Techniken zur Konstruktion verteilter und technisch eingebetteter Anwendungssysteme

Wolf-Gideon Bleek

Diplomarbeit

Universität Hamburg

Fachbereich Informatik

Arbeitsbereich Softwaretechnik

August 1997

Erstbetreuer:
Prof. Dr. Heinz Züllighoven

Zweitbetreuer:

Prof. Dr. Horst Oberquelle

Diese Diplomarbeit wurde am Fachbereich Informatik der Universität Hamburg zur teilweisen Erfüllung der Anforderungen zum Erlangen des Titels Diplom-Informatiker eingereicht.

Erklärung:
Ich versichere hiermit, diese Arbeit selbständig und unter ausschließlicher Zuhilfenahme der in der Arbeit aufgeführten Hilfsmittel erstellt zu haben.

Großhansdorf, den 8. August 1997

Wolf-Gideon Bleek
Erlenring 19
22927 Großhansdorf
Tel. (04102) 697130
Fax (04102) 697183

Betreuung:
Prof. Dr. Heinz Züllighoven (Erstbetreuer)
Prof. Dr. Horst Oberquelle (Zweitbetreuer)

Prof. Dr. Heinz Züllighoven

Arbeitsbereich Softwaretechnik (SWT)
Fachbereich Informatik
Universität Hamburg
Vogt-Kölln-Straße 30
22175 Hamburg

Prof. Dr. Horst Oberquelle

Arbeitsbereich Angewandte und sozialorientierte Informatik (ASI)
Fachbereich Informatik
Universität Hamburg
Vogt-Kölln-Straße 30
22175 Hamburg

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Überblick über die Arbeit

Die Unterteilung von komplexen Anwendungssystemen in Komponenten[1], die zum Zeitpunkt der Benutzung zusammengestellt werden können, ist ein spannendes und aktuelles Entwicklungsgebiet im Bereich der softwaretechnischen Gestaltung von Anwendungssystemen. Die hier präsentierte Arbeit stellt dazu die Ergebnisse eines Projektes dar, in dem gerade dieser Ansatz entwickelt wurde. Es wird die Konzeption und Umsetzung eines modularen, verteilten Anwendungssystems im Mehrprozeßraum vorgestellt, wobei zur Realisierung der Aufteilung in einzelne Komponenten (Prozesse) ein Koordinationsmittel, der Umgebungsprozeß, eingeführt wird. In diesem Kapitel wird die Motivation für diese Arbeit dargelegt und ein Überblick über die einzelnen Kapitel gegeben.

1.1 Motivation und Zielsetzung

Ein Defizit der objektorientierten Anwendungsentwicklung ist die Granularität der Gliederungseinheiten. Auf der einen Seite sind Objekte die Einheit der Gliederung. Sie stellen eine zu feine Granularität dar, wenn es darum geht Werkzeuge zu beschreiben. Auf der anderen Seite stehen Anwendungssysteme, die als Einheit zu groß sind, um ein Software-System zu gliedern. Es war deshalb notwendig, nach einer Gliederungseinheit zu suchen, die eine Aufteilung von Anwendungssystemen in geeignet große Komponenten ermöglicht.

Ziel der mit dieser Arbeit verbundenen Forschung war es deshalb, Software-Systeme in der Art in Komponenten aufzuteilen, daß sich diese als eine praktikable Einheit oberhalb der Objektebene und unterhalb des Anwendungssystems darstellen. Dabei soll ebenso eine Wiederverwendung der Komponenten ermöglicht werden, wie es bereits auf der Objektebene praktiziert wird. Werkzeuge sollen in unterschiedlichen Anwendungssystemen verwendet werden können.

Diese Diplomarbeit ist im Rahmen einer Kooperation des Softwaretechnik Centers des Arbeitsbereichs Softwaretechnik der Universität Hamburg, Fachbereich Informatik, mit der Firma Micrologica, Bargteheide, entstanden. Dabei entwickelt die Firma Micrologica Softwarewerkzeuge zur Unterstützung von Telefoniearbeitsplätzen mit Hilfe von

[1] Vgl. Komponenten basierte Software (Entwicklung)

objektorientierten Programmiersprachen. Das Softwaretechnik Center hat bei der Einführung von objektorientierten Entwicklungsmethoden in ausgewählten Projekten mitgewirkt. Im Anschluß an die Einführung der Software-Entwicklungsmethode WAM wurde das Projekt „Online Configuration“ des Micrologica Communication Center gestartet, welches die Grundlage für diese Diplomarbeit bereitet hat.

Die Software-Entwicklungsmethode WAM ist eine Methode zur Entwicklung von Anwendungssystemen für Einzelarbeitplätze. Die Umsetzung von WAM in das Einsatzgebiet der Computer unterstützten Telefonie stellt eine besondere Herausforderung dar, weil man es hierbei mit einem verteilten Anwendungssystem zu tun hat, bei dem - gesteuert durch eine oder mehrere Server-Komponenten - eine Reihe von Klientenapplikationen mit Arbeitsaufträgen versorgt werden. Insbesondere die Anforderung, modular weitere Werkzeuge in den Kontext dieses Anwendungssystems zu integrieren, macht es notwendig, über eine allgemeine Organisation der Werkzeuge nachzudenken. Dies führt einerseits zu einer Standardisierung der internen Anbindung der Werkzeuge, kann aber andererseits ebensogut zu Standardwerkzeugen oder allgemeinen Komponenten des Anwendungssystems leiten.

1.2 Ausgangspunkte und Aufbau der Arbeit

Den Ausgangspunkt für diese Arbeit bildet ein bestehendes Anwendungssystem, das wie oben angedeutet aus einer Reihe von Servern besteht, die Telefonanrufe auf Arbeitsplätze verteilen. Dabei war es bereits möglich, das System durch die freie Wahl der Anzahl der Server zu skalieren. Tätigkeiten konnten entsprechend den Anforderungen über die gewählte Server-Anzahl verteilt werden. Dabei erfolgte die Konfiguration der Server durch sogenannte INI-Files, Dateien in denen gruppiert durch textuelle Beschreibungen die Einstellungen des Servers abgelegt wurden. Die Konfigurationsdateien wurden beim Starten des Servers eingelesen, überprüft und als Grundlage für die Konfiguration des Systems herangezogen. Sie beschreiben die Bestandteile des Servers, die zum Startzeitpunkt erstellt werden.

Auf dieselbe Weise wurden auch die Werkzeuge der Klientenarbeitsplätze konfiguriert. In einer zentralen Konfigurationsdatei waren alle Einstellungen der Klientenarbeitsplätze abgelegt. Durch gemeinsamen Dateizugriff wurde diese allen Arbeitsstationen zur Verfügung gestellt. Ein erstes Auslesen ermöglichte es den Werkzeugen, den Anwendern eine Auswahl zu präsentieren, nach der sich der jeweilige Benutzer anmelden konnte. Anhand dieser Anmeldung wurden weitere Einträge ausgelesen, die spezielle Konfiguration der Werkzeuge für diesen Anwender beinhalten.

Es wird deutlich, daß für die gesamte Konfiguration eines vollständigen Systems eine große Anzahl von Konfigurationsdateien notwendig ist, wobei diese in einem engen Zusammenhang zueinander stehen, denn Werkzeuge der Arbeitsstationen können nur auf Objekte des Servers bezug nehmen, die in diesem eingerichtet wurden.

Zur Vereinfachung der Erstellung der komplexen Konfigurationsdateien wurde von Mitarbeitern der Kundenbetreuung ein Werkzeug entwickelt, das aufgrund der über den Kundenbetrieb gesammelten technischen und anwendungsspezifischen Daten und den frei wählbaren Verknüpfungen im System, eine relationale Datenbank verwaltet. Aus dieser Datenbank konnten auf Knopfdruck entsprechende Konfigurationsdateien erstellt werden.

Bei der Änderung der Einstellungen des Systems durch den Systemadministrator, Supervisor, wurde es notwendig, alle betroffenen Komponenten (im Zweifelsfall auch alle nichtbetroffenen) neu zu starten. Das bedeutet, daß bei Änderungen an der Klientenkonfiguration zumindest die Arbeitsstationen ihre Anwendungen beenden und neu starten mußten. Bei Änderungen an der Server-Konfiguration war es notwendig, den Server herunterzufahren und wieder hochzufahren, damit die neuen Konfigurationsdateien eingelesen werden konnten.

Diese Situation ist aus vielerlei Aspekten nicht akzeptabel. Einer der wichtigsten Gründe ist der Umstand, daß während des Betriebs des Systems keine Änderungen durchgeführt werden konnten, ohne daß das System für eine gewisse Zeit nicht verfügbar war. Das verhinderte in vielen Fällen die Durchführung von Änderungen an der Konfiguration zur Arbeitszeit und verschob diesen Vorgang auf arbeitsfreie Zeiten in den Abend- oder Nachtstunden. Damit war es für den Supervisor unmöglich, auf kurzfristige Ereignisse während des Betriebs einzugehen. Anforderungen mußten lange geplant werden und wurden dann in konzertierten Aktionen gleichzeitig durchgeführt. Eine Prüfung der Änderungen konnte dann nur im laufenden Betrieb durchgeführt werden.

Ausgehend von diesem Entwicklungsstand wurde beschlossen, eine Möglichkeit zu schaffen, das Anwendungssystem während seiner Laufzeit zu konfigurieren. Dieses Vorhaben bildet den Ausgangspunkt der Diplomarbeit.

Im zweiten Kapitel werde ich zuerst die Software-Entwicklungsmethode WAM einführen und auf ihre charakteristischen Merkmale eingehen. Das Kapitel wird aufzeigen, welche Bestandteile das Repertoire bietet und gleichzeitig deutlich machen, welche zusätzlichen technischen Konstruktionen notwendig sein werden.

Das dritte Kapitel bietet einen Überblick über das Anwendungsgebiet Call-Center und wird in die Begriffswelt der Anwendungsdomäne einführen. Anhand von Szenarios werden die typischen Arbeitsabläufe verdeutlicht.

Im vierten Kapitel wird das konkret realisierte Anwendungssystem vorgestellt. Die Werkzeuge des Anwendungssystems und ihr Einsatz wird besprochen. Das Zusammenwirken der allgemeinen Werkzeuge mit denen, die für spezielle Anwendungsfälle erstellten wurden, wird ausgeführt.

Das fünfte Kapitel liefert einen Überblick über den Stand der Kunst bei der technischen Entwicklung der Anwendungssysteme und zeigt die derzeit eingesetzten Trennungs- und Kopplungsmechanismen.

Das sechste Kapitel stellt die Konzeptionsmuster technischer Automat, Sonde und Einstellwerkzeug, die im Kontext des zitierten Anwendungssystems entwickelt wurden, vor.

Kapitel sieben zeigt die im Projekt entwickelte Genese der Software-Entwicklung vom monolithischen zum komponentenbasierten Anwendungssystem. Dazu werden Anwendungssysteme in mehrere Prozesse aufgeteilt. Vormals im Einzelprozeßraum entwickelte Anwendungssysteme können mit Hilfe dieser Konzeption in den Mehrprozeßraum überführt werden.

Der Umgebungsprozeß, eine separate Implementation einer den Arbeitsplatz koordinierenden Umgebung, ist Thema des achten Kapitels. Hier werden seine

Einordnung, die ihm zugeordneten Aufgaben und die Anbindung der Werkzeuge dargestellt.

Das neunte Kapitel versucht die in dieser Arbeit erlangten Ergebnisse noch einmal zusammenzufassen und deren Auswirkungen abzuschätzen. Es wird der Versuch unternommen, einen Ausblick in die zukünftige Entwicklung zu wagen.

1.3 Konventionen dieser Arbeit

Für ein leichteres Verständnis beim Lesen, wurde bei dieser Arbeit versucht, auf bereits etablierte Notationstechniken zurückzugreifen. Wie aus der Kognitionswissenschaft bekannt ist, ist es besonders leicht, in vertrauten graphischen Notationstechniken zu lesen und natürlich im Gegensatz dazu besonders erschwerend, wenn man gezwungen ist, in neuen - noch nicht bekannten - Notationen zu arbeiten.

Es wurde deshalb vermieden, gänzlich neue Darstellungsweisen einzuführen. Allerdings, und dabei ist es durchaus legitim und im Rahmen der gesetzten Regeln, war es an einigen Stellen notwendig, die Darstellungsformen um neu gewählte Elemente zu erweitern.

Eine textuelle Konvention ist an dieser Stelle besonders bemerkenswert. Alle Wortbildungen, die durch Worte der englischen und deutschen Sprache entstehen, werden durch einen Bindestrich miteinander verknüpft. Es wird nicht als angebracht angesehen, Worte der englischen Sprache und der deutschen Sprache direkt miteinander zu verknüpfen.

1.3.1 OMT

Die Object Modeling Technique (OMT) hat sich mittlerweile als ein Standard für die Darstellung von Klassen und deren Beziehungen durchgesetzt. Obwohl im Rahmen der Forschung viele andere ähnlich gute Darstellungsformen entwickelt wurden, ist es mit Berücksichtigung des oben gesagten wichtig, daß nur eine Darstellungsform in der wissenschaftlichen Literatur Verwendung findet. Insbesondere um die Lesbarkeit für einen großen Kreis zu erleichtern.

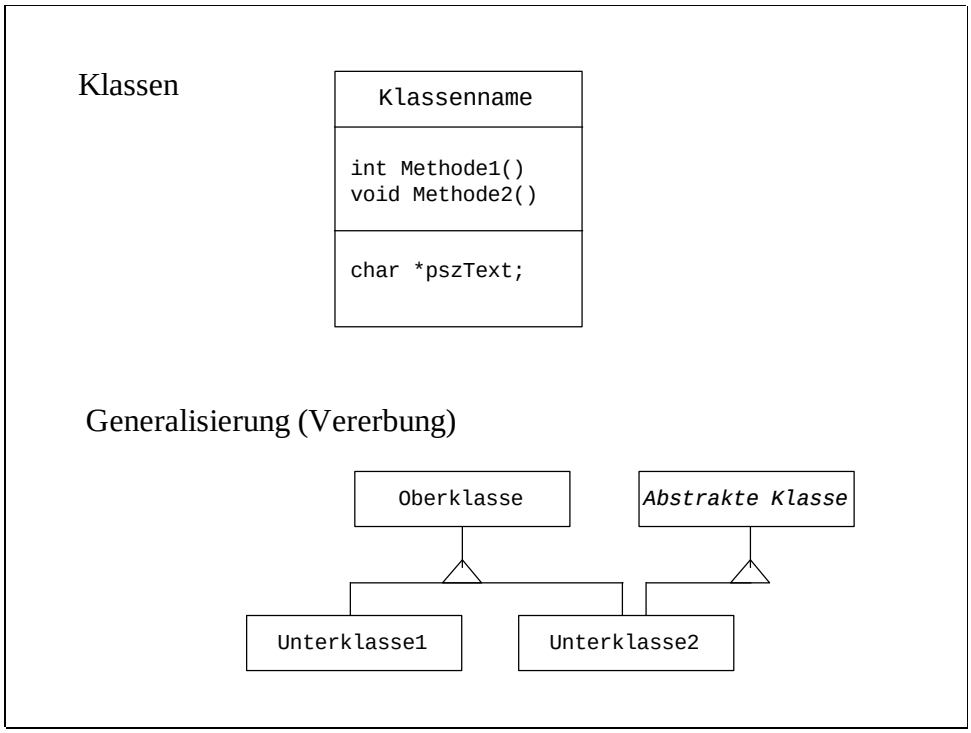

Abbildung 1 OMT Darstellung für Klassen und Vererbung

Klassen werden durch Rechtecke dargestellt, in denen der Klassenname in Courier Schrift angegeben wird. Werden zusätzlich zum Klassennamen auch noch einzelne Methoden der Klasse abgedruckt, dann werden diese, durch eine waagerechte Linie getrennt, unterhalb des Klassennamens aufgeführt. Sind Methoden bzw. Operationen nur abstrakt spezifiziert, wird der Methodenname in kursiver Schrift gesetzt. Attribute einer Klasse sind, wenn nichts anderes gesagt wird, dem privaten Bereich einer Klassendefinition anzurechnen und werden durch eine weitere waagerechte Linie von den Methoden abgegrenzt. Es kann davon ausgegangen werden, daß bei unseren Konstruktionen das Lesen und Verändern von Attributen ausschließlich durch bereitgestellte Methoden ermöglicht wird.

