Konfliktmanagement in kooperativen Anwendungen

Stephan Jacobs
RWTH Aachen
Lehrstuhl für Informatik V
Ahornstr. 55
52056 Aachen
jacobs@informatik.rwth-aachen.de

Helge Kahler, Karin Lehner
Universität Bonn
Institut für Informatik III
Römerstr. 164
53117 Bonn
{kahler, karin}@informatik.uni-bonn.de

Andreas Scherer, Wolfgang Wilkes
FernUniversität Hagen
Praktische Informatik I
Feithstr. 140
58084 Hagen
{andreas.scherer, wolfgang.wilkes}@fernuni-hagen.de

1 Einführung

Konflikte sind ein alltägliches Phänomen. Ein Konflikt liegt vor, wenn gewisse Ziele oder Handlungsorientierungen inkompatibel sind. Wird der Konflikt erkannt, so können die betroffenen Parteien versuchen, ihn zu ihrer Zufriedenheit zu lösen, indem sie sich entweder auf eines der formulierten Ziele einigen oder einen Kompromiß finden, in den sich die formulierten Ziele möglichst gut integrieren lassen.

Parallel zum Wandel von Produktionsprozessen wurde der Konfliktbegriff innerhalb der Sozial- und Arbeitswissenschaften neu bewertet. Traditionelle, tayloristische Arbeitsabläufe basieren darauf, daß einzelne Arbeitsschritte reibungslos durchgeführt werden. Konflikte, etwa eine nicht definitionsgemäß durchgeführte Aufgabe, können die gesamte Produktionsmaschinerie lahmlegen und sind deshalb zu vermeiden. Neue Erkenntnisse der Konfliktforschung haben zu einer Neubewertung des Begriffs geführt. Mittlerweile werden Qualitäten wie Eigeninitiative, Verantwortungsbewußtsein oder Flexibilität, die im Taylorismus als den Arbeitsprozeß störend vermieden wurden, wieder als notwendig erkannt (Wohland 1994). Konflikte gelten als Voraussetzung für Erkenntnisgewinn, als Katalysator für Lernprozesse, als notwendige Grundlage für Kreativität. Konflikte weisen auf Probleme hin und führen zu Veränderungen. Konflikte tragen darüber hinaus zur Festigung der Gruppe bei.

Zur konstruktiven Lösung von Konflikten können dabei verschiedene Konfliktlösungskonzepte aus der Informatik beitragen. Je nach der Anwendungssituation kann die Konflikterkennung und -lösung in verschiedener Weise und in unterschiedlichem Ausmaß technisch vollzogen oder unterstützt werden. In manchen Situationen lassen sich Automatismen vorstellen und realisieren, die das Konfliktmanagement nach vorher festgelegten Regeln übernehmen, während in einem anderen Arbeitszusammenhang Informationstechnik eher eine unterstützende Rolle bei der kreativen Nutzung von Konflikten durch die beteiligten Menschen spielt.

Dieses Papier beschäftigt sich mit verschiedenen Ansätzen der informationstechnischen Unterstützung von Problemlösungsprozessen durch Konfliktmanagement, die die Breite der möglichen Einsatzfelder verdeutlichen können. Die hier dargestellten Ansätze unterstützen die

Zusammenarbeit unterschiedlicher Personen und Gruppen insbesondere im Bereich des Designs mittels Informationstechnologie. Dabei werden Konflikte und entsprechende Behandlungsstrategien auf durchaus unterschiedlichen Ebenen diskutiert, um der Komplexität der Thematik Rechnung zu tragen. Nach einer Diskussion des Konfliktbegriffs folgen Ausführungen zum Konfliktmanagement aus den Bereichen:

- Designmanagement
- Nutzung von CSCW-Systemen
- Entscheidungsunterstützung heterogener Gruppen

2 Der Konfliktbegriff in verschiedenen Disziplinen

In diesem Kapitel wird die Bandbreite des Begriffs Konflikt dargestellt. Ausgehend von einem technischen Verständnis von Konflikt führen die weiteren Betrachtungen bis hin zu sozialwissenschaftlichen Fragestellungen. Es wird deutlich, daß die Einbeziehung des Menschen in komplexe Entscheidungsprozesse in vielen Arbeitsbereichen flexiblere Konfliktbehandlungstrategien erfordern.

2.1 Der Konfliktbegriff in der Informatik

Betrachtet man Ziele und Handlungsorientierung in der traditionellen, technisch orientierten Informatik, so hat der Begriff Konflikt eher den Charakter einer Fehlersituation (etwa konkurrierender Zugriff auf zu benutzende Ressourcen). Dies ist letztlich damit zu erklären, daß auf der rein technischen Ebene keine Verhandlungsprotokolle sinnvoll implementierbar sind. Es müssen Strategien entwickelt werden, die in möglichst allen denkbaren Situationen den Programmablauf ermöglichen. In der Regel werden Konfliktsituationen durch entsprechende Vermeidungsstrategien im Vorfeld unterdrückt oder bei Feststellung des Konflikts im nachhinein einseitig aufgelöst. Beispiele für diese Vorgehensweise finden sich etwa im Bereich der Kommunikationssysteme und Datenbanken.

Kommunikationssysteme: Konflikte in der Datenkommunikation treten beim gleichzeitigen Zugriff auf ein Kommunikationsmedium auf. Protokolle regeln den Datenverkehr und helfen Konflikte zu minimieren und, falls Konflikte doch auftreten, sie zu lösen. Im Ethernetprotokoll regelt beispielsweise das CSMA-Protokoll (Carrier Sense Multiple Access) den Zugriff. Um die Konfliktwahrscheinlichkeit zu senken, hört der Sender (Rechner) das Übertragungsmedium ständig ab. Erst wenn das Medium frei ist, beginnt er zu senden. Bei quasi gleichzeitigem Senden - zwei Sender greifen innerhalb eines Intervalls, das kleiner als die Signallaufzeit ist, auf das Medium zu - entsteht natürlich trotzdem ein Zugriffskonflikt, der aber durch das ständige Abhören entdeckt wird. Der Konflikt wird durch erneutes Senden nach einer zufälligen Wartezeit gelöst. Der Einfluß von Zugriffskonflikten führt dazu, daß üblicherweise nur ein Bruchteil der theoretischen Leistung eines Mediums erreicht wird. Aus

diesem Grund sind Protokolle, die Zugriffskonflikte verringern, bei Kommunikationssystemen von großer Bedeutung (Tanenbaum 1981).

Datenbanken: Konflikte in Datenbanken entstehen dann, wenn der gleichzeitige konkurrierende Zugriff auf ein oder mehrere Datenobjekte (z. B. eine Relation) vorgenommen werden soll. Um die Konsistenz der zu verwaltenden Datenmenge zu erhalten, werden Zugriffsstrategien wie z.B. das Zwei-Phasen-Sperrprotokoll eingeführt (Schlageter und Stucky 1983). Diese erzwingen eine Serialisierung des Objektzugriffs. Im klassischen Fall wird damit in letzter Konsequenz der parallele Zugriff auf Objekte im vorhinein vermieden. Klassische DB-Transaktionskonzepte halten die Vision einer Single-User-Benutzung in einem Multi-User-Datenbanksystem aufrecht.

Die Konfliktbehandlung in den genannten Beispielen bauen auf dem gleichen Prinzip auf. In beiden Bereichen wird der Eindruck erweckt, ein Medium stehe exklusiv zur Verfügung. Diese Vorgehensweise hat sich etwa im Bereich der Datenbanken in vielen Anwendungen als sinnvoll erwiesen (z. B. Buchungssystemen). Für andere Anwendungsbereiche haben sich derart restriktive Strategien als unzureichend herausgestellt. In diesen Fällen wurde von Prinzipien wie Isolation, Forderung der globalen Konsistenz der Daten zu jedem Zeitpunkt Abstand genommen. So tragen etwa neuere Ansätze aus dem Bereich des Transaktionsmangements dem Umstand Rechnung, daß kooperative Arbeitsprozesse den gemeinsamen Zugriff und die kooperative Bearbeitung von möglicherweise sehr komplexen Objekten erforderlich macht. Eine exklusive Reservierung eines gerade bearbeiteten Objektes durch einen Benutzer würde die Zusammenarbeit innerhalb eines Teams unmöglich machen. Typisches Anwendungsumfeld hierfür ist die arbeitsteilige Entwicklung komplexer Designlösungen. Wir werden zu neueren Ansätzen zum Thema Designmanagement in Kapitel 3 eingehen.

2.2 Der Konfliktbegriff in den Sozialwissenschaften

Innerhalb anderer Disziplinen, beispielsweise der Arbeitswissenschaften, der Soziologie oder der Psychologie, besteht schon seit langer Zeit ein großes Interesse an der Untersuchung von Konflikten. In diesen Disziplinen hat sich das Verständnis von Konflikten in den letzten 30 Jahren gewandelt. Geprägt durch das destruktive Potential von Konflikten bei sozialen oder nationalen bzw. internationalen Auseinandersetzungen als Gefahr angesehen. Aus diesem Grunde sollten Konflikte vermieden und eliminiert werden. Dieses Verständnis wurde nach und nach durch das Bewußtsein abgelöst, daß ein Konflikt an sich nichts negatives sei. Vielmehr seien konstruktive bzw. destruktive Effekte Ergebnis eines entsprechenden Konfliktmanagements (Thomas 1976).

Konflikte haben positive und produktive Eigenschaften. Konflikte motivieren und stimulieren (Driver und Steufert 1964). Konflikte verhindern Stagnation und weisen auf Probleme hin. Heterogene Gruppen produzieren qualitativ höherwertige Produkte (Hoffmann et al. 1962). Konflikte festigen Gruppen und führen zur Selbsterkenntnis und sind die Wurzel für Verände-

rungen (Pasch 1994). Diese Erkenntnisse werden von Hall (1971) wie folgt zusammengefaßt: *"conflict, effectively managed, is a necessary precondition for creativity*". Noch einen Schritt weiter geht Heidegger (1979). *Vorhandensein* tritt erst beim *Zusammenbruch* auf. Also erst wenn beispielsweise der Computer nicht mehr funktioniert, wenn also ein Konflikt zwischen der Erwartungshaltung und dem tatsächlichen Zustand des Rechners auftritt, wird der Computer bewußt, wird er *zuhanden* (Winograd und Flores 1988). Zusammenbruch (und damit auch Konflikt) ist nach Heidegger also notwendige Voraussetzung eines Erkenntnisprozesses. Eine detaillierte Analyse typischer Merkmale von produktiven und destruktiven Konfliktprozessen, sowie Strategien Konfliktprozesse produktiv zu gestalten wurden von Deutsch (1969) entwickelt (vergleiche auch Deutsch 1976).

Konflikte sind ein alltägliches Phänomen, das in verschiedenen Disziplinen betrachtet wird. Daher gibt es keine allgemein akzeptierte Definition des Konflikts. In dieser Arbeit verwenden wir eine generelle Definition nach Putnam und Poole (1987). Ein Konflikt wird definiert als: *"the interaction of interdependent people who perceive opposition of goals, aims, and values, and who see the other party as potentially interfering with the realization of these goals [...] this definition highlights three general characteristics of conflict: interaction, interdependence, and incompatible goals"*

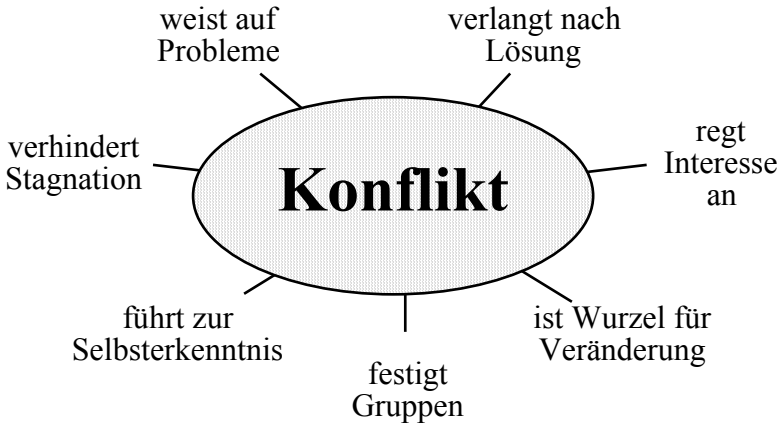

Abb. 1: Positive Eigenschaften des Konflikts nach Pasch (1994)

Diese Definition betont die Aspekte *gegenseitige Beeinflussung*, *wechselseitige Abhängigkeit* sowie *Inkompatibilität von Zielen*. Alle drei Aspekte lassen sich auch innerhalb der Gruppenarbeit, wie z.B. dem kooperativen Entwurf identifizieren.

