February 1999

Kooperative Softwareentwicklung: Konzepte, Modelle und Werkzeuge

Josef Altmann
Universität Linz, altmann@swe.uni-linz.ac.at

Gustav Pomberger
Universität Linz, pomberger@swe.uni-linz.ac.at

Kooperative Softwareentwicklung: Konzepte, Modell und Werkzeuge

Josef Altmann
Universität Linz (altmann@swe.uni-linz.ac.at)

Gustav Pomberger
Universität Linz (pomberger@swe.uni-linz.ac.at)

Inhalt

1 Einleitung

2 Zur Rolle der Kooperation in Softwareentwicklungsprozessen

2.1 Kooperationsaspekte bei verteilter Teamarbeit
2.2 Merkmale von Softwareentwicklungsprozessen
2.3 Merkmale von verteilten, kooperativen Softwareentwicklungsprozessen

3 Gestaltungsgrundsätze für eine Werkzeugunterstützung

4 Ein Modell für kooperative Softwareentwicklungsprozesse

4.1 Prozeßsicht
4.2 Produktsicht

5 Cooperation Assistant – Eine Arbeitsumgebung für kooperative Softwareentwicklung

6 Zusammenfassung und Ausblick

Electronic Business Engineering / 4. Internationale Tagung Wirtschaftsinformatik 1999.
Hrsg.: August-Wilhelm Scheer; Markus Nüttgens. – Heidelberg: Physica-Verlag, 1999

Abstract

Die Entwicklung komplexer Softwaresysteme bedingt eine intensive Zusammenarbeit mehrerer Projektmitarbeiter mit unterschiedlichen Aufgaben. Der Entwicklungsprozeß ist häufig ein zeitlich und räumlich verteilter Arbeitsprozeß, der innerhalb und zwischen spezialisierten Arbeitsgruppen stattfindet. Deshalb gilt es, auf Fragen zur Arbeitsteilung, zur Kommunikation, zur Koordination und zur Kooperation bei der Planung, Entwicklung und Wartung komplexer Softwaresysteme entsprechende Antworten zu finden.

Entwicklungsumgebungen, die die Gruppenarbeit explizit unterstützen, sind eine wesentliche Voraussetzung, um qualitativ hochwertige Softwaresysteme zu erstellen. Die meisten der heute eingesetzten Softwareentwicklungsumgebungen unterstützen primär technische Aspekte und weisen im Bereich der Organisationsunterstützung Lücken auf. Dieser Aufsatz beschreibt ein Modell für kooperative Arbeitsprozesse in Softwareprojekten und eine darauf abgestimmte Entwicklungsumgebung, die sowohl die organisatorischen als auch die technischen Aspekte der Softwareentwicklung in ausgewogener Weise unterstützt. Damit soll ein Beitrag zur Produktivitäts- und Qualitätssteigerung bei verteilter Softwareentwicklung geleistet werden.

Mit dem vorgestellten Modell für kooperative Softwareentwicklung und der darauf aufbauenden Entwicklungsumgebung *Cooperation Assistant* wird einerseits ein Beitrag zur Beseitigung von Leistungsdefiziten in Softwareentwicklungsumgebungen geleistet und andererseits eine Experimentierumgebung zur Verfügung gestellt, die es gestattet, auf empirischer Basis die These zu untermauern oder zu widerlegen, daß der kooperative, cluster-orientierte Entwicklungsansatz im Hinblick auf die Ausschöpfung von Produktivitäts- und Qualitätssteigerungspotentialen dem rigiden phasenorientierten Entwicklungsansatz deutlich überlegen ist, insbesondere bei räumlich und zeitlich verteilter Projektorganisation. Die Arbeiten zur Konzeption des Modells und die Implementierung der Entwicklungsumgebung sind abgeschlossen. Mit der Evaluierung des vorgeschlagenen Ansatzes wurde begonnen.

Die bisher vorliegenden Erfahrungen mit dem Einsatz des *Cooperation Assistant* bestätigen die Annahme, daß der kooperative, cluster-orientierte Entwicklungsansatz sowohl produktivitäts- als auch qualitätssteigernd wirkt. Der vorgestellte Ansatz zeichnet sich vor allem durch die leichte Verständlichkeit des Modells, die intuitive Bedienbarkeit der Werkzeuge und die übersichtliche Darstellung von prozeß- und produktbezogenen Informationen aus.

1 Einleitung

Die notwendigen Voraussetzungen für die wirtschaftliche und professionelle Erstellung qualitativ hochwertiger Softwareprodukte betreffen nicht allein die technischen Aspekte, sondern vor allem auch den Prozeß der Aufteilung der Entwicklungsaufgaben und -aktivitäten innerhalb eines Projektteams, den Aufbau von Kommunikations- und Koordinationsbeziehungen und die Unterstützung von kreativen Problemlösungsprozessen.

Durch die zeitliche und räumliche Aufteilung eines Softwareprojektes erfolgt die Zusammenarbeit zwischen den Projektmitarbeitern häufig asynchron. Eine regelmäßige Synchronisation und die Möglichkeit der Entwickler aktuelle Zustände und Ereignisse zu erfassen, sind für eine effiziente Zusammenarbeit unerläßlich. Die spontane Zusammenarbeit, die direkte und indirekte Kommunikation, der Aufbau eines Gruppenbewußtseins, das Wissen über Einzelheiten und Besonderheiten des Projektes und seines aktuellen Zustandes sowie die Wiederverwendung von Erfahrungswissen sind wichtige Aspekte bei der Unterstützung kooperativer Softwareentwicklung, die Einfluß auf die Wirtschaftlichkeit des Entwicklungsprozesses und die Qualität des herzustellenden Produktes haben.

Betrachtet man die aktuellen Forschungsansätze und heute eingesetzte Entwicklungsumgebungen, so sind gruppenunterstützende Aspekte unterrepräsentiert oder genügen oft nicht den Anforderungen an eine effiziente Unterstützung der Kommunikation und Koordination (Altmann 1998, S. 29 ff.). Es existieren zwar Werkzeuge zur Unterstützung der Gruppenarbeit in Softwareprojekten (Elektronische Postsysteme, Vorgangsbearbeitungssysteme, Gruppeneditoren, synchrone Debugging-Werkzeuge, etc.), diese sind aber meist nicht integraler Bestandteil einer Entwicklungsumgebung oder unterstützen nur ausgewählte Tätigkeiten.

Ablaufsteuernde Ansätze aus dem Bereich der prozeßgesteuerten Softwareentwicklung (siehe z.B. Garg/Jazayeri 1995) oder Ansätze des computerunterstützten Software Engineering (siehe z.B. Balzert 1998, S. 591 ff.) verfolgen das Ziel, komplexe Vorgänge der Softwareentwicklung formal zu beschreiben, um sie in weiterer Folge zu (teil-)automatisieren. Voraussetzung dafür ist eine präzise Beschreibung des Softwareentwicklungsprozesses. Der Entwicklungsprozeß entzieht sich jedoch in der Regel aufgrund der laufenden dynamischen Veränderung einer a priori Beschreibung, die die erforderliche Präzision aufweist. Daher führt der Einsatz von formalisierten Prozeßmodellen in der Praxis selten zum Erfolg (Pomberger/Heinrich 1998). Außerdem können in der Anwendung streng formalisierter Prozeßmodelle und darauf aufbauender automatisierter Steuerungs- und Kontrollinstrumente die Potentiale kreativer Prozesse, die für die Produktqualität oft von entscheidender Bedeutung sind, nicht genutzt werden.

In diesem Beitrag werden ausgehend von einer problemorientierten Betrachtung von Softwareprojekten, die zu einer Auflistung der wesentlichen Merkmale verteilter, kooperativer Softwareentwicklung führt, zunächst universelle Gestaltungsgrundsätze für eine Arbeitsumgebung zur Unterstützung kooperativer, verteilter Softwareprojekte abgeleitet. Die identifizierten Gestaltungsgrundsätze dienen als Grundlage für die Entwicklung eines Modells für kooperative Arbeitsprozesse, das

die Ausgangsbasis für die Analyse, Strukturierung, Verwaltung und Abstimmung von arbeitsteiligen Softwareentwicklungsaufgaben bildet. Daran anschließend wird ein Überblick über eine gruppenunterstützende Arbeitsumgebung für kooperative Softwareentwicklung gegeben, die die praktische Umsetzung des entwickelten Modells für kooperative Arbeitsprozesse in Softwareprojekten ermöglicht. Die abschließende Zusammenfassung enthält einen kurzen Erfahrungsbericht über den Einsatz des entwickelten Prozeßmodells und der unterstützenden Entwicklungsumgebung sowie einen Ausblick auf weitere Forschungsaktivitäten.

