Ansätze zur kollaborativen Softwareerstellung

Die Autoren

Tobias Hildenbrand
Franz Rothlauf
Armin Heinzl

Dipl.-Wirtsch.-Inf. Tobias Hildenbrand
Dr. Franz Rothlauf
Prof. Dr. Armin Heinzl
Universität Mannheim
Lehrstuhl für ABWL und Wirtschaftsinformatik
68131 Mannheim
{hildenbrand | rothlauf | heinzl}
@uni-mannheim.de

Eingereicht am 2006-08-01,
nach zwei Überarbeitungen
angenommen am 2006-12-22
durch Prof. Dr. W. König.

1 Einleitung

Die wachsende Komplexität von Software und deren zunehmende Bedeutung in unterschiedlichen Anwendungsdomänen stellen erhöhte Anforderungen an die Softwareerstellungsprozesse. In der Theorie und Praxis findet ein Wandel hin zur Standardisierung und Wiederverwendung von Komponenten und Prozessen sowie zur Automatisierung von Prozessschritten („Industrialisierung") statt. Hierbei wird die vorherrschende Werkstatt- bzw. Einzelfertigung durch eine an industrielle Fertigungsprinzipien angelehnte verteilte Softwareentwicklung sukzessive abgelöst. Diese Entwicklung verstärkt sich durch die Globalisierung der Softwareerstellung weiter. Anstatt monolithischer Einzelfertigungen können Vorprodukte ausgewiesener Zulieferer mit entsprechenden Mitteln der Softwaretechnik, wie Komponentenmodellen, zu größeren Softwarebausteinen zusammengefasst werden. Über einheitliche Standards, etwa Plattformstrategien oder gemeinsame Entwicklungsprozesse sowie verbesserte Werkzeugunterstützung lassen sich Synergieeffekte durch Spezialisierung, Automation von Arbeitschritten und rationellere Abstimmung der Prozessbeteiligten erzielen.

Wissenschaftliche Arbeiten diskutieren zunehmend Methoden und Techniken zur Unterstützung der Spezialisierung, der Arbeitsteilung und der verteilten Softwareerstellung. Im Rahmen eines vorindustriellen Softwareerstellungsprozesses werden Anwendungen nicht mehr zentral durch eine Softwaremanufaktur, sondern dezentral und gleichzeitig durch unterschiedliche Akteure entwickelt. Dies ermöglicht eine Steigerung der Effizienz durch Arbeitsteilung, führt jedoch zu einem erhöhten Bedarf an Koordination und Kommunikation zwischen den Prozessbeteiligten. Die im Bereich der „kollaborativen Softwareerstellung" (*collaborative software development*) entwickelten Methoden und Werkzeuge sollen diese Änderungen unterstützen.

Dieser Beitrag verfolgt zwei Ziele: Zum einen wird ein Klassifikationsrahmen für die unterschiedlichen Ansätze zur kollaborativen Softwareerstellung entwickelt und zum anderen werden existierende Ansätze und Forschungsarbeiten in diesen Rahmen eingeordnet, charakterisiert und verglichen. Diese Klassifikation der bestehenden Ansätze soll es Unternehmen, die eine arbeitsteilige, stark verteilte und zusammenarbeitsintensive Erstellung von Software

Kernpunkte

Die Erstellung von Software wird zunehmend verteilt und nach vorgegebenen Abläufen organisiert. Der Beitrag gibt einen systematischen Überblick über kollaborative Erstellungsmethoden und -werkzeuge.

- Die Ansätze haben jeweils einen gruppenorientierten, einen prozessorientierten oder einen werkzeugbezogenen Fokus und lassen sich anhand der Kategorien Verteilung, Prozessrichtung und Zusammenarbeitsintensität charakterisieren.
- Gerade in organisatorisch und räumlich verteilten Projektszenarien lassen sich Probleme feststellen. Die Klassifikation und Analyse bestehender Ansätze soll es Unternehmen ermöglichen, die relevanten Methoden und Werkzeuge zu identifizieren.
- Ein Großteil der Arbeiten befinden sich noch in einem vortheoretischen Stadium. Forschern der Wirtschaftsinformatik wird angeraten, Fragen der kollaborativen Softwareerstellung stärker vor dem Hintergrund von Verhaltenstheorien zu behandeln.

Stichworte: Kollaboration, Softwareentwicklung, Software Engineering, Industrialisierung, Softwareerstellungsprozess, kollaborative Softwareerstellung, Kooperation

anstreben, ermöglichen, die für ihre jeweiligen Bedürfnisse relevanten Methoden und Werkzeuge zu identifizieren und entsprechend einzusetzen.

Methodisch lässt sich der vorliegende Beitrag wie folgt einordnen [vgl. Fett06]: Die Analyse der Primärliteratur fokussiert sich auf Forschungsergebnisse in Form von Artefakten (Methoden und Werkzeugen) sowie Erfahrungen in Form einzelner Fallbeispiele. Die untersuchte Literatur wird selektiv in drei Kategorien von Ansätzen (gruppenorientiert, prozessorientiert und werkzeugzentriert) unterteilt. Im folgenden Abschnitt wird zunächst das begriffliche Fundament geschaffen und die einzelnen Dimensionen des eingesetzten Klassifikationsrahmens vorgestellt. Abschnitt 3 klassifiziert und beschreibt die unterschiedlichen Ansätze und gliedert diese thematisch in die drei genannten Kategorien. In Abschnitt 4 werden die wesentlichen Ergebnisse zusammengefasst und vor dem Hintergrund relevanter Verhaltenstheorien diskutiert. Abschließend zeigt der Beitrag zukünftige Herausforderungen und Forschungsfragen auf. Der Beitrag richtet sich primär an Forscher in der Wirtschaftsinformatik, bietet aber anhand von Fallstudien auch Anregungen und Handlungsvorschläge für Praktiker.

2 Kollaborative Softwareerstellung

Im Folgenden wird auf die begrifflichen Grundlagen für kollaborative Softwareerstellungsprozesse eingegangen und ein Klassifikationsrahmen für unterschiedliche Ansätze in der Literatur geschaffen. Auf dieser Basis können kollaborative Entwicklungsszenarien definiert und kategorisiert werden.

Die im Rahmen dieses Beitrags verwendete Zusammenarbeitsbegriffe „Kollaboration“ und „Kooperation“ orientieren sich an der englischsprachigen Literatur aus den Bereichen *Computer-Supported Collaborative Work* (CSCW) und *Software Engineering* (SE). Hier werden diese weitestgehend synonym zur Beschreibung einer verteilten Bearbeitung gemeinsamer Softwareartefakte verwendet. Einige Autoren unterscheiden verschiedene Ebenen der Zusammenarbeit: Diese bestehen in (a) Informieren, (b) Koordinieren, (c) Zusammenwirken („Collaboration“) und (d) Kooperation [Bair89; Krcm92]. Kooperation findet durch gleichberechtigte Teilnehmer statt, wobei individuelle Ziele einem *kollektiven* Gruppenziel untergeordnet werden [Krcm92, 427] und eine ständig intensive Interaktion vorherrscht [Bair89, 210]. Des Weiteren zeichnet sich Kooperation gegenüber den anderen Ebenen durch einen gemeinsamen Prozess und gemeinsame Outputs, bspw. gemeinsam erstellte Dokumente, sowie die kollektive Bewertung des Teams und deren Ergebnisse aus [Bair, 210].

Demgegenüber werden bei einer Kollaboration weiterhin *individuelle*, ggf. antagonistische Ziele verfolgt [Bair89; Altm99], die häufig dann auftreten, wenn unterschiedliche Abteilungen und/oder Unternehmen am Softwareerstellungsprozess beteiligt sind, bspw. wenn die IT-Abteilung eines großen Unternehmens zusammen mit externen Beratern und der eigenen Finanzabteilung eine neue Softwarekomponente zur Rechnungslegung entwickelt. Die Wahrscheinlichkeit einer Zielkongruenz zwischen den beteiligten Entwicklern sinkt mit einer zunehmenden Aufgabenverteilung. Die Konnotation antagonistischer Ziele kommt u. a. darin zum Ausdruck, dass unter dem Begriff der Kollaboration neben einer verteilten Zusammenarbeit vereinzelt auch die „Zusammenarbeit mit dem Feind“ (hier: Wettbewerber) verstanden wird [Dude06].