- Aktionen können nicht isoliert von einander betrachtet werden. Es findet eine *gegenseitige Beeinflussung* der am Entwurf beteiligten Personen statt. Die Beeinflussung unterschiedlicher Aktionen im Concurrent Engineering wird von Reddy et al. (1993) und Cleetus und Reddy (1992) hervorgehoben.
- Die unterschiedlichen Ziele und Sichten auf ein Produkt sind geprägt durch eine *wechselseitige Abhängigkeit.* Das Ändern einer Sicht hat in der Regel auch Folgen für die anderen Sichten. Die gegenseitige Abhängigkeit beispielsweise im Total Quality Management betont Oakland (1989).

- Im kooperativen Entwurf müssen Experten unterschiedlicher Domänen (Entwurf, Produktion, Qualitätsgruppe, Marketing, ...) eine gemeinsame Lösung entwickeln, die optimal bezüglich mehrerer, teilweise *inkompatibler Ziele* sein soll (Breiing und Flemming 1993). Jeder Entwurf stellt daher einen Kompromiß zwischen verschiedenen Zielen dar. Simon (1969 und 1990) folgerte daher, daß ein Entwurf niemals *optimal* sondern immer nur *zufriedenstellend* sein kann.

Das von Thomas (1976) entwickelte Konfliktprozeßmodell baut auf dieser Definition auf (vgl. Abbildung 2). Innerhalb der *Konzeptualisierung* findet eine Auseinandersetzung mit den eigenen Zielen und den Zielen der Gegenpartei statt. Konflikte werden identifiziert, die Ziele in den übergeordneten Kontext relationiert und Lösungsalternativen erarbeitet. Im Zustand *Verhalten* wird sich für eine der Lösungsalternativen entschieden. Dabei spielen nicht nur die eigenen sondern auch die gegnerischen Ziele eine Rolle. Beispielsweise kann es aus taktischen Gründen zur bewußten Aufgabe eigener Ziele kommen. Während der vierten Phase kommt es zur *Interaktion* zwischen den Konfliktparteien. Durch das Verhalten der Gegenpartei kommt es zu einer Rekonzeptualisierung. Die Ziele werden neu verstanden und erneut in Beziehung gesetzt. Neue Handlungsalternativen eröffnen sich. Schließlich einigen sich die Konfliktparteien auf ein gemeinsames Endergebnis.

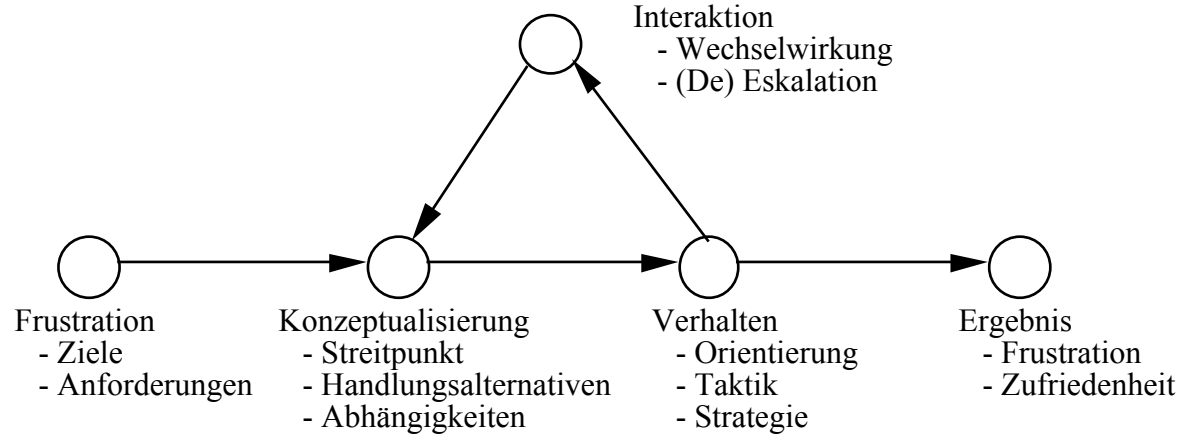

Abb. 2: Konfliktprozeß nach Thomas (1976)

Der oben skizzierte Konfliktprozeß steht und fällt mit der Bereitschaft, sich mit den Zielen der gegnerischen Partei auseinanderzusetzen. Selbst wenn beide Konfliktparteien nicht miteinander kooperieren wollen, ist genau diese Auseinandersetzung entscheidend.

Produktiv gestaltete Konflikte leben von dem Bewußtsein aller, also auch der gegnerischen Ziele. Unterschiedliche Techniken wie beispielsweise *Konfliktdialoge* oder die *Jeder-gewinnt-Methode* (Pasch 1994) sind der Versuch, diesen Prozeß zu systematisieren.

2.3 Eine übergreifende Diskussion des Konfliktbegriffs

Von besonderem Interesse aus der Sicht der Informatik ist es nun, welche Konzepte notwendig sind, um das Konfliktmanagement in komplexen Problemlösungsprozessen zwischen menschlichen Akteuren adäquat zu unterstützen. In Kapitel 2.1 klang bereits an, daß

die traditionellen Lösungstrategien sehr gut für technische Konflikte (z. B. Zugriffskonflikte) geeignet sind, jedoch für kooperative Anwendungen nur von eingeschränktem Nutzen sind. Verlagert man den Schwerpunkt der Computerunterstützung von einer technischen auf eine eher interpersonelle Ebene, so geht damit natürlich eine Veränderung des Verständnis von Konfliktbehandlung einher. Traditionell eher als Ausnahmesituation angesehen, die es möglichst zu vermeiden gilt, entwickelt sich der Konflikt zum natürlichen Bestandteil von Problemlösungsprozessen von Gruppen. Dies ist die Konsequenz aus der dargelegten sozialwissenschaftlichen Sicht des Begriffes.

Heruntergebrochen auf eine technische Ebene bedeutet dies, daß in bestimmten Situationen Konflikte nicht mehr durch restriktive Protokolle behandelt werden können und dürfen. Um dies genauer zu fassen dient der Begriff *Distanz*, der den Abstand innerhalb einer Gruppe beschreibt. Der Begriff Distanz benennt, wie eng zwei Personen miteinander arbeiten. Beispielsweise arbeiten zwei Entwickler, die gemeinsam ein Dokument entwerfen sehr eng zusammen. Umgekehrt ist die Distanz zweier Personen, die zwar im gleichen Büro aber an unterschiedlichen Aufgaben arbeiten, groß. Eine geringe Distanz ist gleichbedeutend mit einer großen Überdeckung der gemeinsamen Bemühung um eine sinnvolle Bearbeitung der Aufgabe. Umgekehrt bedeutet eine große Distanz eine geringe Überdeckung.

Konflikte mit großen Distanzen, beispielsweise Zugriffskonflikte in Datenbanken, können durch strikte, gut formalisierbare Protokolle gelöst werden. Konflikte von Gruppen, die eng zusammenarbeiten, bedürfen umgekehrt eines angepaßten Konfliktmanagements. Am Beispiel des Entwurfs läßt sich die Distanz und die Auswirkung auf das Konfliktmanagement demonstrieren. Zwei Personen, die in unterschiedlichen Projekten arbeiten, greifen auf die gleiche Information, beispielsweise eine Datenbank der lieferbaren Teile, zu. Der Zugriffskonflikt wird durch ein einfaches Transaktionsprotokoll gelöst. Die beiden Entwickler erhalten keine Information über den Konflikt. Sie bleiben voneinander isoliert.

Zwei Entwickler, die im gleichen Projekt aber an unterschiedlichen Dokumenten arbeiten, müssen ihre Arbeit gegenseitig aufeinander abstimmen. Abhängigkeiten zwischen den Dokumenten können beispielsweise durch Notationsmechanismen propagiert werden.

Zwei Entwickler, die mit Hilfe eines Shared-Editors gemeinsam ein Dokument bearbeiten, müssen ihre Tätigkeit genau aufeinander abstimmen. Konflikte, beispielsweise über das Layout des Dokuments, werden durch Kommunikation über die Vor- und Nachteile der einzelnen Alternativen gelöst. Wegen der geringen Distanz ist ein Konfliktmanagement mit Hilfe eines Transaktionsprotokolls nicht möglich. Statt dessen können zusätzliche Kommunikationseinrichtungen oder Werkzeuge zum Management von Zielen eingesetzt werden.

Aus diesen Ausführungen wird deutlich, daß bei Problemlösungsprozessen mit abnehmender Distanz der Schwerpunkt des Konfliktmanagements auf der Unterstützung zwischenmenschli-

cher Protokolle liegt. Ein technisches Konfliktmanagement hat bei abnehmender Distanz eine immer geringere Bedeutung.

2.4 Anforderungen an Konfliktmanagement

2.4.1 Design-Management

Verwaltung von Abhängigkeiten: Bei der Entwicklung komplexer Objekte entstehen eine Reihe von Beschreibungen von Komponenten und Subkomponenten, die in der Regel unterschiedlichen Phasen des Entwurfsprozesses zugeordnet sind. In dem komplexen Beziehungsgeflecht dieser Objekte stellt die computerunterstützte Verwaltung von Abhängigkeiten einen wichtigen Betrag zum Konfliktmanagement dar. Für den Konfliktlösungsprozeß in einem solchen komplexen Beziehungsnetzwerk können wir die folgenden Phasen unterscheiden:

1. *Konfliktauslösung*: Der Konflikt wird durch eine Aktion auf einem oder mehreren Objekten initiiert (z. B. Änderungsoperation)
2. *Konfliktpropagierung und Konflikterkennung:* Die Beziehungen werden als "Meldedrähte" benutzt, über die andere Objekte von dieser Aktion informiert werden. Die Art der Beziehung bestimmt dabei, ob und welche Folgeänderungen auf diesen Objekten ausgeführt werden.
3. *Konfliktlösung:* Ist der Konflikt dokumentiert worden, so können zu geeigneter Zeit bestimmte Mechanismen (Verhandlungsprotokolle, group commit) eingesetzt werden, um den Konflikt aufzulösen, was z.B. heißen kann, die alternativen Versionen zu einer Version zusammengefaßt werden.

Die Ausführungen in Kapitel 3 beschäftigen sich mit dem Themenkomplex Design-Management unter besonderer Berücksichtigung der Verwaltung aktiver Abhängigkeiten für das Konfliktmanagement.

2.4.2 Konfliktmanagement in CSCW-Systemen

Teamverträgliche Nutzung: Neben fachlichen und rein technischen Konflikten bei der Arbeit in Konstruktionsteams müssen auch Konflikte berücksichtigt werden, die durch die Nutzung der unterstützenden Software erst manifest werden und die aus der Spannung von Offenheit und Abgeschlossenheit bei der Kooperation resultieren. Ebenso wie Offenheit für einen Austausch von Meinungen und auch Arbeitsobjekten für jede Kooperation unabdingbar ist, benötigen einzelne Mitglieder des Teams auch Abgeschlossenheit, um störungsfrei arbeiten zu können. Für nicht computerunterstütztes Arbeiten wird z. B. der Wunsch, nicht gestört zu werden, ausgedrückt durch ein Schließen der Tür oder die Umleitung des Telefons. Zudem existieren Konventionen in einer Arbeitsgruppe für "materielle Abgeschlossenheit", die es beispielsweise erlauben, *auf* dem Schreibtisch eines Kollegen nach einem Dokument zu suchen, nicht jedoch *im* Schreibtisch. Dies alles dient dazu, die für die Arbeit auch im Team

nötige Privatheit des Einzelnen zu gewährleisten. Hier ist es für kooperationsunterstützende Systeme unabdingbar, sich dieses Spannungsfeldes von Offenheit und Abgeschlossenheit anzunehmen und Mechanismen zur Verfügung zu stellen, die die aus der Spannung entstehenden Konflikte regeln helfen.

2.4.3 Heterogene Gruppenentscheidungen

Intuitive Visualisierung: Der zentrale Teil des Konfliktprozesses besteht aus dem Zyklus Konzeptualisierung, Verhalten, Rekonzeptualisierung. In diesem Zyklus werden die Positionen der einzelnen Konfliktparteien einander gegenübergestellt, wird ein Bewußtsein für die Ziele der anderen Konfliktpartei geschaffen. Die Bedeutung des eigenen Handelns und der eigenen Ziele für die gegnerische Partei wird geklärt.