Um die Vererbungsbeziehung zwischen Klassen darzustellen, wird die unter „Generalisierung“ in der Abbildung angegebene Schreibweise verwendet. Klassen, die über eine Linie miteinander verbunden sind, bei der sich ein Dreieck in der Mitte befindet, stehen in einer Vererbungsbeziehung. Die oberhalb angeordnete Klasse ist die Oberklasse zur unterhalb dargestellten. Die Spitze des Dreiecks zeigt immer zur Oberklasse. Die untere Klasse erbt alle Methoden und Attribute der oberen. Handelt es sich bei der Oberklasse um eine abstrakte Klasse, es kann also nie ein Objekt der Klassse angelegt werden, dann wird dies dargestellt, indem der Klassenname kursiv gesetzt wird.

In Ausnutzung der OMT Darstellungsmethode werden hier lediglich die Symbole für Klassen und Objekte und deren Beziehungen verwendet. Auf die Verwendung der State-Transition Diagrams wird nicht zurückgegriffen, da es in diesem Anwendungsfeld nicht von bedeutung war. Insbesondere ist auf den signifikanten Unterschied zwischen den von der OMT Methode angebotenen Darstellungselementen zur Erstellung von Objekt-Interaction Diagrams hinzuweisen. Hier werden ebenfalls Diagramme mit dieser Bezeichnung verwendet, allerdings mit Berufung auf Jacobson (s.u.).

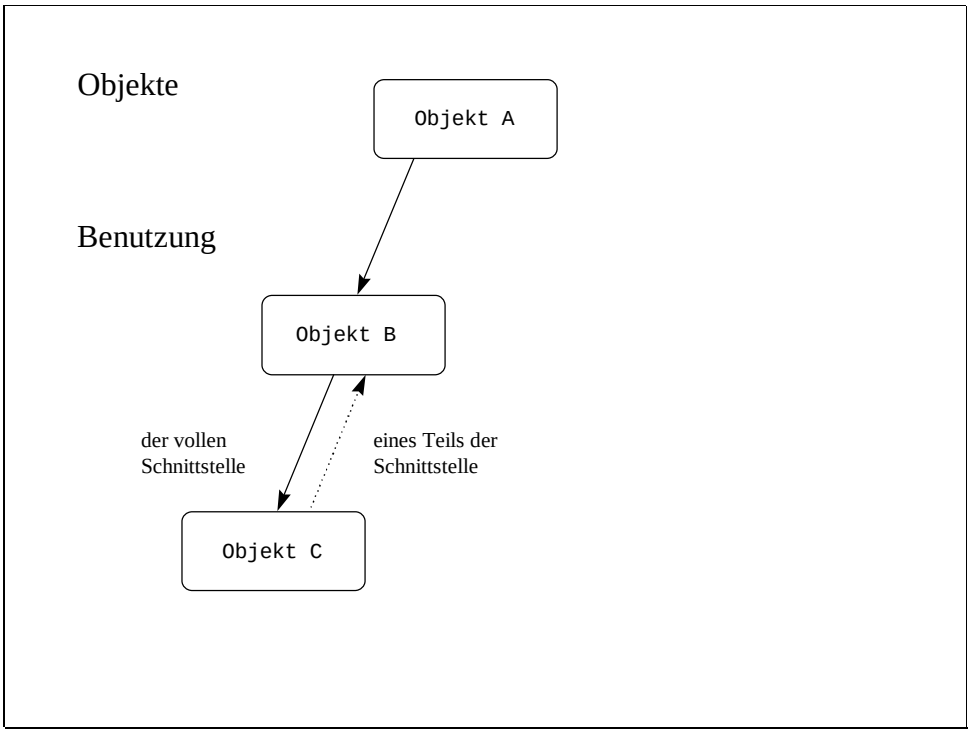

Abbildung 2 OMT Darstellung für Objekte und Benutzung

Die in Abbildung 2 dargestellte Grafik, zeigt die Beziehung von Objekten auf. Objekte werden durch Kästen mit runden Ecken gezeichnet. Pfeile drücken eine Benutzung aus. Das Objekt, von dem der Pfeil ausgeht, erzeugt oder benutzt das Objekt, auf das der Pfeil zeigt. Wird anstelle einer durchgezogenen Linie eine gestrichelte verwendet, handelt es sich um eine Benutztbeziehung unter Ausnutzung nur einer Teilschnittstelle.

1.3.2 Interaktionsdiagramme

Jacobson (vgl. [Jac92], Seite 147) verwendet sogenannte „Interaction Diagrams“ zur Darstellung der Kommunikation von Blöcken:

> *To describe a sequence of stimuli, we use interaction diagrams. In these, we can describe how several blocks communicate by sending stimuli to each other.*

Diese Darstellungsmethode wird verwendet, um die Interaktion von konkreten Objekten zu verdeutlichen. Dabei wird jedes Objekt in seiner Lebenszeit durch eine senkrechte Linie angedeutet. Kästen auf der Linie eines Objektes zeigen an, daß der Kontrollfluß in dieses Objekt gewandert ist. Pfeile von einem Kasten zu einem anderen Objekt geben den Wechsel der Kontrolle in ein anderes Objekt an. Durch einen gestrichelten Pfeil wird die Rückkehr der Kontrolle signalisiert.

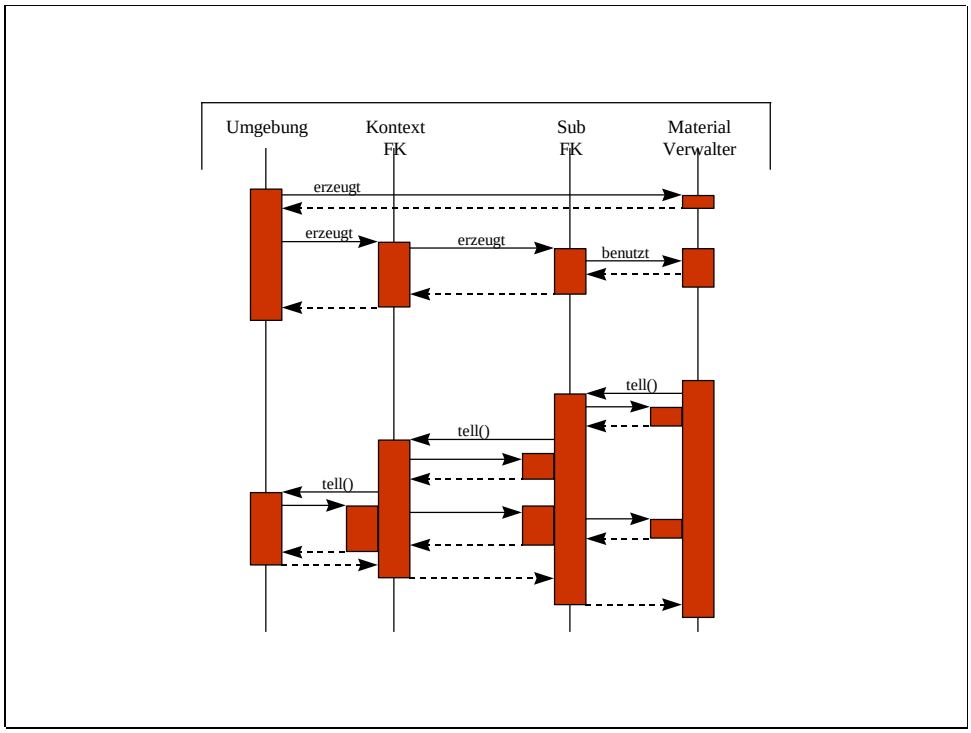

Abbildung 3 Beispiel für ein Interaktionsdiagramm

Das hier gezeigte Beispiel verwendet vier Objekte in zwei getrennten Zusammenhängen. Im oberen Bereich ist die Erzeugung von Objekten dargestellt. Im unteren Bereich sieht man wie die indirekte Benachrichtigung von Objekten immer wieder zu „Rückfragen“ führt, die durch einen zweiten Balken symbolisiert sind.

Später wird gezeigt werden, daß diese Darstellungsmethode nicht nur zur graphischen Darstellung des Kontrollflusses in Einzelprozeßsystemen geeignet ist, sondern sich ebenso für den Mehrprozeßraum anbietet.

1.3.3 Rollendiagramme

Rollendiagramme helfen dem Leser beim Verständnis vom komplexen Beziehungen zwischen Objekten. Die von Reenskoog (vgl. [Ree95], Seite 50) eingeführte Darstellungsmethode unterstützt die Betrachtungsweise von Objektbeziehungen aus unterschiedlichen Blickwinkeln.

Das hier übernommene Beispiel macht deutlich, wie Objekte in einer bestimmten Beziehung zueinander stehen. Die Firma (Company) bezieht vom Lieferanten (Supplier) eine Ware und weist zur Bezahlung die eigene Bank (Payer Bank) an, der Lieferantenbank (Payee Bank) das Geld zu überweisen. Dieses Zusammenspiel wird mit Hilfe der Rollen (Company, Supplier, Payer Bank, Payee Bank) und deren Verknüpfungen dargestellt.

Wichtig herauszuheben an dieser Darstellung ist aber, im Gegensatz zu den Objekt Interaktionsdiagrammen, daß hierbei (funktionelle) Rollen verwendet werden. Diese Darstellungsform gibt z.B. keine Auskunft darüber, welche konkreten Banken in das Geschäft involviert sind. Insbesondere hält diese Darstellungsweise die Möglichkeit offen, daß es sich beim Kreditinstitut der beiden Unternehmen um ein und dasselbe

handeln kann. In einem später realisierten konkreten Software-System kann es während der Laufzeit also durchaus dazu kommen, daß dasselbe Bank-Objekt in einer Transaktion eines Geldbetrages in Form von zwei Rollen beteiligt ist.

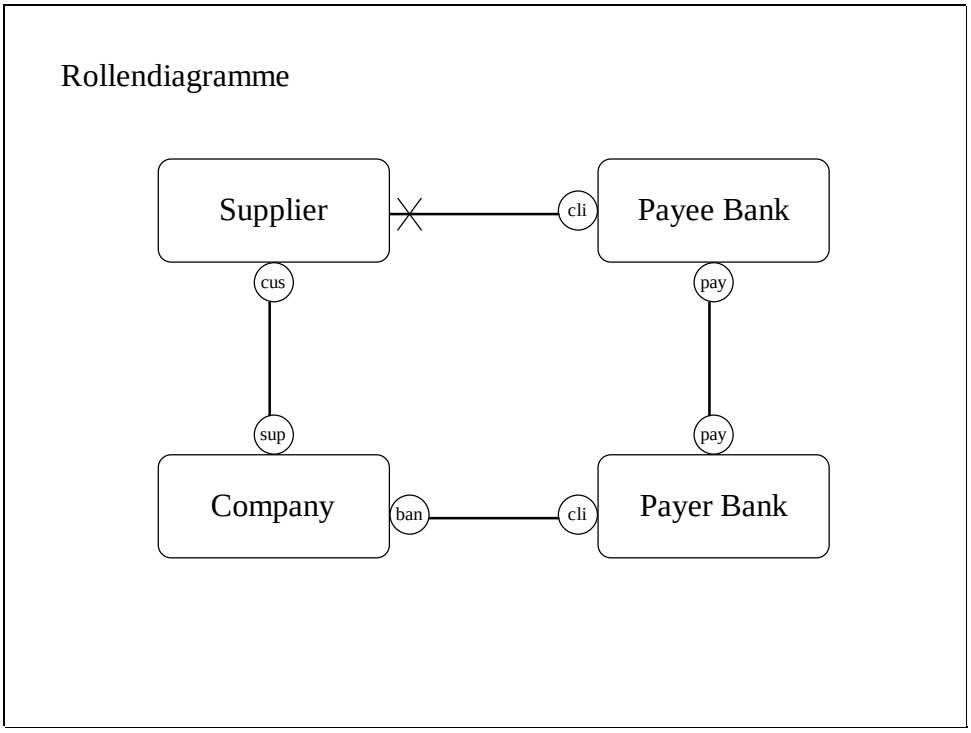

Abbildung 4 Rollendiagramme nach Reenskoog

Ebenso ermöglicht die Rollendarstellung die Zusammenführung von Rollen verschiedener Handlungssituationen. In dieser Graphik wird nur eine Handlung mit ihren Beteiligten aufgeführt. In einer anderen Handlung tritt z.B. der Akteur, der hier die Firma (Company) verkörpert, als ein Lieferant für eine andere Firma auf. In einer weiteren Handlung wäre die Firma der Koordinator eines Projektes. Diese vier Rollen: Kunde, Lieferant und Koordinator werden später in einem agierenden Objekt zusammengefaßt, welches in unterschiedlichen Aktionen in einer der einzelnen Rollen auftreten wird.

Diese Form der Darstellung wird verwendet, um das Design des Systems bezogen auf die unterschiedlichen Handlungen durchschaubarer zu machen und gleichzeitig, um die Argumentationsgrundlage zu besitzen, welche Objekte in welchem Zusammenhang welche Funktionen übernehmen werden.