2 Zur Rolle der Kooperation in Softwareentwicklungsprozessen

Im folgenden werden zunächst ausgehend von einer allgemeinen Betrachtung der Kooperationsaspekte bei verteilter Teamarbeit die Begriffe Kommunikation, Koordination und Kooperation eingeführt. Danach werden die generellen Merkmale von Softwareentwicklungsprozessen und im Anschluß daran die speziellen Merkmale, die verteilte, kooperative Softwareentwicklungsprozesse kennzeichnen, im Hinblick auf die besonderen Kommunikations- und Kooperationsbedarfe diskutiert.

2.1 Kooperationsaspekte bei verteilter Teamarbeit

Kooperationsprozesse, wie zum Beispiel die Softwareentwicklung im Team, erfordern Koordinationsprozesse, um die kooperativen Tätigkeiten der einzelnen Mitarbeiter aufeinander abstimmen zu können. Koordinationsprozesse wiederum setzen Kommunikationsprozesse voraus, wie zum Beispiel den Austausch von Informationen über die verschiedenen Entwicklungsaktivitäten. In Anlehnung an Bauknecht et al. (Bauknecht et al. 1995, S. 12) werden die Begriffe und Wirkungszusammenhänge von Kommunikation, Koordination und Kooperation wie folgt benutzt:

> *Kommunikation* umfaßt den zwischen Kommunikationspartnern stattfindenden Prozeß der Übermittlung und des Austausches von Informationen.

Dient der Austausch von Informationen zur Abstimmung von Aktivitäten zwischen den Gruppenmitarbeitern, so bildet diese Art der Kommunikation die Grundlage für die Koordination arbeitsteiliger Prozesse.

> *Koordination* beruht auf geeigneten Kommunikationsprozessen und umfaßt alle Tätigkeiten, die zur Abstimmung von arbeitsteiligen Aufgaben im Rahmen eines Arbeitsprozesses notwendig sind.

Während die Beteiligten bei Koordinationsprozessen unterschiedliche Arbeitsziele verfolgen, arbeiten im Rahmen der Kooperation mehrere Personen geplant und koordiniert zusammen, um ein gemeinsames Ergebnis zu erreichen.

Kooperation bezeichnet jene Koordination, die zur Vereinbarung gemeinsamer Ziele und zur aufeinander abgestimmten Erreichung eines gemeinsamen Arbeitsergebnisses zwischen den beteiligten Personen notwendig ist.

Die Ausführungen zum Begriff der Kooperation zeigen, daß die Gestaltung, Abstimmung und Steuerung arbeitsteiliger Aufgaben auf Basis von Kommunikations- und Koordinationsvorgängen stattfindet. In Abhängigkeit vom Handlungsspielraum der beteiligten Gruppenmitglieder kann zwischen verschiedenen Kooperationsformen unterschieden werden. Friedrich (Friedrich 1994, S. 19) unterteilt aufbauend auf Popitz et al. (Popitz et al. 1957) Kooperation anhand der Eingriffs- und Gestaltungsmöglichkeiten der Beteiligten in folgende Formen:

- *Teamartige Kooperation* ist durch das Einwirken auf die auszuführenden Handlungen des jeweils anderen Kooperationspartners gekennzeichnet. Es stehen Freiräume für die Gestaltung interner Organisationsstrukturen und Prozeßabläufe zur Verfügung. Die kooperative Arbeit erfolgt auf Basis von abgestimmten Zielen und ausgehandelten Kooperationsregeln. Die Kooperationspartner sind gleichgestellt.
- *Gefügeartige Kooperation* wird durch formale Mechanismen erzwungen. Die Zusammenarbeit vollzieht sich vorwiegend auf Basis sachlicher und technischer Elemente des Leistungserstellungsprozesses. Normierte Arbeitsabläufe und formalisierte Prozesse führen zu einer Fremdkoordination und Einschränkung der Handlungsalternativen. Kooperationswege sind weitgehend vorgegeben, und es liegt eine feste räumliche und zeitliche Zuordnung von Arbeitsplätzen und Arbeitsaufgaben vor. Die Kooperationspartner nehmen verschiedene hierarchische Positionen ein und sind in der Regel nicht gleichgestellt.

Die Auswirkungen der gefügeartigen Kooperation auf den Softwareerstellungsprozeß werden von Pasch (Pasch 1994, S. 19 ff.) als Software-Bürokratie bezeichnet. Kennzeichnendes Merkmal dieser Bürokratie ist die formale Einbettung der einzelnen Entwickler in eine starre Organisationsstruktur. Handlungen werden von einer externen Stelle (z.B. Prozeßmodellierer, Vorgangsbearbeitungssystem) koordiniert und gesteuert. Der Prozeß der Interessensaushandlung und Konfliktverarbeitung wird größtenteils unterdrückt.

Teamartige Kooperation unterstützt unstrukturierte, nicht im vorhinein planbare Aufgabenabwicklungen und die Selbstorganisation im kooperativen Arbeitskontext. Der Konflikt bildet den Ausgangspunkt für einen kontinuierlichen Abstimmungsprozeß, der von den Beteiligten mitgestaltet und kontrolliert werden kann. Teamartige Kooperation unterstützt individuelles Handeln und Kreativität und bezieht den aktuellen Arbeitskontext ein. Im Gegensatz zu gefügeartiger Kooperation sind rechnergestützte Werkzeuge dieser Kategorie durch eine unterstützende Sichtweise geprägt. Diese Sichtweise ermöglicht es den Beteiligten, ein wechselseitiges Verständnis über die kooperative Arbeit aufzubauen. Die Automatisierung eines Arbeitsprozesses steht dabei nicht im Vordergrund.

Die obigen Ausführungen haben unterschiedliche Perspektiven kooperativer Arbeit aufgezeigt. Im folgenden werden zunächst die wesentlichen Merkmale von Softwareentwicklungsprozesse und danach die Merkmale verteilter, kooperativer Softwareentwicklungsprozesse diskutiert.

2.2 Merkmale von Softwareentwicklungsprozessen

Der Softwareentwicklungsprozeß wird wesentlich von dem zugrundegelegten Vorgehensmodell beeinflußt. Ein Vorgehensmodell beschreibt auf abstrakte Weise den allgemeinen Ablauf des Entwicklungsprozesses mit seinen Teilaufgaben, deren Abhängigkeiten und den zu erarbeitenden Ergebnissen sowie die eingesetzten Qualitätssicherungsmaßnahmen. Die übergeordneten Arbeitsschritte werden üblicherweise als Phasen bezeichnet. Im Rahmen des Software Engineerings wurden verschiedene Phasenmodelle entwickelt und gegenübergestellt. Eines der Hauptprobleme bei der Anwendung von Phasenmodellen ist die Abgrenzung der einzelnen Phasen gegeneinander und die Berücksichtigung der Wechselwirkungen zwischen den Phasen. In diesem Beitrag wird auf eine ausführliche Behandlung einzelner Phasenmodelle verzichtet (siehe z.B. Boehm 1988; Pomberger 1996; Sommerville 1996; Meyer 1997; Balzert 1998). Hingegen greifen wir das Cluster-Modell von Meyer (Meyer 1997, S. 923 ff.) für unsere Betrachtungen heraus, weil es einerseits auf die heute übliche Softwaretechnologie der Objektorientierung Bezug nimmt und andererseits durch die Aufteilung eines Projektes auf verschiedene Entwicklungsteams die oben angeführten Hauptprobleme der Anwendung von Phasenmodellen vermeiden helfen will.

Meyer trennt, wie bei den meisten Vorgehensmodellen, zwischen den unterschiedlichen Projektaktivitäten wie Analyse, Entwurf, Implementierung und Wartung. Als wesentlichen Unterschied zu traditionellen Phasenmodellen, wie dem klassischen sequentiellen Phasenmodell oder dem Wasserfallmodell, werden aufeinanderfolgende Aktivitäten nicht als eigenständige, isolierte Schritte betrachtet, sondern als aufeinanderfolgende Systemerweiterungen eingestuft. Abbildung 1 zeigt diesen Entwicklungsprozeß, der nach Meyer dem Wesen der objektorientierten Entwicklung entspricht.

Der Softwareentwicklungsprozeß ist nicht primär durch eine Phasenorientierung und die daraus resultierende Folge von Zwischenprodukten gekennzeichnet, sondern durch die Entwicklung von Komponenten (Teilprodukten). Das Entwicklungsteam ist dabei für alle Phasen verantwortlich und die strenge Phasentrennung wird aufgegeben, d.h. die Grenzen zwischen den Phasen sind oft nicht mehr ersichtlich. Diese Art der Organisation des Entwicklungsprozesses setzt eine intensive Koordination und Kommunikation innerhalb und zwischen den Entwicklungsteams der einzelnen »Projekt-Cluster« voraus, weil (zum Erhalt der Flexibilität und zur Nutzung kreativer Potentiale) weder die Phasenergebnisse innerhalb der Cluster noch die Cluster-Ergebnisse im vorhinein vollständig und formal spezifiziert werden.