Im Entwurf sind Entwurfsdokumente die "materialisierten" Positionen eines Konflikts. Konfliktmanagement muß daher auf diesen Dokumenten aufbauen, die Dokumente miteinander verbinden und die verbundenen Dokumente als Grundlage informeller Kommunikation benutzen. Eine formale Darstellung des Konflikts basierend auf den Dokumenten ist daher notwendig.

Gegenseitiges Verständnis läßt sich allerdings nicht (allein) mit Hilfe formaler Mechanismen erzielen - insbesondere dann, wenn die Formalismen wechselseitig unbekannt sind. Diskussion, Erklärung und Intuition, also informelle Techniken, sind entscheidender Bestandteil von Konfliktmanagement. Die am Konflikt beteiligten Parteien kennen zwar ihre eigene Position sind aber nicht mit den Zielen und der Sprache der Gegenpartei vertraut. Daher ist die in den Entwurfsdokumenten verwendete Sprache als Darstellung der Konfliktpositionen nicht geeignet.

Konfliktmanagement beruht sowohl auf formalen als auch auf informellen Elementen. Dabei ist das Wechselspiel zwischen formaler Interrelierung der konfligierenden Dokumente und informeller Diskussion das zentrale Problem des Konfliktmanagements. Die Brücke zwischen diesen beiden Polen muß durch eine intuitiv verständliche Visualisierung geschlagen werden, die einerseits die Konfliktpositionen formal repräsentiert und so einen Anschluß zu den Entwurfsdokumenten schafft und andererseits Grundlage für informelle Diskussion und gegenseitiges Verständnis ist.

2.5 Verwandte Arbeiten

Unter dem Stichwort Konfliktmanagement sind in den letzten zwei Jahren mehrere Ansätze und Systeme vorgestellt worden. Diese werden im folgenden charakterisiert. Anschließend wird der Einfluß des Forschungsgebietes computerunterstütztes Arbeiten (Computer Supported Cooperative Work, CSCW) auf Konfliktmanagement untersucht.

Konfliktmanagement: In den letzten Jahren sind unterschiedliche Ansätze und Systeme entwickelt worden, die in Workshops oder Sonderausgaben von Journalen unter dem Oberbegriff Konfliktmanagement vorgestellt wurden (vgl. beispielsweise AAAI 94, CERA 94).

Die Ansätze ermöglichen das verteilte Entwickeln von Teilentwürfen, die in einer gemeinsamen formalen Sprache dargestellt werden. Durch die Formalisierung können bei der Integration der einzelnen Teile Konflikte erkannt und Lösungsmöglichkeiten entwickelt werden.

Das Design Collaboration Support System (DCSS) basiert auf einem Blackboard-System das mit Hilfe eines Netzwerkes und Assistenten verschiedene Entwurfswerkzeuge verbindet (Klein 1993). Die Entwurfsdokumente werden in Form von Entwurfsentscheidungen und -begründungen (design rationales) miteinander verbunden. Konflikte werden als Verletzungen dieser Struktur erkannt und dem Entwickler mitgeteilt. DCSS unterstützt den Entwickler, indem es die konfligierenden Strukturen darstellt.

Einen noch dezentraleren Ansatz stellen ViewPoints dar (Finkelstein et al. 1992). ViewPoints sind unabhängige Sichten auf den Entwurf. Formal besteht ein ViewPoint aus einer Notation, der Domäne, aus der das zu entwickelnde System stammt, einer aktuellen Spezifikation in der vorgegebenen Notation, eine Beschreibung des Arbeitsplans sowie einem Logbuch, das die bisherige Entwurfsgeschichte des ViewPoints beschreibt. Da jeder ViewPoint seinen eigene Notation hat, gibt es keine gemeinsame Repräsentation, mit der Konflikte beschrieben werden könnten. Konsistenzregeln innerhalb eines ViewPoints können mit Hilfe einer formalen Sprache zur Beschreibung von Beziehungen ausgedrückt werden (Nuseibeh et al. 1994). Konsistenzen zwischen ViewPoints können nur indirekt ausgedrückt werden, indem zwei oder mehr ViewPoints in einem gemeinsamen ViewPoint vereinigt werden.

Kusiak und Wang (1994) beschreiben den Entwurf als das Festlegen von Entwurfsentscheidungen. Diese können mit Entscheidungsvariablen beschrieben werden. Beim verteilten Entwurf gibt es neben unabhängigen Variablen sogenannte *interaction variables*, die von mehreren Entwicklern benötigt werden. Ein Konflikt entsteht, wenn eine Interaktionsvariable mit unterschiedlichen Werten belegt wird. Kusiak und Wang entwickeln ein mathematisches Verhandlungsmodell, daß mit Hilfe von Nutzenfunktionen die optimalen Werte für die einzelnen Variablen ermittelt.

In den meisten Ansätzen zum Konfliktmanagement wird nicht wie in Easterbrook et al. (1994) explizit zwischen Verletzung von Inkonsistenzen und Konflikt unterschieden. Inkonsistenzen sind Verletzungen von formalen Regeln, die beispielsweise zwischen unterschiedlichen Dokumenten definiert werden. Konflikte sind Inkompatibilitäten zwischen den Zielen der am Entwurf beteiligten Parteien. Konflikte werden nicht notwendigerweise durch eine Inkonsistenz entdeckt. In diesem Sinne unterstützen die unter dem Namen Konfliktmanagement dargestellten Systeme nicht das Management von Konflikten, sondern das Management von Konsistenz in verteilten Entwicklungsumgebungen.

Die Unterscheidung zwischen Konflikt und Inkonsistenz versuchen einige Ansätze zu umgehen, indem sie ein Modell definieren, das die Beziehungen zwischen Zielen, Entscheidungen und Entwurfsdokumenten formalisiert (vgl. z.B. Klein und Lu 1990, Ramesh und Sengupta

1994). Die Verletzung dieses Modells stellt dann einen formalisierten Konflikt dar. Die durch diese Struktur gewonnene zusätzliche Sicht auf die einzelnen Entwurfsdokumente kann zwar als Unterstützung im Konfliktmanagement angewendet werden. Allerdings kann aus dem Fehlen von Inkonsistenzen nicht auf die Abwesenheit von Konflikten geschlossen werden, da immer nur ein kleiner und vereinfachter Teil der tatsächlichen Ziele und Beziehungen dargestellt wird.

Im Gegensatz zu den in Kapitel 1 gestellten Anforderungen konzentrieren sich die gerade genannten Ansätze nur auf die formale Seite des Konflikts. Die Notwendigkeit informeller Kommunikation sowie das Wechselspiel zwischen formalen Modellen und informeller Kommunikation bleibt unberücksichtigt.

CSCW: Unter dem Schlagworten computergestütze kooperative Arbeit (Computer Supported Cooperative Work, CSCW) und Groupware hat sich in den letzten zehn Jahren ein neues Forschungsgebiet eröffnet, das sich mit der Rechnerunterstützung für Teamarbeit beschäftigt. Obwohl sich bisher keine einheitliche Definition von Kooperation durchgesetzt hat, gibt es verschiedene anerkannte Klassifizierungen, um die unterschiedlichen Ansätze zusammenzufassen.

Die bekannteste, die Raum-Zeit-Matrix (Ellis et al. 1991), unterscheidet in der Zeitdimension zwischen synchroner und asynchroner, und in der Raumdimension zwischen lokaler und räumlich verteilter Arbeit. Typische Vertreter von synchroner Arbeit sind rechnergestützte Sitzungsräume (Electronic Meeting Rooms, EMS) für lokale Anwendungen und Video- oder Computerkonferenzen für räumlich verteilte Anwendungen. Asynchrone Kooperation wird mit Hilfe von elektronischen Anschlagbrettern (Electronic Meeting Rooms, EMS) oder Email unterstützt. Rodden hat in dieser Matrix vier Klassen unterschiedlicher Anwendungen eingeordnet: Systeme für Nachrichtenaustausch, Computerkonferenzen, rechnergestützte Sitzungsräume und CoAutoren Systeme (Rodden 1991).

Trotz großer technischer Fortschritte (vgl. z.B. Okada et al. 1994) ist die Einführung von CSCW Systemen bislang hinter den formulierten Erwartungen zurückgeblieben (vgl. z.B. Oberquelle 1991, Grudin 1994). Unter anderem verhindert die isolierte Betrachtung von Kooperation und Anwendung den Durchbruch rechnergestützter Teamarbeit.

Im CSCW entwickelte Konzepte lassen sich nicht auf Konfliktmanagement im Entwurf übertragen. Augenblicklich wird zwar der Konflikt als Bestandteil von Kooperation identifiziert, allerdings gibt es bislang wenige Konzepte, die sich mit Konfliktmanagement beschäftigen (Easterbrook 1993). Für den Umgang mit Konflikten, die bei der Aktivierung einzelner Funktionen in einem Vorgangsbearbeitungssystem entstehen können, liegen empirische Ergebnisse einer szenariobasierten Feldstudie vor, die ein Vorgehen wie das in Kapitel 4 vorgeschlagene nahelegen (vgl. Rohde et al. im Druck).

Projektmanagement: Spätestens seit den programmierttechnischen Großprojekten wird in der Informatik um die Frage, wie ein Team von Softwareentwicklern zu organisieren sei, kon-

trovers diskutiert (Brooks 1975). Kernproblem der Softwareteams sind suboptimale Kommunikation und mangelnde Kontextvermittlung.

Dieses Problem wird durch Bildung unterschiedlicher Teamstrukturen angegangen. Dabei haben sich zwei gegensätzlichen Philosophien herauskristallisiert (Mantei 1981, Constantine 1993). Ziel autokratischer Organisationsstrukturen ist eine Verringerung und Steuerung der Kommunikation. Im Zentrum des Chief-Programmer-Teams, einem der bekanntesten autokratischen Ansätze, sitzt der sogenannte Chef-Programmierer. Er entwickelt einen Top-Down Architektur des Gesamtsystems und verteilt Programmieraufgaben an die ihm untergeordneten Programmierer. Eine Kommunikation der Programmierer untereinander ist innerhalb dieses Ansatzes nicht vorgesehen. Tatsächlich entwickeln sich aber gerade in größeren Projekten informelle Kommunikationsnetzwerke, über die auf direktem Wege Informationen ausgetauscht werden.

Im Gegensatz zu autokratischen stehen demokratische Organisationsprinzipien. In demokratischen Gruppen wird Software-Entwicklung nicht als ein rein technischer, sondern vor allem als ein sozialer Prozeß angesehen, in dem die Kommunikation zwischen den Projektmitgliedern eine Hauptrolle spielt . Ziel dieses Ansatzes ist es nicht, Kommunikation zu reduzieren sondern Kommunikation angstfrei zu gestalten, Kommunikation zur Identifikation mit dem Gruppenziel zu benutzen. Auch in demokratischen Gruppen gibt es Regeln für die Kommunikation zwischen den Mitarbeitern. Allerdings ist es nicht Ziel dieser Regeln, die Kommunikation einzuschränken, sondern durch Einsatz bestimmter Methoden die Kommunikation beispielsweise ziel- oder konfliktorientiert zu gestalten. Der Nachteil demokratischer Strukturen liegt darin, daß die Teams zuviel Eigendynamik entwickeln, riskante Entscheidungen treffen und durch das Management schwieriger zu kontrollieren sind (Mantei 1981).

Konflikt- und Projektmanagement ergänzen sich gegenseitig. Dies bedeutet, daß nur ein abgestimmtes Wechselspiel zwischen Projekt- und Konfliktmanagement zum Erfolg führt. In diesem Sinne erfordert erfolgreiches Konfliktmanagement ein dazu passendes Projektmanagement. Schmidt (1994) bezieht Organisationen im Konfliktmanagement mit ein und konstatiert, daß der Konfliktbegriff für Organisationen neu überdacht werden muß, da organisationale Grenzen gerade für post-tayloristische Arbeitsformen unscharf sind.