2 Die Software-Entwicklungsmethode WAM

Der Arbeitsbereich Softwaretechnik des Fachbereichs Informatik der Universität Hamburg setzt zur Entwicklung von Anwendungssystemen die Software-Entwicklungsmethode WAM ein. Das Akronym[2] WAM steht dabei für Werkzeug, Automat und Material, die drei wesentlichen Entwurfsmetaphern der Methode. Die Entwicklungsmethode WAM ist einerseits eine objektorientierte Methode, indem sie Objekte der realen Welt mit softwaretechnischen Objekten repräsentiert. Werkzeuge, Materialien und Automaten werden in der realen Welt identifiziert und in geeigneter Weise durch Klassen abgebildet, die zur Laufzeit in Form von konkreten Objekten existieren. WAM ist andererseits auch eine evolutionäre Methode, wenn in wiederholten Rückkopplungen z.B. mit Hilfe von Dokumenten ein gefestigtes gemeinsames Verständnis über die Anwendungssituation der späteren Benutzer gebildet wird.

In diesem Kapitel wird die Software-Entwicklungsmethode WAM motiviert und das von ihr zugrunde gelegte Leitbild eingeführt. Die wesentlichen Metaphern des WAM-Repertoirs werden dem Leser mit ihrem Einsatz vorgestellt. Eine zentrale Rolle in diesem Kapitel nimmt die Diskussion der Metaphern Automat und Umgebung ein. Sie bilden die Grundlage für die später motivierte technische Umsetzung in den Mehrprozeßraum.

2.1 Die Software-Entwicklungsmethode WAM

Die Software-Entwicklungsmethode WAM wurde von Budde und Züllighoven begründet (vgl. [BZ90]) und seitdem intensiv weiterentwickelt (vgl. z.B. [KGZ94], [RZ95], [Rie95], [Gry96]). Ihr Fokus ist die Entwicklung von interaktiven Anwendungssystemen. Vor dem Hintergrund eines ausgewählten Menschenbildes wird mit Hilfe von Entwurfsmetaphern der Anwendungsbereich analysiert. Die Methode WAM eignet sich besonders für den Einsatz in Arbeitsbereichen, die dem

[2] Akronym das; ↑Initialwort
Initialwort das; Kurzwort, das aus zusammengerückten Anfangsbuchstaben gebildet ist (vgl. [Dud90], Seite 346)

Büroarbeitsplatz ähnlich sind. Außerdem wird als Grundannahme vorausgesetzt, daß es sich bei dem Arbeitsplatz um eine vom Arbeitenden selbstbestimmte Arbeit handelt.

WAM stellt dem Software-Entwickler einen methodischen Rahmen zur Verfügung, der auf vier grundlegenden Bereichen fußt. Die zentrale Position nimmt dabei das *Leitbild* ein. Wie oben bereits erwähnt, verkörpert es das der Entwicklung zugrundeliegende Menschenbild. Dafür wurde als Leitbild der „Arbeitsplatz für eigenverantwortliche Tätigkeit“ gewählt. Neben dem Leitbild stehen für WAM unterschiedliche *Vorgehensweisen* zur Verfügung. Die hier eingesetzte und vertretene Vorgehensweise ist das evolutionäre Prototyping (vgl. [Flo84]). Mit Hilfe der *Konstruktionstechnik* werden u.a. Dokumenttypen und deren Bearbeitung beschrieben. Es werden als Dokumente neben den Programmtexten und Klassendiagrammen Szenarios, Systemvisionen, Visionen und Glossare verwendet. Und schließlich stellt WAM einen Satz aufeinander abgestimmter Entwurfsmetaphern zur Verfügung. Derzeit verfügt man über die Metaphern Werkzeug, Material, Automat und Umgebung, auf die hier ebenfalls näher eingegangen wird.

2.2 Das Leitbild vom selbstbestimmten Arbeitsplatz

Unter einem Leitbild verstehen man nach Gryczan:

> *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. (vgl. [Gry96], Seite 99)*

Damit wird dem Software-Entwickler etwas an die Hand gegeben, worunter er stets seine Arbeit überprüfen und die Richtung seines Denkens korrigieren kann. Es wurde folgendes Leitbild gewählt:

Der gut ausgestattete Arbeitsplatz für qualifizierte menschliche Tätigkeit

Warum „gut ausgestattet“? Weil nur dann Arbeit effektiv unterstützen werden kann, wenn innerhalb der Arbeitssituation kein Mangel herrscht. Sollte die Arbeit durch Situationen unterbrochen werden, in denen es an Arbeitsmitteln oder Arbeitsgegenständen mangelt, dann wirkt sich dies negativ sowohl auf die Arbeitssituation als auch auf das Ergebnis der Arbeit aus.

Warum „qualifiziert“? Weil die Annahme zugrunde liegt, daß der Anwender seine Arbeitsschritte selbständig plant und verrichtet. Dies ist aber nur mit der nötigen Qualifikation möglich. Deshalb wird diese eingefordert.

Warum „menschliche Tätigkeit“? Unter einer menschlichen Tätigkeit versteht man diejenigen Aufgaben, die sinnvollerweise von Menschen verrichtet werden sollten. Das sind Tätigkeiten mit Verantwortung und insbesondere Tätigkeiten, die sich nicht automatisieren lassen.

Im Einsatzkontext des Call Centers, der später (Kapitel 3) noch genauer dargestellt wird, finden sich unterschiedlich ausgeprägte Arbeitsplätze. Insbesondere gibt es in diesem Einsatzfeld durchaus Arbeitsplätze, die nicht unter dem hier aufgestellten Leitbild betrachtet werden können, weil der Aspekt der Qualifikation vollständig verdrängt wird. Der Schwerpunkt soll deshalb auch nur auf jene Arbeitsplätze gelegt werden, die diesem Anspruch genügen. Dazu zählt auf jeden Fall der Arbeitsplatz des Systemadministrators.

Die Anwendung der Entwicklungsmethodik WAM läßt sich aber durchaus auch für das Gesamtsystem mit seinen unterschiedlich gearteten Arbeitsplätzen rechtfertigen. Denn bei der Entwicklung der Software-Werkzeuge ist teilweise noch nicht bekannt, in welchem Einsatzkontext diese verwendet werden sollen. Es können deshalb die Maßstäbe der Entwicklungsmethode angelegt werden, da ihre technische Umsetzung ein Werkzeug ermöglicht, das auf einem wie oben beschriebenen Arbeitsplatz eingesetzt werden kann.

Erst die Anforderungen des Kunden an einen Arbeitsplatz, wie z.B. den des Telefonsachbearbeiters konkretisieren das Einsatzfeld derart, daß bestimmt werden kann, ob es sich um eine weitestgehend eigenverantwortliche Arbeit handelt, oder die Arbeit mit weniger qualifizierter Tätigkeit gleichzusetzen ist.

Der Arbeitsplatz des Systemadministrators, der ausschlaggebend für die in dieser Arbeit diskutierten Fachthemen ist, stellt einen im hohen Grade qualifizierten Arbeitsplatz dar. Die dort ausgeübte Tätigkeit betrifft Entscheidungen von großer Tragweite und hat sofort Auswirkungen auf eine Reihe von Mitarbeitern und den weiteren Verlauf ihrer Arbeitsgestaltung.

Das Anwendungsgebiet, welches im Mittelpunkt dieser Arbeit steht, ist im Gegensatz zu den bisherigen Einsatzfeldern der WAM Methodik in einem Aspekt grundsätzlich verschieden. Werkzeuge in Bereich der Telefonie müssen in der Lage sein, Live-Informationen darzubieten und zu manipulieren. Das bedeutet, daß die Werkzeuge in einem engen Kontakt zu (Telefon-)Anlagen stehen und deren Aktionen für den Anwender nachvollziehbar machen.

Das Leitbild bietet einen Maßstab zur Einordnung von Arbeitsplätzen. Das nächste und das übernächste Kapitel wird in die unterschiedlichen Arbeitsplatzperspektiven eines Telefoniesystems einführen. Wenngleich sowohl für den Arbeitsplatz eines Telefonsachbearbeiters als auch für den des Systemadministrators die Methodik WAM bei der Software-Entwicklung zum Einsatz kommt, so ist doch ein deutlicher gradueller Unterschied in der Bewertung der Arbeitsplätze festzustellen.

Durch den Einfluß des Leitbildes entstehen Anwendungssysteme, bei denen der Anwender aus einer Palette von Werkzeugen wählen kann, die für die Aufgaben des Arbeitsplatzes entwickelt wurden. Gleichzeitig wird aufgrund der Qualifikation des Anwenders keine vordefinierte Reihenfolge in die Werkzeugbenutzung gelegt. Alle Werkzeuge können in dem Arbeitszusammenhang, der vom Anwender bestimmt wird, frei ausgewählt werden. Insbesondere stellen die Werkzeuge genau denjenigen Entscheidungsspielraum zur Verfügung, der zwischen automatisierbarem Anteil auf der einen Seite und den durch menschliche Entscheidungen geprägten Aufgaben auf der anderen Seite unterscheidet.

2.3 Die Entwurfsmetaphern des WAM Repertoires

Die Methode WAM stellt dem Entwickler eine Reihe von Metaphern für den Analyse- und Konstruktionsprozeß zur Verfügung. Das Fremdwörterbuch definiert den Begriff Metapher wie folgt:

> *[gr-lat] sprachlicher Ausdruck, bei dem ein Wort, eine Wortgruppe aus seinem eigentlichen Bedeutungszusammenhang in einen anderen übertragen wird, ohne daß ein direkter Vergleich zwischen Bezeichnendem und Bezeichnetem vorliegt; bildhafte Übertragung (z.B. das Haupt der Familie) (vgl. [Dud90], Seite 496)*

Die Metaphern sollen helfen, Dinge in der realen Welt zu identifizieren, um sie schließlich softwaretechnisch umsetzen zu können. Ein wesentlicher Vorteil dieser Metaphern liegt darin begründet, daß Menschen, die mit ihnen umgehen, bereits ein intuitives Verständnis davon haben, was sie darunter verstehen. Diese gemeinsame Sprache, bei der Beschreibung von Sachverhalten in der realen Welt, kann man sich beim Software-Entwicklungsprozeß zunutze machen. Als konkrete Metaphern stehen in WAM zur Verfügung:

- Werkzeug
- Automat
- Material
- Umgebung

Diese sollen nun kurz beschrieben werden:

2.3.1 Werkzeug, Material

Die Metaphern Werkzeug und Material hängen eng miteinander zusammen. Riehle schreibt zum Begriff Werkzeug:

> *Werkzeuge sind Arbeitsmittel, welche zur Sondierung und Manipulation der Arbeitsgegenstände, der Materialien, dienen. (vgl. [Rie95], Seite 7)*

Es werden beim Umgang mit Arbeitsgegenständen, den Dingen, die im Arbeitsprozeß kreiert bzw. verändert werden, zwei unterschiedliche Umgangsweisen unterschieden. Das *Sondieren* ist ein Vorgang, bei dem keine Eigenschaften eines Arbeitsgegenstandes geändert werden. Der *manipulative Umgang* hingegen, wie der Name schon sagt, verändert das Material nachweislich.

Wulf schreibt zum Begriff Material:

> *Materialien werden von Werkzeugen bearbeitet. Im Rahmen einer Anwendung werden sie damit zum Arbeitsgegenstand oder Bestandteil eines Arbeitsergebnisses. (vgl. [Wul95], Seite 14)*

Werkzeuge werden technisch über sogenannte Aspekte mit Materialien verbunden. Das bedeutet, daß Werkzeuge nicht konkrete Materialien kennen, sondern diese nur über (abstrakte) Aspekte ansprechen können. Materialien wiederum besitzen die Eigenschaften einer Reihe von Aspekten. Passen Werkzeug- und Material-Aspekte zusammen, kann das Werkzeug das konkrete Material sondieren oder manipulieren.

2.3.2 Automat

Der Automat ist die Metapher des WAM Repertoires, die immer dann angewendet wird, wenn sich wiederholende Aufgaben ohne Zutun des Anwenders ausgeführt werden müssen.

Gryczan definiert den Begriff des Automaten wie folgt:

> *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. (vgl. [Gry96], Seite 123)*

Als Ergebnis wird weiter festgestellt:

> *Automaten implementieren eine Routine, die eine Handlungsfolge auf ihren formalisierbaren Anteil reduziert und insgesamt erledigt. Einmal eingestellt und aktiviert, benötigen Automaten dazu keine interaktive Benutzeraktion. (vgl. [Gry96], Seite 128)*

Grundlegend für Automaten sind demnach zwei wesentliche Eigenschaften. Erstens werden sie nur für diejenigen Aufgabe verwendet, die in irgendeiner Art und Weise formalisierbar und gleichzeitig routinisierbar sind. Und zweitens, was als direkte Konsequenz daraus abzuleiten ist, haben Automaten selbst keine „interaktive Benutzeraktion“.

Aus den Projekten, an deren Fortschreibung ich beteiligt war, wurde als Erfahrung herausgezogen, daß alle dort konstruierten Automaten einer stetigen, wenn auch seltenen Einstellung bedurften. Zwar war der grundsätzliche Charakter eines Automaten im Gegensatz zu einem Werkzeug fest greifbar. Trotzdem hat es sich gezeigt, daß in gewissen Abständen, deren Länge deutlich vom Anwendungsfeld abhängig ist, diverse Einstellungen am Automaten notwendig waren.

Das daraus resultierende Problem stellte sich wie folgt dar: Einerseits sind Automaten ohne „interaktive Benutzeraktion“ ausgestattet. Das heißt, es gibt keine Interaktionskomponente, mit der der Zustand und die Einstellungen des Automaten visualisiert werden. Dies bedingt zweitens, daß auch keine Interaktionskomponente für die Änderung der o.g. Einstellungen vorgesehen ist. Drittens ist die räumliche Distanz zwischen dem den Automaten ausführenden Rechner und dem Benutzer, der die Einstellungen vornehmen will, üblicherweise so groß, daß keine „direkte“ Einflußnahme möglich ist.[3]

In diesem Text wird sich deshalb einer weiteren Metapher bedient, die ihm Anwendungsfeld des Automatenbegriffs durchaus eine Berechtigung hat. Im Rahmen von Wartungsarbeiten, die in regelmäßigen Abständen von qualifiziertem Personal oder bei Bedarf an den betriebenen Automaten durchgeführt werden, wird hierfür spezielles

[3] Diese Eigenschaft trifft nicht auf jedes Anwendungsfeld zu. Hier wird dem noch einzuführenden Gedanken vorgegriffen, daß Automaten im Sinne des Server-Begriffs eingesetzt werden.

Werkzeug verwendet, mit dem einzelne Parameter eines Automaten eingestellt werden können. Daraus wird der Begriff des *Einstellwerkzeugs* abgeleitet. Ein spezielles Softwarewerkzeug, das dazu benutzt wird, die Parameter eines Automaten zu visualisieren und dem Anwender die Möglichkeit zu geben, einzelne Parameter zu verändern.

