Diplomarbeit

Universität Hamburg
Fachbereich Informatik
Arbeitsbereich Softwaretechnik

Softwarekonstruktion nach WAM zur Unterstützung von Kooperation an einem Ort am Beispiel eines Pausenplanersystems

Andreas Hartmann
August 1998

Betreuung:
Prof. Dr. Heinz Züllighoven
Dr. Daniel Moldt
Universität Hamburg

Diese Diplomarbeit wurde dem Fachbereich Informatik der Universität Hamburg zur teilweisen Erfüllung der Anforderungen zur Erlangung des Titels Diplom-Informatiker eingereicht.

Erklärung:

Hiermit versichere ich, die vorliegende Arbeit selbständig und unter ausschließlicher Zuhilfenahme der in der Arbeit aufgeführten Hilfsmittel erstellt zu haben.

Quickborn, am 16.02.1999

_____

Andreas Hartmann
Feldbehnstraße 80
25451 Quickborn

Betreuung: Prof. Dr. Heinz Züllighoven
Arbeitsbereich Softwaretechnik

Dr. Daniel Moldt
Arbeitsbereich Theoretische Grundlagen der Informatik

Fachbereich Informatik
Universität Hamburg

1 Einleitung

Menschliches Arbeitshandeln ist geprägt durch gemeinsames Handeln mehrerer Personen. Die Arbeit des Menschen wird verstärkt durch den Einsatz von Computern begleitet. Als Folge dieser Tatsache rücken Softwaresysteme, welche eine Gruppe von Menschen bei der Bewältigung von Aufgaben unterstützen, zunehmend in den Mittelpunkt des Interesses. Es entwickelte sich der neue Forschungsbereich des Computer Supported Cooperative Work (CSCW), welcher sich insbesondere mit Fragestellungen hinsichtlich der sinnvollen und effizienten Unterstützung von Zusammenarbeit durch Softwaresysteme beschäftigt.

> „Over the last half-dozen years, Computer-Supported Cooperative Work has emerged as an identifiable research field focused on the role of the computer in group work.“ ([Gre88], S. 5)

Die Zusammenarbeit von Menschen kann ausgesprochen viele verschiedene Formen annehmen. Dadurch werden Hersteller von Software, die Zusammenarbeit unterstützt, immer wieder mit neuen Fragestellungen konfrontiert. Ein weitverbreitetes Bemühen besteht darin, die Art der Zusammenarbeit zu kategorisieren und innerhalb einzelner Kategorien Erkenntnisse zu sammeln sowie Konzepte zu entwickeln, die eine einheitliche Sichtweise auf diese Teilbereiche schaffen. An dem Arbeitsbereich Softwaretechnik der Universität Hamburg werden ebenfalls Forschungen angestellt, die sich mit dem Bereich der CSCW beschäftigen. Jedoch betrachten die verschiedenen Forschungsrichtungen lediglich einen Teilbereich möglicher Zusammenarbeit mehrerer Personen. Ich beschäftige mich in dieser Arbeit mit einem Teilbereich der Kooperation und versuche, ein Konzept zur Konstruktion von Software zu entwickeln, welches diesen Bereich sowohl konzeptionell als auch praktisch erfaßt.

1.1 Ziele der Arbeit

Die Ziele dieser Arbeit lassen sich wie folgt umreißen:

- Ich möchte in dieser Arbeit ein Konzept entwickeln, welches geeignet ist, um bei der Entwicklung von Software zur Unterstützung einer konkreten Form der Zusammenarbeit eingesetzt zu werden. Zu diesem Zweck werde ich zunächst eine Taxonomie von Zusammenarbeit aufstellen, um daraus die hier betrachtete Form abzuleiten. Ich werde untersuchen, wodurch sie gekennzeichnet ist und werde im Hinblick darauf ein Konzept vorstellen, welches sich sowohl in fachlicher als auch in technischer Hinsicht an diesen Ergebnissen orientiert. Ich werde bei diesem Konzept verschiedene Ergebnisse der CSCW Forschung berücksichtigen und mich darüber hinaus an den bisherigen Forschungsergebnissen des Arbeitsbereiches Softwaretechnik

der Universität Hamburg orientieren. Insbesondere werde ich die dort erarbeiteten softwaretechnischen Konzepte zur Unterstützung von Zusammenarbeit erweitern.

- Um zu überprüfen, ob sich die Ergebnisse dieser Arbeit auch in einer praktischen Anwendung als benutzbar erweisen, werde ich ein System vorstellen, welches auf diesen Resultaten basiert und der Planung von Pausenaufsichten an einer Schule dient.

- Für die Entwicklung einer solchen praktischen Anwendung ist es insbesondere notwendig, über eine Technologie zu verfügen, welche verteilte Softwareentwicklung ermöglicht. Ich werde untersuchen inwiefern ein konkretes Rahmenwerk, welches sich in der Welt der Softwareentwicklung etabliert hat, die Entwicklung von verteilter Software mit Hilfe des entwickelten Konzeptes unterstützt. Es handelt sich dabei um das Rahmenwerk „Object Linking and Embedding" (OLE).[1] Da OLE in ein weiteres Rahmenwerk, die „Microsoft Foundation Classes" (MFC), welche die Entwicklung von Softwareanwendungen mit fensterbasierter Oberfläche unterstützt, integriert ist, werde ich letzteres ebenfalls, allerdings nur in geringerem Umfang, vorstellen.

1.2 Überblick

Zunächst stelle ich den grundsätzlichen Standpunkt und die Forschungsergebnisse des Arbeitsbereiches Softwaretechnik der Universität Hamburg vor. Ich konzentriere mich dabei auf die Aspekte, welche im Hinblick auf die Entwicklung von verteilten Softwaresystemen wichtig sind. Danach widme ich mich der Zusammenarbeit von Menschen, um herauszufinden, wodurch diese gekennzeichnet ist. Ich nehme eine Einteilung in verschiedene Formen der Zusammenarbeit vor und schildere bereits gewonnene Erkenntnisse hinsichtlich dieser Formen. Anschließend zeige ich, welche Formen von Zusammenarbeit durch Konzepte des Arbeitsbereiches unterstützt werden. Im Anschluß daran werde ich ein Konzept vorstellen, welches sich in den grundsätzlichen Standpunkt des Arbeitsbereiches einfügt und eine weitere Art der Zusammenarbeit unterstützt. Diese zunächst fachliche Betrachtung wird ergänzt durch Ansätze zur technischen Realisierung. Die Rahmenwerke MFC und OLE, welche die Softwareentwicklung insbesondere hinsichtlich verteilter Software erleichtern und welche für die Entwicklung des Pausenplanersystems benutzt wurden, stelle ich

[1] Sowohl MFC als auch OLE sind Produkte der Firma Microsoft. Ich werde im weiteren Verlauf der Arbeit die Microsoft Foundation Classes aus Gründen der Verständlichkeit als „die MFC" abkürzen und den Begriff im Singular verwenden. Ebenso werde ich von „OLE" sprechen.

danach vor, um das erarbeitete Konzept im letzten Teil der Arbeit durch die Vorstellung des Pausenplanersystems zu verifizieren.

Teile dieser Arbeit basieren auf einer Zusammenarbeit mit Tim Krauß, der ebenfalls eine Diplomarbeit schreibt. Insbesondere der Entwurf und die Implementation des Softwaresystems sind zu großen Teilen aus unserer Zusammenarbeit entstanden.[2] Seine Arbeit beschäftigt sich mit der Fragestellung, ob sich das MFC Rahmenwerk eignet, um im Rahmen der Lehre des Arbeitsbereiches Softwaretechnik benutzt werden zu können. Der Schwerpunkt liegt dabei auf der Werkzeugkonstruktion und -komposition. Aufgrund der bereits erwähnten Integration von OLE in die MFC lag es daher nahe, sich hinsichtlich der Verteilungstechnologie für OLE zu entscheiden und nicht andere Lösungen zu favorisieren.

1.3 Grafische Konventionen

Ich habe in der Arbeit verschiedene Formen der Notation bei den grafischen Darstellungen verwendet. Die Notation für die Grafiken im vorderen Teil der Arbeit ist selbst erklärend und orientieren sich nicht an einer konkreten Form der Darstellung. Im hinteren Teil der Arbeit benutze ich häufiger Diagramme, die der Erläuterung von konkreten Implementationen dienen. Sie entsprechen in der Darstellungsweise der UML Notation.[3] Darüber hinaus finden sich ebenfalls vorwiegend im hinteren Teil der Arbeit Interaktionsdiagramme, die dynamische Zusammenhänge verdeutlichen sollen. Sie basieren auf einer Notation, welche von Lilienthal et al. in [LS96] vorgeschlagen wird. Selbige ist der Darstellungsweise von Interaktionsdiagrammen in der UML Notation sehr ähnlich.

---

[2] Es handelt sich hier um [Kra98].

[3] Die Abkürzung UML steht für die „Unified Modelling language“, eine grafische Notation, die von der Firma Rational entwickelt wurde, um eine standardisierte Notation für verschiedene Dokumente des objektorientierten Entwicklungsprozesses zu definieren. Sie ist beschrieben in [Rat97].

2 Die WAM Methode

Die Entwicklung eines Softwaresystems stellt hohe Anforderungen an die beteiligten Gruppen. Softwareentwicklung sollte im Rahmen eines Vorgehensmodells stattfinden und erfordert ein hohes Maß an Kommunikation und Lernfähigkeit. Die Anforderungen an die beteiligten Personen fassen Kilberth et al. wie folgt zusammen:

> „• Die zentrale Aufgabe beim Softwareentwurf besteht darin, bei den Entwicklern ein Verständnis für den Anwendungsbereich zu schaffen.
>
> • Die Grundlage für dieses wachsende Verständnis ist die Kommunikation zwischen allen am Entwicklungsprozeß beteiligten Personen.
>
> • Die Entwickler müssen berücksichtigen, daß ein Anwendungsbereich auch während des Entwicklungsprozesses ständig Änderungen unterworfen ist.“ ([KGZ94], S. 13)

Der Arbeitsbereich Softwaretechnik der Universität Hamburg propagiert eine Reihe von Konzepten, die versuchen, den genannten Anforderungen gerecht zu werden. Im weiteren Verlauf der Arbeit werde ich den Begriff der „WAM Methode“ verwenden. Diesen möchte ich als Synonym für die an dem Arbeitsbereich Softwaretechnik der Universität Hamburg gelehrten Konzepte benutzen. Er beinhaltet zum einen die Softwareentwicklung mit Hilfe des Methodenrahmens STEPS. Zum anderen zähle ich das Leitbild des Arbeitsplatzes für qualifizierte menschliche Tätigkeit sowie verschiedene Entwurfsmetaphern zur WAM Methode hinzu. Ich werde diese Bereiche im Folgenden eingehend erläutern.

2.1 Einordnung

Der von Christiane Floyd an der Freien Universität Berlin entwickelte Methodenrahmen STEPS (**S**oftware**t**echnik für **e**volutionäre, **p**artizipative **S**ystemgestaltung) versteht die Softwareentwicklung als Lernprozeß.

Der Begriff evolutionär konzentriert sich dabei im wesentlichen auf den Erfahrungs- und Entwicklungsprozeß, den die beteiligten Personen im Rahmen einer zyklischen Vorgehensweise durchlaufen. Als Folge davon wird die Software, also die entstehenden Programmversionen, ebenfalls durch das zyklische Vorgehen sukzessive verfeinert. Die Erkenntnisse aus einem Zyklus fließen jeweils in den nachfolgenden Zyklus mit ein. Sowohl Fehler als auch Rückschritte werden in dieser Entwicklung als normal angesehen. Damit steht STEPS im Gegensatz zu anderen Vorgehensweisen, wie beispielsweise dem Phasenmodell, welches, nach dem Durchlauf verschiedener Phasen, im Endprodukt mündet.

Der Begriff partizipativ steht für die Beteiligung verschiedener Personengruppen am Entwicklungsprozeß. Diese Gruppen umfassen neben Entwicklern und Projektleitern insbesondere künftige Benutzer und andere Fachleute des Anwendungsbereiches. Aus den jeweils unterschiedlichen Sichtweisen dieser Gruppen auf den Anwendungsbereich resultiert ein hoher Bedarf an Kommunikation zwischen diesen, um dem entgegenzuwirken, was Bannon und Robinson als „ontological drift“ bezeichnen.[4] Da jede Gruppe Aussagen anders interpretiert, ist es möglich, daß sich eine zunehmende Verfälschung von Aussagen und Tatsachen etabliert. Dieses Phänomen kann sich über den gesamten Entwicklungsprozeß erstrecken und das Ergebnis der Arbeit vollständig entstellen. Diesem wird durch eine starke Kommunikation entgegengewirkt, welche wiederum dazu führt, daß die Beteiligten einen intensiven Lernprozeß durchlaufen. Insbesondere folgt daraus seitens der Systementwickler ein hoher Zuwachs an Verständnis für den Anwendungsbereich. Die zyklische Vorgehensweise sorgt für ein immer wiederkehrendes, erneutes Bewerten der Ergebnisse durch die beteiligten Gruppen. Die Partizipation künftiger Benutzer des Systems am Entwicklungsprozeß sorgt dafür, daß sie aktiv an der Gestaltung des Systems mitarbeiten können. Die Erkenntnisse aus der gemeinsamen Antizipation von künftigen Arbeitssituationen fließen in den Lernprozeß mit ein.

2.2 Das Leitbild des Arbeitsplatzes für qualifizierte menschliche Tätigkeit

Verschiedene Personengruppen sind an dem Softwareentwicklungsprozeß beteiligt. Diese betrachten den Anwendungsbereich aus jeweils verschiedenen Blickwinkeln. Jedoch ist es notwendig, daß trotz dieser unterschiedlichen Sichtweisen ein einheitliches Verständnis des Anwendungsbereiches geschaffen wird, um eine Kommunikations- und Diskussionsgrundlage gewährleisten zu können. Dieses läßt sich durch die Verwendung eines Leitbildes erreichen. Ein Leitbild definiert die Ziele einer Softwareentwicklung und hilft Menschen, sich im Anwendungsbereich orientieren zu können. Es legt gleichermaßen den Umgang aller am Prozeß beteiligten Personen miteinander fest.

> „Ein Leitbild ist eine benannte, mit Absicht eingenommene, grundsätzliche Sichtweise. Es ist eine Orientierung, die von Menschen angenommen wird, anhand der sie einen Ausschnitt von Realität wahrnehmen, verstehen und gestalten. Das Leitbild bestimmt, wie Entwickler und Benutzer bei der Systementwicklung wechselseitig miteinander umgehen und beschreibt die Gestaltungsziele bei der Softwareentwicklung.“ ([Gry96], S. 99)

[4] Vergleiche dazu [BR91].

Der Arbeitsbereich Softwaretechnik der Universität Hamburg propagiert das Leitbild des Arbeitsplatzes für qualifizierte menschliche Tätigkeit. Es stellt das Verhältnis zwischen Benutzer und Anwendungssystem in den Vordergrund und beschreibt die Eigenschaften der Benutzer wie folgt:

> „• Die Benutzer besitzen die notwendige Kompetenz und Fähigkeit, ihre Arbeit selbständig zu erledigen.
>
> • Es gibt keinen Grund, einen vordefinierten Arbeitsablauf einzuführen, da die Steuerung der Vorgänge durch den Benutzenden selbst flexibel gestaltet werden kann und soll.
> • Die zu erledigenden anspruchsvollen Aufgaben verlangen individuell einstellbare und organisierbare (Software)Umgebungen.“ ([Rie95], S. 6)

Das Leitbild geht also von qualifizierten Anwendern aus und steht damit, ebenso wie STEPS, im direkten Gegensatz zur tayloristischen Sichtweise, die den Anwender als einen Bediener des Systems im Rahmen eines auszuführenden Vorganges sieht. Das Handeln des Menschen ist situativen Veränderungen ausgesetzt. Diese Veränderungen fordern von dem Anwender die Fähigkeit, sich flexibel auf neu entstehende Situationen einstellen zu können. Ein Softwaresystem sollte diese Flexibilität unterstützen und dem Menschen ermöglichen, aufgrund seiner Erfahrung geeignete Aktionen auszuwählen und die notwendigen Handlungen durchzuführen.

2.3 Die Entwurfsmetaphern

Ein Leitbild sorgt bei den am Entwicklungsprozeß beteiligten Personengruppen für ein einheitliches Verständnis des Anwendungsbereiches. Entwurfsmetaphern helfen bei der Konkretisierung dieses Leitbildes:

> „Eine Entwurfsmetapher ist eine bildhafte Vorstellung, die ein Leitbild fachlich und konstruktiv konkretisiert. Eine Entwurfsmetapher hat sowohl eine fachliche, als auch eine technische Interpretation. Die fachliche Interpretation ist die Interpretation in der Umgangskategorie. Die technische Interpretation ist die Interpretation in der Maschinenkategorie.“ ([Gry96], S. 100)

Entwurfmetaphern helfen also einerseits bei der Orientierung im Anwendungsbereich und unterstützen andererseits ebenso die Übergänge von der Analyse über den Entwurf bis hin zur Implementation. Erreicht wird dieses dadurch, daß Metaphern in ihrer technischen Interpretation konkrete Vorschläge zur Softwarekonstruktion enthalten.

Zur WAM Methode zähle ich die Entwurfsmetaphern von Material, Aspekt, Automat, Werkzeug und Umgebung. Sie orientieren sich an der Identifikation der Gegenstände des Anwendungsbereiches, mit denen ein Mensch seine Aufgaben erledigt. In Abhängigkeit seiner Tätigkeit benutzt er Gegenstände, die er als Werkzeuge oder als Materialien sieht.

> „Menschen sind gewohnt, bei der täglichen Arbeit ihre Arbeitsmittel und –gegenstände als Werkzeuge oder Materialien zu sehen. Dieses gilt nicht nur in handwerklichen oder produzierenden Bereichen, sondern auch bei der Büroarbeit.“ ([KGZ94], S. 24)

Ziel dieser Entwurfsmetaphern ist es, daß der Benutzer diese Gegenstände seines Anwendungsbereiches auch in einem Softwaresystem wiederfindet und erkennt. Die Umstellung seiner Arbeit durch die Einführung eines Softwaresystems wird minimiert, da ihm der Umgang mit diesen Werkzeugen und Materialien vertraut ist. Es wird deutlich, daß diese Metaphern durchgängigen Charakter für alle Bereiche des Softwareentwicklungsprozesses haben.

> „Beim objektorientierten Entwurf lassen sich interaktive Anwendungssysteme als sinnvolle Zusammenstellung von Werkzeugen und Materialien in einer Arbeitsumgebung modellieren.“ ([KGZ94], S. 24)

Die Aufteilung der Gegenstände des Anwendungsbereiches in Werkzeuge und Materialien muß keineswegs zu disjunkten Mengen führen. So ist es möglich, daß Gegenstände, die im Rahmen einer Verwendung als Material erscheinen, anderweitig verwendet als Werkzeug identifizierbar sind. Als Beispiel möchte ich hier einen Bleistift anführen, der im Rahmen einer Formularbearbeitung als ein Werkzeug identifiziert werden kann, andererseits, wenn er angespitzt wird, ein Material darstellt.

Wie ich bereits eingangs dieses Kapitels erwähnt habe, ist eine Entwurfsmetapher geprägt durch eine Unterscheidung in fachliche und technische Interpretation. Durch diese Interpretationen wird es möglich, die Verbindung vom anwendungsfachlichen Kontext zu technischen Entwurfsentscheidungen zu knüpfen. Meyer legte bereits 1988 nahe, zwischen den Objekten des Anwendungsbereiches (den externen Objekten) und den programmiersprachlichen Repräsentanten (den internen Objekten) zu unterscheiden.[5] Ich werde diese Unterscheidung bei der nun folgenden Vorstellung der Metaphern deutlich machen.

[5] Vergleiche dazu [Mey90], S. 72.

2.3.1 Die Entwurfsmetapher Material

Materialien sind Arbeitsgegenstände, die bearbeitet werden. Sie liegen für den Benutzer am Arbeitsplatz greifbar vor. Materialien sind aber auch solche, die nicht greifbar sind, jedoch trotzdem von dem Menschen bearbeitet werden. In ([KGZ94], S. 15) werden Renditen oder Vertragskonditionen als Beispiel für solche Materialien angeführt. Gryczan beschreibt die Entwurfsmetapher des Materials wie folgt:

> „Material:
>
> fachliche Interpretation: Ein Material ist im Rahmen einer Aufgabenerledigung ein Arbeitsgegenstand. Materialien lassen sich in bestimmter Weise bearbeiten, d.h. verändern und sondieren.
>
> technische Interpretation: Eine Materialklasse definiert Operationen zur Veränderung und Sondierung des Zustandes von Objekten einer Klasse.“ ([Gry96], S. 102)

Materialien sind unabhängig von einer Darstellung zu entwerfen. Sie machen keine Annahmen über ihre Darstellung auf Bildschirmen, oder über ihre Repräsentation in einer Datenbank. Der Übergang von Materialien aus der Anwendungswelt in ihre technische Repräsentation ist üblicherweise begleitet von Abstraktionsprozessen, die sich an den Eigenschaften und Umgangsformen der Materialien orientieren.

2.3.2 Die Entwurfsmetapher Werkzeug

Werkzeuge sind Gegenstände des Anwendungsbereiches, die ein Benutzer für die Bearbeitung von Materialien verwendet. Bearbeitung bedeutet in diesem Zusammenhang sowohl die Veränderung als auch die Sondierung von Materialien. Eine eindeutige Übertragung von Werkzeugen des Anwendungsbereiches in das System ist häufig nicht möglich. Es gibt im Anwendungsbereich Werkzeuge, wie zum Beispiel ein Stift, die einen sehr universellen Charakter besitzen. Andere Werkzeuge sind hingegen sehr speziell. Es entsteht häufig die Situation, daß eine direkte Übertragung eines Werkzeugs in ein Softwaresystem keinen Sinn macht, da so der eigentliche Zweck dessen, wozu ein solches Werkzeug im Anwendungsbereich benutzt wird, verloren geht. So würde man die Übertragung eines Stiftes, mit dem ein Formular ausgefüllt werden soll, im Anwendungssystem in Form eines Formulareditors realisieren, um den Charakter der auszuführenden Tätigkeit nicht zu verlieren. Man wird bei dem Entwurf von Werkzeugen also weniger danach fragen, welche Werkzeuge es im Anwendungsbereich gibt, sondern vielmehr, wofür diese Werkzeuge benutzt werden.

Bei der Übertragung von Werkzeugen aus dem Anwendungsbereich in ein Softwaresystem spielt weiterhin die Handhabung derselben eine gravierende Rolle. Werkzeuge sind grundsätzlich reaktiv konzipiert. Bei reaktiven Werkzeugen werden Operationen ausschließlich durch Handlungen des Benutzers ausgelöst. Werkzeuge schreiben dem Benutzer keinerlei Ausführungsfolgen vor. Die Auswahl der Kommandos ist dem Benutzer überlassen. Diese Freiheit wird lediglich durch nicht mögliche Handlungsfolgen eingeschränkt.

Die Betrachtung von Werkzeugen bezog sich bisher auf deren fachliche Interpretation. Die technische Interpretation eines Werkzeugs beleuchtet den Aufbau und die Konzeption desselben beim Entwurf. Werkzeuge beinhalten einerseits die fachliche Funktionalität, die aus dem Anwendungsbereich heraus definiert wird. Andererseits haben Werkzeuge eine sichtbare Komponente, die Informationen darstellt und dem Benutzer eine Benutzung ermöglicht. Deshalb bestehen Werkzeuge immer aus einer Funktionskomponente (FK) und einer Interaktionskomponente (IAK). Die FK repräsentiert das Werkzeug innerhalb der Software und enthält das fachliche Wissen über die Verwendung des Werkzeugs. Die IAK ist für die Darstellung des Werkzeug selbst, für die Darstellung der für die Bearbeitung oder Sondierung des Materials relevanten Informationen und für die Entgegennahme von Kommandos durch die Benutzer verantwortlich.

Die beiden Komponenten werden über lose Kopplung miteinander verknüpft. Zum einen wird dadurch eine softwaretechnische Unabhängigkeit erreicht, zum anderen ist es möglich, unterschiedliche IAKs an eine FK zu koppeln. Dieses ist dann sinnvoll, wenn Informationen auf unterschiedliche Art und Weise dargestellt werden sollen, wie es beispielsweise bei statistischen Auswertungen der Fall ist und die Ergebnisse sowohl in Tabellenform als auch in grafischer Form angezeigt werden sollen.

Realisiert wird die lose Kopplung dadurch, daß die IAK die FK benutzt, während die FK über einen Beobachtermechanismus an die IAK geknüpft ist. Ein Beobachtermechanismus sorgt dafür, daß sich der Beobachter (hier die IAK) bei der FK als Beobachter vermerkt und die FK die IAK mittels einer sehr einfachen (Beobachter-)Schnittstelle über Veränderungen informiert.[6] Diese Konstruktion wird in Abbildung 1 durch ein Auge (IAK beobachtet) und durch eine Glocke (FK benachrichtigt) symbolisiert. Im Vordergrund steht hier weniger die konkrete Form der Umsetzung eines Beobachtungsmechanismus, sondern vielmehr die Tatsache, daß zwischen den beteiligten Komponenten eine lose Kopplung entsteht, welche flexibel in der Handhabung ist.

[6] Zum Beobachtermuster siehe [GJH+95], S. 293ff.

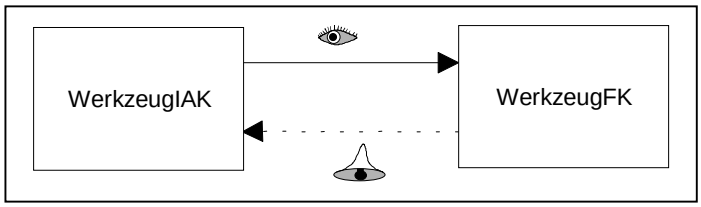


Abbildung 1 Kopplung von Interaktions- und Funktionskomponente

Diese lose Kopplung ermöglicht es, mehrere verschiedene Interaktionskomponenten an eine Funktionskomponente zu binden, da die Anzahl der Beobachter beliebig gewählt werden kann. Weitere Anregungen zur Werkzeugkonstruktion und –komposition finden sich in [Rie93], [Rie95] und [RZ95]. Zu den Variationsmöglichkeiten des Beobachtermusters lassen sich Anregungen in [GHJ+95], [RW96], [Vli97a] und [Vli97b] finden.

2.3.3 Die Verwendung von Werkzeugen und Materialien

Werkzeuge werden benutzt, um Materialien zu verändern oder zu sondieren. Die Auswahl eines angemessenen Werkzeugs für die Bearbeitung orientiert sich an dem Zweck der Bearbeitung. Es ist nicht jedes Werkzeug für jeden beliebigen Zweck geeignet. Vielmehr orientiert sich die Auswahl an dem Verwendungszusammenhang. Es ist jedoch durchaus sinnvoll, ein Werkzeug für die Bearbeitung verschiedener Materialien zu benutzen. So kann ein Formulareditor beispielsweise für die Bearbeitung von verschiedenen Formularen dienen. Um diesen gewünschten Verwendungszusammenhang explizit formulieren zu können bedient sich WAM der Entwurfsmetapher des Aspektes. Werkzeuge bearbeiten das Material niemals unter ihrem Typ. Vielmehr benutzt ein Werkzeug nur den Teil der Schnittstelle eines Materials, der für diesen Verwendungszusammenhang vorgesehen ist. Technisch gesehen bedeutet dieses, daß die Umgangsformen des Materials, die für den gewünschten Verwendungszusammenhang notwendig sind, in einem Aspekt zusammengefaßt werden.

Ein Aspekt beschreibt auf fachlicher Ebene die Form des Umgangs mit dem Material. Dieser Umgang wird in der technischen Interpretation durch eine Menge von Operationen ausgedrückt. Diese Operationen definieren das Verhalten einer Klasse, die über diesen Aspekt verfügt. Eine mögliche technische Realisierung eines Aspektes ist die einer abstrakten Oberklasse. Andere technische Realisierungen sind jedoch meines Erachtens durchaus möglich. Im Folgenden werde ich in grafischen Darstellungen, welche sich nicht konkret auf eine Implementation beziehen, das UML Symbol für ein Interface benutzen.

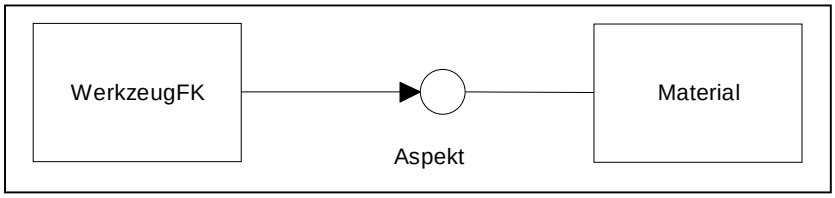


Abbildung 2 Verknüpfung von Werkzeug, Aspekt und Material

Die WerkzeugFK kennt lediglich das Interface des Materials. Das Material stellt dieses Interface zur Verfügung. Die Wahl der Aspekte entscheidet darüber, wie universell oder speziell ein Werkzeug verwendbar ist. Crüsemann et al. versuchten beispielsweise in [CHP+95], ein System mit möglichst generischen, also allgemein verwendbaren, Werkzeugen zu entwickeln.

UML Interfaces und Aspekte

Die UML verfügt neben der Möglichkeit, abstrakte Oberklassen darstellen zu können, ebenfalls über die Fähigkeit, die Benutzung einer Teilmenge von Operationen einer Klasse zu beschreiben. Diese Teilmenge von Operationen ist zusammengefaßt in einem Interface. Die grafische Darstellung eines Interfaces läßt neben der Interpretation, daß sich hinter dem Interface eine abstrakte Oberklasse verbirgt, ebenfalls andere Interpretationen offen. Die UML beschreibt ein Interface wie folgt:

> „An interface is the use of a type to describe the externally-visible behavior of a class, component, or other entity[...]“ ([Rat97], S. 25)

Der Begriff eines “type” ist wiederum wie folgt beschrieben:

> “A type is descriptor for objects with abstract state, concrete external operation specifications, and no operation implementations.” ([Rat97], S. 24)

So wären auch Interfaces im Sinne von Java eine mögliche technische Interpretation eines Aspektes.

2.3.4 Die Entwurfsmetapher Automat

Die Entwurfsmetapher Automat hat sich durch die Arbeit von Gryczan ([Gry96]) in der WAM Welt etabliert. Sie interpretiert automatisierte Vorgänge als einen Bestandteil des menschlichen Arbeitshandelns. Eine Definition eines Automaten lautet wie folgt:

> „Vorrichtung, die nach dem Einrichten und Beschicken vorbestimmte Handlungen nach einem Auslöseimpuls selbständig und zwangsläufig, unter Umständen auch überwacht und geregelt, auf mechan., elektr., [...] Wege ablaufen läßt.“ [Mey95]

Aufgrund dieser Definition entsteht zunächst der Eindruck, als wäre ein Automat für das Leitbild des Arbeitsplatzes für qualifizierte menschliche Tätigkeit unpassend, da es den Menschen in eine passive Rolle drängt. Gryczan argumentiert jedoch, daß der Umgang mit Automaten häufig durch

den Menschen gewollt ist, da sie ihm „lästige Routinearbeiten“ abnehmen und ein „vorhersehbares Ergebnis“ liefern.

> „Ein Automat ist die Vergegenständlichung einer formalisierten Routine, die über längere Zeiträume ohne äußere menschliche Eingriffe ablaufen kann. Ein Automat realisiert eine formalisierte Routine, die weitgehend ohne Kontextinformationen oder interaktive Steuerung auskommt. Einmal eingestellt, läuft die formalisierte Routine ab und produziert vorab festgelegte Resultate. Das Ergebnis oder der Effekt eines Automaten wird wieder in die situierte Handlung eingebettet.“ ([Gry96], S. 123)

Dieses schränkt er ein durch den Vorbehalt, daß der Anteil der durch den Automaten durchgeführten Arbeitshandlungen anwendungsfachlich vertretbar bleiben muß, also die Grundmerkmale des menschlichen Handelns nicht durch die Verwendung von Automaten eingeschränkt werden dürfen.[7]

Automaten repräsentieren Schablonen, die, nach der Einstellung von situationsspezifischen Parametern, stabil ablaufen und ein vorhersehbares Ergebnis liefern. Zu beachten ist dabei die Tatsache, daß Handlungsfolgen nur auf Automaten übertragen werden sollten, sofern der Benutzer diesen Ablauf vollständig durchdrungen und verstanden hat. Folglich sind die produzierten Ergebnisse aus seiner Erfahrung heraus plausibel und schlüssig. Da Automaten nur Teile des situierten Arbeitshandeln ausmachen und die Ergebnis wiederum in das Handeln des Menschen eingebettet werden, spricht Gryczan auch von „kleinen Automaten“.

2.3.5 Die Entwurfsmetapher Umgebung

Die Vorstellung der Entwurfsmetapher Umgebung möchte ich mit einem Zitat von Gryczan et al. beginnen:

> „Werkzeug und Material schaffen die entscheidende Verbindung von den am Arbeitsplatz gewohnten Arbeitsmitteln und –gegenständen zu den neuen Komponenten eines Softwaresystems. Sie müssen jedoch auch „ihren Platz haben“: die Arbeitsumgebung als weitere Entwurfsmetapher orientiert sich an Büroarbeitsplätzen, die wir uns entsprechend unseren Gewohnheiten und Ordnungsprinzipien einrichten.“ ([GWZ96], S. 5)

[7] Vergleiche dazu [Gry96], S. 120.

Die Arbeitsumgebung repräsentiert den Arbeitsplatz, an dem ein Benutzer tätig ist. Sie ist gekennzeichnet durch eine „vertraute Unauffälligkeit im Gebrauch“ ([BZ90], S. 98). Der Mensch richtet sie nach seinen eigenen Wünschen ein. Die Umgebung, welche die für die Arbeit des Benutzers relevanten Arbeitsmittel und Gegenstände verwaltet, erfüllt verschiedene andere Funktionalitäten:

> „The notion of environment allows us to think about constraints between tools and materials. The environment must provide means for ensuring consistency between them.“ ([RZ95], S. 6)

Die sinnvolle Benutzung von Materialien mit Werkzeugen über den Verwendungszusammenhang wird durch die Arbeitsumgebung gewährleistet. Innerhalb einer Umgebung darf ein Material von mehreren Werkzeugen bearbeitet werden. Ein Benutzer darf natürlich erwarten, daß Veränderungen eines Materials mittels eines Werkzeugs ebenso in allen anderen Werkzeugen, welche dieses Material bearbeiten, sichtbar sind.[8] Die Umgebung sorgt also dafür, daß Werkzeuge gegebenenfalls über die Veränderung von Material informiert werden.

Die Einführung der Arbeitsumgebung ermöglicht es, verschiedene Arbeitsplätze zu unterscheiden.[9] Dieser Ansatz führt den Begriff des Arbeitsplatzes eines Menschen ein, der in seiner Umgebung tätig ist, verläßt aber auch gleichzeitig die Orientierung des Leitbildes vom Arbeitsplatz für qualifizierte menschliche Tätigkeit an einem Einzelarbeitsplatz. Die Unterscheidung von Arbeitsplätzen eröffnet die Perspektive, verschiedene Arbeitsplätze in Beziehung zueinander setzen zu können. Dieses ist insbesondere dann wichtig, wenn man sich vor Augen führt, daß menschliches Arbeitshandeln sehr häufig durch gemeinsames Handeln mehrerer Personen geprägt ist. Jeder Mensch arbeitet zwar in seiner ihm vertrauten Umgebung, jedoch ist es möglich, diese Umgebungen im Hinblick auf die Zusammenarbeit von Menschen miteinander zu verknüpfen. Um gemeinsames Arbeitshandeln modellieren zu können, ist es notwendig, sich über die Zusammenarbeit der beteiligten Personen ein klares Bild zu verschaffen.

Die technische Interpretation der Metapher stellt eine Reihe von Anforderungen an den Entwurf einer Arbeitsumgebung. Ein Ereignisverwalter sorgt für die Benachrichtigung von Werkzeugen, sobald sich Materialien verändert haben. Die Werkzeuge beobachten diesen Ereignisverwalter. Abbildung 3 zeigt zwei Werkzeugobjekte, die sich bei dem Ereignisverwalter als Beobachter angemeldet haben, um Nachrichten über die Veränderung von Material mitgeteilt zu bekommen.

---

[8] Vergleiche dazu [Gry96] S. 142.

[9] Siehe [Gry96], S. 137f.

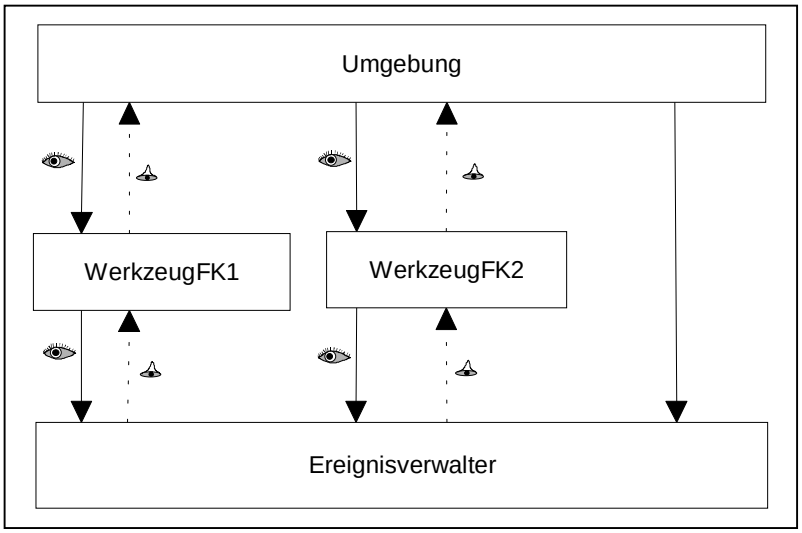


Abbildung 3 Der Ereignisverwalter und zwei Werkzeugobjekte in der Umgebung

Ändert die WerkzeugFK1 das Material, so teilt sie dieses dem Ereignisverwalter mit, welcher seinerseits die WerkzeugFK2 von dieser Änderung in Kenntnis setzt.

Eine weitere wichtige Funktion der Arbeitsumgebung ist die Verwaltung von Material. Materialien werden der Umgebung durch Automaten zur Verfügung gestellt. Dazu verfügt eine Umgebung über

- einen Materialverwalter. Dieser existiert in jeder Umgebung genau einmal. Bei ihm können Materialien nachgefragt werden. Er verwaltet die Materialien in einem Materialmagazin. Er bedient sich weiterhin eines oder mehrerer Materialversorger, um Material aus Datenbanken oder aus anderen persistenten Speichermedien zu beschaffen.
- ein Materialmagazin. Bereits angeforderte und somit beschaffte Materialien werden dort verwaltet, um mehrfaches Beschaffen zu verhindern.
- einen oder mehrere Materialversorger. Materialversorger sind Automaten, welche den Zugriff auf eine Datenbank ermöglichen. Eine Anfrage bei einem Materialversorger liefert eine Materialsammlung.

Eine Erweiterung des Umgebungskonzeptes für mehrere zusammenarbeitende Arbeitsumgebungen wird mittels eines Materialkoordinators realisiert. Dieser wird von den aktuellen Materialbearbeitungen der einzelnen Umgebungen informiert und entscheidet über Zuteilung des gewünschten Materials. Ich werde im weiteren Verlauf der Arbeit die technischen Aspekte der Umgebung im Hinblick auf mögliche Kooperation mehrerer Benutzer noch intensiver untersuchen.

2.4 Zusammenfassung und Ausblick

Ich habe hier das Leitbild des Arbeitsplatzes für qualifizierte menschliche Tätigkeit vorgestellt, sowie die Metaphern für Material, Werkzeug, Aspekt, Automat und Umgebung dargelegt. Im Mittelpunkt des Leitbildes steht der Mensch als qualifiziertes und eigenverantwortliches Individuum, welches in seinem Arbeitshandeln nicht durch ein Softwaresystem eingeschränkt werden sollte. Dem Menschen wird die Möglichkeit gegeben, seine Arbeitsumgebung gemäß seinen persönlichen Präferenzen zu konfigurieren. Er wird keinen Reihenfolgebedingungen hinsichtlich der Benutzung seiner Arbeitsmittel und –gegenstände unterworfen. Selbst ein ausgewogenes Maß an Automatisierung läßt sich in sein Arbeitshandeln sinnvoll integrieren, um ihn von lästigen Routinearbeiten zu befreien. Weiterhin habe ich angedeutet, daß die Metapher der Arbeitsumgebung Möglichkeiten eröffnet, Zusammenarbeit von mehreren Menschen zu unterstützen.

Die vorgestellten Konzepte werden ich insbesondere hinsichtlich der Unterstützung von Zusammenarbeit noch vertiefend betrachten. Es wird deutlich, daß die Ergebnisse von Gryczan die Zusammenarbeit mehrerer Menschen unter verschiedenen Einschränkungen ermöglichen. Diese Beschränkungen betreffen das gleichzeitige Zusammenarbeiten mehrerer Personen und insbesondere das gleichzeitige Bearbeiten desselben Materials im Rahmen dieser Zusammenarbeit. Ich werde im weiteren Verlauf der Arbeit ein Konzept vorstellen, welches sich in das Leitbild des Arbeitsplatzes für qualifizierte menschliche Tätigkeit einfügt und die Entwicklung von Software zur Unterstützung einer anderen Form der Zusammenarbeit von Menschen ermöglicht.

3 Kooperation und Koordination

Dieses Kapitel beschäftigt sich mit der Zusammenarbeit von Menschen. Nach einer Einleitung und der Klärung einiger Begriffe stelle ich eine Taxonomie von Formen der Zusammenarbeit auf. Ich weise im Rahmen der Diskussion dieser Bereiche insbesondere auf die Probleme hin, die bei der Übertragung der Zusammenarbeit auf Softwaresysteme entstehen können. Nachfolgend zeige ich auf, welche Bereiche der Zusammenarbeit durch die WAM Methode behandelt werden. Ich konzentriere mich im weiteren Verlauf auf eine Form der Zusammenarbeit, welche nicht durch die WAM Methode unterstützt wird. Zu diesem Zweck führe ich anhand von Szenarios in diesen Bereich ein, diskutiere ihn, um dann die wesentlichen Merkmale dieser Form von Zusammenarbeit darzulegen. Anschließend stelle ich ein Konzept vor, welches zur Entwicklung von Softwaresystemen benutzt werden kann, um die Zusammenarbeit mehrerer Personen an einem Ort zu unterstützen. Abschließend untersuche ich, ob dieses Konzept für das Leitbild des Arbeitsplatzes für qualifizierte menschliche Tätigkeit geeignet ist.

3.1 Einleitung

Menschliche Arbeit ist häufig dadurch gekennzeichnet, daß mehrere Personen an der Verrichtung einer Aufgabe beteiligt sind. Diese Personen arbeiten auf die unterschiedlichsten Arten und Weisen miteinander. Folglich müssen Softwaresysteme die jeweils für den Arbeitskontext angebrachte Form der Zusammenarbeit unterstützen. Insbesondere ist dabei wichtig, daß sich ein Softwaresystem an der Zusammenarbeit der Personen orientiert und nicht den Benutzern eine Form der Zusammenarbeit vorschreibt. Sørgaard erkennt, daß dieses häufig nicht der Fall ist:

> „The fact, that people share tasks, work closely together in teams, know each other, exercise mutual solidarity, etc., is not only unreflected in current computer systems. It is often hampered by the way the computer system change the conditions of work.“ ([Sør88], S. 2)

Er erwähnt weiterhin in seinem Artikel, daß es lediglich zwei Formen der Arbeitsunterstützung durch Softwaresysteme gibt. Zum einem handelt es sich dabei um die Unterstützung von Arbeit an einem Einzelplatz, während zum anderen Zusammenarbeit unterstützt wird, die sich durch formalisierte Abläufe auszeichnet und fest definierten Regeln unterliegt. Als eine Folge daraus erkennt er die Anforderung an die CSCW Forschung, die Unzulänglichkeiten heutiger Computersysteme im Hinblick auf die Zusammenarbeit von Menschen zu beseitigen. Die Notwendigkeit der flexiblen Form von Zusammenarbeit hat auch Robinson erkannt. Er ist der Ansicht, daß es sehr schwer ist, konkrete Formen der Zusammenarbeit von Menschen zu antizipieren. Er schließt dieses

aus der Beobachtung, daß Computersysteme häufig anders benutzt werden, als es die Entwickler des Systems vorgesehen haben. Der Grund hierfür findet sich darin, daß Arbeitssituationen, in denen Menschen sich befinden, immer unterschiedlich sind und sie deswegen nicht im voraus bedacht werden können:

> „Unanticipated use of computer artefacts reflects the fact, that work itself is undetermined until realised in situ." ([Rob93], S. 189)

Er vertritt in seinem Artikel die Meinung, daß dem Menschen durch ein Softwaresystem die Möglichkeit gegeben werden muß, sich im Rahmen alltäglicher Dinge eine eigene Form der Zusammenarbeit zu schaffen. Robinson bezeichnet diese alltäglichen Dinge mit dem Begriff der „common artefacts". Sie sind durch gewisse Eigenschaften im Umgang gekennzeichnet. Der übliche Umgang mit diesen Dingen schafft beim Menschen ein implizites Verständnis über die Arbeitssituation. Ein von ihm genanntes Beispiel ist dabei ein Schlüsselregal in einem Hotel.[10] Die eigentliche Funktion eines Schlüsselregals besteht darin, daß man Schlüssel hineinlegen oder daran aufhängen kann und man schnell und leicht erkennen kann, ob ein bestimmter Schlüssel gegenwärtig vorhanden ist. Die Personen, die an der Rezeption arbeiten, haben jedoch darüber hinaus die Benutzung so organisiert, daß eine Vielzahl verschiedener Arbeitssituationen am Zustand des Schlüsselregals abgelesen werden können. Sie benutzen dieses Regal, um sich oder den Gästen Nachrichten zukommen zu lassen oder um Rechnungen für Gäste bereitzulegen. Sie können aufgrund der Anzahl der Schlüssel auf die Belegung des Hotels schließen und andere wissenswerte Informationen an diesem Regal ablesen.

Die möglichst zwanglose Herausbildung von Formen der Zusammenarbeit ist also sehr wichtig. Antizipierte Abläufe wirken diesen eher entgegen, als daß sie förderlich wären. Es wird deutlich, daß die Unterstützung der Zusammenarbeit von Menschen ein sehr komplexes Thema ist. Um sie durch Softwaresysteme zu unterstützen, ist es unumgänglich, die konkrete Form der Zusammenarbeit eingehend zu analysieren. Bei der Gestaltung von Systemen ist darauf zu achten, daß den Benutzern ein größtmöglicher Freiraum gewährt wird, in dem sie selbst die Form ihrer Zusammenarbeit definieren können.

[10] Vergleiche dazu [Rob93].

3.2 Begriffe

Die Begriffe der Kooperation sowie der Koordination möchte ich hier von Gryczan übernehmen:

> „Als Kooperation wird das Zusammenwirken (mindestens) zweier Personen bezeichnet, die ihre Handlungen auf Grundlage wechselseitig abgestimmter Ziele und Pläne ausführen." ([Gry96], S. 60)

> „Koordination ist die wechselseitige Abstimmung bei der Kooperation." ([Gry96], S. 60)

Arbeiten verschiedene Personen an einem gemeinsamen Ergebnis, so ist diese Zusammenarbeit durch bestimmte Regeln definiert. Sie basieren auf den Erfahrungswerten der Beteiligten und werden von ihnen im Laufe der Zeit erlernt und immer weiter verbessert.

> „There is a big difference between two person's first attempt at carrying something together and people with experience do it" ([Sør88], S. 3)

Koordination kann also auch als ein Lernprozeß gesehen werden, der die Zusammenarbeit der beteiligten Personen sukzessive verfeinert. Weiß eine an dem Prozeß beteiligte Person, in welchem Zustand sich die Bearbeitung befindet, so kann sie daraus ableiten, welche Schritte im weiteren Verlauf der Arbeit notwendig sind, um das angestrebte Ergebnis zu erzielen.

Das Computersystem sollte den Menschen bei seiner Arbeit unterstützen. Um eine sinnvolle Unterstützung gewährleisten zu können, ist es notwendig, daß sich der Benutzer innerhalb des Systems orientieren kann und daß ihm darüber hinaus Möglichkeiten zur Beeinflussung an die Hand gegeben werden. Maaß definiert den Begriff der Transparenz wie folgt:

> "Ein System ist für eine Benutzerin transparent, wenn die Benutzerin sich ein zweckmäßiges mentales Modell vom System gemacht hat (Orientierung) und seine Anwendungs- und Handhabungsfunktionen im Rahmen ihrer Arbeitsaufgabe zielgerichtet einsetzen und einrichten kann (Beeinflussung)." ([Maa94], S. 12)

Insbesondere im Hinblick auf die Unterstützung der Zusammenarbeit mehrerer Personen ist es notwendig, daß den Benutzern die Kooperationssituation vergegenwärtigt wird, also der Orientierungsaspekt dieser Definition die Kooperation mit einschließt. Darüber hinaus müssen im Sinne des Beeinflussungsaspektes Koordinationsmechanismen für die Zusammenarbeit berücksichtigt werden. In der CSCW Literatur etabliert sich der Begriff der „Awareness", welcher sich auf den Orientierungsaspekt konzentriert. Ich zitiere hier Gutwin und Greenberg:

> „First, awareness is knowledge about a dynamic environment, and must be maintained as the environment changes over time. Second, awareness is maintained through perceptual information gathered through the environment. Third, awareness is generally secondary to some other goal; that is, it is generally *for* something else." ([GG97a], S. 1)

Dem Benutzer müssen die Veränderungen innerhalb der Arbeitssituation verdeutlicht werden. Dieses muß in einer für ihn wahrnehmbaren Form geschehen. Die Transparenz ist relevant für die Kooperationssituation. Im weiteren Verlauf dieser Arbeit wird deutlich werden, daß im Rahmen der Zusammenarbeit mehrerer Personen ein enger Zusammenhang zwischen dem Orientierungsaspekt der Transparenz (also der Awareness) und dem Beeinflussungsaspekt besteht.