Ein wesentlicher Unterschied zu konventionellen Phasenmodellen kommt bei dem von Meyer vorgeschlagenen Modell dadurch zustande, daß nach der Erstellung einer Grobarchitektur für das Gesamtsystem eine Aufteilung in Teilprojekte vorgenommen wird. Die dabei entstehenden Cluster repräsentieren Teilsysteme (Komponenten), die von kleinen Entwicklungsteams realisiert werden. Abbildung 1 zeigt, daß im Cluster-Modell eine Sequentialisierung des Entwicklungsablaufes angestrebt wird, allerdings nicht für das Gesamtsystem. Nach Abschluß der Aufteilung des Gesamtprojektes wird für jeden Cluster ein Mini-

Entwicklungsprozeß definiert. Die Teilprojekte laufen gleichzeitig oder zeitlich versetzt ab. Verschiedene Projektziele werden zu unterschiedlichen Terminen erreicht. Am Ende jeder Teilentwicklung schließt sich eine Verallgemeinerungsphase an, um wiederverwendbare Softwarekomponenten zu identifizieren und anschließend cluster- bzw. projektübergreifend zur Verfügung zu stellen. Daraus ergibt sich ein paralleler bzw. sich überlappender Entwicklungsprozeß, der eine intensive Kommunikation und Koordination der beteiligten Entwicklungsteams erfordert.

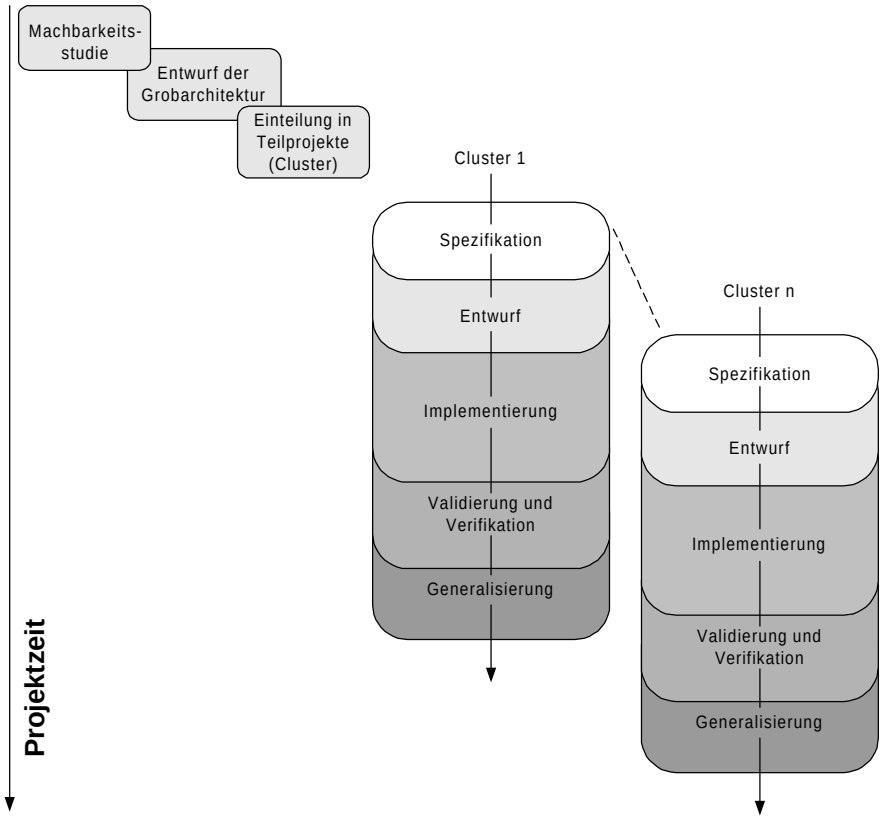

Abbildung 1: Cluster-Modell nach Meyer (Meyer 1997, S. 927)

Floyd/Züllighoven (Floyd/Züllighoven 1997, S. 646) unterscheiden bei der Softwareerstellung zwischen produkt- und prozeßbezogenen Entwicklungsaktivitäten.

- *Produktbezogene Aktivitäten* umfassen Anforderungsermittlung und Systementwicklung. In Vordergrund steht das Softwareprodukt mit den

jeweiligen Entwicklungsergebnissen, bestehend aus Prototypen, Programmen und Dokumenten. Der Herstellungsprozeß kann anhand von vordefinierten Phasen mit vordefinierten Zwischenergebnissen gegliedert werden.

- *Prozeßorientierte Aktivitäten* dienen zur Koordination und Kooperation des Entwicklungsprozesses und umfassen Produktverwaltung, Qualitätssicherung und Projektkoordination. In einem laufenden Projekt werden sogenannte Referenzlinien zur Sicherung von Zwischenergebnissen eingeführt. Referenzlinien bezeichnen Projektzustände, die die Entwickler und Benutzer zur Synchronisation ihrer gemeinsamen kooperativen Arbeitsprozesse vereinbart haben.

Ein Problem ist, daß Softwareentwicklung oft nur aus einer der zwei verschiedenen Perspektiven betrachtet wird. Die produktorientierte Sichtweise stellt das arbeitsteilig erstellte Softwareprodukt mit seinen spezifischen Dokumententypen und (prototypischen) Systemversionen in den Mittelpunkt. Im Gegensatz dazu umfaßt die prozeßorientierte Sichtweise den arbeitsteiligen Entwicklungsprozeß, der sich in den verschiedenen Dokumenten widerspiegelt. Es ist naheliegend, daß eine Integration beider Sichtweisen zur Meisterung der Probleme in und der Komplexität von Softwareprojekten angebracht ist.

Ausgehend von diesem Grundverständnis über den Softwareentwicklungsprozeß werden im folgenden die Merkmale von Softwareentwicklungsprojekten skizziert. Grundlage dafür sind Literaturstudien sowie Erfahrungen und Beobachtungen aus eigenen Softwareprojekten.

- *Wachsende Komplexität*: Die Anforderungen an Softwareprodukte steigen kontinuierlich. Die wachsende Komplexität von Softwaresystemen und die ständige Erweiterung und Anpassung an neue Anforderungen erfordern die Kooperation mehrerer Personen. Die Größe und Komplexität der meisten Softwareprojekte führen zwangsweise zu einer Aufteilung der Projektaufgaben. Die Zusammenarbeit findet in spezialisierten Teams und zwischen verschiedenen Teilprojekten und Bereichen wie Programmierung, Projektmanagement und Qualitätssicherung statt.
- *Nicht formalisier- und nicht automatisierbarer Prozeß:* Softwareentwicklung unterscheidet sich von der Herstellung der meisten anderen Produkte dadurch, daß der Anteil der erforderlichen kreativen Tätigkeiten wesentlich höher ist als der Anteil der erforderlichen reproduktiven Tätigkeiten. In kreativen Arbeitsprozessen ist das Resultat nicht im vorhinein bestimmt. Die Vielfältigkeit der Softwareentwicklungsaufgaben und die zu erwartenden Probleme erfordern ein hohes Maß an Flexibilität, Kreativität und ständige Lernprozesse (Floyd 1995, S. 35).
- *Hohes Risikopotential:* Die Planung und Ausführung eines Softwareentwicklungsprojektes ist aufgrund von in der Regel unvollständigen oder sich ändernden Anforderungen sowie durch die Schwierigkeiten bei der Aufteilung der Projektaufgaben mit Unsicherheiten und Risiken behaftet (Curtis et al. 1988, S. 1271 ff.). Die Anzahl von zusammenwirkenden Teilprojekten bzw. Projektgruppen, deren Entwicklungstätigkeiten sich gegenseitig beeinflussen, sind am Anfang des Projektes nicht vollständig bestimmbar.

- *Hoher Kommunikationsbedarf*: Im Gegensatz zu anderen Produktionsprozessen (z.B. für Hardwaresysteme), bedingt durch die vorher angegebenen Merkmale, ist insbesondere der informelle Informationsaustausch zwischen den Projektbeteiligten von besonderer Bedeutung. Im Gegensatz zur formellen Kommunikation, die durch ein Vorgehensmodell und die darin definierten Ergebnisse festgelegt ist, weisen Bischofberger et al. (Bischofberger et al. 1995, S. 136) auf die Notwendigkeit und Bedeutung informeller Kommunikation während des Projektablaufes hin. Die verteilte gemeinsame Entscheidungsfindung sowie die Koordination im Team erfordert meist einen spontanen und flexiblen Austausch von Informationen zwischen den Softwareentwicklern.
- *Hoher Dokumentationsbedarf:* Der Dokumentationsprozeß erfordert neben der Produktdokumentation auch die Prozeßdokumentation, um die Nachvollziehbarkeit von Entwicklungsentscheidungen zu gewährleisten. Der hohe Freiheitsgrad und das geringe Ausmaß an verfügbaren bewährten theoretischen Konzepten läßt eine Entwurfs- oder Implementierungsentscheidung nicht immer sofort nachvollziehbar erscheinen. Die Nutzbarmachung gemeinsamen Wissens über technische und administrative Entscheidungen, Schwierigkeiten und deren Ursachen und Behebung, geführte Konversationen zwischen Softwareentwicklern, Änderungs- und Fehlerstatistiken ist eine wesentliche Voraussetzung, um die Zusammenarbeit der Entwickler zu unterstützen.