Während also Werkzeuge im allgemeinen auf Materialien über Aspekte arbeiten, diese visualisieren und manipulieren können, werden Einstellwerkzeuge Automaten als ihr Material verwenden. Der wesentliche unterschied besteht hier aber in der Möglichkeit, daß der Zustand eines Automaten über die Zeit hinweg einer stetigen Änderung unterworfen ist, die nicht alleine durch das Werkzeuge bzw. lokale Anwendungssystem kontrolliert werden kann.

Über den technischen Aufbau von Einstellwerkzeugen wird in Kapitel 6 ausführlich berichtet werden.

2.3.3 Umgebung

Gryczan definiert die Metapher „Arbeitsumgebung“ folgendermaßen

> *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 Automaten festgelegt. (vgl. [Gry96], Seite 138)*

Aus dem Umstand heraus, daß verschiedene Werkzeuge sowie Automaten in einem Anwendungssystem zusammengefaßt werden müssen, ist die Umgebung die geeignete Metapher, um diese Zusammenfassung auch auf der Metaphernebene deutlich zu machen.

Diese Umgebungsmetapher läßt sich konform zu herkömmlichen Arbeitsplätzen einführen und unterstreicht die Abgrenzung zu anderen Arbeitsplätzen. Dies ist vergleichbar mit einem dem Arbeitsplatz zugeordneten Schreibtisch und macht die Örtlichkeit der Metapher deutlich. Außerdem bietet sich so die Möglichkeit der Unterscheidung verschiedener Arbeitsplätze, indem die Umgebung für jeden einzelnen Arbeitsplatz eine Arbeitsumgebung zur Verfügung stellt.

Ebenso wie der Raum, in dem die Arbeit verrichtet wird, keine Anforderungen an die Reihenfolge der durchzuführenden Arbeiten stellt, so ist ebenfalls die Umgebung in dieser Hinsicht keinesfalls restriktiv.

Die Entwurfsmetapher Umgebung stellt gleichzeitig den Ursprung eines Anwendungssystems dar. Sie beherbergt die Werkzeuge und Automaten und ist für deren Verwaltung, d.h. Erzeugung und Beenden (im Sinne des Schließens der Werkzeuge) verantwortlich. Sie ist ebenfalls die einzige Instanz, die eine Übersicht über alle verfügbaren und laufenden Komponenten des Anwendungssystems hat.

2.4 Zusammenfassung

Wenngleich der Grundgedanke der Entwicklungsmethode WAM den qualifizierten Arbeitsplatz als Leitbild voraussetzt, ist es durchaus möglich, qualitativ hochwertige Software mit Hilfe der Bestandteile dieser Methode zu bauen, die auch auf weniger qualifizierten Arbeitsplätzen eingesetzt wird. Insbesondere der flexible Einsatz der Werkzeuge auf unterschiedlich qualifizierten Arbeitsplätzen wird so ermöglicht.

Die Methode stellt ein Leitbild und Metaphern zur Verfügung, die dem Software-Entwickler bei der Systemgestaltung helfen. Dokumente unterstützen die schriftliche und kommunikative Weiterentwicklung des Gestaltungsprozesses.

3 Das Anwendungsgebiet Call Center

Der zunehmende Stellenwert der Telefontätigkeit innerhalb von Unternehmen hat zu einem eigenständigen Arbeitsgebiet und vielerorts auch zu einer eigenständigen Abteilung, dem Call Center, geführt. Unter einem Call Center wird z.B. eine Abteilung eines Unternehmens verstanden, die sich darauf spezialisiert hat, in telefonischen Kontakt mit dem Kunden zu treten. Sei es dabei über aktives Anrufen oder das Entgegennehmen von Telefongesprächen. Der konzeptionelle Schritt vom Telefondienst als eine Tätigkeit unter mehreren im Arbeitsalltag zum Telefonieren als Hauptaufgabe ist die Grundlage für den Call Center Gedanken. Ungeachtet dessen kann der Einsatz von Telefonie innerhalb eines Unternehmens auch eine technische Unterstützung erfahren, wenn die Ausprägung Call Center dabei keinen notwendigen Einsatz findet.

In diesem Kapitel werden die verschiedenen Arbeitsplätze im Zusammenhang mit einem Call Center betrachtet und damit die fachlichen Grundsteine für die Anforderungen an ein Softwaresystem gestellt, daß im Rahmen dieser Diplomarbeiter näher untersucht werden wird.

Das Kapitel untergliedert sich in fünf Unterabschnitte, die jeweils einen fachlichen Bereich des Call Centers zum Thema haben. Der Telefonarbeitsplatz wird im ersten Abschnitt als ein einführendes Beispiel vorgestellt. Dieser Arbeitsplatz stellt eine Kombination von Rechnerarbeitsplatz und klassischem Telefonarbeitsplatz dar. Der zweite Abschnitt führt dann in die Entwicklungsgeschichte des Call Centers ein und versucht die unterschiedlichen Ebenen des Begriffs Call Center zu beleuchten. Im dritten Abschnitt wird näher auf die in diesem Rahmen verwendete Technologie eingegangen und das Verfahren der Automatic Call Distribution (ACD) vorgestellt. Der vierte Abschnitt geht einen Schritt weiter und greift die Gedanken der ACD in der Art auf, daß daraus ein Metakonzept entwickelt wird: nämlich das der Jobvermittlung. Im fünften Abschnitt wird der Aspekt der Konfiguration erläutert und welche Qualitätsunterschiede sich dabei ergeben. Abschließend wird im sechsten Abschnitt auf die Erweiterung eines Call Centers eingegangen, die man benötigt, wenn mit verteilten Standorten gearbeitet werden soll und welche zusätzlichen Anforderungen dies aufwirft.

3.1 Beispiel Telefonarbeitsplatz

Ein Telefonarbeitsplatz ist eine besondere Form des Arbeitsplatzes, die neben besonderen Anforderungen an den Platz selbst auch zusätzliche Anforderungen an die technische Ausstattung stellt. Neben dem Telefon als zentrales Arbeitsmedium tritt zumeist ein Computer als Unterstützung für begleitende Arbeit. Diese speziellen Eigenschaften des Arbeitsplatzes werden jetzt näher untersucht.

3.1.1 Historie

Die Entstehung des Telefonarbeitsplatzes ließe sich sicherlich bis hin zur Einführung des Telefons zurückverfolgen. Dann im Zusammenhang mit von der Post angebotenen Vermittlungsdiensten[4], wo innerhalb der Ortsnetze noch kein Selbstwählldienst vorhanden war. Der Rahmen dieser Einführung wird aber auf den jüngeren Zeitraum begrenzt, in dem der Einsatz von Telefonie nicht nur für ihren Selbstzweck (Vermittlung von Gesprächen) benutzt wurde, sondern für andere wirtschaftliche Dienstleistungen.

Bereits im Stadium des gewerblichen Einsatzes lassen sich zwei charakteristische Formen von Arbeitsplätzen unterscheiden. Der erste Arbeitsplatz, abgeleitet vom Telefonistenarbeitsplatz, ist dafür vorgesehen, eingehenden Anrufe (z.B. von Kunden) entgegenzunehmen. Diese Form der Arbeitsplätze wird vielfältig eingesetzt. Beispiele für dessen Anwendung finden sich in Bestellannahmen, bei Produktauskünften und bei Aufforderungen zur Meinungsäußerung. Die zweite Form des Telefoneinsatzes könnte man als eher aktive Ausprägung von seiten des Unternehmens charakterisieren. Hierbei wird die dienstleistende Firma selbst tätig und ruft den Kunden aktiv an. Einsatzfelder dafür sind z.B. Versicherungs- oder Finanzmakler aber auch Rückrufaktionen auf Antwortpostkarten und ebenfalls Meinungsumfragen.

3.1.2 Wann ist ein Telefonarbeitsplatz ein Telefonarbeitsplatz?

Die heutige Penetration des Privat- und Berufslebens mit Telefonen macht die Definition eines Telefonarbeitsplatzes besonders schwierig. Telefone befinden sich mittlerweile an fast allen Arbeitsplätzen (insbesondere Büroarbeitsplätzen) direkt oder sind in der näheren Umgebung. Die Existenz eines Telefons kann deshalb noch nicht ausschlaggebend für die Entscheidung sein, ob der spezielle Arbeitsplatz als Telefonarbeitsplatz bezeichnet werden kann. Ein gutes Indiz für die Einordnung des Arbeitsplatzes ist die anteilige Zeit, die ein Arbeitnehmer während seiner gesamten Arbeitszeit mit einem bestimmten Gegenstand bzw. Arbeitsmittel verbringt. Diese Auffassung wird deshalb für die Definition eines Bildschirmarbeitsplatzes herangezogen:

> *„Bildschirmarbeit bedeutet das Arbeiten mit Computersystemen in unterschiedlichen Tätigkeitsbereichen.“ (vgl. [BAS89], Seite 2)*

In Anlehnung an dieses Verfahren und unsere Erkenntnisse definieren wir deshalb:

[4] Der Einsatz von Frauen in diesem Beruf hat, und das sei an dieser Stelle angemerkt, ursächlich etwas mit dem Frequenzspektrum der weiblichen Stimme zu tun. Der Eingeschränkte Frequenzbereich der eingesetzten Übertragungstechnik war besser für die höheren Frequenzen der weiblichen Stimme geeignet, als für die tiefer liegenden der meisten männlichen Gesprächsteilnehmer.

Definition 1. Telefonarbeitsplatz

Ein Telefonarbeitsplatz ist ein Arbeitsplatz, an dem die Erfüllung der Arbeitsaufgabe durch überwiegende Arbeit mit einem Telefon durchgeführt wird.

Beispiele für solche Arbeitsplätze finden sich z.B. in der telefonischen Bestellannahme eines Versandhauses im Gegensatz zur Postabteilung, in der die schriftlichen Bestellungen eingehen. Ein anderes Beispiel sind Versicherungsmakler, die Ihre Kunden anrufen, um für ein neues Produkt zu werben.

Die Anwendung des Begriffs „Telefonarbeitsplatz“ wird heutzutage außerhalb der Büroarbeitsplätze enden. Da sich mit dem Telefon derzeit noch ein stationärer Gedanke verbindet, der aber vielleicht schon in naher Zukunft durch mobile Telekommunikation aufgelöst werden wird. Allerdings muß dann die Frage erlaubt sein, warum eine Person, die intensiv mit Telekommunikation beschäftigt ist, gleichzeitig die Notwendigkeit verspürt, ortsunabhängig zu sein. Auf diese Problematik und die Entwicklung der Telefonarbeitsplätze kann im Rahmen dieser Arbeit leider nicht weiter eingegangen werden.

Die Fortführung der technischen Entwicklung im Bereich der Büroarbeitsplätze hat neben der Penetration mit Telefonen auch zu einer intensiven Einführung von Personalcomputern an verschiedensten Arbeitsplätzen geführt. Heutzutage kann man im allgemeinen davon ausgehen, daß jeder Büroarbeitsplatz (Kapitel 2, Leitbild) mit einem Telefon und einem Personalcomputer ausgestattet ist.

3.1.3 Szenarios des Telefonarbeitsplatzes

In diesem Unterabschnitt wollen wir Szenarios vorstellen, die wir bei der Analyse des Umfeldes erstellt haben. Sie sollen einen Eindruck darüber vermitteln, welche Tätigkeiten der Mitarbeiter zur Erledigung eines Anrufs an einem bestehenden Softwaresystem durchzuführen hat.

Das erste Szenario zeigt, wie der Beginn der Arbeit an einem Telefonarbeitsplatz im Rahmen des vorgestellten Systems aussieht:

Szenario „Am Arbeitsplatz anmelden“

Der Mitarbeiter startet zu Beginn seiner Arbeit zuerst den Rechner und bootet das Betriebssystem sowie die Benutzungsoberfläche Windows. Im Autostart des Fenstersystems ist das Programm „Agentenmonitor“ und das Programm „Gruppenmonitor“ sowie die Kundenapplikation eingestellt, so daß alle benötigten Programme automatisch gestartet werden. Diese Zeit wartet der Mitarbeiter ab.

Der „Gruppenmonitor“ dient zur Anzeige der Auslastung der Abteilung des Mitarbeiters und zeigt in Form von roten und grünen Felder die Belegung der anderen Kollegen an. Die Kundenapplikation wird vom Mitarbeiter für die eigentliche Aufgabenerledigung verwendet.

Der „Agentenmonitor“ ist das Programm zur Verbindung des Telefons mit dem PC-Arbeitsplatz. Der Mitarbeiter wählt deshalb beim Agentenmonitor seinen Namen aus der Liste der Benutzer aus

und trägt dann seine Nebenstellennummer des an seinem Arbeitsplatz befindlichen Telefons ein. Die Anmeldung ist damit abgeschlossen.

In dem Fenster des Agentenmonitors erscheint neben einem Telefonsymbol mit der Nebenstellennummer für das zugeordnete Telefon auch eine Liste von sog. Gruppen, die für jeweilige Arbeitsgruppen des Mitarbeiters stehen.

Für das Anwendungssystem leiten wir daraus folgende Anforderung ab:

Anforderung 2. Identifizierung

Es ist notwendig, neben der Identität des Anwenders auch seinen Telefonarbeitsplatz zu kennen. Der Telefonarbeitsplatz wird über die Telefon- bzw. Nebenstellennummer des benutzten Apparates gekennzeichnet. Die Kombination aus Benutzeridentität und der Telefonnummer stellt die Identifizierung *dar.*

Aufgrund der Identität des Anwenders, die durch Anmeldung über den Namen festgestellt wurde, kann das System heraussuchen, für welche Aufgaben der Mitarbeiter qualifiziert ist. Das schlägt sich in den sogenannten Gruppen nieder, für die der Mitarbeiter eingetragen wurde.

Das folgende Szenario beschreibt, wie an einem Telefonarbeitsplatz ein Gespräch bzw. ein Anruf[5] entgegengenommen wird:

Szenario „Einen Anruf entgegennehmen“

Um einen Anruf zugestellt zu bekommen, muß der Agentenmonitor dazu verwendet werden, sich beim System anzumelden und zusätzlich der Mitarbeiter in mindestens einer Gruppe angemeldet sein.

Gibt es einen Anruf für die angemeldete(n) Gruppe(n), dann kann dieser auf den Arbeitsplatz vermittelt werden. Dazu stellt das System einerseits den Anruf auf den bei der Anmeldung eingetragenen Apparat einer Telefonanlage und zeigt andererseits dies durch eine Markierung der Gruppe mit der Farbe Rot an. Die Kundendaten werden an die Kundenapplikation übergeben und das Fenster der Kundenapplikation tritt in den Vordergrund.

Das Szenario hat gezeigt, daß das Werkzeug Agentenmonitor im Falle eines Anrufs dem Sachbearbeiter eine Reihe von Informationen mitliefert. Dazu zählt die sogenannte Gruppe, über die der Anruf hereingekommen ist.