3.3 Einordnung von Kooperation

Kooperation kann sehr vielgestaltig sein. Sie läßt sich durch verschiedenste Kriterien charakterisieren. Ich möchte hier Kriterien aufzählen, die meines Erachtens eine gravierende Rolle bei der Kooperation von Menschen spielen:

- Ort
  Personen befinden sich am gleichen Ort (Konferenz, Besprechung) oder an verschiedenen Orten (Telefonat).
- Zeit
  Zwei Personen beschäftigen sich gleichzeitig mit der Erfüllung derselben Aufgabe (synchron), oder sie erfüllen sie zu verschiedenen Zeiten (asynchron).
- Art der Nutzung von Arbeitsgegenständen.
  Arbeitsmittel können gleichzeitig, nacheinander oder abwechselnd benutzt werden.
- Der Grad der Transparenz
  Die Präsenz anderer Personen wird dem Benutzer vermittelt. Er wird entweder über die Aktivitäten anderer Mitarbeiter vollständig im unklaren gelassen, oder er wird präzise über deren Handlungen informiert.

Ich beschränke mich hier auf diese vier Merkmale, obwohl in der Literatur durchaus mehr als diese genannt werden. Kirsche gibt beispielsweise in [Kir94], neben den hier genannten, die Art des Vollzugs und die Anzahl der an der Zusammenarbeit beteiligten Personen als weitere Merkmale von Kooperation an. Ich werde im weiteren Verlauf des Kapitels eine grobe Einteilung von

Kooperation anhand der Kriterien Ort und Zeit vornehmen und diskutieren. Trotzdem verdienen die beiden zuletzt genannten Kriterien eine besondere Beachtung.

Der Zugriff auf die Arbeitsgegenstände spielt eine ausschlaggebende Rolle bei der Koordination. Der Begriff des „shared material“ wird von Sørgaard als wichtiges Koordinationsmedium erkannt:

> „There are many aspects of cooperative work. One of these is the use of shared materials. Much cooperation is based on silent coordination mediated by the shared material used in the work process.“ ([Sør88], S. 1)

Die Koordination erfolgt durch den Zustand des Materials, sowie durch seinen Ort. Diese beiden Faktoren geben dem Benutzer Informationen über einen Bearbeitungszustand und verringern die Notwendigkeit einer expliziten Absprache. Den Ort eines Materials als Koordinationsmöglichkeit anzusehen, birgt allerdings verschiedene Restriktionen in sich, die in der Natur des Materials begründet sind. Ich fasse hier kurz zusammen, welche Folgen Roock und Wolf in [RW98] für das Konzept des Ortes von Materialien sehen:

- Ein Material kann immer nur an einem Ort sein.
- Der Ort, an dem sich das Material befindet, muß nicht unbedingt allen Beteiligten bekannt sein.
- Der Zugriff auf ein Material ist nicht möglich, sofern sich das Material an einem Ort befindet, der nicht zugänglich ist.

Material ist also für einen Benutzer nur dann verfügbar, wenn er weiß, wo es sich befindet und er diesen Ort erreichen kann. Weiterhin kann die Verfügbarkeit durch die Tatsache eingeschränkt sein, daß das Material gerade bearbeitet wird und entweder eine synchrone Bearbeitung nicht zugelassen ist, oder ihm die Bearbeitung durch den aktuell Bearbeitenden verwehrt wird.

Es wird deutlich, daß es wichtig ist, feststellen zu können, wo und in welchem Zustand sich das Material befindet. Die Transparenz der Arbeit innerhalb einer Gruppe nimmt eine wichtige Rolle bei der Kooperation ein. Ich erwähnte bereits in der Einführung in das Leitbild des Arbeitsplatzes für qualifizierte menschliche Tätigkeit, daß Gryczan einen hohen Grad an Transparenz für eine sinnvolle Kooperation fordert.

Der Grad der Transparenz ist meines Erachtens in jedem Falle hoch anzusetzen, um dem Mitarbeiter seine Arbeit innerhalb einer Arbeitsgruppe zu vergegenwärtigen. Die Rolle des Materials muß in jeder Kooperationssituation neu überprüft werden. Es wird jedoch im weiteren Verlauf deutlich werden, daß die später vorgenommene Einteilung in Kooperationsformen zumindest teilweise die Möglichkeiten des Zugriffs auf Material implizit festlegt.

3.4 Eine Taxonomie von Kooperation

Um eine Unterstützung von Zusammenarbeit durch Softwaresysteme zu ermöglichen, ist es zunächst notwendig, sich ein Bild von der Vielfalt der möglichen Kooperationssituationen zu machen. Zu diesem Zweck möchte ich an dieser Stelle eine grobe Taxonomie von Zusammenarbeit anhand der Kriterien Ort und Zeit vornehmen. Ich betrachte dabei die Zusammenarbeit in ihrer natürlichen Form, also ohne die Unterstützung durch Softwaresysteme. Im Folgenden werde ich dann die einzelnen Teilbereiche näher betrachten und die Eigenheiten der Übertragung dieser Kooperationsformen auf Softwaresysteme benennen. In der Tabelle der Abbildung 4 unterscheide ich durch Spalten zwischen der Zusammenarbeit an einem Ort und an verschiedenen Orten. Durch Zeilen unterscheide ich zwischen der synchronen Zusammenarbeit, also der Zusammenarbeit, welche gleichzeitige Aktivitäten beteiligter Personen zwingend erfordert, und der asynchronen Zusammenarbeit, bei der keine gleichzeitige Aktivität notwendig ist.

	Gleicher Ort	**Verschiedene Orte**
Gleiche Zeit (synchron)	Tafel, Besprechungstisch	Telefon, Videokonferenz
Verschiedene Zeiten (asynchron)	schwarzes Brett	Brief, Postkorb

Abbildung 4 Einteilung von Kooperation

Innerhalb dieser Bereiche findet sich jeweils wiederum ein breites Spektrum an Formen der Zusammenarbeit. Dieses Spektrum entsteht durch die konkreten Formen der Kooperation und dem daraus resultierenden Koordinationsbedarf zwischen den beteiligten Personen.

Ich werde nun die Formen der Kooperationen im einzelnen betrachten und dabei die Folgen und Probleme eines Überganges von der realen Welt in die Welt der Softwaresysteme verdeutlichen. Bei diesem Übergang sind verschiedene Aspekte zu berücksichtigen. Insbesondere steht dabei im Mittelpunkt, wie die konkrete Kooperationsunterstützung durch das Softwaresystem gestaltet werden soll. Daraus resultiert unmittelbar die Frage nach dem Koordinationsbedarf für die beteiligten Personen. Insbesondere ist dabei zu untersuchen wie die Koordination zwischen den Personen gestaltet werden kann und es stellt sich die Frage, welche Mechanismen zu diesem Zweck in das System eingebracht werden sowie ob und wie sich die beteiligten Personen darüber hinaus außerhalb des Systems koordinieren.

Im Rahmen dieser Überlegungen nimmt das zu bearbeitende Material eine Schlüsselrolle ein. Es ist zu prüfen, ob es einen konkurrierenden Zugriff auf gemeinsames Material gibt und welche Mechanismen benötigt werden, um den Zugriff sinnvoll zu koordinieren.

Die Awareness ist für die Kooperation ebenfalls ein wichtiger Aspekt. Grundsätzlich ist Transparenz für die Kooperation mehrerer Personen ein wichtiger Faktor. Nichtsdestotrotz erfordern verschiedene Formen der Zusammenarbeit unterschiedliche Intensitäten der Awareness. Es ist also notwendig zu überprüfen wie stark den beteiligten Personen ihre Rolle innerhalb eines Kooperationsprozesses vergegenwärtigt werden muß. Darüber hinaus können Awareness Informationen koordinierenden Charakter haben, ohne daß eine regelnde Koordinationskomponente dazu nötig wäre. Ein Beispiel dafür wäre etwa der Zugriff auf ein Material, welches von einer anderen Person bereits bearbeitet wird. Durch einen Hinweis auf die aktuelle Bearbeitung des Materials könnte eine solche Konfliktsituation unterbunden werden, weil der Benutzer deshalb von einer simultanen Bearbeitung absieht.

Es wird sich zeigen, daß die genannten Kooperationsaspekte bei den hier identifizierten Kooperationsformen eine unterschiedliche Rolle spielen.

3.4.1 Asynchrone Kooperation an verschiedenen Orten

Die beteiligten Personen führen ihre Tätigkeiten an verschiedenen Orten zu verschiedenen Zeiten aus. Folglich können sie ein Material nur nacheinander bearbeiten, da es sich ja zu einem Zeitpunkt nur an einem Ort befinden kann. Ein gemeinsamer Zugriff auf ein Material ist ausgeschlossen, da die Menschen nicht gleichzeitig tätig sind. Vielmehr muß Material, nachdem es an einem Ort von einer Person bearbeitet wurde, zu einem anderen Ort gelangen, an dem es von einer anderen Person weiter bearbeitet werden kann. Transparenz kann für den Benutzer in hohem Maße gewährleistet werden. Informationen über den Aufenthaltsort und Zustand von Material stehen zur Verfügung. Dieses fördert insofern die Kooperation, als daß sich alle Teilnehmer über diesen Zustand informieren können und aufgrund ihrer Erfahrung sowohl feststellen können, inwiefern der gewünschte Bearbeitungszustand erreicht ist, als auch in der Lage sind abzuschätzen, ob und wann mit der Bearbeitung des Materials zu rechnen ist. Eine tiefergehende Diskussion dieser Kooperationsform erfolgt in Kapitel 3.5. Dort wird gezeigt, daß es eine Entwurfsmetapher gibt, die diese Form der Kooperation unterstützt.

3.4.2 Synchrone Kooperation an verschiedenen Orten

Zur gleichen Zeit führen Menschen an verschiedenen Orten Tätigkeiten aus, die einem gemeinsamen Ziel dienen. Sie arbeiten an verschiedenen Materialien, die im Rahmen der Aufgabe der Arbeitsgruppe relevant sind. Ein gemeinsames Bearbeiten eines Materials ist nicht möglich, da Materialien sich immer nur an einem Ort befinden können. Jedoch ist eine Zusammenarbeit, bei der verschiedene Personen verschiedene Materialien bearbeiten, um sie im weiteren Verlauf der Arbeit wieder zu einem Material zusammenzuführen, im Rahmen dieser Kooperation durchaus denkbar.

Ein Beispiel dafür wäre eine Kundenakte, die bearbeitet werden muß. Sie enthält mehrere Verträge, die gleichzeitig von verschiedenen Sachbearbeitern ausgefüllt und unterschrieben werden können. Zu diesem Zweck können die Sachbearbeiter die Verträge aus der Akte herausnehmen und bearbeiten. Während die Kundenakte aus obiger Argumentation heraus nicht durch mehrere Sachbearbeiter simultan bearbeitet werden kann, ist es jedoch durchaus möglich, die Verträge als „Teilmaterialien" zu betrachten, die jeweils gleichzeitig von verschiedenen Personen bearbeitet werden.

Koordinationsmechanismen können notwendig sein, um die Synchronizität dieser Kooperation zu gewährleisten. Allerdings können sich diese nicht auf konkurrierenden Materialzugriff beziehen, da dieser hier nicht möglich ist.

Räumliche Barrieren werden durch den Einsatz von Technologie zunehmend aufgehoben. Beispielsweise wird es durch ein Telefon oder eine Videoübertragung möglich, daß mehrere Personen an einer Besprechung teilnehmen, obwohl sie sich an verschiedenen Orten befinden. Sie können also in einen anderen Raum hinein sehen oder hören. Durch Softwareunterstützung ist es nun auch möglich, daß sich mehrere Personen rechnergestützt an einen „virtuellen Ort" begeben, an dem sie kooperieren können. Fitzgerald et al. sprechen in diesem Zusammenhang von „social worlds", die einen Bereich kennzeichnen, in dem eine oder mehrere Personen tätig sein können.[11] Eine „social world" muß einen Platz und eine Bedeutung haben. Es handelt sich dabei bewußt nicht um einen Raum, da der Platz nur einen semantischen Charakter hat und keine physikalische Ausdehnung besitzt. Roock und Wolf vertreten mit der Raummetapher einen ähnlichen Ansatz.[12] Zwei Personen, die sich an verschiedenen geografischen Orten befinden, können denselben virtuellen Raum betreten, um dort zu kooperieren. Personen können andere Personen in einem Raum wahrnehmen. Weiterhin können sie in andere Räume hineinsehen und feststellen, wer sich dort befindet.

Beide erwähnten Ansätze basieren auf der Idee, daß sich Personen aufeinander zu bewegen. Sie „treffen" sich, um zu kooperieren. Der Treffpunkt ist ein ausgezeichneter Platz, den die Beteiligten kennen. Insofern liegen auf softwaregestützter Ebene Parallelen zu der synchronen Kooperation an einem Ort nahe.

Durch die Schaffung eines virtuellen Ortes, an dem sich Personen treffen können, entsteht allerdings eine völlig neue und bisher nicht da gewesene Kooperationssituation. Diese kann sich

[11] Vergleiche dazu [FKT95].

[12] Siehe [RW98].

folglich auch stark von der bisherigen unterscheiden und dementsprechend ganz andere Kooperations- und Koordinationsmechanismen erfordern.

3.4.3 Asynchrone Kooperation an einem Ort

Bei der Kooperation an einem Ort, aber zu unterschiedlichen Zeitpunkten, befindet sich das Material zwangsläufig an dem Ort, welcher für die Kooperation definiert wurde. Koordinationsmechanismen sind hier von der konkreten Arbeitssituation abhängig. Handelt es sich hier um eine Kooperation bei der sich die beteiligten Personen in ihren Aktivitäten abwechseln, wie es etwa bei Schichtdienst der Fall wäre, so ist eine regelnde Komponente für den konkurrierenden Materialzugriff nicht notwendig, da er, aufgrund der asynchronen Form der Kooperation, nicht stattfinden kann. Trotzdem sind möglicherweise andere Koordinationsaspekte wichtig, welche auf aktuelle Bearbeitungszustände hinweisen und gegebenenfalls eine regelnde Komponente im Hinblick darauf notwendig machen.

Finden sich jedoch verschiedene Personen zu beliebigen Zeiten an dem Kooperationsort ein, so erfordert dieses möglicherweise ganz andere Koordinationsmechanismen. Hier ist insbesondere eine regelnde Koordinationskomponente notwendig, welche den Materialzugriff steuern muß. Es bieten sich hier verschieden Ansätze an. Sie kann nach dem Leser/Schreiber[13] Konzept konkurrierenden Nachfragenden gegebenenfalls den Zugriff auf bereits in Bearbeitung befindliche Materialien untersagen, oder aber möglicherweise ein Konzept von Original und Kopie realisieren.

Die Raummetapher (vgl. [RW98]) läßt die Interpretation zu, daß eine Person, die sich an ihrem Schreibtisch befindet, in einen anderen Raum hineinsehen kann. In diesem Raum kann sich ein schwarzes Brett befinden. Wenn Personen neue Informationen an das schwarze Brett hängen, so kann derjenige, der dieses schwarze Brett sehen kann, diese Änderung erkennen. Hier lassen sich wiederum gewisse Ähnlichkeiten zur synchronen Kooperation an einem Ort erkennen, welche wiederum eine Koordinationsnotwendigkeit zur Folge hat, welche im Abschnitt über synchrone Kooperation an einem Ort näher beschrieben wird.

3.4.4 Synchrone Kooperation an einem Ort

Verschiedene Personen arbeiten gleichzeitig an einem Ort. Das Material befindet sich an diesem Ort. Koordinationsaspekte sind hier sehr wichtig, da jede Person in der Lage ist, auf das Material

[13] Die Synchronisationsbedingungen des Leser/Schreiber Konzeptes finden sich z.B. in [JV87] auf Seite 188.

zuzugreifen. Insofern ist es naheliegend, eine regelnde Koordinationskomponente in das System zu integrieren, welche eine der bereits in Kapitel 3.4.3 beschriebenen Zugriffsstrategien realisiert.

Über diese Lösungen hinaus wurden verschiedene Versuche unternommen, eine gleichzeitige Bearbeitung des Materials zu ermöglichen:

- In [Gre86] wurde mit CES (Collaborative Editing System) der Versuch unternommen mehrere Personen gleichzeitig einen Text bearbeiten zu lassen. Die zugrunde liegende Idee war dabei, den Text in Passagen aufzuteilen, welche durch eine Schreibsperre einem Bearbeiter zugeordnet wurde, während andere diese Passage ebenfalls sehen, aber nicht ändern können.
- In [FS86] wird das System Cognoter beschrieben, welches den konkurrierenden Zugriff auf Materialien nicht einschränkt. Es wird dem Benutzer lediglich vermittelt, wer augenblicklich an dem Material arbeitet. Er kann selbst entscheiden, ob er eine gleichzeitige Bearbeitung aufnimmt oder nicht.

Insbesondere diese Ansätze unterstreichen die Feststellung von Gutwin und Greenberg, daß die Awareness für eine Kooperation an einem Ort sehr hoch sein sollte, da sich dadurch die Notwendigkeit zur Koordination verringert:

> „Workspace awareness reduces the effort needed to coordinate tasks and resources, helps people move between individual and shared activities, provides a context in which to interpret other's utterance, and allows anticipation of other's actions." ([GG97a], S. 2)

Ein Ansatz, der ein sehr hohes Maß an Transparenz realisiert, ist das von Stefik et al. entwickelte WYSIWIS („What You See Is What I See"). Die strenge Form des WYSIWIS bedeutet, daß alle beteiligten Personen immer exakt dasselbe sehen. Dieses geht soweit, daß eine Person auch die Cursor der anderen Personen auf seinem Bildschirm sieht. Insbesondere bei zunehmender Mitarbeiterzahl führt dieser Ansatz jedoch sehr schnell zu Verwirrung und Undurchschaubarkeit, da die Beteiligten mit einer Fülle von Informationen überhäuft werden. Daher stufte Stefik diesen Ansatz als nahezu ungeeignet für kooperative Arbeit ein:

> „[...],we have discovered that WYSIWIS is at once crucial and too inflexible in its strictest sense. WYSIWIS must be relaxed for all our software tools to better accommodate important interactions in meetings. Relaxations to WYSIWIS are characterised in terms of constraints on its four key dimensions: display space, time of display, subgroup population, and congruence of view." ([SBL+86] S. 276)

Stefik et al. entwickelten daraus den Begriff des „entspannten WYSIWIS“[14], welches sich im Bereich der synchronen Kooperation zunehmend konsolidiert. Dabei werden Einschränkungen hinsichtlich der vier oben genannten Aspekte zugelassen. Aufgrund der Beschränkung, die ein Bildschirmarbeitsplatz mit sich bringt, muß nicht notwendigerweise alles auf einmal sichtbar sein. Ebenso muß nicht jede Änderung für die beteiligten Kooperationspartner sofort sichtbar sein. Weiterhin schlagen die Autoren vor, die beteiligten Personen Gruppen zuzuordnen, um somit Informationsbereiche dadurch vergröbern zu können, daß die Aktivitäten dieser Gruppen anderen Personen nur angedeutet werden. Ebenso ist eine Kongruenz der Oberfläche der beteiligten Personen nicht erforderlich. In [OO93] wurden diese Ideen mit einem verteilt arbeitenden Editor „ShrEdit“ umgesetzt. Es wurde damit untersucht, welche Vor- und Nachteile das softwaregestützte Zusammenarbeiten im Rahmen von Diskussionsrunden mit sich bringt. Der Materialzugriff erfolgt in diesem System gleichzeitig.

3.4.5 Diskussion der Taxonomie

Bei der Betrachtung der verschiedenen Kooperationsformen fällt auf, daß die in der realen Welt vorgenommene Einteilung von Kooperation anhand von Zeit und Ort bei der Abbildung auf ein Softwaresystem nicht eindeutig bestehen bleibt. Es zeigen sich Ähnlichkeiten zwischen der synchronen Kooperation an einem und verschiedenen Orten, sobald die Möglichkeit besteht einen gemeinsamen Ort mit Hilfe eines Softwaresystems zu simulieren. Ebenfalls kann es in Abhängigkeit der konkreten Ausprägung Gleichnisse zwischen der synchronen und der asynchronen Kooperation an einem Ort geben.

Als Folge davon gehen die verschiedenen Bereiche bei der Abbildung der Kooperationsformen auf unterstützende Softwaresysteme fließend in einander über. Insbesondere die Möglichkeit einen gemeinsamen virtuellen Kooperationsort zur Verfügung zu stellen sorgt dafür, daß die Unterscheidung in Kooperation an einem und an verschiedenen Orten nicht notwendigerweise vorgenommen werden muß. Allerdings folgt aus diesen Veränderungen, daß sich ebenfalls die Kooperationssituation verändert. Arbeiteten bisher Personen räumlich getrennt voneinander, so treffen sie sich nun an einem Ort mit anderen Personen. Die Zusammenarbeit, welche sich im Laufe der Zeit herausgebildet hat, nimmt nunmehr eine andere Form an. Diese Veränderungen haben wiederum hat eine differenzierte Koordinationshandhabung zur Folge. Die Koordinationsmechanismen, welche in der realen Welt benutzt wurden, um eine sinnvolle Zusammenarbeit zu ermöglichen lassen sich

[14] Der Begriff des „relaxed WYSIWIS“ stammt aus [SBL+86].

möglicherweise nicht ohne weiteres auf ein Softwaresystem übertragen, so daß hier andere Mechanismen benutzt werden müssen.

Eine weitere Erkenntnis ist die, daß in den verschiedenen Bereichen ähnliche Mechanismen benötigt werden. Ein Beispiel findet sich in der häufig erkannten Notwendigkeit nach Awareness-Informationen beziehungsweise Transparenz, um die Koordinationsmöglichkeiten der Kooperationspartner zu verbessern.

Darüber hinaus werden oftmals ähnliche Mechanismen benötigt, welche die Koordination des Materialzugriffs unterstützen.

Aus der Erkenntnis heraus, daß die Abbildung von realen Kooperationssituationen auf eine softwaregestützte Ebene zu neuen Arbeitssituationen führen kann, folgt, daß Konzepte der realen Welt und bekannte Mechanismen überdacht werden müssen. Es muß festgestellt werden, ob sie im Rahmen einer Softwareunterstützung Bestand haben.

Es wird ebenfalls deutlich, daß durch die Einführung eines gemeinsamen virtuellen Ortes ein neuer Themenbereich der Kooperation entsteht. Im weiteren Verlauf dieser Arbeit werde ich mich einem Teil dieser Form der Zusammenarbeit widmen, bei der es gewünscht ist, daß beteiligte Personen die Beschränkungen, die ihnen durch verschiedene Aufenthaltsorte auferlegt werden, überwinden. Dieses erfordert insbesondere eine kritische Betrachtung der Art und Weise, wie diese Personen miteinander umgehen und wie sie sich und den Zugriff auf Arbeitsmaterialien koordinieren.

3.5 Kooperationsformen und Entwurfsmetaphern

Gryczan hat den Begriff der Arbeitsumgebung benutzt, um sowohl asynchrone als auch synchrone Kooperation an verschiedenen Orten durch ein Softwaresystem unterstützen zu können:

> „Die Bearbeitung von Materialien findet in der Arbeitsumgebung statt. Der Benutzer einer Arbeitsumgebung hat den exklusiven Zugriff auf die in der Umgebung vorhandenen Werkzeuge, Automaten und Materialien. Daraus folgt, daß es aus verschiedenen Umgebungen keinen simultanen Zugriff auf Materialien geben kann." ([Gry96], S. 141)

Da das Material nur an einem Ort sein kann, ist es folglich von anderen Orten aus nicht erreichbar. Die Bearbeitung des Materials muß also sequentiell stattfinden. Wird ein Material von einem anderen Benutzer angefordert, so wird ihm dieser Zugriff verwehrt, da es anderweitig in Bearbeitung ist. Er kann jedoch in Abhängigkeit der Arbeitssituation gegebenenfalls eine Kopie des Materials bekommen. Die Regeln, die sich hinsichtlich der Bearbeitung etabliert haben, werden in Form von Prozeßmustern explizit gemacht:

> „Ein Prozeßmuster ist ein gemeinsames Material zur Vergegenständlichung eines kooperativen Arbeitsprozesses. Durch das Prozeßmuster werden Verantwortlichkeiten von Personen oder Rollenträgern und Tätigkeiten in einem kooperativen Arbeitsprozeß festgelegt.[...]" ([Gry96], S. 178)

Da mehrere Personen an dem Arbeitsprozeß beteiligt sind, die sich nicht in demselben Raum aufhalten, befindet sich das Material immer in dem Raum derjenigen Person, die augenblicklich mit der Bearbeitung beschäftigt ist. Sobald diese Person die Arbeit beendet hat, heftet sie das bearbeitete Material in eine Vorgangsmappe und legt es in ihren Postausgangskorb. Transportautomaten sorgen für den Versand dieser Mappe zu derjenigen Person, die als nächstes für die Bearbeitung dieses Materials vorgesehen ist. Dort wird die Vorgangsmappe in den Posteingangskorb gelegt und für die betreffende Person zugreifbar. Diese Prozeßmuster sind flexibel. Dem Bearbeiter ist es möglich, die Bearbeitungsreihenfolge situativ zu verändern. Er kann, sofern er es für nötig erachtet, das Prozeßmuster adaptieren, um auf Ausnahmesituationen, die eine Abänderung des üblichen Ablaufes erzwingen, reagieren zu können. Dem Mitarbeiter wird die flexible Handlungsweise nicht genommen, die ihm, in Anlehnung an das Leitbild für qualifizierte menschliche Tätigkeit, eingeräumt werden sollte.

3.6 Kooperation an einem Ort

Ich habe in dem vorangegangenem Kapitel aufgezeigt, daß die Grenzen der Bereiche innerhalb der Taxonomie bei einer Übertragung auf ein Softwaresystem nicht bestehen bleiben. Ich betrachte im Folgenden einen Bereich von Kooperation, welcher an einem Ort stattfindet und nenne ihn deshalb „Kooperation an einem Ort“. Allerdings möchte ich bestimmte Einschränkungen an diese Bezeichnung knüpfen. Welche das sind, werde ich in Kapitel 3.6.2 zeigen, in dem ich die Merkmale dieser Kooperationsform im einzelnen benennen werde. Grundsätzlich gehe ich bei meinen Betrachtungen davon aus, daß Benutzer den Kooperationsort zu beliebigen Zeiten besuchen können und Tätigkeiten innerhalb ihres Aufgabenbereiches an diesem Ort nachgehen können. Insbesondere hinsichtlich der Koordination des Materialzugriffes sind also adäquate Mechanismen in Abhängigkeit einer konkret betrachteten Kooperationssituation notwendig.

Bevor ich die von mir betrachtete Kooperationsform charakterisiere, möchte ich zunächst ein Beispiel vorstellen. Ich betrachte die Kooperationssituation bei der Pausenplanung in einer Schule anhand zweier Szenarios[15]. Weitere Dokumente des objektorientierten Entwicklungsprozesses, welche die Entwicklung des Pausenplanersystems begleitet haben, finden sich im Anhang dieser Arbeit.

3.6.1 Zwei Beispielszenarios

Die Szenarios betrachten die Kooperationssituation, wie sie ohne Unterstützung durch ein Softwaresystem vorliegt. Mit ihrer Hilfe möchte ich dann die Art und Weise, wie sich Personen in einer solchen Situation koordinieren, herausarbeiten und die Merkmale, durch die diese Form der Kooperation gekennzeichnet ist, definieren.

3.6.1.1 Überblicksszenario des Bereiches Pausenplanung

Die Pausenplanung an der Schule beinhaltet die Teilaufgaben Lehrkörperpflege und Belegen der Pausen mit Aufsichtspersonal. Ein oder mehrere Sekretariatsmitarbeiter sind für die Pflege des Lehrkörpers verantwortlich. Modifikationen durch Veränderungen des Personalstamms sowie Veränderungen der zeitlichen Verfügbarkeit der einzelnen Lehrer werden von den Mitarbeitern im Sekretariat durchgeführt. Die Belegung von Pausen mit Lehrpersonal wird durch einen designierten Pausenplanersteller durchgeführt. Dieser Pausenplanersteller ist verantwortlich für eine konfliktfreie Einteilung der Lehrer. Er muß bei der Einteilung der Aufsichten die Anzahl der Aufsichts-

[15] Das Überblicks- und das Aufgabenszenario werden von Lilienthal und Züllighoven in [LZ96] neben verschiedenen anderen Verfeinerungen des Dokumenttyps „Szenario“ vorgeschlagen.

pflichten der Lehrer berücksichtigen und auf deren zeitliche Verfügbarkeit achten. Die Informationen über das Lehrerpersonal werden in einem Ordner mit der Aufschrift „Lehrpersonal“ gesammelt. Es gibt einen weiteren Ordner mit der Aufschrift „Pausenplanung“, in dem die Informationen über die Aufsichtseinteilungen enthalten sind. Diese Ordner stehen in einem Regal, welches sich in dem Verwaltungsraum der Schule befindet.

3.6.1.2 Aufgabenszenario Lehrkörperpflege

Der Sekretariatsmitarbeiter wird davon in Kenntnis gesetzt, daß sich die zeitliche Verfügbarkeit eines Lehrers verändert hat. Er begibt sich in den Verwaltungsraum, um die Veränderung in dem Ordner „Lehrpersonal“ zu vermerken.

Kurz vorher hat sich der Pausenplanersteller dazu entschlossen, eine Pausenplanung durchführen. Er hat sich zu diesem Zweck in den Verwaltungsraum der Schule begeben, hat sowohl den Ordner „Lehrpersonal“, als auch den Ordner „Pausenplanung“ aus dem Regal genommen und ist in sein Zimmer gegangen. Dort hat er die Ordner auf seinen Tisch gelegt, sie aufgeschlagen und damit begonnen, Lehrer für die Pausenaufsichten einzuteilen.

Als der Sekretariatsmitarbeiter den Verwaltungsraum erreicht, stellt er fest, daß beide erwähnten Ordner nicht im Regal vorhanden sind. Er schließt daraus, daß sie vom Pausenplanersteller zum Zweck der Pausenplanung in Beschlag genommen wurden, da dieser der einzige ist, der diese beiden Ordner zusammen verwendet. Der pflichtbewußte Sekretariatsmitarbeiter begibt sich zu dem Zimmer des Pausenplaners, um ihm die Veränderung möglichst schnell mitzuteilen. Er weiß, daß der Pausenplanersteller durch diese Veränderung möglicherweise einen Teil der Arbeit verwerfen muß, sofern dieser den besagten Lehrer bereits verplant hat.

Der Sekretariatsmitarbeiter betritt den Raum des Pausenplaners. Nach einer Begrüßung verständigen sich die beiden über sein Anliegen. Da der Pausenplanersteller den Lehrer noch nicht verplant hat, bittet er den Sekretariatsmitarbeiter an seinen Tisch und bedeutet ihm, daß er die Änderungen im Ordner „Lehrpersonal“ durchführen kann. Dieses sollte der Sekretariatsmitarbeiter so durchführen, daß der Pausenplanersteller noch weiterhin Einblick in den Ordner hat, um die Informationen, die er für die weitere Durchführung der Pausenplanung benötigt, einsehen zu können. Sobald der Sekretariatsmitarbeiter seine Tätigkeit beendet hat, schiebt er den Ordner wieder in seine ursprüngliche Position zurück. Der Pausenplanersteller bedankt sich für die prompte Mitteilung der Information mit dem Hinweis darauf, daß er andernfalls Mehrarbeit gehabt hätte. Der Sekretariatsmitarbeiter geht zurück in das Sekretariat, während der Pausenplanersteller seine Arbeit fortsetzt.

3.6.1.3 Analyse der Szenarios

Das erste Szenario gibt einen Überblick über die Aufgaben der beteiligten Personen, während das zweite eine konkrete Arbeitssituation beschreibt. Zu letzterem gibt es eine beliebige Anzahl von Variationen. Dieses Szenario schildert lediglich einen möglichen Ablauf. Es fällt zunächst auf, daß die Anzahl der Kooperationspartner sehr gering ist. Diese Tatsache ist der Situation förderlich gewesen, da sich der Sekretariatsmitarbeiter, im Falle mehrerer Pausenplanersteller, möglicherweise nicht die Mühe gemacht hätte, denjenigen zu suchen, welcher zur Zeit die Pausenplanung ausführt.

Eine weitere Erkenntnis aus diesem Szenario ist die Tatsache, daß eine effektive und sinnvolle Kooperation gewünscht ist, da sie den Menschen Mühen und Arbeit erspart. In diesem Szenario hat die prompte Mitteilung der Information durch den Sekretariatsmitarbeiter zu einer Arbeitsersparnis beim Pausenplanersteller geführt.

Weiterhin wird deutlich, daß jeder die Aufgaben und Rollen der beteiligten Personen kennt und aus einer Arbeitssituation heraus Dinge interpretieren kann. Robinson kennzeichnet dieses, wie bereits im Kapitel 3.1 erwähnt, mit dem Begriff der „common artefacts“.[16] Das Regal im Verwaltungsraum kann als ein solches identifiziert werden. Der Sekretariatsmitarbeiter kann aus der Tatsache, daß beide Ordner fehlen, schließen, daß der Pausenplanersteller in seinem Zimmer sitzt und mit der Planung von Pausen beschäftigt ist, obwohl nur das Fehlen der Ordner für ihn offensichtlich ist. Ein Neuling im Sekretariat würde möglicherweise nur das Fehlen bemerken und zum Sekretariat zurückkehren.

Die Erfahrung der beteiligten Personen sorgt ebenfalls dafür, daß sie wissen, wo sich die Ordner befinden. Allen ist sowohl der Aufbewahrungsort der Ordner bekannt, als auch der mögliche Aufenthaltsort, sofern sie nicht im Regal des Verwaltungsraumes vorhanden sind.

Bei den Vorfällen im Zimmer des Pausenplaners fällt auf, daß die Koordination der beteiligten Personen über explizite Kommunikation stattfindet. Die beiden Personen sprechen über das Anliegen und vereinbaren daraufhin eine Strategie, die situativ angemessen erscheint. Neben dieser expliziten Kommunikation kann ebenfalls eine implizite stattfinden. Ein Beispiel dafür zeigt sich in der Tatsache, daß der Pausenplanersteller aus dem Zurückschieben des Ordners in seine ursprüngliche Situation schließen kann, daß der Sekretariatsmitarbeiter seine Arbeit beendet hat.

[16] Vergleiche dazu [Rob93].

Schließlich ist die Art und Weise der Bearbeitung interessant. Beide Personen haben, zumindest teilweise, identisches Material simultan bearbeitet.[17] Ebenso hätte der Pausenplanersteller seine Tätigkeit kurz unterbrechen können, da er weiß, daß die Änderungen des Sekretariatsmitarbeiters üblicherweise sehr schnell ausgeführt werden. Alternativ dazu hätte der Pausenplanersteller auch zunächst seine Arbeit beenden können. Allerdings würde dieses, wie bereits angedeutet, gegebenenfalls Mehrarbeit für ihn bedeuten.

3.6.2 Merkmale der Kooperation an einem Ort

Basierend auf dieser Analyse sowie Erkenntnissen aus der CSCW Forschung werde ich nun die Merkmale der Kooperation mehrerer Personen an einem Ort im einzelnen aufzeigen.

- Die Anzahl der Personen ist üblicherweise eher gering.

  Es wird von Kirsche in [Kir94] aufgezeigt und hier unterstrichen, daß Kooperation mehrerer Personen an einem Ort üblicherweise nur mit einer kleinen Anzahl an Kooperationspartnern sinnvoll zu realisieren ist.

- Die Kooperationspartner sind sich bekannt.

  Die Personen, die an einer Kooperation beteiligt sind, kennen sich oder die Rollen, die sie im Rahmen der Gruppenarbeit innehaben und wissen um die Verantwortlichkeiten und Aufgaben der einzelnen Kooperationspartner.

- Der Ort spielt eine zentrale Rolle bei dieser Kooperation.

  Dieser Ort muß den Kooperationspartnern bekannt sein. Weiterhin ist es sinnvoll, daß dieser Ort für alle Beteiligten erreichbar ist.

- Der Zeitpunkt der Partizipation einer Person ist unbestimmt.

  Die beteiligten Personen begeben sich zu beliebigen Zeitpunkten an den Ort der Kooperation und erledigen dort ihre Aufgaben.

- Alltägliche Gegenstände unterstützen Koordination.

[17] Ich verstehe in diesem Falle unter Bearbeitung sowohl das Verändern als auch das Sondieren des Materials. Vergleiche dazu das Kapitel 2.3.2 „Die Entwurfsmetapher Werkzeug“.

Die Beachtung von alltäglichen Gegenständen vermitteln den Kooperationspartnern wesentlich mehr Informationen, als es auf den ersten Blick den Anschein haben kann. Darüber hinaus ermöglichen sie den Beteiligten eine sukzessive Verbesserung der Koordination, indem die Kooperationspartner in wiederkehrenden Situationen wiederkehrende Verhaltensweisen an den Tag legen.

- Es gibt wenige Regeln zur Koordination

  Die Koordination der beteiligten Personen wird nur wenig über festgelegte Regeln definiert. Vielmehr wird Koordination über explizite und implizite Kommunikation im Laufe der Zeit herausgebildet. Allerdings möchte ich anmerken, daß Koordination durch implizite Kommunikation eine sehr starke Ähnlichkeit zu Regeln aufweist. So bedeutet das Fehlen der beiden Ordner „in der Regel“, daß die Ordner beim Pausenplanersteller sind.

- Flexibles Handeln ist ausgesprochen wichtig für Kooperation an einem Ort

  Die Kooperationspartner werden sich immer wieder in einer neuen Situation wiederfinden. Diese Tatsache wurde bereits von Robinson in seinem Artikel über die unvorhersehbare Benutzung festgestellt. Vordefinierte Arbeitsabläufe sind in dieser Form der Kooperation ausgesprochen selten und auch nicht erwünscht, da sie die Flexibilität der beteiligten Personen stark mindern.

- Simultaner Zugriff auf die Arbeitsgegenstände ist durchaus möglich.

  Der Ort der Arbeitsgegenstände ist bekannt. Alle Beteiligten können die Materialien bearbeiten. Sie können sich dazu entschließen, nacheinander oder simultan Material zu bearbeiten.

- Transparenz muß gewährleistet sein.

  Ein hohes Maß an Transparenz ist für die beteiligten Personen notwendig. Nur so sind sie in der Lage, flexibel reagieren zu können und effektiv zusammenzuarbeiten. Transparenz wird durch explizites Signalisieren oder durch Erfahrung mit alltäglichen Dingen erreicht.

Ich werde nun anhand dieser Kennzeichen ein Konzept vorstellen, welches versucht, diese Eigenschaften bei der Übertragung der Kooperation auf Softwaresysteme zu beachten.

3.7 Kooperation in der Gruppenarbeitsumgebung

Menschen arbeiten üblicherweise an ihrem Arbeitsplatz. Eine Definition des Arbeitsplatzes lautet wie folgt:

> „1. Der Ort, an dem eine Tätigkeit verrichtet wird, einschließlich der Einrichtung (Maschinen, Geräte, Möbel).
>
> 2. Der abstrakte Tätigkeitsbereich als Zusammenfassung der Funktionen eines Beschäftigten und der Anforderungen, die an ihn gestellt werden.“ [Mey95]

Diese Definition konzentriert sich auf zwei Aspekte. Der eine bezieht sich konkret auf den Ort, an dem der Mensch Arbeit verrichtet, während der zweite sich auf die mit der Stelle, die eine Person bekleidet, verbundenen Aufgaben bezieht. Ich habe gezeigt, daß beide Aspekte für das gegenseitige Verständnis und die Kooperationsfähigkeit relevant sind. Jedoch macht diese Definition keine Aussage über die Person oder die Personen, die an diesem Ort Arbeit verrichten. Gryczan benutzt den Begriff der Arbeitsumgebung, um den Ort, an dem ein Mensch Arbeit verrichtet, zu definieren:

> „Eine Arbeitsumgebung zur Unterstützung qualifizierter menschlicher Tätigkeiten ist der Ort für eine anwendungsfachlich motivierte Zusammenstellung von Werkzeugen, Automaten und Materialien. Durch die Arbeitsumgebung werden keine Reihenfolgebedingungen für die Verwendung von Werkzeugen und Arbeitsmitteln festgelegt.“ ([Gry96], S. 138)

Der Mensch kann sich seinen Arbeitsplatz nach seinem eigenen Ermessen einrichten. Er ist nicht gebunden an Vorschriften zur Benutzung von Werkzeugen und Materialien. Ihm wird kein Ablauf vorgeschrieben. Gryczan geht davon aus, daß ein Mensch in seiner Arbeitsumgebung tätig ist. Kooperierende Personen haben jeweils eine eigene Umgebung. Kooperation findet über Wege zwischen diesen Umgebungen statt. Die angeführten Definitionen des Arbeitsplatzes und der Arbeitsumgebung beziehen sich auf einzelne Personen.

Ich möchte die Sicht auf den Arbeitsplatz von Menschen um die Möglichkeit erweitern, daß mehrere Personen in einer Umgebung tätig sein können. Als ein weiteres Beispiel möchte ich hier die Situation in einer Kfz-Werkstatt anführen.

Mehrere Mechaniker arbeiten gemeinsam an einem Auto, um es wieder in Stand zu setzen. Das Werkzeug ist in dieser Werkstatt in Werkzeugkisten vorhanden oder hängt verteilt an der Wand. Das Material ist das Auto, welches in der Mitte der Werkstatt auf einer Hebebühne steht. Der Arbeitsplatz der Mechaniker ist die Werkstatt. Es wird den Mechanikern nicht vorgeschrieben, wie

sie ihre Arbeit zu verrichten haben. Die Zusammenarbeit der Mechaniker ist dadurch gekennzeichnet, daß sie sich sehr häufig koordinieren müssen. In Abhängigkeit der Schäden an dem Auto, welche sie erst sukzessive bei der Untersuchung und Reparatur feststellen, müssen sie vereinbaren, wer welche Teile bearbeitet, oder ob sie gemeinsam an einer Schadensbehebung arbeiten. Natürlich kann es Interdependenzen zwischen der Werkstatt und anderen Arbeitsumgebungen geben. So muß ein Auto, nachdem es repariert wurde, in die Lackierhalle gebracht werden, um neu lackiert zu werden. Das Material Auto wechselt von einer Arbeitsumgebung in eine andere.

Im Szenario aus Kapitel 3.6.1.2 arbeiten ebenfalls zwei Personen an einem Ort. Dieser Ort ist das Zimmer des Pausenplaners, in das sich der Sekretariatsmitarbeiter begeben hat. Personen, welche im Rahmen einer Arbeitsgruppe tätig sind, können durch ein Softwaresystem dahingehend unterstützt werden, daß ihnen eine Gruppenarbeitsumgebung zur Verfügung gestellt wird, zu der sie sich begeben können, um dort ihren Tätigkeiten nachzukommen. Ich unterstütze dabei die Ansicht von Fitzgerald et al., welche die Notwendigkeit eines für die Gruppenarbeit relevanten Ortes fordern. Dieser Ort hat eine Bedeutung. Es ist der ausgesuchte Ort, an dem die Kooperation mehrerer Personen zu einem bestimmten Zweck, nämlich der Erfüllung der an die Gruppe gestellten Aufgabe, stattfindet. Personen begeben sich zu dem Zweck der Kooperation an diesen Ort. Sie finden dort Materialien und Werkzeuge vor, anhand derer sie ihre Tätigkeiten ausführen. Dieser Gedanke wurde ebenfalls von Gutwin und Greenberg als „shared workspace“ eingeführt.[18] Eine Person verfügt somit neben ihrem privaten Arbeitsplatz über die Möglichkeit, sich an andere Orte zu begeben. Ist sie Mitglied in verschiedenen Arbeitsgruppen, so kann sie von ihrem Arbeitsplatz aus die Orte, die für verschiedene Gruppenarbeiten existieren, erreichen.

Ich möchte den Ort, an dem mehrere Personen zusammenarbeiten, mit dem Begriff der Gruppenarbeitsumgebung definieren.

> Eine **Gruppenarbeitsumgebung** ist ein Ort, an dem Mitglieder einer Arbeitsgruppe zusammenarbeiten. Sie ist von allen beteiligten Personen zu beliebigen Zeitpunkten erreichbar und enthält alle für die Aufgaben der Gruppe relevanten Werkzeuge, Automaten und Materialien. Die Zusammenarbeit der Personen ist durch ein hohes Maß an Flexibilität gekennzeichnet.

Für die Gruppenarbeitsumgebung gilt, daß keine Reihenfolgebedingungen für die Verwendung der Arbeitsmittel festgelegt werden. Sie zeichnet sich dadurch aus, daß nur Personen in diese Umgebung gelangen, die Teilnehmer an dem Kooperationsprozeß sind. Wichtig ist dabei zu vermerken,

[18] Vergleiche dazu [GG97a], S. 2.

daß Kooperationen auch zwischen Gruppenarbeitsumgebungen durchaus möglich sind. Dieses habe ich bereits im Beispiel der Kfz-Werkstatt aufgezeigt, in dem die Mechaniker mit den Lackierern der Lackierhalle kooperieren.

Die Gruppenarbeitsumgebung ist ein Ort, der von verschiedenen Personen erreichbar ist. Der Arbeitsplatz einer Person, welcher nur ihr persönlich zugänglich ist, wird folglich um eine öffentliche Komponente erweitert.

Wie aus den verschiedenen Beispielen bereits hervorgeht, eignet sich die Gruppenarbeitsumgebung vorwiegend für die Zusammenarbeit innerhalb kleinerer Gruppen. Es sollten nur so viele Personen sein, daß die verschiedenen Aspekte der Zusammenarbeit, wie Transparenz und Koordinationsfähigkeit durch implizite und explizite Kommunikation, nicht gefährdet sind. Wird die Anzahl der beteiligten Personen zu groß, so ist eine sinnvolle Kooperation und damit eine effiziente Zusammenarbeit nicht mehr möglich. Es würde sicherlich wenig Sinn machen, die Zusammenarbeit vieler Angestellter innerhalb eines Krankenhauses mit Hilfe einer Gruppenarbeitsumgebung zu modellieren, da die Anzahl der beteiligten Personen so hoch ist, daß eine sinnvolle Koordination ausgeschlossen ist.

Ich habe bereits festgestellt, daß die Koordinationsnotwendigkeit bei der Kooperation an einem Ort ausgesprochen hoch ist. Diese Notwendigkeit wird beeinflußt durch Faktoren wie Transparenz und Materialverfügbarkeit. Gesteuert wird die Koordination im wesentlichen durch die Kommunikation zwischen den beteiligten Personen sowie deren Erfahrungswerte. Die Mechaniker werden den Zustand des Autos sondieren und daraufhin entscheiden, was zu tun ist. Ebenso werden sie entscheiden, wer was zu tun hat und in welcher Reihenfolge sie vorgehen werden. Diese Strategie werden sie möglicherweise bei der Feststellung weiterer Schäden revidieren müssen.

Die Fähigkeit, sich im Rahmen der Zusammenarbeit flexibel auf veränderte Umstände einstellen zu können, ist notwendig. Arbeiten mehrere Mechaniker an dem Auto, so werden sie entweder gemeinsam einen Teil des Autos, wie beispielsweise den Motor, bearbeiten, oder sie werden ihre Arbeit so koordinieren, daß jeder verschiedene Teile bearbeitet, um Konfliktsituationen aus dem Weg zu gehen. In dieser Situation spielt ebenso die Verfügbarkeit der Werkzeuge eine große Rolle. Es würde beispielsweise wenig Sinn machen, den Arbeitsablauf so festzulegen, daß zwei Mechaniker ein Schweißgerät benötigen, obwohl nur eines vorhanden ist.