Die Kollaborative Softwareentwicklung (KSE, *Collaborative Software Development*, kurz CSD, teilweise auch: *Cooperative Software Development*) beschäftigt sich im Allgemeinen mit Methoden und Werkzeugen zur Abdeckung der Kommunikations- und Koordinationsbedarfe innerhalb eines Softwareentwicklungsprozesses, die für die Planung, Durchführung und Abstimmung aller aufgabenbezogenen sowie organisatorisch, zeitlich und räumlich verteilten Aktivitäten erforderlich sind. Diese sollen alle prozess- und produktbezogenen Aktivitäten der Beteiligten, deren Ziel die Erstellung eines (gemeinsamen) Softwareprodukts ist, abdecken [vgl. Altm99, 20]. Speziell von Kollaboration in der Softwareerstellung wird gesprochen, wenn im Rahmen einer verteilten Durchführung von Softwareentwicklungsaktivitäten eine häufige Interaktion mit intensivem Wissensaustausch der beteiligten Akteure notwendig ist, und ggf. antagonistische Ziele vorliegen [Bair89; HPGF02; MeSz03]. Bezogen auf mögliche unterschiedliche Dimensionen der Aufgabenverteilung wird ein Softwareentwicklungsprozess dann als „kollaborativ“ bezeichnet, wenn er die folgenden Eigenschaften aufweist (nähere Erläuterungen siehe unten):

- organisatorische Verteilung (über mehrere Abteilungen und/oder Unternehmen),
- zeitliche Verteilung (asynchrone Zusammenarbeitsprozesse) und
- räumliche Verteilung (über mehrere Standorte und/oder Stockwerke).

Diese drei Verteilungsdimensionen bilden die zentralen Determinanten der KSE. Eine stärkere organisatorische, zeitliche oder räumliche Verteilung eines Softwareerstellungsprozesses wirkt sich negativ auf die Zielkongruenz der Beteiligten aus und grenzt KSE damit von der rein kollektiven Kooperation im SE ab [vgl. Bair89; Krcm92]. Es ist zu beachten, dass die Übergänge zwischen der rein kollektiven, nichtkollaborativen Kooperation und der KSE fließend sind und ein Softwareentwicklungsprozess mit zunehmender organisatorischer, räumlicher oder zeitlicher Verteilung in stärkerem Maße kollaborativ wird.

Neben diesen drei Verteilungsmerkmalen führt die stärkere Ausprägung folgender prozessbezogener Merkmale zu mehr Kollaboration (nähere Erläuterungen siehe unten):

- vertikale, interdisziplinäre Zusammenarbeit (über mehrere Phasen mit unterschiedlichen Rollen) bei gleichzeitig
- hoher Zusammenarbeitsintensität (bez. des zeitlichen Umfangs und der Häufigkeit).

In Analogie zur organisatorischen, zeitlichen und räumlichen Verteilung wird der kollaborative Charakter eines Softwareerstellungsprozesses mit zunehmender Ausprägung dieser beiden Kriterien stärker. KSE-Prozesse stellen in der Regel erhöhte Anforderungen an die (Werkzeug-)Unterstützung aufgabenbezogener Koordinations- und Kommunikationsprozesse [HeMo03; MeSz03; HRDS04].

Zusammenfassend unterscheidet sich KSE von anderen Softwareentwicklungsprozessen dahingehend, dass sie in höherem Maße „organisatorisch“, „zeitlich“ oder „räumlich“ verteilt abläuft. Trotz dieser Verteilung soll im Rahmen eines Softwareentwicklungsprojekts jedoch weiterhin phasenübergreifend (interdisziplinär) und intensiv zusammengearbeitet werden.

Bevor die einzelnen Dimensionen der o. g. Definition im Folgenden näher erläutert werden, soll der Charakter eines KSE-Projekts beispielhaft anhand eines Offshore-Projekts verdeutlicht werden. In diesem sind in der Regel mindestens drei Unternehmen (Kunde, Softwarehersteller und Offshore-Dienstleister) beteiligt. Aufgrund von Zeitverschiebung und Effi-

zienzvorteilen wird häufig zeitlich, räumlich und organisatorisch verteilt gearbeitet. Eine strikte Trennung der Verantwortlichkeiten und Verteilung der Aufgaben nach Phasen hat sich allerdings häufig als problematisch erwiesen [CaAg01; HeMo03]. Aus diesem Grund wird eine intensive Zusammenarbeit der beteiligten Akteure über alle Phasen hinweg verfolgt [HiGN06]. Hierfür werden spezielle Methoden und Werkzeuge benötigt, bspw. für das Verfolgen von Erweiterungen, Änderungen und Abhängigkeiten ([SoHR07], siehe Abschnitt 3.3.3).

In den vorhergehenden Absätzen wurde gezeigt, dass sich die KSE sowohl anhand unterschiedlicher Dimensionen der Verteilung, der prozessbezogenen Richtung als auch der Intensität der gemeinsamen Entwicklungsaktivitäten charakterisieren lässt. Im Folgenden werden die entsprechenden Dimensionen näher erläutert und aus der Literatur begründet:

Organisatorische Verteilung, also die Verteilung von Arbeitsschritten über mehrere betriebliche Organisationseinheiten, wird in der Literatur häufig auch als *unit of distribution* mit den Ausprägungen inter-organisatorisch (zwischenbetrieblich) und intra-organisatorisch (innerbetrieblich) bezeichnet [MeSz03]. Starke organisatorische Verteilung, d. h. die Zusammenarbeit von vielen organisatorisch getrennten Gruppen, hat zahlreiche organisatorische, ökonomische und juristische Implikationen für den Softwareerstellungsprozess [MeSz03].

Bei der räumlichen Verteilung von Softwareentwicklungsaktivitäten unterscheidet man räumlich nahe zusammenarbeitende (*collocated*) und räumlich verteilte (*distributed*) Organisationseinheiten [Altm99]. Fehlender persönlicher Kontakt bei der täglichen Zusammenarbeit wirkt sich selbst bei kurzen Distanzen und räumlichen Barrieren, wie die Verteilung über mehrere Stockwerke eines Gebäudes, unmittelbar auf Kommunikations- und Kollaborationsformen aus [Bair89; Krcm92]. Zahlreiche Arbeiten im Bereich *Distributed Software Development* (DSD) beschäftigen sich daher speziell mit den soziotechnischen Implikationen räumlich verteilter Softwareprojekte [HeMo03]. Arbeiten in diesen Projekten zudem mehrere rechtlich selbstständige Unternehmen zusammen (inter-organisatorische Verteilung, s. o.) oder werden Aktivitäten über mehrere Zeitzonen und Kulturräume hinweg verteilt (*Offshore Software Development*), kann sich dies zusätzlich auf die Kommunikations- und Koordinationsgestaltung auswirken, bspw. durch die höhere organisatorische und/oder nationale Distanz sowie sprachliche Probleme [CaAg01].

Die bekannten Klassifikationen der Zusammenarbeit im CSCW-Bereich unterscheiden neben den beiden bisher genannten Dimensionen auch die zeitliche Verteilung der Zusammenarbeit [Bair89; Krcm92]. Hierbei geht es darum, ob Entwicklungsaufgaben innerhalb eines Teams gleichzeitig (synchron) oder zu unterschiedlichen Zeitpunkten (asynchron) bearbeitet werden. Informationen, insbesondere die Softwareartefakte wie Anforderungsbeschreibungen und Modelle, werden dabei entweder synchron bearbeitet oder gepuffert [Krcm92; DeDe04]. Herrscht keine ständig unmittelbarer Synchronisation der Zusammenarbeit wird häufig auch von nebenläufiger Softwareentwicklung gesprochen (*Concurrent Software Engineering*) [DeRi93, Altm99, HRDS04]. Diese Art der Zusammenarbeit findet bezüglich der Koordination asynchron statt, auch wenn keine Zeitverschiebung oder ähnliche zeitliche Hindernisse existieren.