Hesse/Frese (Hesse/Frese 1994, S. 179 ff.) stellten bei einer empirischen Untersuchung von 29 Softwareprojekten in 19 Firmen fest, daß der Aufwand für Abstimmungsprozesse z.B. zum Austausch von Informationen bei ca. 40 % des gesamten Entwicklungsaufwandes liegt. Je stärker die Entwicklungstätigkeiten verschiedener Projektmitarbeiter voneinander abhängen, desto intensiver müssen diese miteinander kooperieren, sich untereinander informieren und gegenseitig auf dem aktuellen Stand halten.

2.3 Merkmale von verteilten, kooperativen Softwareentwicklungsprozessen

Die Betrachtungen der generellen Merkmale des Softwareentwicklungsprozesses im vorhergehenden Abschnitt verdeutlichen unter anderem, daß Softwaresysteme einer bestimmten Komplexität nicht in monolithischer Form realisierbar sind, sondern in Teilsysteme aufgespalten werden müssen. Die damit verbundene aufgabenbezogene, räumliche und zeitliche Verteilung eines kooperativen Softwareentwicklungsprozesses weist folgende Merkmale auf:

- *Aufgabenbezogene Verteilung:* Die wachsende Komplexität von Softwaresystemen erfordert eine aufgabenbezogene Aufteilung eines Projektes in Teilprojekte; daneben aber auch eine projektübergreifende Spezialisierung innerhalb einer Entwicklungsorganisation. Entsprechend den Fähigkeiten werden Mitarbeiter bzw. Teams für spezifische Teilprozesse in der Softwareentwicklung herangezogen. Dies erfordert eine geeignete Koordinationsstrategie.

- *Zeitliche Verteilung*: In Abhängigkeit vom Projektumfang werden Teilprojekte sequentiell oder nebenläufig bearbeitet. Verschiedene Teilprojekte können gleichzeitig abgewickelt werden. Verschiedene Projektziele werden zu unterschiedlichen Terminen erreicht und erfordern daher eine Synchronisation bzw. Koordination der Arbeitsvorgänge.
- *Räumliche Verteilung:* Neben der aufgabenbezogenen und zeitlichen Aufteilung von Entwicklungsaktivitäten findet zusätzlich eine Verteilung von Projektaktivitäten auf verschiedene Teams statt, die aus organisatorischen und/oder wirtschaftlichen Gründen räumlich verteilt sein können. Die räumliche Verteilung erfordert die Bereitstellung einer geeigneten Kommunikationsinfrastruktur zur Abdeckung der Kommunikations-, Koordinations- und Kooperationsbedarfe.

Aufbauend auf dem allgemeinen Kooperationsbegriff und den bisher dargestellten Merkmalen von verteilten Softwareentwicklungsprozessen verstehen wir unter dem Begriff »Kooperative Softwareentwicklung« das Folgende:

> *Kooperative Softwareentwicklung* umfaßt die Abdeckung der Kommunikations- und Koordinationsbedarfe innerhalb eines Softwareentwicklungsprozesses, die für die Planung, Durchführung und Abstimmung aller aufgabenbezogenen, zeitlich und räumlich verteilten Aktivitäten erforderlich sind. Kooperative Softwareentwicklung umfaßt dementsprechend alle prozeß- und produktbezogenen Aktivitäten aller Beteiligten, deren gemeinsames Ziel die Erstellung eines Softwareproduktes ist.

Die verteilte, kooperative Softwareentwicklung erfordert nicht nur eine organisatorische Unterstützung, sondern bedarf auch einer geeigneten Werkzeugunterstützung. Im Bereich der rechnergestützten Gruppenarbeit sind in den letzten Jahren verschiedene Systeme zur Unterstützung kooperativer Arbeit entwickelt worden. Diese Systeme bieten in der Regel noch keine zufriedenstellende Unterstützung für die kooperative Softwareentwicklung, weil sie meist eine unzureichende konzeptionelle und technische Integration von Prozeß- und Produktsicht zur Verfügung stellen und in zu geringem Ausmaß den informellen Informationsaustausch flexibel unterstützen.

3 Gestaltungsgrundsätze für eine Werkzeugunterstützung

In diesem Abschnitt werden auf Grundlage der in Abschnitt 2 vorgestellten Merkmale von verteilten, kooperativen Softwareentwicklungsprozessen die Gestaltungsgrundsätze für eine Unterstützung der Gruppenarbeit in Softwareprojekten abgeleitet. Die identifizierten Gestaltungsgrundsätze bilden die Grundlage für die Entwicklung eines Modells für kooperative Arbeitsprozesse in Softwareentwicklungsprojekten und das Modell bildet die Grundlage für die Realisierung einer Entwicklungsumgebung zur Unterstützung kooperativer Softwareentwicklungsprozesse.

Die Gestaltungsgrundsätze im einzelnen sind:

- *Transparenz*: Der Begriff Transparenz kann im softwaretechnischen oder im anwendungsfachlichen Kontext betrachtet werden. Hier beschränken wir uns auf den anwendungsfachlichen Kontext. Angewandt auf die Unterstützung kooperativer Softwareentwicklung bedeutet das, daß die Softwareentwickler über den Archivierungsort eines Artefaktes, die Verteilung von Entwicklungsaktivitäten und die Konkurrenzsituation zu den übrigen kooperierenden Projektmitarbeitern informiert sind. Den Projektmitarbeitern soll ein nachvollziehbares Bild über begrenzte Ressourcen und ihre räumliche, zeitliche und aufgabenbezogene Verteilung vermittelt werden. Die Abstimmungsprozesse und die Auswirkungen von Zugriffen auf Artefakte sollen klar erkennbar sein.
- *Individuelle und gruppenorientierte Sichten auf den aktuellen Arbeitsprozeß:* Aufgrund der Aufgabenverteilung benötigen die Projektmitarbeiter eine individuelle Sicht auf den aktuellen Entwicklungsprozeß, um eine Übersicht über ihre Zuständigkeiten für bestimmte Aktivitäten zu erhalten. Zusätzlich ist für die kooperierenden Softwareentwickler eine gruppenorientierte Sicht notwendig, um die Aktivitäten im Gesamtkontext beurteilen und die Arbeitsvorgänge im Team synchronisieren bzw. koordinieren zu können.
- *Ein gemeinsamer Arbeitsbereich:* Der Zweck eines gemeinsamen Arbeitsbereiches besteht in der Bereitstellung und konsistenten Verwaltung von prozeß- und produktbezogenen Artefakten. Zusätzlich muß den Projektmitarbeitern eine Kooperationsumgebung zur Verfügung gestellt werden, in der sie ihre Arbeit organisieren, strukturieren und durchführen können. Die Organisation des Arbeitsbereiches umfaßt die Festlegung der dazugehörigen Mitarbeitern, der Arbeitsmaterialien und der Werkzeuge, die für die Zusammenarbeit verwenden werden können. Die Einführung von Rollen, die mit bestimmten Rechten und Pflichten verbunden sind, soll eine Differenzierung zwischen den Mitarbeitern eines Arbeitsbereiches ermöglichen und den Zugang zu den Artefakten regeln.
- *Förderung eines Gruppenbewußtseins:* Der Aufbau eines Gruppenbewußtseins erfordert, daß für alle Softwareentwickler die Struktur des Entwicklungsprozesses ersichtlich gemacht werden kann. Dies setzt einen Ereignis- bzw. Benachrichtigungsdienst voraus, der die Projektmitarbeiter über abgeschlossene und laufende Aktivitäten automatisch informiert und die Orientierung über die Aktivitäten der Kooperationspartner in einem Softwareprojekt ermöglicht. Dourish/Bellotti (Dourish/Bellotti 1992) bezeichnen das Bewußtsein über Einzel- und Gruppenaktivitäten als Awareness.
- *Aushandelbarkeit einer Konfliktlösung:* Konfliktsituationen (z.B. konkurrierender Zugriff auf Artefakte) sollen nicht vollständig durch formale und automatisierbare Ablaufbeschreibungen reguliert werden. Im Gegenteil soll die kooperative Unterstützungsumgebung so konzipiert sein, daß die Kontrolle über den kooperativen Arbeitsprozeß vollständig bei den kooperierenden Softwareentwicklern belassen wird und die eigenverantwortliche Nutzung von Handlungsspielräumen erlaubt ist (Floyd 1995, S. 34). Die Art und Weise der Kooperation im laufenden Entwicklungsprozeß kann je nach Anliegen und Sachverhalt unterschiedlich ausfallen. Kooperationsmodelle und Konventionen

beschreiben und regeln die Zusammenarbeit innerhalb eines Softwareentwicklungsteams.