Im Rahmen der Arbeit innerhalb einer Gruppenarbeitsumgebung ist es wichtig, daß ein hohes Maß an Transparenz gewährleistet wird. Die Benutzer werden vom System über die Präsenz und die Aktivitäten anderer Gruppenmitglieder informiert. Neben dieser Information wäre es je nach Koordinationsbedarf möglich, eine direkte Kommunikation zwischen den Anwesenden zu

integrieren. Robinson sieht diesen Punkt als sehr wichtig an, da gerade die Kombination aus direkter Kommunikation und intuitivem Verständnis der Situation sich in einem Prozeß niederschlägt, der die Koordination fördert. Aus der Fähigkeit der Menschen, eine aktuelle Situation diskutieren zu können, entstehen die Erkenntnisse, die wiederum dazu führen, daß eine Interpretation von späteren Situationen möglich ist. Er formuliert diesen Gedanken mit dem Begriff der „double level language“:

> „‘Double level language‘ is a phrase intended to catch the idea, that implicit, often indirect communication (through artefacts) and explicit communication (speech, ad hoc notes) are not alternatives, but complementary and mutually supportive.“ ([Rob93], S. 196)

Die Koordination anhand des Materials, wie sie von Robinson und Sørgaard bereits erkannt wurde, spielt in dieser Arbeitsumgebung eine große Rolle. Ich hatte bereits bei der Einführung der Kooperationsform Ansätze von Greif, Olson et al. und Stefik et al. erwähnt, die auf verschiedene Arten und Weisen den Zugriff auf Material koordinieren. Ich bin der Meinung, daß Material, im Rahmen der hier betrachteten Kooperationsform, prinzipiell für alle Beteiligten jederzeit zugreifbar sein muß, sofern nicht fachliche Gründe, wie zum Beispiel Sicherheitsaspekte oder Kompetenzfragen, dafür verantwortlich sind, den Zugriff auf ein Material nur einem Teil von Personen zu ermöglichen. Material kann immer nur an einem Ort sein. Da es von diesem Ort, im Rahmen der betrachteten Kooperationsform, nicht entfernt werden darf, kann jeder Mensch dieses Material sehen. Den gleichzeitigen Zugriff auf ein solches Material per se zu verhindern, wäre meines Erachtens nicht vertretbar. Die Koordination über den Zugriff auf dieses Material kann nicht durch ein Softwaresystem antizipiert werden. Vielmehr bleibt es den Mitarbeitenden überlassen, ihre eigene Form der Kooperation herauszubilden. Ob es innerhalb des Systems eine Komponente gibt, welche den Zugriff auf Materialien koordiniert und welches Konzept dabei letztlich Verwendung findet, hängt von der konkreten Kooperationssituation ab. Grundsätzlich sollten anwendungsfachliche Aspekte darüber entscheiden, wie der konkurrierende Zugriff auf das Material gehandhabt wird. Ich bin der Meinung, daß sogar der simultane Zugriff auf gemeinsame Materialien erlaubt sein kann, sofern es die Kooperationssituation angemessen erscheinen läßt. Allerdings setzt dieses ein adäquates Maß an Transparenz für alle beteiligten Personen voraus. Es sollte dann dem Menschen überlassen bleiben, in einer konkreten Arbeitssituation zu entscheiden, ob eine gleichzeitige Bearbeitung des Materials sinnvoll ist, oder nicht. Dieses hängt stark von der Kooperationssituation und dem Arbeitszusammenhang ab. Die Entscheidung, welche Art des Materialzugriffes fachlich möglich, erlaubt und sinnvoll ist, muß immer wieder neu getroffen werden.

3.8 Zusammenfassung und Ausblick

In diesem Kapitel habe ich versucht, dem Leser einen Einblick in den heutigen Stand der CSCW Forschung zu geben. Ich habe auf die Probleme hingewiesen und aufgezeigt, welche Lösungsansätze existieren. Weiterhin habe ich das Konzept der Arbeitsumgebung nach Gryczan vorgestellt, welches es ermöglicht, daß Menschen im Rahmen einer Zusammenarbeit an verschiedenen Orten kooperieren können. Darüber hinaus habe ich die Kooperation an einem Ort fachlich motiviert und versucht, die Merkmale dieser Kooperationsform zu bestimmen. Ich habe den Begriff der Gruppenarbeitsumgebung entwickelt, welcher die Kooperation von mehreren Menschen an einem, durch ein Softwaresystem bereitgestellten, virtuellen Ort im Sinne der Definition des Begriffes ermöglicht.

Im bisherigen Verlauf meiner Untersuchungen habe ich aus den Erkenntnissen über die Zusammenarbeit an einem Ort ein fachliches Konzept zur Unterstützung durch Softwaresysteme abgeleitet. Im weiteren Verlauf der Arbeit werde ich zeigen, welche Anforderungen technischer Art daran geknüpft sind. Einem Vorschlag einer technischen Realisierung, welche diesen Anforderungen gerecht wird, folgt die Vorstellung des Pausenplanersystems. Dieses System wurde von Tim Krauß und mir entwickelt, um, unter anderem, zu zeigen, daß sich das vorgestellte Konzept auch in der Praxis einsetzen läßt.

4 Software für Kooperation an einem Ort aus technischer Sicht

In diesem Kapitel formuliere ich, basierend auf den Erkenntnissen hinsichtlich der Gruppenarbeitsumgebung, ein Reihe von technischen Anforderungen an ein solches kooperationsunterstützendes System und zeige daran anschließend auf, wie eine technische Interpretation dieses Begriffs unter Berücksichtigung der vorgenannten Anforderungen und Probleme aussehen kann.

Um dem Leser einen Eindruck von der Arbeit mit einem solchen Softwaresystem zu geben, schildere ich anhand eines Beispiels die Rahmenbedingungen in Form zweier Systemvisionen. Sie orientieren sich an den Szenarios aus Kapitel 3.6.1. Aus diesen Visionen kristallisieren sich unmittelbar verschiedene Anforderungen an Softwaresysteme heraus, welche die betrachtete Form der Kooperation unterstützen sollen.

4.1 Visionen des Umganges mit einem Pausenplanersystem

Die sehr kurz gehaltene Überblicksvision verdeutlicht, wie die Personen an der Pausenplanung beteiligt sind.[19] Die Handhabungsvision verdeutlicht die Zusammenarbeit des Sekretariatsmitarbeiters und des Pausenplanerstellers.

4.1.1 Eine Überblicksvision

Das Pausenplanersystem unterstützt die mit der Pausenplanung beauftragten Personen. Jede Person, die an der Pausenplanung mitarbeitet, verfügt über einen Computer. Diese Computer, auf denen das System installiert ist, sind miteinander durch ein Netzwerk verbunden. Die Personen arbeiten an ihrem Arbeitsplatz und müssen diesen für die Tätigkeiten, die mit der Pausenplanung verbunden sind, nicht verlassen. Sie führen sie statt dessen mit Hilfe des Computers aus. Zu diesem Zweck können die relevanten Informationen zur Pflege des Lehrkörpers und der Pausenbeaufsichtigungen, welche bisher in den Ordnern „Lehrpersonal“ und „Pausenplanung“ aufbewahrt wurden, nun über das Softwaresystem zugegriffen werden. Den Benutzern werden für die jeweilige Tätigkeiten (Änderungen im Lehrerstamm, Veränderung von Pauseneinteilungen u.ä.) adäquate Werkzeuge mit grafischer Oberfläche angeboten, welche ein übersichtliches und effizientes Arbeiten ermöglichen. Das System bietet die Möglichkeit an, verschiedene Versionen der jeweiligen Arbeitsmittel (Lehrkörper, Pausenplan) verwalten zu können. Dadurch sind die Benutzer beispielsweise in der

[19] Insbesondere die Überblicksvision ist sehr kurz gehalten, da sich eine ausführlichere Version im Anhang dieser Arbeit befindet. Nichtsdestotrotz möchte ich sie an dieser Stelle aufführen, um dem Leser die Anforderungen an kooperationsunterstützende Softwaresysteme verdeutlichen zu können.

Lage, eine Testbelegung durchführen zu können. Langfristig wäre eine Erweiterung um einen Automaten, welcher eine automatische Einteilung von Beaufsichtigungen vorschlägt, denkbar.

4.1.2 Eine Handhabungsvision

Möchte der Sekretariatsmitarbeiter Herr Müller eine Veränderung am Lehrkörper durchführen, so startet er auf seinem Computer das Pausenplanersystem, indem er mit der Maus ein Symbol auf seinem Bildschirm anklickt. Das System fordert ihn zur Eingabe seines Namens auf. Dem Benutzer werden danach auf seinem Bildschirm die Arbeitsmittel präsentiert, welche sich in der Gruppenarbeitsumgebung „Pausenplanung“ befinden. Als Materialien kann Herr Müller verschiedene Lehrkörper und Pausenpläne erkennen. Linien, welche jeweils zwei Symbole miteinander verbinden, repräsentieren die Verknüpfung eines Lehrkörpers mit einem Pausenplan. In einer Leiste am oberen Rand der Darstellung der Gruppenarbeitsumgebung steht: „Anwesend sind Schmidt und Müller“. Herr Müller weiß nun, daß sich der Pausenplanersteller Herr Schmidt ebenfalls in der Umgebung befindet. Neben dem Symbol eines Pausenplans und dem eines Lehrkörpers taucht der Name Schmidt ebenfalls auf. Der Sekretariatsmitarbeiter kann daraus ersehen, daß Herr Schmidt diese beiden Materialien bearbeitet. Da beide gemeinsam für die Pausenplanung benötigt werden, schließt er daraus, daß Herr Schmidt gegenwärtig mit der Einteilung von Pausen beschäftigt ist.

Herr Schmidt wird seinerseits durch das System darauf hingewiesen, daß sich Herr Müller soeben in der Gruppenarbeitsumgebung „Pausenplanung“ eingefunden hat. Da Herr Schmidt die Aufgaben von Herrn Müller im Rahmen der Pausenplanung kennt, weiß er nun, daß Herr Müller eine Änderung am Lehrkörper durchführen wird. Neben den Materialien befinden sich in der Umgebung eine Reihe von Werkzeugen, die ebenfalls durch Symbole dargestellt werden. Der Sekretariatsmitarbeiter kann nun die gewünschte Bearbeitung des Lehrkörpers mit dem Werkzeug *Lehrkörperbearbeiter* starten, indem er mittels der Maus das betreffende Werkzeugsymbol auf das Symbol für den Lehrkörper fallen läßt. Es öffnet sich das Werkzeug und er kann mit der Bearbeitung beginnen. Das Materialsymbol ist nun auch mit seinem Namen versehen. Dem Pausenplanersteller wird dadurch verdeutlicht, daß der Sekretariatsmitarbeiter augenblicklich den Lehrkörper bearbeitet. Nachdem der Sekretariatsmitarbeiter die zeitliche Verfügbarkeit des Lehrers Meier verändert hat, schließt er das Werkzeug wieder. Er verläßt die Gruppenarbeitsumgebung, indem er auf einen Knopf mit der Aufschrift „Verlassen“ drückt.

Das System weist Herrn Schmidt darauf hin, daß Herr Müller die Umgebung verlassen hat. Außerdem stellt Herr Schmidt fest, daß eine Pauseneinteilung auf der Oberfläche des Werkzeugs *Pausenplaner* die Farbe gewechselt hat und nun rot gefärbt ist. Er weiß, daß der Sekretariatsmitar-

beiter Herr Müller soeben den Lehrkörper bearbeitet hat. Daher wird ihm klar, daß Herr Müller die zeitliche Verfügbarkeit von Lehrer Meier verändert hat. Er tauscht nun die Pauseneinteilung von Meier mit der von Lehrer Schulze. Nun ist keine rot gefärbte Belegung mehr zu sehen. Der Pausenplan ist konfliktfrei. Der Pausenplanersteller Herr Schmidt drückt zufrieden auf den Knopf mit der Aufschrift „Drucken“, nimmt den gedruckten Pausenplan aus dem Drucker und hängt ihn im Lehrerzimmer auf.

4.1.3 Diskussion der Visionen

Die genannten Personen arbeiten an zwei voneinander entfernten physischen Orten. Sie begeben sich mit Hilfe des Softwaresystems an einen virtuellen Ort. Dieser Ort wird ihnen mit Hilfe eines Werkzeuges präsentiert. An diesem Ort werden den Benutzern Werkzeuge und Materialien zur Verfügung gestellt, mit denen sie ihre Aufgaben im Rahmen der Pausenplanung bewältigen können.

Es wird deutlich, daß sich die zwei Personen Herr Müller und Herr Schmidt gegenseitig bekannt sind und sie um ihre Aufgabenbereiche und Verantwortlichkeiten wissen. Es werden verschiedene Awareness-Informationen zur Verfügung gestellt. Die Personen wissen, wer anwesend ist und können darüber hinaus erkennen, wer womit beschäftigt ist. Es lassen sich Ähnlichkeiten zu dem Szenarios aus Kapitel 3.6.1 feststellen: wiederum kann der Sekretariatsmitarbeiter Herr Müller an der aktuellen Bearbeitungssituation erkennen, daß Herr Schmidt gegenwärtig mit der Pausenplanung beschäftigt ist.

Die Visionen zeigen deutlich, daß es im System keine regelnde Koordinationskomponente für den konkurrierenden Materialzugriff gibt. Herr Müller kann den Lehrkörper bearbeiten, obwohl dieser zu dem Zeitpunkt bereits von Herrn Schmidt bearbeitet wird. Aufgrund der Tatsache, daß Herr Müller den Lehrkörper zur Pausenplanung lediglich „lesend“ benutzt und ihm die dem Lehrkörper beigebrachten Veränderungen unmittelbar sichtbar sind, erscheint diese Lösung plausibel. Es wurde bereits in der Diskussion der Szenarios deutlich, daß diese Herangehensweise letztlich zu einer Arbeitserleichterung für den Pausenplaner führen kann. Die Koordination ist folglich außerhalb des Systems angesiedelt. In den Szenarios verständigen sich die beiden Personen über die auszuführenden Tätigkeiten, indem sie sich unterhalten. Es wäre also sicherlich denkbar, daß Herr Schmidt Herrn Müller anruft, um ihn von der bevorstehenden Änderung in Kenntnis zu setzen.

Aus diesen Erkenntnissen heraus werden ich nun die Anforderungen sowie die notwendigen Mechanismen, welche für ein solches System relevant sind, im einzelnen benennen und diskutie-

ren. Sie müssen in einem System, welches die betrachtete Form der Kooperation unterstützen soll, ausreichend Berücksichtigung finden.

4.2 Anforderungen

Ich werde nun verschiedene Anforderungen an Softwaresysteme zur Unterstützung von Kooperation an einem Ort im einzelnen benennen und motivieren. Diese Anforderungen werden im weiteren Verlauf der Arbeit in die technische Interpretation der Gruppenarbeitsumgebung mit einfließen.

4.2.1 Materialverwaltung

Material, als ein zentrales Element in einem Softwaresystem, muß persistent sein. Ein Mensch, der einen Ordner auf seinem Schreibtisch ablegt und sein Büro verläßt, darf erwarten, daß sich dieser Ordner immer noch dort befindet, wenn er sein Büro wieder betritt. Dieses trifft ebenfalls auf Material innerhalb von Softwaresystemen zu. So müssen die in der Vision geschilderten Änderungen beim nächsten Systemstart natürlich immer noch im System vorhanden sein. Folglich müssen Materialien in einen persistenten Zustand überführt werden, bevor das Softwaresystem beendet wird. Ebenfalls müssen die Materialien von dort auf Anfrage wieder rekonstruiert werden können.

Bei dem betrachteten Material handelt es sich um gemeinsam genutztes Material. Die Benutzer haben alle die Möglichkeit, auf dieses Material zuzugreifen. Al Folge davon muß eine geeignete Strategie für die Koordination gewählt werden. Fachliche Aspekte, die den Zugriff auf Materialien einschränken, müssen gegebenenfalls in das System integriert werden. Weiterhin sind Mechanismen notwendig, die dafür sorgen, daß allen Benutzern der gleiche, aktuelle Zustand dieser Materialien präsentiert wird. Ebenfalls bedarf es geeigneter Mechanismen, welche für die Konsistenz der Materialien sorgen. Ich werde sie in den nachfolgenden Kapiteln aufgreifen.

4.2.2 Materialkonsistenz

Die Veränderung von Materialien kann möglicherweise dazu führen, daß andere, abhängige Materialien ihren Zustand ändern. Dem abhängigen Material ist diese Änderung allerdings nicht bekannt, sofern es ihm nicht explizit mitgeteilt wird. Um die Problematik der Materialkonsistenz zu verdeutlichen, möchte ich hier ein weiteres Beispiel aus dem Pausenplanersystem anführen.

Eine Pausenplanung ist genau dann erfolgreich durchgeführt, wenn für alle Pausen in ausreichender Anzahl Beaufsichtiger eingeteilt sind und keine Konflikte aus der eingeschränkten Verfügbarkeit des Lehrpersonals resultieren. Kündigt ein Lehrer seine Stelle, so löscht ein Sekretariatsmitarbeiter diesen aus dem Lehrkörper. Der Pausenplan enthält nun Einteilungen eines Lehrers, welchen es an der Schule nicht mehr gibt. Er ist inkonsistent geworden. Es ist notwendig, daß das Material

Pausenplan wieder in einen konsistenten Zustand gebracht wird indem die ungültig gewordenen Belegungen entfernt werden. Ein weiteres Beispiel ist in obiger Vision erkennbar. Die vorgenommene Änderung durch den Sekretariatsmitarbeiter führt dazu, daß der Pausenplan nicht mehr konfliktfrei ist. Diese Tatsache muß dem Pausenplan selbst mitgeteilt werden.

Ist also ein Material von anderen Materialien abhängig, so muß ein Mechanismus dafür sorgen, daß dieses wieder in einen konsistenten Zustand gebracht wird. Durch das Löschen anderer Materialien kann es sich in einem Zustand befinden, in dem die Konsistenz nicht mehr gewährleistet ist. Durch die Veränderung eines anderen Materials kann es seinen Zustand verändern, ohne es zu merken. Insbesondere der letzte Fall ist sehr stark von der Art und Weise abhängig, wie die Materialien intern aufgebaut sind. Ziel eines geeigneten Konzeptes für die Gewährleistung von Materialkonsistenz sollte jedoch sein, daß außerhalb der Materialien keine Annahmen über deren inneren Aufbau gemacht werden sollten. Ich stelle im Folgenden drei Ansätze vor, bei denen auf unterschiedliche Art und Weise die Konsistenz der Materialien gehandhabt wird.

4.2.2.1 Materialkonsistenz durch Werkzeuge

Ein naheliegender Ansatz ist möglicherweise, daß das Werkzeug, welches ein Material bearbeitet, abhängige Materialien von Veränderungen in Kenntnis setzt. Jedoch ist es im konkreten Fall häufig so, daß das Werkzeug die betroffenen Materialien überhaupt nicht kennt beziehungsweise nicht kennen sollte und es fachlich nicht motivierbar ist, warum sich dieses aufgrund der vorgestellten Problematik ändern sollte. Beispielsweise erscheint es wenig sinnvoll, daß das Werkzeug Lehrkörperbearbeiter das Material Pausenplan kennt, um ihm im Falle der Veränderung eines Lehrkörpers eine Aufforderung zur Aktualisierung zukommen zu lassen. Im Rahmen des WAM Ansatzes kennt überdies das Werkzeug ohnehin den konkreten Typ des Materials nicht, da es lediglich unter dem Bearbeitungsaspekt bekannt ist.

4.2.2.2 Materialkonsistenz durch Materialien

Eine andere Möglichkeit, die Materialkonsistenz zu gewährleisten, besteht darin, daß Materialien selbständig abhängige Materialien von Veränderungen in Kenntnis setzen. So könnte in vorgenanntem Beispiel der Lehrkörper dem Pausenplan eine Nachricht senden mit dem Hinweis darauf, daß ein Lehrer entfernt wurde. Als Folge davon würden die Werkzeuge mit Konsistenzproblemen der Materialien nicht belastet. Realisieren ließe sich dieses mit einem Beobachtungsmechanismus zwischen den betroffenen Materialien. Allerdings erfordert dieser Ansatz darüber hinaus, daß Werkzeuge von der Veränderung abhängiger Materialien informiert werden, sofern sie dieselben zur Zeit der Veränderung bearbeiten. Das grundsätzliche Problem dieses Lösungsansatzes besteht darin, daß die Materialien den Kontext der Veränderungen nicht kennen. Um eine vollständige

Konsistenz gewährleisten zu können, müssen diese Nachrichten zwischen den Materialien bei jeder Veränderung versandt werden, auch wenn es für den momentanen Bearbeitungszustand nicht erforderlich ist. Das System würde im Extremfall mit einer Fülle von Nachrichten über Materialveränderungen überflutet werden.

4.2.2.3 Materialkonsistenz durch einen Konsistenzverwalter

Eine weitere Lösung, die auch hier gewählt wurde, sieht eine Instanz im System vor, welche die Abhängigkeiten zwischen Materialien kennt und die von Materialveränderungen in Kenntnis gesetzt wird. Sie kann dann diese Nachricht an die betroffenen Materialien delegieren. Diese Information muß dem abhängigen Material verzögerungsfrei zugehen, um zu verhindern, daß es in einem inkonsistenten Zustand benutzt wird. Zu beachten ist dabei die Tatsache, daß aus einzelnen Abhängigkeiten zwischen Materialien Abhängigkeitsketten entstehen können. So würde im Falle des Pausenplanersystems die Veränderung des Ausschlußzeitraumes eines Lehrers die Veränderung des Lehrers, damit die Veränderung des Lehrkörpers und letztlich damit die Veränderung des Pausenplanes bedeuten. Die Tragweite dieser Veränderung ist letztlich von der Art und Weise abhängig, wie Materialien konstruiert und zueinander in Beziehung gesetzt werden. Wie bereits erwähnt, darf ein solcher Mechanismus keinerlei Annahmen über den inneren Aufbau von Materialien machen. Ich halte an dieser Stelle lediglich fest, daß für die Materialkonsistenz gesorgt werden muß. Die Beschreibung der konkreten Lösung findet sich im Kapitel 8.2.2.4 „Die Konsistenzverwaltung“. Durch diesen Lösungsansatz wird einerseits dafür gesorgt, daß Werkzeuge nicht damit belastet werden, Materialien „mit zu bearbeiten“, die sie nicht bearbeiten sollten. Andererseits wird dem Problem der Materialveränderung ohne Kontextwissen ebenfalls aus dem Weg gegangen. Insofern erscheint diese Lösung eleganter als die beiden vorgenannten. Es ist allerdings anzunehmen, daß die hier vorgeschlagene Lösung nur im begrenztem Maße nutzbar ist. Übersteigt die Menge und Vielfalt der Materialien eine augenblicklich unbekannte Grenze, so wird der Verwaltungsaufwand aller Wahrscheinlichkeit nach zu einem Engpaß werden. Für den Extremfall von Massendaten wird sie gänzlich ungeeignet sein.

4.2.3 Werkzeugkonsistenz

Werkzeuge erfragen Materialien bei dem Materialverwalter. Verändern sie diese Materialien, so müssen alle anderen Werkzeuge, welche ebenfalls diese Materialien bearbeiten, auf die Veränderung hingewiesen werden, damit sie die Informationen, die sie dem Benutzer präsentieren, aktualisieren können. Wiederum dient hier die Handhabungsvision als Beispiel. Das Werkzeug Pausenplaner muß die Veränderung des Lehrkörpers berücksichtigen, da eine Pauseneinteilung

nach der Veränderung als konfliktbehaftet gekennzeichnet werden muß. Es wird also ein Benachrichtigungsmechanismus benötigt, der die betroffenen Werkzeuge über Veränderungen informiert.

4.2.4 Transparenz

Die Anforderung an ein kooperationsunterstützendes Softwaresystem, ein hohes Maß an Transparenz zur Verfügung zu stellen, habe ich bereits in Kapitel 3.6.2 fachlich motiviert. Die technische Interpretation erfordert eine Instanz innerhalb des Systems, welche Informationen über die anwesenden Benutzer und deren Aktivitäten pflegt. Da mehrere Benutzer in der Lage sind, gleichzeitig auf Materialien zuzugreifen, ist es im Rahmen einer sinnvollen Koordination notwendig, daß ebenfalls Informationen über den aktuellen Zugriff auf diese Materialien zur Verfügung stehen. Denkbar wäre auch, Materialien für die Benutzung zu sperren, sofern sie bereits bearbeitet werden. Dieses ist allerdings von der speziellen Form der Kooperation abhängig. Auch hier ist wiederum ein geeigneter Benachrichtigungsmechanismus notwendig, um die Aktualität in einer sich verändernden Arbeitssituation gewährleisten zu können.

4.2.5 Benachrichtigungsmechanismus

Bereits mehrfach habe ich die Notwendigkeit eines geeigneten Benachrichtigungsmechanismus angesprochen. Die Komponenten des Softwaresystems sind auf verschiedene Rechner innerhalb eines Netzwerkes verteilt. Diese Komponenten müssen miteinander kommunizieren können. Die Realisierung eines Softwaresystems zur Unterstützung kooperativer Arbeit setzt also einen geeigneten Mechanismus zur Kommunikation von Softwarekomponenten über Rechnergrenzen hinweg voraus. Dieser Mechanismus umfaßt einerseits die Möglichkeit, entfernte Softwarekomponenten zu benutzen und andererseits die Fähigkeit, in umgekehrter Richtung Benachrichtigungen an die benutzende Komponente zu senden. Beide müssen insbesondere bei der Kooperation an einem Ort einem hohen Anspruch hinsichtlich der Performanz genügen, da Veränderungen jeglicher Art unmittelbar für alle Beteiligten sichtbar sein müssen. Nur so ist eine sinnvolle Transparenz zu gewährleisten. Die technischen Einzelheiten hinsichtlich der Realisierung der Kommunikation werde ich in Kapitel 4.4 eingehend diskutieren.

4.3 Technische Interpretation der Gruppenarbeitsumgebung

Aus den Visionen wurde deutlich, daß ein Softwaresystem, welches Kooperation unterstützt, innerhalb eines Netzwerkes arbeitet. An den Rechnern, die an diesem Netzwerk angeschlossen sind, arbeiten verschiedene Personen mit dem Softwaresystem. Es gibt auf diesen Rechnern Prozesse, die miteinander kommunizieren müssen. Die Menschen begeben sich mittels des Softwaresystems an den virtuellen Ort der Zusammenarbeit. Dieses ist der Ort der Gruppenarbeitsumgebung, der durch das Softwaresystem zur Verfügung gestellt wird.

4.3.1 Die Gruppenarbeitsumgebung innerhalb eines verteilten Softwaresystems

Die technische Repräsentation der Gruppenarbeitsumgebung in Form eines Umgebungsobjektes kann sich auf einem beliebigen Rechner befinden. Den Prozeß, in dem das Gruppenarbeitsumgebungsobjekt vorhanden ist, werde ich im Folgenden mit dem Begriff „administrativer Prozeß“ bezeichnen.

Die Rechner, an denen die Mitarbeiter tätig sind, benötigen ebenfalls eine Umgebung, welche für die Erzeugung von Werkzeugen sorgt. Ich werde sie im Folgenden mit dem Begriff „lokale Umgebung“ benennen.

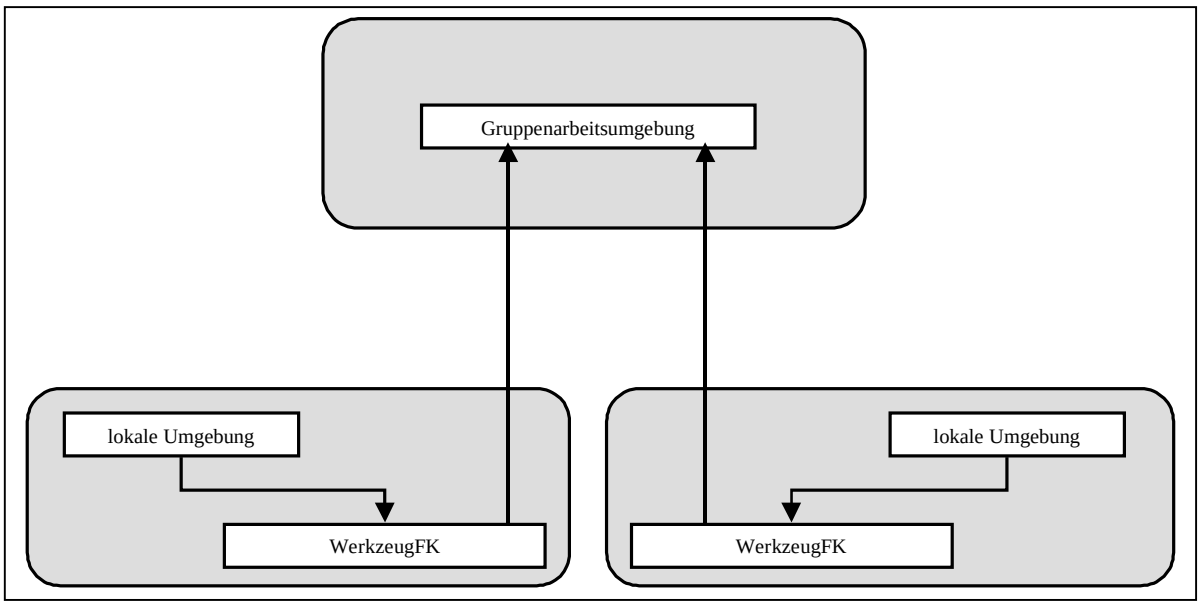


Abbildung 5 Die Gruppenarbeitsumgebung und Werkzeuge

Es ist also in der technischen Interpretation der Gruppenarbeitsumgebung zwischen dem Gruppenarbeitsumgebungsobjekt innerhalb des administrativen Prozesses und den lokalen Umgebungen, welche sich innerhalb der Prozeßräume der Rechner befinden, die von Arbeitsgruppenmitgliedern benutzt werden, zu differenzieren. Die lokalen Umgebungen sind für die Erzeugung von Werkzeu-

gen zuständig, welche ihrerseits mit dem Gruppenarbeitsumgebungsobjekt kommunizieren. Abbildung 5 deutet die Beziehung der lokalen Umgebungen und der Gruppenarbeitsumgebung an. Diese stark vereinfachte Grafik werde ich nun sukzessive mit Leben füllen. Zu diesem Zweck werde ich im ersten Schritt die Anforderungen hinsichtlich der Bereiche Materialverwaltung und Werkzeugkonsistenz behandeln. Abbildung 6 zeigt diese Konstruktion.

4.3.2 Materialverwaltung und Werkzeugkonsistenz

Um die Materialverwaltung zu gewährleisten, verfügt die Gruppenarbeitsumgebung über einen Materialverwalter. Dieser nimmt Anfragen nach Material entgegen und benutzt Materialversorger, um auf Datenbanken oder andere Datenquellen zugreifen zu können und das gewünschte Material zur Verfügung stellen.

Das Konzept des Materialversorgers stammt von Gryczan:

> „Ein Materialversorger ist ein Automat, der die Transformation eines Materials von einer externen Repräsentation zu einer internen Repräsentation (und umgekehrt) übernimmt.“ ([Gry96],S. 147)

Um den Materialverwalter nicht fest an die Versorger zu koppeln, werden die einzelnen Versorger durch die Gruppenarbeitsumgebung erzeugt und an den Materialverwalter übergeben. Wurde ein Material durch einen Versorger rekonstruiert, so wird es von dem Materialverwalter in Materialmagazinen gehalten. Im Falle einer erneuten Anfrage nach einem bereits rekonstruierten Material kann der Materialverwalter auf das Exemplar im Materialmagazin zurückgreifen. Werkzeuge fragen bei dem Materialverwalter dieses Material nach. Das Werkzeug trägt sich bei dem Ereignisverwalter als Beobachter ein, um über die Änderungen von Materialien informiert zu werden. Somit kann die Konsistenz von Werkzeugen gewährleistet werden. Verändert nun ein Werkzeug ein Material, so gibt es diese Änderung dem Ereignisverwalter bekannt. Alle Werkzeuge, die sich bei dem Ereignisverwalter als Interessenten für eine Ereignis „Materialveränderung“ angemeldet haben, werden nun davon in Kenntnis gesetzt und können gegebenenfalls ihre Informationen oder ihre Präsentation aktualisieren.

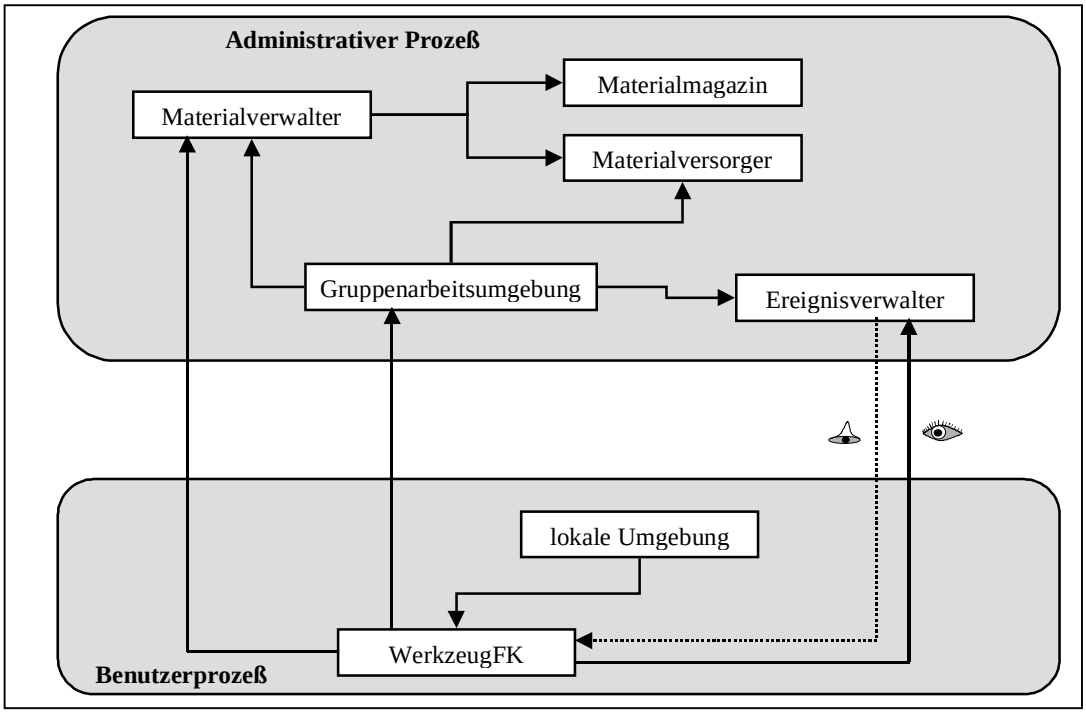


Abbildung 6 Die technische Konstruktion der Gruppenarbeitsumgebung

Mit Ausnahme der Gruppenarbeitsumgebung stammen die genannten Komponenten sowie deren Verantwortlichkeiten vorwiegend aus der technischen Interpretation der Arbeitsumgebung nach Gryczan.[20] Allerdings bezieht sich seine Interpretation auf Umgebungen, welche sich jeweils innerhalb der lokalen Prozesse befinden. In jedem Prozeß gibt es genau ein Exemplar dieser Umgebung. Wenn ein Softwaresystem die Kooperation mehrerer Personen unterstützen soll, so müssen diese Umgebungen über einen Materialkoordinator kommunizieren. Dieser Materialkoordinator, welcher sich in dem administrativen Prozeß befindet, regelt den konkurrierenden Zugriff auf Materialien.

Um den definierten Anforderungen an die Gruppenarbeitsumgebung gerecht werden zu können, ist es notwendig, dem System zwei weitere Komponenten hinzuzufügen. Ich habe bereits die Notwendigkeit festgestellt, daß das System über die Möglichkeit verfügen muß, Benutzer zu „kennen“ und deren Aktivitäten verfolgen zu können. Des weiteren ist hier der angedeutete Aspekt der Materialkonsistenz noch nicht berücksichtigt. Im zweiten Schritt werde ich das bestehende Konzept um diese beiden Bereiche erweitern. Abbildung 7 zeigt die Konstruktion auf, die diese beiden Aspekte berücksichtigt. Es befinden sich zwei weitere Komponenten, Benutzerverwalter und Materialkonsistenzverwalter, im System.

[20] Vergleiche dazu [Gry96], S. 144ff.

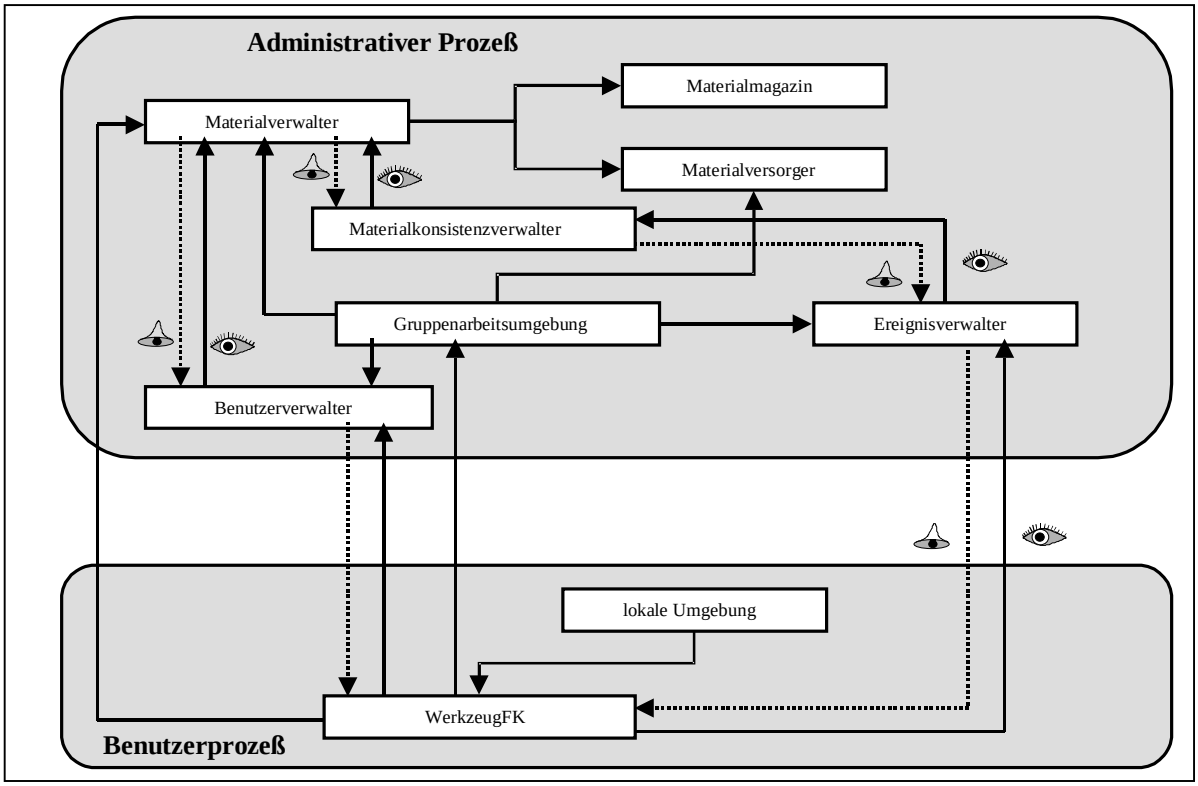


Abbildung 7 Die Gruppenarbeitsumgebung mit Benutzer- und Materialkonsistenzverwalter

4.3.3 Transparenz in der Gruppenarbeitsumgebung

Der Bereich der Verwaltung von Benutzerpräsenz und –aktivität wird durch den Benutzerverwalter übernommen. Der Benutzerverwalter kennt alle im System befindlichen Benutzer. Er kann Auskunft darüber geben, wer sich innerhalb der Gruppenarbeitsumgebung befindet. Weiterhin weiß er über die Aktivitäten der einzelnen Benutzer Bescheid. Werkzeuge, welche die Aktivitäten anderer Benutzer präsentieren, um somit eine höhere Transparenz zu ermöglichen, können von ihm die relevanten Informationen erfragen. Zusätzlich dazu können sie sich bei dem Benutzerverwalter als Beobachter anmelden, damit dieser sie von Aktivitätsveränderungen in Kenntnis setzt. Der Benutzerverwalter trägt sich bei dem Materialverwalter als Interessent für solche Ereignisse ein. Bevor ein Werkzeug die Bearbeitung eines Materials beginnen kann, muß es dieses zunächst beim Materialverwalter erfragen. Der Benutzerverwalter wird über den Beobachtungsmechanismus von dem Materialverwalter über dieses Ereignis informiert. Alle anderen Werkzeuge werden nun ihrerseits durch den Benutzerverwalter von dieser Aktivitätsveränderung in Kenntnis gesetzt.

4.3.4 Materialkonsistenz in der Gruppenarbeitsumgebung

Eine in Abbildung 7 ebenfalls hinzugekommene Komponente ist der Materialkonsistenzverwalter. Er kennt die Beziehungen und Abhängigkeiten zwischen den innerhalb des Systems vorhandenen Materialien. Er kann entscheiden, ob die Veränderung eines Materials ebenfalls Veränderungen

anderer Materialien zur Folge hat. Er wird durch die Gruppenarbeitsumgebung erzeugt und ihm wird der Materialverwalter bekannt gemacht. Dort meldet er sich als Interessent für die Veränderung der Bearbeitungssituation an. Dadurch ist er in der Lage festzustellen, welche Materialien bearbeitet werden, um daraus wiederum ableiten zu können, welche anderen Materialien gegebenenfalls durch eine Materialveränderung modifiziert werden. Der Ereignisverwalter meldet sich beim Materialkonsistenzverwalter als Interessent für Ereignisse an, welche die Veränderung von Materialien betreffen, damit er wiederum andere Werkzeuge informieren kann, die ebenfalls durch diese Änderung betroffen sind.

Fragt ein Werkzeug ein Material nach, so erfährt der Materialkonsistenzverwalter dieses über den Benachrichtigungsmechanismus und merkt sich diese Information. Führt ein Werkzeug eine Veränderung an einem Material durch, so muß es dieses dem Ereignisverwalter mitteilen, welcher diese Information an den Materialkonsistenzverwalter weiterleitet. Dieser prüft nun, ob es Materialien gibt, die durch diese Änderung ebenfalls betroffen sind. Zu diesem Zweck kann er möglicherweise Material beim Materialverwalter nachfragen. Sind andere Materialien von der Änderung betroffen, so sorgt er zunächst dafür, daß diese in einen konsistenten Zustand überführt werden und macht danach eine Rückmeldung an den Ereignisverwalter, der dann alle Werkzeuge, welche die betroffenen Materialien bearbeiten, von dieser Veränderung informiert.

Ein Vorschlag, wie sich ein Materialkonsistenzverwalter auf einer allgemeingültigen Basis konstruieren und implementieren läßt, findet sich im Kapitel 8.2.2.4.

4.4 Kommunikation innerhalb verteilter Softwaresysteme

Die Kommunikation innerhalb verteilter Softwaresysteme habe ich bereits als notwendige Anforderung identifiziert. An dieser Stelle möchte ich den Rahmen der technischen Realisierungsmöglichkeiten etwas detaillierter betrachten.

Im Kontext der verteilten Komponenten bezieht sich der Begriff Kommunikation auf die Interaktion von Objekten. Der Aufruf einer Methode eines entfernten Objektes, das Ausführen dieser Operation, sowie das Zurückliefern eines Ergebnisses an das aufrufende Objekt kennzeichnet die Interaktion zwischen diesen. Ausgehend von dieser Interpretation des Begriffes Kommunikation lassen sich die Begriffe synchrone und asynchrone Kommunikation auch beschreiben als synchrone und asynchrone Operationsaufrufe:

> “Synchronous call: A function call that does not allow further instructions in the calling process to be executed until the function returns [...].”

> "Asynchronous call: A call to a function that is executed separately so that the caller can continue processing instructions without waiting for the function to return [...]." [MSO97]

Synchrone Kommunikation bedeutet also, daß ein lokales Objekt die Methode eines entfernten Objektes aufruft und die Ausführung weiterer Anweisungen so lange angehalten wird, bis nach Ausführung der Methode des entfernten Objektes ein Ergebnis zurückgeliefert wird. Erst dann wird mit der Ausführung der Operationen fortgefahren. Die Reihenfolge der Bearbeitung der Operationen bleibt also streng sequentiell. Im Falle einer asynchronen Kommunikation wird lediglich der Operationsaufruf ausgeführt. Das lokale Objekt fährt dann mit seiner Programmausführung fort, während das entfernte Objekt mit der Ausführung der aufgerufenen Methode beschäftigt ist. Das lokale, also aufrufende, Objekt bekommt später von dem entfernten Objekt eine Nachricht über die erfolgte Bearbeitung der Operation sowie das Ergebnis mitgeteilt.

Im Folgenden sollen nun die Konsequenzen aufgezeigt werden, die für bzw. gegen die jeweilige Kommunikationsmethode sprechen. Diese Überlegungen stammen zum Teil aus [GL96].

Die Benutzung asynchroner Kommunikation bedeutet für den Benutzer eines Objektes, daß er die abgeschlossene Ausführung einer aufgerufenen Methode nicht voraussetzen kann. Durch diese Tatsache gerät die Diskussion in den komplexen Bereich der nebenläufigen Prozesse. Da dieser Bereich jetzt nicht vertieft werden soll, begnüge ich mich mit einem Zitat:

> „Das Verhalten nebenläufiger Systeme kann allerdings sehr komplex und dementsprechend schwer zu analysieren sein [...] Sequentielle Prozesse erlauben oft auch logisch einfachere Darstellungen und Vereinfachungen." ([JV87], S. 115)

Setzt die weitere Programmausführung die Ergebnisse einer Operation eines entfernten Objektes voraus, sind, im Falle asynchroner Kommunikation, Mittel zur Synchronisation notwendig. Die Programmierung asynchroner Kommunikation setzt einen höheren Entwicklungsaufwand voraus, da neben den Operationsaufrufen weiterer Aufwand betrieben werden muß, um die Ergebnisse dieser Aufrufe entgegenzunehmen und zu verarbeiten.

Die synchrone Kommunikation bringt verschiedene Nachteile mit sich. Befinden sich innerhalb eines Systems mehrere Objekte, die auf ein entferntes Objekt zugreifen, so muß dieses die Resultate in einer akzeptablen Zeit erbringen. Im Falle einer synchronen Kommunikation steht neben der Zeit, die dieses Objekt für die Ausführung der Operation benötigt, weiterhin noch die Zeit, die benötigt wird, um die Operationsaufrufe über das Netzwerk an das Objekt zu übermitteln sowie das Ergebnis wieder zurückzusenden. Hierzu addiert sich darüber hinaus die Zeit, welche das Objekt braucht, bis es den Aufruf entgegennehmen kann, da es möglicherweise noch mit der

Ausführung einer oder mehrerer anderer Operationen beschäftigt ist. Abschließend kommt noch die Zeit dazu, welche die Kommunikationspartner benötigen, um die Informationen, welche sie übermitteln wollen, für den Netzwerktransport aufzubereiten.

Diese Zeitfaktoren können sehr stark variieren. Es lassen sich nur begrenzt Aussagen über das Zeitverhalten eines Netzwerkes machen, da dieses essentiell von der Belastung abhängig ist. Aussagen über die Auslastung des benutzten Objektes hängen sehr stark sowohl von der Komplexität der Funktionen, die es zu erbringen hat, als auch von der Häufigkeit der Aufrufe ab. Dieses läßt sich allenfalls anhand von Wartesystemen simulieren und die Aussagen sind somit nicht verbindlich. Die Gesamtwartezeit ist in Abbildung 8 in Form einer Summe zusammengefaßt.

$$T_{\text{Wartezeit}} = T_{\text{Kodierung beim benutzenden Objekt}} + T_{\text{Transport des Aufrufes}} + T_{\text{Dekodierung beim benutzen Objekt}} + T_{\text{verbleibende Ausführungszeit anderer Operationen}} + T_{\text{Ausführung}} + T_{\text{Kodierung des Ergebnisses}} + T_{\text{Ergebnistransport}} + T_{\text{Ergebnisdekodierung}}$$

Abbildung 8 Gesamtwartezeit eines Kunden bei synchroner Kommunikation

Im Gegensatz dazu würde eine Formel, welche die Wartezeit bei asynchroner Kommunikation ausdrückt, wesentlich kürzer ausfallen. Im einfachsten Falle würde sie lediglich die Zeit der Kodierung des Aufrufes beim Kunden enthalten. Alle anderen Zeitfaktoren würden in die Gesamtausführungszeit mit eingehen, würden sich jedoch nicht als Wartezeit seitens des Kunden niederschlagen. Andererseits könnten zusätzliche Verzögerungen durch Quittieren des Aufrufes hinzukommen. Das Quittieren des Aufrufes kann entweder durch die Transportschicht, oder aber sogar erst durch die entfernte Komponente vor der Ausführung der Operation erfolgen.

Neben den genannten Überlegungen bezüglich der Performanz und des Entwicklungsaufwandes spielen auch noch andere Faktoren eine Rolle. Blockiert bei synchroner Kommunikation das benutzte Objekt, so führt dieses dazu, daß das benutzende Objekt ebenfalls blockiert ist. Dieses gilt analog für eine asynchrone Kommunikation, sofern der Aufrufende auf das Ergebnis einer Operation warten muß. Ebenso können Ausnahmesituationen beim benutzten Objekt Auswirkungen auf das benutzende Objekt haben. Da die Kopplung bei asynchroner Kommunikation wesentlich geringer ausfällt, ist das Ausfallrisiko entsprechend kleiner. Um dem Ausfallrisiko bei synchroner Kommunikation vorzubeugen, sind entsprechende Mechanismen zum Schutz einzubringen, die wiederum Mehraufwand für die Entwicklung bedeuten.

Grundsätzlich läßt sich also sagen, daß einerseits asynchrone Kommunikation für verteilte Systeme besser geeignet ist, da sie effizienter und sicherer ist, jedoch andererseits für die Entwicklung ein erheblicher Mehraufwand notwendig ist.

Für die in dieser Arbeit betrachtete Form der Kooperation bietet sich insbesondere im Hinblick auf die möglichst zügige Aktualisierung der Awareness-Informationen asynchrone Kooperation an. Während bei lokalen Systemen die Zeit für die Aktualisierung der Informationen eine eher untergeordnete Rolle spielt, ist sie für verteilte Systeme ein entscheidendes Kriterium um deren Benutzbarkeit zu gewährleisten.

4.5 Zusammenfassung und Ausblick

Ich habe in diesem Kapitel Anforderungen technischer Natur an Softwaresysteme definiert, die Kooperation an einem Ort unterstützen sollen. Nachfolgend habe ich meine technische Interpretation der Gruppenarbeitsumgebung vorgestellt. Diese Interpretation erfüllt die genannten Anforderungen und gibt einen Leitfaden zur Realisierung von solchen Softwaresystemen. Ich habe weiterhin Aspekte der Kommunikation innerhalb von verteilten Softwaresystemen betrachtet, da sie insbesondere bei verteilten Systemen zur Unterstützung von Kooperation an einem Ort sehr wichtig sind.