Die Prozessrichtung gibt an, ob die Zusammenarbeit innerhalb oder zwischen bestimmten Wertschöpfungsstufen im Softwareerstellungsprozess stattfindet. Der SE-Prozess lässt sich prinzipiell in einzelne Prozessstufen bzw. inhaltliche Disziplinen unterteilen. Diese bestehen bspw. in „Analyse und Design“ sowie „Implementierung“ (vgl. *Rational Unified Process*, kurz RUP). Der Begriff „Disziplin“ aus dem RUP wird hier bewusst dem der „Phase“ vorgezogen, da letzteres ein sequenzielles Wasserfallmodell impliziert. Somit können bezüglich der Prozessrichtung zwei Ausprägungen unterschieden werden: innerhalb einzelner Disziplinen spricht man von horizontaler Zusammenarbeit und bei interdisziplinären Zusammenarbeitsformen von vertikaler Kollaboration (zur prozessualen Unterstützung von Gruppenarbeit vgl. [Krcm92]). Im Zuge der zunehmenden Industrialisierung und organisatorischen Verteilung unterscheiden einzelne Autoren die Prozessrichtung auch anhand zwischenbetrieblicher „Software-Wertschöpfungsketten“ (*Software Value Chains*, [MeSz03]). Bei der unternehmensübergreifenden Zusammenarbeit innerhalb eines solchen „Softwareökosystems“ ergibt sich die Richtung aus dem Verhältnis der Kollaborationspartner in der Wertschöpfungskette [MeSz03]. Horizontale Kollaborationsbeziehungen finden sich zwischen Unternehmen derselben Wertschöpfungsstufe [MeSz03], wohingegen vertikale Beziehungen eine „Lieferanten-Kunden-Beziehung“ ausdrücken, bspw. wenn ein Unternehmen die Anforderungsspezifikation erstellt und ein anderes auf dieser Basis das zugehörige Grobkonzept [HiRH06].

Die Zusammenarbeit im Softwareerstellungsprozess kann mit unterschiedlich hoher Intensität der Interaktion betrieben werden (Zusammenarbeitsintensität). Dieser Wert ist umso größer, je mehr Informationen zwischen unterschiedlichen Akteuren ausgetauscht werden und je häufiger bzw. umfangreicher dies geschieht [Bair89; CaAg01]. Die Intensität wird dadurch bestimmt, ob tatsächlich gemeinsam an einem Objekt gearbeitet wird oder die Zusammenarbeit mittelbar über den systematischen Austausch von wissensbasierten Leistungen und halbfertigen Softwareartefakten wie Komponenten erfolgt [DLPH98; MeSz03]. KSE-Prozesse, sowohl horizontal als auch vertikal, weisen ungleiche Grade des Mitwirkens (*Involvement*) der Beteiligten an kollektiven Bemühungen auf [Bair89, 209], was sich in unterschiedlichen Rollen manifestiert. Die Intensität zwischenbetrieblicher (organisatorisch verteilter) Softwareerstellung lässt sich aus juristischer Sicht dadurch klassifizieren, inwieweit es sich um eine lose oder eine vertraglich bzw. kapitalmäßig geregelte Beziehung handelt. Joint Ventures bspw. erfordern im Gegensatz zu strategischen Allianzen gemeinsamen Kapitaleinsatz und resultieren in einer intensiveren Zusammenarbeit, da auch Kosten und Erträge auf die beteiligten Unternehmen verteilt werden [HiRH06].

Im folgenden Abschnitt sollen unterschiedliche Ansätze, die ein oder mehrere dieser Kriterien erfüllen, vorgestellt und analysiert werden.

3 Ansätze zur kollaborativen Softwareerstellung

Unter Verwendung des Klassifikationsrahmens lassen sich Ansätze zur KSE wie folgt untergliedern:

- vorhandene Arbeitsgruppen (im Fokus stehen Menschen und soziotechnische Systeme),
- zugrunde liegendes Prozessmodell (aufgabenbezogene und methodische Fragestellungen) und
- eingesetzte Kollaborationswerkzeuge (softwaretechnische Instrumente und Werkzeuge).

Allen Ansätzen liegt die Problemstellung zugrunde, dass im Rahmen von SE-Prozessen zahlreiche unterschiedliche, möglicher-

weise verteilte Akteure mit dem Ziel der systematischen Verbesserung der Zusammenarbeit organisiert werden müssen [DLPH98]. Hierfür lassen sich gruppenorientierte, prozessorientierte und werkzeugzentrierte Lösungsansätze unterscheiden.

3.1 Allgemeine gruppenorientierte Ansätze

Gruppenorientierte Ansätze zeichnen sich durch die zentrale Betrachtung des Faktors „Mensch“ sowie die Betonung soziotechnischer Fragestellungen aus. Es stehen die Zusammensetzung der Arbeitsgruppen, Koordinationsmechanismen und Kommunikationsmittel, beobachtete Interaktionsmuster sowie die davon abhängige individuelle und kollektive Leistung im Mittelpunkt. Die im Folgenden analysierten Ansätze bieten Handlungsempfehlungen, die auf sozialwissenschaftlichen Verhaltenstheorien (bspw. Gruppendynamik) beruhen und im jeweiligen Wirtschaftsinformatikkontext empirisch ermittelt wurden. Im Zentrum steht bei den gruppenorientierten Ansätzen die Fragestellung, wie man Gruppenarbeit im Kontext der KSE (vgl. Definition in Abschnitt 2) optimal organisiert und unterstützt. Vorgehensmodelle und unterstützende Werkzeuge zur Umsetzung werden in diesen Ansätzen allerdings nicht spezifiziert.

Zahlreiche Arbeiten setzen sich mit dem Gruppenverhalten und der optimalen Zusammensetzung von Softwareentwicklungsteams auseinander – bspw. im Hinblick auf ein effektiveres Projektmanagement und eine verbesserte Teamleistung. Im Zentrum stehen die Kommunikation und die Koordination der Arbeitsabläufe mehrerer, möglicherweise räumlich verteilter Gruppenmitglieder während des Softwareerstellungsprozesses. *Team Coordination Strategies* [AnZm02] und *Work Team Diversity* [HPGF02] stellen Ansätze zur Koordination räumlich nicht verteilter Kollaborationsszenarien mit hoher Intensität und synchronen sowie vor allem asynchronen Kommunikationsformen (zeitliche Verteilung) dar. Die zentrale These von Adres und Zmud besteht darin, dass „organische Koordinationsstrategien“ (informelle Kommunikation, gemeinsame Entscheidungsfindung bei dezentraler Entscheidungsautonomie) in Softwareentwicklungsteams erfolgreicher sind als andere [AnZm02], wohingegen bei dem letzteren Ansatz die soziale Integration der Teammitglieder trotz etwaiger Diversitäten im Vordergrund steht.

Layzell et al. untersuchen speziell das Verhalten von räumlich verteilt arbeitenden Teams (*Distributed Teams*) und geben Richtlinien für die aufgabenorientierte Organisation verteilter Entwicklergruppen [LaBF00]. Dabei wird von einer primär synchronen und intensiven Form der Zusammenarbeit ausgegangen. Für solche Szenarien werden regelmäßige physische Treffen und die Rotation von Mitarbeitern über mehrere Standorte, ein proaktives Projektmanagement sowie gemeinsame Standards, vor allem zur Qualitätssicherung, empfohlen. Die fundamentalen Probleme verteilter Teams sind laut Layzell et al. sozialer und organisatorischer Natur und weniger technologisch bedingt [LaBF00]. *Global Virtual Teams* betrachten neben den Effekten räumlicher Verteilung auch solche der organisatorischen Verteilung in Form zwischenbetrieblicher Zusammenarbeit auf Teamebene [MRMK00]. Zusätzlich zur Kommunikation und Koordination innerhalb der einzelnen Entwicklergruppen werden hierbei auch die Zusammenarbeit zwischen mehreren Gruppen innerhalb eines Projekts untersucht und Handlungsempfehlungen sowohl für räumlich verteilte als auch für nicht verteilte Teams abgeleitet. So sollen insbesondere virtuelle Teams die Zusammenarbeit schnell und systematisch aufnehmen und intensivieren. Der Faktor „Vertrauen“ spielt in räumlich und organisatorisch verteilten *Virtual Teams* eine besonders große Rolle. Hierzu muss man den Teams zugestehen, Technologien, das organisatorische Umfeld und die Teamstruktur selbst anzupassen [MRMK00].

Neben den vorgestellten Ansätzen für die Gruppenorganisation in zusammenarbeitsintensiven Szenarien existieren Arbeiten, die empirisch häufig auftretende Muster der Kommunikation und Interaktion identifizieren. Diese Muster beschreiben kollaborative Beziehungen zwischen räumlich verteilt arbeitenden Gruppenmitgliedern und deren Kommunikationsmittel (*Patterns of Contact and Communinaction* [KrEG88]). Sie analysieren Regelmäßigkeiten bei der Arbeitsorganisation, insbesondere Aktivitäten und Interaktionen in Softwareprojekten (*Patterns of Cooperative Interaction*, [MaSo04]). Unter Verwendung einer ethnomethodischen Perspektive werden dabei empirische Muster für die Zusammenarbeit in Kleingruppen von Softwareentwicklern beschrieben, die bei der Gestaltung neuer Arbeitsgruppen und Vorgehensmodellen helfen sollen [MaSo04].

