits bleiben die notwendigen Freiräume für die Verantwortlichen bei der Bearbeitung "ihrer" Bausteine. Selbsteuerungsprozesse werden nicht durch Planung unterbunden, sondern gefördert.

- Das Projektmanagement benötigt ein Instrumentarium zur Projektstatus-Bestimmung und zur Überprüfung von (Zwischen-) Ergebnissen. Im EOS-Modell treten an die Stelle der Meilensteine an den Phasenenden differenziertere *Revisionspunkte*. Ein möglicher Revisionspunkt kann etwa folgendermaßen definiert sein: "Komponenten A, B, C abgeschlossen, Komponente D und Klassen E, F implementiert, Komponenten G und H entworfen, Komponente J analysiert".

  Revisionspunkte werden nicht vollständig zu Projektbeginn vorgeplant, sondern ihre (anfängliche Grob-) Planung wird im Laufe des Projektfortschritts verfeinert und den laufenden Gegenbenheiten und Erfordernissen im Projekt angepaßt. Dies erfordert entsprechend differenzierende Hilfsmittel zur Projektplanung und -verfolgung für die Projektmanager. Diese dienen hauptsächlich der Information über den Projektstand und sollten nicht als Planungskorsett mißverstanden werden.

- An die Stelle eines zu Projektbeginn verabschiedeten Projektplans treten Entscheidungsketten, in denen von Tätigkeit zu Tätigkeit, von Revisionspunkt zu Revisionspunkt und von Zyklus zu Zyklus Beschlüsse überprüft und vor dem Hintergrund der jeweils aktuellen Lage modifiziert werden. Auch bei solchermaßen dynamisierter Planung können die Projektmanager auf bewährte Instrumente zur Statusbestimmung, Plan-/Ist-Kontrolle und Kostenberechnung für seine laufenden Projekte zurückgreifen.

- Software-Entwicklung ist ein kontinuierlicher projektübergreifender Prozeß. Mit dem Einsatz einer einmal entwickelten Software ist deren Entwicklung nicht abgeschlossen. Im EOS-Modell bilden die Entwicklungszyklen die Voraussetzung für einen Wissenstransfer von Zyklus zu Zyklus - sei es innerhalb eines Projekts oder zum nächsten, benachbarten Projekt.

Evolutionäre Software-Entwicklung macht das Projektmanagement nicht leichter, sondern stellt vielmehr neue Herausforderungen an die Verantwortlichen. So lassen sich nicht mehr im voraus genaue Aussagen über den Umfang von größeren Entwicklungsvorhaben machen. Dies kann sich in der Akquisitionsphase sogar erschwerend auswirken. In Wahrheit handelt es sich aber um inhärente Ungenauigkeiten, die das Phasenmodell mit seiner scheinbaren Planungspräzision nur verdeckt. Am Ende erweist sich die evolutionäre Methode als die ehrlichere, denn im Unterschied zum Phasenkonzept, paßt sie die Planungsdaten laufend der Realität an.

In der betrieblichen Praxis wird der Erfolg evolutionärer Entwicklungsmethoden entscheidend davon abhängen, ob sich eine dazu passende, realitätsgerechtere Projektplanung durchsetzen läßt. Diese müßte u.a. den Anspruch aufgeben, Projekte minutiös vorplanen und starr nach Plan ablaufen lassen zu können.

5 Ausblick

Das EOS-Modell ist seit seiner ersten Veröffentlichung auf vielfache positive Resonanz gestoßen und wird gegenwärtig von kleineren und mittleren Software-Hersteller- und Beratungsunternehmen erprobt. In den nächsten Entwicklungsstufen sollen die Verfahren zur objektorientierten Modellbildung und das Instrumentarium zum Projektmanagement und zur Qualitätssicherung weiter ausgebaut werden.

Literaturhinweise:

[Boe 88] B.W. Boehm: A spiral model of software development and enhancement, Computer, May 1988, pp. 61-72

[BKM 84] R. Budde, K. Kuhlenkamp, L. Mathiassen, H. Züllighoven (eds.): Approaches to prototyping, Springer 1984

[F-B 92] M. Frese, F. Brodbeck: Psychologische Aspekte der Software-Entwicklung, IBM-Nachrichten 42, Heft 309, S. 15-19 (1992)

[F-H 93] M. Frese, W. Hesse: The work situation in software development - Results of an empirical study, ACM SIGSOFT Software Engineering Notes, Vol. 18, No. 3, pp. A-65 - A-72 (1993)

[HBS 92] W. Hesse, U. Bittner, J. Schnath: Results from the IPAS Project: Influences of methods and tools, quality requirements and project management on the work situation of software developers; in: P. Elzer, V. Haase (Eds.): Proc. Fourth IFAC/IFIP Workshop on Experience with the Management of Software Projects, Annual Review in Automatic Programming, Vol. 16, Part II, Pergamon Press 1992

[H-F 94] W. Hesse, M. Frese: Zur Arbeitssituation in der Software-Entwicklung - Resümee einer empirischen Untersuchung, in: Informatik Forschung und Entwicklung 9.4, pp. 179-191 (1994)

[HMF 92] W. Hesse, G. Merbeth, R. Frölich: Software-Entwicklung - Vorgehensmodelle, Projektführung und Produktverwaltung, Handbuch der Informatik, Band 5.3 Oldenbourg 1992

[H-W 94] W. Hesse, F. Weltz: Projektmanagement für evolutionäre Software-Entwicklung; in: Information Management 3/94, pp. 20-33, (1994)

[I-S 92] Thema: Projektmanagement für objektorientierte Software-Entwicklung, Themenheft, Informatik-Spektrum, Bd. 15, Heft 5 (1992)

[Leh 80] M.M. Lehman: Programs, life cycles, and laws of software evolution, Proceedings of the IEEE, Vol. 68, No. 9, pp. 1060-1076 (1980)

[Mey 88] B. Meyer: Object-oriented software construction, Prentice Hall 1988

[W-O 92] F. Weltz, R. Ortmann: Das Softwareprojekt - Projektmanagement in der Praxis, Campus-Verlag 1992