Wenngleich das direkte Gespräch mit dem Kunden vorbei sein kann, so ist die Arbeit, die den angestoßenen Sachverhalt betrifft, nicht mit dem Auflegen beendet. Das folgende Szenario macht dies deutlich:

Szenario „Nachbearbeitung eines Anrufs“

Nachdem der Mitarbeiter das Kundengespräch beendet hat, findet die Nachbearbeitung statt. In dieser Zeit wird der Mitarbeiter

[5] Die Unterscheidung zwischen Gespräch und Anruf ist dadurch gekennzeichnet, daß bei einem Gespräch nicht implizit feststeht, wer hier wen angerufen hat. Bei einem Anruf gehen wir davon aus, daß diese Information bekannt ist.

unvollständige Kundendaten ausfüllen und entscheiden, welchen Erfolg das Gespräch hatte. Hierunter fällt z.B. die Entscheidung, ob einem Kunden weitere Unterlagen zugeschickt werden sollen, oder ob ein erneuter Anruf in einem gewissen Zeitraum notwendig ist.

Nachdem diese Daten erfaßt worden sind, schaltet sich der Mitarbeiter von diesem Gespräch frei und steht wieder für ein neues Gespräch zur Verfügung.

Erst mit dem hier herausgearbeiteten Freischalten ist die Bearbeitung des Telefonats von seiten des Sachbearbeiters abgeschlossen. Man erkennt, daß eine Entscheidung über die vollständige Bearbeitung eines Anrufs nicht ausschließlich über das Auflegen des Hörers bzw. Beenden der Verbindung erkannt werden kann.

Um die Möglichkeit verschiedenartiger Ausgänge von Telefongesprächen anzudeuten, soll das nächste Szenario ein anderes Ende eines Telefongesprächs aufzeigen. Hierbei muß vorweg geschickt werden, daß der Anruf von einem Anrufautomaten (s.u.), einem sogenannten PowerDialler™, hergestellt wird. Somit hat eigentlich aus der Sicht des Kunden der Sachbearbeiter ihn angerufen.

Szenario „Recall eintragen“

Der Agentenmonitor schickt einen Job an den FirstScreen (FS). Der Agentenmonitor (AM) sendet Fensterposition und -breite an den FS. Der AM sendet „zeige dich“ (Show) an den FS.

Der FS überprüft, ob es sich um einen PowerDialler™-Job handelt. Wenn ja wird mit Hilfe der Campaign-ID der Name der Kampagne ermittelt. Mit Hilfe der Stammdaten-ID und dem Customer-Handle werden die Stammdaten ermittelt. Der FS stellt das Fenster erst dann dar, wenn alle Daten verfügbar sind.

Als erstes sichert der FS den Job mit dem Code FreeDef2 an den Agentenmonitor. Nun führt der Sachbearbeiter das Kundengespräch.

Der Sachbearbeiter trägt einen neuen Anrufzeitpunkt ein; eventuell eine andere Telefonnummer und einen Kommentar. Das wird vom FS auf Konsistenz überprüft. Durch explizites Drücken eines Knopfes wird der Recall in die Datenbank eingestellt. Der FS gibt daraufhin eine Antwort über den Erfolg dieser Eintragung (nur im Fehlerfall).

Mit der Qualifizierung des Jobs schaltet der Sachbearbeiter den FS/AM frei. Der FirstScreen trägt dazu zuerst die Qualifizierung in den Job ein, sichert diesen, indem er ihn zum AM zurückschickt, und wünscht dann die Freischaltung vom AM.

Dieses Szenario zeigt, daß automatische Anrufe z.B. dazu führen, daß der Sachbearbeiter zuerst über den Gesprächspartner informiert werden muß. Dies wird mit Hilfe der „Stammdaten“ sichergestellt, die wesentliche Informationen über den Kunden enthalten. In den Fällen, in denen das Gesprächsziel nicht erreicht wurde, muß ein weiterer Anruf durchgeführt werden. Dazu sind Kommentar und eventuell andere Telefonnummer einzutragen. In jedem Fall wird das Ergebnis des Telefongesprächs am Ende eingetragen.

Weitere Szenarios aus dem Projekt „FirstScreen“:

Szenario „ICM-Job“

Das FirstScreen-Programm wird mit dem Agentenmonitor gestartet. Das Programm besitzt standardmäßig kein Fenster. Der Agentenmonitor schickt einen Job an den FirstScreen. Der Agentenmonitor gibt dem FirstScreen die Fensterposition und -breite. Danach sendet der Agentenmonitor den Befehl zur Anzeige des FirstScreen. Nun zeigt der FirstScreen die Daten des Jobs in einem Fenster an. Nach der Beendigung des Gespräches betätigt der Sachbearbeiter den Freischalteknopf. Der Agentenmonitor schickt den Zustand freigeschaltet und die Aufforderung, das Fenster zu schließen, an den FirstScreen.

Das Szenario „ICM-Job“ entspricht der Erwartung eines „normalen“ Verlaufs eines Telefongesprächs.

Szenario „Incoming Call FirstScreen mit DBText“

Das FirstScreen-Programm wird mit dem Agentenmonitor gestartet. Das Programm besitzt standardmäßig kein Fenster. Der Agentenmonitor schickt einen Job an den FirstScreen. Der Agentenmonitor gibt dem FirstScreen die Fensterposition und -breite. Danach sendet der Agentenmonitor den Befehl zur Anzeige des FirstScreen. Dieses Kommando wird ignoriert, bis die Datenbankanfrage erfolgt ist.

Mit Hilfe des CalledDevice wird eine Anfrage an eine Kundendatenbank (Access) gestellt. Als Ergebnis kommt ein Text zurück. Dieser Text wird in einem speziellen Feld dargestellt.

Der FirstScreen zeigt nun die Daten des Jobs in einem Fenster an. Nach der Beendigung des Gespräches betätigt der Sachbearbeiter den Freischalteknopf. Der Agentenmonitor schickt den Zustand freigeschaltet und die Aufforderung, das Fenster zu schließen, an den FirstScreen.

Das Szenario „Incoming Call FirstScreen mit DBText“ stellt eine Verfeinerung des vorhergehenden vor. Um genauere Informationen über den anrufenden Kunden bereits bei der Begrüßung zur Verfügung zu haben, wird eine Datenbankanbindung verwendet, die Kundendaten aufgrund der übermittelten Rufnummer heraussucht.

3.2 Definition des Call Centers

Die historische Entwicklung des Call Centers ist eine hauptsächlich in den USA vollzogene Entwicklung. Während das Telefonieren im allgemeinen Arbeitsgeschäft eine Aufgabe unter vielen ist, wurde dort im Rahmen der Rationalisierung festgestellt, daß durch verstärkte Nutzung des Telefons eine erhebliche Einsparung im Bereich des Außendienstpersonals möglich ist. Anfahrten zum Kunden können insoweit entfallen, als daß die Kommunikation mit dem Kunden über Telefon bzw. danach dann auf postalischem Wege erledigt werden kann. Der Grundsatz „Telefonieren anstatt besuchen“ ermöglicht es, die Arbeit effektiver zu gestalten. Denn nach einem Gespräch mit dem Kunden, kann nach einer kurzen Pause sofort der nächste Kunde bedient werden.

Ein Mißstand, der bei konventionellem Telefoneinsatz unvermeidlich ist, stellt die Problematik dar, daß Anrufer häufig bis zu 10 mal klingeln lassen müssen, bevor sie einen Ansprechpartner am Apparat haben. Daraufhin folgt meist das wiederholte Vortragen des persönlichen Anliegens, bevor man mit der richtigen Person verbunden ist. Alleine nur in jedem zweiten Fall findet man überhaupt einen hilfsbereiten Gesprächspartner vor. Diese drei Punkte sprechen leider gegen den Gedanken, eine schnelle und qualitativ hochwertige Dienstleistung von Seiten des Unternehmens dem Kunden anbieten zu können.

Es gibt eine weitere ebenfalls negative Entwicklung im Bereich des Telefoneinsatzes. Kunde und Mitarbeiter der Firma werden falsch zugeordnet. Dies passiert immer dann, wenn Anrufe der Kunden zu einer ungeeigneten Person vermittelt werden. Im schlimmsten Fall kann die Weitergabe einer Telefonnummer bzw. Durchwahl eines qualifizierten Mitarbeiters dazu führen, daß dieser mit einfachen Anfragen und Vorgängen „zugeschüttet“ wird und seiner eigentlichen Tätigkeit, der Beantwortung spezieller Probleme, überhaupt nicht mehr nachkommen kann. Ein anderes Extrem ist die Weitervermittlung von Kundengesprächen an Mitarbeiter, die in der Kommunikation mit Außenstehenden nicht geschult wurden. Sie können einen unbeabsichtigt schlechten Eindruck beim Kunden hinterlassen.

Während die bisher angeführten Schwierigkeiten sowohl in den USA als auch in Europa vorzufinden sind, so gibt es signifikante Unterschiede beim Einsatz von Call Centern auf den beiden Kontinenten. Der zentrale Unterschied besteht in der Telefonaffinität der Bewohner der Länder. Die nordamerikanischen Bürger besitzen eine wesentliche höherer Affinität zum Telefon als dies auf dem europäischen Kontinent vorzufinden ist. Dies mag vielleicht auch an dem anderen Umgang mit Distanzen in den USA, bedingt durch die größeren Entfernungen, liegen. Nordamerikaner greifen wesentlich häufiger zum Telefon, um kleine Probleme des Alltags aber auch wichtige Dinge zu erledigen. Außerdem finden sich in den USA zusätzliche Konventionen und Technologien um das Telefon, die in Europa erst langsam Einzug finden. So ist es in den USA z.B. durchaus üblich, Telefonnummern nach der Vorwahl mit Buchstaben anzugeben, die sich auf jeder amerikanischen Telefontastatur neben den Ziffern befinden. Zusätzlich gilt in den USA eine einheitliche Länge der Telefonnummern, was es für die Anwender einfacher macht, sich diese zu merken. Ein technischer Gesichtspunkt ist die erhebliche Verbreitung des sogenannten Mehrfrequenz Wahlverfahrens (MFV). Es basiert auf der Überlagerung von zwei Tonfrequenzen für jede Wahlziffer, wodurch schnell und sicher mit Hilfe von Tönen gewählt werden kann. Diese Technologie kann nun aber ebenso für bestimmte automatisierbare Dienstleistungen eingesetzt werden (s.u.).

Ein besonders europäisches Problem ist die Bindung zwischen dem Kunden und seinem persönlichen Ansprechpartner. Diese Tradition ist in den USA weniger ausgeprägt. Bis zu einer bestimmten Größe eines Geschäftes sind dort durchaus wechselnde Ansprechpartner, die jederzeit ersetzt werden können, üblich. In Europa gibt es eine direkte Bindung zwischen Kunden und einem Ansprechpartner, die über Jahre hinweg aufrechterhalten und gepflegt wird. Vor der Einführung eines Call Centers muß diese starke Bindung zwischen Kunden und persönlichem Ansprechpartner zuerst aufgeweicht werden, damit der Kunde mit dem bei jedem neuen Anruf quasi zufällig ausgewählten Gesprächspartner zurechtkommt.

3.2.1 Effektive Einsatzmöglichkeiten für Call Center

Die Motivation für den Einsatz eines Call Centers kann in einer notwendigen Effizienzsteigerung begründet liegen. Diese Entscheidung wird meistens aufgrund wirtschaftlicher Gründe motiviert und tritt im Rahmen einer Rationalisierung auf. Ein anderer Grund kann die qualitative Verbesserung des Telefonkontaktes der Firma mit seinen Kunden sein. Hierbei stehen unterschiedliche Fragen im Vordergrund. Zum Beispiel kann sich die Firma dafür interessieren, wieviele Kunden keinen Gesprächspartner erreichen, weil sie auf ein Besetztzeichen treffen. Dies kann nur mit Hilfe einer sogenannten Warteschleife erreicht werden, die ausreichend groß dimensioniert ist. Viele Unternehmen sind beim erstmaligen Inbetriebgehen einer Warteschleife davon überrascht, wieviele Anrufe unbeantwortet blieben, die nun mit Hilfe der Belegung von Plätzen in der Warteschleife sichtbar gemacht werden.

Die Auswirkungen eines frustrierten Kunden, der es nach vielfachen Wählversuchen aufgibt, das Unternehmen zu erreichen, sind nicht zu unterschätzen. Tritt dies verstärkt auf, wird der Kunde unzufrieden. Eine goldene Regel der Kundenbetreuung sagt, daß ein zufriedener Kunde seine Erfahrungen etwa drei weiteren Personen mitteilt, ein unzufriedener Kunde hingegen, teilt seine negativen Erfahrungen bis zu zehn Personen mit. Der Einsatz eines Call Centers kann also in diesem Bereich dazu beitragen, daß die Erreichbarkeit eines Unternehmens ohne zusätzliche Mitarbeiter erhöht wird. Damit kann auch die Zufriedenheit des Kunden in diesem Punkt sichergestellt werden.

Da es insbesondere kostengünstiger ist, einen vorhandenen Kunden zu halten, als einen neuen Kunden zu gewinnen, macht sich alleine deshalb die Steigerung der Kundenzufriedenheit bezahlt und ist durchaus sinnvoll.

Eine deutliche Effizienzsteigerung erfährt der Telefonbetrieb durch die automatische Vermittlung des Anrufs an einen dafür qualifizierten Arbeitsplatz. Wie später noch gezeigt werden wird, gibt es eine Reihe von Kriterien, die bereits im Vorwege darüber entscheiden können, welcher Personenkreis einen Anruf entgegennehmen kann. Werden diese genutzt und stehen zusätzliche Informationen wie zum Beispiel die Telefonnummer des Anrufers[6] zur Verfügung, dann kann noch vor dem Zustandekommen eines Gespräches eine elektronische Kundenakte besorgt und auf dem Bildschirm des Sachbearbeiters angezeigt werden (vgl. Szenario).

Betrachtet man Telefontätigkeiten, bei denen das Anrufen eines Kunden von Seiten des Unternehmens im Vordergrund steht, bietet gerade dieser Bereich ab einem bestimmten Anrufvolumen ein großes Potential an Effizienzsteigerung. Ein Sachbearbeiter ist bei einer konventionellen Wählmethode in der Lage, pro Stunde etwa 5 bis 7 Anrufe bei einer durchschnittlichen Gesprächszeit von 1,5 bis 2 Minuten durchzuführen. Wesentlich zum Zeitverbrauch trägt das Heraussuchen und Wählen der Rufnummern sowie das Wiederholen des Wählvorgangs, wenn der Kunde nicht erreicht wird bzw. der Anschluß besetzt ist. Dadurch werden hochqualifizierte Sachbearbeiter mit langweiligen und frustrierenden Tätigkeiten belastet. Durch den Einsatz einer automatischen Wähleinrichtung (eines sogenannten PowerDiallers™) werden die Sachbearbeiter von

[6] Mit Hilfe des ISDN (Integrated Services Digital Network), was bereits bei vielen Firmen Einsatz findet, wird die Telefonnummer des Anrufers noch vor dem Entgegennehmen eines Telefongespräches übermittelt.

dieser Tätigkeit entlastet: Der Wählautomat hat eine Liste von anzurufenden Telefonnummern, die er mit einer gewissen Anzahl von Nummern gleichzeitig durchgeht. Erreicht der Automat durch seine Wählversuche einen Kunden, wird das Gespräch sofort zu einem Sachbearbeiter durchgestellt. Mit dieser Unterstützung lassen sich innerhalb von einer Zeitstunde weit über 20 Gespräche bearbeiten, weil alle Wählversuche, bei denen besetzt ist oder bei denen keiner das Gespräch entgegennimmt, ohne Beteiligung einer Person abgehandelt werden können.