Ähnlich wie bei den Mustern lassen sich auch archetypische Ausprägungen von Entwicklergruppen (*Team Archetypes*) differenzieren [Sawy04]. Dieser Ansatz geht davon aus, dass weder das Vorgehensmodell noch die Entwicklungswerkzeuge einen Einfluss auf Software-Produktqualität und Gruppenleistung haben und fokussiert die sozialen Prozesse in Softwareprojekten. Dabei werden drei grundlegende Typen von Entwicklungsprojekten identifiziert. Der „sequenzielle Typ“ orientiert sich an den Prinzipien industrieller Produktionstechnik, was bedeutet, dass Teammitglieder in spezialisierten Funktionen durch formalisierte Interaktionen zwischen diesen Funktionen kommunizieren. Dies entspricht einem Wasserfallmodell, wobei in den einzelnen Phasen, bspw. bei der Anforderungsanalyse, Spezialisten eingesetzt werden ([Sawy04, 96], vgl. nächster Abschnitt). Der Typ „Gruppe“, der auf sozialpsychologischen Theorien basiert, ist zusammenarbeitsintensiver. Viele Aufgaben werden dabei gruppen- und funktionsübergreifend bearbeitet. Dies entspricht eher der Definition von KSE (vgl. Abschnitt 2) und impliziert, dass auch phasenübergreifend (vertikal) häufiger zusammengearbeitet wird. Der Typ „Netzwerk“ bildet sich um das Softwareprodukt und Aufgaben werden entsprechend der vorhandenen Fähigkeiten an Individuen und Kleingruppen verteilt. Hierbei bilden sich die Teams bspw. um bestimmte Softwarekomponenten, die spezielle Kenntnisse, wie Netzwerkprogrammierung, erfordern. Diese Erkenntnisse dienen der situationsgerechten Gestaltung von Teamstrukturen in unterschiedlich zusammenarbeitsintensiven KSE-Szenarien.

Die analysierten Arbeiten bieten Hinweise auf erste verhaltenstheoretisch und empirisch ermittelte Ansätze zur KSE – allerdings keine konkreten Umsetzungsempfehlungen in Form von Prozessen und Werkzeugen. Daher werden im Folgenden weitere prozessorientierte und werkzeugzentrierte Ansätze untersucht, die versuchen, die organisatorische, räumliche und zeitliche Trennung zu überwinden, um eine vertikale, intensive Zusammenarbeit zu ermöglichen.

3.2 Prozessorientierte Ansätze

Prozessorientierte Ansätze betrachten weniger soziotechnische Fragestellungen, sondern formal strukturierte Vorgehensweisen bei der Softwareerstellung, bspw. in Form eines Prozess- oder Vorgehensmodells. Beispiele hierfür sind sequenzielle und iterative Modelle (Wasserfall- und Spiralmodell) sowie neuere Prozessrahmenwerke wie der RUP. Zur Strukturierung der Prozesse die-

nen die einzelnen Phasen des Softwarelebenszyklus, von der Erhebung der Kundenanforderungen hin zur Fertigstellung der Software. Da bisher keine Vorgehensmodelle für einen vertikal durchgängigen KSE-Prozess (vgl. Abschnitt 2) existieren, werden die Ansätze im Folgenden hinsichtlich unterschiedlicher Phasen charakterisiert. Es werden Arbeiten aus den Bereichen (1) Anforderungsanalyse, (2) Entwurf und Modellierung sowie (3) Implementierung, Test und Wartung unterschieden. Im Anschluss wird in einem Fallbeispiel dargelegt, wie ein vertikal integrierter Zusammenarbeitsprozess aussehen bzw. in welchen Szenarien ein derartiges Vorgehensmodell Verwendung finden könnte. Dies soll die praktische Relevanz der methodischen Unterstützung von KSE-Projekten weiter unterstreichen.

3.2.1 Kollaborative Anforderungsanalyse

Bisher wurden speziell bei der Anforderungserhebung und -analyse unterschiedliche kollaborative Ansätze entwickelt und erprobt. Da in dieser frühen Projektphase die Interessen vieler Anspruchsgruppen (*Stakeholder*) koordiniert werden müssen [Somm06, 118], wurden in diesem Bereich bereits einige interessante Ansätze entwickelt. Diese unterscheiden sich in der Art der unterstützten Verteilung sowie der Werkzeugunterstützung. Das WinWin-Spiralmodell basiert auf der nach dem Harvard-Konzept der Verhandlungstechnik entwickelten *Theory W* [BoBo94]. Grundannahme ist hierbei, dass der Erfolg von KSE-Projekten davon abhängt, dass alle beteiligten Parteien Positives aus der Kollaboration ziehen und nicht die Anforderungen einer Partei überrepräsentiert werden. Um dies zu erreichen sollen asymmetrische *Win-Lose*-Situationen zwischen einzelnen *Stakeholders* durch systematische Verhandlungs- und Moderationstechniken vermieden werden, um das Gesamtrisiko des Projekts zu minimieren. WinWin stellt eine Anwendung dieser Theorie und gleichzeitig eine Weiterentwicklung des klassischen Spiralmodells dar. Dabei werden sich verändernde Interessen unterschiedlicher Stakeholder sowie deren Koordination über mehrere Iterationen hinweg berücksichtigt, indem diese kontinuierlich konsolidiert werden (hohe Intensität der Zusammenarbeit). Die EasyWinWin-Methodik stellt die mittlerweile vierte Generation des WinWin-Ansatzes dar [Grue00]. EasyWinWin (EWW) zeichnet sich durch einen flexiblen, iterativen Erhebungsprozess aus, der es auch ermöglicht, im laufenden Prozess neue Beteiligte zu integrieren. Außerdem wird der Aufbau von gegenseitigem Vertrauen durch physische Treffen und moderierte Diskussionen gefördert. Allerdings ist die Methodik komplex und trotz spezieller *Groupware* nicht primär für ein verteiltes Umfeld konzipiert. Die Zusammenarbeit findet mit hoher Intensität und teilweise asynchron statt. Die Groupware-basierte EWW-Variante ARENA ermöglicht eine Komplexitätsreduktion, ein räumlich verteiltes Arbeiten und einen asynchronen Informationsaustausch, bspw. durch die Protokollierung von Diskussionen [BoGB01]. ARENA erlaubt ausschließlich asynchrones, aber dafür verteiltes und mobiles Zusammenarbeiten bei der Anforderungserhebung. Allerdings ist ARENA aus technischen Gründen nicht mehr mit der ursprünglichen WinWin-Groupware kombinierbar [BoGB01].

3.2.2 Kollaborative Entwurfs- und Modellierungsprozesse

Entwurfs- und Modellierungsprozesse setzen Anforderungen in Architektur- bzw. detailliertere Komponentenmodelle als Basis für die anschließende Implementierung um. Kollaborative Ansätze in dieser Disziplin wurden häufig aus anderen Ingenieurdisziplinen übertragen (bspw. Computer-Aided bzw. Concurrent-Design-Prozesse im Automobilbau). Erste Ansätze, auf die hier nicht weiter eingegangen wird, waren *Joint Application Design* (JAD) [Augu91] und *Participatory Design* (PD) [MuKu93].

Im *Concurrent Software Engineering* (kurz CSE) laufen im Gegensatz zu JAD und PD möglichst viele Entwicklungsaktivitäten parallel und überlappend ab [DeRi93]. Das bedeutet, dass dabei nicht nur synchron zusammengearbeitet wird [vgl. auch Altm99]. *Collaborative Software Engineering Methodology* (CSEM) beschreibt als CSE-Variante ebenfalls die nebenläufige, asynchrone Zusammenarbeit unterschiedlicher Beteiligter im Rahmen des Softwareentwurfsprozesses [DLPH98]. Nutzer sollen in drei unterschiedlichen Prozessstufen eingebunden werden: zunächst wenige Repräsentanten, dann kleinere Benutzergruppen und schließlich die gesamte Nutzergemeinschaft. Nebenläufige Entwurfsprozesse werden auch immer stärker in UML-basierten Methoden zur modellgetriebenen Softwareentwicklung (*Model-Driven Development*, kurz MDD) und automatischen Quelltexterzeugung integriert. Wenn auch bisher nur wenige fundierte Methoden existieren, ermöglichen moderne Entwicklungswerkzeuge, wie bspw. *Gentleware Poseidon* und *IBM Rational XDE* nebenläufige verteilte und modellgetriebene Softwareerstellungsprozesse (*Concurrent MDD*, kurz CMDD). Hierbei unterstützt die Software die asynchrone, verteilte Zusammenarbeit, indem die Modelle, ähnlich wie der Quelltext in Konfigurationsmanagementsystemen, zentral verwaltet und versioniert werden. Hierzu bieten die genannten Werkzeuge serverseitige Repositories, die Änderungen in den Modellen registrieren und Unterschiede feststellen können. Konflikte durch Änderungen unterschiedlicher Entwickler an einem Modell sollen somit, wie bei der Code-Versionierung in einem CVS-System, reduziert werden.