- *Bereitstellung von vordefinierten Arbeitsabläufen:* Die Bearbeitungsreihenfolge von ausgewählten Entwicklungsaktivitäten in kooperativen Arbeitsprozessen soll in Form von Plänen, Anleitungen und Prozeßmustern bereitgestellt werden, die den Handlungsablauf einer Aufgabe (z.B. Codeinspektion) beschreiben. Dies soll die kooperierenden Personen bei der wechselseitigen Koordination und Abwicklung von Arbeitsschritten und bei der Festlegung von Zuständigkeiten unterstützen. Im Sinne von Suchman (Suchman 1987, S. 53) dürfen solche Koordinationspläne demnach nicht als algorithmisierte, d. h. ausführbare Vorschriften und Regeln verstanden werden. Koordinationspläne bzw. vordefinierte Arbeitsabläufe sollen eine punktuelle Anleitung und Orientierung bei der arbeitsteiligen Erledigung einer Aufgabe darstellen, ohne daß eine im formalen Sinne geschlossene Handlungsabfolge vorliegt und ohne daß sie zwingend eingehalten werden müssen.
- *Expliziter und impliziter Informationsaustausch:* Expliziter Informationsaustausch zwischen Softwareentwicklern erfolgt durch das Versenden von Nachrichten, Projektberichten, Anmerkungen, etc. Als Kooperationsmedium können elektronische Postsysteme oder elektronische Anschlagbretter eingesetzt werden. Impliziter Informationsaustausch soll dann erfolgen, wenn Änderungen an Artefakten, die sich in einem gemeinsamen Arbeitsbereich befinden, vorgenommen werden. Diese Änderungen sollen vom Werkzeug automatisch weitergeleitet und in der gemeinsamen Arbeitsumgebung sichtbar gemacht werden. Ein Ereignis- bzw. Signalmechanismus soll betroffene Projektmitarbeiter über Veränderungen an gemeinsam genutzten Artefakten benachrichtigen. Dies kann über eine Nachricht oder durch eine entsprechende Darstellung in der Benutzerschnittstelle des betroffenen Arbeitsplatzes erfolgen.
- *Transparenz der Prozeßgeschichte:* Während des Entwicklungsprozesses entsteht eine Prozeßgeschichte, die entsprechend dokumentiert und für einen späteren Zugriff den Projektbeteiligten verfügbar gemacht werden soll. Die Dokumentation soll zum Beispiel Informationen über die durchgeführten Aktivitäten, Störgrößen, Abhängigkeiten zu anderen Aktivitäten, Lösungsalternativen, Erkenntnisse und dergleichen bereitstellen. Dies verbessert das Gesamtverständnis und gewährleistet eine effiziente Problemlösung durch Wiederverwendung von Erfahrungswissen der Projektmitarbeiter.
- *Status- und Kontextinformationen:* Die von den Mitarbeitern benötigten Statusinformationen über ein Projekt sind entweder organisationsbezogener oder projektspezifischer Natur. Organisationsbezogene Informationen geben Auskunft über die Zusammenstellung der Arbeitsgruppen und deren Verantwortungsbereiche. Projektspezifische Informationen betreffen zum Beispiel den Status von Aktivitäten, Ursachen für Modifikationen und Hinweise für zu beachtende Erweiterungen. Eine kooperative Unterstützungsumgebung soll diese Status- und Kontextinformationen zur Verfügung stellen, um den Projektmitarbeitern die Orientierung im aktuellen Arbeitskontext zu erleichtern.

Aus den beschriebenen Gestaltungsgrundsätzen wird im folgenden Abschnitt ein Modell für kooperative Softwareentwicklungsprozesse abgeleitet. Das vorgestellte Modell bildet die Grundlage für die Realisierung einer geeigneten Kooperationsumgebung.

4 Ein Modell für kooperative Softwareentwicklungsprozesse

Ziel des Modells ist die Beschreibung von Entwicklungsaktivitäten zusammen mit den diesen zugeordneten Projektmitarbeitern und der Beziehungen zwischen den Arbeitsabläufen. Abbildung 2 enthält die für das Verständnis des Gesamtmodells wesentlichen Komponenten (in OMT-Notation; Rumbaugh et al. 1991).

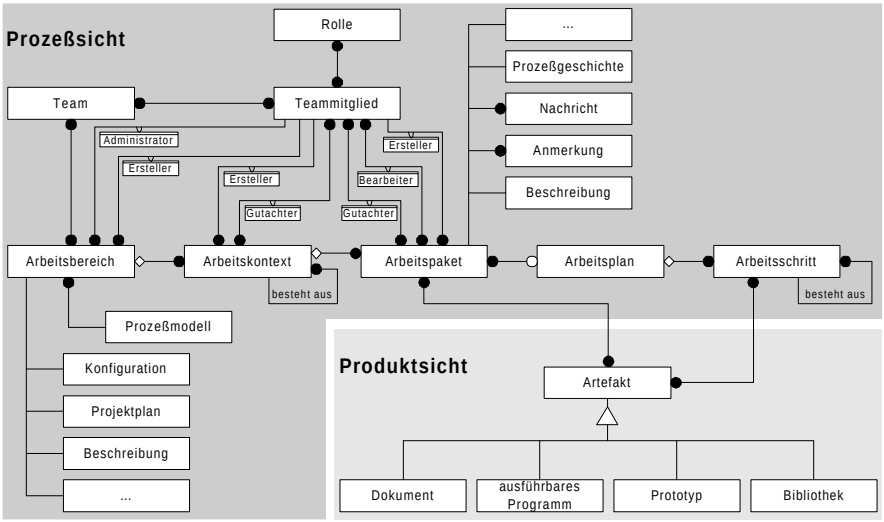

Abbildung 2: Modell für kooperative Softwareentwicklungsprozesse

Das Modell gliedert sich in zwei, logisch zusammenhängende Bereiche:

- *Prozeßsicht*: Die Komponenten zur Abbildung der Prozeßsicht, die zum Kern des Modells gehören, bilden die Basis für die Koordination und Kooperation während des Softwareentwicklungsprozesses. Das prozeßbezogene Modell beschreibt die Aufgabeninhalte der Entwicklungsaktivitäten und deren Beziehungen. Dazu gehören auch die Zuordnung der Prozeßbeteiligten zu den einzelnen Aufgaben und die Sicherstellung der Aufzeichnung des Entwicklungsverlaufes.
- *Produktsicht*: Die Produktsicht umfaßt die Ergebnisse eines Softwareprojektes. Die Entwicklungsergebnisse setzen sich unter anderem aus Dokumenten, ausführbaren Programmen, Prototypen und Bibliotheken zusammen.

Im folgenden werden die Komponenten des Modells vorgestellt und die in diesem Zusammenhang wesentlichen Begriffe erklärt.

4.1 Prozeßsicht

Das Kernstück des Modells ist das Prozeßmodell. Die prozeßbezogene Strukturierung eines Softwareentwicklungsprojektes erfolgt durch eine Aufspaltung einer Gesamtaufgabe in mehrere überschaubare Teilprojekte und Teilaufgaben. Eine Teilaufgabe wird im Modell durch ein Arbeitspaket repräsentiert und setzt sich aus einem oder mehreren Arbeitsschritten zusammen. Ein Arbeitskontext beschreibt ein Teilprojekt und verwaltet eine Gruppe von logisch zusammengehörigen Arbeitspaketen. Mehrere Arbeitskontexte zusammen bilden einen gemeinsamen Arbeitsbereich, der alle Entwicklungsaktivitäten eines bestimmten Teilprojektes zusammenfaßt. Der gemeinsame Arbeitsbereich dient den Mitarbeitern als Einstiegspunkt in ein Softwareentwicklungsprojekt und bildet die Ausgangsbasis für die Koordination und Kooperation der Entwicklungsaktivitäten. Die einzelnen Mitarbeiter können in Abhängigkeit ihrer Rolle(n) auf die prozeß- und produktbezogenen Informationen zugreifen und sich über den Status des Entwicklungsprozesses informieren. Die Aktivitäten, die mit einzelnen Artefakten durchgeführt werden, sind je nach Konfiguration eines Arbeitsbereiches für bestimmte Teilnehmer sichtbar und werden in einer Prozeßgeschichte aufgezeichnet.