Insgesamt tragen diese Punkte dazu bei, die Kosten des Unternehmens zu senken und die Zufriedenheit des Kunden zu erhöhen. Gleichzeitig darf man natürlich nicht verschweigen, daß durch den Einsatz der modernen Telefontechnologie die Effizienz auch in der Weise gesteigert wird, daß die Arbeitsplatzanforderungen an die einzelnen Mitarbeiter stark steigen. Durch entsprechende Gegenmaßnahmen wie regelmäßige Pausen und klar eingeteilte Arbeitszeiten für die Telefonarbeit kann aber die Zufriedenheit mit dem Arbeitsplatz erhalten werden.

Abschließend muß festgehalten werden, daß insbesondere die Möglichkeit der Auswertung und des Supervising für das einsetzende Unternehmen erstmals die Möglichkeit zur Verfügung stellt, einen Überblick über die Telefonaktivitäten seiner Mitarbeiter zu bekommen.

3.2.2 Call Center und Geschäftsprozesse

Wird der Einsatz eines Call Centers unter dem Gesichtspunkt der Geschäftsprozesse betrachtet, dann lassen sich hier interessante Gesetzmäßigkeiten feststellen und bestimmte Einsatzgebiete herausarbeiten.

Die grundlegende Feststellung im Rahmen des Telefoneinsatzes als Arbeitsmittel und Arbeitsmedium ist, daß häufig ein kommender Anruf zu mindestens einem Rückruf führt. Dies ist dadurch begründet, daß in vielen Fällen, entweder die gewünschte Person durch einen Kollegen vertreten wird, der nur einen Gesprächswunsch notiert, oder aber daß die vom Anrufer gestellte Frage ohne weitere „Nachforschungen“ nicht beantwortet werden kann. Es ist nun wünschenswert, diesen wiederholt auftretenden Kausalzusammenhang zwischen Anrufen und Rückrufen durch ein Hard- und Softwaresystem in der Weise zu unterstützen, daß für den Sachbearbeiter eine deutliche Arbeitserleichterung entsteht.

Durch die automatische Erfassung aller eingehenden Anrufe und deren Qualifizierung kann dazu beigetragen werden. Unter einer Qualifizierung versteht man die Bewertung des Gespräches im Sinne einer Auswahl zwischen

- Gespräch erledigt
- Rückruf erwünscht
- Ruft erneut an
- Folgegespräch notwendig

etc.

Definition 3. Qualifizierung
Unter dem Qualifizieren *eines Anrufs versteht man die Bewertung des Anrufs am Gesprächsende. Diese Bewertung entscheidet darüber, ob das Gespräch im Rahmen*

einer Tätigkeit diese abgeschlossen hat oder ob weitere (telefonische) Gespräche notwendig sind.

Die Kategorien zur Qualifizierung hängen stark vom Einsatzfeld ab. Das soll anhand von Qualifizierungen für ausgehende Anrufe verdeutlicht werden:

- Anrufbeantworter
- Person nicht anwesend
- Person ruft zurück
- Neuanruf heute abend
- Neuanruf morgen
- Gespräch erledigt

Man sieht, daß bei eingehenden Anrufen z.B. die Kategorien „Anrufbeantworter“ und „Person nicht anwesend“ gar nicht auftreten können.

Das oben angesprochene Problem des „ewigen“ Weitervermittelns kann mit Hilfe eines Call Centers minimiert werden. Dazu wird eine zweistufige Gesprächsannahme definiert. In der ersten Stufe werden die Anrufer lediglich entgegengenommen, begrüßt und nach ihrem Anliegen gefragt. Diese Arbeit kann von wenig qualifizierten Personen durchgeführt werden, deren kommunikative Fähigkeiten besonders hoch sind. Durch eine Vorqualifizierung der Gespräche wird von diesen Personen mit ausgewählten Fragen ermittelt, welcher Mitarbeiterkreis der Firma die bestmögliche Bearbeitung der Kundenfrage leisten kann. Mit diesen Informationen wird der Anruf wieder in das Call Center System gegeben und von diesem entsprechend weitergeleitet.

Definition 4. Vorqualifizierung

Unter dem Vorqualifizieren *eines Anrufs versteht man die Einordnung des Anrufs am Gesprächsbeginn. Durch ausgewählte Fragen des Sachbearbeiters wird der Kunde eine Gruppe zugeordnet. Diese Bewertung entscheidet, welcher Personenkreis als nächstes das Gespräch entgegennehmen wird.*

Ein wichtiger Gesichtspunkt der Vorqualifizierung ist die Möglichkeit, bereits in dieser Phase gewisse Kundendaten aufzunehmen. Diese Kundendaten werden vom Call Center System mit dem Anruf an den Arbeitsplatz des späteren Sachbearbeiters mitvermittelt. Dadurch entfällt für diese hochqualifizierte Person eine weitere lästige Routinetätigkeit.

Bietet ein über mehrere Standorte verteiltes Unternehmen eine telefonische Hilfestellung für Kunden an, die deren Produkt einsetzen, dann stellt sich in diesem Einsatzfeld häufig das Problem, daß die Spezialisten für verschiedene Fragestellungen auf unterschiedliche Standorte verteilt sind. Mit Hilfe eines Call Centers lassen sich diese Standorte zu einem virtuellen Standort kombinieren, bei dem das Weiterreichen eines Gesprächs - und auch der bereits ermittelten Informationen über den Anrufer - durch die zugrundeliegende Technologie erledigt wird.

Durch die Einführung eines mehrstufigen Konzeptes wird ein weiterer negativer Effekt im Arbeitsablauf vermieden: der Sägezahn-Effekt. Darunter versteht man die durch das Telefon hervorgerufene Unterbrechung im Arbeitsfluß. Es ist allgemein bekannt, daß erst

nach einer gewissen Einarbeitungszeit, die Leistung eines Mitarbeiters an einer bestimmten Aufgabe auf seinem persönlichen Maximum ist. Durch einen eingehenden Anruf wird der Mitarbeiter in seinem Arbeitsfluß unterbrochen und die Konzentration auf die zu bearbeitende Aufgabe sinkt. Nach dem Anruf ist wieder eine gewisse Zeit notwendig, bis der Mitarbeiter die volle Konzentration dafür aufwenden kann. Es ist deshalb notwendig, die telefonischen Unterbrechungen im Arbeitsfluß einerseits auf ein Minimum zu reduzieren und andererseits die Telefongespräche auf eine bestimmte Tageszeit zu konzentrieren, so daß kein unmotivierter Wechsel zwischen verschiedenen Tätigkeiten für den Mitarbeiter notwendig wird.

Diese Anforderungen werden durch einen dreistufige Unterteilung der Mitarbeiter realisiert. In der ersten Stufe, dem First Level, werden Telefongespräche von wenig qualifizierten Mitarbeitern entgegengenommen, vorqualifiziert und einfachste Fragen bereits beantwortet. Alle nicht bereits beantworteten Anfragen werden nach der Vorqualifizierung an die zweite Stufe, den Second Level, weiter gereicht. Hier sitzen Sachbearbeiter mit einer guten Qualifikation für das Problemfeld, die bereits einen Großteil der gestellten Fragen beantworten können. Nur in Ausnahmefällen, wenn keine befriedigende Antwort gefunden werden konnte, wird der Kunde mit der dritten Stufe, dem Third Level, verbunden. Der Third Level besteht aus hochqualifizierten Mitarbeitern, deren Hauptaufgabe in der Entwicklung und nicht im Kundenkontakt liegt.

3.2.3 Entscheidung zwischen Inbound und Outbound

Der Einsatz eines Call Centers kann grundsätzlich in den zwei bereits angedeuteten Bereiche erfolgen: Eingehende Anrufe (sog. Inbound-Geschäft) und Ausgehende Anrufe (sog. Outbound-Geschäft).

Definition 5. Inbound

Unter Inbound *versteht man all diejenigen Anrufe, die aktiv durch den Kunden durchgeführt werden. Daß heißt der Kunde ruft das Unternehmen an. Charakteristisch bei diesen Anrufen ist, daß vorher im allgemeinen nur wenig über den Kunden bekannt ist.*

Die Qualität des Wissens über den Kunden kann durch drei Maßnahmen erheblich gesteigert werden. Erstens kann eine im ISDN mitgelieferte Telefonnummer des Anrufers dazu führen, daß die Person bereits vor dem Entgegennehmen des Gesprächs identifiziert wird. Zweitens kann einem Kunden eine spezielle Durchwahl bzw. Nachwahl[7] gegeben werden, die ihn im Moment des Anrufens auf dieser Nummer identifiziert. Drittens können Anrufer (wie oben bereits erwähnt) durch Eingabe ihrer Kundennummer (o.ä.) vorqualifiziert werden.

Definition 6. Outbound

Unter Outbound *versteht man all diejenigen Anrufe, die aktiv durch das Unternehmen durchgeführt werden. Daß heißt die Firma ruft den Kunden an. Bei*

[7] Durchwahl: Ziffern einer Telefonnummer, die vor dem Zustandekommen eines Freizeichens gewählt werden
Nachwahl: Ziffern, die nach dem Zustandekommen eines Gesprächs gewählt werden

diesem Typus von Anrufen sind bereits im Vorwege ein Großteil der Informationen über den Kunden vorhanden und stehen beim Gespräch zur Verfügung.

Der Inbound-Bereich eignet sich besonders zum Einsatz für Hotlines, für Produkt-Support, für Auftragsannahme und Direct Response TV.

Der Outbound-Bereich ist prädestiniert für Inkasso, Marktforschung und Kundenbindung. Außerdem eignet er sich für Kundenbefragungen, Kundenresonanz, Support (im Falle eines Rückrufs, wenn das Problem nicht sofort beim Anruf des Kunden geklärt werden konnte) und für allgemeine Rückrufe. Outbound Call Center sind für das Inkassogeschäft besonders geeignet, weil der telefonische Mahnweg wegen des persönlichen Gesprächs wesentlich erfolgreicher ist als der schriftliche.

3.3 Automatic Call Distribution

Unter „automatischer Anrufverteilung“ oder englisch „Automatic Call Distribution“ versteht man die Technologie, die notwendig ist, um eine große Menge von Anrufen (eingehend oder ausgehend) auf eine Reihe von Telefonarbeitsplätzen so zu verteilen, daß eine optimale Gesprächsbearbeitung ermöglicht wird.

Definition 7. Automatic Call Distribution

Automatic Call Distribution *(ACD) bezeichnet die Technologie, mit der es möglich ist, eine große Menge von eingehenden oder ausgehenden Anrufen auf Telefone zu verteilen.*

Man spricht in diesem Zusammenhang von Service-Grad, wenn man die Qualität eines Call Centers messen möchte. Der Service-Grad gibt in Prozent an, wieviele Anrufe erfolgreich bearbeitet werden konnten. Von der Hundert-Prozent-Marke werden diejenigen abgezogen, die wegen eines Besetztzeichens keine Verbindung bekommen haben; es werden diejenigen Anrufe abgezogene, die wegen einer zu langen Wartezeit in der Warteschlange aufgelegt habe und es werden diejenigen Anrufe abgezogen, die wegen technischer Probleme (z.B. Fehlvermittlung) nicht zum richtigen Ansprechpartner vermittelt werden konnten.

Zur Realisierung der ACD sind eine Reihe von Voraussetzungen zur erfüllen, die in diesem Abschnitt näher vorgestellt werden sollen. Außerdem werden die Schwächen dieses Ansatzes aufgezeigt.

Grundsätzlich geht man bei der ACD davon aus, daß Anrufe in größerer Menge z.B. auf einer dafür bestimmten Sammelnummer eingehen. Unter einer Sammelnummer stehen mehrere Amtsleitungen zur Verfügung. Die Sammelnummer selbst ist nur eine logische Einheit, die auf (eine oder mehrere) Nebenstellen einer Telefonanlage geleitet werden kann. Die Aufgabe der ACD ist es nun, diese eingehenden Anrufe auf eine vorher definierte Menge von Telefonarbeitsplätzen zu verteilen. Dabei wird berücksichtigt, ob und welche Arbeitsplätze davon besetzt sind, so daß nur auf die Arbeitsplätze ein Gespräch geleitet wird, an denen auch ein Mitarbeiter sitzt. Außerdem wird eine gewissen Anzahl von überzähligen Gesprächen in eine Warteschlange vermittelt und dort gehalten, bis ein Arbeitsplatz frei wird.

Die Strategie, mit der die Anrufe auf die Arbeitsplätze verteilt werden, kann unterschiedlich gewählt werden. So ist es z.B. in vielen Fällen sinnvoll, den Anrufer zuerst mit einem Ansage- und Begrüßungstext zu empfangen. Danach wird der Anruf, sofern keine Plätze frei sind, zuerst in die Warteschleife geleitet. Die Warteschleife entspricht einem FIFO (First In First Out) Speicher. Wird ein Arbeitsplatz eines Sachbearbeiters für ein Gespräch frei, wird der nächste Anruf aus der Warteschleife genommen und auf den Arbeitsplatz vermittelt.

3.3.1 Technische Grundlagen der Anrufverteilung

Die technischen Grundlagen für eine automatische Anrufverteilung konzentrieren sich im wesentlichen auf die Steuerung der Telefonanlage. Gespräche, die auf der oben beschriebenen Sammelnummer eingehen, müssen durch die ACD auf einen freien Arbeitsplatz oder in die Warteschlange geleitet werden. Dazu sendet die ACD Steuerbefehle an die Telefonanlage, in denen die Vermittlung des Gespräches auf einen neuen Apparat gefordert wird. Dieser neue Apparat kann dann entweder das Mitarbeitertelefon sein oder ein Platz in der Warteschlange.