3.2.3 Kollaboration in Implementierung, Test und Wartung

Ansätze zur kollaborativen Implementierung, Test und Wartung weisen durch ihren Bezug zum Quelltext eine höhere Werkzeugorientierung auf als jene der beiden bisher behandelten Phasen. Zusammenarbeitsintensive Prozesse in der Implementierungsphase können synchron, in Form von *Shared Workspaces* bzw. *Application Sharing* (*Collaborative Editing*) und/oder asynchron durch Versionierungssysteme unterstützt werden [HRDS04]. Für das gemeinsame Editieren von Quelltextdateien existieren mittlerweile zahlreiche Werkzeuge, wie *SubEthaEdit*. Die Entwickler sehen hierbei die jeweiligen Änderungen ihrer räumlich verteilten Kollegen synchron direkt im Dokument vor sich. Allerdings kommt die methodische Unterstützung der implementierungsbezogenen Zusammenarbeitsprozesse noch zu kurz. In der Praxis sowie in der Literatur lässt sich zunehmend eine „Rückbesinnung“ auf die soziotechnischen Probleme innerhalb der KSE beobachten (vgl. Abschnitt 3.1). Im Rahmen dieser Entwicklung sind vor allem personenzentrierte Ansätze für kleine bis mittelgroße Entwicklergruppen hervorzuheben, so genannte agile Entwicklungsmethoden, wie bspw. *Extreme Programming* (XP) [Beck99]. Charakteristische zusammenarbeitsintensive Prozesse im Rahmen von XP sind paarweise Programmierung, was bedeutet, dass immer zwei Entwickler gleichzeitig an einem Stück Quelltext arbeiten (einer schreibt und einer begutachtet das Ergebnis), und eine starke Einbindung des Kunden vor Ort. Auch

XP bietet nur wenige methodische Vorschriften und sieht keine räumlich und zeitlich verteilte Zusammenarbeit vor. Auch eine mögliche organisatorische Verteilung der Entwicklungsaufgaben wird bei diesem Ansatz nicht expliziert. Allerdings existieren erste Ansätze, XP durch entsprechende Werkzeuge wie Voice-over-IP, Videokonferenzen und paralleles, synchrones Editieren von Quelltext (*Collaborative Editing*, s. o.) auch in einem räumlich verteilten Umfeld einzusetzen.

Im Open-Source-Bereich haben sich zusammenarbeitsintensive, codezentrierte Prozesse etabliert, die auch in großen, verteilten Projekten effizient und effektiv sind [FeFi01]. Bei der Open-Source-Softwareentwicklung (OSSE) wird der Quelltext für alle Entwickler und sonstige Beteiligte offen zugänglich verwaltet. Ein Versionierungswerkzeug wie z. B. CVS hat hierbei eine Doppelfunktion: zum einen als zentrales Koordinationsinstrument durch Rechtevergabe bzw. Versionskontrolle sowie als Distributionssplattform im Internet [FeFi01]. Zur effizienteren Koordination der stark verteilten Implementierungs- und Testaktivitäten hat nur eine begrenzte Anzahl an Entwicklern schreibenden Zugriff auf den Quelltext. Diese müssen Änderungen und neue Funktionen der übrigen Entwickler erst begutachten und genehmigen (*Peer Review and Approval*, vgl. [FeFi01]). Derartige zusammenarbeitsintensiven, asynchronen Praktiken können auch kommerzielle Entwicklungsprojekte beschleunigen, insbesondere durch die Verwendung von OSSE-Entwicklungsswerkzeugen [AuBS02]. Auf diese Werkzeuge soll detaillierter im nachfolgenden Abschnitt 3.3 eingegangen werden. Die CSI- bzw. CAIS-Methode (*Collaborative Software Inspection* bzw. *Collaborative Asynchronous Inspection of Software*) ist ein verteilter Ansatz zur kollaborativen Qualitätssicherung auf Hypertext-Basis als Teil der Postimplementierungsphase (Test). Er bietet eine werkzeuggestützte Methode zur verteilten, asynchronen Quelltextinspektion, die darauf abzielt, synchrone physische Treffen zu minimieren [MDTR93]. Ebenso in diese Kategorie fallen Ansätze zur verteilten und asynchronen Wartung bereits entwickelter Software (*Collaboration in Software Maintenance*, kurz CSM, [LoRo93]). Der Ansatz unterstützt die verteilte Zusammenarbeit mehrerer Softwareingenieure, indem ebenfalls ein hypertextbasiertes System zum Festhalten von Wartungsentscheidungen und deren Bezug zum Quelltext bereitgestellt wird.

3.3 Werkzeugzentrierte Ansätze

Bei den Werkzeugen zur Unterstützung zusammenarbeitsintensiver Softwareerstellungsprozesse lässt sich eine Evolution der Entwicklungsumgebungen bezüglich der integrierten Kommunikationsmöglichkeiten und der Integration verteilter Softwareartefakte beobachten. Im Folgenden werden zunächst in Abschnitt 3.3.1 (1) Groupware-basierte Ansätze, (2) integrierte Entwicklungsumgebungen sowie (3) zentrale Plattformen für verteilte, zusammenarbeitsintensive Entwicklungsprozesse analysiert. Des Weiteren werden in Abschnitt 3.3.2 existierende Plattformansätze mittels der KSE-Dimensionen analysiert und abschließend anhand eines kommerziellen Projektbeispiels illustriert.

3.3.1 Werkzeugkategorien

Bereits Ende der achtziger Jahre wurden in Softwareprojekten CSCW-Werkzeuge [siehe ScKr96] in Verbindung mit separaten Editoren und einfachen Werkzeugen zur Quelltextverwaltung genutzt [DeDe04]. Groupware aus dem CSCW-Bereich unterstützt zusammenarbeitsintensive, räumlich verteilte Teamarbeit sowie zeitlich verteilte als auch synchrone Zusammenarbeit [Krcm92; ScKr96]. Drei Anwendungsfälle in der KSE sind (1) „lokal und synchron“ (bspw. in der Anforderungsanalyse, vgl. Abschnitt 3.2.1), (2) „verteilt und synchron“ (z. B. *Collaborative Editing*, siehe Abschnitt 3.2.3) sowie (3) „verteilt und asynchron“ (etwa bei Gruppenkalendern und Versionierungssystemen) von besonderer Bedeutung [Krcm92; DeDe04]. Die Systeme werden innerhalb einer Organisation über den gesamten Softwarelebenszyklus hinweg eingesetzt. In der Literatur beschäftigen sich auch zahlreiche empirische Arbeiten mit dem Einfluss von Groupware auf KSE-Prozesse [KTCB92; LaBF00; DeDe04], bspw. zur gemeinsamen Problemlösung oder Entscheidungsfindung. Diese sind vor allem für den Einsatz in räumlich stark (global) verteilten KSE-Projekten von Bedeutung [HeMo03].

Integrierte Entwicklungsumgebungen (*Integrated Development Environments*, kurz IDE) hingegen versuchen, dem einzelnen Entwickler eine einheitliche Oberfläche und Arbeitsumgebung zu präsentieren und mehr individuelle Produktivität zu gewährleisten, indem mehrere Entwicklungswerkzeuge integriert und standardisiert werden. IDEs vereinen neben dem Quelltexteditor Compiler, grafische Werkzeuge für den Zugriff auf Versionierungssysteme, Groupware-Applikationen ebenso wie Werkzeuge für Architekturentwurf, *Unit Testing* und *Debugging*. Die von IDEs bereitgestellte Funktionalität wird häufig auch unter dem Begriff *Computer-Aided Software Engineering* (*CASE Tools*) zusammengefasst [Somm06, 85]. IDEs sind primär auf die Bedürfnisse individueller Entwicklerarbeitsplätze zugeschnitten und unterstützen die verteilte Zusammenarbeit von Entwicklerteams daher nur ansatzweise, z. B. durch die Anbindung an ein Versionierungssystem [Somm06, 88]. Sie werden in der Regel innerhalb einer Organisation eingesetzt und unterstützen keinen durchgängigen vertikalen Prozess hoher Intensität, sondern einzelne horizontale Disziplinen wie Implementierung und Test (siehe Abschnitt 3.2.3). Als Beispiele für IDEs lassen sich die *Open-Source*-Lösungen *Eclipse* und *NetBeans* sowie kommerzielle Werkzeuge von IBM Rational und Borland nennen. Neuere Ansätze versuchen bewusst, diese IDEs um CSCW-Werkzeuge zu erweitern, um somit eine bessere Unterstützung für räumlich verteilte Softwareprojekte und einen gemeinsamen Kontext zu schaffen (*contextual collaboration*) [CDHP03; HRDS04].