Dieser Vorstellung folgt nun eine kritische Betrachtung der Ergebnisse, bei der die Grenzen des Konzeptes der Gruppenarbeitsumgebung aufgezeigt werden. Dazu gehören einerseits Einschränkungen, welche im Konzept begründet sind und sich bereits abgezeichnet haben. Andererseits existieren Bereiche, in welche ich bei meinen Untersuchungen noch nicht vorgedrungen bin. Deshalb läßt sich insbesondere über die Tragfähigkeit des Konzeptes im Rahmen dieser Bereiche noch keine Aussage machen.

Der Anspruch an diese Arbeit besteht nicht nur in der Entwicklung eines einsetzbaren Konzeptes, sondern darüber hinaus soll es sich in den Kontext von WAM einfügen. Inwiefern dieses gelungen ist , soll im Folgenden ebenfalls untersucht werden.

5 Diskussion der Gruppenarbeitsumgebung

Ich möchte in diesem Kapitel kritisch zu dem vorgestellten Konzept der Gruppenarbeitsumgebung Stellung beziehen. Ich zeige Einschränkungen sowie noch nicht betrachtete Problembereiche dieses Konzeptes auf. Um überprüfen zu können, inwiefern sich die Gruppenarbeitsumgebung in den WAM Kontext einfügt, stelle ich sie dem Begriff der Arbeitsumgebung nach Gryczan gegenüber und arbeite Gemeinsamkeiten und Unterschiede heraus.

5.1 Einschränkungen und offene Punkte

Das Konzept der Gruppenarbeitsumgebung findet im Bereich der von mir in dieser Arbeit definierten Kooperation an einem Ort Verwendung. Es stellt damit einen möglichen Lösungsansatz dar, um Menschen bei der Zusammenarbeit zu unterstützen. Mit diesem Ansatz sind allerdings verschiedene Einschränkungen verbunden, welche ich im Folgenden nochmals nennen möchte.

Diese Form der Kooperation ist lediglich für die Zusammenarbeit von Personen in verhältnismäßig kleinen Teams geeignet. Die beteiligten Menschen koordinieren ihre Arbeit entweder explizit, oder die Koordination findet implizit durch ein geschaffenes Verständnis für die jeweilige Arbeitssituation statt. Es leuchtet ein, daß eine solche Form der Kooperation im Rahmen großer Personengruppen nicht realisierbar ist. Der hohe Koordinationsbedarf würde in einem solchen Falle dazu führen, daß die beteiligten Personen einen Großteil ihrer Arbeit mit der Koordination selbst verbringen würden. Als Folge davon würde die Effizienz der Arbeit sinken.

In diesem Zusammenhang offenbart sich ein grundsätzliches Problem der Gruppenarbeitsumgebung. Es wird im Rahmen der Zusammenarbeit an einem Ort versucht, eine konkrete Zusammenarbeit mehrerer Personen mit Hilfe eines Softwaresystems an einem virtuellen Ort zu ermöglichen. Es stellt sich die Frage, ob diese Abbildung jeweils für eine konkrete Form der Zusammenarbeit ausreichend ist. Trotz aller Ansätze wie beispielsweise WYSIWIS oder Videounterstützung bleiben immer noch Zweifel, ob jegliche Form der Kooperation durch ein Softwaresystem unterstützt werden kann, oder ob vielmehr durch eine ungenaue Abbildung gerade die Aspekte verloren gehen, die eine Zusammenarbeit von mehreren Personen an einem wirklich existierenden Ort wertvoll machen.

Eine weitere Einschränkung besteht im Bereich der Anwendungen mit Massendaten. Ich erwähnte bereits in Kapitel 4.2.1, daß die vorgeschlagene Lösung für die Materialverwaltung durch den damit verbundenen Verwaltungsaufwand begrenzt ist. Darüber hinaus legt sie nahe, daß Materialien an dem Ort der Gruppenarbeitsumgebung zur Verfügung gestellt werden. Da es sich dabei um einen einzigen Prozeß handelt, der Materialien für alle beteiligten Komponenten zur Verfügung

stellen muß, wäre dieses System im Falle der Bearbeitung von Massendaten hoffnungslos überlastet. Meines Erachtens liegt es aber in der Natur der betrachteten Kooperationsform, daß sich das zu erwartende Materialaufkommen häufig im Rahmen halten wird. Nichts desto trotz muß die Frage, inwiefern sich die Handhabung von Massendaten sinnvoll integrieren läßt, hier unbeantwortet bleiben.

5.2 Gruppenarbeitsumgebung und Arbeitsumgebung

Ich werde nun die Charakteristika der Arbeitsumgebung nach Gryczan nennen und sie der Gruppenarbeitsumgebung gegenüberstellen, um feststellen zu können, inwiefern diese beiden Konzepte übereinstimmen bzw. voneinander abweichen.

5.2.1 Die Arbeit innerhalb einer Umgebung

Menschen arbeiten in ihrer Arbeitsumgebung. Sie können die Arbeitsgegenstände nach ihren eigenen Wünschen organisieren. Die Arbeitsumgebung macht keine Annahmen über Reihenfolgen der Benutzung dieser Gegenstände. Die Gruppenarbeitsumgebung ermöglicht mehreren Benutzern die Arbeit innerhalb einer Umgebung. Dementsprechend sind alle Benutzer für die Einrichtung derselben verantwortlich. Genauso, wie Mechaniker in einer Werkstatt gemeinsam beschließen, wo Werkzeuge und Verbrauchsmittel wie Schrauben oder Öl aufbewahrt werden, geschieht dieses ebenso innerhalb der Gruppenarbeitsumgebung.

5.2.2 Bearbeitung von Material

In der Arbeitsumgebung sind die zu bearbeitenden Materialien lokal vorhanden. Die Bearbeitung von Materialien ist nur innerhalb einer Umgebung möglich. Simultaner Zugriff auf Material findet nach Gryczan innerhalb einer Arbeitsumgebung nur durch verschiedene Werkzeuge statt.

> „Materialien werden in Arbeitsumgebungen exklusiv bearbeitet. Ein simultaner Zugriff aus verschiedenen Arbeitsumgebungen ist nicht möglich. Der konkurrierende Zugriff auf Materialien wird durch das Konzept von Original und Kopie umgesetzt.“ ([Gry96], S. 143)

Die Gruppenarbeitsumgebung sieht die Möglichkeit des gemeinsamen Zugriffs auf Material ebenfalls vor. Grundsätzlich ist es möglich, Material gleichzeitig zu bearbeiten. Ob es gegebenenfalls sinnvoll ist, eine regelnde Koordinationskomponente für den Materialzugriff in das System zu integrieren, hängt im einzelnen von der betrachteten Kooperationssituation ab. Da Material immer nur an einem Ort sein kann, ist es nicht möglich, gleichzeitig auf Material zuzugreifen, welches sich in unterschiedlichen Gruppenarbeitsumgebungen befindet. Sowohl für eine Materialkoordina-

tion innerhalb einer Gruppenarbeitsumgebung als auch zwischen verschiedenen wäre eines der in Kapitel 3.4.3 und 3.5 angedeuteten Konzepte durchaus denkbar.

5.2.3 Aktuelle Sicht auf Material

Werden Materialien von verschiedenen Werkzeugen bearbeitet, so müssen diese Werkzeuge jederzeit eine aktuelle Sicht auf das Material präsentieren. Diese Werkzeugkonsistenz wird über einen Benachrichtigungsmechanismus realisiert:

> „Der simultane Zugriff auf ein Material durch verschiedene Werkzeuge wird durch einen Benachrichtigungsmechanismus auf Umgebungsebene geregelt. Dadurch wird erreicht, daß Werkzeuge und Automaten eine aktuelle Sicht auf das Material präsentieren können.“ ([Gry96], S. 143)

Dieser Benachrichtigungsmechanismus bleibt ebenfalls erhalten und ist sogar zwingend notwendig. Die über diesen Mechanismus miteinander verknüpften Werkzeuge, die zugleich ein Material bearbeiten, werden jedoch nicht mehr notwendigerweise von einem einzelnen Menschen benutzt.

5.2.4 Transparenz und Awareness

Das Leitbild des Arbeitsplatzes für qualifizierte menschliche Tätigkeit fordert, daß Softwaresysteme die Tätigkeiten der Menschen unterstützen sollen. Um die Zusammenarbeit mehrerer Personen sinnvoll unterstützen zu können, ist es deshalb insbesondere notwendig, ihnen ihre Rolle innerhalb einer Gruppe zu vergegenwärtigen. So können sie Abweichungen von vordefinierten Arbeitsprozessen erkennen und entsprechend reagieren. Awareness-Informationen bilden die Gewähr dafür, daß dieses möglich bleibt. Für die Arbeit innerhalb der Gruppenarbeitsumgebung wird diese Forderung unterstrichen, da sich gerade dort Koordinationsaspekte häufig auf Awareness und Transparenz abstützen. Je mehr Koordination über die Transparenz innerhalb der Kooperation erreicht wird, desto höher ist der Stellenwert, welchen sie einnimmt.

5.3 Abschließende Betrachtung

Die Gruppenarbeitsumgebung ist ein mögliches Konzept, wie die Zusammenarbeit mehrerer Personen an einem virtuellen Ort unterstützt werden kann. Voraussetzung für ihren Einsatz ist allerdings, daß eine entsprechende Kooperationssituation vorliegt. Ich habe in der Taxonomie der Kooperation gezeigt, daß sich die Zusammenarbeit von Menschen nicht problemlos auf ein unterstützendes Softwaresystem abbilden läßt. Häufig wird sich die konkrete Form der Zusammenarbeit verändern. Da es diese vorher möglicherweise gar nicht gegeben hat, erfordert die Umstel-

lung ebenfalls neue Konzepte und Metaphern. Es muß darauf geachtet werden, daß der eigentliche Sinn und Zweck der Zusammenarbeit nicht verfälscht wird.

Die Kooperation an einem Ort. basiert auf der Annahme, daß den Benutzern die Möglichkeit eingeräumt werden soll, ihre Arbeitsweisen nach ihren eigenen Wünschen zu organisieren. Das System versucht nicht, Annahmen über die konkrete Form der Kooperation zu machen. Vielmehr wird den Benutzern die Möglichkeit gegeben, Kooperation und notwendige Koordination außerhalb des Systems zu definieren, um sie mit Hilfe des Softwaresystems eigenverantwortlich und effizient durchzuführen. Das System kann dieses geeignet unterstützen, indem Mittel zur expliziten Kommunikation und Koordinationsmechanismen zur Verfügung gestellt werden, Transparenz für die beteiligten Personen gewährleistet wird und alltägliche Dinge sowie der Umgang mit ihnen im System abgebildet werden.

Das Konzept der Gruppenarbeitsumgebung ist meiner Meinung nach geeignet, um im Rahmen des Leitbildes des Arbeitsplatzes für qualifizierte menschliche Tätigkeit eingesetzt werden zu können. Die angeführte Gegenüberstellung zeigt, daß die Gruppenarbeitsumgebung als Interpretation des Arbeitsplatzes mehrerer Personen nicht im Gegensatz zum Begriff der Arbeitsumgebung als Beschreibung des Arbeitsplatzes einer Einzelperson steht, sondern diese Sichtweise vielmehr erweitert. Die technische Interpretation läßt dieses ebenfalls zu.

Offen bleibt, wie die Kooperation der beteiligten Personen konkret aussieht. Das vorgeschlagene Konzept definiert lediglich die Rahmenbedingungen für eine sinnvolle Kooperation. Im Rahmen einer tatsächlichen Ausgestaltung eines Anwendungssystems muß überprüft werden, mit welchen Mitteln die Zusammenarbeit unterstützt werden kann. Als ein Beispiel dient hier das Pausenplanersystem, welches im Anhang durch verschiedene Systemvisionen vorgestellt wird. Darüber hinaus beschäftigen sich die Kapitel 7 und 8 mit der Realisierung desselben.

5.4 Ausblick

Nachdem dieses Kapitel ausschließlich der Einordnung und Überprüfung der bisher erarbeiteten Konzepte diente, werde ich nun, bevor ich die konkrete Realisierung des Pausenplanersystems vorstellen werde, die Grundlagen vermitteln, welche zum Verständnis des Gesamtsystems notwendig sind. Dieses betrifft sowohl die zugrundeliegende Technologie als auch die im Rahmen der Entwicklung verwendeten Werkzeuge zur Anwendungsentwicklung.

6 Entwicklung verteilter Anwendungen mit OLE und der MFC

In diesem Kapitel möchte ich eine Technologie vorstellen, welche sowohl Konzepte als auch Realisierungen für die Entwicklung verteilter Anwendungen anbietet. Es handelt sich dabei ausschließlich um Produkte der Firma Microsoft. Sie hat im Laufe der Zeit eine Reihe von Rahmenwerken entwickelt, die für verschiedene Entwicklungsbereiche geeignet sind. Unter anderem handelt es sich dabei um die MFC und um OLE.[21] Ich möchte eine Einführung in diese Rahmenwerke geben, um einerseits beim Leser ein Verständnis zu schaffen, welches es ermöglicht, den technischen Entwurf des Pausenplanersystems nachzuvollziehen und andererseits um aufzuzeigen, auf welchem Stand sich die verfügbare Technologie für verteilte Anwendungen befindet. Nicht zuletzt hat die oben erwähnte Firma einen nicht geringen Anteil an der Herstellung von Werkzeugen zur Softwareentwicklung.

Die genannten Rahmenwerke verfügen über eine außerordentlich hohe Komplexität. Deshalb beleuchte ich nur die Aspekte näher, welche für diese Arbeit, also für die Entwicklung verteilt arbeitender Softwaresysteme, wichtig sind. Weiterhin weise ich darauf hin, daß ich die MFC nur zu einem geringen Teil vorstellen werde. Tim Krauß hat sich in seiner Arbeit wesentlich intensiver mit der MFC auseinandergesetzt.[22]

6.1 Die Rahmenwerke

Rahmenwerke zur Anwendungsentwicklung sind aus der heutigen Welt der Softwareentwicklung nicht mehr wegzudenken. Die Benutzung von Rahmenwerken verspricht geringere Produktionskosten, geringere Fehleranfälligkeit, verbesserte Wartbarkeit sowie viele weitere Vorteile, die letztlich eine bessere und effizientere Softwareentwicklung ermöglichen. Trotz einer starken Konsolidierung gibt es immer noch Probleme, welche die Nutzung von Rahmenwerken mit sich bringt. Pree weist in [Pre96] auf die Problembereiche hin, die sich den Entwicklern bei der Benutzung offenbaren. Ich fasse sie hier zusammen:

---

[21] Ich möchte anmerken, daß es nicht Bestandteil dieser Arbeit ist, festzustellen, ob es sich bei der MFC und OLE tatsächlich um Rahmenwerke handelt oder nicht. Tatsächlich werden beide als solche in der Literatur bezeichnet.

[22] Es handelt sich bei dieser Arbeit um [Kra98]. Sie beschäftigt sich mit dem Konzept von Document/View und untersucht die mögliche Benutzung der MFC im Rahmen der in WAM enthaltenen Entwurfsmetaphern.

- Die meisten Rahmenwerke lassen sich nur erweitern, wenn dieselbe Programmiersprache oder sogar dieselbe Entwicklungsumgebung benutzt wird.
- Die Frage nach der Kompatibilität von Rahmenwerken untereinander bleibt unbeantwortet. Vielfach müssen Rahmenwerke für die Benutzung adaptiert werden, oder können überhaupt nicht in Zusammenarbeit mit anderen benutzt werden.
- Die geeignete Wahl von Oberklassen des Rahmenwerks stellt Designentscheidungen immer wieder vor Probleme. Veränderungen in denselben können das gesamte Rahmenwerk in Frage stellen.[23]
- Es gibt keine oder nur schlechte Werkzeuge für die Rahmenwerkentwicklung und –spezialisierung.

Pree stellt weiterhin fest, daß Rahmenwerke wie OpenDoc/SOM und OLE/COM zumindest einen Teil dieser Probleme beseitigt haben. Im weiteren Verlauf dieses Kapitels werden ich aufzeigen, was sich hinter der MFC und OLE verbirgt und welche von Pree erwähnten Probleme durch diese Rahmenwerke gelöst sind.

6.1.1 Dokumente und Sichten auf Dokumente

Die Rahmenwerke basieren auf der Document/View (=Dokument/Sicht) Architektur. Motivation derselben ist die Entkopplung der Daten und der Funktionalität von deren Darstellung. Ähnlichkeiten zu dem „Model View Controller“ (=MVC) Ansatz bei Smalltalk[24], der technischen Interpretation der Werkzeugmetapher sowie zu anderen Rahmenwerken sind deutlich zu erkennen. In der Document/View Architektur repräsentieren Dokumente die Daten und die Funktionalität der Applikation:

> „Document class objects,[...], manage the application's data.“ [Mic97a]

Dokumente sind für die Pflege und die Bearbeitung der Daten verantwortlich. Sichten hingegen dienen sowohl der Präsentation der Daten auf der Oberfläche als auch der Entgegennahme von Kommandos durch den Benutzer:

> „The view's responsibilities are to display the document's data graphically to the user and to accept and interpret user input as operations on the document.“ [Mic97a]

[23] Dieses Problem ist in der Literatur auch als das „fragile base class problem“ bekannt.

[24] Zu der MVC Architektur vergleiche [Lew95].

Durch die Entkopplung der Daten und der Funktionalität von der Präsentation soll es möglich sein, flexible Formen der Darstellung wählen zu können, ohne daß dieses Auswirkungen auf die Funktionalität der Anwendung hat. Ein häufig genanntes Beispiel sind Auswertungen auf statistischen Daten, welche einerseits in Tabellenform, andererseits auch in Form eines Diagramms dargestellt werden können. Verschiedene Sichten auf diese Daten werden ermöglicht, indem die Auswertungen durch das Dokument durchgeführt werden und zwei verschiedene Sichten mit diesem verknüpft werden.

Die Attraktivität der Document/View Architektur im Rahmen der MFC besteht darin, daß zwischen Dokumenten und Sichten bereits definierte Zusammenhänge und Interaktionen im Rahmenwerk verankert sind. Ein Beobachtermuster ist zwischen ihnen vorgesehen. Die dazu notwendigen Registrierungen werden automatisch durch das Rahmenwerk vorgenommen. Darüber hinaus sind noch weitere Mechanismen in das Rahmenwerk integriert. Ein Entwickler, der sich dieses Rahmenwerkes bedient, muß sich also weniger Gedanken um die Implementation solcher Details machen, sondern er kann sich im Rahmen der Benutzung dieser Mechanismen verstärkt auf den anwendungsfachlichen Aspekt der Softwareentwicklung konzentrieren.

Das Konzept von Document/View findet sich in vielen Details der hier untersuchten Rahmenwerke wieder. Deshalb werde ich im Folgenden häufiger den Begriff des Dokumentes oder der Sicht auf ein Dokument benutzen. Für einen vertiefenden Einblick in die Document/View Architektur verweise ich auf [Kra98].

6.1.2 Das OLE Rahmenwerk

Brockschmidt begründet in [Bro95] die Existenz von OLE in der gewachsenen Notwendigkeit nach Standards. Monolithische Anwendungen wachsen zu ungeahnter Komplexität. Im Verlaufe der Zeit entstand zunehmend der Wunsch nach standardisierten Komponenten, welche der stark anwachsenden Komplexität entgegenwirken und den Entwicklern darüber hinaus eine höhere Flexibilität versprechen. Ein früh entwickeltes Beispiel für eine solche Standardkomponente stellt die „Open database connectivity“ (= ODBC) dar. Jeder Hersteller, der sich an diesen Standard hält, eröffnet damit einem Benutzer die Möglichkeit, über eine definierte Menge von Operationen verfügen zu können. Benutzer von ODBC können folglich eine durch den Standard definierte Grundfunktionalität voraussetzen, um auf Daten aus einer Datenbank zuzugreifen. Die Hersteller können überdies die Funktionalität ihrer Produkte über den Standard hinaus erweitern. Benutzer können gegebenenfalls zur Laufzeit die erweiterte Funktionalität erfragen und benutzen. Ich werde zeigen, daß dieses Konzept der Mindestanforderung an eine Komponente ebenfalls in COM verankert ist.

Ende der 80er Jahre entstanden die ersten Ansätze, Daten verschiedener Applikationen in einem Dokument zusammenfassen zu können. Unmittelbar daraus entstand ein Protokoll, welches zur Laufzeit den Datentransfer zwischen Anwendungen ermöglicht. Dieses DDE[25] Protokoll bildete die Grundlage für die erste Version von OLE. Man erkannte später, daß OLE 1.0 lediglich die spezielle Form eines Mechanismus ist, der es ermöglicht, Softwarekomponenten an Anwendungen anknüpfen zu können, um so deren Funktionalität zu erhöhen. Die Verallgemeinerung dieses Konzeptes begründete die Version 2.0. Die Infrastruktur dafür bildet das Component Object Model (= COM) . Brockschmidt geht nicht davon aus, daß es in der Zukunft eine nachfolgende Version von OLE geben wird:

> „In fact, there is no OLE 3.0, nor are there plans for such a release. This is because of the reusable design idea: OLE's architecture accommodates new technologies [...] without requiring modification to the base designs." ([Bro96], S. 2)

Die Basis der Technologie ist gelegt. Da diese Technologie insbesondere für Erweiterungen konzipiert ist, wird es nicht mehr notwendig sein, eine neue Version ins Leben zu rufen.

OLE dient heute dazu, komponentenbasierte Softwareentwicklung zu unterstützen. Die Frage danach, was OLE eigentlich ist, beantwortet Brockschmidt wie folgt:

> „OLE is a unified environment of object-based services with the capability of both customizing those services and arbitrarily extending the architecture through custom services, with the overall purpose of enabling rich integration between components." ([Bro95], S. 9)

Eine Komponente wiederum ist nach Brockschmidt :

> „A software component is reusable piece of code and data in binary form, that can be plugged into other software components of other vendors with relatively little effort." ([Bro95], S. 8)

Die Idee, die zur Entwicklung von OLE führte, besteht also darin, Softwareentwicklung als Zusammenstecken von Bausteinen betreiben zu können. Folglich war es notwendig eine Technologie zu entwickeln, mit der Bausteine gefertigt werden können und weiterhin die Möglichkeit zu schaffen, diese Bausteine zu kombinieren, um komplexere Gebilde daraus zu entwickeln. COM stellt das Konzept zur Verfügung, welches das Erstellen und Verknüpfen von Bausteinen ermög-

[25] DDE steht für „Dynamic data exchange".

licht. OLE hingegen benutzt COM und stellt eine Reihe komponentenbasierter Dienste zur Verfügung.

Es gibt nicht viele Rahmenwerke, welche die komponentenbasierte Softwareentwicklung unterstützen. Orfali et al. stellen fest, daß es lediglich ihrer zwei gibt, die heutzutage als Standard gelten können:

> „If CORBA is the industry's leading standard for components, then *Microsoft's Object Linking and Embedding (OLE)* is the *de facto*, other standard'."([OHE96], S. 283)

Sie meinen, daß, nachdem große Mainframe Architekturen zunehmend durch Client/Server Architekturen abgelöst werden, sich die Architektur der Komponenten als die Technologie der Zukunft konsolidieren wird. CORBA und OLE bilden hier die beiden Standards, die diese Technologie ermöglichen.

6.1.3 Das MFC Rahmenwerk

Die MFC unterstützt die Entwicklung von Anwendungen auf Plattformen der Windows Familie. Microsoft selbst definiert die MFC wie folgt:

> „A set of C++ classes that encapsulate much of the functionality of applications written for the Microsoft Windows operating systems." ([Mic97a])

Die erste Version der MFC wurde im April 1992 veröffentlicht. Neben Klassen zur Kapselung des API[26] des Windows Betriebssystems, enthielt sie ebenfalls Containerklassen, Runtime Type Information (RTTI), sowie Ansätze zur Unterstützung der Persistenz von Objekten. Sie enthielt gleichermaßen die Unterstützung von OLE 1.0. Bis zur heutigen Zeit ist die Versionsnummer der MFC auf 4.0 angestiegen. Sie enthält eine wesentlich feinere Struktur, eine breitere Palette an Sichten- und Dokumentenklassen, Multithreading, Exceptions, Unterstützung von Datenbankzugriffen bis hin zu ODBC, die Unterstützung aller Dienste von OLE 2.0 und vieles mehr.[27] Ich werde hier die Vorstellung des MFC Rahmenwerkes nicht weiter intensivieren, sondern nur an den relevanten Stellen auf Ausschnitte der MFC eingehen, um Zusammenhänge verdeutlichen zu können.

---

[26] API steht für „Application Programming Interface" und bezeichnet die Operationen, die ein System nach außen für die Benutzung zur Verfügung stellt.

[27] Diese Informationen stammen aus [Bri97].

6.2 Ein Überblick über OLE

Brockschmidt zeigt 16 verschiedene Dienstbereiche von OLE auf. Da die Erklärung dieser Dienste den Rahmen dieser Arbeit sprengen würde, bediene ich mich einer Skizze von Orfali et al., welche ich um die Teilbereiche von OLE, wie sie in [Mic97b] genannt werden, erweitere:

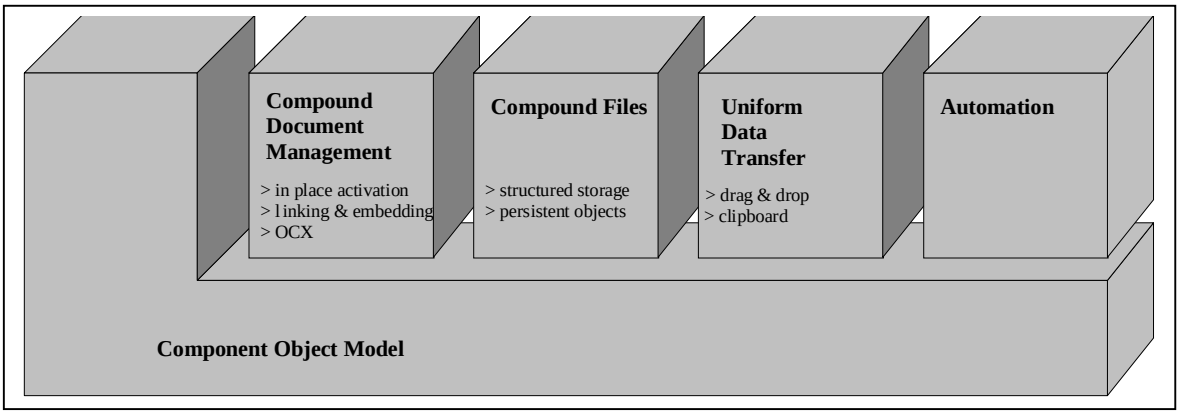


Abbildung 9 Die Architektur von OLE

Es läßt sich dieser Abbildung entnehmen, daß COM die Grundlage bildet, die OLE nutzt, um Dienste zur Verfügung zu stellen. Ich werde die einzelnen Bereiche von OLE kurz vorstellen und andeuten, welche Dienste sie anbieten.

- COM
  stellt die Infrastruktur für OLE zur Verfügung. COM ermöglicht das Benutzen von Softwarekomponenten durch Interfaces.

- Compound Document Management

  bietet die Möglichkeit, Daten fremden Formates beliebig in Dokumente zu integrieren.

  - linking & embedding
    In die Daten einer Anwendung lassen sich beliebig Fremddaten integrieren. So läßt sich die Tabelle einer Kalkulationsanwendung in eine Textverarbeitung integrieren.

  - in place activation
    Sind Daten fremden Formates innerhalb der Daten einer Anwendung enthalten, so ermöglicht „in place activation", daß die Anwendung, mit der diese Fremddaten erstellt wurden, innerhalb der einbettenden Anwendung ausgeführt werden kann. So würde innerhalb der Textverarbeitung die Kalkulationsanwendung gestartet werden, um eine Bearbeitung zu ermöglichen. Ein separater Start der Kalkulationsanwendung sowie nachträgliches Importieren der modifizierten Daten ist somit nicht mehr notwendig.

- OCX
  Der Begriff OCX bezeichnet eine Architektur für die Entwicklung von spezialisierten Interaktionstypen. Diese sind dann ebenfalls wieder als eigenständige Softwarekomponenten allgemein verfügbar. In diesem Zusammenhang ist ein OCX als eine Komponente zu sehen, die inplace activation unterstützt.

- Compound Files

  ermöglichen die Integration von Fremddaten in die persistenten Dokumente von Anwendungen.

  - persistent objects
    Über einen Streaming Mechanismus wird die Persistenz von Objekten unterstützt

  - structured storage
    Daten können in einer Datei (einem Dokument) so strukturiert abgelegt werden, daß in einer beliebigen Verschachtelungstiefe Fremddaten integriert werden können.

- Uniform data transfer

  ermöglicht, Daten aus beliebigen Anwendungen in eine Standardrepräsentation zu bringen.

  - drag & drop[28]
    Über diese Standardrepräsentation sind Fremdanwendungen in der Lage, bei Drag & Drop auf allgemeingültiger Basis Kommandos auszuführen.

  - clipboard
    Das Clipboard (im deutschen auch als Zwischenablage bekannt) ist ein Werkzeug, welches beliebige Daten in einer Standardrepräsentation aufnehmen und sie auf Wunsch in andere Anwendungen einfügen kann.

- Automation
  stellt die Möglichkeit bereit, dynamisch auf Attribute und Methoden von Komponenten zuzugreifen.

---

[28] Obwohl möglicherweise nicht mehr erwähnenswert, sei hier nichts desto trotz angemerkt, daß drag & drop für die Möglichkeit steht, Dinge auf dem Bildschirm mit der Maus zu bewegen (drag) und an einer gewünschten Stelle fallen zu lassen (drop), um damit eine Reaktion beim System auszulösen.

Da COM die Grundlage für die weiteren Dienste von OLE bildet, werde ich diese Spezifikation genauer vorstellen. Anschließend werde ich den Bereich der Automation näher betrachten, da wir sie benutzt haben, um das Pausenplanersystem als verteiltes Softwaresystem zu entwickeln. Ich sehe davon ab, die anderen Bereiche von OLE vorzustellen, da sie den Rahmen dieser Arbeit bei weitem sprengen würden und darüber hinaus in keinem Zusammenhang mit dieser Arbeit stehen.

6.3 COM, die Basis

COM bildet die gesamte Grundlage für OLE und ermöglicht die Entwicklung von Softwarekomponenten. Solche Komponenten können eine beliebige Komplexität haben. Eine Komponente könnte zur Berechnung einfacher trigonometrischer Funktionen entwickelt werden, oder es könnte sich dabei ebenso um eine komplexe Textverarbeitung handeln.

6.3.1 Komponenten

COM ist nicht etwa eine Sprache oder eine Umgebung, sondern nur eine Spezifikation. Ich fasse eine Definition von Rogerson aus [Rog97] zusammen, um zu beschreiben, was Komponenten sind, die nach der COM Spezifikation entwickelt werden. COM Komponenten

- sind vollständig sprachunabhängig. Sie können in beliebigen Sprachen entwickelt und benutzt werden.
- werden in Binärform geliefert.
- beinhalten einen automatischen Versionsmechanismus, welcher die Koexistenz verschiedener Versionen eines Objektes ermöglicht.
- können innerhalb eines Netzwerkes beliebig lokalisiert und benutzt werden. Eine Komponente auf einem entfernten System wird genauso behandelt, wie eine auf dem lokalen System.

Komponenten bieten Dienste an. Ein Benutzer eines Dienstes ist ein Kunde. Um zu gewährleisten, daß Kunden Dienste identifizieren können, erhält jeder Dienst bei der Entwicklung eine eindeutige Identifikation. Damit ein Kunde herausfinden kann, welche Dienste eine Komponente anbietet, muß diese ein Mindestmaß an Kommunikationsfähigkeit implementieren. Dieses wird durch ein Interface ermöglicht, welches jede Komponente anbieten muß. Anhand dieses Interfaces kann ein Kunde zur Laufzeit Informationen über die Eigenschaften einer Komponente erfragen. Dieses Interface heißt *IUnknown*. Abbildung 10 zeigt die Methoden, die *IUnknown* umfaßt.

```
QueryInterface (IID,Interface)
AddRef()
Release()
```

Abbildung 10 Das IUnknown Interface

Die Methode *QueryInterface* ermöglicht dem Kunden einer Komponente herauszufinden, welche Interfaces diese Komponente über *IUnknown* hinaus weiterhin anbietet. Die IID ist der bereits erwähnte eindeutige Identifikator eines Interfaces. Bei der Anfrage liefert die Komponente das gewünschte Interface, sofern sie es zur Verfügung stellt. Andernfalls weiß der Kunde, daß der gewünschte Dienst nicht von dieser Komponente angeboten wird. Die Methoden *AddRef* und *Release* dienen der Referenzzählung auf eine Komponente. Der Kunde inkrementiert anhand von *AddRef* die Referenz, sobald er einen Dienst einer Komponente in Anspruch nimmt. Hat er die Benutzung beendet, dekrementiert er die Referenz mittels *Release*. Die Komponente, welche selbst für ihre Lebenszeit verantwortlich ist, löscht sich selbst, sobald der Referenzzähler Null erreicht hat.

6.3.2 Interfaces und Wiederverwendung

Ein wesentlicher Vorteil der Objektorientierung liegt in dem hohen Grad der Wiederverwendung von Klassen. Dieses wird einerseits durch die Benutzung von Exemplaren einer Klasse erreicht. Andererseits werden Klassenhierarchien entwickelt, um dann die Umgangsformen von Oberklassen in Spezialisierungen wiederverwenden zu können. Üblicherweise ist diese Vorgehensweise dadurch gekennzeichnet, daß man Oberklassen entwickelt und die Implementation der Methoden in abgeleiteten Klassen sukzessive verfeinert oder die Implementation dieser Methoden aus den Oberklassen wiederverwendet. COM Interfaces sind abstrakten Oberklassen gleichzusetzen. Sie definieren über Methoden die Umgangsformen, die allen Objektem, welche dieses Interface anbieten, gemein sind. Dieses legt die Nutzung der Vererbung von Interfaces nahe. Die Vererbung im Sinne einer Vererbung von Implementation wird jedoch durch COM nicht unterstützt. Es ist in COM nicht möglich, die Implementation einer Methode in einer abgeleiteten Klasse wiederzuverwenden. Genauer betrachtet bietet COM generell nicht die Möglichkeit an, von einer Klasse zu erben, welche ein Interface implementiert. Gleichwohl ist die Wiederverwendung von Interfaces in COM möglich. Ich werde im weiteren Verlauf des Kapitels zeigen, wie dieses funktioniert. Rogerson führt ein Problem, welches Pree ebenfalls erkennt und von mir in Kapitel 6.1 bereits erwähnt wurde, als Argumentation gegen die Vererbung von Interfaceimplementationen an. Er stellt fest, daß eine Vererbungsbeziehung eine sehr feste Kopplung einer Klasse mit einer anderen Klasse darstellt und dieses hinsichtlich Veränderungen von Oberklassen einen möglicherweise umfangreichen Änderungsaufwand zur Folge hätte. Darüber hinaus sieht er die Vererbung als ein Mittel an, welches die Kunden einer Komponente nicht ausreichend gegen Veränderung schützt.

Und gerade dieses ist ausgesprochen wichtig bei Komponenten, die jederzeit und überall benutzt werden können. Brockschmidt erwähnt neben diesen Argumenten weiterhin als Grund, daß es wenig einleuchtend erscheint, für ein sprachunabhängiges Konzept wie Komponenten ein sprachabhängiges Konzept wie die Vererbung zu benutzen.

Um die Vererbungsproblematik zu verdeutlichen, möchte ich ein Beispiel anführen. Ich nehme an, es gäbe eine Klasse X, welche von dem Interface A erbt und dessen Schnittstelle implementiert. Y ist eine Spezialisierung von X und soll zu diesem Zweck von X erben. Y soll zusätzlich zu Interface A auch Interface B anbieten. Y erbt folglich von B und implementiert dessen Interface. COM unterstützt jedoch nicht, wie in Abbildung 11 auf der linken Seite angedeutet, daß Y von X erbt und damit das Interface A bereits implementiert ist. Statt dessen muß Klasse Y beide Interfaces erben und beide implementieren oder die Implementation des Interfaces A an ein Exemplar der Klasse X delegieren.

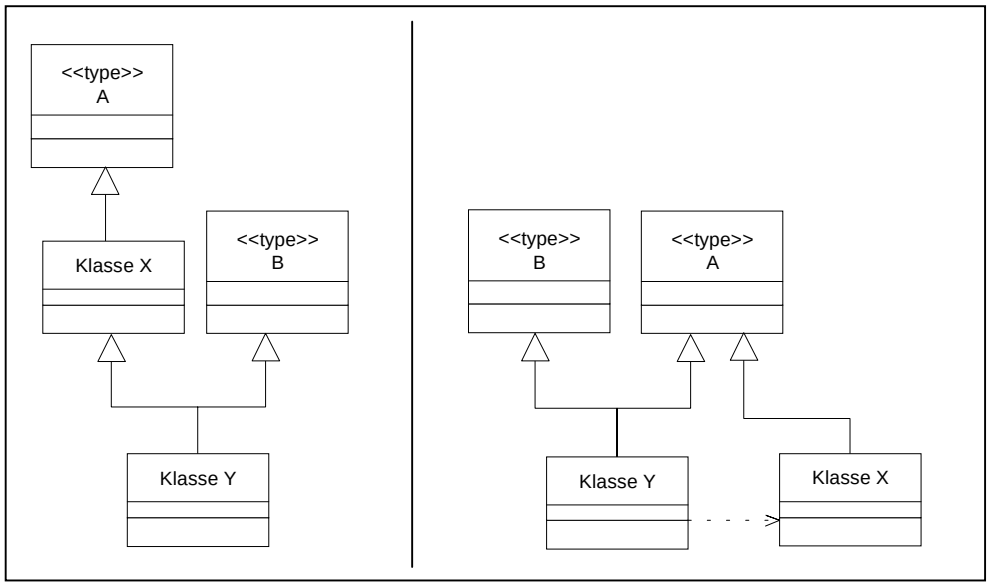


Abbildung 11 Vererbung von Interfaces

Die Konstruktion, wie sie in Abbildung 11 rechts aufgeführt ist, wird durch COM unterstützt. Die Klasse, welche Interfaces wiederverwenden möchte, benutzt die Implementation dieser Interfaces durch andere Klassen. Möchte ein Kunde dieses Interface benutzen, so erfragt er es über *QueryInterface*. Er kann nun Methoden dieses Interfaces nutzen.

Brockschmidt nennt in seinem Buch eine Reihe von Möglichkeiten, wie COM Objekte mit Interfaces ausgestattet werden können. Ich habe sie in der Abbildung 12 in Beziehung zueinander gesetzt. Interfaces lassen sich ganz allgemein unterteilen in solche, deren Implementation nicht wiederverwendbar ist und solche, die dieses ermöglichen. Letztere werde ich im weiteren Verlauf genauer beleuchten, da die Wiederverwendung, wie bereits angedeutet, nicht über Vererbung

möglich ist und deshalb der benutzte Mechanismus meines Erachtens etwas genauer betrachtet werden sollte.

Zunächst jedoch möchte ich die Verwendung von Interfaces betrachten, dessen Implementationen nicht wiederverwendet werden können. Zum einen handelt es sich dabei um Mehrfachvererbung. Die Komponente erbt von verschiedenen Interfaces und implementiert deren Methoden. Dieses ist vergleichbar mit dem Mehrfacherben von abstrakten Klassen. Möchte eine Komponente ein Interface zur Verfügung stellen, so kann sie nicht durch Vererbung auf eine andere Implementation zurückgreifen, sondern muß das Interface jedesmal von neuem implementieren. Im Bereich der „enthaltenen Klassen“ liegt der Fall etwas anders.[29] Hier werden Klassen ineinander geschachtelt. Innerhalb der Klasse der Komponente wird eine Klasse deklariert, welche von dem gewünschten Interface erbt. Außerdem wird innerhalb der Komponente jeweils ein Exemplar der besagten Klasse deklariert. Insbesondere hinsichtlich der Lebenszeit der Interfaceimplementierungen ist dieses Konzept einfach gehalten, da sie mit der Lebenszeit der Komponente endet. Diese Problematik wird im weiteren Verlauf der Interfacediskussion klarer werden.

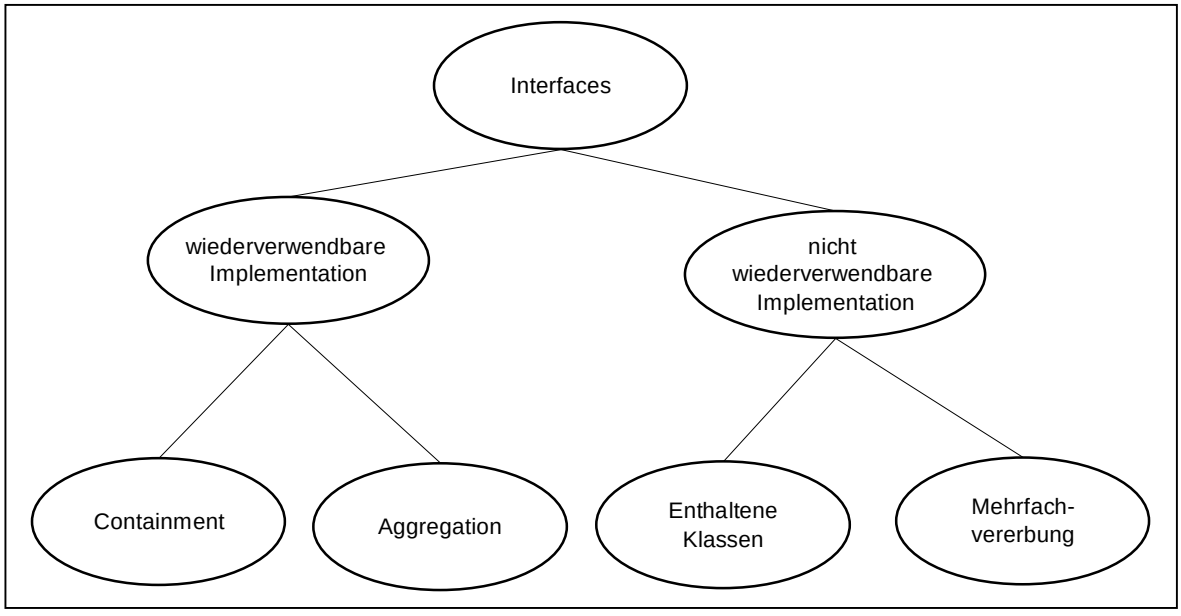


Abbildung 12 Interfaces bei COM

[29] Den Begriff „enthaltene Klassen“ habe ich direkt aus dem Englischen übernommen. In der Literatur werden Begriffe wie „nested classes“ [Gos95] oder „contained classes“ [Bro95] benutzt.

COM stellt zwei verschiedene Ansätze zur Verfügung, wie die Implementation eines Interfaces wiederverwendet werden kann. Beispielsweise würde ein Kunde von einem Objekt Y das Interface A erfragen. Der erste Ansatz ist der des „Containment“. Y implementiert das Interface A. Dem Wunsch des Kunden wird dadurch entsprochen, daß ihm das A Interface von Y übergeben wird. Die Aufrufe des Kunden gehen an die Implementation des A Interfaces von Y. Von dort werden sie an X delegiert. Abbildung 13 zeigt einen Kunden, der ein Objekt vom Typ Y benutzt, welches über Containment das Interface von X wiederverwendet.

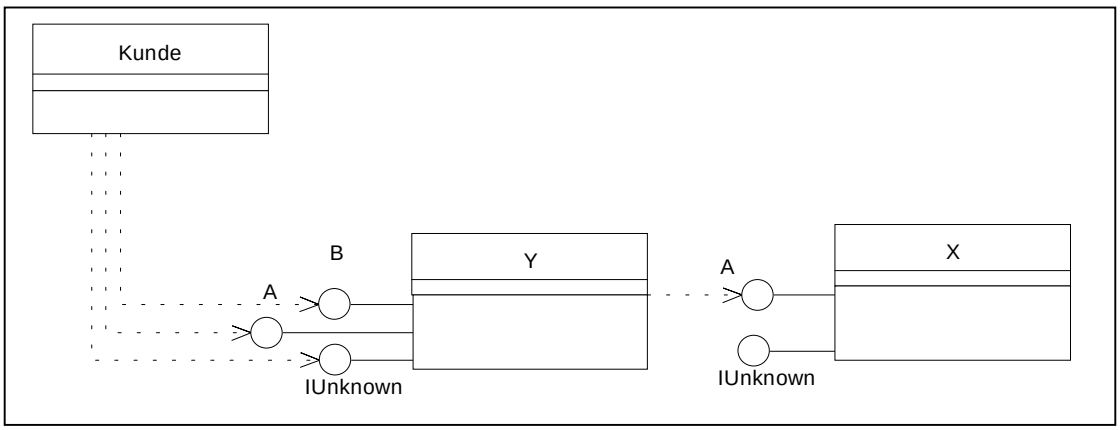


Abbildung 13 Wiederverwendung von Interfaces über Containment

Aufrufe an das Interface A werden an X delegiert. Y muß also sowohl sein eigenes Interface B implementieren, als auch das Interface A.

Der zweite Ansatz ist die Aggregation. Y liefert dem Kunden auf seine Anfrage hin das A Interface von X. Abbildung 14 illustriert die Aggregation. Dem Kunden wird direkt das Interface A von X zur Verfügung gestellt. Es resultiert das Problem, daß Y Aufrufe über A an X nicht bekannt sind. Dieses wirft insbesondere Probleme im Hinblick auf die Benutzung der Methoden des *IUnknown* Interfaces auf, da Y die Lebenszeit von X kontrollieren muß.

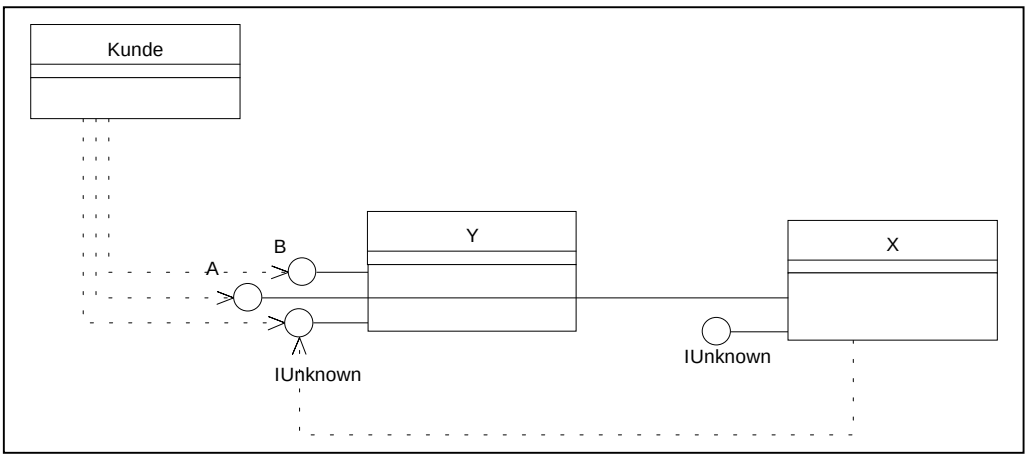


Abbildung 14 Wiederverwendung von Interfaces über Aggregation

Dieses Problem wird bei der Aggregation dadurch umgangen, daß benutzende Klassen ihr *IUnknown* Interface bei den Klassen, die das Interface implementieren, anmelden. So werden Aufrufe an das Interface des benutzten Objektes an das benutzende Objekt delegiert. Es fällt auf, daß im Falle der Aggregation das benutzte Objekt „weiß“, ob es Teil eines benutzenden Objektes ist oder nicht.

6.3.3 Komponenten in verschiedenen Prozeßräumen

Komponenten können sich in beliebigen Prozeßräumen befinden. Um die Kommunikation gewährleisten zu können, bedient sich COM des Proxy Konzeptes, wie es beispielsweise bei Gamma et. al. beschrieben ist:

> „A remote proxy provides a local representative for an object in a different address space.“ ([GJH+95], S. 208)

Hier wird eine Komponente, die sich in einem anderen Prozeßraum befindet, durch einen Stellvertreter repräsentiert, welcher sich genau so verhält, als wäre er das entfernte Objekt selbst. Aufrufe an diesen Stellvertreter leitet derselbe über ein geeignetes Netzwerkprotokoll an einen Empfänger im Prozeßraum der zu benutzenden Komponente weiter. Dieser Empfänger wird üblicherweise als „Stub“ bezeichnet. Der Stub ruft dann die Komponente selbst. Ergebnisse der Operation werden über den Stub wieder zurück an das Proxy gesandt, welches dann dieses Ergebnis als Resultat der Operation an den Aufrufenden zurückliefert. Die Kommunikation eines Kunden mit einer Komponente findet über ein solches Proxy statt. Abbildung 15 zeigt die Struktur bei der Benutzung eines COM Objektes.

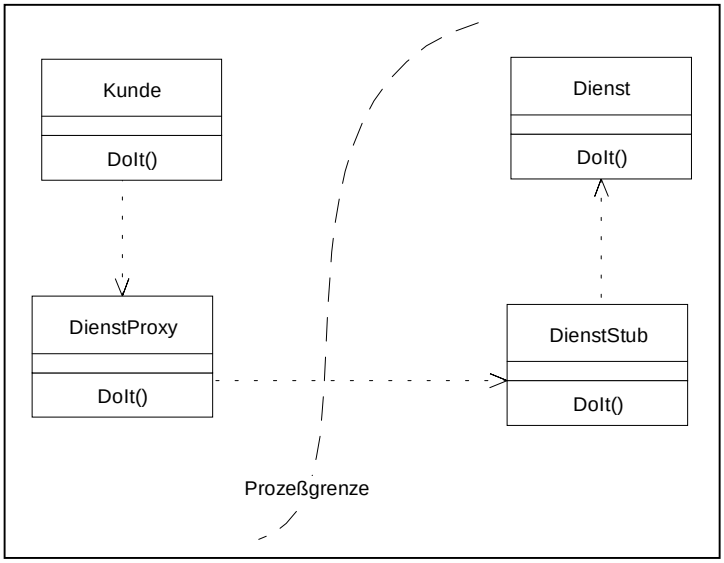


Abbildung 15 Proxys und Stubs bei COM

Der Aufruf an dieses Objekt wird lediglich durch das Proxy über einen Mechanismus, welcher „Marshalling“ genannt wird, an einen Stub delegiert, welcher sich im Prozeßraum des Objektes befindet. Dort findet dann der Aufruf an das Objekt statt. Interfaces werden in der „Interface Definition Language“ (=IDL) spezifiziert.[30] Im späteren Verlauf werde ich ein Werkzeug beschreiben, welches die notwendigen Proxies und Stubs für die Entwicklung von Komponenten automatisch erzeugt.