3 Unterstützung des Konflikt-Management in technischen Bereichen durch Versions- und Konfigurations-Mechanismen

3.1 Konflikte und Produkt-Lebenszyklus

Im Laufe des Lebenszyklus eines Produktes treten normalerweise viele Konflikte auf, deren Lösung erforderlich ist, um das Produkt fertigstellen und am Markt anbieten zu können. Man kann zwei wesentliche Konfliktbereiche unterscheiden:

1. Ein Produkt durchläuft in seinem Lebenszyklus verschiedene Phasen. Betrachten wir z.B. den Produktentwicklungsprozeß im Automobilbau, etwa die Getriebeentwicklung, so können folgende wesentliche Phasen identifiziert werden:

- **Konstruktion**: In diese Phase fällt der Entwurf eines Getriebes, d.h. Festlegen der allgemeinen Designspezifikationen und Anfertigung von Detail- und Fertigungszeichungen
- **Produktionsplanung** (Fertigungsvorbereitung, Logistik): In Abstimmung mit der Fertigung findet eine Überprüfung der Verfügbarkeit der notwendigen Maschinen bzw. des erforderlichen Materials statt. Materialflußpläne und produktionstechnische Abläufe werden analysiert und optimiert.
- **Fertigung**: Hier wird der Produktionsprozeß im Detail festgelegt. Zu den Aufgaben des Fertigungsingenieurs gehört etwa die Auswahl der Maschinen, sowie die Feststellung der Reihenfolge der Bearbeitungs- und Zusammenbauschritte.
- **Kostencontrolling**: Der gesamte Konstruktions- und Planungsprozeß wird durch entsprechende betriebswirtschaftliche Untersuchungen begleitet, um den unternehmerischen Erfolg zu sichern.

Die unterschiedlichen Aspekte des Produktes, die in den einzelnen Phasen betrachtet werden, führen häufig zu Konflikten. Beispielsweise werden fertigungstechnische Schwierigkeiten bei der Konstruktion üblicherweise noch nicht berücksichtigt. Häufig sind auch die Ziele der Fertigung - möglichst einfach und preiswert zu produzieren - und die Ziele der Konstruktion - beispielsweise eine große Funktionalität zu implementieren - in vielen Fällen inkompatibel. Trotzdem ist die Fertigung von den Konstruktionszeichnungen abhängig, die Distanz zwischen den Konfliktparteien ist also relativ gering.

2. Auch innerhalb der einzelnen Phasen arbeiten in der Regel mehrere Personen an der Konstruktion oder Fertigung des Produktes. Diese Personen müssen ihre Aktivitäten abstimmen, um nicht durch Mißverständnisse oder nicht erkannte Inkonsistenzen Fehler entstehen zu lassen.

Der traditionelle Entwurf bietet keine Möglichkeit, um die Konflikte, etwa zwischen Fertigung und Konstruktion, zu lösen. Moderne Entwurfsparadigmen wie Concurrent Engineering betonen dagegen den wechselseitigen Einfluß der unterschiedlichen Phasen und versuchen diese Phasen bewußt interagieren zu lassen. Wir wollen im folgenden untersuchen, wie diese Wechselwirkungen durch technische Mechanismen unterstützt werden können. Dabei gehen wir davon aus, daß sich der Produktlebenszyklus durch die entstehenden Daten/Dokumente und ihre Bearbeitung widerspiegelt. Wechselwirkungen ergeben sich dann, wenn Teile eines Dokumentes (etwa eine Konstruktionszeichnung) mit Teilen eines anderen Dokumentes (etwa einer Fertigungsplanung) verknüpft sind und Änderungen an diesen Teilen vorgenommen werden. Dies wird in aller Regel auch die entsprechenden Teile des anderen Dokumentes betreffen. Insbesondere sind natürlich auch die Anforderungen, die sich aus den einzelnen Phasen ergeben, explizit in speziellen Dokumenten darzustellen und mit den

entsprechenden Produktdaten in Bezug zu setzen, so daß sich Änderungen von Anforderungen bzw. die Feststellbarkeit der Nichterfüllbarkeit von Anforderungen in der Datenhaltung niederschlägt.

Aufgrund der komplexen Beziehungsstrukturen der Produktdaten ist das erste Ziel die Erkennung und Dokumentation von (potentiellen) Konflikten. Dabei bedienen wir uns einer bekannten Technologie: Wir benutzen Versionsmechanismen, um durch die Erzeugung von alternativen Versionen Konflikte zu dokumentieren. Der anschließende Problem-Lösungsprozeß wird durch das Zusammenführen ("Merging") der alternativen Versionen abgeschlossen. Die Erkennung von Konflikten, d.h. im wesentlichen von Operationen auf Daten, die "nicht kompatibel" zu anderen, in Beziehung stehenden Daten sind, wird dabei durch Techniken aus der Konfigurations-Verwaltung vorgenommen und führt zu der Erzeugung der alternativen Versionen.

3.2 Versions-Verwaltung in Konstruktionsanwendungen

Die Verwaltung von Versionen stellt ein wesentliches Element für die Unterstützung von Konstruktions- und Produktionsplanungsanwendungen dar. Versionen werden aus unterschiedlichen Gründen eingesetzt, u.a. aus den folgenden:

- kurzfristige Dokumentation der letzten Arbeitsschritte
- langfristige Dokumentation der wichtigsten Entscheidungen
- Dokumentation von alternativen Entwicklungen (d.h. verschiedene Lösungsansätze für dieselbe Spezifikation)
- Verwaltung von unterschiedlichen Produkten die durch Variation von einigen Parametern oder durch kleine Konstruktionsänderungen entstehen (z.B. Software-Produkt für unterschiedliche Betriebssysteme, Motor-Baugruppen für unterschiedliche Leistungsanforderungen).

Entsprechend den unterschiedlichen Einsatzgebieten gibt eine Vielzahl von Begriffen, die Versionen für bestimmte Zwecke bezeichnen: Revisionen, Varianten, Alternativen, etc.

3.3 Alternative Versionen zur Dokumentation von Konflikten

Im Zusammenhang mit Konfliktmanagement sind insbesondere Versionen zur Darstellung von parallelen Entwicklungen interessant, die Entwicklungsalternativen darstellen. Sie dokumentieren einen Konflikt: Z.B. können sie mehrere Möglichkeiten zur Realisierung einer Spezifikation darstellen, sie können aber auch mehrere Anforderungen (alternative Spezifikationen) dokumentieren, die von verschiedenen Seiten an das spezifische Produkt gestellt werden. Im Zuge der weiteren Produktentwicklung muß aus diesen Alternativen eine von allen Seiten akzeptierte Lösung ermittelt werden, d.h. die Auflösung des Konfliktes dokumentiert sich im Entstehen einer gemeinsamen Nachfolgeversion aller Alternativen (*version merging*).

Die heute verfügbaren Versionsverwaltungssysteme bieten durchgängig Möglichkeiten an, um alternative Versionen darzustellen und damit derartige Konflikte zu dokumentieren (z.B. Katz 1990, Cellary und Jamier 1990, Tichy 1985). Anschließend allerdings lassen sie den Benutzer in der Regel allein, er erhält keine Unterstützung bei der Auflösung des Konfliktes. Es gibt nur einige wenige Systeme, die zumindest eine rudimentäre Konfliktlösungsstrategie beinhalten. Ein Beispiel dafür stellt das von SUN entwickelte System NSE (Network Software Environment) dar (vgl. Miller 1989, Adams et. al. 1989):

> NSE verwendet ein modifiziertes optimistisches Verfahren zur Synchronisation, d.h. zunächst können mehrere Benutzer ein Objekt verändern, wobei vom System aus alternative Versionen für die einzelnen Benutzer erzeugt werden. Beim Zurückschreiben kontrolliert das System, ob die ursprüngliche Version in der Zwischenzeit verändert wurde. Ist dies der Fall, muß der Benutzer, der zurückschreiben will, seine Änderungen mit den zuvor erfolgten Änderungen abgleichen, d.h. er darf die in der Zwischenzeit geänderte Version nicht einfach überschreiben, sondern muß seine Änderungen in die existierende Version "hineinmischen".

Leider findet eine weitergehende Unterstützung der Konfliktlösung in Versionsverwaltungssystemen nicht statt. Hier wäre eine Kombination von fortgeschrittenen Versionsverwaltungssystemen mit Groupware-Techniken sehr wünschenswert. Einsetzbare Techniken sind z.B.:

- Mechanismen zur frühzeitigen Information beim Checkout, so daß bereits zu diesem Zeitpunkt Kontaktaufnahme der beiden Konfliktparteien möglich ist, oder durch
- Verhandlungsprotokolle, die den Prozeß der Konfliktlösung strukturieren und beispielsweise durch ein Group-Commit abschließen.

3.4 Konflikte in Konfigurationen von Versionen

Die Versionsverwaltung wird vor allem dadurch kompliziert, daß es nicht reicht, einzelne, isolierte Objekte zu betrachten. Vielmehr müssen die komplexen Beziehungsstrukturen von Objekten auch von der Versionsverwaltung berücksichtigt werden: Objekte sind zusammengesetzt aus anderen Objekten, Konstruktionen können aus verschieden Sichten heraus dargestellt werden (Geometrie, Stückliste, kinematische Zusammenhänge, Produktions-Planung, etc.; diese Sichten entsprechen häufig bestimmten Phasen des Produktlebenszyklus), und sie können auf unterschiedlichen Abstraktions-Ebenen dargestellt werden (Spezifikations-Ebene, Realisierungs-Ebene). Somit sind zur vollständigen Dokumentation eines Produktes eine Vielzahl von Objekten (oder Dokumenten) erforderlich, die stark voneinander abhängen, und die auf vielen Ebenen versioniert sein können. Neben den einzelnen Versionen müssen also auch die Konfigurationen von Versionen verwaltet werden. Eine wesentliche Aufgabe der Konfigurationsverwaltung besteht darin, die Konsistenz der Beziehungsstrukturen zu überwachen, d.h., zu kontrollieren, ob die "richtigen" Versionen "richtig" in Beziehung gesetzt worden sind.

Gerade im Concurrent Engineering werden die Versionen von verschiedenen Stellen aus parallel entwickelt und bearbeitet. Aufgrund ihrer Abhängigkeiten kommt es dabei natürlicherweise zu einer Vielzahl von Konflikten, ausgelöst durch die unterschiedlichen Anforderungen und Sichtweisen der einzelnen Bereiche. Diese Konflikte führen zu Verletzungen von Constraints, die sich über die Beziehungen auf andere Objekte/Versionen fortpflanzen können.

Beispiel:

Zur Spezifikation einer Komponente gehört eine genaue Beschreibung ihrer Schnittstelle, in der die Konstruktionselemente definiert sind, die zur Benutzung der Komponente bekannt sein müssen. Dadurch kann die Komponente bereits in komplexeren Bauteilen benutzt werden, bevor ihre Konstruktion abgeschlossen ist. Benutzung und Konstruktion finden also gleichzeitig statt. Zwei Arten von Konflikten können dabei entstehen:

- Bei der Implementierung kann sich herausstellen, daß Elemente der Spezifikation widersprüchlich sind (z.B. Größe und Leistungsanforderung), so daß die Spezifikation geändert werden muß.
- Bei der Benutzung kann sich herausstellen, daß zusätzliche oder veränderte Anforderungen notwendig sind.

Wird nun einseitig die Spezifikation geändert (z.B. vom Benutzer der Komponente), so entsteht ein Konflikt mit der anderen Seite (der Konstruktion der Komponente). Falls die Komponente in mehreren komplexeren Bauteilen benutzt wird, entsteht auch ein Konflikt mit diesen anderen Benutzern. Somit muß der Wunsch nach Änderung der Spezifikation dokumentiert werden und außerdem muß ein Konfliktlösungsprozeß angestoßen werden, der zur (gemeinschaftlichen) Definition einer neuen Spezifikation führt.

Für Konflikte und den Konfliktlösungsprozeß in einem solchen komplexen Beziehungsnetzwerk können wir die folgenden Phasen unterscheiden:

1. *Konfliktauslösung*: Der Konflikt wird durch eine Aktion auf einem Objekt ausgelöst. Meistens handelt es sich dabei um eine Änderungsoperation, etwa die Korrektur eines Fehlers bei der Konstruktion einer Komponente oder die Veränderung einer Spezifikation.
2. *Konfliktpropagierung:* Die Beziehungen werden als "Meldedrähte" benutzt, über die andere Objekte von dieser Aktion informiert werden. Die Art der Beziehung bestimmt dabei, ob und welche Folgeänderungen auf diesen Objekten ausgeführt werden. Diese Folgeänderungen können z.B. den Konflikt automatisch lösen, etwa beim automatischen Generieren einer Stückliste aus geänderten Entwürfen. Sie können aber auch lediglich den Konflikt dokumentieren, z.B. durch die Erzeugung alternativer Versionen oder durch die Überführung von Versionen in bestimmte Zustände (die vor allem den erlaubten Operationsumfang auf der Version bestimmen). Weitere Aktionen können die direkte Benachrichtigung des Bearbeiters des Objektes sein.