Kollaborationsplattformen (*Collaborative Software Development Platforms*, CSDPs) bieten zentrale Koordinations- und Kommunikationsfunktionen für asynchrone Zusammenarbeitsprozesse räumlich stark verteilter Entwicklerteams über das Internet an, wie sie im Rahmen von *Open-Source*-Projekten bereits erfolgreich eingesetzt werden [FeFi01; AuBS02; Robb05]. Es existieren neben ersten kommerziellen Lösungen auch zahlreiche quelloffene und experimentelle Ansätze aus Forschungsprojekten. Solche Plattformen, wie bspw. *SourceForge*, stellen spezialisierte Erweiterungen serverbasierter CSCW-Lösungen dar [AuBS02]. Die grundlegenden Funktionalitäten einer Kollaborationsplattform lassen sich in die Kategorien Koordination und Kollaboration sowie *Community Building* unterteilen [Bair89]. Neben Versionierungssystemen für Quelltext und andere Softwareartefakte sowie weiteren Kommunikationswerkzeugen (Foren, Wikis etc.) bieten CSDPs eine detaillierte Aufgabenverwaltung an (*Issue Tracker*) [Robb05]. CSDPs ermöglichen somit die Nachvollziehbarkeit einzelner Prozessschritte sowie der Beziehungen zwischen Softwareartefakten (*Traceability,* vgl. [Somm06, 163f.]). Der Zugriff auf solche Kollaborationsplattformen erfolgt entweder über eine IDE oder per Browser

[Robb05]. Für den zwischenbetrieblichen Einsatz können CSDP von einem unabhängigen Anbieter als *Application Service* über das Internet bereitgestellt werden. Im Gegensatz zu IDEs sollen CSDPs einen vertikal durchgängigen Entwicklungsprozess abdecken. Neben asynchronem Informationsaustausch werden hierbei immer häufiger auch synchrone Zusammenarbeitsprozesse durch Werkzeuge wie Chat oder IP-Telefonie unterstützt [CaAg01; HeMo03]. Solche serverbasierten Lösungen decken damit ein Höchstmaß an Funktionalität für die KSE ab (vgl. Abschnitt 2).

3.3.2 Experimentelle Kollaborationsplattformen

Neben den kommerziellen existieren experimentelle Kollaborationsplattformen und Prototypen. Im Folgenden wird auf Besonderheiten im Vergleich zu am Markt erhältlichen, kommerziellen CSDPs eingegangen [vgl. Robb05]. Ingesamt wurden zur Vorbereitung dieses Beitrags 12 Plattformen analysiert [siehe HiRH06]. Bei keiner der experimentellen Plattformen werden intra- und interorganisatorische Zusammenarbeitsszenarien systematisch unterschieden und unterstützt. Die Ansätze lassen sich anhand der unterschiedlichen Berücksichtigung zeitlichen Verteilung sowie der unterstützten Zusammenarbeitsintensität differenzieren. Umfassende Unterstützung von KSE-Prozessen bieten ConversationBuilder, GENESIS, JAZZ und MILOS. Diese Ansätze werden im Folgenden näher erläutert:

ConversationBuilder unterstützt unter anderem eine gemeinsame Verwaltung versionierter Artefakte und die verteilte Zusammenarbeit in Form von *Conversations* [KTCB92]. GENESIS (*GEneralised eNvironment for procEsS management in cooperatIve Software engineering*) umfasst ein Workflow- und Artefaktmanagementsystem und bietet umfangreiche Koordinations- und Kommunikationsfunktionen. Es werden sowohl prozess- als auch produktorientierte Softwareprojekte unterstützt, wobei GENESIS mit einer eigenen Sprache zur Prozessdefinition arbeitet [GaRi02]. Das JAZZ-Projekt, das von IBM unterstützt wird, stellt eine kollaborative Erweiterung zur quelloffenen Eclipse-IDE dar. Es vertritt den Ansatz, die Nutzer in ihrer gewohnten IDE zu belassen und lediglich kontextbezogene Erweiterungen für die Kollaboration in Softwareprojekten zu bieten [CDHP03]. MILOS bietet Werkzeuge zur Prozessmodellierung, Projektplanung und -umsetzung sowie Rückverfolgbarkeit und Änderungsbenachrichtigung. Zusätzlich existiert eine Anbindung an *Microsoft Project* [MSHK99].

Eine vollständige tabellarische Übersicht aller analysierten Ansätze aus den Abschnitten 3.1 bis 3.3.2 (gruppenorientiert, prozessorientiert und werkzeugzentriert) mit den einzelnen Dimensionen und Charakteristika von KSE-Prozessen findet sich in [HiRH06].

3.3.3 Fallbeispiel des Einsatzes einer Kollaborationsplattform

Kollaborationsplattformen haben sich vor allem in stark verteilten, zusammenarbeitsintensiven Szenarien, wie bspw. im Open-Source-Bereich oder in global verteilten Offshore-Entwicklungsprojekten (vgl. Abschnitt 2) bewährt. Der Einsatz traditioneller Prozesse und Werkzeuge in Offshore-Projekten führt aufgrund „vertikaler Brüche“ zwischen den einzelnen Disziplinen immer wieder zu Problemen [HiGN06].

Die Firma *VA Software* ist ein Hersteller von Kollaborationssoftware für große Entwicklungsprojekte. VA Software setzt zur Weiterentwicklung ihrer beiden Hauptprodukte *SourceForge.net* (frei zugängliche Plattform für OSSE-Projekte, vgl. Abschnitt 3.2.3) sowie *SourceForge Enterprise* (kommerzielle Lösung für Unternehmen), eine Kollaborationsplattform ein. Der KSE-Prozess ist dabei zwischen mehreren Standorten in den USA und Chennai, Indien, organisatorisch, räumlich und durch die Zeitverschiebung zeitlich verteilt. Die verwendete Entwicklungsmethodik orientiert sich an der OSSE, was in diesem Fall bedeutet, dass die einzelnen Entwicklerteams (a) interdisziplinär und (b) interorganisatorisch (zwischen den einzelnen Divisionen in den USA und Indien) zusammengesetzt sind. Die Entwicklungsaufgaben werden komponentenbezogen verteilt, sodass jedes der organisatorisch und räumlich verteilten Teams eine Komponente über alle vertikalen Phasen des Entwicklungszyklus hinweg bearbeitet und verantwortet. Durch die starke Verteilung sowie durch die Abhängigkeiten der Softwarekomponenten untereinander wird eine hohe Zusammenarbeitsintensität im Projekt notwendig, da trotz möglicher Zielkonflikte zwischen den Teams die gemeinsame Architektur eingehalten und weiterentwickelt werden muss [HiGN06]. Die dadurch entstehenden erhöhten Anforderungen bezüglich Information, Koordination und Kollaboration (vgl. Abschnitt 2) werden durch die in Abschnitt 3.3.1 genannten Funktionen unterstützt. Zu den eingesetzten Funktionen zählen Wiki-Webs und Foren für asynchrone Kommunikation und Wissensmanagement, Versions- und Konfigurationsmanagementwerkzeuge zur Koordination der Änderungen am Quelltext, ein Aufgabenverwaltungssystem (*Issue Tracker*) zur Prozesskoordination sowie entsprechende IDE-Erweiterungen für die Unterstützung räumlich verteilter Kollaboration auf Entwicklerebene. Eine Stärke dieser plattformbasierten Lösung liegt in der Vernetzung der Inhalte innerhalb eines gemeinsamen Projektkontexts [CDHP03; HRDS04], die auch eine bessere Nachvollziehbarkeit der KSE-Prozesse (*Traceability*) ermöglicht (siehe auch [HiGN06]).