Nachdem die grundsätzliche Konzeption für die Zerlegung eines Softwareentwicklungsprojektes umrissen wurde, wird zum besseren Verständnis auf ausgewählte Komponenten des Modells eingegangen.

- *Prozeßmodell*: Ein Prozeßmodell im Sinne des vorgestellten Modells definiert die Grundstruktur eines Entwicklungsprozesses, die sich aus vorgefertigten Arbeitskontexten, Arbeitspaketen und Arbeitsschritten zusammensetzt. In der Regel wird das Prozeßmodell nicht für jedes Projekt von Grund auf neu entwickelt. Generische Prozeßmodelle für unterschiedliche Arten von Softwareprojekten werden in einem Prozeßmodellkatalog abgelegt und stehen als Ausgangsbasis für die Durchführung von Entwicklungsvorhaben zur Verfügung.
- *Arbeitsschritt*: Ein Arbeitsschritt repräsentiert eine Tätigkeit innerhalb eines Arbeitspaketes. Ein Arbeitsschritt wird durch die Bezeichnung und die Beschreibung, den Bearbeitungsstatus und den Verweis auf den nächsten Arbeitsschritt repräsentiert. Die Beschreibung enthält eine Arbeitsanleitung für die Durchführung eines Arbeitsschrittes. Als Eingangsdokumente können Musterdokumente beigelegt werden. Ein Arbeitsschritt kann durch andere Arbeitsschritte verfeinert werden. Auf diese Weise entsteht eine hierarchische Struktur von Arbeitsschritten, die einen Arbeitsplan für die Bearbeitung eines Arbeitspaktes darstellen.
- *Arbeitsplan*: Ein Arbeitsplan besteht aus Arbeitsschritten. In diesem Modell ist der Arbeitsplan als eine Handlungsanleitung bzw. als ein hierarchischer Wegweiser zur Unterstützung der Bearbeitung eines Arbeitspaketes zu verstehen. Arbeitspläne stellen keine Verfahrensvorschriften dar, die das Handeln des Aufgabenbearbeiters vollständig steuern. In Abhängigkeit von den jeweiligen

Projektkonventionen, die zu Beginn eines Entwicklungsprojektes festgelegt werden, können Abweichungen von vorgegebenen Arbeitsplänen von den Paketbearbeitern begründet und in den jeweiligen Arbeitsschritten dokumentiert werden. Bewährte Arbeitspläne (»best practice«) für unterschiedliche Entwicklungstätigkeiten (z.B. Fehlerkorrektur, Integrationstest) werden in einem Katalog gesammelt und stehen für die Erstellung anderer Arbeitspakete zur Verfügung.

- *Arbeitspaket*: Ein Arbeitspaket beschreibt einen zielgerichteten Bearbeitungsauftrag, der eine logische Einheit bildet und von einem Projektmitarbeiter ausgeführt wird. Der Ersteller eines Arbeitspaketes kann die Bearbeitung entweder selbst vornehmen oder sie an einen Mitarbeiter des Projektteams delegieren. Ein Projektmitarbeiter kann die ihm zugewiesenen Arbeitspakete bearbeiten, zurückweisen oder nach erfolgter Koordination mit dem Ersteller an eine andere Person weiterleiten. Die Koordination des Arbeitsablaufes sowie die Vergabe von Zugriffsrechten auf Teile eines Arbeitspaketes liegen im Aufgabenbereich des Erstellers. Ein Gutachter ist für die Überprüfung und Abnahme der Ergebnisse eines Arbeitspaktes zuständig. Der Ersteller trägt die Verantwortung für die erarbeiteten Ergebnisse und koordiniert die Bearbeitung von Korrekturanregungen zwischen dem Gutachter und dem Bearbeiter. Ein Arbeitspaket enthält Angaben über den Ersteller, den Bearbeiter, den Gutachter, die Rolle(n), die ein Bearbeiter einnehmen muß, um die Aufgabe bearbeiten zu dürfen, den Aufgabentyp, die Priorität, den Bearbeitungsstatus, das Erstellungsdatum, das geplante und tatsächliche Fertigstellungsdatum, die Änderungsdaten und das Begutachtungsdatum. Die Dokumentation des Arbeitsfortschrittes, der erzielten Ergebnisse und der Resultate des Begutachtungsprozesses werden von den jeweiligen Projektmitarbeitern an die Paketbeschreibung elektronisch angefügt. Jedes Arbeitspaket enthält eine Prozeßgeschichte, die aus einer Liste von ausgeführten Arbeitsereignissen besteht. Das Arbeitspaket ist das wichtigste Mittel zur Unterstützung der Zusammenarbeit zwischen den Projektmitarbeitern. Durch den Austausch von Arbeitspaketen wird der arbeitsteilige Softwareerstellungsprozeß abgebildet. Angeheftete Anmerkungen (elektronische »Post-it«) und paketbezogene Nachrichten unterstützen zusätzlich die informelle Kooperation in einem Entwicklungsteam und die Dokumentation der Entwicklungsentscheidungen.
- *Arbeitskontext*: In einem Arbeitskontext werden logisch zusammengehörige Arbeitspakete verwaltet und er dient als hierarchisches Strukturierungsmittel für einen Arbeitsbereich. Ein Arbeitskontext kann wiederum durch andere Arbeitskontexte verfeinert werden. Der Ersteller eines Arbeitskontextes koordiniert die Reihenfolge und Ausführung der Arbeitspakete und trägt die Verantwortung für die Ergebnisse des Arbeitskontextes. Die Gutachter eines Arbeitskontextes überprüfen das Gesamtergebnis eines Arbeitskontextes, das sich aus den Teilresultaten der einzelnen Arbeitspakete zusammensetzt. Der Ersteller eines Arbeitskontextes leitet die von den Gutachtern erstellten Korrekturanregungen an die Verantwortlichen der jeweiligen Arbeitspakete weiter und überwacht die termingerechte und korrekte Ausführung der Überarbeitungsaufträge. Arbeitskontexte gliedern einen gemeinsamen Arbeitsbereich in unterschiedliche Interessenschwerpunkte. Dadurch wird die

Orientierung in kooperativen Entwicklungsprozessen gefördert, da sich die Koordination und Kooperation nicht mehr lediglich auf einzelne Arbeitspakete beschränkt, sondern sich in einem größeren Zusammenhang, dem Arbeitskontext, vollzieht.

- *Arbeitsbereich*: Ein Arbeitsbereich umfaßt mehrere Entwicklungsaktivitäten, die zur Erreichung eines gemeinsamen Zieles einer bzw. mehrerer Projektgruppen erforderlich sind. Ein Arbeitsbereich ist hierarchisch in unterschiedliche Arbeitskontexte und Arbeitspakete gegliedert. Die Beschreibung eines Arbeitsbereiches enthält sämtliche Informationen, die zur Planung und Ausführung eines Projektes notwendig sind. Die wichtigsten Artefakte sind der Projektauftrag und die Beschreibung der Projektorganisation. Die Struktur eines Arbeitsbereiches, bestehend aus Arbeitskontexten und Arbeitspaketen, wird von den Projektmitarbeitern laufend ergänzt und überarbeitet. In bestimmten Intervallen überprüft der Projektleiter die Arbeitsbereichsstruktur und paßt sie an den aktuellen Bearbeitungsprozeß an. Eine präskriptive Beschreibung des Entwicklungsprozesses wird vom vorgestellten Arbeitsbereich nicht unterstützt, da sie der geringen Vorbestimmtheit und hohen Komplexität von Softwareentwicklungsprozessen nicht entspricht.

4.2 Produktsicht

Die Produktsicht bezieht sich auf das zu entwickelnde Softwareprodukt. Das Softwareprodukt setzt sich aus ausführbaren Programmen, Bibliotheken mit wiederverwendbaren Softwarebausteinen, Prototypen und Entwicklungsdokumenten zusammen. Die Produktsicht wird abstrakt durch ein Artefakt repräsentiert, das in mehreren Instanzen mit verschiedenen Ausprägungen vorkommen kann.

Artefakt: Ein Artefakt repräsentiert einen Arbeitsgegenstand im Softwareentwicklungsprozeß. Sowohl das ausführbare Programmsystem selbst als auch sämtliche Dokumente des Entwicklungsprozesses stellen Artefakte dar.

In diesem Abschnitt wurde das Basismodell vorgestellt, das zur Modellierung, d.h. zur Strukturierung und zur Beschreibung eines Softwareentwicklungsprozesses vorgeschlagen wird. Im Vordergrund steht die Vergegenständlichung der Zuständigkeiten der einzelnen Projektmitarbeiter, die wechselseitige Abstimmung des Arbeitsablaufes und die Beschreibung von Arbeitsszenarien.