Proxies, Stubs und Skeletons

Ich möchte hier kurz die mißverständliche Verwendung der Begriffe Proxy, Stub und Skeleton bei OLE und CORBA klarstellen. Geht man von einer benutzenden Komponente (Client) und einer benutzten Komponente (Server) aus, so wird bei COM die Kommunikationskomponente auf der Serverseite als **Stub** bezeichnet, während das Pendant auf der Clientseite als **Proxy** bezeichnet wird. Bei CORBA hingegen wird die Kommunikationskomponente auf der Serverseite als **Skeleton** bezeichnet und das Pendant auf der Clientseite als **Stub**.

6.3.4 Erzeugung von Komponenten

Die Erzeugung von Komponenten wird durch eine Fabrik ermöglicht. Diese Fabrik ist wiederum eine COM Komponente mit dem Interface *IClassFactory*. Sie verfügt über die Möglichkeit, COM Komponenten anhand des eindeutigen Identifikators zu erzeugen. Der Benutzer wird allerdings mit diesen Dingen nicht konfrontiert. Er ruft lediglich eine öffentliche Methode von COM und bekommt im Erfolgsfalle das gewünschte Interface der durch die Fabrik erzeugten Komponente geliefert.

6.3.5 DCOM

COM ermöglicht die Benutzung von Komponenten in verschiedenen Prozeßräumen. Diese Prozeßräume müssen sich jedoch alle auf einer Maschine befinden. COM ermöglicht nicht die Benutzung von Komponenten, die sich auf einem beliebigen Rechner innerhalb eines Netzwerkes befinden. Ende 1996 wurde COM um das „Distributed Component Object Model“ (= DCOM) erweitert. Es handelt sich hierbei um eine Erweiterung auf Ebene des Betriebssystems. DCOM ermöglicht, daß Komponenten unabhängig von ihrem wirklichen Aufenthaltsort innerhalb eines Netzwerkes zur Verfügung stehen. Komponenten, die nach der COM Spezifikation entwickelt

[30] Die IDL wurde von OSF/DCE entwickelt und von Microsoft im wesentlichen übernommen.

werden, sind nach der Konfiguration des Betriebssystems gleichermaßen im Rahmen von DCOM benutzbar.

> „To all intents and purposes, the act of distribution is transparent to both the consumer and the provider.“ [Mic97a]

Der Begriff „transparent“ ist in diesem Zusammenhang als technische Interpretation zu sehen und bezeichnet damit die Tatsache, daß die Verteilung nicht sichtbar ist. Eine Anpassung von Komponenten entfällt also bei der Erweiterung auf DCOM.

6.3.6 COM Zusammenfassung

Ich habe vorgestellt, wie COM die Infrastruktur bereitstellt, um komponentenbasierte Softwareentwicklung betreiben zu können. Ich habe darüber hinaus die verschiedenen Möglichkeiten der Wiederverwendung vorgestellt und erklärt, wie Komponenten erzeugt und benutzt werden. Ebenfalls habe ich aufgezeigt, in welchem Zusammenhang COM zu DCOM steht. Auf dem Konzept von COM bauen alle weiteren von OLE angebotenen Dienste auf. Ich werde nun zeigen, wie die Automation COM benutzt und welche Folgen dieses hat.

6.4 OLE Automation

Im bisherigen Verlauf des Kapitels 6 haben ich vorgestellt, wie COM die Möglichkeit eröffnet, Komponenten über Interfaces unabhängig von ihrer Implementationssprache und ihres Aufenthaltsortes zu benutzen. Automation geht über dieses Konzept hinaus. Sie ermöglicht die Benutzung von beliebigen Komponenten über ein einziges Interface mit dem Namen *IDispatch*. Dieses Interface bietet Metainformationen über die Methoden einer Komponente an und ermöglicht zur Laufzeit den Zugriff darauf. Komponenten, die das *IDispatch* Interface anbieten, werden Automationsserver genannt. Ich werde das *IDispatch* Interface näher betrachten und in Kapitel 6.4.2 zeigen, woher diese Metainformationen kommen.

6.4.1 Das IDispatch Interface

Das *IDispatch* Interface bietet Metainformationen über die Methoden einer Komponente an.

```
Invoke (...)
GetIDsOfNames (...)
GetTypeInfoCount (...)
GetTypeInfo (...)
```

Abbildung 16 Schnittstelle von IDispatch

Die Methode *GetTypeInfoCount* zeigt an, ob Metainformationen zu den Methoden verfügbar sind. Ist dieses der Fall, so lassen sich über *GetTypeInfo* die Informationen über Methodennamen, Formalparameter etc. erfragen. *Invoke* ermöglicht letztlich den Aufruf einer Methode. Die Parameter der einzelnen Methoden hier zu betrachten wäre einerseits sehr umfangreich und würde andererseits das Verständnis nicht verbessern. Deshalb und insbesondere weil es ein Werkzeug gibt, welches die Handhabung dieses Interfaces stark vereinfacht, verzichte ich hier auf eine genauere Betrachtung.[31] Wichtig ist meines Erachtens das Verständnis für die Art und Weise, wie auf diese Metainformationen zugegriffen werden kann. Im folgenden Kapitel wird deutlich werden, wann und wo diese Metainformationen entstehen.

6.4.2 Metainformationen aus Type Libraries und der ODL

Ich habe bereits in Kapitel 6.3.3 erwähnt, daß das Interface einer Komponente in der IDL beschrieben wird. Um allerdings die Metainformationen des *IDispatch* Interfaces verfügbar machen zu können, wurde die IDL erweitert. Es entstand die „Object Definition Language“ (= ODL). Der Entwickler beschreibt die Methoden seiner Komponente mit der ODL. Aus der ODL wird dann eine Type Library generiert. Benutzer eines Automationsservers können entweder über das *IDispatch* Interface die Methodeninformationen erfragen und diese benutzen, oder sie können, basierend auf den in der Type Library enthaltenen Informationen, das Interface nutzen. Letztere enthält die Metainformationen, die zur Laufzeit durch *IDispatch* zur Verfügung gestellt werden.

6.5 Kommunikation in zwei Richtungen

Ich möchte an dieser Stelle ein weiteres Interface vorstellen, welches in meinen Augen sehr wichtig und sinnvoll ist. Es handelt sich dabei um das Interface *IConnectionPoint.* Ich habe bisher gezeigt, wie über ein Interface Verbindung zu einer Komponente aufgenommen werden kann. Das *IConnectionPoint* Interface ermöglicht nun weiterhin, auch eine Verbindung in die entgegengesetzte Richtung aufzubauen. Ein Kunde kann sich bei einer Komponente als Empfänger von Nachrichten registrieren. Ich werde diese Verbindung im Folgenden als Verbindungspunkt bezeichnen. Das Aufbauen einer solchen Verbindung setzt voraus, daß der Empfänger über ein Empfangsinterface verfügt, das Automationsobjekt dieses kennt und einen Verbindungspunkt für ein solches Interface zur Verfügung stellen kann. Verbindungspunkte werden immer in Verbindungspunktbehältern aufbewahrt. Über dieses *IConnectionPointContainer* Interface kann der

[31] Genauere Informationen zum *IDispatch* Interface finden sich in [Bro95], [Rog97] und [OHE96].

Kunde herausfinden, ob das Automationsobjekt einen Verbindungspunkt für das gewünschte Interface zur Verfügung stellt.

```
FindConnectionPoint (IID,...)
EnumConnectionPoints (...)
```

Abbildung 17 Das Interface "IConnectionPointContainer"

Sofern der Verbindungspunkt über *FindConnectionPoint* gefunden wird, bekommt der Kunde das Interface des Verbindungspunktes geliefert.

```
Advise (IUnknown,...)
Unadvise (...)
EnumConnections (...)
GetConnectionPointContainer (...)
GetConnectionInterface (...)
```

Abbildung 18 Das Interface „IConnectionPoint“

Der Kunde kann sich nun über *Advise* bzw. *Unadvise* an- und abmelden. Abbildung 19 zeigt die Anmeldung eines Verbindungspunktes. Es handelt sich hier um einen Beobachter, der von einem bestimmten Ereignis unterrichtet werden möchte. Aus der Abbildung ist nicht eindeutig ersichtlich, daß der Kunde über ein Interface mit dem Identifikator IID_EINBEOBACHTER verfügt. Dieses meldet er bei dem Verbindungspunkt an, der nur genau dieses Interface akzeptiert. Da der Verbindungspunkt beim *Advise* nur ein *IUnknown* Interface übergeben bekommt, versichert er sich mittels *QueryInterface*, daß derjenige, der sich anmeldet, auch tatsächlich dieses Interface unterstützt.

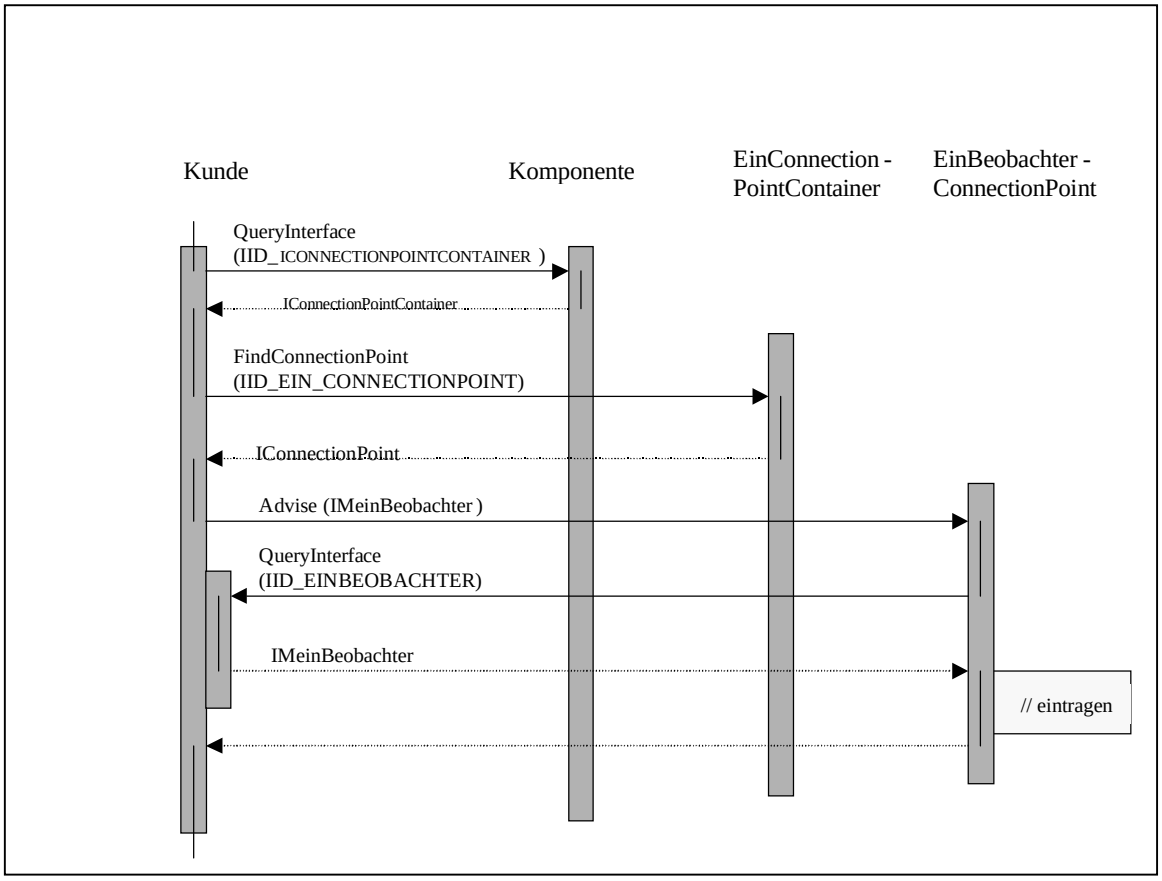


Abbildung 19 Interaktionen bei der Anmeldung eines Verbindungspunktes

Das Automationsobjekt kann nun, sofern notwendig, mit *EnumConnections* über den angemeldeten Interessenten iterieren und eine Benachrichtigungsmethode des angemeldeten Interfaces aufrufen.

Dieses Konzept hat auf den ersten Blick eine gewisse Ähnlichkeit mit dem Beobachtermuster von Gamma et al.. Vlissides identifiziert dieses in [Vli97a] jedoch als ein neues Muster mit dem Namen „Multicast“. In seinem Artikel erwähnt er allerdings, daß die „Gang of Four“[32] noch nicht zu einem Ergebnis gekommen ist, ob es sich hier um ein neues Muster handelt, oder ob es lediglich eine Variation des Beobachtermusters ist.

6.6 OLE und die MFC

Nachdem ich nun sowohl OLE als auch die MFC eingeführt habe, möchte ich nun den Zusammenhang zwischen diesen beiden aufzeigen. Die MFC dient als Rahmenwerk zur Anwendungsentwicklung. Ein Entwickler wird die anwendungsfachliche Funktionalität in dieses Rahmenwerk integrieren, um zu dem gewünschten Anwendungssystem zu kommen. Wenn er sich im Rahmen der Entwicklung des Systems die Dienste von OLE zunutze machen möchte, so muß er dieses

[32] Bei der „Gang of Four“ handelt es sich um die Autoren des Buches [GJH+95]: Erich Gamma, Richard Helm, Ralph Johnson und John Vlissides.

frühzeitig berücksichtigen. Insbesondere kann dieses Auswirkungen auf die Vererbungshierarchie innerhalb seiner Anwendung haben. Die MFC hingegen verfügt in ihrer Klassenhierarchie bereits über die Unterstützung von OLE Diensten. Notwendige Beziehungen und Eigenschaften sind in des Rahmenwerk eingepflegt. Der Entwickler kann OLE Dienste benutzen ohne dadurch einem Mehraufwand an Entwicklung gegenüber zu stehen. Wie diese Unterstützung aussieht, werde ich im Folgenden vorstellen.

Die MFC unterstützt bereits die Entwicklung von Automationsobjekten. Zu diesem Zweck gibt es in der Vererbungshierarchie eine Klasse, die über ein *IDispatch* Interface verfügt. Ich habe bereits ausgeführt, daß sie zu diesem Zweck nicht von *IDispatch* erbt, sondern eine Klasse benutzt, die das *IDispatch* Interface implementiert. Sie heißt *CCmdTarget*. Diese Klasse ist innerhalb der MFC die Oberklasse, die es ermöglicht Nachrichten zu empfangen.[33] Der Entwickler spezialisiert diese Klasse, um ein Automationsobjekt zu definieren. Ein *IDispatch* Aufruf an ein Objekt dieser Klasse wird von der Oberklasse *CCmdTarget* entgegengenommen und an die spezialisierten Methoden des Objektes delegiert.

Zusätzlich dazu gibt es auf der Kundenseite ebenfalls eine Unterstützung. Möchte der Entwickler ein Automationsobjekt in seiner Anwendung benutzen, so muß er zu diesem Zweck das *IDispatch* Interface des Automationsobjektes erfragen, die Parameter zusammenstellen und über die Methode *Invoke* die gewünschte Methode aufrufen. Alternativ dazu kann er sich aber auch der Klasse *ColeDispatchDriver* bedienen, welche ihm einen Teil der notwendigen Arbeit abnimmt. Sie ermöglicht die unkomplizierte Herstellung einer Verbindung zum Automationsobjekt und übernimmt darüber hinaus die automatische Referenzzählung auf das Automationsobjekt. Abbildung 20 zeigt die grundsätzliche Struktur auf.

[33] In der MFC gibt es nur relativ wenige virtuelle Methoden, die von abgeleiteten Klassen überschrieben werden. Jedoch ist es möglich, Methoden in eine Nachrichtentabelle einzutragen, um sich bei Ereignissen durch das Rahmenwerkes rufen zu lassen. Näheres dazu findet sich in [Kra98].

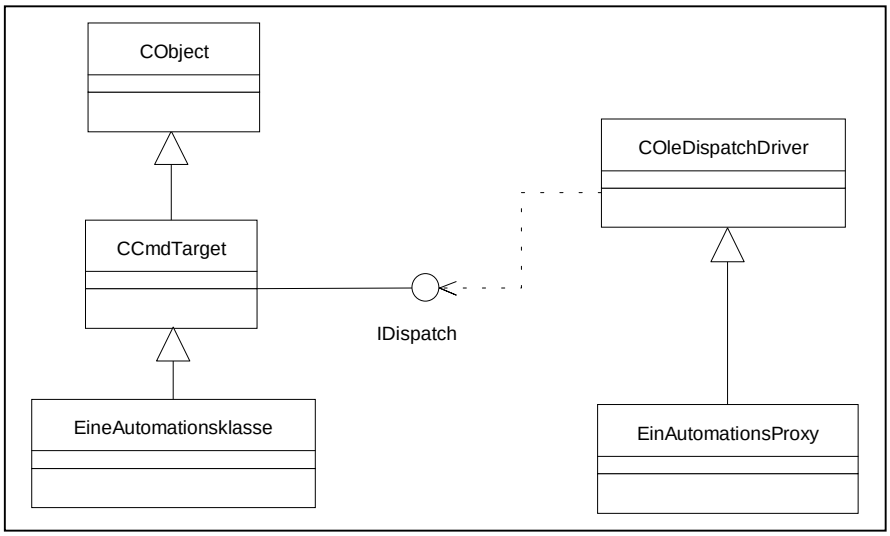


Abbildung 20 Vererbungsstruktur der MFC für OLE Automation

Bei der Definition der Klasse „EineAutomationsklasse“ legt der Entwickler fest, welche Methoden über Automation gerufen werden können. Diese Information wird einerseits in der Type Library vermerkt und andererseits in der Klasse selbst festgehalten. Das Automationsproxy, welches nun eine Methode aufrufen möchte, tut dieses über eine Methode von *ColeDispatchDriver*. Dieser ruft *Invoke*, die Methode vom *IDispatch* Interface, die über Parameter festlegt, welche Methode des Automationsobjektes gerufen werden soll. *CCmdTarget* entschlüsselt diese Information und ruft die betreffende Methode des Automationsobjektes. In nachfolgendem Kapitel werde ich die Dynamik dieser Zusammenarbeit verdeutlichen.

6.7 Dynamische Aspekte der Automation

Um den Eindruck von der Automation zu vervollständigen, möchte ich an dieser Stelle aufzeigen, wie sich ein System zur Laufzeit verhält. Ich gehe von dem Fall aus, daß ein Kunde die Erzeugung eines Automationsservers wünscht. Abbildung 21 zeigt die Interaktionen der beteiligten Komponenten. Nach erfolgreicher Erzeugung bekommt er von dem OLE API das *IDispatch* Interface dieses Automationsservers. Er weist dieses *IDispatch* Interface per Copy Konstruktor auf das AutoObjektProxy zu. Danach ruft er die Methode DoIt(). Diese Methode ruft eine Methode von der Oberklasse *ColeDispatchDriver*. Sie stellt die notwendigen Parameter zusammen und ruft per *Invoke* den hier nicht eingezeichneten *DispatchStub* von AutoObjekt.

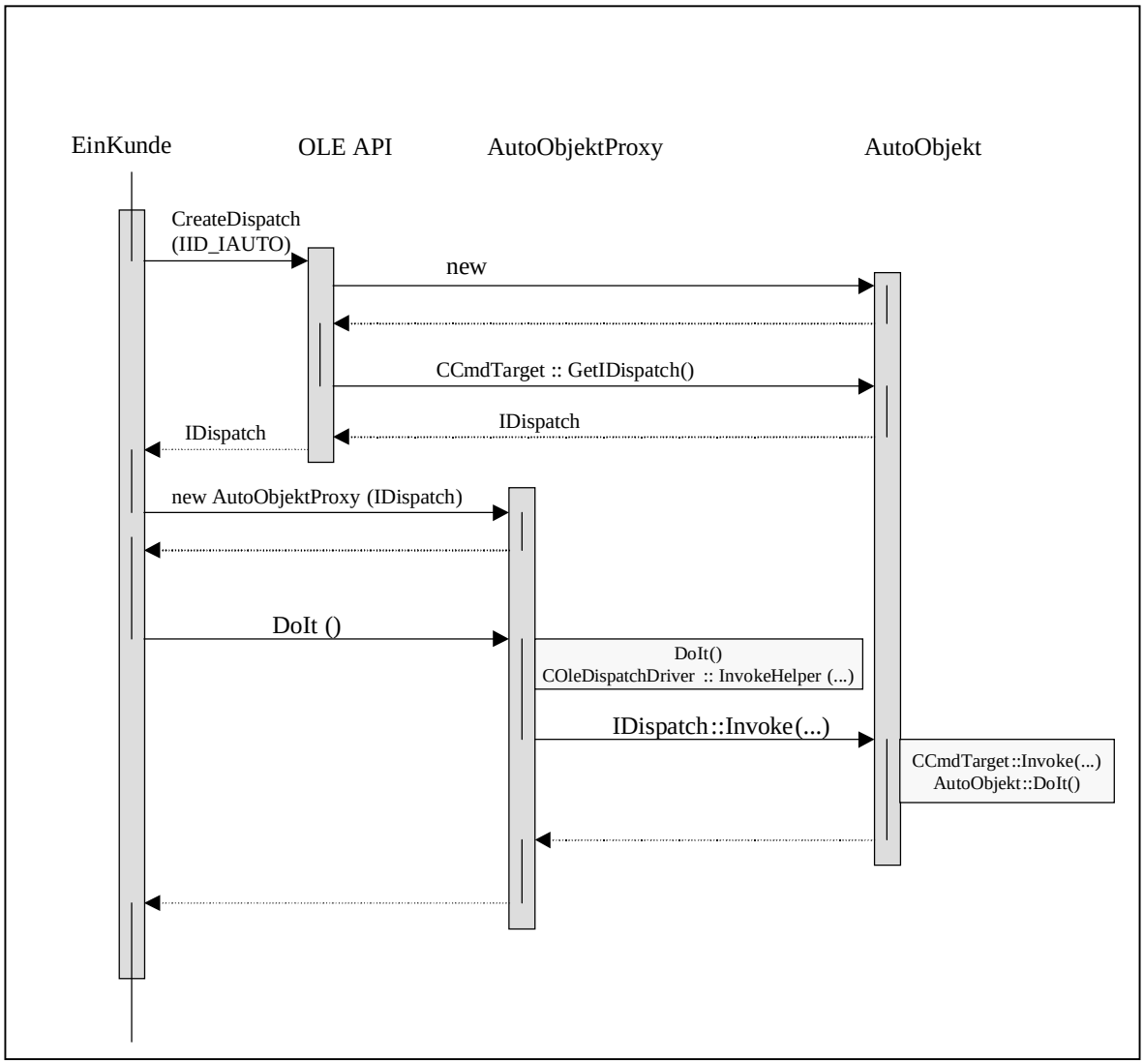


Abbildung 21 Ein Kunde erzeugt und benutzt einen Automationsserver

Dieser delegiert den Aufruf weiter an die Methode *Invoke*, welche in der Klasse *CCmdTarget* implementiert ist. Über die Nachrichtentabelle wird dort die Methode DoIt() identifiziert und aufgerufen.

6.8 Visual C++ : OLE und die MFC in der Entwicklungsumgebung

Nach der Vorstellung der Rahmenwerke möchte ich abschließend dem Leser einen Eindruck davon vermitteln, wie sie im Rahmen der Entwicklungsumgebung benutzt werden. Meines Erachtens spielt neben den Rahmenwerken als solchen auch die Benutzbarkeit für den Entwickler eine wichtige Rolle. Dieses betrifft insbesondere die Entwicklungsumgebung und zur Verfügung gestellte Werkzeuge. Es handelt sich bei der Entwicklungsumgebung um das Microsoft Visual Studio. Sie unterstützt das Entwickeln mit unterschiedlichen Programmiersprachen unter einer Oberfläche. Bei den weiteren Ausführungen beziehe ich mich auf die Werkzeuge, die im Rahmen

der Entwicklung von Anwendungen mit Visual C++ Version 5.0 angeboten werden. Aus pragmatischen Gründen werde ich den Begriff der Entwicklungsumgebung, in Anlehnung an den englischen Ausdruck „Integrated Development Environment", mit IDE abkürzen. Obwohl die bisher vorgestellten Techniken relativ kompliziert sind, ist die Entwicklung von OLE Funktionalität erheblich einfacher, da sie durch verschiedene Werkzeuge unterstützt wird. Hinzu kommen eine Vielzahl von Makros, welche einem Entwickler nicht geringfügig Schreibarbeit ersparen und, sofern er ihre Bedeutung verstanden hat, ebenfalls die Übersichtlichkeit erhöhen.

6.8.1 Entwicklung eines Automationsobjektes

Entscheidet sich der Entwickler dazu, ein Automationsobjekt zu implementieren, so teilt er dieses der IDE durch ein Werkzeug mit dem Namen „Class Wizzard" mit. Dieser erzeugt automatisch ein Codeskelett einer Klasse, welche eine Spezialisierung von *CCmdTarget* ist. Darüber hinaus ermöglicht er weiterhin, die Methoden, welche diese Klasse über Automation zur Verfügung stellen soll, zu definieren. Die eingegebenen Informationen werden zum einen in eine ODL Datei geschrieben, aus der später die Type Library entstehen wird und zum anderen werden sie ebenfalls in das bereits angefertigte Klassenskelett integriert. Innerhalb des Codes befinden sich bereits Aufrufe, die zur Initialisierung von OLE notwendig sind. Die IDE sorgt weiterhin für die Erstellung eines Registrierungsscripts um das Automationsobjekt beim Betriebssystem anmelden zu können. Gegebenenfalls gewünschte Verbindungspunkte werden über ein Deklarations- und ein Implementationsmakro implementiert. Diese sorgen für das Ausstatten mit den notwendigen Interfaces sowie für eine Initialisierung der Nachrichtentabellen. Das Übersetzen des Codes sorgt für die Erstellung der Type Library aus dem in der ODL deklarierten Interface. Das Automationsobjekt kann nun benutzt werden.

6.8.2 Benutzung eines Automationsobjektes

Möchte der Entwickler in seiner Anwendung Automationsobjekte benutzen, so kann er dieses zur Laufzeit über seine Methoden befragen oder er legt die Benutzung mittels der dazu gehörige Type Library zur Entwicklungszeit fest. In letzterem Falle teilt er dem Class Wizzard mit, daß er aus einer Type Library ein Automationsobjekt benutzen möchte. Der Class Wizzard generiert daraufhin ein Proxy, indem er den Code einer Spezialisierung von der Klasse *COleDispatchDriver* generiert. Die Methodenimplementation und die Aufbereitung der Parameter ist in dem Code dieser Klasse bereits vorgenommen. Danach kann der Entwickler das Automationsobjekt über das Proxy in seiner Anwendung benutzen. Die direkte Konfrontation mit *IDispatch* Aufrufen bleibt ihm vollständig erspart.

6.9 Zusammenfassung und Ausblick

Ich habe in diesem Kapitel versucht, den Leser in die Rahmenwerke MFC und OLE einzuführen. Dabei habe ich mich auf die Teile beschränkt, welche im weiteren Verlauf der Arbeit relevant sind. Die Einführung beschränkte sich auf COM und OLE Automation. Weiterhin habe ich angedeutet, wie OLE in die MFC integriert ist und welche Konsequenzen dieses für die Entwicklung von verteilter Software hat. Darüber hinaus habe ich die Entwicklung einer Automationskomponente nachgezeichnet, um dem Leser zu verdeutlichen, wie sich diese Arbeit gestaltet.

Die Ziele der Arbeit umfassen neben der Aufstellung eines Konzeptes zur Entwicklung von Softwaresystemen, welche die Kooperation an einem Ort unterstützen sollen, ebenso die Verifizierung der Benutzbarkeit desselben. Das Pausenplanersystem ist mit Hilfe des Konzeptes der Gruppenarbeitsumgebung entwickelt worden und stützt sich auf die Rahmenwerke MFC und OLE. Die bisher vorgestellten Ergebnisse sind in die Entwicklung des Pausenplanersystems eingeflossen und werden im folgenden Kapitel, der Vorstellung des Pausenplanersystems, verwendet werden.

7 Fachliche Aspekte des Pausenplanersystems

Ein methodisches Vorgehen bei der Softwareentwicklung im Rahmen der WAM Methode empfiehlt die Verwendung von verschiedenen Dokumenten. Sie dokumentieren den Entwicklungsprozeß, dienen dem Verständnis der beteiligten Personen und definieren Erkenntnisse, Vorstellungen und Ziele im Hinblick auf das zu erstellende Softwaresystem.

Für die Entwicklung des Pausenplanersystems wurden uns durch den Arbeitsbereich Softwaretechnik der Universität Hamburg Szenarios und CRC Karten zur Verfügung gestellt. Wir haben diese durch ein Glossar, weitere Szenarios und eine Vielzahl von Visionen ergänzt. Diese Dokumente befinden sich im Anhang dieser Arbeit. Die Visionen geben einen Überblick über das System sowie einen detaillierten Einblick in dessen Handhabung. Hinzu kommen verschiedene Werkzeug- und Materialvisionen.

Da es sich bei dem Pausenplanersystem um eine Software handelt, welche eine Gruppe von Personen bei einer kooperativen Aufgabe unterstützt, habe ich die im Anhang befindlichen Visionen, welche die Handhabung des Systems beschreiben, in Kapitel 4 zusätzlich um zwei weitere ergänzt, welche im speziellen die Kooperationssituation der beteiligten Personen beschreiben. Meines Erachtens ist damit der Dokumentation des Entwurfs ausreichend genüge getan. Nichts desto trotz möchte ich nochmals die konkrete Kooperationssituation im Pausenplanersystem näher betrachten, um dann weitere Ansätze zur Unterstützung der Kooperationssituation aufzuzeigen, welche das Pausenplanersystem nicht realisiert.

7.1 Die Kooperationssituation im Pausenplanersystem

Die Gruppenarbeitsumgebung innerhalb des Pausenplanersystems wird repräsentiert durch ein schwarzes Brett, an dem die Materialien hängen, die im Rahmen der Aufgaben der beteiligten Personen bearbeitet werden. Die Personen, die sich am schwarzen Brett befinden, werden jederzeit über die Präsenz anderer Benutzer informiert. Zusätzlich dazu werden die Benutzer über die Tätigkeiten anderer Mitarbeiter informiert. Dieses geschieht dadurch, daß die Materialsymbole, die sich am schwarzen Brett befinden, die Namen der Personen aufweisen, welche dieses Material gegenwärtig bearbeiten. Auf eine stärkere Annäherung an das WYSIWIS Konzept haben wir hier verzichtet, da es unserer Meinung nach für die Kooperationssituation nicht hilfreich ist.[34] Wir haben damit ein ausreichendes Maß der Transparenz für die Benutzer verwirklicht. Die Anzahl der Mitarbeiter innerhalb dieser Gruppe von Personen ist gering. Ein Benutzer, welcher feststellt, daß

[34] Vergleiche dazu Kapitel 3.4.4.

bereits eine andere Person am schwarzen Brett zugegen ist, kann unmittelbar antizipieren, welche Tätigkeiten diese Person ausführt.

Die Benutzer können uneingeschränkt die Materialien des schwarzen Brettes bearbeiten. Einzige Ausnahme bildet hier der Pausenplan, welcher nicht durch den Sekretariatsmitarbeiter bearbeitet werden darf. Diese Tätigkeit obliegt ausschließlich dem Pausenplanersteller. Wir haben also lediglich fachliche Restriktionen in die Realisierung des Pausenplanersystems einfließen lassen. Andere Restriktionen hinsichtlich der Bearbeitung von Materialien haben wir als nicht adäquat für diese Kooperationssituation eingestuft.

Die Möglichkeit des simultanen Materialzugriffs wirkt sich nicht negativ auf die Kooperationssituation aus. Vielmehr wurde deutlich, daß unter bestimmten Umständen diese Möglichkeit eher förderlich für die effiziente Arbeit der beteiligten Personen sein kann.

7.2 Erweiterung der Kooperation

Ich habe bei der eingehenden Betrachtung der fachlichen Kooperationsaspekte in Kapitel 3.7 das Konzept der „double level language“ von Robinson angedeutet. Die implizite und die explizite Kommunikation der beteiligten Kooperationspartner fördern die Fähigkeit der Koordination in unerwarteten Arbeitssituationen. Das Pausenplanersystem sieht eine explizite Kommunikationsmöglichkeit über das Softwaresystem nicht vor. Sie kann jedoch entweder durch andere Softwaresysteme gewährleistet werden, oder das Telefon fungiert als ein Medium zur expliziten Kommunikation zwischen den Kooperationspartnern.

Eine weitere meines Erachtens sinnvolle Erweiterung sehe ich in der realitätsgetreueren Übertragung des schwarzen Brettes. Robinson erkennt die Notwendigkeit, die Dinge des täglichen Lebens mit ihren Eigenschaften möglichst bruchlos in ein Softwaresystem zu übertragen, um den Menschen die Möglichkeit zu geben, mit Hilfe dieser Dinge eine eigene Form der Kooperation herauszubilden. So wäre es bei einem schwarzen Brett durchaus naheliegend, auch andere Dinge, wie beispielsweise Notizzettel, an diesem befestigen zu können, um somit die beteiligten Personen von Umständen oder Änderungen in Kenntnis zu setzen, die nicht ohne weiteres erkennbar sind.

8 Der technische Entwurf des Pausenplanersystems

In den vorangegangenen Kapiteln habe ich die Kooperation an einem Ort motiviert sowie technische Voraussetzungen für die Erstellung von Software zur Unterstützung dieser Kooperationsform diskutiert. Das Pausenplanersystem wird fachlich im Anhang anhand verschiedener Dokumente des objektorientierten Entwicklungsprozesses vorgestellt. In diesem Kapitel möchte ich die technische Realisierung des Pausenplanersystems genauer beleuchten. Im Verlaufe des Kapitels wird deutlich, daß ich für bestimmte Problembereiche Lösungen erarbeitet habe, die ich in der bisherigen Diskussion als suboptimal identifiziert habe. Der Grund dafür liegt in dem Schwerpunkt dieser Arbeit. Neben der Untersuchung von Kooperation an einem Ort möchte ich untersuchen, wie die Entwicklung von verteilter Software mit Hilfe der OLE/COM Architektur realisiert werden kann. Ich habe also versucht, vorwiegend mit Lösungen, die mir im Rahmen dieser Architektur zur Verfügung stehen, zu arbeiten. An den betreffenden Stellen werde ich auf diese Entscheidungen hinweisen.

Kapitel 8.1 gibt ein Überblick über alle am System beteiligten Komponenten. Die nachfolgenden Kapitel behandeln die einzelnen Bereiche Gruppenarbeitsumgebung, Werkzeuge und Materialien. Dann gehe ich näher auf die Art und Weise ein, wie der Zugriff auf Material im Pausenplanersystem realisiert ist. Neben der Kommunikation wird im folgenden Kapitel das Benachrichtigungskonzept vorgestellt. Um dem Leser aus den bis dahin präsentierten Fragmenten einen Gesamteindruck zu vermitteln, folgt eine technische Ablaufvision, welche die betrachteten Aspekte vereint.

8.1 Die Komponenten des Pausenplanersystems

Die Visionen des Pausenplanersystems im Anhang dieser Arbeit geben einen detaillierten Einblick in die im System befindlichen Werkzeuge und Materialien. Ich möchte hier einen Überblick darüber geben, wie diese Komponenten insbesondere im Hinblick auf die Gruppenarbeitsumgebung zueinander in Beziehung stehen. Zu diesem Zweck habe ich in Abbildung 22 eine aktuelle Bearbeitungssituation „fotografiert“. Sie gibt einen Überblick über die einzelnen Objekte, die sich in den verschiedenen Prozeßräumen befinden. Zum Zeitpunkt des „Auslösens“ sind sowohl Pausenplanersteller als auch Sekretariatsmitarbeiter damit beschäftigt, Tätigkeiten im Rahmen der Pausenplanung auszuführen. Der Pausenplanersteller benutzt das Werkzeug Pausenplaner, um Lehrer für Pausenaufsichten einzuteilen, während der Sekretariatsmitarbeiter gerade eine Änderung am Lehrkörper vornimmt. Die Abbildung abstrahiert über die gesamte Menge an beteiligten Objekten, um die Übersichtlichkeit zu wahren.

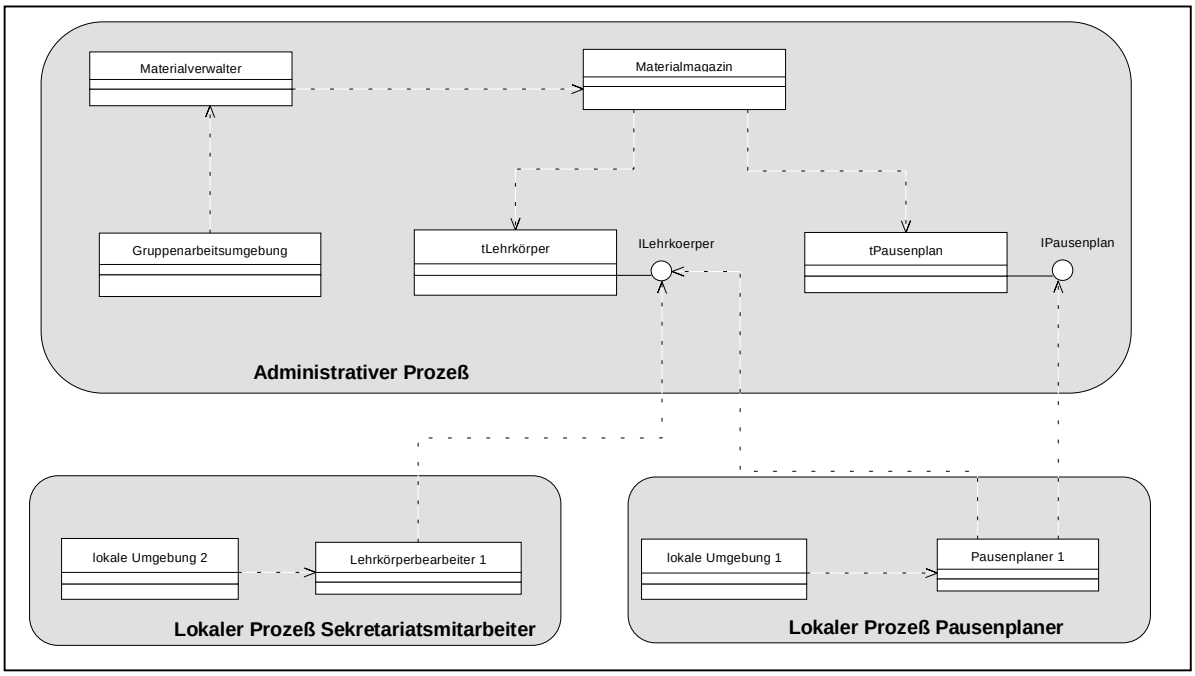


Abbildung 22 Bearbeitungssituation mit Sekretariatsmitarbeiter und Pausenplanersteller

Es lassen sich in der Abbildung drei verschiedene Prozeßräume, welche grau unterlegt sind, erkennen. Im oberen Bereich der Abbildung befindet sich der administrative Prozeß, während es sich bei den unteren um die Prozesse auf den Rechnern des Sekretariatsmitarbeiters und des Pausenplanersteller handelt. Es spielt für das System keine Rolle, auf welchem Rechner sich der administrative Prozeß befindet. DCOM ist in der Lage, die Gruppenarbeitsumgebung auf einem beliebigen Rechner zu lokalisieren. Man kann hier erkennen, daß sich die Materialien im Prozeß der Gruppenarbeitsumgebung befinden und in das Materialmagazin geladen wurden. Die Objekte Lehrkörper und Pausenplan werden von dem Werkzeug „Pausenplaner 1“ unter ihren Interfaces *ILehrkörper* und *IPausenplan* benutzt. Das Werkzeug „Lehrkörperbearbeiter 1“ benutzt den Lehrkörper ebenfalls unter dem Interface *ILehrkörper*.

Aus der Abbildung wird weiterhin deutlich, daß verschiedene Objekte des Systems über Prozeßgrenzen hinweg miteinander kommunizieren. Sie sind als Komponenten im Sinne von COM entworfen. Ich werde darauf im einzelnen noch eingehen.

8.2 Die Gruppenarbeitsumgebung

Das Objekt, welches die Gruppenarbeitsumgebung repräsentiert, befindet sich innerhalb des administrativen Prozesses. Es ist als Automationsserver entworfen und ermöglicht so den lokalen Komponenten den Zugriff.

8.2.1 Die Erzeugung der Gruppenarbeitsumgebung

Die Gruppenarbeitsumgebung ist als Singleton[35] konzipiert. Von ihr gibt es nur ein einziges Exemplar im gesamten System. Dieses wird erzeugt, sobald die erste lokale Umgebung gestartet wird und versucht, mit der Gruppenarbeitsumgebung Kontakt aufzunehmen. COM sorgt dafür, daß die Gruppenarbeitsumgebung anhand ihres globalen eindeutigen Identifikators lokalisiert wird und erzeugt dieses Exemplar. Ein weiterer Versuch, eine Gruppenarbeitsumgebung zu starten, liefert dem Anfragenden das bereits existierende Exemplar. Dieses ist verbunden mit einem Inkrementieren der Referenz auf dieses Objekt. Das Beenden einer lokalen Umgebung führt wiederum zu einem Dekrementieren der Referenz. Sobald die letzte lokale Umgebung beendet wird und der Referenzzähler Null erreicht, zerstört sich das Exemplar selbständig.[36]

Die Gruppenarbeitsumgebung sorgt für die Erzeugung und Initialisierung der weiteren Komponenten innerhalb ihres Prozesses. Im Folgenden werde ich diese Komponenten vorstellen und dem Leser einen tieferen Einblick in deren Konstruktion gewähren.

8.2.2 Die Komponenten der Gruppenarbeitsumgebung

In diesem Abschnitt über die einzelnen Komponenten der Gruppenarbeitsumgebung abstrahiere ich von den speziellen Eigenschaften des MFC Rahmenwerkes, um nicht von der wesentlichen Information dieses Kapitels abzulenken und damit die Verständlichkeit zu schmälern. Darüber hinaus werden die Einzelheiten des Rahmenwerkes und die Art und Weise der Konstruktion mit demselben in [Kra98] ausgiebig betrachtet. Hier möchte ich vielmehr auf die fachlich relevanten Elemente der Gruppenarbeitsumgebung eingehen und aufzeigen, wie sie konstruiert sind und in welchem Zusammenhang sie zueinander stehen. Ich möchte anmerken, daß ich den vorgestellten Materialversorger an dieser Stelle nicht näher diskutiert werde. Persistenz im Pausenplanersystem wird nicht über Materialversorger gewährleistet. Vielmehr macht sich die Materialverwaltung einen Serialisierungsmechanismus der MFC zunutze, um die Daten in eine Datei zu schreiben und von dort wieder zu lesen. Hierbei sind die Materialobjekte in der Lage, eine speicherbare Repräsentation von sich selbst abzugeben, welche persistent gemacht wird und es ermöglicht, diese Objekte wieder zu rekonstruieren. Dieser eher pragmatische Ansatz bei der Implementation soll hier weder näher vorgestellt werden, noch hat er Auswirkungen auf die eigentliche Konstruktion der in dieser Arbeit betrachteten Konzepte.

---

[35] Zum Singleton Pattern vergleiche [GHJ+95], S. 127ff.

[36] Vergleiche dazu Kapitel 6.3.1 über die Referenzzählung bei COM.

8.2.2.1 Der Materialverwalter

Der Materialverwalter wird durch die Gruppenarbeitsumgebung erzeugt. Er kennt die konkreten Materialien, die er verwaltet, nicht und ist somit in der Lage, Materialien beliebigen Typs zu verwalten. Alle Materialien des Pausenplanersystems, die nicht vom Typ Pausenplan oder Lehrkörper sind, sind Bestandteil dieser Materialien. So befindet sich ein Lehrer immer in einem der Lehrkörper, die am schwarzen Brett verfügbar sind. Daher verwaltet der Materialverwalter in seinem Magazin ausschließlich Materialien vom Typ Pausenplan oder Lehrkörper.

Der Materialverwalter ist im Zusammenhang mit Materialmagazin und Materialversorger in der Lage, verschieden Zugriffsstrategien zu realisieren. Mögliche Ziele einer Strategie könnten entweder sein, den Speicherbedarf zu minimieren, oder die Geschwindigkeit des Zugriffs auf die Materialobjekte zu maximieren. So könnten, je nach Zugriffsstrategie, nicht benutzte Objekte möglichst schnell aus dem Speicher entfernt werden um den Speicherbedarf gering zu halten, oder alle in Datenbanken vorhandenen Objekte zu Programmbeginn in den Speicher geladen werden, um die Performanz zu maximieren. In [JV87] werden Strategien diskutiert, die an dieser Stelle je nach Zielsetzung umgesetzt werden könnten.

Ich habe die Variante gewählt, bei der alle Materialien im Speicher gehalten werden und dadurch ein maximaler Durchsatz ermöglicht wird. Insbesondere im Hinblick auf die geringe Materialmenge und die hohe Anforderung an die Performanz des Systems erscheint mir diese Lösung am sinnvollsten.

8.2.2.2 Das Materialmagazin

Das Materialmagazin enthält alle im System befindlichen Materialobjekte. Es ist ein Behälter, der über verschiedene Zugriffsmöglichkeiten verfügt. Das Materialmagazin ermöglicht das Auffinden eines Materials anhand seiner Klassenzugehörigkeit und seines Namens. Weiterhin kann es alle Objekte eines Typs liefern. Diese Lösung ließ sich relativ einfach durch den in der MFC enthaltenen RTTI Mechanismus erreichen.[37]

8.2.2.3 Die Benutzerverwaltung

Die Benutzerverwaltung hat die Aufgabe, über die Aktivitäten der einzelnen Benutzer Buch zu führen. Die Schnittstelle der Klasse ist in Abbildung 23 aufgeführt. Der Materialverwalter teilt der

[37] Der Begriff RTTI steht für „RunTime Type Information“ und ist künftig fester Bestandteil der Sprache C++.

Benutzerverwaltung mit, sofern sich neue Benutzer beim System anmelden, abmelden oder neue Materialien zur Bearbeitung anfordern.

```
bool IstBenutzerBereitsAngemeldet(string Benutzer);
void GibMaterialienZuBenutzer(string Benutzer ,list <string> &);
void GibBenutzerZuMaterial(string Material, list<string> &);
void VerwaltungsEintragHinzufuegen (string Benutzer ,Material *);
void VerwaltungsEintragLoeschen(string Benutzer, Material *);
void BenutzerEintragLoeschen(string Benutzer);
Material *MaterialInBearbeitung(string Klasse, string Objekt);
void EintraegeLoeschen();
```

Abbildung 23 Öffentliche Schnittstelle von der Klasse Benutzerverwaltung

Fragen Werkzeuge nach den Aktivitäten anderer Benutzer, so kann die Benutzerverwaltung Auskunft darüber geben, welche Benutzer welche Materialien bearbeiten. Das Werkzeug „Schwarzes Brett“ präsentiert dem Benutzer alle Lehrkörper und Pausenpläne, die es im System gibt. Zusätzlich dazu ist neben den Materialsymbolen vermerkt, welche Benutzer dieses Material augenblicklich bearbeiten.

8.2.2.4 Die Konsistenzverwaltung

Zunächst möchte ich die Ziele für die Entwicklung einer Konsistenzverwaltung, die ich bereits verschiedentlich angedeutet habe, nochmals explizit nennen:

- Es dürfen keine Annahmen über den inneren Aufbau von Materialien gemacht werden.
- Das fachliche Wissen über Abhängigkeiten von Materialien sollte möglichst nur einer Instanz innerhalb des Systems bekannt sein. Die Kopplung mit dem Gesamtsystem sollte möglichst gering ausfallen.
- Das zu entwickelnde Konzept soll allgemeingültig sein und sich somit in verschiedenen Kontexten anwenden lassen können.

Das erste Ziel ist einleuchtend und sorgt dafür, daß das Geheimnisprinzip der Objektorientierung nicht gefährdet wird. Jedoch bedarf dieses einer genaueren Betrachtung. Ich möchte an dieser Stelle den Pausenplan als Diskussionsgrundlage verwenden. Dieser verfügt über eine sondierende Methode *ExistierenAufsichtskonflikte,* welche Auskunft darüber gibt, ob Konflikte hinsichtlich der zeitlichen Verfügbarkeit des Lehrpersonals und den Aufsichtseinteilungen existieren. Eine sondierende Methode eines Exemplars muß immer denselben Wert liefern, sofern sich sein Zustand nicht ändert. Der Zustand eines Objektes wird beschrieben durch die Menge der Ausprägungen seiner Exemplarvariablen. Legt man im Rahmen einer Implementation diese Zustandsdefinition zugrunde, so verändert sich der Zustand eines Pausenplanes nicht, wenn die Verfügbarkeit eines

Lehrers, welche zu dem Lehrkörper gehört, der mit dem betrachteten Pausenplan verknüpft ist, geändert wird. Dieses steht jedoch im Widerspruch dazu, daß die oben erwähnte, sondierende Methode ein anderes Ergebnis liefert, nachdem diese Veränderung des Lehrers stattgefunden hat. Konsequenterweise müßte man folglich argumentieren, daß das Geheimnisprinzip der Objektorientierung verletzt wäre. Durch die Konsistenzverwaltung wird dieser Verletzung entgegengewirkt, indem der Pausenplan davon in Kenntnis gesetzt wird, daß sich sein Zustand, der sich in den Ergebnissen der sondierenden Methoden äußert, verändert hat. Durch diesen Mechanismus wird dafür gesorgt, daß es für die interne Konstruktion des Materials unerheblich ist, ob solche Veränderungen intern abgefangen werden, oder nicht.