Das kontextbasierte Wissen über die Abhängigkeiten zwischen einzelnen Artefakten erleichtert die Koordination der Aufgabenerfüllung und die Kommunikation der Team-Mitglieder [SoHR07]. Insbesondere im Änderungsmanagement können so wertvolle Informationen für das Abschätzen des Ausmaßes und der Auswirkungen von Änderungen an Softwareartefakten (*Change Propagation* und *Impact*-Analyse) extrahiert und bei Änderungen gezielt Benachrichtigungen an die betroffenen Entwickler versendet werden [SoHR07]. Gerade in stark verteilten Projekten, wie dem hier genannten Offshore-Projekt (organisatorisch, räumlich und zeitlich verteilt), sind solche Funktionen essenziell. Werden diese nicht realisiert und die KSE-Prozesse vertikal nicht integriert, kann dies sehr leicht zu Informationsverlusten führen.

4 Schlussbetrachtung

Ziel dieses Beitrags ist es, einen systematischen Überblick über existierende Ansätze zur KSE zu liefern. Zu diesem Zweck werden drei Dimensionen der KSE sowie unterschiedliche Kategorien von Ansätzen identifiziert. Der entwickelte Klassifikationsrahmen setzt sich zum einen aus der organisatorischen, räumlichen und zeitlichen Verteilung sowie zum anderen der Prozessrichtung und der Zusammenarbeitsintensität zusammen. Der Beitrag unterscheidet Ansätze mit soziotechnischen, methodischen oder werkzeugbezogenen Schwerpunkten.

Vergleicht man die analysierten Ansätze, lässt sich zusammenfassend feststellen, dass die rein werkzeugzentrierten Ansätze des SE deutlich überwiegen. Existierende Vorgehensmodelle und Prozessrahmenwerke gehen jedoch weder dediziert und über alle Phasen des Softwarelebenszyklus hinweg

auf die unterschiedlichen Dimensionen und Merkmale der KSE ein, noch werden die einzelnen horizontalen Ansätze integriert (vgl. Abschnitte 3.2.1 bis 3.2.3). Die hier betrachteten methodischen Ansätze fokussieren lediglich bestimmte Phasen wie bspw. die Anforderungsanalyse. Überdies betrachten nur wenige Ansätze soziale und verhaltenstheoretische Fragestellungen bez. zusammenarbeitsintensiver Softwareerstellung und die damit verbundenen soziotechnischen Probleme (siehe Abschnitt 3.1). Viele Entwicklungsmethoden und Werkzeuge, die immer häufiger auch in groß angelegten, verteilten Projekten im Unternehmenskontext eingesetzt werden, haben ihren Ursprung in der OSSE, wie im Fallbeispiel 3.3.3 gezeigt wurde. Allerdings sind diese Methoden und Werkzeuge nicht ohne weiteres in den kommerziellen Bereich übertragbar und erfordern methodische Anpassungen [vgl. HiGN06]. Was bisher fehlt sind Ansätze, die versuchen, auf Basis verhaltenstheoretischer und soziotechnischer Erkenntnisse umfassende methodische Hilfestellung samt Werkzeugunterstützung zu identifizieren.

Zukünftig besteht zur Unterstützung von KSE-Prozessen ein weitergehender Forschungsbedarf unter theoretischen und praktischen Gesichtspunkten. Von praktischer Seite besteht Bedarf zur Gestaltung integrierter Lösungsansätze zur methodischen und softwaretechnischen Unterstützung von KSE-Prozessen unter gleichzeitiger Berücksichtigung soziotechnischer Gesichtspunkte. Die größte Herausforderung besteht in der Ausgestaltung und Einführung integrierter Ansätze für diese interdisziplinäre und praktisch relevante Problemstellung. KSE-Forschung soll den Unternehmen in Zukunft helfen, die für ihre Bedürfnisse richtigen Methoden und/oder Werkzeuge zu finden. Zudem müssen im betrieblichen Kontext häufig spezifische Qualitätsstandards eingehalten werden, wie bspw. die durchgängige Nachvollziehbarkeit des gesamten Entwicklungsprozesses im Rahmen des Capability Maturity Model. Darüber hinaus verlangen auch andere, branchenspezifische Standards, wie die der US-amerikanischen Food and Drug Administration in der Medizintechnik, Prozesstransparenz und Nachverfolgbarkeit von Prozessschritten. Gerade bei starker Verteilung der Ressourcen und einer hohen Intensität der Zusammenarbeit stellt diese Anforderung eine Herausforderung dar.

Aus theoretischer Sicht ist anzumerken, dass die vorgestellten Ansätze und Werkzeuge nur selten theoretische Annahmen und Analysen über das Verhalten der Gruppenmitglieder und die sozialen Interaktionsprozesse zu Grunde legen. Insofern bietet sich an, den Prozess der ingenieurwissenschaftlichen Konstruktion von Softwaresystemen mit verhaltenswissenschaftlichen Theorien zu verknüpfen. Dies sollte eine Kernaufgabe der Wirtschaftsinformatik sein. Als Ausgangspunkt zur theoretischen Betrachtung können die Social Exchange Theory, die Resource Dependency Theory oder die Theory of Planned Behavior herangezogen werden. Auf der Basis von Theorien gilt es zu untersuchen, wie Entwicklungsprozesse, Gruppenverhalten und Werkzeugeinsatz zu gestalten sind, damit die Erstellung von Softwareprodukten effizient und qualitativ erfolgen kann [vgl. hierzu Bair89, 208].

In Abschnitt 3.1 wurden bereits gruppenorientierte Ansätze vorgestellt, die auf sozialwissenschaftlichen Verhaltenstheorien basieren. Insbesondere Gruppendynamik, Gruppenverhalten, soziale Netze und sozialer Austausch (Social Exchange) spielen hierbei eine Rolle. Zur Überwindung der Verteilungsdimensionen der KSE bieten sich mehrere Methoden und Werkzeuge aus dem SE an, wobei diese nur selten unter Berücksichtigung von Verhaltenstheorien konzipiert wurden. Forschungsarbeiten im Bereich *Human-Computer Interaction* haben aber bereits auf der Individualebene gezeigt, dass dies möglich und sinnvoll ist. Aus diesem Grund besteht eine weitere Herausforderung in der interdisziplinären Untersuchung von Gruppenverhalten (Fokus: Mensch), Entwicklungsprozessen (Fokus: Aufgabe) und Werkzeugunterstützung (Fokus: Technik). Diese Untersuchung bewegt sich aufgrund des Zusammenspiels von Mensch, Aufgabe und Technik im Kern der Wirtschaftsinformatik als Wissenschaft. Das Beispiel „Offshore-Entwicklung“ hat gezeigt, dass die KSE-Problematik auch unter praktischen Gesichtspunkten sehr relevant ist.

Literatur

[Altm99] *Altmann, Josef:* Kooperative Softwareentwicklung – Rechnerunterstützte Koordination und Kooperation in Softwareprojekten. Dissertation, Universitätsverlag Rudolf Trauner, Linz, 1999.

[AnZm02] *Andres, Hayward P.; Zmud, Robert W.:* A Contingency Approach to Software Project Coordination. In: Journal of Management Information Systems 18 (2002) 3, S. 41–70.

[AuBS02] *Augustin, L.; Bressler, D.; Smith, G.:* Accelerating Software Development through Collaboration. In: Proceedings of the International Conference on Software Engineering 2002 (ICSE'02). Orlando 2002, S. 559–563.

[Augu91] *August, J. H.:* Joint Application Design: The Group Session Approach to System Design. Yourdon Press, 1991.

[Bair1989] *Bair, J.:* Supporting cooperative work with computers: addressing meetingmania. In: Proceedings of the Thirty-Fourth IEEE Computer Society International Conference (COMPCON Spring '89), 1989, S. 208–217.

[Beck99] *Beck, K.:* Embracing Change with Extreme Programming. In: IEEE Computer, IEEE Computer Society Press 32 (1999) 10, S. 70–77.

Abstract

Approaches to Collaborative Software Development

Software development is becoming more and more complex. Traditionally, the software development process rather corresponds to job-shop manufacturing. Therefore, the ever growing demands for all kinds of software as well as the ongoing globalization require a more efficient development process. Both scientific literature and practical experience postulate a necessary industrialization of software development and design novel forms of specialization and collaboration. Existing approaches to collaborative software development can be classified according to multiple categories, namely organizational, spatial, and temporal distribution, as well as the process direction and the intensity of collaboration. This article aims at giving a comprehensive overview over existing approaches. For this reason, these approaches are compared by means of a common classification scheme. This in turn enables choosing and combining different approaches in terms of methods and tools as the company's situation demands.