3. *Konfliktlösung:* Ist der Konflikt dokumentiert worden, so können zu geeigneter Zeit die oben beschriebenen Mechanismen (Verhandlungsprotokolle, group commit) eingesetzt werden, um den Konflikt aufzulösen, was schließlich zum "Merging" die alternativen Versionen zu einer Version führt.

3.5 Möglichkeiten existierender Konfigurations-Verwaltungssysteme

Die vorhandenen Konfigurations-Verwaltungssysteme bieten erste Ansätze zur Erkennung von derartigen Konflikten und - über das automatische Setzen von bestimmten Konsistenz-Zuständen für in Beziehung stehende Objekten - zur Weiterleitung des Wissens über den Konflikt an relevante Objekte. Weiterhin werden Mechanismen vorgeschlagen, die auch dazu dienen können, die Konfliktlösung zu unterstützen, wie z.B. automatische Folgeänderungen (in der Regel durch das Aufrufen von bestimmten Tools/Methoden, etwa in Adele (Estublier und Casallas 1994) oder eine automatische Versionierung auf verschiedenen Ebenen der Konfiguration, etwa in Katz (1990). Jedoch sind die Strategien für die Behandlung von Änderungen häufig fest verdrahtet, und es fehlen Möglichkeiten, um benutzerdefinierte Strategien und Konfliktlösungs-Protokolle zu integrieren.

3.6 Aktive Beziehung zur dynamischen Definition der Abhängigkeiten von Objekten

Einen Ansatz zur Unterstützung von anwendungsspezifischen Konfliktlösungs-Strategien in derartigen komplexen Beziehungsstrukturen wie der Konfigurations-Verwaltung stellen *aktive Beziehungen* dar (Kemper et. al. 1995). Die Semantik einer aktiven Beziehung wird beschrieben durch die Art, wie Aktivitäten auf einer Seite der Beziehung zur anderen Seite weitergeleitet werden. Dieses Weiterleitungsverhalten wird definiert in Form von *Beziehungsregeln,* speziellen ECA Regeln (Event-Condition-Action-Rules, siehe etwa Buchman 1994). Eine Beziehungsregel ist sensitiv auf ein bestimmtes Ereignis auf einem der Objekte, die durch die Beziehung verbunden sind, z.B. der Aufruf einer bestimmten Methode. Tritt dieses Ereignis auf, so werden, gesteuert durch den Bedingungs- und Aktionsteil der Regel, Methoden der anderen Objekte aufgerufen.

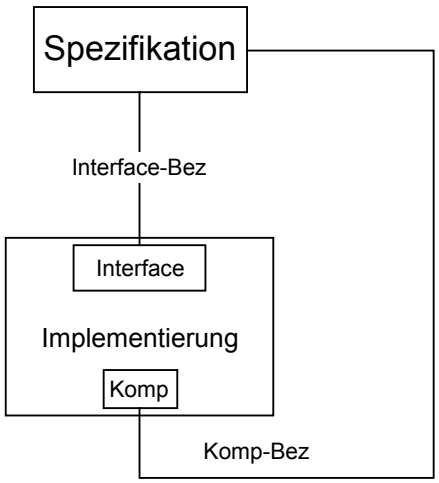

Abb. 3: Modellierung von Spezifikation, Implementierung und Benutzung von Design-Objekten

In Abb. 3 ist die Situation dargestellt, die bereits im Beispiel beschrieben wurde. Das Entity *Implementierung* enthält mehrere Subentities, zum einen die (ererbte) Spezifikation des Interfaces, zum anderen (evtl. mehrere) Komponenten, von denen genau die Interface-Spezifikation benutzt wird (zum Beispiel zum Anschluß an andere Teile der Konstruktion). Die *Interface-Beziehung* und die *Komponenten-Beziehung* müssen die Konsistenz und Übereinstimmung zwischen den Interface-Daten innerhalb des Implementierungs-Objektes und den entsprechenden Spezifikationen überwachen. Bei Konflikten müssen geeignete Maßnahmen zur Weiterleitung der Information über das Auftreten eines Konfliktes und zum Lösen des Konfliktes angestoßen werden.

Zwei Fälle sind zu unterscheiden:

1. Bei der Konstruktion wird festgestellt, daß das Konstruktionsziel (also die ererbte Spezifikation des Interfaces) so nicht realisiert werden kann.
2. Bei der Konstruktion wird festgestellt, daß die Spezifikation einer Komponente ungeeignet ist und daher die Komponente (d.h. ihre Spezifikation und ihre Realisierung) geändert werden muß.

Fall 1 wird als harter Konflikt behandelt werden, da die spezifizierte Komponente so nicht fertiggestellt werden kann. Daher sollen alle Benutzer darüber informiert werden, indem ihr entsprechendes Subobjekt *Komp* einen anderen Zustand annimmt (*Spezifikation-in-Änderung*). Je nach dem speziellem Objekttyp des umgebenden Objektes *Implementierung* wird dadurch bestimmt, ob hier noch weitergearbeitet werden darf oder ob direkt ein Konfliktlösungs-Mechanismus ausgelöst wird.

Fall 2 wird als weicherer Konflikt angesehen, da möglicherweise nur in diesem einen Fall die Nutzung der Komponente nicht möglich ist. Daher wird zunächst nur eine alternative Spezifikationsversion erzeugt, und es wird nur die Implementierung der Komponente durch Überführung in den Zustand *Änderungsbedarf* informiert.

Aktive Beziehungen sind also ein Instrument, das vor allem für die ersten beiden Phasen des Konfliktlösungsprozesses in einem derartigen Netzwerk geeignet ist: Es unterstützt die Konflikterkennung und die Konflikt-Propagierung. Zur Dokumentation eines Konfliktes werden alternative Versionen erzeugt, die die verschiedenen Requirements darstellen. über die aktive Beziehungen können Tools angestoßen werden, die bei der Auflösung des Konfliktes mitwirken (z.B. ein Verhandlungs- oder Abstimmungs-Tool, das das gemeinsame Mischen alternativer Versionen unterstützt).

4 Konflikte durch die Nutzung von Groupwarefunktionalität

Seit einiger Zeit werden verstärkt Konzepte der Gruppenarbeit diskutiert, von denen man sich synergetische Effekte für die Nutzung in der Organisation vorhandener Kenntnisse und Fähigkeiten, eine größere Kundennähe oder eine "Verschlankung" der Organisation erhofft. Dabei spielt auch der Einsatz von Software zur Unterstützung kooperativer Arbeit eine wichtige Rolle. Von dieser "Groupware" wird erwartet, daß sie die Zusammenarbeit z. B. der Personen in einer Arbeitsgruppe angemessen unterstützt.

Das Design von Groupware gestaltet sich jedoch auch dadurch schwierig, daß die Arbeitsweise von Gruppen ungleich größere Designanforderungen an das System stellt als einzelplatzorientierte Software. Insbesondere die mannigfaltigen Formen, in denen Gruppen kooperieren können, und die Berücksichtigung unterschiedlicher Personen in einer Gruppe mit ihren jeweiligen Arbeits-, Sicht- und Verhaltensweisen stellen hohe Anforderungen an das Design von Groupware.

Bei der Kooperation verschiedener Personen in einer Gruppe kommen die oben genannten Aspekte der gegenseitigen Beeinflussung, wechselseitigen Abhängigkeit und Inkompatibilität der Ziele auch bei der Nutzung der Funktionalität einer Groupware stark zum Tragen. Dies wird beispielsweise manifest, indem mit Hilfe der Groupware Zugriffsrechte vergeben werden, wobei dies jedoch nicht zur Zufriedenheit aller Beteiligten geschieht. Während eine Person A ein Interesse daran haben mag, auf ein bestimmtes Dokument zugreifen zu können, kann die Person B, die das Dokument erzeugt hat, ein Interesse daran haben, es vor dem Zugriff anderer zu schützen, z. B. weil es noch nicht fertig bearbeitet ist. Bezüglich einer gemeinsam zu erledigenden Arbeit mögen beide Standpunkte ihre Berechtigung haben. Kooperationsunterstützende Software sollte einen so entstehenden Konflikt zwischen dem Wunsch von A, auf das Dokument zugreifen zu können und dem von B, es zu schützen, aufdecken helfen und zu seiner konstruktiven Wendung beitragen können.

Im folgenden sollen einige Mechanismen zur Konfliktbehandlung für solche Fälle vorgestellt werden, in denen ein Konflikt zwischen den Beteiligten manifest wird durch die Nutzung einer Groupwarefunktionalität, wie beispielsweise die Erteilung von Zugriffsrechten in einer Arbeitsgruppe.

Die Arbeiten hierzu werden in Zusammenarbeit mit POLITeam-Projekt[1] geleistet, aus dem auch die angeführten Beispiele gewählt wurden, die sich auf die beteiligten Pilotanwender und die dem Projekt zugrunde liegende Groupware LinkWorks[2] beziehen. Die genannten Mechanismen zur Konfliktbehandlung bei der Nutzung von Groupware sollen für ausgewählte Funktionen implementiert, auf die Nutzungssituation angepaßt und auf ihre Kooperationstauglichkeit untersucht werden.

4.1 Mechanismen zur Konfliktbehandlung bei der Nutzung von Groupware

Es stellt sich die Frage, wie mit Groupware-Funktionalitäten, die mit Konfliktpotentialen verbunden sind, umgegangen werden soll. Konflikte sind konstitutiv für kooperative Arbeitszusammenhänge: Es gibt keine dauerhafte Kooperation ohne Konflikt. Konflikte bieten den Beteiligten Anknüpfungspunkte, um Schwachstellen ihrer Kooperation aufzudecken oder zu erkennen, was bei der Zusammenarbeit noch thematisiert werden muß.

In der Regel ist diese Funktionalität für ihren Benutzer sinnvoll und hilfreich zur Aufgabenerledigung, so daß ihre generelle Deaktivierung nicht ratsam ist. Vielmehr sollte sie weitere Funktionalität speziell für Konfliktpotentiale bereit stellen, die es ermöglicht, die negativen Effekte von Konflikten zu vermindern oder zu beseitigen und die positiven zu fördern.

Dabei sind neben einer Konfigurierung der Groupware *vor* der Nutzung auch Mechanismen *während* der Nutzung sowohl zur Aufdeckung von Konfliktpotentialen als auch zur Auflösung entstandener Konflikte nötig. Drei dieser Mechanismen sollen im folgenden vorgestellt werden.

Gegensteuerbarkeit

Üblicherweise hat der Aktivator einer Funktion die Möglichkeit, eine Funktion ohne Einflußmöglichkeit anderer zu aktivieren. In diesem Fall wird von Steuerbarkeit gesprochen. In LinkWorks ist es beispielsweise einer Person A stets möglich zu sehen, wer gleichzeitig im System arbeitet. Die Kontrolle über eine Situation (Beispielsweise die Abfrage der eingeloggten Benutzer) liegt dann allein beim Aktivator der Funktion. Der Fall, daß Betroffenen à priori Interventionsmöglichkeiten bereitstehen, soll als Gegensteuerbarkeit bezeichnet werden. Gegensteuerbarkeit kann einer Person B beispielsweise erlauben, mit LinkWorks zu arbeiten, ohne daß eine andere eingeloggte Person dies erfahren kann. Eine

---

[1] Gegenstand des POLITeam-Projektes ist die (Weiter-)Entwicklung eines Softwaresystems, das räumlich verteilte und zeitlich versetzte Gruppenarbeit unterstützt. Im Projektkonsortium sind vertreten die Universität Bonn, die Gesellschaft für Mathematik und Datenverarbeitung St. Augustin und die VW-GEDAS, Pilotanwender sind ein Fahrzeughersteller, ein Bundesministerium und mehrere Landesministerien. Weitere Informationen finden sich unter `http://orgwis.gmd.de:80/POLITeam`.

[2] LinkWorks ist ein eingetragenes Warenzeichen der Digital Equipment Corporation.