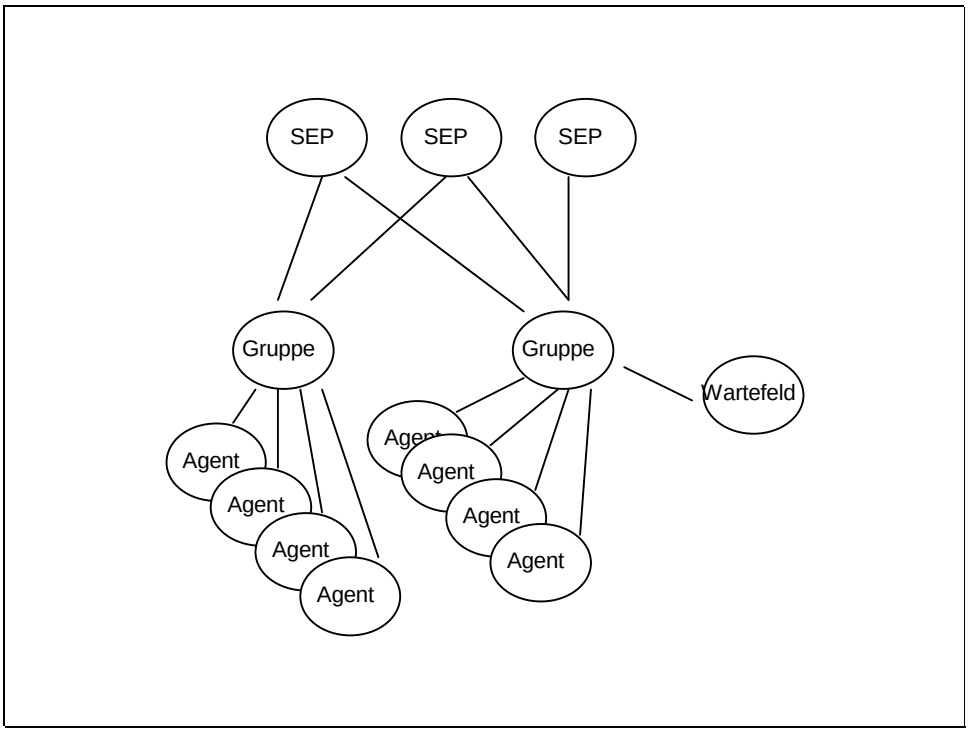

Abbildung 5 Logische Organisation eines Call Centers

Die Abbildung 5 zeigt die organisatorische Struktur einer Anrufverteilung. Die über Sammelnummern eingehende Anrufe gelangen in einen Service Entry Point (SEP). Von diesem wird entschieden, welche Gruppe von Mitarbeitern qualifiziert ist, den Anruf entgegen zu nehmen. Daraufhin gelangt der Anruf in eine Gruppe. Die Gruppe ist eine organisatorische Einheit, die das Wissen über die Belegung der Sachbearbeiter besitzt. Sind alle Sachbearbeiter belegt, wird der Anruf in die Warteschlange (hier Wartefeld genannt) geleitet. Dies löst eine Vermittlung in der Telefonanlage aus. Wird ein Mitarbeiter (hier Agent genannt) frei, überprüft die Gruppe, ob noch Anrufe in der

Warteschlange stehen. Diese werden dann über einen neuen Vermittlungsauftrag an die Telefonanlage zum Agenten übertragen.

Die Steuerleitung zur Telefonanlage, mit der die Gesprächsvermittlung von der Ansage zum Wartefeld und vom Wartefeld zum Agenten durchgeführt wird, ist ein sogenannter CLI-Link. Unter CLI versteht man das „Command Line Interface“, eine genormte Schnittstelle, über die Steuerbefehle an eine Telefonanlage geschickt werden können. Mit Hilfe dieser Steuerbefehle können Gesprächsvermittlungen durchgeführt werden. Ein Gespräch kann von einer Nebenstelle zu einer anderen Nebenstelle innerhalb der Anlage transferiert werden.

In dieser einfachen Form der ACD wird neben einem Server, der die Logik der Anrufverteilung übernimmt, keine weitere PC Hardware benötigt. Erst wenn man einen Schritt weitergeht und auch die Kundendaten in Beziehung zu einem Telefongespräch setzt, wird eine zusätzliche Integration einer vorhandenen PC-Infrastruktur möglich.

Durch die Kombination der Telefonanlage mit einer ACD-Anlage und dem PC-Netzwerk kann eine optimale Integration erreicht werden. Befindet sich neben dem Telefon an jedem Arbeitsplatz ein Personalcomputer, der mit einem Server über ein lokales Netzwerk miteinander verbunden ist, kann nicht nur das Telefongespräch an den Arbeitsplatz vermittelt werden, sondern auch zusätzliche Informationen zum Telefongespräch, die z.B. die Kundenakte oder andere Gesprächsinformationen enthalten.

Die nachstehende Abbildung zeigt schematisch, wie ein solcher Call Center Server einerseits mit der Telefonanlage verbunden ist und andererseits durch eine Einbindung in das LAN von der Kommunikation zum PC profitiert. Bemerkenswert an dieser Konstellation ist, daß es keine Verbindung zwischen Telefon und PC pro Arbeitsplatz gibt. Diese Zuordnung wird rein virtuell hergestellt.

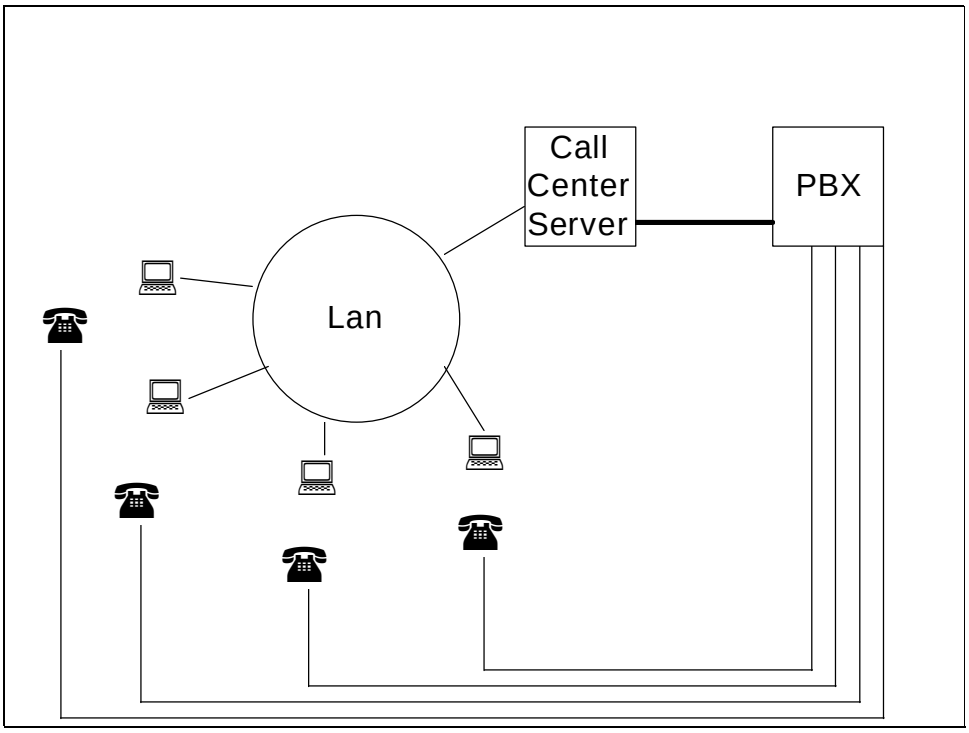

Abbildung 6 Verknüpfung zwischen LAN und Telefonanlage

An jedem Arbeitsplatz stehen mindestens Standardtelefone zur Verfügung. Das macht deutlich, daß als technische Grundlage für die ACD keine besonderen Telefone notwendig sind. Ansätze, bei denen die Telefondaten (Gesprächsdigitalisierung) über das PC-Netz gesendet werden und dementsprechend das Telefon mit dem PC verbunden sein muß, sind nicht notwendig. Wenngleich in vielen Anwendungsfällen, besondere Telefone verwendet werden, bei denen das Halten eines Telefonhörers durch einen sogenannten Sprachbügel ersetzt wird und das Abheben beim Entgegennehmen des Gesprächs durch einen technische Einrichtung automatisiert wird.

3.4 Jobvermittlung als Metakonzept

Bei der reinen Anrufvermittlung wird das Telefongespräch von einem Telefon zu einem nächsten weitervermittelt. Dieses Konzept bietet zwar über die Gruppierung bereits genügend Ansätze zu einer automatisierten Anrufverteilung, nutzt aber die Potentiale einer Computerunterstützung nur unzureichend aus.

Werden die vom Call Center Server vermittelten Gespräche nicht nur als Telefongespräche aufgefaßt, sondern vielmehr als Arbeitsaufträge, die im System so lange kursieren, bis sie ausreichend bearbeitet worden sind, dann entsteht ein Metakonzept, das hier kurz vorgestellt werden soll.

Das Konzept, welches Entsteht, wenn zu einem Anruf parallel ein Auftrag im System entsteht, wird Jobvermittlung genannt. In diesem Fall erzeugt jeder Anruf, der von außen in das System gelangt oder vom System selbst ausgelöst wird, einen sogenannten Job. Der Job ist die logische Zusammenfassung von Informationen, die dieses Gespräch begleiten. Zu den Attributen eines Jobs gehört der Entstehungszeitpunkt und eine Liste von Instanzen, die mit der Bearbeitung des Jobs bisher betraut wurden. Außerdem gibt

es eine Job-Charakterisierung unter den Attributen, die es potentiellen Job-Empfängern erleichtert zu entscheiden, ob sie für die Bearbeitung des Jobs in Frage kommen oder nicht.

Erst wenn ein Job unter den Komponenten eines Call Center Servers an eine zuständige Instanz verteilt werden konnte, löst diese Instanz im Gegensatz zum ACD System, die Vermittlung des dazugehörigen Gesprächs auf den entsprechenden Arbeitsplatz aus.

Die nachfolgende Abbildung zeigt dies exemplarisch:

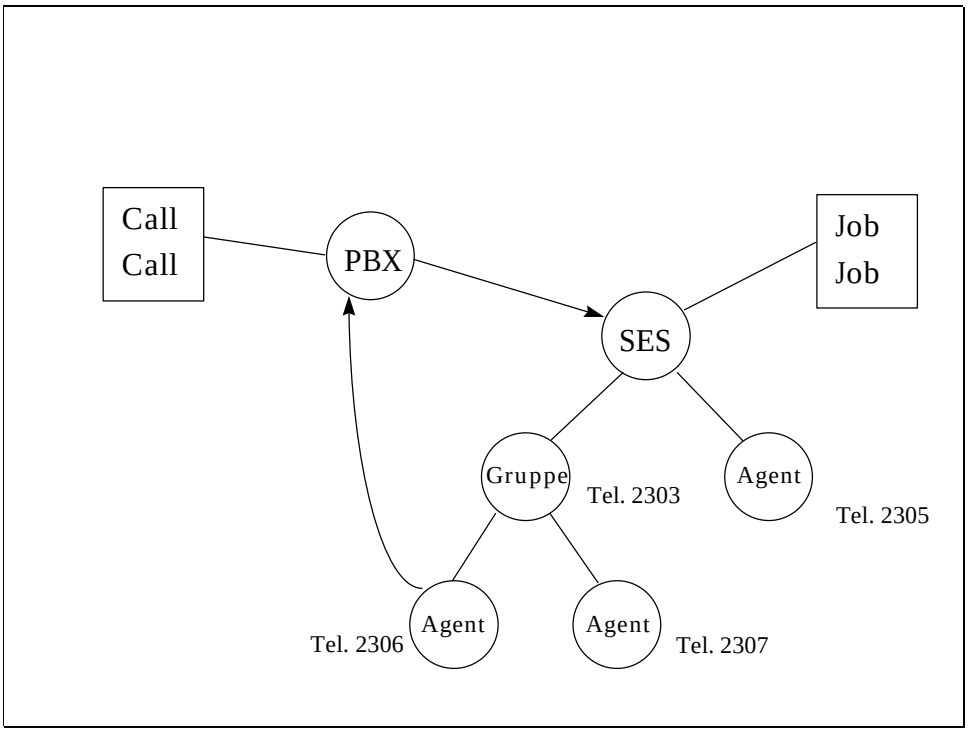

Abbildung 7 Jobfluß

Der Anruf eines Kunden (links in Abbildung 7) wird über die Telefonanlage (PBX) an den Service Event Supervisor (SES) signalisiert, welcher daraufhin dazu passend einen Job erzeugt (rechts im Bild). Der Job selbst wird unter den Komponenten weitergereicht und von einer Instanz akzeptiert (hier Agent 2306). Dieser löst dann die Vermittlung des dazugehörigen Telefongesprächs aus.

Neben den bisher angesprochenen Informationen kann ein Job auch folgende Informationen mit sich tragen:

- Wahlversuche
- Erfolglose Verbindungen
- Referenz auf Kundendaten
- Wiederholungen
- Qualifikation des Jobs
- Zeitstempel, Protokoll

Es bleibt wichtig festzuhalten, daß die Lebenszeit des Jobs länger sein kann als der Anruf, der zu diesem Job gehört. Das ist insbesondere dann der Fall, wenn spätere Rückrufe notwendig sind. Dafür verbleibt der Job im System und wird weitergeführt, wenn der Zeitpunkt des Rückrufs gekommen ist.

3.5 Konfiguration

Die Anforderungen des Betreibers eines Call Centers an dessen Anpassungsfähigkeit soll Thema dieses Abschnitts sein. Unter einer Konfiguration eines Call Centers verstehen wir die Beschreibung seiner Komponenten und ihres Zusammenspiels.

Eine Konfiguration eines Call Centers legt fest, welche Service Entry Points es gibt und welche Telefonnummern diesen zugeordnet sind. Sie legt weiterhin fest, welche Gruppen im System vorhanden sind und welche Dienstleistungen sie erbringen können. Für jede Gruppe wird ggf. eine Warteschlange mit einer bestimmten Anzahl von Plätzen definiert. Dazu kommen die Agenten und ihre Zuordnung in die unterschiedlichen Gruppen. Neben diesen herausgegriffenen Merkmalen gibt es weitere, auf deren Betrachtung wir an dieser Stelle verzichten wollen.

Eine so definierte Konfiguration kann für ein geplantes Einsatzszenario angepaßt sein. Ändert sich dieses Szenario, weil z.B. die Dienstleistungen der angenommenen Firma wechseln, dann muß auch die Konfiguration daraufhin angepaßt werden. Dies tritt z.B. ein, wenn bei Saisonbetrieb zu bestimmten Monaten Änderungen notwendig sind. Oder aber im Dienstleistungsbereich, wenn der Call Center Betreiber für unterschiedliche Kunden immer wieder entsprechende Kapazitäten anbietet, kann es notwendig sein, auf eine geänderte Konfiguration zurückzugreifen.