5 Cooperation Assistant – Eine Arbeitsumgebung für kooperative Softwareentwicklung

In diesem Abschnitt wird eine Arbeitsumgebung für kooperative Softwareentwicklung vorgestellt, die die praktische Umsetzung des im Abschnitt 4 beschriebenen Modells für kooperative Arbeitsprozesse in Softwareprojekten ermöglicht.

Die Arbeitsumgebung unterstützt räumlich und zeitlich getrennt agierende Projektmitarbeiter im Kontext typischer Softwareentwicklungstätigkeiten.

Die Arbeitsumgebung *Cooperation Assistant* wurde als Experimentierplattform am C. Doppler Labor für Software Engineering an der Johannes Kepler Universität Linz (Altmann/Weinreich 1998) entwickelt. Die in *Cooperation Assistant* realisierten Konzepte zur Unterstützung verteilter, kooperativer Softwareentwicklung beruhen auf dem Ziel, die produkt- und prozeßbezogene Sichtweise harmonisch miteinander zu vereinen. Die Umgebung basiert auf dem Ansatz gemeinsamer Arbeitsbereiche. Eine kooperativ zu lösende Aufgabe wird im Arbeitsbereich als Arbeitspaket dargestellt.

Die Vorstellung aller Werkzeuge dieser umfangreichen Arbeitsumgebung und das Eingehen auf Implementierungsdetails würde den Rahmen dieses Beitrags überschreiten. Im folgenden wird der *Cooperation Assistant* daher nur überblicksartig anhand eines Anwendungsszenarios vorgestellt. Als Implementierungsprache wurde aus Gründen der Kompatibilität mit anderen zu integrierenden Softwareentwicklungswerkzeugen und wegen der Verbreitung die Sprache C++ gewählt. Als Implementierungsplattform wurde das im C. Doppler Labor entwickelte Framework *ObjectWire* (Weinreich/Altmann 1997), das die Realisierung verteilter, objektorientierter Softwarearchitekturen besonders unterstützt, herangezogen.

Abbildung 3 zeigt einen Überblick über ausgewählte Werkzeuge der Kooperationsumgebung aus der Sicht eines Projektmitarbeiters. Der *Cooperation Assistant* stellt Werkzeuge für die Strukturierung, Überwachung und Bearbeitung von kooperativen Arbeitsprozessen zur Verfügung. Aus Platzgründen beschränken wird uns auf die Beschreibung der Werkzeuge Agenda ①, Arbeitskontextverwalter ② und Arbeitspaketeditor ③.

Abbildung 3 zeigt die Projektstruktur des Entwicklungsprojektes »ObjectWire - In Progress«, die in der Agenda ① dargestellt ist. Die dargestellte Struktur des Softwareentwicklungsprojektes gibt dem Projektmitarbeiter »josef« ⑨ einen Überblick über die logische Zusammengehörigkeit der einzelnen Entwicklungstätigkeiten der Projektgruppe. Das Gesamtprojekt besteht in diesem Anwendungsszenario aus den Teilprojekten »Monitoring«, »Configuration« und »Cooperative Software Development«, die zu Projektbeginn vom Projektleiter festgelegt wurden.

Die Überwachung der Reihenfolge und des Arbeitsfortschrittes der Entwicklungstätigkeiten eines Teilprojektes erfolgt im Arbeitskontextverwalter ②. Abbildung 3 zeigt einen Arbeitskontextverwalter, der alle laufenden und abgeschlossenen Arbeitspakete des Arbeitskontextes »Configuration« darstellt. Der Ereignisdienst der kooperativen Arbeitsumgebung erzeugt bei einem Zustandsübergang eines Arbeitspaketes ein Ereignisobjekt. Ein Zustandsübergang findet zum Beispiel beim Erzeugen, Versenden, Ablehnen oder Akzeptieren eines Arbeitspaketes statt. Der Benachrichtigungsdienst informiert die Projektmitarbeiter über die stattgefundenen Ereignisse. Die Ereignisse werden im Arbeitskontextverwalter neben den betroffenen Arbeitspaketen grafisch dargestellt ⑩, z.B. bedeutet das Symbol, daß das Arbeitspaket verändert wurde.

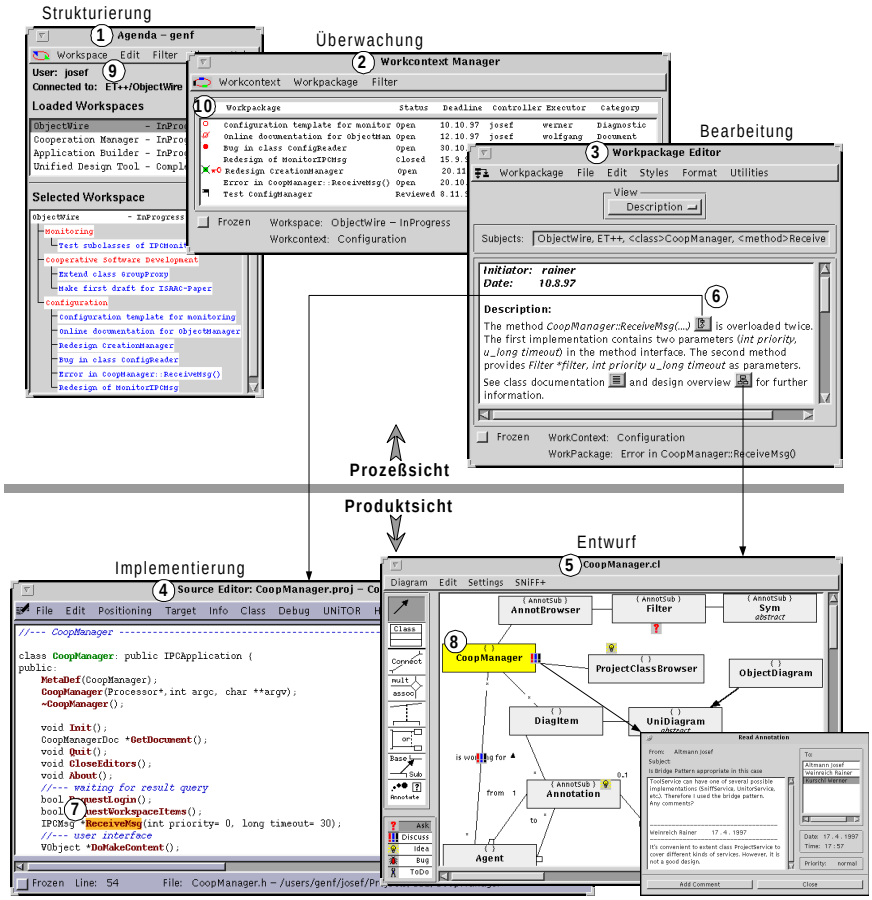

Abbildung 3: Anwendungsszenario

Der Arbeitspaketeditor ③ unterstützt die gemeinsame Bearbeitung von Entwicklungsaufgaben. Die Beschreibung eines Arbeitspaketes spiegelt sowohl die Aufgabenbeschreibung als auch den Bearbeitungsverlauf eines Arbeitspaketes wider. Der Arbeitspaketeditor stellt Werkzeuge zur Bearbeitung eines Arbeitspaketes und zur Unterstützung der Kommunikation und Koordination zwischen den Arbeitsbereichsangehörigen zur Verfügung. Die Verbindung der Prozeßsicht mit der Produktsicht erfolgt durch die Integration von bestehenden Softwareentwicklungswerkzeugen. Im Anwendungsbeispiel wurden die Programmierumgebung SNiFF ④ von der Firma *TakeFive* (vgl. SNIFF+ 1998) und ein am C. Doppler Labor entwickeltes Entwurfswerkzeug ⑤ in den *Cooperation Assistant* eingebunden. Die für die Produktsicht verfügbaren Werkzeuge können daher von Projekt zu Projekt verschieden sein.

Arbeitspaketbeschreibungen, Arbeitspläne, Hinweise und Kommentare werden in Form von Verweisen sowohl zueinander als auch zu Dokumenten in Beziehung gesetzt. Die in der Beschreibung des Arbeitspaketes eingefügten Verweise ergänzen die Aufgabenspezifikation um jene Softwareentwicklungsdokumente, die für die Bearbeitung des Arbeitspaketes benötigt werden. Die Aktivierung eines Verweises ⑥ führt zu einem Aufruf der externen Anwendung. Im Anwendungsszenario wird an jene Stellen in der Implementierung ⑦ und im Entwurf ⑧ verzweigt, die mit der Aufgabenstellung des Arbeitspaketes im Zusammenhang stehen. Anschließend kann der Softwareentwickler mit der Bearbeitung der Aufgabe im jeweiligen Softwareentwicklungswerkzeug beginnen. Die Ergebnisse des Bearbeitungsvorganges werden vom Paketbearbeiter im Arbeitspaketeditor dokumentiert. Am Ende des Bearbeitungsvorganges wird das Arbeitspaket als abgeschlossen gekennzeichnet und automatisch dem zuständigen Begutachter, der mit dem Arbeitspaketeditor festgelegt wurde, weitergeleitet.