Flexibilisierung der Starrheit von Steuerbarkeit und Gegensteuerbarkeit für konkrete Situationen verspricht Aushandelbarkeit (s.u.).

Transparenz

LinkWorks stellt mit dem Suchtool eine Funktionalität zur Verfügung, mit deren Hilfe netzweit Dokumente gesucht werden können. Die Interviews nach der Erstinstallation von LinkWorks bei den Anwendern ergaben u. a., daß die Benutzer mit Hilfe des Suchtools, jedoch ohne es zu wissen, unabsichtlich auf den (virtuellen) Schreibtischen von anderen Benutzern Objekte umbenannten oder ihnen Rechte darauf entzogen, ohne sie zu benachrichtigen und ohne daß die Betroffenen sich dagegen wehren konnten. Hier sind zwei Mechanismen zur Aufdeckung des Konfliktpotentials bzw. des tatsächlichen Konflikts notwendig: Zum einen soll dem Aktivator der Suchfunktion klar sein, daß er einer anderen Person ein Dokument auf deren Schreibtisch verändert, zum anderen soll die betroffene Person darüber informiert werden, warum ein Objekt auf dem Schreibtisch jetzt nicht mehr unter seinem alten Namen zu finden oder nicht mehr zugreifbar ist. Hierzu dienen die konfigurations- und die aktivierungsbezogene Transparenz:

Konfigurationsbezogene Transparenz

Die konfigurationsbezogene Transparenz informiert den Benutzer über die Funktionalität, die ein System insbesondere in der aktuellen Konfiguration zur Verfügung stellt, vornehmlich über diejenige Funktionalität, die Konfliktpotentiale birgt. Dabei soll die Funktionalität sowohl aus der Sicht des Aktivators, als auch aus der Sicht des Betroffenen einer Aktivierung einer Funktion dargestellt werden, so daß sich die Benutzer ein Bild davon machen können, wie eine spezielle Funktionalität auf alle Beteiligten wirkt. Konfigurationsbezogene Transparenz kann durch die Information über mit Konfliktpotentialen verbundene Funktionalität dazu beitragen, bei Benutzern ein stärkeres Bewußtsein für die Auswirkungen der eigenen Arbeit am System auf andere zu schaffen.

Aktivierungsbezogene Transparenz

Aktivierungsbezogene Transparenz macht dem Betroffenen den Gebrauch einer speziellen Funktion durch den Aktivator z. B. über eine abrufbare Systemnachricht sichtbar. Im Fall des oben beschriebenen Beispiels eines auf dem Schreibtisch umbenannten Dokuments wüßte also der Betroffene, warum er das Dokument nicht mehr unter dem alten Namen auf seinem Schreibtisch vorfindet.

Notwendig sind neben den *Informations*mechanismen, wie sie Transparenz zur Verfügung stellt, also auch *Interventions*mechanismen. Im folgenden soll dazu die Aushandelbarkeit vorgestellt werden.

Aushandelbarkeit

Im Gegensatz zur Transparenz bietet Aushandelbarkeit den Betroffenen die Möglichkeit, gegen die Entscheidung des Aktivators zu intervenieren. Dabei kann abhängig von der

Konfiguration der Betroffene dem Aktivator mitteilen und vom System aufzeichnen lassen, daß er mit der Aktivierung einer Funktion nicht einverstanden ist und um eine Nichtausführung bitten, oder er kann die Aktivierung der Funktion durch sein Veto ganz verhindern.

Als Beispiel für verschiedene Aushandlungsformen kann das Umbenennen eines Objekts oder die Änderung des Zugriffsprofils auf dem Schreibtisch einer anderen Person in LinkWorks dienen. In der ursprünglichen LinkWorks-Implementation war dies erlaubt, es fand also kein Aushandlungsprozeß statt. Sinnvoll kann hier eine einschleifige strukturierte Aushandlung sein, bei der der betroffenen Person A mitgeteilt wird, daß eine andere Person B ein Objekt auf ihrem Schreibtisch umbenennen oder das Zugriffsprofil ändern will, und A dazu die Zustimmung erteilen muß. Dazu könnte der betroffenen Person eine Dialogbox mit der entsprechenden Information und den Optionen "Zustimmung" und "Ablehnung" angezeigt werden. Auch eine semistrukturierte mehrschleifige Aushandlung kann hier sinnvoll sein, bei der die Beteiligten sich über einen zusätzlichen Text-, Audio- oder Video-Kanal darüber einigen, ob, wie und wann Person B das Objekt auf dem Schreibtisch von Person A umbenennen oder dessen Zugriffsprofil ändern darf.

Welche Form der Aushandelbarkeit an welchen Stellen sinnvoll ist und welche Default-Werte angenommen werden sollen, hängt stark von der konkreten Aufgabe und Zusammensetzung der Gruppe ab, die ein aushandlungsfähiges System einsetzt.

4.2 Organisatorische Voraussetzungen

Da die Designer nicht antizipieren können, welche der Konfliktpotentiale einer Groupware während der Nutzung in den Anwendungsfeldern mit seinen verschiedenen Gegebenheiten aktuell werden, muß die Groupware flexibel sein. Um diese Flexibilität nicht zu einer wahllosen Beliebigkeit der tatsächlichen Konfigurierung und Anpassung werden zu lassen, ist eine Strukturierung der Alternativen notwendig, die von den Designern nicht allein geleistet werden kann. Vielmehr ist es unabdingbar, die Benutzer bei der Analyse möglicher Konfliktpotentiale in der Designphase ebenso zu involvieren wie später bei der Konfigurierung der Groupware.

Konfigurierung

Die Konfigurierung der Groupware für bestimmte Funktionen sollten sich auf drei Ebenen erstrecken. Auf einer relativ abstrakten und verbindlichen Ebene sollten konfliktrelevante Fragen von grundlegender Bedeutung im Rahmen einer Betriebs-, bzw. Dienstvereinbarung geregelt werden. Darüber hinaus sollten konkretere Konfigurationen ebenso wie die Analyse möglicher Konfliktpotentiale auf einer Benutzerbefragung und diesbezüglichen Workshops basieren, auf dem Benutzer ihre Sicht potentieller Konflikte bei der Nutzung der Groupware diskutieren und abschließen die Ergebnisse der Diskussion in eine später auch noch änder- und verfeinerbare Konfiguration umsetzen. Schließlich sollte es für eine Arbeits(unter)gruppe

die Möglichkeit geben, Anpassungen an der Groupware vorzunehmen, die die Konfiguration für spezielle Teilbereiche der Aufgabenstellung oder für bestimmte Zeiten verfeinern.

4.3 Bemerkungen

Die hier vorgeschlagenen Mechanismen Gegensteuerbarkeit, konfigurationsbezogene Transparenz, aktivierungsbezogene Transparenz und Aushandelbarkeit können bei der notwendigen Aufdeckung und Behandlung von Konflikten bei der Nutzung von Groupware unterstützend wirken. Da jedoch Konflikte nie allein technisch begründet sind, kann an diese Mechanismen nicht der Anspruch erhoben werden, mit sämtlichen Konflikten in einer computerunterstützten Gruppe angemessen umzugehen. Vielmehr verbleibt ein weiter Bereich von konfliktären Situationen, die aus der Dynamik der Gruppe resultieren und für deren Behandlung jede Groupware stets eine nur unangemessene Unterstützung bietet. Zweifelsohne ist bei vielen Formen der computerunterstützten Kooperation neben einer technischen immer eine organisatorische Unterstützung einer Arbeitsgruppe notwendig, die in regelmäßigen Treffen, einer angemessenen Benutzerbetreuung und anderem bestehen kann. Trotzdem sind Mechanismen, wie sie hier vorgeschlagen werden, auf Dauer für die Kooperation einer Gruppe unabdingbar, da sie viele der auftretenden Konflikte sichtbar und durch einen darüber entstehenden Dialog nutzbar machen. Für eine sinnvolle Computerunterstützung für Arbeitsgruppen sind die Berücksichtigung der Konflikte bei der Nutzung von Groupware und entsprechende technische und organisatorische Maßnahmen von entscheidender Wichtigkeit.

5 CoDecide - Konfliktmanagement im Entwurf

Konfliktmanagement beruht sowohl auf formalen als auch auf informellen Elementen. Dabei ist das Wechselspiel zwischen formaler Interrelierung der konfligierenden Dokumente und informeller Diskussion das zentrale Problem des Konfliktmanagements. Die Lücke zwischen diesen beiden Polen muß durch eine intuitiv verständliche Visualisierung geschlagen werden, die einerseits die Konfliktpositionen formal repräsentiert und so einen Anschluß zu den Entwurfsdokumenten schafft und andererseits Grundlage für informelle Diskussion und gegenseitiges Verständnis ist.

5.1 Sichtenkopplung

Für die verschiedenen Sichten im Entwurf wurden unterschiedlichste Repräsentation entwickelt. Mit Hilfe dieser Repräsentation wird innerhalb der Sicht gearbeitet. Erst die oft bildhafte Darstellung abstrakter Sachverhalte macht es möglich, sich eine Vorstellung von einer Sicht zu machen. So werden beispielsweise in ER-Diagrammen die abstrakten Begriffe "Entität" und "Relation" durch Rechtecke bzw. Rauten visualisiert. Verbindungslinien zwischen den graphischen Elementen stellen die Abhängigkeiten zwischen Entitäten und Relationen dar.

Für das Management von Konflikten besteht eine noch größere Notwendigkeit einer geeigneten Repräsentation. (Komplizierte) Konflikte können nicht allein kognitiv gelöst werden. Konfliktmanagement muß deshalb genauso wie der Entwurf nicht nur auf einer "abstrakten Kopplungsebene" stattfinden, sondern benötigt eine adäquate Visualisierung (vgl. Abbildung 4).

Durch den Konflikt werden die Konfliktpositionen, die Sichten der einzelnen Entwickler, miteinander verbunden, die Sichten werden miteinander gekoppelt. Eine geeignete Visualisierung von Kopplungen ist fast noch wichtiger als die Darstellung beim Arbeiten innerhalb einer Sicht:

- ¥ Verschiedene Sichten basieren auf unterschiedlichen Vorstellungen. Abstrakt sind mehrere Sichten nur schwer faßbar.
- ¥ Bei der Kopplung von Sichten sind verschiedene Personen beteiligt. Typischerweise sind die einzelnen Personen oft nur mit einer Sicht vertraut, die jeweils andere Sicht wird von einer anderen Person vertreten. Gerade für innerhalb einer Sicht ungeübte Personen ist die geeignete Visualiserung aber von entscheidender Bedeutung.
- ¥ Der zur konstruktiven Lösung eines Konflikts notwendige Perspektivwechsel (Deutsch 1976) wird durch den Wechsel in eine eigenständige Konfliktnotation unterstützt. Umgekehrt ist der Perspektivwechsel nur schwer möglich, wenn aus der eigenen Notation heraus argumentiert wird. Die Entwickler sind Gefangene ihre eigenen Selbstverständlichkeit (Westerlund und Sjöstrand 1981).

Typischerweise wird zur Kopplung von Sichten die gleiche Notation benutzt wie zur Arbeit innerhalb einer Sicht, also zum eigentlichen Entwurf. So werden bei der Kopplung von Daten- und Prozeßsicht beispielsweise ER- und SA-Diagramm verwandt. Diese Diagramme wurden aber entwickelt, um innerhalb einer Sicht zu arbeiten. Sie sind nicht geeignet, um zwischen Sichten zu arbeiten. Für die Kopplung von Sichten, also für das Konfliktmanagement, ist eine eigenständige Visualisierung notwendig.