Keywords: Collaboration, Software Development, Software Engineering, Industrialization, Software Development Processes, Inter-Organizational Software Development, Cooperation

[BoBo94] *Boehm, Barry W.; Bose, Prasanta:* A Collaborative Spiral Software Process Model Based on Theory W. In: Proceedings of the 3rd International Conference on the Software Process, Applying the Software Process, 1994.

[BoGB01] *Boehm, Barry W.; Gruenbacher, Paul; Briggs, Robert O.:* EasyWinWin: A Groupware-Supported Methodology For Requirements Negotiation. In: 23rd International Conference on Software Engineering, IEEE Computer Society, 2001.

[CaAg01] *Carmel, E.; Agarwal, R.:* Tactical Approaches for Alleviating Distance in Global Software Development. In: IEEE Software 18 (2001) 2, S. 22–29.

[CDHP03] *Cheng, Li-Te; de Souza, Cleidson R.B.; Hupfer, Susanne; Patterson, John; Ross, Steven:* Building Collaboration into IDEs. In: ACM Queue 1 (2003) 9, S. 40–50.

[DeDe04] *DeFranco-Tommarello, Joanna; Deek, Fadi P.:* Collaborative Problem Solving and Groupware for Software Development. In: Information Systems Management 2 (2004), S. 67–80.

[DLPH98] *Dean, D. L.; Lee, J. D.; Pendergast, M. O.; Hickey, A. M.; Nunamaker, J. F.:* Enabling the Effective Involvement of Multiple Users: Methods and Tools for Collaborative Software Development. In: Journal of Management Information Systems 14 (1998).

[DeRi93] *Dewan, P.; Riedl, J.:* Toward Computer-Supported Concurrent Software Engineering. In: IEEE Computer 26 (1993) 1, S. 17–27.

[Dude06] *Duden* – Die Deutsche Rechtschreibung. 24. Aufl., Bibliographisches Institut, Mannheim 2006.

[FeFi01] *Feller, J.; Fitzgerald, B.:* Understanding Open Source Software Development. Addison-Wesley, 2001.

[Fett06] *Fettke, Peter:* State-of-the-Art des State-of-the-Art: Eine Untersuchung der Forschungsmethode „Review" innerhalb der Wirtschaftsinformatik. In: Wirtschaftsinformatik 48 (2006) 4, S. 257–266.

[GaRi02] *Gaeta, Matteo; Ritrovato, Pierluigi:* Generalised Environment for Process Management in Cooperative Software Engineering. In: Proceedings of the 26th Annual International Computer Software and Applications Conference (COMPSAC'02), Oxford, England, 2002, S. 1049–1053.

[Grue00] *Gruenbacher, Paul:* Collaborative Requirements Negotiation with EasyWinWin. In: Proceedings of the 11th International Workshop on Database and Expert Systems Applications (DEXA'00), IEEE, 2000.

[HeMo03] *Herbsleb, James D.; Mockus, Audris:* An Empirical Study of Speed and Communication in Globally-Distributed Software Development. In: IEEE Transactions on Software Engineering 29 (2001) 6, S. 481–494.

[HiGN06] *Hildenbrand, Tobias; Geisser, Michael; Nospers, Markus:* Die Übertragbarkeit der Open Source-Entwicklungsmethodik in die Unternehmenspraxis. In: Softwaretechnik-Trends, 26 (2006) 1, S. 37–42. URL: http://pi.informatik.uni-siegen.de/stt/26_1/03_Technische_Beitraege/OSSD-Transfer-STT.pdf, Abruf am 2006-12-08.

[HiRH06] *Hildenbrand, Tobias; Rothlauf, Franz; Heinzl, Armin:* Ansätze zur kollaborativen Softwareerstellung. Arbeitspapier 06/2006, Lehrstuhl für ABWL und Wirtschaftsinformatik, Universität Mannheim, 2006. URL: http://wifo1.bwl.uni-mannheim.de/fileadmin/files/publications/Arbeitspapier-2006-06-Kollaboration-HiRH06.pdf, Abruf am 2006-12-08.

[HPGF02] *Harrison, D. A.; Price, K. H.; Gavin, J. H.; Florey, A. T.:* Time, Teams, and Task Performance: Changing Effects of Surface- and Deep-Level Diversity on Group Functioning. In: Academy of Management Journal 45 (2002) 5, S. 1029–1045.

[HRDS04] *van der Hoek, A.; Redmiles, D.; Dourish, P.; Sarma, A.; Silva Filho, R.; de Souza, C.:* Continuous Coordination: A New Paradigm for Collaborative Software Engineering Tools. In: Proceedings of the 26th International Conference on Software Engineering (ICSE 2004), S. 29–36.

[Krcm92] *Krcmar, H.:* Computerunterstützung für die Gruppenarbeit – Zum Stand der Computer Supported Cooperative Work Forschung. In: Wirtschaftsinformatik 34 (1992) 4, S. 425–437.

[KrEG88] *Kraut, Robert; Egido, Carmen; Galegher, Jolene:* Patterns of Contact and Communication in Scientific Research Collaboration. In: CSCW '88: Proceedings of the 1988 ACM Conference on Computer-Supported Cooperative Work, 1988, S. 1–12.

[KTCB92] *Kaplan, Simon M.; Tolone, William J.; Carrol, Alan M.; Bogia, Douglas P.; Bignoli, Celsina:* Supporting Collaborative Software Development with ConversationBuilder. In: Proceedings of the Fifth ACM SIGSOFT Symposium on Software Development Environments (SDE 5), 1992, S. 11–20.

[LaBF00] *Layzell, P.; Brereton, O.; French, A.:* Supporting Collaboration in Distributed Software Engineering Teams. In: Proceedings of the Seventh Asia-Pacific Software Engineering Conference (APSEC'00), Singapur, 2000.

[LoRo93] *Lougher, Robert; Rodden, Tom:* Supporting Long-term Collaboration in Software Maintenance. In: Proceedings of the Conference on Organizational Computing Systems (COCS '93), New York, 1993.

[MaSo04] *Martin, David; Sommerville, Ian:* Patterns of Cooperative Interaction: Linking Ethnomethodology and Design. In: ACM Transactions on Computer-Human Interaction 11 (2004) 1, S. 59–89.

[MDTR93] *Mashayekhi, Vahid; Drake, Janet M.; Tsai, Wei-Tek; Riedl, John:* Distributed, Collaborative Software Inspection. In: IEEE Software 10 (1993) 5, S. 66–75.

[MeSz03] *Messerschmitt, David G.; Szyperski, Clemens:* Software Ecosystem: Understanding an Indispensable Technology and Industry. MIT Press, 2003.

[MSHK99] *Maurer, F.; Succi, G.; Holtz, H.; Kötting, B.:* Software Process Support over the Internet. In: Proceedings of the 1999 International Conference on Software Engineering (ICSE'99).

[MRMK00] *Majchrzak, A.; Rice, D. E.; Malhotra, A.; King, N.; Ba, Sulin:* Technology Adaption: The Case of a Computer-Supported Inter-Organizational Virtual Team. In: MIS Quarterly 24 (2000) 4, S. 569–600.

[MuKu93] *Muller, M. J.; Kuhn, S.:* Participatory Design. In: Communications of the ACM, 36 (1993) 6, S. 24–28.

[Robb05] *Robbins, J.:* Adopting Open Source Software Engineering (OSSE) Practices by Adopting OSSE Tools. In: *Feller, J.; Fitzgerald, B.; Hissam, S. A.; Lakhani, K. R. (Hrsg.):* Free/Open Source Processes and Tools, MIT Press, 2005, S. 245–264.

[Sawy04] *Sawyer, Steve:* Software Development Teams. In: Communications of the ACM 47 (2004) 12, S. 95–99.

[ScKr96] *Schwabe, G.; Krcmar, H.:* CSCW-Werkzeuge. In: Wirtschaftsinformatik 38 (1996) 2, S. 209–225.

[SoHR07] *de Souza, C.; Hildenbrand, T.; Redmiles, D.:* Towards Visualization and Analysis of Traceability Relationships in Distributed and Offshore Software Development Projects. In: Proceedings of the First International Conference on Software Engineering Approaches For Offshore and Outsourced Development (SEAFOOD), Zuerich, Switzerland, 2007 (angenommen).

[Somm06] *Sommerville, Ian*: Software Engineering. 8. Aufl., Addison-Wesley, Boston, MA, 2006.