Das oben beschriebene Anwendungsszenario läßt bereits erkennen, daß im *Cooperation Assistant* das in Abschnitt 4 dargestellte Modell für kooperative, verteilte Softwareentwicklungsprozesse implementiert ist. Der *Cooperation Assistant* unterstützt die Zusammenarbeit durch den Austausch von gemeinsamen Arbeitsgegenständen und deckt den gesamten Softwareentwicklungsprozeß ab. Allgemeine Funktionalität zur prozeßorientierten Verwaltung von kooperativen Arbeitsabläufen wurde mit der produktorientierten Sicht auf die Artefakte verknüpft. Ebenso können externe Softwareentwicklungswerkzeuge mit Hilfe von Werkzeugadaptern in die Arbeitsumgebung integriert werden. Den Autoren ist keine verfügbare Softwareentwicklungsumgebung bekannt, die sowohl die Prozeß- als auch die Produktsicht gleichermaßen unterstützt und ebenso flexible Kommunikations-, Koordinations- und Kooperationsmechanismen anbietet und darüber hinaus das Einbinden von Fremdwerkzeugen gestattet.

6 Zusammenfassung und Ausblick

Mit dem vorgestellten Modell für kooperative Softwareentwicklung und der darauf aufbauenden Entwicklungsumgebung *Cooperation Assistant* wird einerseits ein Beitrag zur Beseitigung von Leistungsdefiziten in Softwareentwicklungsumgebungen geleistet und andererseits eine Experimentierumgebung zur Verfügung gestellt, die es gestattet, auf empirischer Basis die These zu untermauern oder zu widerlegen, daß der kooperative, cluster-orientierte Entwicklungsansatz im Hinblick auf die Ausschöpfung von Produktivitäts- und Qualitätssteigerungspotentialen dem rigiden phasenorientiertem Entwicklungsansatz deutlich überlegen ist, insbesondere bei räumlich und zeitlich verteilter Projektorganisation.

Die Arbeiten zur Konzeption des Modells und die Implementierung der Entwicklungsumgebung sind abgeschlossen. Mit der Evaluierung des vorgeschlagenen Ansatzes wurde begonnen.

Die bisher vorliegenden Erfahrungen mit dem Einsatz des *Cooperation Assistant* bestätigen die Annahme, daß der kooperative, cluster-orientierte Entwicklungsansatz sowohl produktivitäts- als auch qualitätssteigernd wirkt. Der vorgestellte Ansatz zeichnet sich vor allem durch die leichte Verständlichkeit des Modells, die intuitive Bedienbarkeit der Werkzeuge und die übersichtliche Darstellung von prozeß- und produktbezogenen Informationen aus.

Die Evaluationsforschungsaktivitäten sind jedoch noch nicht abgeschlossen und die derzeit verfügbaren Analysedaten sind für eine empirische Auswertung und Schlußfolgerung im Hinblick auf eine Bestätigung bzw. Widerlegung der oben angeführten These noch nicht ausreichend. Die Evaluation wird auch noch einen längeren Zeitraum in Anspruch nehmen, da nur die Ergebnisse einer entsprechend großen Anzahl auch umfangreicher, komplexer Softwareprojekte eine solide Beurteilungsgrundlage bilden.

Derzeit noch offen, und damit Gegenstand weiterer Forschungs- und Entwicklungsarbeiten sind die Integration von Netzplantechniken zur Termin- und Aufgabensteuerung und -überwachung, der Einbezug (d.h. die Gewinnung und Verwendung) von Metriken zur Prozeßoptimierung und die Konzeption und Bereitstellung von Werkzeugen zur benutzergesteuerten Analyse, Filterung und Sortierung von produkt- und prozeßbezogenen Informationen. Es ist noch nicht ausreichend geklärt, welchen Beitrag die automatische Aufzeichnung der Prozeßgeschichte zur Verbesserung der laufenden Projektabwicklung und zur nachfolgenden Produktverwaltung zu leisten imstande ist.

Literaturverzeichnis

Altmann, J. (1998): Kooperative Softwareentwicklung: Rechnerunterstützte Koordination und Kooperation in Softwareprojekten. Dissertation, Institut für Wirtschaftsinformatik, Johannes Kepler Universität Linz 1998.

Altmann, J./Weinreich, R. (1998): An Environment for Cooperative Software Development: Realization and Implications. In: Proceedings of the Thirty-First Annual Hawaii International Conference on System Sciences, Collaboration Systems and Technology. Ed.: J. F. Nunamaker. Los Alamitos 1998, S. 27-37.

Balzert, H. (1998): Lehrbuch der Software-Technik: Software-Management, Software-Qualitätssicherung, Unternehmensmodellierung. Heidelberg 1998.

Bauknecht, K./Mühlherr, T./Sauter, C./Teufel, S. (1995): Computerunterstützung für die Gruppenarbeit. Bonn 1995.

Bischofberger, W. R./Kofler, T./Mätzel, K.-U./Schäffer, B. (1995): Computer Supported Cooperative Software Engineering with Beyond-Sniff. In: Proceedings of Software Engineering Envrionments. Ed.: M. S. Verrall. Los Alamitos 1995, S. 135-143.

Boehm, B. W. (1988): A Spiral Model of Software Development and Enhancement. In: IEEE Computer, 21(1988)5, S. 61-72.

Curtis, B./Kellner, M. I./Over, J. (1992): Process Modelling. Communications of the ACM, 35(1992)9, S. 75-90.

Dourish, P./Bellotti, P. (1992): Awareness and Coordination in Shared Workspaces. In: Proceedings of the Conference on Computer-Supported Cooperative Work. Ed.: J. Turner, R. Kraut. Baltimore 1992, S. 107-114.

Floyd, C. (1995): Theory and Practice of Software Development. In: Proceedings of the Sixth International Conference on Theory and Practice of Software Development. Ed.: P. D. Mosses, M./Nielsen, M. I./Schwartzbach. New York 1995, S. 25-41.

Floyd, C./Züllighoven, H. (1997): Softwaretechnik. In: Informatik Handbuch. Ed.: P. Rechenberg, G. Pomberger. Wien 1997, S. 641-665.

Friedrich, J. (1994): Defizite bei der software-ergonomischen Gestaltung computerunterstützter Gruppenarbeit. In: Menschengerechte Goupware: Software-ergonomische Gestaltung und partizipative Umsetzung. Ed.: Hartmann A./Herrmann, T./Rohde, M. /Wulf, V.: Stuttgart 1994, S. 13-30.

Garg, P. K./Jazayeri, M. (1995): Process-Centered Software Engineering Environments: A Grand Tour. Technischer Bericht, Abteilung Verteilte Systeme, Technische Universität Wien 1995.

Hesse, W./Frese, M. (1994): Zur Arbeitssitutation in der Software-Entwicklung. Informatik Forschung und Entwicklung, 9(1994)3, S. 179-191.

Meyer, B. (1997): Object-Oriented Software Construction. London 1997.

Pasch, J. (1994): Software-Entwicklung im Team. Berlin 1994.

Pomberger, G./Blaschek, G. (1996): Software Engineering: Prototyping und objektorientierte Software-Entwicklung. München 1996.

Pomberger, G./Heinrich, L. (1998): Prototyping-orientierte Evaluation von Software-Angeboten und Software-Entwicklungsprojekten - Konzepte und Fallstudien. In: Tagungsband der Internationalen Konferenz Evaluation und Evaluationsforschung in der Wirtschaftsinformatik, Oldenbourg 1998.

Popitz, H./Bardt, H. P./Jüres, E. A./Kesting, A. (1957): Technik und Industriearbeit. Soziologische Untersuchungen in der Hüttenindustrie. Tübingen 1957.

Rumbaugh, J./Blaha, M./Premerlani, R./Eddy, W./Lorensen, W. (1991): Object-Oriented Modelling and Design. Englewood Cliffs 1991.

SNiFF (1998): TakeFive Home Page. http://www.takefive.com.

Sommerville, I. (1996): Software Engineering. 5th Ed., Harlow 1996.

Suchman, L. (1987): Plans and Situated Actions. Cambridge UK 1987.

Weinreich, R./Altmann, J. (1997): An Object-oriented Infrastructure for a Cooperative Software Development Environment. In: Proceedings of the Fifth International Symposium on Applied Corporate Computing. Ed.: Y. Martinez-Treviño. Monterrey, Mexiko 1997, S. 45-53.