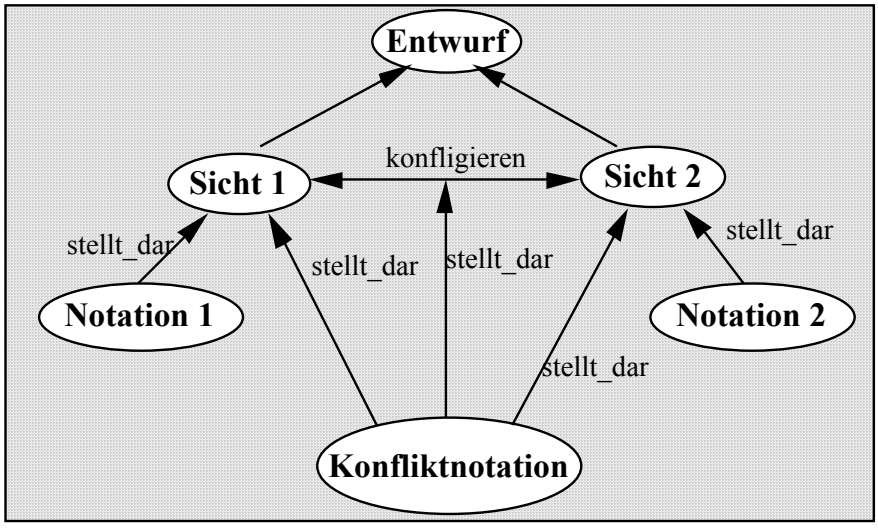

Abbildung 4: Sichtenkopplung mit Hilfe gekoppelter Darstellungen

5.2 CoDecide

CoDecide verwendet miteinander verknüpfte Matrizen zur Darstellung von Konfliktpositionen und Abhängigkeiten. Dabei ermöglichen Matrizen auf natürliche Weise, und damit intuitiv verständlich, die Gegenüberstellung von zwei Positionen. Mit Hilfe der Zeilen und Spalten können die Komponenten der Konfliktpositionen beschrieben werden, die Kreuzungspunkte von Zeilen und Spalten dienen zur Darstellung der Abhängigkeiten.

Der Vorteil von Matrizen liegt in ihrer strengen Struktur und ihrer intuitiven Verständlichkeit. Der hinter der Matrizenstruktur befindliche Formalismus wird nur unbewußt wahrgenommen und wirkt daher auf den Anwender natürlich und nicht störend. Darüber hinaus bietet der Aufbau der Matrizen die Möglichkeit, Matrizen zu verändern. Zeilen und Spalten können vertauscht, gelöscht und hinzugefügt, der Inhalt von Feldern kann verändert werden. Matrizen sind also dekomponier- und konfigurierbar.

Die Eignung von Matrizen als Visualisierungsmittel ist bekannt und wird in verschiedenen Anwendungen wie Tabellenkalkulation eingesetzt. Allerdings ist eine einzelne Matrix begrenzt. Sie verbindet lediglich zwei Sichten A und B. Entwurfskonflikte bestehen allerdings häufig aus mehr als einer Sicht, eine einfache Matrix reicht nicht aus, die Komplexität mehrerer Sichten darzustellen.

Wenn Sicht A nicht nur mit einer sondern mit *n* Sichten verbunden werden soll, sind dementsprechend *n* Matrizen zur Darstellung dieser Kopplungen notwendig. Die Zeile, die ein Element von A darstellt, endet nicht länger an der Grenze der Matrix sondern durchquert alle *n* Matrizen. Genauer: Alle Zeilen sind Elemente unendlich langer Bänder (vgl. Abbildung 5). Jedes Band besteht aus mehreren Zeilen (bzw. Spalten), die Spuren genannt werden.

Bänder laufen nicht parallel. Verschiedene Bänder können sich kreuzen. An den Kreuzungspunkten zweier Bänder entstehen Strukturen, sogenannte Kopplungen, die mit Hilfe von Matrizen visualisiert werden konnen. Kreuzungen zwischen mehr als zwei Bändern sind ebenfalls möglich. In diesem Fall liegt ein n-dimensionale Kopplung vor.

An dieser Kopplung wird offensichtlich, daß zwischen der Kopplung von Bändern als Datenstruktur und der Matrix als Visualisierung unterschieden werden muß. Während strukturell durchaus n-dimensionale Matrizen vorstellbar sind, ist es nur möglich, zweidimensionale Matrizen zu visualisieren. Von mehrdimensionalen Kopplungen sind lediglich zweidimensionale Projektionen darstellbar. Aus diesem Grund wird neben dem Band das Segment als zweite Datenstruktur eingeführt, um die Daten zu visualisieren (vgl. Abbildung 5).

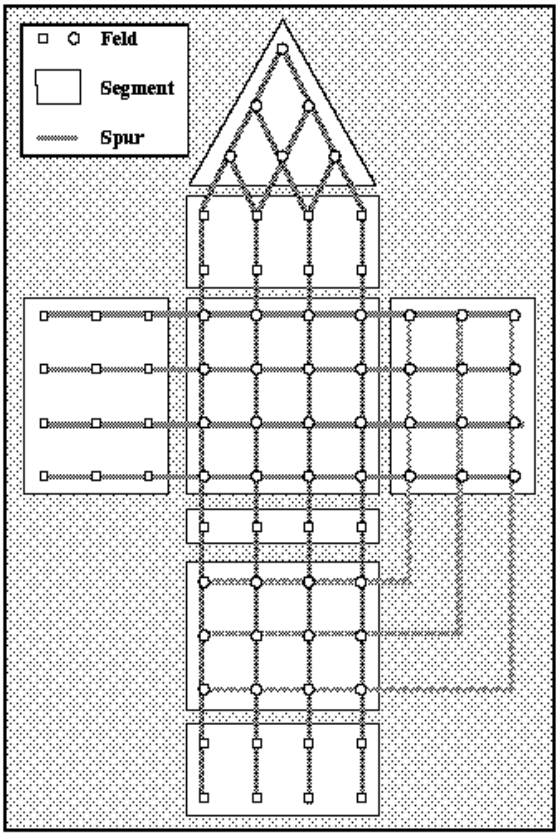

Abbildung 5: Segmente, Felder und Spuren zur Verknüpfung mehrerer Tabellen und Matrizen

Neben der Formalisierung ist die Fähigkeit zur Interaktion eine weitere Voraussetzung für Konfliktmanagement. Interaktion wird ebenfalls mit Multi-Matrizen ermöglicht.

Um mehrere Matrizen miteinander zu verbinden, werden Bänder eingeführt. Im Gegensatz zu Zeilen und Spalten enden Bänder nicht am Ende einer Matrix. Bänder enden auch nicht an den Grenzen eines Bildschirms oder eines Fensters. Das heißt, es ist möglich, Matrizen in unterschiedlichen Fenstern oder sogar auf verschiedenen, verteilten Rechnern durch Bänder aneinander zu koppeln. Auf diese Weise kann eine unmittelbare Interaktion erzielt werden. Die Manipulation einer Matrix macht sich direkt beim Kooperationspartner bemerkbar. Wenn beispielsweise Partner A seine Sicht um ein Element erweitert und eine neue Zeile einfügt, werden bei Partner B die von dieser Sicht abhängigen Matrizen um eine Zeile erweitert.

CoDecide ist ein Baukasten, der die Implementierung von Multi-Matrix-Editoren ermöglicht. CoDecide stellt neben Bausteinen für die Datenstrukturen (Bänder) und Bausteine für die Visualisierung (Segmente) auch noch Interaktionsmöglichkeiten und Schnittstellen zu Datenbanken (SQL, Telos) zur Verfügung, mit denen Entwurfsdokumente in eine CoDecide Darstellung transformiert werden können. Für eine detailliertere Darstellung von CoDecide sei auf (Jacobs 1995) verwiesen.

5.3 Anwendung: Dokumentenintegration im Requirements Engineering

CoDecide eröffnet vielen Anwendungen und Werkzeugen eine kooperative Dimension. CoDecide bietet die Möglichkeit, bereits bestehende Werkzeuge auf einer informalen Ebene miteinander zu verbinden, indem die entsprechenden Sichten miteinander gekoppelt werden.

Im folgenden Beispiel wird ein mit CoDecide entwickeltes Werkzeug in die CASE Umgebung von PRO-ART integriert. Ziel ist es, verschiedene Sichten wie z.B. Daten- und Datenflußsicht einander gegenüberzustellen und auf Konsistenz hin zu überprüfen.

Eins der zentralen Probleme des Requirements Engineerings ist die Gewährleistung der Nachvollziehbarkeit (traceability). Bei der Entwicklung eines Systems ist es notwendig, den Ursprung von Anforderungen herleiten zu können. PRO-ART ist ein Ansatz, die unterschiedlichen Dokumente, die während des RE-Prozesses entstehen, zu integrieren. Dazu werden die einzelnen Dokumente formalisiert und Abhängigkeiten zwischen den Dokumenten gespeichert.

Durch die formale Integration der Dokumente wird Kooperation auf einer methodischen Ebene erzielt. Abhängigkeiten lassen sich formalisieren und abspeichern. Auf einer formalen Ebene wird die Kopplung zwischen unterschiedlichen Sichten (Dokumenten) unterstützt. Allerdings fehlt eine geeignete Darstellung dieser Abhängigkeiten, die es dem Benutzer gestatten, die Sichten und deren Beziehungen im Entwurfsprozeß als Entscheidungsgrundlage zu verwenden.

In Abbildung 6 ist als Beispiel für ein Dokument ein Entity-Relationship-Diagramm zur Darstellung der Datensicht gewählt. Der entsprechende ER-Editor aus der PRO-ART Umgebung ermöglicht die Bearbeitung von ER-Dokumenten. Abhängigkeiten zu anderen Dokumenten wie beispielsweise Datenflußdiagrammen können aus dem Editor heraus erfragt werden. Allerdings gibt es keine übersichtliche Darstellung für die gesamte Kopplung zwischen diesen beiden Sichten. Genauso ist es zwischen informalen Sichten wie beispielsweise textuell formulierten Anforderungen in einem Hypertext-Dokument. Durch gezielte Anfragen erhält man zwar lokale Abhängigkeiten, die globale Kopplung wird allerdings nicht visualisiert.

Die Integration eines entsprechenden Werkzeugs mit Hilfe von CoDecide ermöglicht eine Abhilfe (Pohl und Jacobs 1994). Zwei Sichten können nun einander gegenübergestellt und gemeinsam visualisiert werden. In Abbildung 6 sind diese Sichten das ER-Diagramm und der Hypertext. (Der Hypertext ist aus Platzgründen lediglich in CoDecide visualisiert.) Da die Dokumente in PRO-ART formalisiert sind, kann man mit Hilfe von Transformationsregeln automatisch die für die Kopplung geeignete Darstellung generieren. Grundsätzlich kann dabei ein Informationsverlust entstehen, da CoDecide zur Darstellung lediglich Baumstrukturen und Strings zur Verfügung stellt. Im Falle eines ER-Diagramms bietet es sich beispielsweise an, Spezialisierungen zwischen Entitäten durch Vater-Sohn-Beziehungen in CoDecides Baumstruktur darzustellen. Die Vater-Sohn-Beziehungen können außerdem genutzt werden, um die zu den Entitäten gehörenden Attribute zu visualisieren.

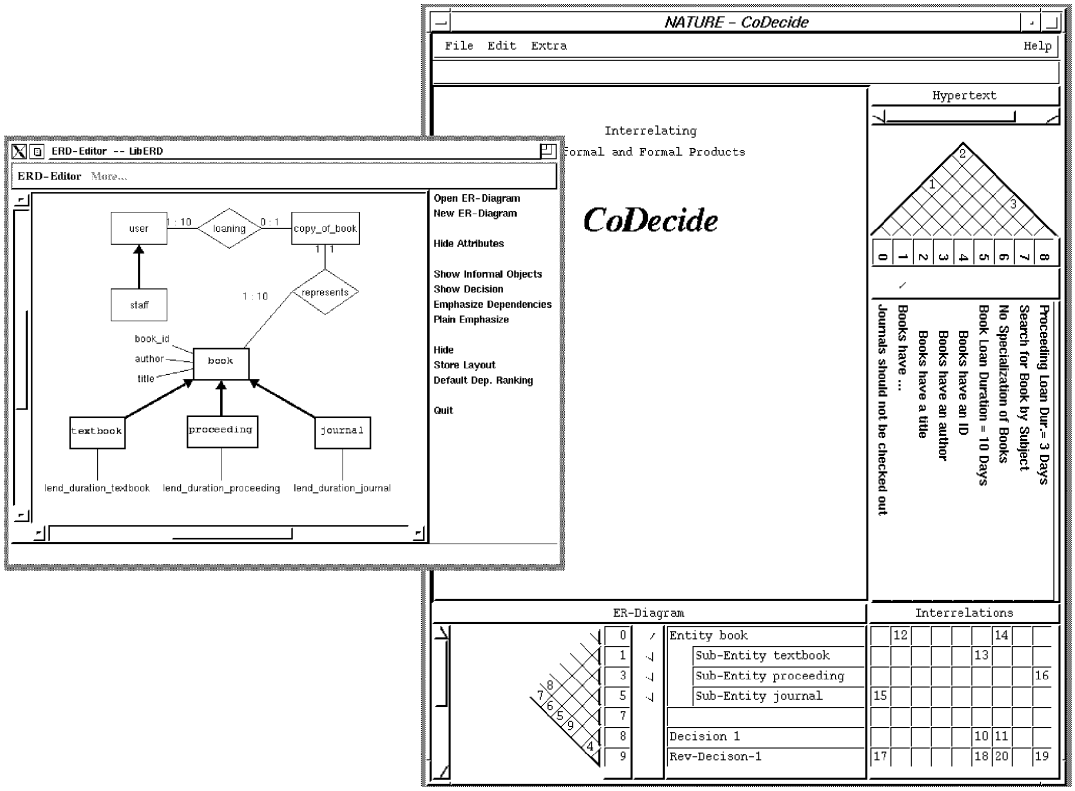

Abb. 6: Integration von CoDecide mit PRO-ART

Die explizite Darstellung von Abhängigkeiten zwischen Sichten ist eins der zentralen Probleme der Nachvollziehbarkeit. Beim Validieren von Spezifikationen muß beispielsweise nachgewiesen werden, daß alle Anforderungen erfüllt sind, und daß alle Funktionalitäten durch die Anforderungen auch tatsächlich verlangt werden. Insbesondere die Einordnung von nicht-funktionalen Anforderungen, wie beispielsweise die Benutzerfreundlichkeit, ist ein bislang ungelöstes Problem.