Aus diesem Beispiel wird deutlich, daß die Gewährleistung von Materialkonsistenz unabhängig von einer Materialimplementation sein muß.

Das zweite Ziel hat verschiedene Konsequenzen für die Gewährleistung von Materialkonsistenz. Zunächst bedeutet die Verfolgung dieses Zieles, daß es nicht möglich ist, die Konsistenz von Materialien durch Werkzeuge, die Materialien bearbeiten, zu gewährleisten. Das fachliche Wissen über Materialzusammenhänge würde sonst in verschiedene Werkzeuge innerhalb des Systems einfließen. Weiterhin hätte eine solche Herangehensweise zur Folge, daß Werkzeuge auch Materialien kennen müßten, die sie nicht bearbeiten. So müßte das Werkzeug „Lehrkörperbearbeiter“ auch den mit dem Lehrkörper verknüpften Pausenplan kennen, um ihn von der Veränderung des Ausschlußzeitraumes eines Lehrers in Kenntnis zu setzen, oder ihm zumindest mitzuteilen, daß er seinen Zustand aktualisieren muß.

Da die Werkzeuge nicht mit der Wahrung von Konsistenz betraut werden sollen, liegt die Überlegung nahe, daß die Materialien selbst einen Teil zu ihrer Konsistenz beitragen müssen. Wie bereits in Kapitel 4.2.2 diskutiert, läßt sich dieses über einen Beobachtungsmechanismus zwischen Materialien realisieren, jedoch wurde diese Lösung von mir aufgrund der zu starken Orientierung an den Kontext der Verwendung verworfen. Würde es ein System geben, welches lediglich dazu dient, den Lehrkörper einer Schule zu verwalten, so wäre ein Beobachtermechanismus, welcher den Lehrkörper als beobachtetes Material identifiziert, sinnlos. Ein Mechanismus, der für die Konsistenz von Materialien sorgt, muß also außerhalb des Materials angesiedelt werden und die Konsistenz in Abhängigkeit des jeweiligen Kontextes erhalten.

Als Folge dieser Überlegungen bin ich zu dem Schluß gekommen, daß es eine Instanz im System geben muß, welche diese Konsistenzbedingungen kennt. Diese Instanz ist der Konsistenzverwalter.

Um dem dritten mir selbst gesteckten Ziel gerecht werden zu können, war es notwendig, daß der Konsistenzverwalter generisch sein muß und somit in den verschiedensten Kontexten verwendbar

ist. Ein von mir bereits mehrfach festgestellter Sachverhalt ist der, daß bei der Entwicklung von generischen Automaten oder Werkzeugen ein Teil der Verantwortlichkeit an das Material delegiert werden muß. So ist es auch in diesem Falle. Um die Generizität des Konsistenzverwalters gewährleisten zu können, ist es notwendig, daß die Materialien ihre Abhängigkeit explizit machen. Zu diesem Zweck erben alle abhängigen Materialien von dem Aspekt *tAbhaengig*. Die Schnittstelle dieses Aspektes ist in Abbildung 24 aufgezeigt.

```
String _GibName();
void _MaterialGeloescht (tAbhaengig *);
void _MaterialVeraendert(tAbhaengig *);
```

Abbildung 24 Die Schnittstelle des Aspektes *tAbhaengig*

Der Konsistenzverwalter macht sich diese Eigenschaft des Materials zunutze, um Materialien von möglichen Veränderungen kontextabhängig in Kenntnis zu setzen. Der Konsistenzverwalter kennt die Abhängigkeiten zwischen den Materialien des Pausenplanersystems. Sobald ein Material angefordert wird, generiert und verwaltet der Konsistenzverwalter alle relevanten Abhängigkeiten. Findet eine Veränderung eines Materials statt oder wird ein Material gelöscht, so wird der Konsistenzverwalter informiert, welcher seinerseits die abhängigen Materialien informiert. Der Konsistenzverwalter benutzt Konsistenzobjekte, um diese Abhängigkeiten auszudrücken. Ein Konsistenzobjekt ist gekennzeichnet durch ein Subjekt (ein Material, von dem andere Materialien abhängig sind), durch einen Abhängigen und durch einen Verantwortlichen, der für die Erzeugung dieses Objektes verantwortlich ist. Konsistenzobjekte können selbst auch wiederum abhängig von anderen Materialien sein, welche wiederum von anderen abhängig sein können und so fort. Aus diesen Überlegungen heraus liegt die Konstruktion, wie sie in Abbildung 25 gezeigt wird, nahe.

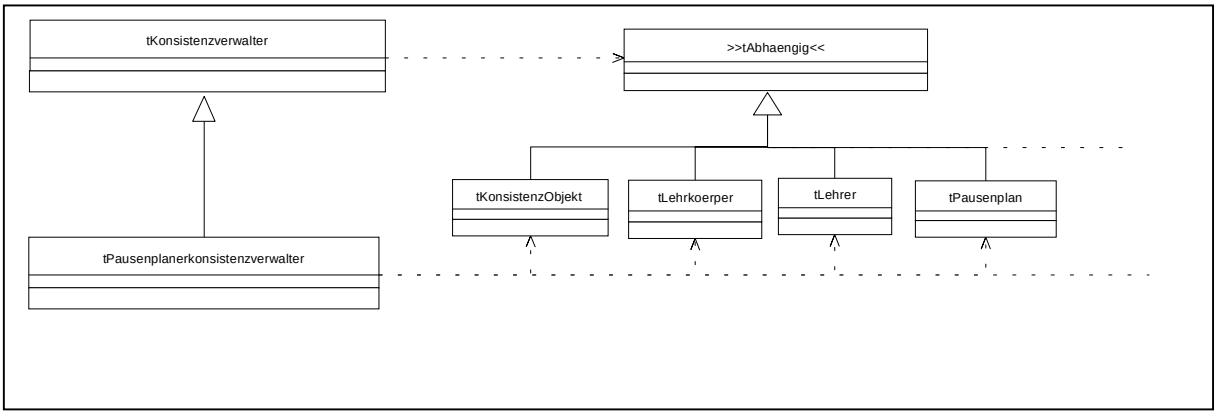


Abbildung 25 Klassendiagramm zur Konsistenzverwaltung

Die Klasse *tKonsistenzverwalter* realisiert eine allgemeingültige Implementation einer Konsistenzverwaltung. Sie kann Konsistenzobjekte löschen, Konsistenzen aktualisieren, sowie die Ereignisse an die abhängigen Materialien weiterleiten. Da die Konsistenzverwaltung fachliches Wissen über

die Materialien des Systems haben muß, wird dieses Wissen in einer abgeleiteten Klasse, der *tPausenplanerkonsistenzverwaltung* eingebracht.

Eine Methode, welche die Konsistenzverwaltung darüber informiert, daß ein Material zur Bearbeitung angefordert wurde, ist abstrakt und muß in einer Konkretisierung implementiert werden. In dieser Methode werden die relevanten Konsistenzobjekte, die notwendig sind, erzeugt und in die Konsistenzverwaltung eingebracht. Wird dem Konsistenzverwalter die Veränderung eines Materials mitgeteilt, so überprüft er die Konsistenzobjekte auf Abhängigkeiten zu dem veränderten Material. Wird eine Abhängigkeit gefunden, so wird diese Veränderung dem Konsistenzobjekt mitgeteilt. Allen abhängigen Materialien wird daraufhin die Veränderung mitgeteilt. Konsistenzobjekte sind nach dem „Composite Pattern“ konstruiert.[38] Abhängige Materialien können sowohl Materialien als auch wiederum Konsistenzobjekte sein. Ich werde dieses anhand eines Beispieles verdeutlichen.

Das Werkzeug Pausenplaner fordert den Pausenplan zur Bearbeitung an. Für alle drei Pausen der Pausenliste werden Konsistenzobjekte erstellt. Sie enthalten jeweils eine Pause und die Pausenliste. Weiterhin wird ein Konsistenzobjekt für Pausenliste und Pausenplan angelegt. Ist der Pausenplan mit einem Lehrkörper versehen, so wird weiterhin ein Konsistenzobjekt für Lehrkörper und Pausenplan angelegt. Der linke Teil der Abbildung 26 zeigt die Liste der Konsistenzobjekte für den beschriebenen Fall. Nachdem die Bearbeitung des Pausenplanes begonnen hat, wird nun auch der Lehrkörper zur Bearbeitung angefordert. Der Lehrkörper enthält zwei Lehrer. Die Konsistenzverwaltung generiert für jeden Lehrer des Lehrkörpers ein Konsistenzobjekt. Beim Einfügen in die Liste der Konsistenzobjekte stellt sie jedoch fest, daß es bereits Konsistenzobjekte gibt, bei denen als Subjekt der Lehrkörper vermerkt ist. Es besteht also sowohl eine Abhängigkeit zwischen Lehrer und Lehrkörper als auch zwischen Lehrkörper und Pausenplan. Die Konsistenzverwaltung kopiert die bereits enthaltenen Konsistenzobjekte für Lehrkörper und Pausenplan und hängt sie an die neuen Konsistenzobjekte für Lehrer und Lehrkörper an. Es entstehen nun Abhängigkeitsketten. Im Falle der Änderung eines Lehrers wird nun das Konsistenzobjekt, welches diesen Lehrer als Subjekt enthält, die Nachricht innerhalb der Kette weitergeben, bis alle Elemente benachrichtigt sind. Der rechte Teil der Abbildung 26 zeigt die entstandenen Konsistenzobjekte, nachdem die Bearbeitung des Lehrkörpers gestartet wurde.

[38] Vergleiche zum „Composite Pattern“: [GJH+95], S.163ff.

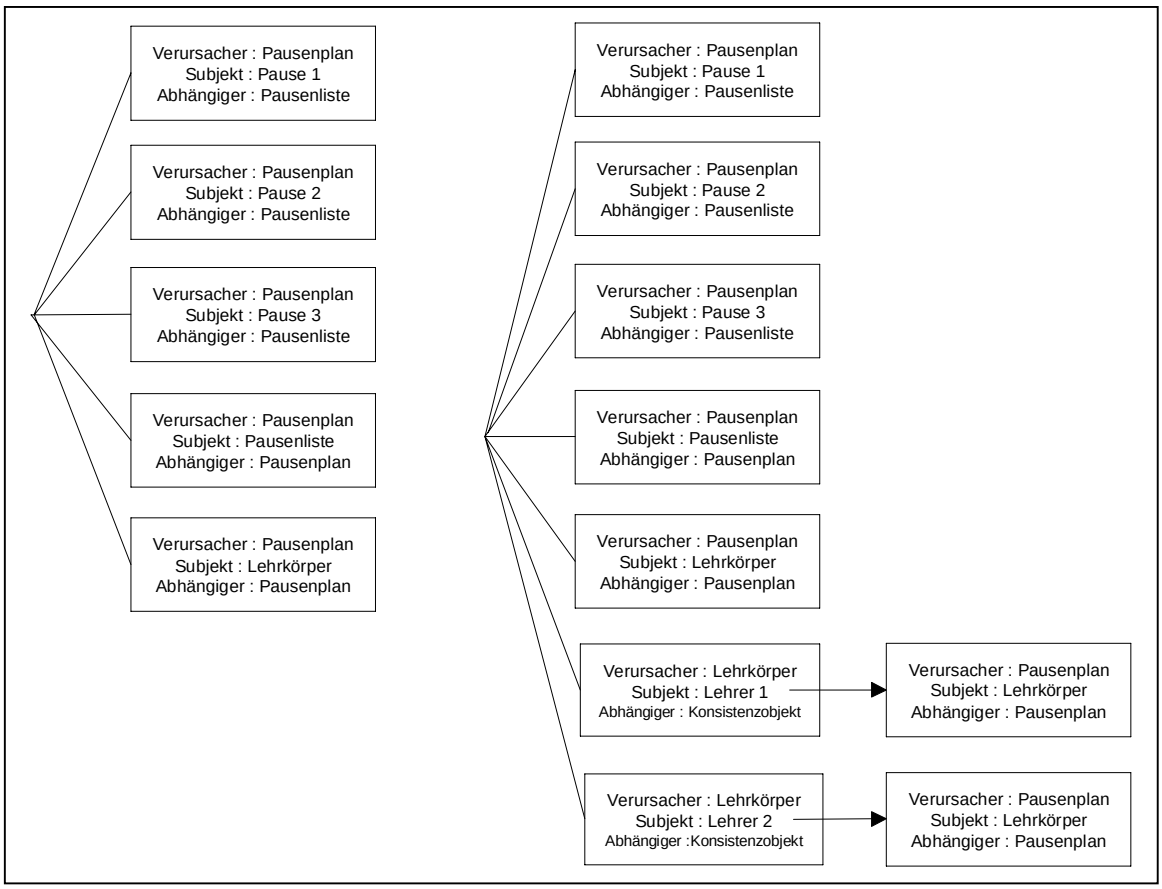


Abbildung 26 Konsistenzobjekte im Verlauf der Pausenplanung

Durch diese Konstruktion ist gewährleistet, daß Veränderungen an Materialien nicht zu inkonsistenten Materialien führen können. Die Granularität, die für die Konsistenzwahrung anzusetzen ist, hängt von den Werkzeugen ab, welche die Materialien bearbeiten. Würde die Granularität auf Ebene der Ausschlußzeiträume angesiedelt, so erfordert dieses Konsistenzobjekte für Ausschlußzeiträume und Lehrer. Um die gesamte Materialkonsistenz gewährleisten zu können, müßte die Konsistenzverwaltung für jedes Materialobjekt Konsistenzobjekte generieren. Dieses hätte eine Fülle derselben und einen hohe Verwaltungsaufwand innerhalb der Konsistenzverwaltung zur Folge. Um diesem zu begegnen, ist in der Implementation die Granularität auf der Ebene der Lehrer und der Pausen festgelegt.

8.3 Die Materialien

In diesem Kapitel werde ich die Materialien des Pausenplanersystems genauer betrachten. Dokumente zum fachlichen Entwurf sowie zum Design derselben findet sich im Anhang dieser Arbeit. Dieses Kapitel beschäftigt sich mit den Materialien im Hinblick auf die Realisierung des Pausenplanersystems als eine kooperative Anwendung. Zunächst beschäftige ich mich mit der Fragestellung, weshalb die Materialien ausschließlich im Adreßraum der Gruppenarbeitsumgebung erzeugt werden. Nachfolgend werde ich mich der Problematik von zusammengesetzten Materialien

widmen. Im weiteren Verlauf werde ich die Interfaces der Materialien sowie die daraus resultierenden Problembereiche diskutieren.

8.3.1 Erzeugung des Materials

Das Pausenplanersystem ist so konzipiert, daß sich das gesamte Material ausschließlich im administrativen Prozeßraum befindet. Andere Prozeßräume bekommen lediglich Proxies auf diese Materialien in Form von Interfaces geliefert. Die Materialien sind also als Komponenten im Sinne von COM implementiert. Sowohl die Rekonstruktion bereits bestehender Materialien als auch die Erzeugung von neuen Materialien wird ausschließlich innerhalb des administrativen Prozesses durchgeführt. Die Rekonstruktion wird über die Materialversorger realisiert. Im Kontext der WAM Metaphern wird die Erzeugung von Materialien hingegen üblicherweise von Werkzeugen übernommen. Für eine kontrollierte Verwaltung von Materialien durch den Materialverwalter ist es im Pausenplanersystem jedoch notwendig, diese im Prozeßraum der Gruppenarbeitsumgebung zu erzeugen. Da es dadurch möglich ist, die Lebenszeit der Materialien mit Hilfe des Referenzmechanismus von COM besser zu kontrollieren, werden Materialien ausschließlich durch die Gruppenarbeitsumgebung erzeugt.

Die Möglichkeit des Transfers von Materialien in verschiedene Prozeßräume besteht theoretisch. COM bietet diesen Dienst jedoch nicht an, sondern ist lediglich dafür ausgelegt, Objekte in anderen Adreßräumen anzusprechen. Ein Objekttransfer müßte von dem Entwickler selbständig implementiert werden. Dieser wäre beispielsweise möglich, indem ein Objekt eine abstrakte Repräsentation von sich erzeugt, diese über COM in einen anderen Adreßraum transportiert wird und über RTTI wieder in seinen ursprünglichen Zustand gebracht werden. Dieser, so wie auch andere Mechanismen, hätten einen hohen Anteil an Entwicklungszeit gefordert.

Die Möglichkeit, einen Objekttransfer auf COM aufzusetzen, haben wir deshalb nicht weiter in Erwägung gezogen. Nichtsdestotrotz wäre es möglich, Materialien in beliebigen Umgebungen des Systems zu erzeugen. COM hätte dieses problemlos unterstützt, da die Proxies den Zugriff auf das Material aus einer beliebigen Umgebung ermöglichen.

In der jetzigen Konstruktion ist es jedoch so, daß die konkrete Implementation der Materialien lediglich im Rahmen der Gruppenarbeitsumgebung bekannt ist.. Die Kunden importieren die Interfaces der Materialien über die Type Library in das jeweilige Proxy. Durch das „Kapseln“ der Materialien innerhalb der Gruppenarbeitsumgebung haben Veränderungen an den Schnittstellen der Materialien lediglich das erneute Importieren der Materialschnittstellen auf Kundenseite zur Folge. Weitere Abhängigkeiten existieren nicht.

Ein weiteres Argument dafür, daß sich alle Materialien im Adreßraum der Gruppenarbeitsumgebung befinden, ist die Tatsache, daß der Materialverwalter alle persistenten Materialien aus den Datenbanken rekonstruieren muß. Da ein Objekttransfer aus genannten Gründen nicht in Frage kommt, sind sie ohnehin an diesen Adreßraum gebunden. Andernfalls müßten alle Kunden ebenso über die Möglichkeit verfügen, Materialien aus Datenbanken zu rekonstruieren.

8.3.2 Materialaspekte und Materialinterfaces

COM sieht vor, daß Komponenten über Interfaces benutzt werden (vgl. Kapitel 6.3.2). Interfaces definieren eine Menge von Methoden, die durch die Komponente, die dieses Interface unterstützt, implementiert werden. Insofern liegt es nahe, Aspekte innerhalb des Systems über Interfaces im Sinne von COM zu implementieren. Dieses setzt voraus, daß die Materialien als Komponenten entworfen werden. Da die Realisierung des Pausenplanersystems in prototypischer Form geschehen ist und die Implementation von Automationsservern, wie bereits in Kapitel 6.4 festgestellt, ausgesprochen einfach für den Entwickler ist, haben wir die Implementation der Materialien mit Hilfe des *IDispatch* Interfaces vorgezogen. Nichtsdestotrotz hätten sich insbesondere die Interfaces der Komponenten angeboten, um Aspekte von Materialien zu realisieren.

8.4 Die Werkzeuge

Aus den Visionen des Pausenplanersystems in Kapitel Anhang B wird deutlich, welche Werkzeuge das Pausenplanersystem beinhaltet und welche Aufgaben sie unterstützen sollen. Da in [Kra98] neben der Konstruktion nach WAM mit Hilfe der MFC auch die Werkzeugkonstruktion eingehend diskutiert wird, werde ich diesen Bereich hier nicht näher erläutern. Der geneigte Leser möge sich dort über den Aufbau und die Konstruktion der Werkzeuge einen Überblick verschaffen. Trotzdem werde ich die Werkzeuge hier kurz aufzählen und deren Funktion erläutern:

Das Werkzeug

- *Schwarzes Brett* ermöglicht den Benutzern, sich in die Gruppenarbeitsumgebung zu begeben und präsentiert die in der Umgebung befindlichen Materialien und Werkzeuge in Form eines schwarzen Brettes.
- *Pausenplaner* unterstützt den Pausenplanersteller bei der Belegung von Pausenaufsichten mit Lehrern. Die Lehrer kann der Pausenplanersteller mit der Maus auf die zu beaufsichtigenden Pausen ziehen und so eine Einteilung vornehmen. Das Werkzeug informiert den Benutzer über den Arbeitsfortschritt sowie gegebenenfalls existierende Aufsichtskonflikte.

- *Lehrkörperbearbeiter* ermöglicht dem Sekretariatsmitarbeiter, die relevanten Informationen über die Lehrer zu pflegen. Diese Informationen umfassen die Stelle des Lehrers sowie die Zeiträume, in denen der Lehrer nicht zu einer Aufsichtspflicht eingeteilt werden soll.
- *Aufsichtsortebearbeiter* ermöglicht die Definition der verschiedenen an der Schule befindlichen Orte, an denen sich die Schüler in der Pause aufhalten und die folglich beaufsichtigt werden müssen.
- *Pausenlistenbearbeiter* unterstützt den Pausenplanersteller bei der Definition der Pausen sowie bei der Festlegung, welche Aufsichtsorte in welcher Pause beaufsichtigt werden müssen.

8.5 Die Kommunikation zwischen den Komponenten

Ich möchte im Folgenden die Realisierung der Kommunikation im Pausenplanersystem erläutern. Zunächst möchte ich im ersten Kapitel begründen, warum die gesamte Kommunikation im Pausenplanersystem synchron ist. Danach behandle ich den Benachrichtigungsmechanismus und die Ereignisse, die im Pausenplanersystem eintreten können.

8.5.1 Synchrone Kommunikation im Pausenplanersystem

In Kapitel 4.4 wurden die Möglichkeiten der synchronen und asynchronen Kommunikation diskutiert. Das Pausenplanersystem bedient sich ausschließlich der synchronen Kommunikation. Der Grund dafür findet sich darin, daß die Kommunikationsmöglichkeiten, die COM anbietet, auf synchroner Kommunikation basieren.

Diese Tatsache hat einen historischen Hintergrund. OLE wurde ursprünglich konzipiert als eine Strategie zur Integration verschiedener Anwendungen und den damit verbundenen Datenformaten. Es basierte auf der DDE Technologie, welche dafür sorgte, daß Daten zwischen verschiedenen Anwendungen ausgetauscht werden konnten, welche sich in verschiedenen Prozessen, aber auf einem Rechner befinden. Diese Form der Kommunikation ist zeitunkritisch, da keine Verzögerungen durch aufwendige Netzwerkkommunikation eintreten. Die in Kapitel 4.4 aufgestellte Formel der Wartezeit reduziert sich dadurch auf den Teil der Zeit, die der Dienstleister mit der Ausführung noch zu bearbeitender Operationen beschäftigt ist sowie einem geringen Verwaltungsaufwand. Somit sahen die Entwickler nicht die Notwendigkeit der asynchronen Kommunikation. Im Laufe der Zeit hat sich die Situation verändert. Es gibt Ansätze zur asynchronen Kommunikation, die aus der Anforderung an Internetanwendungen entstanden sind, jedoch bleiben diese Ansätze zur Zeit

auf diesen Anwendungsbereich beschränkt. Sie haben noch keinen durchgängigen Einzug in die OLE Technologie gehalten.[39]

Somit haben wir das Pausenplanersystem aus pragmatischen Gründen mit synchroner Kommunikation ausgestattet. Es bleibt allerdings zu erwähnen, daß OLE robust gegenüber dem Ausfall von entfernten Kommunikationspartnern ist. Zum einen setzt das System bei jedem Aufruf an ein entferntes Objekt einen Timeout, welcher dafür sorgt, daß der Aufruf einer Operation abgebrochen wird, sobald er eine festgelegte Dauer überschreitet. Zum anderen werden über einen Ausnahmemechanismus Ausnahmesituationen auf der Seite des Dienstleisters abgefangen.

8.5.2 Der Benachrichtigungsmechanismus

Den Benachrichtigungsmechanismus, welcher in das Pausenplanersystem integriert ist, möchte ich in dem nun folgenden Kapitel vorstellen. Aus Platzgründen können hierbei nicht alle Aspekte und Probleme dieses Bereiches angesprochen und diskutiert werden. Tiefergehende Informationen finden sich bei [GJH+95], [Lew95] und [RW96].

8.5.2.1 Vorstellung des Konzeptes

Um die notwendige Kommunikation zwischen Dienstleister und Kunden realisieren zu können, bedient man sich der losen Kopplung anhand eines Benachrichtigungsmechanismus. Der Dienstleister kennt seine Kunden nur unter der Schnittstelle, welche die Funktionalität der Benachrichtigung umfaßt. Es gibt verschiedene Ansätze, diesen Mechanismus zu realisieren. In [GJH+95] wird ein einfaches Beobachtermuster angeführt. Ebenso werden mögliche Erweiterungen genannt. In [RW96] werden verschiedene Reaktionsmuster aufgeführt, aus denen letztlich ein neues, das „Event-Observer“ Muster, entwickelt wird. An dieser Stelle sollen nicht alle diese verschiedenen Ansätze vorgestellt werden, jedoch soll hier argumentiert werden, inwiefern der bestehende Benachrichtigungsmechanismus im Kontext des Pausenplanersystems angebracht ist.

COM sieht die Möglichkeit vor, Komponenten mit Hilfe eines Beobachtermusters lose aneinander zu koppeln. OLE abstrahiert über diesen technischen Abläufen. Der Entwickler erzeugt auf der Beobachterseite einen Stub, welcher die Benachrichtigung erhalten soll. Diesen trägt er durch den Aufruf einer Methode bei dem Beobachteten ein.

Benachrichtigungen werden an das Stub Objekt gesandt, welches sich der Schnittstelle einer aufgeschobenen Oberklasse bedient, um eine Benachrichtigungsmethode beim beobachtenden

[39] Vergleiche hierzu [OHE96] S.285.

Objekt aufzurufen. Abbildung 27 zeigt die Struktur des Benachrichtigungsmechanismus zwischen dem Werkzeug Lehrkörperbearbeiter und dem Ereignisverwalter mit angemeldetem *MaterialVeraendertStub*.[40] Der Kunde muß von der aufgeschobenen Klasse *tMaterialVeraenderungsBeobachter* erben und die Methode *MaterialVerändert* implementieren. Diese Methode wird gerufen, sobald ein „Material verändert“ Ereignis aufgetreten ist.

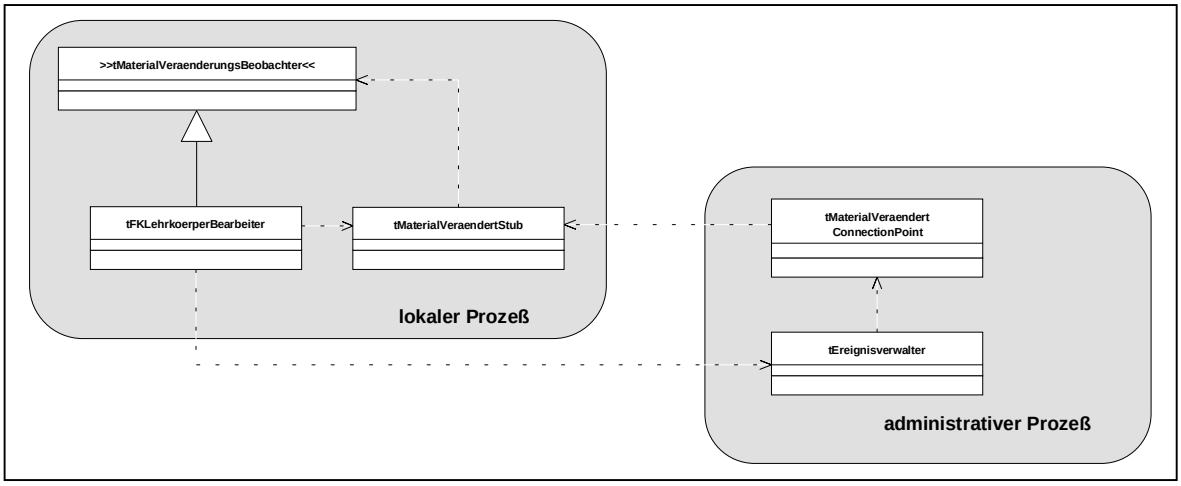


Abbildung 27 Die Verbindung vom Werkzeug Lehrkörperbearbeiter und Ereignisverwalter

COM ermöglicht dem Entwickler auf einfache Art und Weise, Klassen mit Verbindungspunkten zu versehen. Die einfache Deklaration zweier Makros integriert die volle Funktionalität in die beobachtete Klasse. Die Stubs sind wiederum Automationsobjekte, die aus anderen Prozessen gerufen werden können. Wir haben diese spezielle Oberklasse für Beobachter gewählt, um einerseits aus dem Klassendiagramm ableiten zu können, wer hier als Beobachter respektive Beobachteter fungiert, und andererseits, um eine saubere Schnittstelle zwischen Stub und Beobachter zu erhalten. Alternative dazu wäre es möglich gewesen, dem Stub einen Zeiger auf eine Methode zu übergeben. Nachteilig dabei ist die Tatsache, daß der Beobachter im Falle mehrerer angemeldeter Stubs eines Typs in der Methode *MaterialVerändert* den Verursacher zunächst ermitteln muß.

8.5.2.2 Die Ereignisse im Pausenplanersystem

Es gibt zwei verschiedene Ereignisse, die im Pausenplanersystem für Werkzeugkonsistenz, Materialkonsistenz und die Unterstützung von Kooperation relevant sind.

- Material verändert

[40] Die im Pausenplanersystem relevanten Ereignisse werden im folgenden Kapitel vorgestellt.

Dieses Ereignis geht von den Werkzeugen aus und wird von dem Ereignisverwalter empfangen. Von dort geht es sowohl dem Konsistenzverwalter als auch allen interessierten Werkzeugen zu. Der Konsistenzverwalter überprüft aufgrund der Materialidentität, welche die Nachricht enthält, ob es Materialien im System gibt, die von diesem abhängig sind. Ist dieses der Fall, so leitet der Konsistenzverwalter diese Nachricht an den Ereignisverwalter weiter, welcher die davon betroffenen Werkzeuge in Kenntnis setzt. Diese aktualisieren die Präsentation der Informationen.

- Bearbeitungssituation verändert

  Das Anmelden und Abmelden von Benutzern am schwarzen Brett führt über den Benutzerverwalter zu einer Benachrichtigung von den Werkzeugen, die sich für dieses Ereignis angemeldet haben, um etwa die am schwarzen Brett befindlichen Personen anzuzeigen. Darüber hinaus tritt dieses Ereignis ebenfalls auf, sofern ein Werkzeug Material bei dem Materialverwalter nachfragt. Von dort gelangt es über den Benachrichtigungsmechanismus sowohl an den Benutzerverwalter, welcher wiederum angemeldete Werkzeuge informiert, als auch an den Konsistenzverwalter, welcher diese Information benötigt, um seine Konsistenzstruktur zu aktualisieren.

Der Ereignisverwalter sendet jedem angemeldeten Werkzeug Informationen über Materialveränderungen zu, auch wenn ein Werkzeug diese Information nicht benötigt, da es das betreffende Material nicht bearbeitet. Lediglich das Werkzeug, welches für die Veränderung verantwortlich ist, wird nicht benachrichtigt. Es wäre beispielsweise nicht notwendig, ein Werkzeug Lehrkörperbearbeiter von der Veränderung eines Materials vom Typ Pausenplan in Kenntnis zu setzen.

Der Ereignisverwalter liefert die Identität des veränderten Materials mit an das Werkzeug. Es wird so in die Lage versetzt, auf die Aktualisierung der Präsentation zu verzichten, sofern der derzeitige Zustand des Werkzeugs dieses nicht erfordert. Gamma et al. unterscheiden dabei zwischen dem „push model“ und dem „pull model“. Im Rahmen des „push models“ werden alle notwendigen Informationen, welche für die Aktualisierung notwendig sind, vollständig mit übermittelt. Sie argumentieren allerdings, daß in diesem Falle das beobachtete Subjekt Annahmen darüber treffen müßte, welche Informationen die Beobachter benötigen. Gryczan und Lilienthal befürworten in [GL96] für verteilte Anwendungssoftware das „push model“, da ein erneutes Sondieren des Materials und somit gegebenenfalls ein Zugriff über das Netzwerk nicht mehr notwendig ist.

Im Falle einer marginalen Veränderung eines Objektes gäbe es für die Struktur der übermittelten Information im Rahmen des „push models“ zwei Möglichkeiten:

- Die neue Ausprägung ist in der Information enthalten. Ein erneutes Sondieren des Materials wäre nicht notwendig. Dieses würde allerdings bei vielen Veränderungen eine Flut von unterschiedlichen Ereignistypen zur Folge haben.

- Das gesamte Objekt ist in der Information enthalten, so daß ein Ereignistyp für verschiedene Änderungen angebracht ist.

Im Gegensatz zu dem „push model“ basiert das „pull model“ auf der Idee, daß den Interessenten eine anonyme Nachricht zukommt, die keinerlei Aufschlüsse über die tatsächliche Veränderung enthält. Der Benachrichtigte muß daraufhin die durchgeführten Änderungen erfragen.

Das Pausenplanersystem basiert grundsätzlich auf dem „pull modell“. Einzig die Identität des Materials ist in der Benachrichtigung enthalten. Folglich kommt ein erneutes Sondieren des Materials nur in Frage, sofern das Werkzeug das veränderte Material tatsächlich bearbeitet. Ein „push model“ haben wir einerseits aufgrund des fehlenden Objekttransfers und andererseits weil wir die Menge der Ereignistypen begrenzen wollten, im Pausenplanersystem nicht in Betracht gezogen. Die Werkzeuge, die das veränderte Material bearbeiten, müssen alle präsentierten Informationen vollständig neu erfragen.

8.5.2.3 Die Dynamik der Benachrichtigung

Um die Vorgänge bei der Anmeldung eines Beobachters besser nachvollziehen zu können, werde ich die Dynamik des Beobachtens anhand eines Interaktionsdiagramms verdeutlichen. Ein Lehrkörperbearbeiter trägt sich bei dem Ereignisverwalter als Interessent für das Ereignis „Material verändert“ ein. Zu diesem Zweck erzeugt sich das Werkzeug ein Exemplar vom Typ *tMaterial-VeraendertStub*. Diesen Stub meldet es bei dem Ereignisverwalter an.

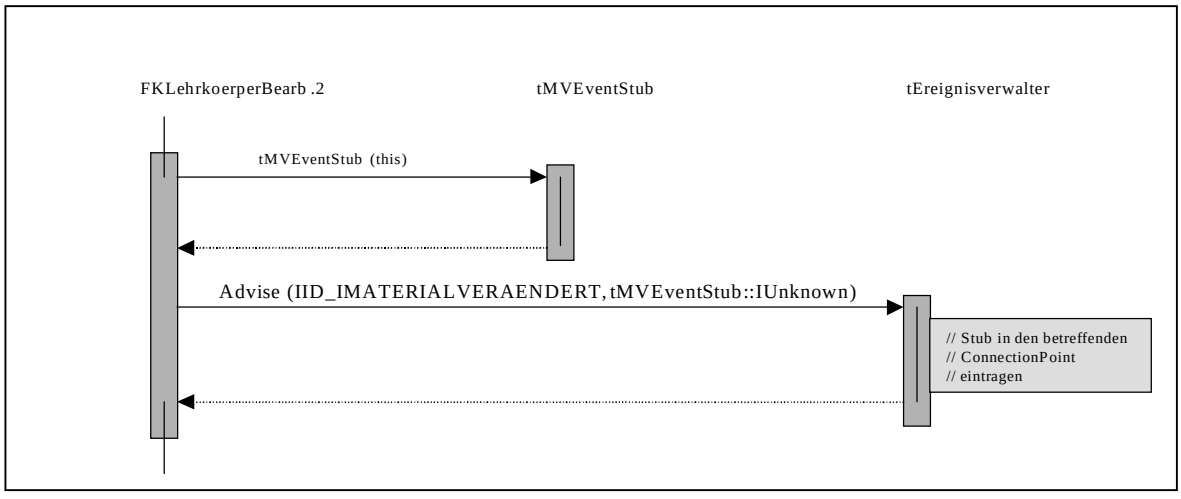


Abbildung 28 Die Anmeldung eines MaterialVeraendert Stubs beim Ereignisverwalter

Der Ereignisverwalter muß über einen Verbindungspunkt verfügen, welcher vom Typ *tMaterial-Veraendert* ist. Abbildung 28 zeigt den Anmeldevorgang in Form eines Interaktionsdiagramms.

Ist diese Verbindung erfolgreich zustande gekommen, so kann der Ereignisverwalter Nachrichten an das angemeldete Werkzeug übermitteln. Dieses möchte ich ebenfalls anhand eines Beispieles aufzeigen. Zwei Werkzeuge vom Typ Lehrkörperbearbeiter greifen synchron auf das Lehrkörperobjekt „LK1“ zu. Zuvor haben sich beide Werkzeuge bereits als Beobachter des Ereignisses „Material Verändert“ beim Ereignisverwalter angemeldet und nacheinander das Material LK1 vom Typ Lehrkörper erfragt. Der Benutzer des Werkzeuges „Lehrkörperbearbeiter1“ verändert nun den Lehrkörper. Dem Ereignisverwalter wird dieses durch Aufruf der Methode *MaterialVeraendert* mitgeteilt. Dieser muß nun alle für dieses Ereignis angemeldeten Werkzeuge von der Veränderung in Kenntnis setzen. Abbildung 29 illustriert die Benachrichtigung des Werkzeugs „LehrkörperBearbeiter 2“ durch den Ereignisverwalter. Im Verbindungspunktbehälter findet er neben dem Verursacher des Ereignisses weiterhin den Stub des zweiten Werkzeugs.

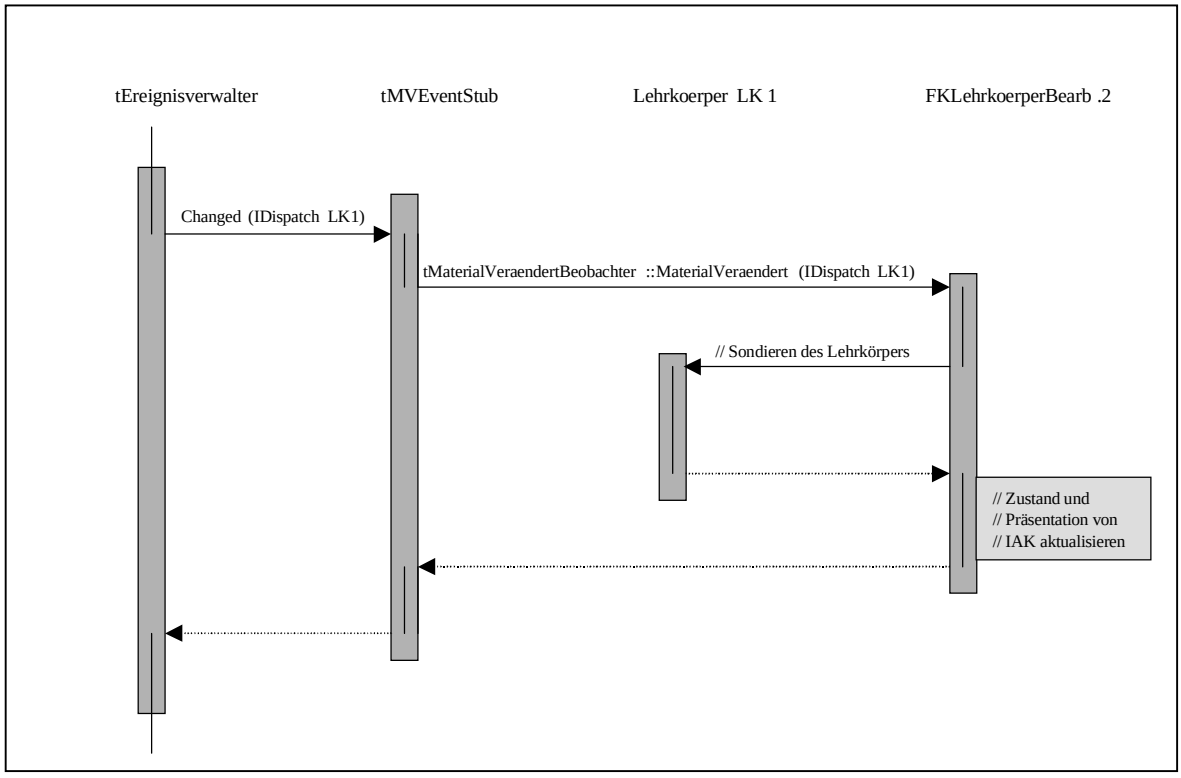


Abbildung 29 Interaktionsdiagramm zum Ereignis "Material verändert"

Dieser Stub wird über die Methode *Changed* unter Angabe des veränderten Materials gerufen. Der Stub seinerseits ruft bei dem bei ihm eingetragenen Beobachter die Methode *MaterialVeraendert* ebenfalls unter Angabe des Materials auf. Das Werkzeug reagiert, indem es die Informationen über den veränderten Lehrkörper neu darstellt.

Der Benachrichtigungsmechanismus ist auf einfache Art und Weise realisiert. Probleme, die durch Ereigniswettlauf auftreten können, sind hier nicht berücksichtigt worden Die Integration weiterer Ereignisse ist allerdings problemlos möglich.

8.6 Zusammenfassung

Ich habe in diesem Kapitel den technischen Entwurf des Pausenplanersystems dargelegt. Der Schwerpunkt bei dieser Betrachtung lag insbesondere auf der Konstruktion als verteiltes System. Die Werkzeuge habe ich nicht näher betrachtet, da sie in [Kra98] sehr genau vorgestellt werden. Statt dessen habe ich mich auf die Konstruktion der Gruppenarbeitsumgebung sowie die Aspekte des komponentenbasierten Ansatzes konzentriert. Insbesondere der Entwurf der Materialien sowie die Realisierung des Benachrichtigungsmechanismus habe ich dabei ausführlich betrachtet.

8.7 Bewertung

Das Pausenplanersystem in seiner derzeitigen Version enthält eine Reihe von Schwächen. Als wichtigen Punkt werte ich die Tatsache, daß Materialien nicht von den Werkzeugen erzeugt werden, sondern durch die Gruppenarbeitsumgebung. Dieses führt dazu, daß der Entwurf der Gruppenarbeitsumgebung keinen allgemeingültigen Charakter hat. Sie kennt konkrete Materialien und ist somit nicht ohne weiteres für andere Anwendungsbereiche wiederverwendbar.

Ein weiteres Manko ist die Tatsache, daß Aspekte im Pausenplanersystem nicht realisiert sind. Dieses ist insbesondere insofern unzureichend, als daß das Interfacekonzept, welches COM anbietet, eine ganz konkrete Implementationsmöglichkeit der Entwurfsmetapher Aspekt ist.

Eine weitere Schwäche des Systems ist die synchrone Kommunikation. Alle Komponenten kommunizieren auf diese Art und Weise miteinander. Die Nachteile dieser Kommunikationsart habe ich bereits ausreichend diskutiert. Es bleibt zu hoffen, daß COM die Möglichkeit der asynchronen Kommunikation zu einem späteren Zeitpunkt anbieten wird, so daß Entwickler nicht dazu gezwungen werden, diese selbst im Rahmen einer Entwicklung mit einbringen zu müssen.

Trotz der erwähnten Schwächen des Systems hat sich gezeigt, daß eine Softwareentwicklung nach dem Konzept der Gruppenarbeitsumgebung zu dem gewünschtem Ziel geführt hat. Aus der Vorstellung des Systems wird deutlich, daß sich die einzelnen Komponenten als benutzbar erweisen. Einige dieser Komponenten, wie beispielsweise die Konsistenzverwaltung oder die Benutzerverwaltung sind so entworfen, daß sie ebenfalls in anderen Kontexten wiederverwendet werden können.

Die durch COM und die MFC zugrunde liegende Plattform hat sich als stabil erwiesen, um Systeme, welche auf dem Konzept der Gruppenarbeitsumgebung basieren, zu realisieren. Trotz einiger pragmatischer Lösungswege bleibt als Gesamteindruck ein System, welches die grundsätzlichen Anforderungen an die Entwicklung verteilter Softwaresysteme erfüllt. Da insbesondere die COM Technologie noch „in den Kinderschuhen steckt“, bleibt zu hoffen, daß im Detail einige Aspekte zukünftig verbessert werden.

9 Abschließende Betrachtung

Die Zielsetzung dieser Arbeit bestand in der Aufstellung eines Konzeptes zur Entwicklung von Softwaresystemen, welches Kooperation an einem Ort unterstützen soll. Bei der Kooperation an einem Ort handelt es sich um eine softwaregestützte Form der Zusammenarbeit mehrerer Personen. Um die möglichen Einsatzziele dieser Kooperationsform besser definieren zu können, habe ich zunächst eine Taxonomie von Formen der Zusammenarbeit in der Realität aufgestellt und aufgezeigt, mit welchen Problemen Entwickler konfrontiert werden, sofern sie versuchen, eine Abbildung eines Bereiches auf ein Softwaresystem zu erreichen. Dabei habe ich verschiedene Erkenntnisse und Ergebnisse aus der CSCW Forschung in die Diskussion mit einfließen lassen.

Die in dieser Arbeit betrachtete Kooperationsform bildet die Zusammenarbeit von Menschen auf eine Gruppenarbeitsumgebung ab, welche den Personen durch ein Softwaresystem zur Verfügung gestellt wird. Ich habe anhand von Merkmalen beschrieben, welche Form der Zusammenarbeit geeignet ist, um im Rahmen dieses Konzeptes unterstützt werden zu können. Darüber hinaus habe ich anhand des Pausenplanersystems gezeigt, wie eine konkrete Übertragung einer Zusammenarbeit auf das vorgestellte Konzept möglich ist. Die erarbeiteten Szenarios und Systemvisionen unterstreichen die zweckmäßige Übertragung der Zusammenarbeit.

Die Diskussion der Gruppenarbeitsumgebung auf fachlicher Ebene hat ergeben, daß Kooperation an einem Ort nur bedingt nutzbar ist. Sie ist beschränkt auf die Zusammenarbeit in kleinen Gruppen, erfordert gegenseitige Kenntnis der beteiligten Personen, ein hohes Maß an Transparenz und ist darüber hinaus häufig durch einen hohen Koordinationsbedarf gekennzeichnet. Letzterer kann sowohl durch Mechanismen außerhalb des Softwaresystems gedeckt werden als auch durch eine regelnde Komponente innerhalb des Systems. Wiederum liegt hier der Schwerpunkt auf der Zielsetzung, Sinn und Zweck der Zusammenarbeit nicht zu verfälschen.

Die Entwicklung des Pausenplanersystems dient hier als Beispiel für eine konkrete Form der Zusammenarbeit an einem Ort, bei der uns die Abbildung als ausreichend für die Zusammenarbeit der beteiligten Personen erscheint. Das fachlich motivierte Konzept der Gruppenarbeitsumgebung als Manifestierung des Ortes der Zusammenarbeit habe ich durch eine technische Interpretation gestützt, welche sich in die Konzepte und Metaphern der WAM Methode einfügt.

In der technischen Interpretation der Gruppenarbeitsumgebung haben sich verschiedene Schwächen offenbart. Sie finden sich allerdings lediglich in Teilbereichen wie beispielsweise der Materialverwaltung und Materialkonsistenz wieder und können gegebenenfalls adaptiert werden ohne den Grundgedanken des vorgestellten Konzeptes zu verwerfen. Im konkreten Anwendungsfall hat sich das Konzept nichtsdestotrotz als tragfähig erwiesen.

Da die konkrete Entwicklung des Pausenplanersystems eine Basis für die Entwicklung von verteilten Softwaresystemen erfordert, habe ich in die Grundlagen der Technologie von COM und OLE eingeführt und gezeigt, wie sich die Arbeit mit diesen Rahmenwerken darstellt. Es ist deutlich geworden, daß die Rahmenwerke eine Entwicklung von verteilten Softwaresystemen ermöglichen und im Ansatz weiterführende Konzepte wie beispielsweise die Grundlagen für ein Beobachtermuster enthalten. Darüber hinaus hat sich gezeigt, daß COM und OLE sinnvoll in die MFC verflochten sind und dem Entwickler dadurch eine akzeptable Grundlage zur Systementwicklung an die Hand gegeben wird. Allerdings haben sich ebenfalls einige Grenzen dieser Technologie, wie etwa fehlender Objekttransfer und mangelhaft unterstützte asynchrone Kommunikation, offenbart.

Es ist meines Erachtens deutlich geworden, daß es möglich ist, die Zusammenarbeit von Menschen im Rahmen der betrachteten Form konzeptuell zu unterstützen. Weiterhin läßt sich feststellen, daß die technischen Voraussetzungen zur Umsetzung dieser Konzepte gegeben sind. Für die Zukunft ist es sicherlich wünschenswert zu überprüfen, ob sich die Ergebnisse dieser Arbeit auch in anderen konkreten Aufgabenstellungen sinnvoll einsetzen lassen.

Anhang A Literaturverzeichnis

[BBG+95] Dirk Bäumer, Reinhard Budde, Guido Gryczan, Karl-Hein Sylla, Heinz Züllighoven
Objektorientierte Konstruktion von Software-Werkzeugen und –Materialien
Informatik-Spektrum, Band 18, Heft 4, August 1995, Springer-Verlag, Berlin, Heidelberg. S. 203-210.