Mit Hilfe der Korrelationsmatrix kann man die Beziehungen (Kopplungen) visualisieren. Dabei beschreibt ein Eintrag in einem Feld der Matrix in den Grad der Erfüllung einer Anforderung. Leere Spalten und leere Zeilen deuten auf nicht erfüllte bzw. nicht geforderte Anforderungen hin.

Neben der Dokumentenintegration wurden eine Reihe weitere Beispiele entwickelt wie beispielsweise das zielorientierte Arbeiten bei der Fabrikplanung (Jacobs und Gebhardt 1995a) und bei der Geschäftsprozeßmodellierung (Jacobs und Holten 1995), bei der Verhandlung (Jarke et al. 1996) sowie bei der Ideengenerierung und -strukturierung (Jacobs 1996).

6 Zusammenfassung und Ausblick

Im vorliegenden Beitrag wurden verschiedene Sichten auf Konflikte bei kooperativer technikunterstützter Arbeit vorgestellt. Es zeigte sich, daß je nach den konkreten Anforderungen des Anwendungszusammenhangs unterschiedliche Konfliktmanagemenstrategien von Vorteil sind. Während beispielsweise bei der Verwaltung von Abhängigkeiten in einer Datenbank

weiterentwickelte traditionelle Vorgehensweisen zur Konflikterkennung und -vermeidung angemessen sind, erfordern die bei der Nutzung von Groupwarefunktionalität und die bei einem fachlichen Entwurfsprozeß zwischen verschiedenen Beteiligten entstehenden Konflikte oft eher ein Verständnis von Konflikt als kreativem Prozeß, der dementsprechend gestaltet und unterstützt werden muß. Eine umfassendere Einordnung weiterer in der Praxis der Arbeit mit informationstechnischen Systemen gegebener Konflikte hinsichtlich der Notwendigkeiten und Möglichkeiten, sie zu erkennen und zu lösen oder ihre kooperative und kreative Lösung durch die beteiligten Personen zu unterstützen, steht noch aus. Ein Konfliktmanagement, das diesen Namen wirklich verdient, darf jedoch nicht eingleisig konzeptioniert sein, sondern muß in verschiedenen Situationen aus einem breiten Spektrum die jeweils angemessene Methode oder Technik zur Verfügung stellen können.

7 Literatur

Adams, E. W., Honda, M. und Miller, T. C. (1989): Object Management in a CASE environment, in: Proc. of the 11th Int. Conf. on Software Engineering, IEEE Computer Science Press, pp. 154-163.

Breiing, A.; Flemming, M. (1993): Theorie und Methoden des Konstruierens. Springer Verlag.

Brooks, F.P.(1975): The Mythical Man-Month, Addison-Wesley

Buchman, A.P. (1994): Active Object Systems, in A. Dogac. M.T. Özsu, A. Biliris, T. Sellis (Eds.):Advances in Object-Oriented Database Systems, Springer Verlag.

Cellary, W. und Jomier, G. (1990): Consistency of Versions in Object-Oriented Databases, Proc. of the 16th Conference on Very Large Data Bases, Brisbane, Australia.

Cleetus, K.J.; Reddy, R. (1992): Concurrent Engineering Transactions. In: Proceedings of Conference and Exposition on CE & CALS, Concurrent Engineering & Computer Aided Logistics Support, Washington, month 6.

Constantine, L.L. (1993): Work Organizations: Paradigms for Project Management and Organization. In: Communications of the ACM, vol. 36, no.10, S. 35-46

Deutsch, M. (1969): Conflicts: Productive and Destructive. In: Journal of Social Issues, Vol.25, No.1, S. 7-41.

Deutsch, M. (1976): Konfliktregelung - Konstruktive und Destruktive Prozesse.

Driver, M.J.; Streufert, S. (1964): The General Incongruity Adaption Level (GIAL).

Easterbrook, S (1993): CSCW - Cooperation or Conflict, Springer Verlag

Easterbrook, S.; Finkelstein, A.; Kramer, J.; Nuseibeh, B. (1994): Coordinationg Distributed ViewPoints: the anatomy of a consistency check. In: Concurrent Engineering: Research and Applications (Special Issue on Conflict Management).

Ellis, C.A., Gibbs, S.J. und Rein, G.L. (1991): Groupware, In: Communications of the ACM, vol. 34, no. 1, S. 39-58.

Estublier, J. und Casallas, R. (1994): The Adele Configuration Manager. In W. F. Tichy (Ed.): Configuration Mangement. Trends in Software, Wiley.

Finkelstein A., Kramer, J., Nuseibeh, B., Finkelstein, L. und Goedicke, M. (1992): Viewpoints: A Framework for Integration Multipole Perspectives in System Development. In: International Journal of Software Engineering and Knowledge Engineering, vol.1, no.2.

Gebhardt, M. (1994): Kohärentes Design durch Sichtenkopplung. Diplomarbeit RWTH Aachen

Grudin ,J. (1994): Groupware and Social Dynamics: Eight Challenges for Developers. In: Communications of the ACM, vol.37, no.1, S. 93-105.

Hall, J. (1971): Decisions, Decisions, Decisions. In: Psychology Today, Vol.5, No.11, S. 51-54.

Heidegger, M. (1979): Sein und Zeit. Tübingen.

Hoffmann, L.R.; Harburg, E.; Maier, N.R.F. (1962): Differences and Disagreements as Factors in Creative Group Problem Solving. In: Journal of Abnormal and Social Psychology No. 64, S. 206-214

Jacobs, S. (1995): Konfliktmanagement im kooperativen Entwurf, RWTH Aachen

Jacobs, S. (1996): Teamwork Support in Requirements Engineering. In: M. Jarke (Hrsg.), The NATURE Approach to Requirements Engineering, Springer

Jacobs, S.; Gebhardt, M. (1995a): Kooperation im Entwurf mit Hilfe lose gekoppelter Sichten. In: Proceedings of the Gemeinsame Jahrestagung der Gesellschaft für Informatik (GI) und Schweizer Informatiker Gesellschaft (SI), Zürich.

Jacobs, S. and Gebhardt, M. (1995b): Supporting Mutual Understanding between Heterogenous Documents. In: Proceedings of the International Workshop on Concurrent/Simultaneous Engineering Frameworks and Applications, Lissabon.

Jacobs, S.; Holten, R. (1995): Goal Oriented Business Modelling - Supporting Decision Making within Information Systems Development. In: Proceedings of the Conference on Organizational Computing Systems, COOCS'95, Milpitas, CA.

Jarke, M., Gebhardt, M., Jacobs, S., Nissen, H.W. (1996): Conflict Analysis Across Heterogenous Viewpoints: Formalization and Visualization. In: Proceedings of the 29th Hawaii International Conference on System Sciences, Hawaii

Katz, R.H. (1990): Toward a Unified Framework for Version Modeling in Engineering Databases, ACM Computing Surveys, Vol. 22, No.4.

Kemper, F., Wilkes, W. und Schlageter, G. (1975): Active Relationships: A Means for Controlled Propagation of Information and Activities in Databases. Informatik Bericht Nr. 188, FernUniversität Hagen.

Klein, M. (ed) (1994): Workshop on Models of Conflict Management in Cooperative Problem Solving.

Klein, M. und Lu, S. (1990): Conflict Resolution in Cooperative Design. In: Journal for Artificial Intelligence in Engineering Design, vol.4, no.4, S.168-180

Kusiak, A. und Wang, J. (1994): Negotiation in Engineering Design. In: Group Decision and Negotiation, vol.3, no.1, S.69-91.

Mantei M. (1981): The Effect of Programmming Team Structures on Programming Tasks. In: Communications of the ACM, vol 24, no.3, S. 106-113

Miller, T. (1989): Configuration Management with the NSE, Proc. of the Intermational Workshop on Environments, Chinon, Frankreich.

Nuseibeh, B., Kramer, J. und Finkelstein, A. (1994): A Framework for Expressing the Relationships between Multiple Views in Requirements Specifications. In: IEEE Transactions on Software Engineering

Oakland, J. S. (1989): TQM - the new way to manage. In: Proc. 2nd Int. Conf. Total Quality Management, IFS Ltd/Springer Verlag, month 6, S. 3-17.

Oberquelle, H. (1991): Kooperative Arbeit und Computerunterstützung, Verlag für Angewandte Psychologie

Okada K., Maeda, F., Ichikawa, Y.und Matsuchita, Y. (1994): Multiparty Videoconferncing at Virtual Social Distances: MAJIC Design. In: Proceedings of the Conference on Computer Supported Cooperative Work, CSCW'94, Chapel Hill

Pasch, J. (1994): Software-Entwicklung im Team - Mehr Qualität durch das dialogische Prinzip bei der Projektarbeit. Springer Verlag.

Pohl, K. und Jacobs, S. (1994): Traceability between Cross-Functional Teams. In: Proceedings of the First COnference on Concurrent Engineering: Reseach and Applications, CE94, Pittsburgh

Putnam, L.L.; Poole, M.S. (1987): Conflict and Negotiation. In: L.W. Porter (ed.), Handbook of Organizational Communication: An Interdiscplinary Perspective, Sage, Beverly Hills, S. 549-599

Reddy, Y.V.R.; Srinivas, K.; Jagannathan, V.; Karinthi, R. (1993): Computer Support for Concurrent Engineering. In: COMPUTER, Vol. 27, No. 1, month 1, 1993, S. 12-16

Rodden, T. (1991): A Survey of CSCW Systems. In: Interacting with Computers, vol.3, no.3, S. 319-353

Rohde, M.; Pfeifer, A.; Wulf, V. (im Druck): Konfliktmanagement bei Vorgangsbearbeitungssystemen. In: Wirtschaftsinformatik (erscheint 1996)

Schlageter, G.; Stucky, W. (1983): Datenbanksysteme. 2. Auflage, Teubner Studienbücher.

Schmidt, K. (1994): The Organization of Cooperative Work: Beyond the "Leviathan" Conception of the Organization of Cooperative Work. In: Proceedings of the CSCW 1994, S. 101-112

Simon, H. A. (1969): The Sciences of the Artificial. Second Edition. Ed. Herbert A. Simon, The MIT Press, Boston

Simon, H. A. (1990): Die Wissenschaft vom Künstlichen. Kammerer & Unverzagt

Tanenbaum, M. (1981): Computer Networks. Prentice Hall

Thomas, K. (1976): Conflict and Conflict Management. In: M.D. Dunnette, Handbook of Industrial and Organizational Psychology Wiley, S. 889-935.

Tichy, W. F. (1985): RCS - A System for Version Control. Software - Practice and Experience, Vol. 15, No.7.

Westerlund, G. und Sjöstrand, S. (1981): Organisationsmythen. Klett-Cotta, Stuttgart

Winograd, T. und Flores, F. (1988): Erkenntnis, Maschinen, Verstehen. Rotbuch Rotationen.

Wohland, G. (1994): Jenseits von Taylor - Irritation als Methode. Der GMD-Spiegel, Vol.24, Nr.3, S.22-26.