[Ble97] Wolf-Gideon Bleek
Techniken zur Konstruktion verteilter und technisch eingebetteter Anwendungssysteme
Diplomarbeit an der Universität Hamburg, Fachbereich Informatik, Arbeitsbereich Softwaretechnik 1997

[Boo94] Grady Booch
Object-Oriented Analysis and Design with Applications. 2nd Edition
Benjamin,Cummings 1994

[BR91] Liam Bannon, Mike Robinson
Questioning Representations
Proceedings of the Second European Conference on Computer-Supported Cooperative Work 25-27 September 1991 Amsterdam

[Bri97] Bristol Technology
MFC History and Evolution
http://www.bristol.com/Bibliography/mfchistory.htm 1997

[Bro95] Kraig Brockschmidt
Inside OLE , Second Edition
Microsoft Press 1995

[Bro96] Kraig Brockschmidt
What OLE is really about
Microsoft Corporation 1997

[BFK+87] Daniel G. Bobrow, Gregg Foster, Kenneth Kahn, Stan Lanning, Mark Stefik, Lucy Suchman
Beyond the Chalkboard : Computer Support for Collaboration and Problem Solving in Meetings
aus: CACM 30(1) S.32-47 1987

[BZ90] Reinhard Budde, Heinz Züllighoven
Software-Werkzeuge in einer Programmierwerkstatt – Ansätze eines hermeneutisch fundierten Werkzeug- und Maschinenbegriffs
Bericht der Gesellschaft für Mathematik und Datenverarbeitung Nr. 182
Oldenbourg Verlag 1990

[CHP+95] Jan Crüsemann, Andreas Hartmann, Thomas Poggendorf, Jan Spiess
Objektorientierte Entwicklung eines Bibliothekssystems
Studienarbeit an der Universität Hamburg, Fachbereich Informatik, Arbeitsbereich Softwaretechnik 1995

[CHY+98] P. E. Chung, Y. Huang, S. Yajnik, D. Liang, J. Shih, C.-Y. Wang, Y.-M. Wang
DCOM and Corba
Side by side, step by step and layer by layer
C++ Report Vol 10 / No 1 Januar 1998

[COO+93] Mark Carter , Judith S. Olson, Gary M. Olson, Marianne Storrøsten
Groupwork Close Up : A Comparison of the Group Design Process with and without a simple Group Editor
ACM Transactions on Information Systems Vol. 11 No. 4 October 1993 S 321-348

[FKM95] Geraldine Fitzpatrick, Simon Kaplan, Tim Mansfield
Physical spaces, virtual places and social worlds: A study of work in the virtual
Dept. of Computer Science, University of Queensland, Qld, Australia
CRC for Distributed Systems Technology, Australia
Dept of Computer Science, University of Illinois, USA

[FKT95] Geraldine Fitzpatrick, Simon M. Kaplan, William J. Tolone
Work, Locales, and Distributed Social Worlds
Proceedings of the Fourth European Conference on Computer Supported Cooperative Work, 10 – 14 September 1995 Stockholm, Sweden

[For96] Doug Forguson
Programming with the MFC Document Template Architecture
C++ Report Vol.:8 Iss.: 3

[FS86] Gregg Foster, Mark Stefik
Theory and practice of a Colab-orative tool.
Proceedings of the CSCW Conference 1986 Austin Texas S. 7-15

[GG97a] Carl Gutwin, Saul Greenberg
Workspace Awareness
ACM CHI'97 Workshop on Awareness in Collaborative Systems Atlanta, Georgia März 1997

[GG97b] Carl Gutwin, Saul Greenberg
Effects of Awareness Support on Groupware Usability
ACM CHI'97 Workshop on Awareness in Collaborative Systems Atlanta, Georgia März 1997

[GG97c] Carl Gutwin, Saul Greenberg
Studying Awareness in Contact Facilitation
ACM CHI'97 Workshop on Awareness in Collaborative Systems Atlanta, Georgia März 1997

[GHJ+95] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides
Design Patterns. Elements of Reusable Object-Oriented Software
Addison-Wesley Publishing Company 1995

[GL96] Guido Gryczan , Carola Lilienthal
Kommunikation in verteilten Systemen
Universität Hamburg, Softwaretechnik Center 1996

[Gos95] Crispin Goswell
The COM Programmer's Cookbook
Microsoft Office Product Unit 1995

[Gre86] Irene Greif
Data Sharing in Group Works
Proceedings of the CSCW Conference 198 Austin Texas S 175-183

[Gre88] Irene Greif
Computer-Supported Cooperative Work: A book of Readings
Morgan Kaufmann Publishers, Inc. 1988

[GS88] Irene Greif, Sunil Sarin
Computer-based Real-Time Conferencing Systems
aus : Computer-Supported Cooperative Work : A book of readings S. 397 – 420
Morgan Kaufman Publishers, Inc. 1988 San Mateo California

[Gry96] Guido Gryczan
Prozeßmuster zur Unterstützung kooperativer Tätigkeit
Deutscher Universitätsverlag 1996

[GWZ96] G. Gryczan, M. Wulf, H. Züllighoven
Prozeßmuster für die situierte Koordination kooperativer Arbeit
In: H. Krcmar, H. Lewe, G. Schwabe (Hrsg.): Herausforderung Telekooperation, D-CSCW'96, Springer 1996, S. 89 - 103.

[JV87] E. Jessen, R. Valk
Rechensysteme, Grundlagen der Modellbildung
Springer Verlag 1987

[KGZ94] Klaus Kilberth, Guido Gryczan, Hein Züllighoven
Objektorientierte Anwendungsentwicklung
Vieweg Verlag 1994

[Kir94] Thomas Kirsche
Kooperation und Koordination in verteilten Systemen
aus Verteilte Systeme – Grundlagen und zukünftige Entwicklung, Hartmut Wedekind, WI
Wissenschaftsverlag 1994

[Kra98] Tim Krauss
Softwarekonstruktion nach WAM unter Nutzung des MFC Rahmenwerkes.
Diplomarbeit an der Universität Hamburg, Fachbereich Informatik, Arbeitsbereich
Softwaretechnik 1998

[Kru96] David J. Kruglinski
Inside Visual C++
Microsoft Press 1996

[Lam96] R. Lam
A C++ CGI framework
Report-Nr.: RC 20460

[Lew95] Simon Lewis
The MVC Architecture
aus The Art and Science of Smalltalk S. 91–102
Prentice Hall 1995

[LZ96] C. Lilienthal, H. Züllighoven
Techniques and Tools for Continuous User Participation
In: J. Blomberg, F. Kensing, E. Dykstra-Erickson (Eds.): PDC'96 Proceedings of the Participatory Design Conference. Cambridge, Massachusetts, USA. 13 - 15 November 1996. pp. 153 – 159.

[LS96] Carola Lilienthal, Wolfgang Strunk
Documenting frameworks by visualizing dynamics
Universität Hamburg, Arbeitsbereich Softwaretechnik, Beitrag zur Tools 96

[Maa94] Susanne Maaß
Transparenz – Eine zentrale Software-ergonomische Forderung
Bericht Nr. 170 Fachbereich Informatik Universität Hamburg 1994

[Mar97a] Robert C. Martin
Cross Casting : The Capsule Pattern
C++ Report Vol 9 / No 6 Juni 1997

[Mar97b] Robert C. Martin
UML Tutorial : Class Diagrams
C++ Report Vol 9 / No 8 September 1997

[Mat97] Mirko Matytschak
Verteilte Anwendungen mit OLE-Remote-Automation
Objekt Spektrum 5/97 Seite 62 - 69

[Mey90] Bertrand Meyer
Objektorientierte Softwareentwicklung
Hanser / Prentice Hall 1990

[Mey95] Meyers grosses Taschenlexikon in 24 Bänden
B.I. Taschenbuchverlag 1995

[Mic97a] Microsoft
Visual C++ 5.0 , OLE Glossar

[Mic97b] Microsoft
Visual C++ Online Help
Visual C++ Version 5.0 1997

[Mic97c] Microsoft
The Component Objct Model : Technical Overwiew

[Pre96] Wolfgang Pree
Frameworks – Past, present, future
Object Magazine Vol 6 Iss 3 1996 S. 24 - 26

[OHE96] R. Orfali, D. Harkey, J. Edwards
The Essential Distributed Objects Survival Guide
Wiley & Sons 1996

[Oni96a] Fritz Onion
Object Persistence in MFC
C++ Report Vol.:8 Iss.: 1

[Oni96b] Fritz Onion
Dynamic Data Structure Serialization in MFC
C++ Report Vol.:8 Iss.: 3

[Rat97] Rational Rose
UML Notation Guide V. 1.0.1
Rational http://www.rational.com

[Rie93] Dirk Riehle
Objektorientierte Architektur nach WAM am Beispiel von SANE
Studienarbeit an der Universität Hamburg, Fachbereich Informatik, Arbeitsbereich Softwaretechnik. 1993

[Rie95] Dirk Riehle
Muster am Beispiel der Werkzeug und Material Metapher
Diplomarbeit an der Universität Hamburg, Fachbereich Informatik, Arbeitsbereich Softwaretechnik. 1995

[Rob93] Mike Robinson
Design for unanticipated use...
Proceedings of the Third European Conference on Computer-Supported Cooperative Work 13-17 September 1993, Mailand S. 187 - 202

[Rog97] Dale Rogerson
Inside COM
Microsoft Press 1997

[RW96] Stefan Roock, Henning Wolf
Konzeption und Implementierung eines Reaktionsmusters für objektorientierte Softwaresysteme
Studienarbeit an der Universität Hamburg, Fachbereich Informatik, Arbeitsbereich Softwaretechnik. 1996

[RW98] Stefan Roock, Henning Wolf
Die Raummetapher zur Entwicklung kooperationsunterstützender Softwaresysteme für Organisationen
Diplomarbeit an der Universität Hamburg, Fachbereich Informatik, Arbeitsbereich Softwaretechnik. Januar 1998

[RZ95] Dirk Riehle, Heinz Züllighoven
A Pattern Language for Tool Construction and Integration based on The Tools and Materials Metaphor
In: Pattern Languages of Program Design Hrsg: J. Coplien, D. C. Schmidt
Addison Wesley 1995 S. 9 - 42

[SBL+86] M.Stefik, D.G.Bobrow, S.Laming, D. Tatar
WYSIWIS Revised. Early Experiences with Multi-User Interfaces.
Proceedings of the CSCW Conference 1986 Austin Texas S.276-290

[Sch96] Alexander Schill
Rechnergestützte Gruppenarbeit in verteilten Systemen
Prentice Hall 1996

[Sør88] Pål Sørgaard
Object oriented programming and computerised shared materials
Proceedings of the second European Conference on Object Oriented Oriented Programming (ECOOP'88) Oslo August 1988

[Sør89] Pål Sørgaard
A Framework for Computer Supported Cooperative Work
Report ISSN 0105-8517 März 1989

[Ste96] Dominik Stein
Definition und Klassifikation der Begriffswelt um CSCW, Workflow Computing, Workgroup Computing, Groupware, Workflow Management
Seminararbeit an der Universität Gesamthochschule Essen 1996

[Vli97a] John Vlissides
Multicast
C++ Report Vol 9/ No 8 September 1997

[Vli97b] John Vlissides
Multicast – Observer = Typed Messages
C++ Report Vol 9 / No 10 November – December 1997

[WW94] Martina Wulf, Dirk Weske
Konzepte zur Materialversorgung verteilter Werkzeugumgebungen am Beispiel der Anbindung einer objektorientierten Datenbank.
Studienarbeit an der Universität Hamburg, Fachbereich Informatik, Arbeitsbereich Softwaretechnik 1994

Anhang B Visionen des Pausenplanersystems

1 Einleitung

Die vorliegenden Visionen repräsentieren einen Teil der verschiedenen Dokumente des objektorientierten Entwurfsprozesses nach WAM . Ein weiterer Teil wurde uns von dem AB SWT der Uni Hamburg zur Verfügung gestellt. Dazu gehört das Pausenplaner Szenario, die CRC-Karten und die Systemvision vom 9. Oktober 1996, sowie das Szenario „Pausenaufsichtsplan erstellen“ und die Systemvision „Gemeinsame Nutzung durch Pausenplanersteller und Sekretariat“. Hinzugefügt haben wir, neben diesen Visionen, eine Handlungsstudie sowie das Glossar.

2 Überblicksvision des Pausenplanersystems

Die Pausen einer Schule müssen beaufsichtigt werden. Für die Beaufsichtigung sind die Lehrer des Lehrkörpers zuständig. Das Pausenplanersystem soll dem Pausenplanersteller helfen, eine zulässige und sinnvolle Pausenplanung durchführen zu können. Das Szenario „Pausenaufsichtsplan erstellen“ beschreibt, wie diese Aufgabe bisher gelöst wurde. Die in dem Szenario beschriebene Vorgehensweise der Erstellung von Lehrerkarten und die Belegung eines Pausenplanes mit diesen Lehrerkarten soll durch das Pausenplanersystem voll unterstützt werden. Die Arbeitsmittel Lehrerkarten und Pausenplan werden im Rechner durch Materialien modelliert. Das System soll die Teilaufgaben, die notwendig sind, um dieser Aufgabe nachzukommen, vereinfachen.

- Das Durchführen einer Pausenbelegung wird, wie bisher, durch Ablegen einer Lehrerkarte auf einem Übersichtsplan realisiert. Einziger Unterschied besteht darin, daß Plan und Karten auf dem Bildschirm zu sehen sind. Das Feststellen von Konflikten in der Pausenzuordnung, welches bisher sehr kompliziert gewesen ist, wird dem Benutzer durch das System abgenommen. Er wird sofort nach erfolgter Belegung auf diesen Konflikt hingewiesen.

- Die Berechnung der Aufsichtspflichten wird dem Benutzer durch das System abgenommen.

- Die Pflege des Lehrkörpers ist einfach zu realisieren. Kam im bisherigen System ein Lehrer neu an die Schule, so mußte der Pausenplanersteller mehrfach Lehrerkarten ausfüllen, sowie manuell eine Sortierung vornehmen. Im zukünftigen System beschränken sich die notwendigen Tätigkeiten auf die Eingabe, der für den Lehrer relevanten Informationen.

- Analog zu der Pflege des Lehrkörpers, gestaltet sich die Pflege der Pausen ebenso einfach und effizient. Eine Veränderung in den Pausen führt zur sofortigen Anpassung des Pausenplanes.

- Das Erstellen von Kopien eines Pausenplanes, um ihn beispielsweise im Lehrerzimmer aufzuhängen oder jedem Lehrer eine Kopie zukommen zu lassen, läßt sich leicht durch mehrfaches Ausdrucken realisieren.

Jedem Benutzer wird durch das Pausenplanersystem ein virtuelles schwarzes Brett am Bildschirm präsentiert. Alle Benutzer müssen sich mit ihren Namen beim System anmelden. Das schwarze Brett ist ein allgemein zugänglicher Kooperationsort, an dem jeder Benutzer die vorgefundenen Materialien betrachten oder bearbeiten kann. Benutzer können die angehefteten Materialien mit den Werkzeugen, welche sie am schwarzen Brett vorfinden, bearbeiten. Sie können neues Material anheften oder nicht mehr benötigtes Material abnehmen. Jedes Material erhält nach der Bearbeitung durch einen Benutzer einen Stempel, in dem der Zeitpunkt der letzten Betrachtung, der letzten Bearbeitung und der Name des Benutzers vermerkt ist. Eine genauere Beschreibung des schwarzen Brettes findet sich in der Werkzeugvision schwarzes Brett in Kapitel 6.1.

Am schwarzen Brett des Pausenplanersystems gibt es die Werkzeuge **Lehrkörperbearbeiter**, **Pausenlistenbearbeiter** und den **Pausenplaner**. Sie werden durch Symbole repräsentiert. Im System gibt es die Materialien Pausenplan und Lehrkörper. Auch sie werden in Form von Symbolen am schwarzen Brett sichtbar gemacht. Von diesen Materialien kann es mehrere Exemplare geben (siehe Abbildung 30).

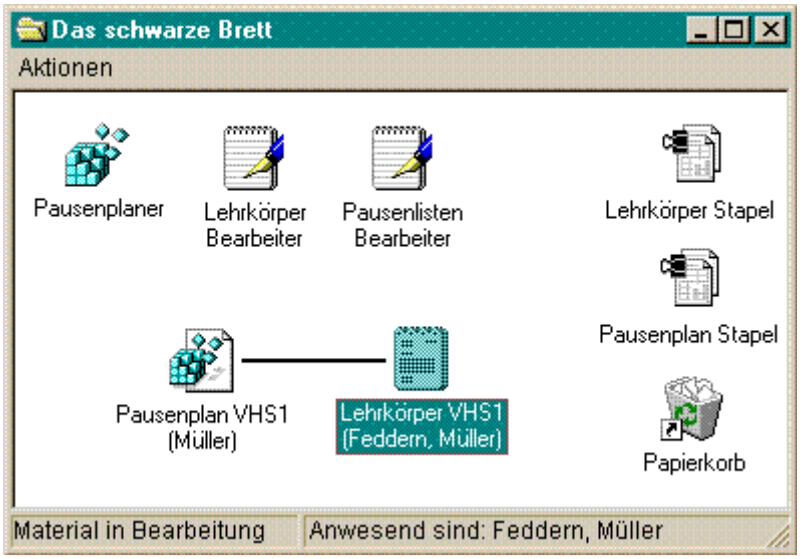


Abbildung 30: Oberflächenprototyp für das schwarze Brett Werkzeug

3 Einbettungsvision

Es ist eine Einbettung des Pausenplanersystems in das MFC Rahmenwerk vorgesehen. Es wird auf dem Betriebssystem Windows NT oder Windows 95 aufsetzen. Das Pausenplanersystem ist ein Mehrbenutzersystem. Mehrere Benutzer können gleichzeitig mit dem System arbeiten. Die Verteilung soll in technisch Hinsicht durch das DCOM Objektmodell realisiert werden.

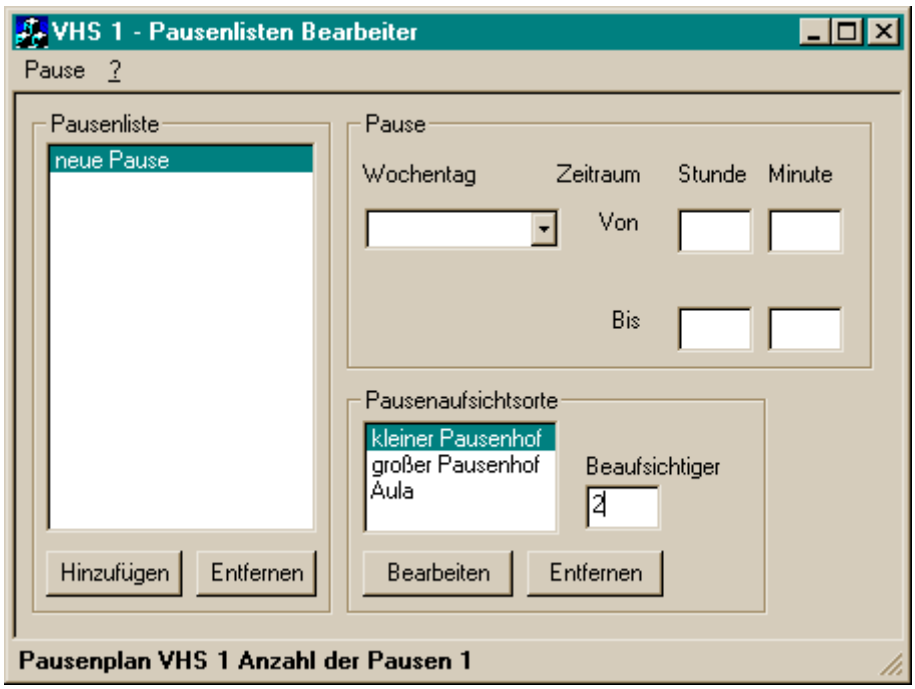
4 Fachliche Ablaufvision: Lehrkörperbearbeitung durch das Sekretariat

Diese Ablaufvision stützt sich im wesentlichen auf die gleichnamige Handlungsstudie (siehe Handlungsstudie Bearbeitung des Lehrkörpers durch das Sekretariat). Eine Veränderung des Lehrkörpers durch den Sekretariatsmitarbeiter (siehe Fachliche Ablaufvision: Lehrkörperbearbeitung durch das Sekretariat Kapitel 4) hat zur Folge, daß der Pausenplan auf seine Konsistenz geprüft werden muß (siehe Materialvision Pausenplan Kapitel 7.7). Diese Prüfung wird vom Pausenplanersteller vorgenommen. Er erkennt durch die Farbe des Pausenplansymbols, daß eine Überprüfung durchzuführen ist. Anhand des Materialstempels kann er sehen, welcher Benutzer den Lehrkörper bearbeitet hat und zu welchem Zeitpunkt die Bearbeitung erfolgte. Sobald er die Überprüfung abgeschlossen hat (siehe Handhabungsvision neuen Pausenplan bearbeiten Kapitel 5.3) wird das Symbol des Pausenplanes wieder in seiner ursprünglichen Farbe angezeigt. Die neue Darstellung signalisiert dem Sekretariatsmitarbeiter, daß der Pausenplanersteller auf die Veränderung des Lehrkörpers reagiert hat. Er kann auch anhand des Materialstempels erkennen, wer den Lehrkörper zuletzt sondiert hat und zu welchem Zeitpunkt die Betrachtung erfolgte.

5 Handhabungsvisionen

5.1 Handhabungsvision Erstellen eines neuen Pausenplanes

Um einen neuen Pausenplan zu erstellen, zieht der Pausenplanersteller mit der Maus einen neuen Pausenplan vom Pausenplanstapel des schwarzen Brettes und läßt ihn auf eine frei Stelle fallen. Es erscheint ein Symbol mit der Bezeichnung „neuer Pausenplan“. Die Bezeichnung des Pausenplanes kann nun geändert werden, indem der Pausenplanersteller die Schrift unterhalb des Symbols mit der Maus anklickt und eine andere Bezeichnung einträgt. Um die Pausen der Schule in den Pausenplan einzutragen, startet der Pausenplanersteller die Bearbeitung des neuen Pausenplanes mit dem Pausenlistenbearbeiter Werkzeug, indem er das Symbol des Pausenplanes auf das Werkzeugsymbol zieht. Das Starten der Bearbeitung kann auch ausgeführt werden, indem er das Werkzeug auf das Material zieht.



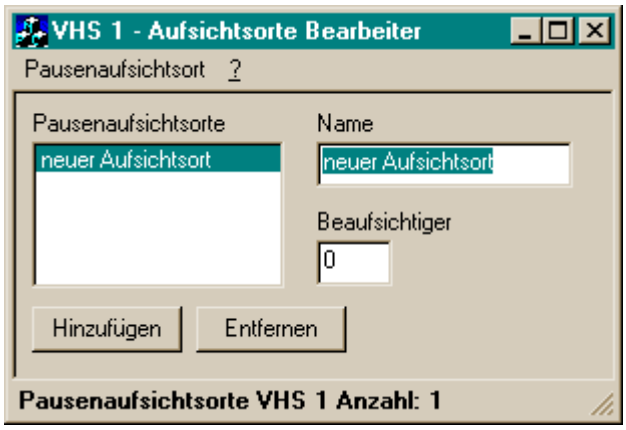


Abbildung 31: Oberflächenprototypen für die Pausenlisten- und Aufsichtsortebearbeiter Werkzeuge

Der Pausenplanersteller definiert nun die verschiedenen Pausenaufsichtsorte. Hierzu wählt er im Menü des Pausenlistenbearbeiter Werkzeugs „Pause/Aufsichtsorte bearbeiten“ und startet dadurch das Werkzeug Aufsichtsortebearbeiter. Er klickt auf den „Hinzufügen“ Knopf oder wählt im Menü „Aufsichtsort/hinzufügen“. In der Liste der Aufsichtsorte erscheint ein neuer Eintrag mit der Bezeichnung „neuer Aufsichtsort“. Er kann diese Bezeichnung sowie die Mindestanzahl der Pausenbeaufsichtiger, welche für diesen Ort als Voreinstellung dienen sollen, verändern (siehe Werkzeugvision Aufsichtsortebearbeiter Kapitel 6.4). Nachdem der Pausenplanersteller alle Aufsichtsorte in das System eingegeben hat, beendet er die Bearbeitung durch Klicken auf den „Beenden“ Systemknopf. Dieser befindet sich ganz rechts auf der oberen Fensterleiste.

Nun fügt er die Pausen im Werkzeug Pausenlistenbearbeiter hinzu. Der Pausenplanersteller klickt auf den „Hinzufügen“ Knopf oder wählt den Menüpunkt „Pause/Hinzufügen“. In der Pausenliste erscheint daraufhin eine neue Pause mit der Darstellung „neue Pause“. Diese ist sofort selektiert und der Pausenplanersteller kann jetzt, im rechten Teil des Fensters, Wochentag und Anfangs- bzw. Endzeitpunkt mit Stunde und Minute eingeben. Daraufhin ändert sich die Darstellung der Pause in der Pausenliste. Als Voreinstellung sind alle Pausenaufsichtsorte der Schule in der Ortsliste vermerkt. Zu jedem Aufsichtsort gilt zunächst die voreingestellte Anzahl der Pausenbeaufsichtiger, so wie sie im Aufsichtsortebearbeiter Werkzeug eingegeben wurde. Der Pausenplanersteller kann nun nicht gewünschte Pausenaufsichtsorte aus der Liste entfernen, in dem er den zu entfernenden Ort in der Liste selektiert und auf den „Entfernen“ Knopf der Pausenaufsichtsorte klickt. Er kann auch die Anzahl der Pausenbeaufsichtiger pro Pausenort im „Beaufsichtigter“ Textfeld ändern.

Der Pausenplanersteller beendet die Bearbeitung der Pausenliste, nachdem er alle Pausen hinzugefügt hat (siehe Abbildung 32: Pausenlistenbearbeiterwerkzeug nach bearbeiteter Pausenliste). Dazu klickt er auf den „Beenden“ Systemknopf oder wählt den Menüpunkt „Pause/Bearbeitung beenden“.

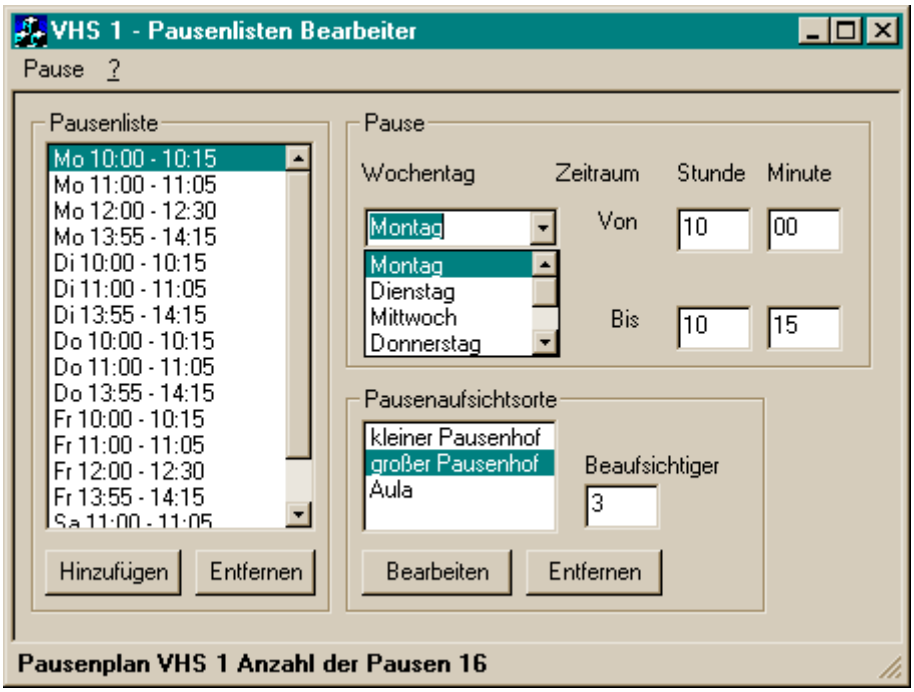


Abbildung 32: Pausenlistenbearbeiterwerkzeug nach bearbeiteter Pausenliste

5.2 Handhabungsvision Erstellen eines neuen Lehrkörpers

Der Benutzer zieht einen neuen Lehrkörper vom Lehrkörperstapel des schwarzen Brettes und läßt ihn auf eine freie Stelle fallen. Es erscheint ein neues Lehrkörpersymbol mit dem Namen „neuer Lehrkörper. Die Bezeichnung des Lehrkörpers kann nun geändert werden, indem der Benutzer die Schrift unterhalb des Symbols mit der Maus anklickt und eine andere Bezeichnung einträgt. Danach startet er die Bearbeitung des noch leeren Lehrkörpers mit dem Lehrkörperbearbeiter Werkzeug, indem er das Symbol des Lehrkörpers auf das Werkzeug zieht. Nun kann der Benutzer die Lehrer der Schule in den Lehrkörper aufnehmen.

Dazu klickt der Benutzer auf den „Aufnehmen“ Knopf oder er wählt den Menüpunkt „Lehrer/Aufnehmen“. In der Lehrerliste erscheint daraufhin ein neuer Lehrer mit dem Namen „neuer Lehrer“. Dieser ist sofort selektiert und der Benutzer kann jetzt, im rechten Teil des Fensters, den echten Namen und die Stelle des Lehrers eingeben. Daraufhin ändert sich die Darstellung des Namens in der Lehrerliste.

Die Liste der Ausschlußzeiträume ist zunächst leer. Um einen neuen Ausschlußzeitraum für den Lehrer angeben zu können, muß der Benutzer auf den „Zeitraum Hinzufügen“ Knopf klicken oder den Menüpunkt „Ausschlußzeitraum/Hinzufügen“ wählen. In der Liste der Ausschlußzeiträume erscheint daraufhin ein neuer Ausschlußzeitraum mit der Darstellung „neuer Ausschlußzeitraum“. Dieser ist sofort selektiert und der Benutzer kann jetzt im rechten Teil des Fensters den Anfangs- und Endzeitpunkt mit Wochentag, Stunde und Minute eingeben. Daraufhin ändert sich die Darstellung des Zeitraumes in der Liste der Ausschlußzeiträume. Der neue Zeitraum erscheint jetzt in textueller Form gleich den anderen Zeiträume in der Liste. Um einen Ausschlußzeitraum eines Lehrers wieder zu entfernen, muß der Benutzer in der Liste der Ausschlußzeiträume den gewünschten Ausschlußzeitraum selektieren und dann auf den „Zeitraum Entfernen“ Knopf klicken oder den Menüpunkt „Ausschlußzeitraum/Entfernen“ wählen.

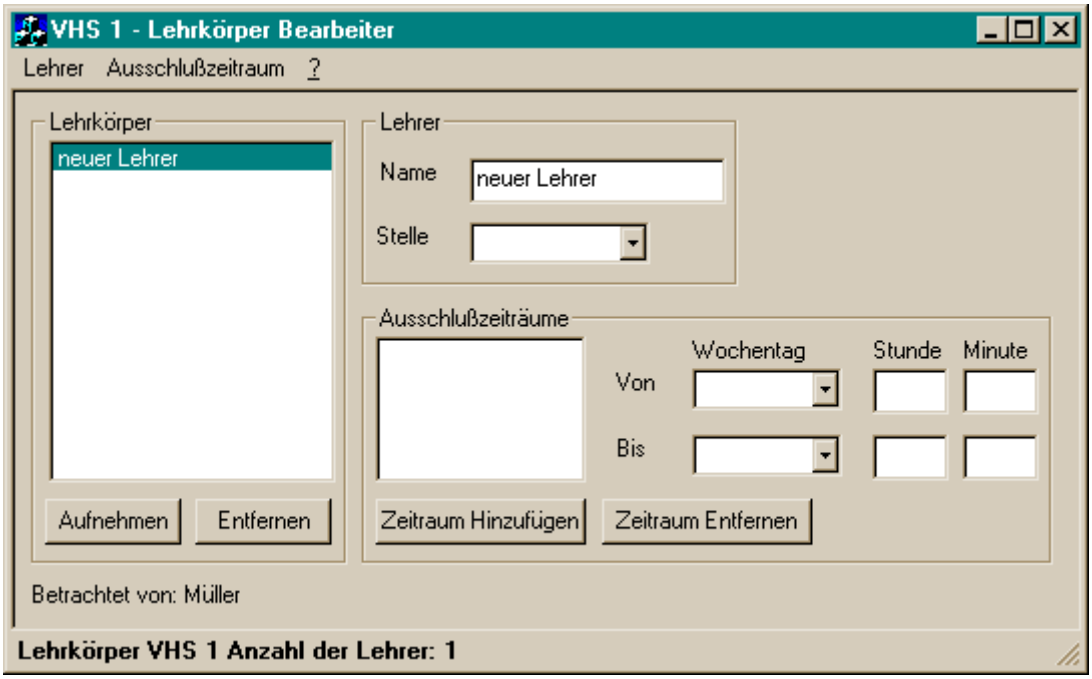


Abbildung 33: Oberflächenprototyp des Werkzeugs Lehrkörperbearbeiter

Hat der Benutzer alle gewünschten Lehrer in den Lehrkörper aufgenommen (siehe Abbildung 34: Lehrkörperbearbeiter nach der Bearbeitung), so klickt er auf den „Beenden“ Systemknopf oder wählt den Menüpunkt „Lehrer/Bearbeitung beenden“.

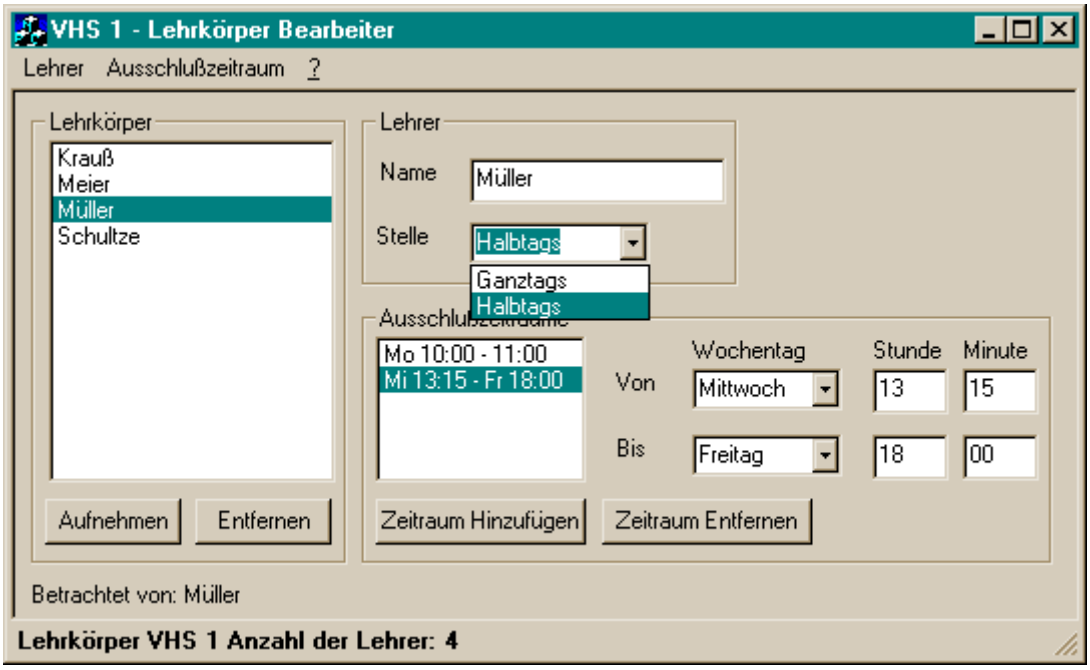


Abbildung 34: Lehrkörperbearbeiter nach der Bearbeitung

5.3 Handhabungsvision neuen Pausenplan bearbeiten

Bevor der Pausenplanersteller mit der Bearbeitung des neuen Pausenplanes beginnen kann, muß er den Pausenplan zunächst mit einem Lehrkörper am schwarzen Brett verknüpfen. Dazu zieht er das Lehrkörpersymbol mittels der Maus auf des Pausenplansymbol. Es erscheint eine Linie zwischen dem Symbol des Pausenplanes und dem Symbol des Lehrkörpers (siehe Abbildung 30).

Der Pausenplanersteller startet nun die Bearbeitung des Materials Pausenplan mit dem Pausenplaner Werkzeug. Dazu zieht er das Symbol des Pausenplanes auf das Symbol des Pausenplaners. Der Lehrkörper und der noch leere Pausenplan werden daraufhin im Werkzeug dargestellt (siehe Abbildung 35: Oberflächenprototyp für das Pausenplaner Werkzeug).

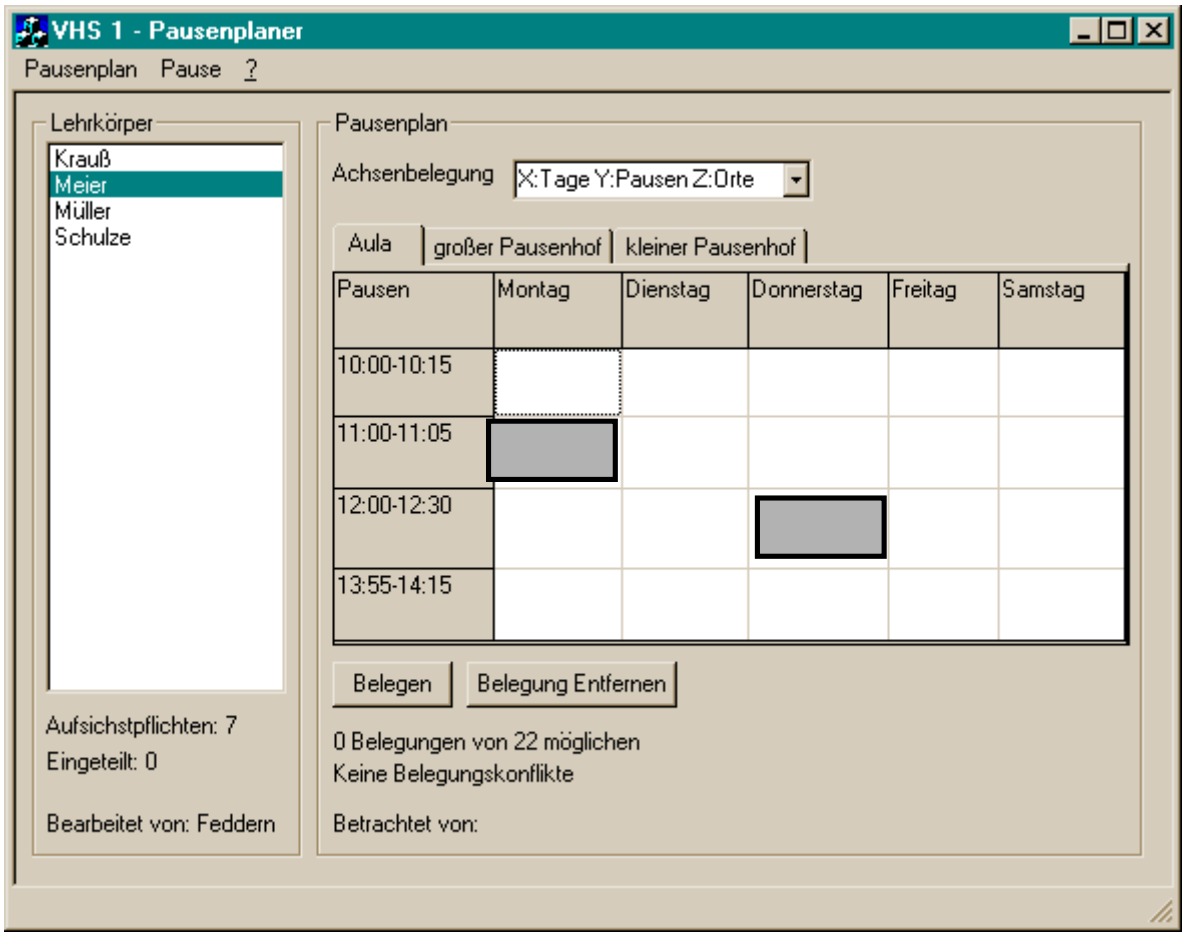


Abbildung 35: Oberflächenprototyp für das Pausenplaner Werkzeug

Der Benutzer kann die Darstellung von Pausen, Wochentagen und Orten im Pausenplan beliebig einstellen. Dazu klickt er mit der Maus auf die Auswahlbox „Achsenbelegung“ und kann die gewünschten Belegung für die x-, y- und z-Achse wählen. Die Materialpräsentation des Pausenplans ändert sich daraufhin sofort. Abbildung 35 zeigt die Einstellung „X: Tage Y: Pausen Z:

Orte“. Mit dieser Einstellung befinden sich die Pausenaufsichtsorte auf den Aktenreitern. Der Pausenplanersteller wählt nun den zu bearbeitenden Aufsichtsort, indem er mit der Maus auf den gewünschten Karteireiter klickt.

Jetzt kann der Pausenplanersteller die Pausenbelegung durchführen. Um eine Pause mit einem Lehrer aus dem Lehrkörper zu belegen, muß der Pausenplanersteller in der Lehrerliste einen Lehrer selektieren. Die Statuszeile zeigt an, wie viele Aufsichtspflichten der Lehrer hat und wie oft er bereits eingeteilt wurde. Im Pausenraster werden die Felder, in denen der Lehrer Ausschlußzeit-räume hat, farbig hervorgehoben. Jetzt kann der Pausenplanersteller mittels Drag & Drop den Lehrer auf ein Feld des Pausenplanes ziehen. Bei diesem Vorgang kann der Pausenplanersteller zusätzlich, anhand der Darstellung des Mauszeigers erkennen, ob ein Belegen des Feldes überhaupt möglich ist oder ob ein Konflikt besteht. Das Belegen der Pause mit dem Lehrer ist im Konfliktfall dennoch möglich. Der Lehrername wird dann in diesem Feld farbig hervorgehoben.

Zieht der Pausenplanersteller den Lehrer auf ein schon belegtes Feld, so wird die alte Belegung vorher entfernt. Der Pausenplanersteller kann auch Belegungen innerhalb des Pausenplans mittels Drag & Drop verschieben. Ein Ziehen auf ein schon belegtes Feld bewirkt dann einen Lehrertausch der Pausenbelegungen.

Der Pausenplanersteller wird im Pausenplaner Werkzeug über den Zustand des Pausenplanes informiert. Ihm wird angezeigt, wie viele Pausenbelegungen für eine vollständige Pausenplanung notwendig sind, die Anzahl der bereits getätigten Pausenbelegungen sowie die Anzahl der vorliegenden Belegungskonflikte. Belegungskonflikte eines Pausenplanes werden auch im Symbol des Pausenplanmaterials am schwarzen Brett sichtbar.

Der Pausenplanersteller beendet die Bearbeitung des Pausenplanes nachdem er die notwendigen Pausenbelegungen durchgeführt hat (siehe Abbildung 36: Pausenplaner nach der Bearbeitung). Dazu klickt er auf den „Beenden“ Systemknopf oder wählt den Menüpunkt „Pause/Bearbeitung beenden“.

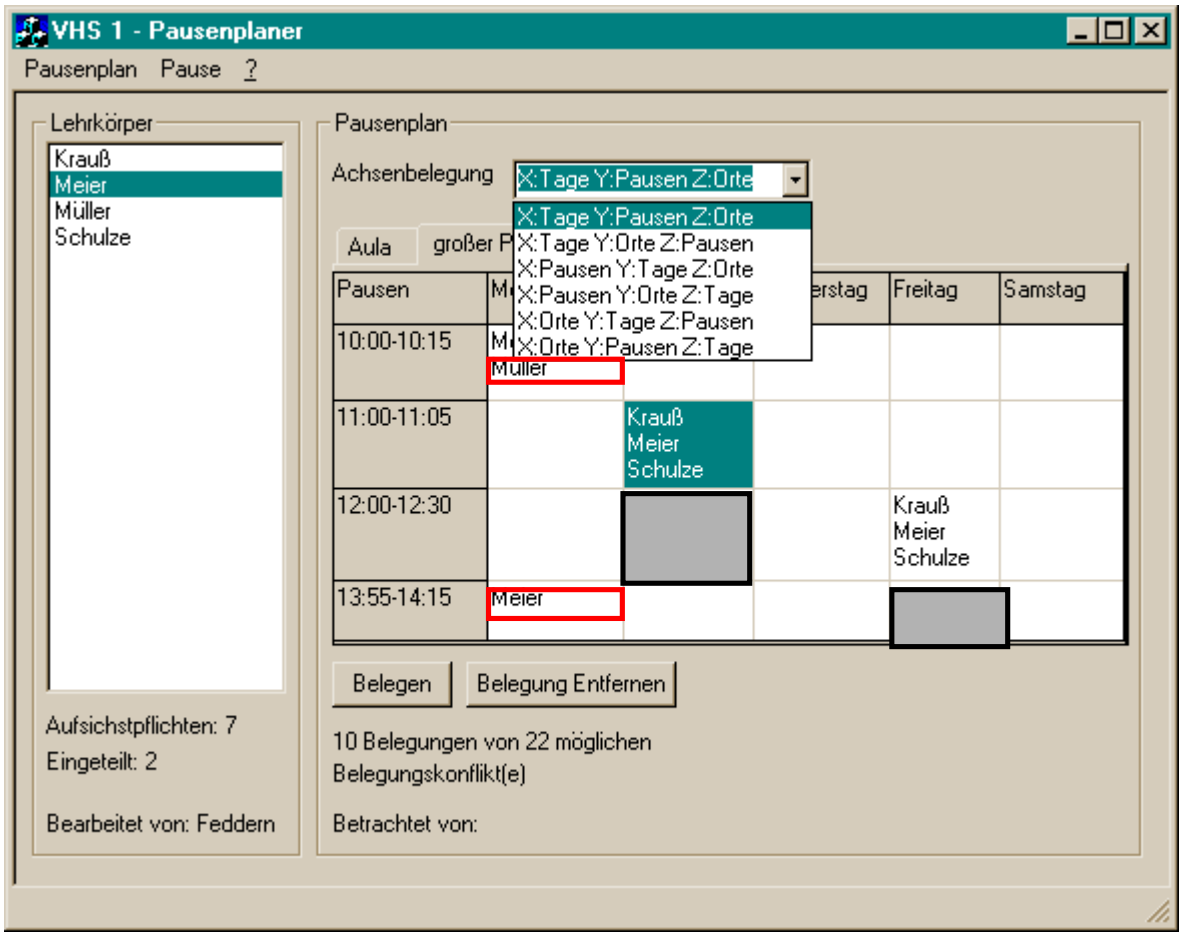


Abbildung 36: Pausenplaner nach der Bearbeitung

6 Werkzeugvisionen

6.1 Werkzeugvision schwarzes Brett

Das schwarze Brett ist ein allgemein zugänglicher Ort. Benutzer können sich dorthin bewegen und die Materialien betrachten oder bearbeiten. Es können sich gleichzeitig mehrere Benutzer dort einfinden um verschiedene Tätigkeiten auszuführen. Sie müssen, sofern sie gleichzeitig aktiv sind, miteinander kooperieren. Aus diesem Grund nennen wird das schwarze Brett auch Kooperation-sort. Benutzern wird vergegenwärtigt, welche anderen Benutzer sich zur selben Zeit am schwarzen Brett befinden. Daher müssen Benutzer im System namentlich bekannt sein. Die Benutzer können die angehefteten Materialien mit den Werkzeugen, welche sie am schwarzen Brett vorfinden, bearbeiten. Sie können neues Material anheften oder nicht mehr benötigtes Material abnehmen.

Das schwarze Brett soll synchrone und asynchrone Kooperation der Benutzer ermöglichen. Man kann am eigenen Bildschirm erkennen, welche anderen Benutzer sich zur selben Zeit am schwar-

zen Brett befinden und welche Materialien sie bearbeiten. Die Benutzer können nacheinander oder simultan auf die Materialien zugreifen. Ein gemeinsamer lesender Zugriff auf ein Material ist möglich, gemeinsamer schreibender Zugriff soll jedoch nicht gestattet sein. Das bedeutet, daß nur ein Benutzer das Material bearbeiten kann. Es können aber zur gleichen Zeit beliebig viele Benutzer das Material betrachten. Ob der Benutzer auf ein Material schreibenden oder lesenden Zugriff hat, bestimmt das verwendete Werkzeug. Die gemeinsame Bearbeitung eines Materials, kann den beteiligten Benutzern durch das Werkzeug mitgeteilt werden, wenn dies im Anwendungskontext sinnvoll erscheint.

Über die Arbeitsumgebung des schwarzen Brettes werden einzelne Veränderungen an einem Material durch ein bearbeitendes Werkzeug, wie z.B. eine Pausenbelegung, den anderen Werkzeugen des Pausenplanersystems, die das gleiche Material betrachten, sofort gemeldet. Alle Benutzer sehen somit immer dem aktuellen Materialzustand.

6.1.1 Materialpräsentation

Jedem Benutzer wird durch das Pausenplanersystem ein virtuelles schwarzes Brett am Bildschirm präsentiert. In der unteren Fensterleiste werden die Namen der Benutzer aufgelistet, die sich zur selben Zeit ebenfalls am Brett befinden. Werkzeuge, Materialien und Materialstapel werden durch Symbole repräsentiert. Diese sind mit Namen versehen, die unter den Symbolen sichtbar sind. Befindet sich ein Material in Bearbeitung oder wird es betrachtet, so erscheint unter dem Symbol in Klammern auch die Namen der Bearbeiter und Betrachter. In Abbildung 37 befinden sich Feddern und Müller am schwarzen Brett. Feddern bearbeitet den „Lehrkörper VHS 1(Feddern, Meier)“, welcher weiterhin noch von Müller betrachtet wird. Der Name „Feddern“ des Bearbeiters wird zusätzlich in der linken Hälfte der Statuszeile angegeben. „Müller“ ist der Name des Benutzers, der das Material zur Zeit betrachtet.

Jedes Symbol auf dem schwarzen Brett hat ein Kontextmenü. Dieses wird durch Klicken mit der rechten Maustaste auf das Symbol sichtbar. Im Kontextmenü der Materialien befinden sich weiterhin Informationen über den Zeitpunkt der letzten Bearbeitung und den Zeitpunkt der letzten Betrachtung. Ebenso die Namen der Bearbeiter oder Betrachter. Diese Informationen werden aktualisiert, sobald der letzte Benutzer die Bearbeitung beziehungsweise die Betrachtung des Materials abgeschlossen hat. So können andere Benutzer erkennen, wer zuletzt das Material bearbeitet oder betrachtet hat.

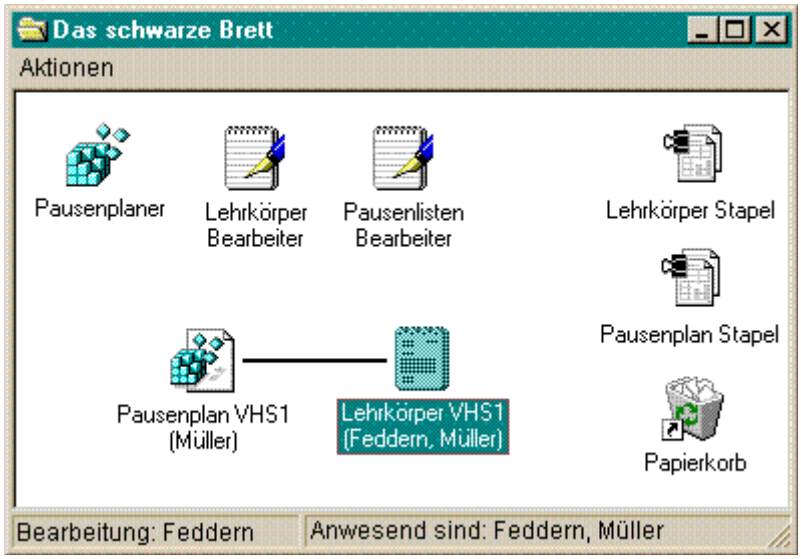


Abbildung 37: Oberflächenprototyp des Werkzeugs Schwarzes Brett

6.1.2 Aktionen des Benutzers

Neue Materialien anheften:

Will der Benutzer neue Materialien an das schwarze Brett anheften, so zieht er mit der Maus vom gewünschten Materialstapel ein neues Material und läßt es an einer freien Stelle fallen. Zu jedem Materialtyp gibt es einen eigenen Stapel. Es erscheint ein Symbol mit der Bezeichnung „neuer/neue <Materialtyp>“. Die Bezeichnung des Materials kann nun geändert werden, indem der Benutzer die Schrift unterhalb des Symbols mit der Maus anklickt und eine andere Bezeichnung einträgt.

Materialien abnehmen:

Will der Benutzer nicht mehr benötigte Materialien vom schwarze Brett abnehmen, so zieht er mit der Maus das gewünschte Material auf das Papierkorb Symbol. Eine Sicherheitsabfrage folgt und fragt den Benutzer, ob er das Material tatsächlich vernichten will. Materialien, welche eine Verknüpfung besitzen können nicht vom schwarzen Brett abgenommen werden. Will der Benutzer verknüpfte Materialien abnehmen, so muß er zuvor die Verknüpfung lösen.

Material bearbeiten oder Betrachten:

Das Bearbeiten oder Betrachten eines Materials durch ein Werkzeug kann durch einen Drag & Drop Vorgang mit den Symbolen auf dem schwarzen Brett initiiert werden. Es wird entweder ein Werkzeug auf ein Material gezogen oder ein Material auf ein Werkzeug. Kann ein Werkzeug ein Material nicht bearbeiten so wird dies angezeigt. Andernfalls öffnet sich das Werkzeug und die Bearbeitung des Materials kann beginnen.

Materialien verknüpfen:

Im Pausenplanersystem ist nur die Verknüpfung von Pausenplan und Lehrkörper vorgesehen. Der Benutzer initiiert die Verknüpfung, indem er den Lehrkörper mittels der Maus auf den Pausenplan zieht. Der Pausenplan darf vorher keine Verknüpfung besitzen. Es erscheint eine Linie zwischen dem Symbol des Pausenplanes und dem Symbol des Lehrkörpers. Der Pausenplan benutzt nach einer getätigten Verknüpfung den Lehrkörper. Lehrer aus dem Lehrkörper können nun die Pausen des Pausenplanes beaufsichtigen. Ein Lehrkörper kann mit mehreren Pausenplänen verknüpft sein.

Verknüpfung von Materialien lösen:

Will der Benutzer nicht mehr benötigte Verknüpfungen zwischen Pausenplan und Lehrkörper lösen, so klickt er mit der rechten Maustaste auf das Symbol des beteiligten Pausenplan Materials. Daraufhin erscheint das Kontextmenü und der Benutzer wählt dann „Verknüpfung lösen“. Eine Sicherheitsabfrage folgt und fragt den Benutzer, ob er die Verknüpfung tatsächlich lösen will.

Schwarzes Brett verlassen:

Will der Benutzer die Bearbeitung oder Betrachtung der Materialien beenden, so klickt er auf den „Beenden“ Systemknopf oder wählt den Menüpunkt „Aktionen/schwarzes Brett verlassen“. Ein Verlassen ist nur möglich, wenn zum Zeitpunkt des Verlassens keine Materialien bearbeitet werden. Ist dies dennoch der Fall, so muß der Benutzer die Bearbeitung des Materials beenden.

6.2 Werkzeugvision Lehrkörperbearbeiter

Mit dem Werkzeug Lehrkörperbearbeiter kann der Benutzer das Material Lehrkörper (Siehe Materialvision Lehrkörper Kapitel 7.5) erstellen oder bestehendes Material pflegen. Neue Lehrer (Siehe Materialvision Lehrer Kapitel 7.1) können in den Lehrkörper aufgenommen werden. Von der Schule abgehende Lehrer können aus dem Lehrkörper entfernt werden. Name, Stelle und Ausschlußzeiträume eines Lehrers können verändert werden. Der Lehrkörper wird als Teil des Materials Pausenplan (Siehe Materialvision Pausenplan Kapitel 7.7) im Kapitel 6.4.2) verwendet.

6.2.1 Materialpräsentation

Eine Liste aller Lehrer, die sich im Lehrkörper befinden, wird auf der linken Seite des Fensters gezeigt. Die Namen der Lehrer werden alphabetisch geordnet aufgelistet. Der Benutzer kann in der Liste einen Lehrernamen selektieren. Dadurch werden Name, Stelle und Ausschlußzeiträume des selektierten Lehrers auf der rechten Seite des Fensters gezeigt. Die Ausschlußzeiträume werden in

einer Liste geordnet dargestellt. In der Statuszeile wird angezeigt, wie viele Lehrer sich momentan im Lehrkörper befinden. Der Benutzer kann im unteren Teil des Werkzeuges sehen, wer das Lehrkörper Material im Moment betrachtet.

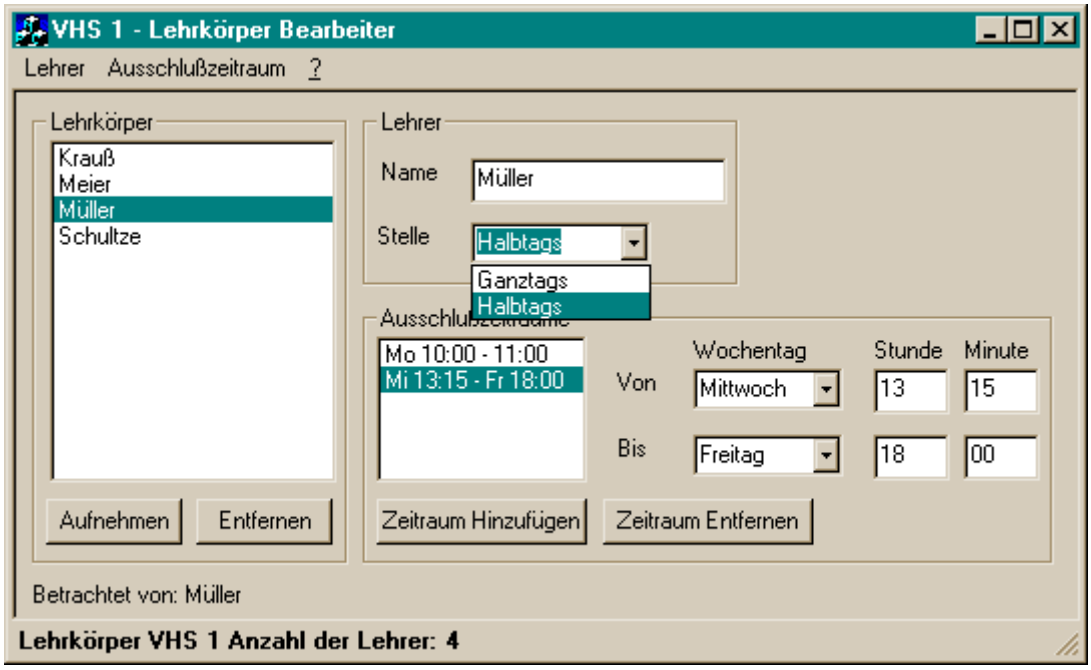


Abbildung 38: Oberflächenprototyp des Werkzeugs Lehrkörperbearbeiter

6.2.2 Aktionen des Benutzers

Neuen Lehrer in den Lehrkörper aufnehmen:

Der Benutzer klickt auf den „Aufnehmen“ Knopf oder er wählt den Menüpunkt „Lehrer/Aufnehmen“. In der Lehrerliste erscheint daraufhin ein neuer Lehrer mit dem Namen „neuer Lehrer“. Dieser ist sofort selektiert und der Benutzer kann jetzt, im rechten Teil des Fensters, den echten Namen und die Stelle des Lehrers eingeben. Daraufhin ändert sich die Darstellung des Namens in der Lehrerliste. Die Liste der Ausschlußzeiträume ist zunächst leer. Um einen neuen Ausschlußzeitraum für den Lehrer angeben zu können, muß der Benutzer auf den „Zeitraum Hinzufügen“ Knopf klicken oder den Menüpunkt „Ausschlußzeitraum/Hinzufügen“ wählen. (siehe Operation Ausschlußzeitraum eines Lehrers hinzufügen)

Lehrer aus dem Lehrkörper entfernen:

Um einen Lehrer aus dem Lehrkörper zu entfernen, muß der Benutzer in der Lehrerliste einen Lehrer selektieren und dann auf den „Entfernen“ Knopf klicken oder den Menüpunkt „Leh-

rer/Entfernen“ wählen. Eine Sicherheitsabfrage folgt und fragt den Benutzer, ob der Lehrer tatsächlich entfernt werden soll.

Ausschlußzeitraum eines Lehrers hinzufügen:

Der Benutzer klickt auf den „Zeitraum Hinzufügen“ Knopf oder er wählt den Menüpunkt „Ausschlußzeitraum/Hinzufügen“. In der Liste der Ausschlußzeiträume erscheint daraufhin ein neuer Ausschlußzeitraum mit der Darstellung „neuer Ausschlußzeitraum“. Dieser ist sofort selektiert und der Benutzer kann jetzt im rechten Teil des Fensters den Anfangs- und Endzeitpunkt mit Wochentag, Stunde und Minute eingeben. Daraufhin ändert sich die Darstellung des Zeitraumes in der Liste der Ausschlußzeiträume. Der neue Zeitraum erscheint jetzt in textueller Form gleich den anderen Zeiträume in der Liste.

Ausschlußzeitraum eines Lehrers entfernen:

Um einen Ausschlußzeitraum eines Lehrers zu entfernen, muß der Benutzer in der Liste der Ausschlußzeiträume den gewünschten Ausschlußzeitraum selektieren und dann auf den „Zeitraum Entfernen“ Knopf klicken oder den Menüpunkt „Ausschlußzeitraum/Entfernen“ wählen.

Name eines Lehrers verändern:

Um den Namen eines Lehrers zu verändern, muß der Benutzer in der Lehrerliste den gewünschten Lehrer selektieren. Daraufhin werden die Eigenschaften des ausgewählten Lehrers angezeigt. Jetzt kann der Name in dem Textfeld verändert werden. Die Änderung erscheint sofort in der Lehrerliste.

Stelle eines Lehrers verändern:

Um die Stelle eines Lehrers zu verändern, muß der Benutzer in der Lehrerliste den gewünschten Lehrer selektieren. Daraufhin werden die Eigenschaften des ausgewählten Lehrers angezeigt. Jetzt kann die Art der Stelle in der Auswahlbox verändert werden.

Ausschlußzeitraum eines Lehrers verändern:

Um den Ausschlußzeitraum eines Lehrers zu verändern, muß der Benutzer in der Lehrerliste den gewünschten Lehrer selektieren. Daraufhin werden die Ausschlußzeiträume des ausgewählten Lehrers innerhalb der Liste der Ausschlußzeiträume angezeigt. Der Benutzer wählt nun den gewünschten Ausschußzeitraum aus dieser Liste. Jetzt können Anfangs- und Endzeitpunkt des Ausschlußzeitraums in den Textfeldern und den Auswahlboxen verändert werden. Die Änderungen erscheinen sofort in der Liste der Ausschlußzeiträume.

Bearbeitung beenden:

Der Benutzer klickt auf den „Beenden“ Systemknopf oder wählt den Menüpunkt „Lehrer/Bearbeitung beenden“.

6.3 Werkzeugvision Pausenlistenbearbeiter

Mit dem Pausenlistenbearbeiter Werkzeug kann der Pausenplanersteller die Pausenliste bestehender Pausenpläne pflegen. Pausen (Siehe Materialvision Pause Kapitel 7.4) können zur Pausenliste hinzugefügt oder entfernt werden. Der Zeitraum, die Aufsichtsorte und die Anzahl der Pausenbeaufsichtiger einer Pause können verändert werden. Die Pausenliste wird als Teil des Materials Pausenplan (Siehe Materialvision Pausenplan Kapitel 6.4.2) verwendet.

6.3.1 Materialpräsentation

Eine Liste aller Pausen wird auf der linken Seite des Fensters gezeigt. Die Zeiträume der Pausen werden geordnet aufgelistet. Der Pausenplanersteller kann in der Liste eine Pause selektieren. Dadurch werden Wochentag, Zeitraum und eine Liste der Pausenaufsichtsorte der selektierten Pause auf der rechten Seite des Fensters gezeigt. Die Liste der Pausenaufsichtsorte enthält neben den Namen derselben ebenso die Anzahl der Pausenbeaufsichtiger. Selektiert der Pausenplanersteller einen Ort aus dieser Liste, so wird die Anzahl der Pausenbeaufsichtiger in einem Textfeld angezeigt und kann dort verändert werden.

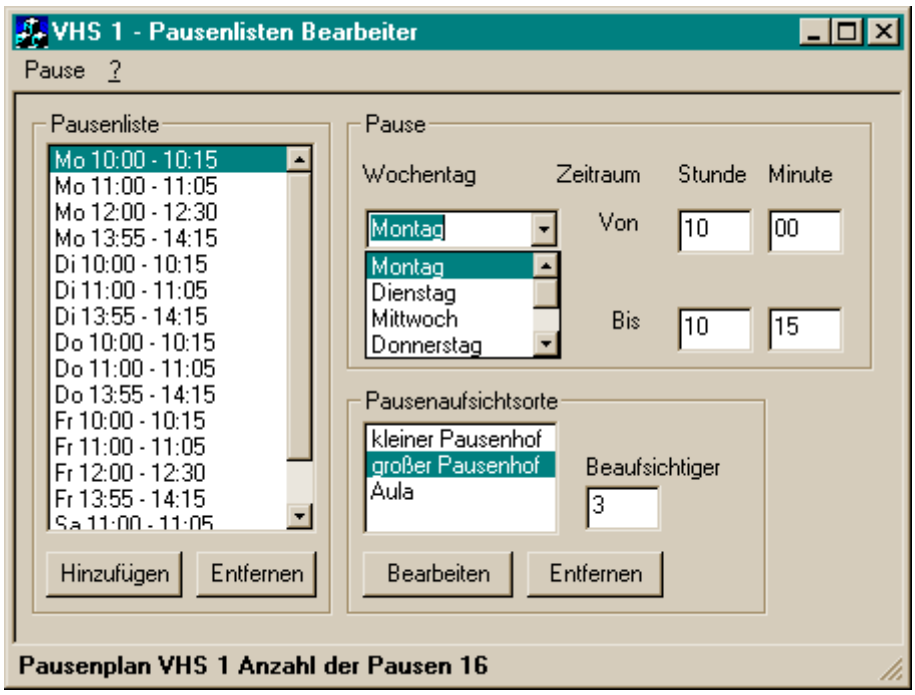


Abbildung 39: Oberflächenprototyp des Werkzeugs Pausenlistenbearbeiter

6.3.2 Aktionen des Pausenplanerstellers

Pause zur Pausenliste hinzufügen:

Der Pausenplanersteller klickt auf den „Hinzufügen“ Knopf oder wählt den Menüpunkt „Pause/Hinzufügen“. In der Pausenliste erscheint daraufhin eine neue Pause mit der Darstellung „neue Pause“. Diese ist sofort selektiert und der Pausenplanersteller kann jetzt, im rechten Teil des Fensters, Wochentag und Anfangs- bzw. Endzeitpunkt mit Stunde und Minute eingeben. Daraufhin ändert sich die Darstellung der Pause in der Pausenliste. Als Voreinstellung sind alle Pausenaufsichtsorte der Schule in der Ortsliste vermerkt. Zu jedem Aufsichtsort gilt die voreingestellte Anzahl der Pausenbeaufsichtiger (siehe Werkzeugvision Aufsichtsortebearbeiter Kapitel 6.4). Der Pausenplanersteller kann nun die Pausenaufsichtsorte, die in dieser Pause nicht beaufsichtigt werden müssen, aus der Liste entfernen (siehe unten „Pausenaufsichtsort einer Pause entfernen“). Er kann auch die Anzahl der Pausenbeaufsichtiger pro Pausenort ändern (siehe unten „Anzahl der Pausenbeaufsichtiger an einem Pausenaufsichtsort ändern“).

Pause aus der Pausenliste entfernen:

Um eine Pause aus der Pausenliste zu entfernen, muß der Pausenplanersteller in der Pausenliste eine Pause selektieren und dann auf den „Entfernen“ Knopf klicken oder den Menüpunkt „Pause/Entfernen“ wählen.

Zeitraum einer Pause verändern:

Um den Zeitraum einer Pause zu verändern, muß der Pausenplanersteller in der Pausenliste die gewünschte Pause selektieren. Daraufhin wird der Zeitraum der ausgewählten Pause angezeigt. Der Pausenplanersteller kann nun den Wochentag in der Auswahlbox verändern. Anfangs- und Endzeitpunkt der Pause können in den Textfeldern verändert werden. Die Änderungen erscheinen sofort in der Pausenliste.

Pausenaufsichtsorte eines Pausenplans bearbeiten:

Pausenaufsichtsorte eines Pausenplanes werden mit dem Aufsichtsortebearbeiter Werkzeug bearbeitet (siehe Werkzeugvision Aufsichtsortebearbeiter Kapitel 6.4). Um das Werkzeug zu starten muß der Pausenplanersteller auf den „Bearbeiten“ Knopf der Pausenaufsichtsorte klicken oder im Menü „Pause/Aufsichtsorte bearbeiten“ wählen.

Pausenaufsichtsort einer Pause hinzufügen:

Um einen Pausenaufsichtsort zu einer Pause hinzuzufügen, muß der Pausenplanersteller in der Pausenliste die gewünschte Pause selektieren. Danach wählt er im Menü „Pause/Aufsichtsort hinzufügen“. Es erscheint ein Untermenü, welches neben den einzelnen Pausenorten auch einen

Menüpunkt „alle Orte“ enthält. Durch die Auswahl eines Menüpunktes kann er der Pause einen oder alle Aufsichtsorte hinzufügen. Das Hinzufügen von schon zugeordneten Pausenaufsichtsorten bleibt wirkungslos.

Pausenaufsichtsort einer Pause entfernen:

Der Pausenplanersteller selektiert in der Pausenliste die gewünschte Pause. Daraufhin werden alle Pausenaufsichtsorte angezeigt. Der Pausenplanersteller selektiert in der Liste den zu entfernenden Ort und klickt auf den „Entfernen“ Knopf der Pausenaufsichtsorte oder wählt im Menü „Pause/Aufsichtsort entfernen“. Der Pausenaufsichtsort wird gelöscht.

Anzahl der Pausenbeaufsichtiger an einem Pausenaufsichtsort ändern:

Der Pausenplanersteller selektiert in der Pausenliste die gewünschte Pause. Daraufhin werden alle Pausenaufsichtsorte angezeigt. Der Pausenplanersteller selektiert in der Liste den gewünschten Ort und kann nun die Anzahl der Pausenbeaufsichtiger im Textfeld ändern.

Bearbeitung beenden:

Der Pausenplanersteller klickt auf den „Beenden“ Systemknopf oder wählt den Menüpunkt „Pause/Bearbeitung beenden“.

6.4 Werkzeugvision Aufsichtsortebearbeiter

Mit dem Aufsichtsortebearbeiter Werkzeug kann der Pausenplanersteller die Orte, an denen in der Schule Pausenaufsichten wahrgenommen werden müssen, eingeben und pflegen. Er kann neue Pausenaufsichtsorte hinzufügen oder entfernen. Zu jedem Pausenaufsichtsort kann er die Anzahl der voreingestellten Pausenaufsichtsorte festlegen. (siehe Materialvision Pause Kapitel 7.4).

6.4.1 Materialpräsentation

Eine Liste aller Pausenaufsichtsorte wird auf der linken Seite des Fensters gezeigt. Die Namen der Orte werden alphabetisch geordnet aufgelistet. Der Pausenplanersteller kann in der Liste einen Pausenort selektieren. Dadurch werden Name und die Anzahl der Pausenbeaufsichtiger des selektierten Ortes auf der rechten Seite des Fensters gezeigt. In der Statuszeile wird angegeben, wie viele Pausenaufsichtsorte bisher eingegeben wurden .

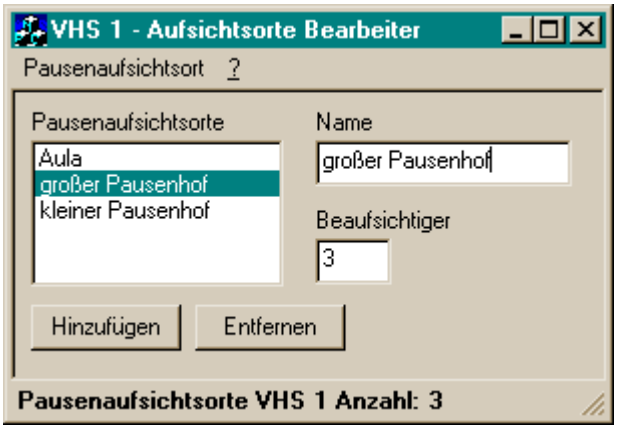


Abbildung 40: Oberflächenprototyp des Werkzeugs Aufsichtsortebearbeiter

6.4.2 Aktionen des Pausenplanerstellers

Pausenaufsichtsort hinzufügen:

Der Pausenplanersteller klickt auf den „Hinzufügen“ Knopf oder er wählt den Menüpunkt „Pausenaufsichtsort/Hinzufügen“. In der Liste erscheint daraufhin eine neuer Ort mit der Darstellung „neuer Aufsichtsort“. Dieser ist sofort selektiert und der Pausenplanersteller kann jetzt, im rechten Teil des Fensters, den Namen und die voreingestellte Anzahl der Pausenbeaufsichtiger eingeben. Daraufhin ändert sich die Darstellung des Ortes in der Liste.

Pausenaufsichtsort entfernen:

Um einen Ort aus der Liste zu entfernen, muß der Pausenplanersteller in der Liste einen Pausenaufsichtsort selektieren und dann auf den „Entfernen“ Knopf klicken oder den Menüpunkt „Pausenaufsichtsort/Entfernen“ wählen.

Namen eines Pausenaufsichtsortes verändern:

Der Pausenplanersteller muß in der Liste den gewünschten Pausenaufsichtsort selektieren. Daraufhin wird der Name des ausgewählten Ortes in der Textbox angezeigt. Der Pausenplanersteller kann nun den Namen in der Textbox verändern. Die Änderungen erscheinen sofort in der Liste.

Anzahl der voreingestellten Pausenbeaufsichtiger an einem Pausenaufsichtsort ändern:

Der Pausenplanersteller muß in der Liste den gewünschten Pausenaufsichtsort selektieren. Daraufhin werden alle Pausenaufsichtsorte angezeigt. Daraufhin wird die Anzahl der voreingestellten Pausenbeaufsichtiger des ausgewählten Ortes in der Textbox angezeigt. Der Pausenplanersteller kann nun die Anzahl in der Textbox verändern.

Bearbeitung beenden:

Der Pausenplanersteller klickt auf den „Beenden“ Systemknopf oder wählt den Menüpunkt „Pausenaufsichtsort/Bearbeitung beenden“.

6.5 Werkzeugvision Pausenplaner

Mit dem Pausenplaner Werkzeug kann der Pausenplanersteller alle Pausenaufsichten der Schule an die Lehrer des Lehrkörpers vergeben. Die Vergabe wird im Material Pausenplan (Siehe Materialvision Pausenplan Kapitel 7.7) festgehalten. Eine Pausenaufsicht vergibt der Pausenplanersteller dadurch, das er eine Pause aus der Pausenliste mit einem Lehrer des Lehrkörpers belegt. Der Pausenplanersteller kann Belegungskonflikte erkennen und daraufhin Belegungen entfernen oder tauschen. Die verwendete Pausenliste des Pausenplans wird im Pausenlistenbearbeiter Werkzeug (Siehe Werkzeugvision Pausenlistenbearbeiter Kapitel 6.3) verwaltet. Der verwendete Lehrkörper des Pausenplans wird im Lehrkörperbearbeiter Werkzeug (Siehe Werkzeugvision Lehrkörperbearbeiter Kapitel 6.2) verwaltet.

6.5.1 Materialpräsentation

Der Lehrkörper des Pausenplans wird in einer Liste auf der linken Seite des Fensters gezeigt. Die Namen der Lehrer werden dort alphabetisch geordnet aufgelistet. Unter der Liste werden die Aufsichtspflichten des gerade selektierten Lehrers angezeigt. Man kann auch sehen, wie viele Aufsichtspflichten man schon eingeteilt hat. Darunter kann man sehen, wer das Lehrkörper Material im Moment bearbeitet.

Der Pausenplan wird in einem Raster auf der rechten Seite des Fensters gezeigt. Zeilen und Spalten dieses Rasters sind dynamisch veränderbar und passen sich der Pausenliste des Pausenplans an. Das bedeutet, daß im Plan beliebige Wochentage, Aufsichtsorte und Zeiträume für Pausen dargestellt werden können. Pausen mit gleichen Zeiträumen werden in einer Zeile des Rasters dargestellt. Dadurch kann es vorkommen, daß einzelne Felder einer Zeile gesperrt sind, da an den Wochentagen dieser Felder keine Pause zu dieser Zeit vorgesehen ist.

Der Benutzer kann die Darstellung von Pausen, Wochentagen und Orten im Pausenplan beliebig einstellen. Dazu klickt er mit der Maus auf die Auswahlbox „Achsenbelegung“ und kann die gewünschten Belegung für die x-, y- und z-Achse wählen. Die Materialpräsentation des Pausenplans ändert sich daraufhin sofort. Abbildung 41: Oberflächenprototyp für das Werkzeug Pausenplaner zeigt die Einstellung „X: Tage Y: Pausen Z: Orte“. Mit dieser Einstellung befinden sich die Pausenaufsichtsorte auf den Aktenreitern. Unter den Knöpfen „Belegen“ und „Belegung Entfernen“ wird die Gesamtzahl der Belegungen des Pausenplans und die Anzahl der schon getätigten

Belegungen aufgezeigt. Hier kann man auch sehen, ob Belegungskonflikte bestehen oder nicht. Darunter kann der Pausenplanersteller sehen, wer das Pausenplan Material im Moment betrachtet.

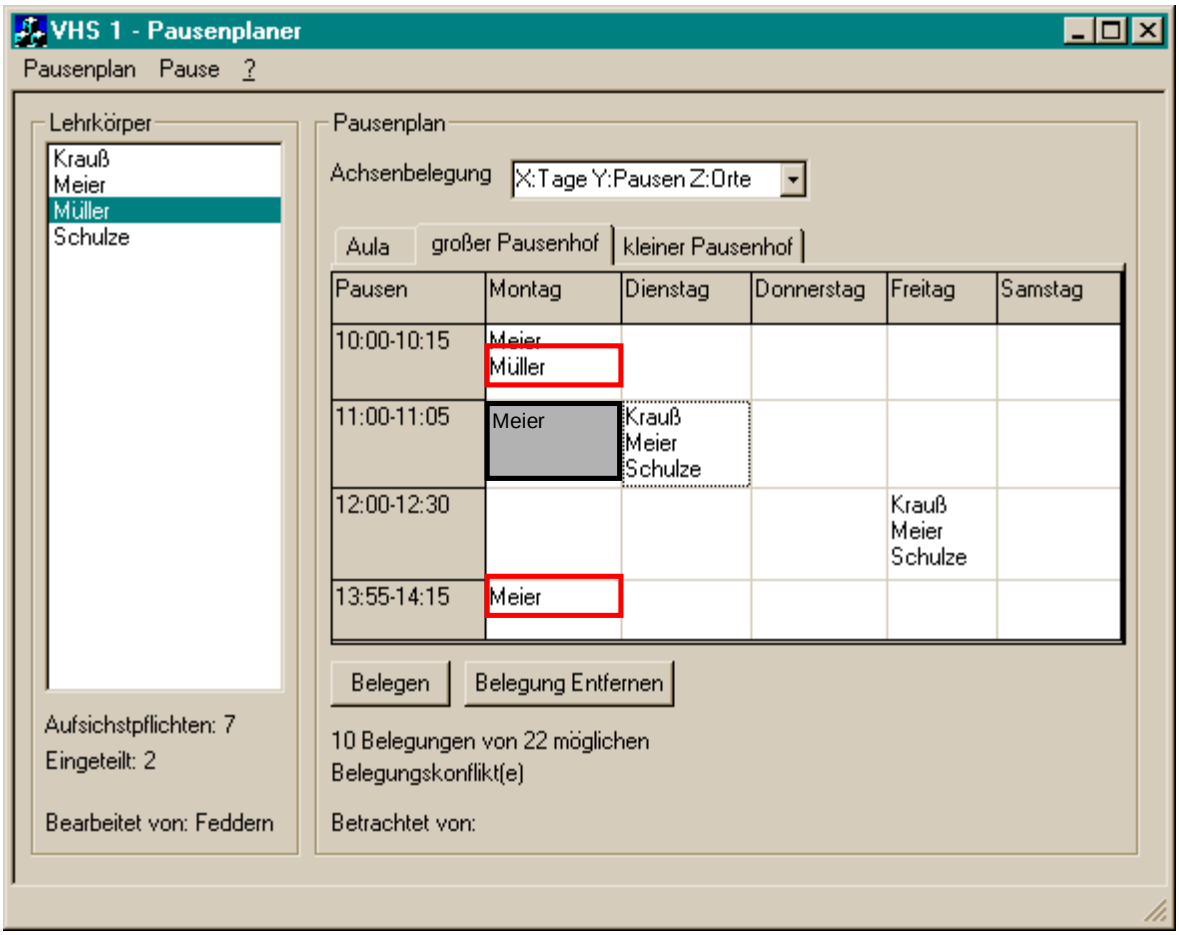


Abbildung 41: Oberflächenprototyp für das Werkzeug Pausenplaner

6.5.2 Aktionen des Pausenplanerstellers

Pause mit Lehrer belegen:

Um eine Pause mit einem Lehrer aus dem Lehrkörper zu belegen, muß der Pausenplanersteller in der Lehrerliste einen Lehrer selektieren. Die Statuszeile zeigt an, wie viele Aufsichtspflichten der Lehrer hat und wie oft er schon eingeteilt wurde. Im Pausenraster werden die Felder, in denen der Lehrer Ausschlußzeiträume hat, farbig hervorgehoben. Jetzt kann der Pausenplanersteller mittels Drag & Drop den Lehrer auf ein Feld des Pausenplanes ziehen. Bei diesem Vorgang kann der Pausenplanersteller zusätzlich, anhand der Darstellung des Mauszeigers erkennen, ob ein Belegen des Feldes überhaupt möglich ist, oder ob ein Konflikt besteht. Das Belegen der Pause mit dem Lehrer ist im Konfliktfall dennoch möglich. Das Feld wird dann farbig hervorgehoben. Zieht der

Pausenplanersteller den Lehrer auf ein schon belegtes Feld, so wird die alte Belegung vorher entfernt. Der Pausenplanersteller kann auch Belegungen innerhalb des Pausenplans mittels Drag & Drop verschieben. Ein Ziehen auf ein schon belegtes Feld bewirkt dann einen Lehrertausch der Pausenbelegungen.

Mehrere Pausen mit einem Lehrer belegen:

Der Pausenplanersteller selektiert ein oder mehrere Felder des Pausenplanes und klickt dann auf den „Belegen“ Knopf oder er wählt den Menüpunkt „Pause/Mit Lehrer belegen“. Die ausgewählten Pausen werden jetzt mit dem in der Lehrerliste selektierten Lehrer belegt. Alte Belegungen werden vorher entfernt.

Belegung einer Pause entfernen:

Der Pausenplanersteller selektiert ein oder mehrere Felder des Pausenplanes und klickt dann auf den „Entfernen“ Knopf oder wählt den Menüpunkt „Pause/Belegung entfernen“. Die Belegung sämtlicher ausgewählter Pausen wird daraufhin entfernt.

Alle Pausenbelegungen entfernen:

Der Pausenplanersteller wählt den Menüpunkt „Pausenplan/Alle Belegungen entfernen“. Nach einer Sicherheitsabfrage, ob der Pausenplanersteller den Pausenplan tatsächlich leeren möchte, werden sämtliche Pausenbelegungen entfernt.

Auswahl eines Pausenaufsichtsortes (Vorausgesetzte Einstellung „X: Tage Y: Pausen Z: Orte“):

Über dem Pausenplan gibt es für jeden Pausenaufsichtsort der Schule einen Aktenreiter. Die Namen der Aufsichtsorte sind auf den Aktenreitern zu sehen. Der Pausenplanersteller wählt den gewünschten Pausenaufsichtsort durch einen Klick auf den richtigen Aktenreiter. Daraufhin erscheint der Pausenplan für den ausgewählten Ort[41].

Pausenplan drucken:

Der Pausenplanersteller wählt den Menüpunkt „Pausenplan/Drucken“. Der Pausenplan wird ausgedruckt.

Bearbeitung beenden:

[41] Es sollte evtl. möglich sein ein zweites Fenster des Pausenplans zu öffnen, um zwei Pausenaufsichtsorte parallel zu sehen und zu bearbeiten.

Der Pausenplanersteller klickt auf den „Beenden“ Systemknopf oder wählt den Menüpunkt „Pausenplan/Bearbeitung beenden“.

7 Materialvisionen

7.1 Materialvision Lehrer

Das Material Lehrer repräsentiert den Lehrer einer Schule. Dieser Lehrer ist charakterisiert durch seinen Namen und die Stelle, die er bekleidet. Neben der Aufgabe, an der Schule zu unterrichten ist er, im Rahmen seiner Stelle, verpflichtet, eine gewisse Anzahl von Pausenaufsichten zu führen. Es gibt ganze, halbe und viertel Stellen. Aufgrund seiner Stelle, oder auch anderer Gründe, ist ein Lehrer nicht immer an der Schule anwesend. Die Informationen über die Zeiten, zu denen er nicht verfügbar ist, werden in Ausschlußzeiträumen festgehalten.

- Veränderung des Lehrers

  Die Eigenschaften eines Lehrers werden im Rahmen des Pausenplanersystems bearbeitet. Name und Stelle des Lehrers soll der Benutzer mit Hilfe eines Werkzeuges verändern können. Ebenso kann er die Ausschlußzeiträume des Lehrers bearbeiteten. Es können Ausschlußzeiträume hinzugefügt und entfernt werden.

- Verfügbarkeit eines Lehrers

  Soll ein Lehrer die Aufsicht einer Pause übernehmen, so wird die Pause mit ihm belegt. In diesem Zusammenhang kann der Lehrer darüber Auskunft geben, ob er innerhalb eines Zeitraumes einsatzfähig ist. So würde es wenig Sinn machen, einen Lehrer für die Pause von 8.40 einzuteilen, obwohl er erst in der vierten Stunde (um 11.00) das erste Mal Unterricht gibt und demnach zur Pausenzeit nicht anwesend ist.

7.2 Materialvision Aufsichtsort

Ein Pausenaufsichtsort ist ein Bereich in einer Schule, in dem sich die Schüler während der Pause aufhalten. In einer Pause muß eine definierte Mindestanzahl von Pausenbeaufsichtigern zugegen sein. Die Anzahl der einzuteilenden Vertreter stimmt in der Regel mit diesem Wert überein.

- Veränderung der Mindestanzahl von Pausenbeaufsichtigern.

  Die Mindestanzahl von Pausenbeaufsichtigern muß verändert werden können.

- Veränderung der Bezeichnung eines Aufsichtsortes.

7.3 Materialvision Pausenaufsichtsort

Ein Pausenaufsichtsort ist ein Material, welches sich auf einen Aufsichtsort bezieht und in einer Pause mit einer Mange von Beaufsichtigern und Vertretern belegt wird.

- Veränderung der Mindestanzahl von Pausenbeaufsichtigern.

  Die Mindestanzahl von Pausenbeaufsichtigern muß verändert werden können. Geschieht dieses, so wird die durch den Aufsichtsort vorgegebene Voreinstellung überschrieben.

- Zuordnung von Lehrern

  Einem Pausenaufsichtsort, als Eigenschaft einer Pause, werden eine Menge von Lehrern zugeordnet.

- Zuordnung von Vertretern

  Dem Pausenaufsichtsort muß ebenso wie Lehrer, eine Menge von Vertretern zugeordnet werden können.

7.4 Materialvision Pause

Eine Pause bezieht sich auf einen wöchentlich wiederkehrenden Zeitraum[42]. Sie ist dadurch gekennzeichnet, daß dieser Zeitraum beaufsichtigt werden muß. An einer Schule kann es mehrere Aufsichtsorte geben. Es existieren beispielsweise verschiedene Pausenhöfe, beziehungsweise es muß sowohl innerhalb als auch außerhalb des Schulgebäudes Aufsicht geführt werden. Die Beaufsichtigung eines Pausenaufsichtsortes erfordert einen oder mehrere Lehrer. Im Krankheitsfall eines Pausenbeaufsichtigers muß eine Vertretung eingeteilt sein. Diese übernimmt dann die Beaufsichtigung der Pause. Die Mindestanzahl von Pausenbeaufsichtigern kann durch eine untere Grenze festgelegt werden. Diese ist mit der Anzahl der eingeteilten Vertretungen identisch.

Eine Pausenaufsicht wird eingeteilt, indem ein Aufsichtsort einer Pause mit Lehrern belegt wird.

- Veränderung einer Pause

  Da jede Schule eine eigene Pauseneinteilung vornimmt, muß der Zeitraum und die Mindestanzahl von Pausenbeaufsichtigern einer Pause bearbeitet werden können. Diese Eigenschaften sollen auch nachträglich verändert werden können.

---

[42] Siehe dazu den Abschnitt über den wöchentlich wiederkehrenden Zeitraum

- Belegung eines Aufsichtsortes einer Pause mit einem Lehrer (Pausenaufsicht oder Vertretung der Pausenaufsicht)

  Im Falle der Belegung einer Pause mit einem Lehrer, wird der Lehrer bei der Pause mit dem gewünschten Pausenaufsichtsort eingetragen.

- Feststellen eines Konfliktes

  Durch die Belegung eines Aufsichtsortes einer Pause kann ein Belegungskonflikt entstehen, wenn sich die Pausenzeit mit einem Ausschlußzeitraum des Lehrers überschneidet. Die Pause kann diesen Konflikt feststellen.

7.5 Materialvision Lehrkörper

Der Lehrkörper stellt das Lehrerpersonal an einer Schule dar. Der Lehrkörper enthält eine Menge von Lehrern.

- Verwaltung des Lehrerbestandes

  Der Bestand an Lehrer kann sich ändern. Der Lehrkörper ist ein Behälter, der Lehrer verwaltet. Er kann neue Lehrer aufnehmen und es können Lehrer aus ihm entfernt werden. Er kann Lehrer herausgeben und nach ihnen suchen.

7.6 Materialvision Pausenliste

An einer Schule gibt es neben einer Menge von Unterrichtsstunden eine Menge von Pausen. Diese Pausen liegen zwischen den Unterrichtsstunden. Die Pausenliste nimmt alle Pausen der Schule auf.

- Verwaltung der Pausen

  Jede Schule hat andere Pausen. Es kann eine unterschiedliche Anzahl von Pausen geben. Die Pausenliste wird verändert. Es können Pausen hinzugefügt sowie entnommen werden. Die Pausenliste kann Pausen herausgeben und nach ihnen suchen.

7.7 Materialvision Pausenplan

An einer Schule gibt es eine Menge von Pausen und eine Menge von Lehrern, sowie eine Menge von Aufsichtsorten. Eine Pausenaufsicht kommt zustande, indem ein Aufsichtsort eines Elementes der Pausenliste mit einem Element des Lehrkörpers belegt wird. Der Pausenplan enthält eine Pausenliste. Er kann mit einem Lehrkörper verknüpft werden. Sobald es eine solche Verknüpfung gibt, ist es möglich Pausenaufsichten einzuteilen. Der Pausenplan benutzt die Materialien Lehrkör-

per und Pausenliste. Es handelt sich somit um einen fachlichen Behälter, der den Lehrkörper benutzt und die Pausenliste verändert.

- Durchführung von Pausenbelegungen

  Der Pausenplan führt die notwendigen Operationen durch um Pausenaufsichten zu erstellen. Dafür benötigt er ein hohes Maß an fachlichem Wissen. Er muß die Pausenbelegung durchführen, Konfliktsituationen feststellen sowie erkennen, ob ein Pausenplan vollständig belegt ist.

- Konflikterkennung

  Der Pausenplanersteller möchte wissen, wo und wie viele Konflikte bei der Planung der Pausenaufsichten aufgetreten sind. Der Pausenplan kann diese Konflikte für alle Lehrer, oder auch nur für einzelne Lehrer, sowie für Pausen ermitteln.

- Ermittlung statistischer Informationen

  Für jeden Lehrer resultiert aus seiner Stelle sowie der Gesamtanzahl der Pausen eine Anzahl von Aufsichtspflichten. Die Ermittlung der Aufsichtspflichten, sowie der Pausenaufsichten des Lehrers gehört zu den Verantwortlichkeiten des Pausenplanes.

7.8 Materialvision Wöchentlich wiederkehrender Zeitraum

Ein wöchentlicher wiederkehrender Zeitraum bezeichnet einen Zeitspanne zwischen zwei definierten Zeitpunkten, die sich jede Woche wiederholt. Er kann als Vergegenständlichung der Aussage : „Immer Montags von 17°° bis 18°°Uhr“ verstanden werden. Ein wöchentlicher Zeitraum kann sich über mehrere Tage erstrecken.

- Veränderung des Zeitraumes

  Ein wöchentlich wiederkehrender Zeitraum kann verändert werden.

- Feststellung von Überschneidungen

  Ein wöchentlich wiederkehrender Zeitraum kann feststellen, ob er sich mit einem anderen überschneidet.

Anhang C Glossar des Pausenplanersystems

Aufsichtsort Ein Ort an einer Schule, an dem sich in den *Pausen* Schüler aufhalten und dort demzufolge Aufsicht geführt werden muß.

Aufsichtsortebearbeiter Eine Werkzeug, mit dessen Hilfe die Benutzer des *Pausenplanersystems* die *Aufsichtsorte* an einer Schule definieren.

Ausschlußzeitraum Ein *Ausschlußzeitraum* ist eine Zeitspanne, in welcher ein *Lehrer* nicht zur *Pausenaufsicht* eingeteilt werden sollte.

Benutzer Der Benutzer ist die Person, die mit dem späteren System arbeiten wird. Innerhalb des *Pausenplanersystems* gibt es mehrere Arten von Benutzern. Sie verkörpern dabei unterschiedliche funktionelle Rollen. Der *Pausenplanersteller* oder *Sekretariatsmitarbeiter* sind Benutzer.

Lehrer Der *Lehrer* an einer Schule. Er ist im Kontext des *Pausenplanersystems* eine Person, die *Pausenaufsichten* führen muß, also ein *Pausenbeaufsichtiger*.

Lehrerkarte Eine Karte, auf der die für die Pausenplanung relevanten Informationen notiert sind. Diese Informationen sind im einzelnen der Name des *Lehrers*, die Stelle, die Anzahl der Aufsichtspflichten und die Ausschlußzeiträume des *Lehrers*. Die Anzahl der *Lehrerkarten* zu einem *Lehrer* entspricht der Anzahl der Aufsichtspflichten, denen er im Rahmen seiner Stelle nachzukommen verpflichtet ist. (siehe dazu Szenario Pausenaufsichtsplan erstellen)

Lehrkörper Der *Lehrkörper* ist anwendungsfachlich die Menge der *Lehrer*, die an einer Schule beschäftigt sind. Systemtechnisch bezeichnet der *Lehrkörper* einen Behälter, der die *Lehrer* der Schule enthält und verwaltet.

Lehrkörperbearbeiter Ein Werkzeug, welches benutzt wird, um einen *Lehrkörper* zu pflegen.

Notiz Eine *Notiz* ist ein Kommunikationsmittel. Sie dient der Mitteilung einer Information an einen oder mehrere Personen. Sie kann konkret an eine Person gerichtet sein, oder ungerichtet sein. Ungerichtet bedeutet, daß sie alle Personen betrifft, die zufällig oder bewußt mit dieser *Notiz* konfrontiert werden. Die Adresse ist üblicherweise auf einer *Notiz* vermerkt. Außerdem enthält sie das Datum der Erstellung, sowie ihren Autor.

Pause Meyers Taschenlexikon beschreibt die *Pause* als „Unterbrechung einer (körperlichen oder geistigen) Tätigkeit, um Ermüdung und damit Leistungsabfall zu vermeiden“. Im Kontext des *Pausenplanersystems* wird der Begriff der *Pause* jedoch in einer spezialisierteren Form verwendet. Er bezeichnet konkreter die *Pausen*, die es an einer Schule gibt und Unterrichtsstunden unterbrechen.

Pausenaufsicht Die *Pausenaufsicht* bezeichnet eine Menge von *Pausenbeaufsichtigern*, mit denen eine *Pause* belegt ist, um sie zu beaufsichtigen Eine *Pausenaufsicht* bezieht sich immer auf eine *Pause*. Der Begriff meint einerseits die Tätigkeit eine *Pause* zu beaufsichtigen (eine Person führt die *Pausenaufsicht*). Andererseits meint der Begriff ebenso die funktionelle Rolle und die damit verbundenen Verantwortlichkeiten (eine Person ist die *Pausenaufsicht*).

Pausenaufsichtsort Ein Ort einer zu beaufsichtigenden *Pause*. Dieser eher technisch definierte Begriff orientiert sich einerseits an dem Ort, an dem Aufsicht zu führen ist, andererseits aber auch an dem Zeitpunkt einer *Pause* und definiert somit beide Eigenschaften.

Pausenaufsichtsplan siehe *Pausenplan*

Pausenbeaufsichtiger Eine funktionelle Rolle an einer Schule. Die Person, die mit der Beaufsichtigung einer *Pause* an einer Schule betraut ist. Es gibt *Pausenbeaufsichtiger* mit unterschiedlicher Qualifikation (siehe dazu das Szenario Pausenaufsichtsplan erstellen). Es gibt *Lehrer*, die verantwortlich *Pausen* beaufsichtigen dürfen, andere *Lehrer*, die dieses nicht verantwortlich tun dürfen und Zivildienstleistende, die ebenso der Qualifikation eines verantwortlichen *Pausenbeaufsichtigers* nicht entsprechen.

Pausenlistenbearbeiter Werkzeug um die *Pausen* des *Pausenplans* zu pflegen

Pausenplan Der *Pausenplan* ist die Verknüpfung eines *Lehrkörpers* mit einer Menge von *Pausen*. Gemeint ist damit sowohl der anwendungsfachlich vorliegenden Papierplan, auf dem der *Pausenplanersteller Lehrerkarten* ablegt (siehe Szenario Pausenausichtsplan erstellen), als auch das systemtechnische Pendant in Form des Materials (siehe Materialvision Pausenplan).

Pausenplaner	Softwarewerkzeug um *Pausenaufsichten* einzuteilen Es unterstützt den *Pausenplanersteller* bei der Erfüllung seiner Aufgabe, *Pausen* mit *Lehrern* zu belegen.
Pausenplanersystem	Das zu erstellende Gesamtsystem.
Pausenplanersteller	Eine funktionelle Rolle für die Person, die an einer Schule die Aufgabe hat, die Pausenplanung vorzunehmen. In der Regel wird ein *Lehrer* mit dieser Aufgabe betraut.
Sekretariatsmitarbeiter	Eine Person, die im Sekretariat der Schule beschäftigt ist. Sie ist auch ein Benutzer des Pausenplanersystems, da sie mit der Pflege des *Lehrkörpers* betraut ist.