All dies führt dazu, daß man zu einem Konfigurationsmanagement gelangt, bei dem Konfigurationen gesichert und wiederverwendet und bei dem Änderungen an Konfigurationen durchgeführt werden können. Diese Änderungen möchte man möglichst während des Betriebs durchführen, damit ein möglichst reibungsloser Ablauf gewährleistet ist. Insbesondere betrifft dies kleinere Änderungen, die z.B. durch Mitarbeiterfluktuation entstehen.

Das in Kapitel 4 vorzustellende Anwendungssystem stellt einen Lösungsansatz dar, der die Online Konfiguration eines Call Centers ermöglicht. Dabei werden ebenfalls verteilte Standorte unterstützt, die im folgenden Abschnitt untersucht werden.

3.6 Verteilte Call Center

Wie oben bereits angesprochen ergeben sich aus der räumlichen Aufteilung eines Unternehmens auch spezielle Anforderungen an die Aufteilung eins Call Centers. In diesem Abschnitt des Kapitels wird deshalb untersucht und gezeigt, wie prinzipiell die Verknüpfung mehrerer Standorte auch die Verteilung von Anrufen zwischen diesen Standorten ermöglicht.

Die räumliche Verteilung eines Call Centers kann aus verschiedenen Gründen motiviert sein. Ist das Call Center nur zur Erfüllung einer Aufgabe gedacht und wird auf der „grünen Wiese“ geplant, spricht dennoch eine Reihe von Argumenten für eine Verteilung. Erstens ist im Einzugsgebiet eines Einsatzortes nur eine begrenzte Anzahl von qualifizierten Mitarbeitern zu gewinnen. Das bedeutet, daß auch das Call Center nur

eine gewisse Größe annehmen kann. Durch eine räumliche Aufteilung wird auch das Einzugsgebiet entsprechend vergrößert. Zweitens erhöht die mehrfache Einrichtung eines Call Centers die Ausfallsicherheit des Gesamtsystems, da kurzfristig ein anderer Standort ausgefallene Kapazitäten ersetzen kann. Das dritte Argument für mehrere Standorte bei gleichem Einsatzfeld findet sich in der Weiterentwicklung von Technologie und Einsatzgebiet. Neue Funktionalitäten bzw. Aufgabenfelder können an einem Standort erst in kleinem Rahmen ausprobiert werden, bevor sie auf das gesamte Unternehmen Call Center übertragen werden.

Bei Unternehmen, deren Standorte historisch gewachsen sind, und bei denen bestehende Mitarbeiter in ein Call Center eingebunden werden, z.B. zur Erfüllung von Hifestellung bei Kundenproblemen, macht es keinen Sinn, ein zentrales Call Center zu fordern. Hier müssen die Spezialisten standortübergreifend in ein virtuelles Call Center eingebunden werden. Technische Einrichtungen ermöglichen dann das Vermitteln des Kundengesprächs an den Ort, an dem ein gesuchter Spezialist sitzt.

Die nachfolgende Abbildung zeigt ein Beispiel für ein Call Center eines Unternehmens, das über drei Standorte miteinander vernetzt wurde:

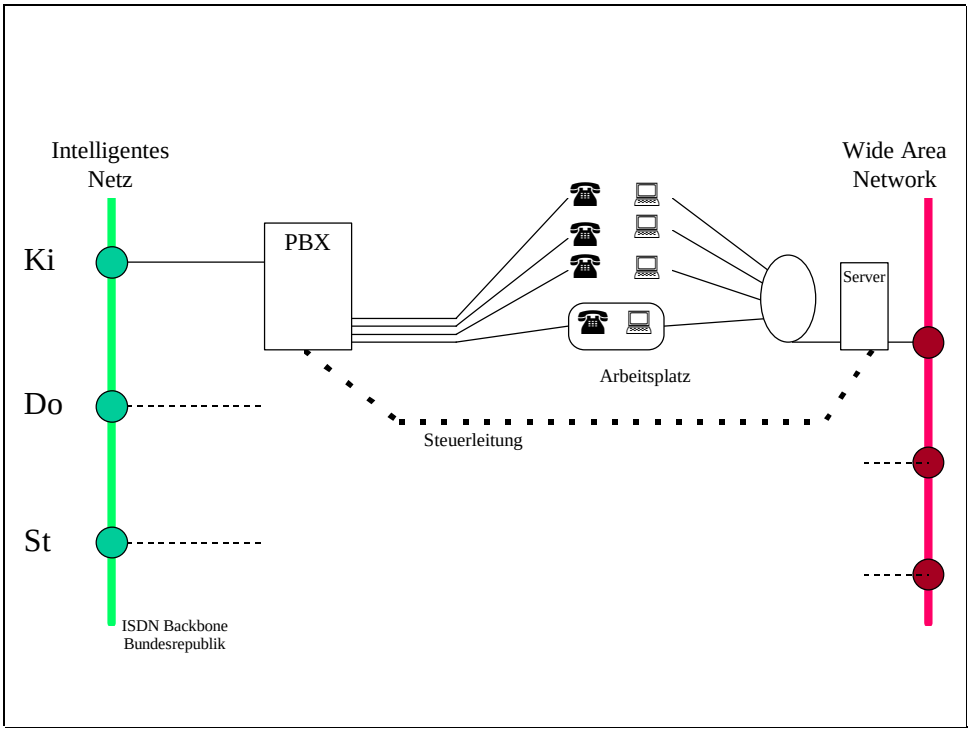

Abbildung 8 Prinzipieller Aufbau eines verteilten Call-Centers

Nimmt man für das Beispiel an, daß das Unternehmen seinen Kunden eine Rufnummer im Intelligenten Netz[8] der Deutschen Telekom zur Verfügung stellt, wenn diese eine Informationen zu seinen Produkten wünschen.

In der Abbildung 8 erkennt man auf der linken Seite die drei Standorte, die in den beispielhaft gewählten Orten Kiel, Dortmund und Stuttgart liegen. Sie sind über das

[8] Produktbezeichnung

Intelligente Netz miteinander verbunden. Dieses Netz leitet den Anruf des Kunden zuerst an den Standort, der räumlich am nächsten liegt. Nur, wenn dieser z.B. aus Kapazitätsmangel den Anruf ablehnt, wird das Gespräch noch vor dem Entgegennehmen an einen anderen Standort weitergeleitet.

Gelangt ein Anruf über die Nebenstellenanlage (PBX, engl. Public Branch Exchange) in das Call Center wird dazu ein Job erzeugt, der dieses Gespräch über seinen Verlauf hinweg begleitet. Entsprechend der Vermittlungsstrategie des Call Centers gelangt der Anruf ggf. nach einem Aufenthalt in einer Ansage- oder Warteschleife zu einem Mitarbeiter des First Level, der den Anruf vorqualifiziert und einfache Fragen des Kunden sofort beantwortet.

Bleiben Fragen des Kunden offen, wird das Gespräch bedingt durch seine Vorqualifizierung an einen Kreis von Mitarbeitern weitervermittelt (Second Level), die für die Beantwortung seiner vorgetragenen Fragen qualifiziert sind. Nehmen wir hierbei an, daß diese Mitarbeiter noch am selben Standort sitzen. Die Weitergabe des Gesprächs erfolgt also ausschließlich über die lokale Telefonanlage. Dazu sendet der Call Center Server (in der Abbildung mit Server bezeichnet) einen Steuerbefehl an die Telefonanlage, die daraufhin das Gespräch an eine andere Nebenstelle weiterleitet.

Erst wenn auch hier keine ausreichende Beantwortung des Sachverhalts erzielt werden konnte, muß das Gespräch zum Third Level weitergeleitet werden. Dieser ist im allgemeinen über die Standorte verteilt und nicht am lokalen Ort vorzufinden. Gegebenenfalls sind Mitarbeiter sogar über Heimarbeitsplätze in die Beratung integriert. Der Call Center Server gibt dazu ein Steuerkommando an die Telefonanlage, das Gespräch an einen anderen Standort zu vermitteln. Dann erfolgt die Weitergabe des Gesprächs wieder über das Intelligente Netz. Zusätzlich zum Gespräch werden auch die Jobinformationen weitergereicht. Diese sendet der Call Center Server über das Wide Area Network an den Server am Zielstandort oder an den Heimarbeitsplatz. Dort werden Job und Telefongespräch wieder zusammengefaßt und an einen Arbeitsplatz vermittelt.

3.7 Zusammenfassung

In diesem Kapitel wurden Einsatzgebiete für Telefonarbeitsplätze anhand von Beispielszenarios vorgestellt. Die Unterscheidung zwischen eingehenden und ausgehenden Anrufen wurde getroffen. Bezugnehmend auf die historische Entwicklung der Call Center konnte eine signifikante Unterscheidung zwischen den USA und Europa aufgezeigt werden. In Europa besteht eine intensive Bindung des Kunden zu speziellen Mitarbeitern. Es gibt nur ein geringes Verhältnis zum Telefon und dessen Einsatz für Tätigkeiten im Geschäftsfeld.

Als Beispiele für Einsatzmöglichkeiten von Call Centern wurden Hotlines, Marketing, Produkt Support und Direct Response TV genannt. Im Zusammenhang der Einsatzfelder wurden die wichtigsten Unterscheidungen und Tätigkeiten im Bereich der Telefonie eingeführt. Die Gesprächsrichtung wird mit Inbound oder Outbound bezeichnet, durch technische Einheiten oder Personal kann ein Gespräch vorqualifiziert und qualifiziert werden.

Der Abschnitt über die ACD Technologie hat auf die Grundlagen der Telefonie verwiesen. Diese wurde mit dem Jobkonzept verglichen, welches weitaus mehr Möglichkeiten bietet.

Die Konfiguration von Call Centern wurde als wesentliche Anforderung motiviert. Wesentliche Forderung dabei ist die Möglichkeit, die Konfiguration während des Betriebs vorzunehmen und auf aktuelle Anforderungen schnell reagieren zu können. Verteilte Call Center stellen dafür ein besondere Anwendungsfeld dar, weil sie durch die räumliche Anordnung nur schwer mit herkömmlichen Verfahren konfiguriert werden können.

4 Das *Online Config Tool*

Der hier präsentierte Prototyp stellt die komponentenbasierte Realisation eines Online Config Tools zur Konfiguration eines verteilten Call Centers während der Laufzeit dar. Der gewählte Ansatz wird hier mit Hilfe von Visionen motiviert und in Form von Screenshots in der Umsetzung vorgestellt.

Die zentral handelnde Person ist der Administrator. Mit den im vorherigen Kapitel beschriebenen Szenarios läßt sich seine funktionalle Rolle von denen der Sachbearbeiter abgrenzen. Zu seinen Tätigkeiten gehört die Überwachung eines laufenden Call Centers und dessen Anpassung an die sich aus der Arbeit ergebenden Anforderungen.

4.1 Online Konfiguration

In diesem Abschnitt wird eine Auswahl der Visionen vorgestellt, die als Diskussionsgrundlage für die Erstellung der Software-Werkzeuge dienten. Ein begleitender Screenshot zeigt die Umsetzung des Vorgangs in ein reales Werkzeug. Dabei sollen die Visionen dem Leser dazu dienen, anhand von zwei ausgewählten Fällen exemplarisch den Umgang mit dem erstellten Anwendungssystem kennenzulernen.

Der Startvorgang des Online Config Tools stellt die erste Integrationsschwelle bei der Einbettung in bestehende Betriebssystemumgebungen dar. Einerseits muß hier die Anforderung der Benutzeridentifikation (vgl. Anforderung 1) erfüllt werden, andererseits kommen im Anwendungsfeld der Telefonie neben der Benutzeridentifikation zusätzliche Angaben hinzu. Wie die nachfolgende Vision zeigt, werden alle sich während einer Anmeldung nicht ändernden Angaben in diesem Vorgang kombiniert und in ein Werkzeug eingebettet.

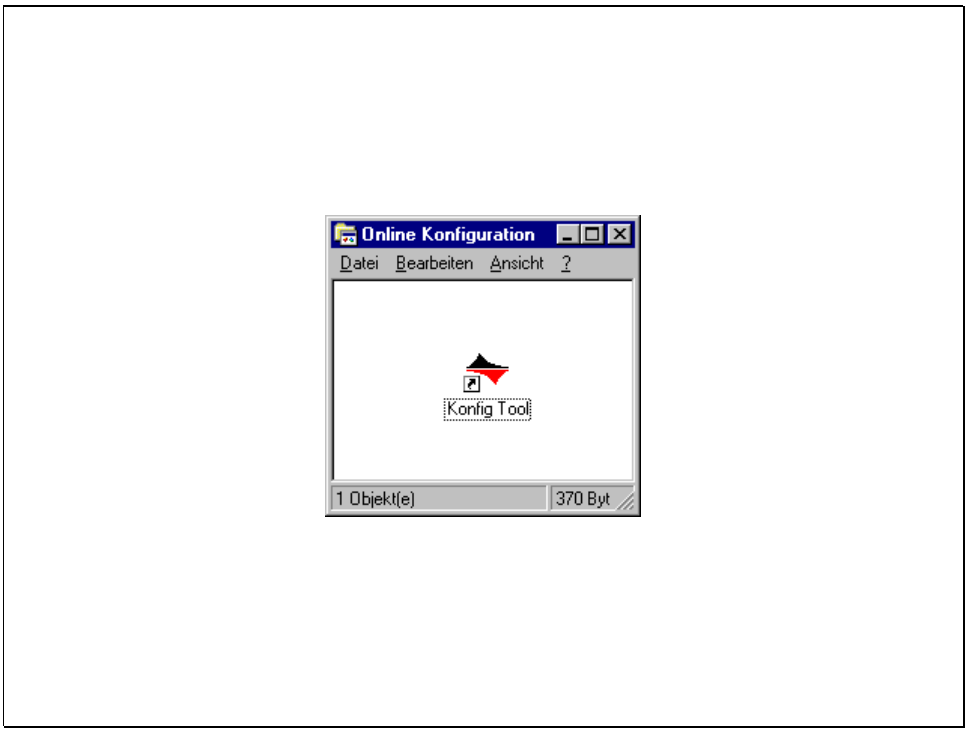

Abbildung 9 Das Anwendungssystem

Die Abbildung 9 zeigt eine Programmgruppe mit der Verknüpfung des Startbatches für dieses Anwendungssystem. Für den Anwender gibt es keinen unterschied zu anderen Programmen und Anwendungssystemen.

Vision 1: Das Anwendungssystem starten

Nachdem der Administrator das Online Config Tool über einen Doppelklick gestartet hat, wird er aufgefordert, sich anzumelden. Im Anmeldedialog (siehe Abbildung) wird er aufgefordert, seinen Standort, den Anwendernamen sowie die Nebenstellenanlage und die Nebenstellennummer anzugeben. Gegebenenfalls ist auch die Eingabe eines Kennwortes erforderlich.

Um die Integration des Systems in bestehende Umgebungen zu erhöhen wird ein Großteil der Eingaben bereits mit Standardwerten vorbelegt. So ist der Standort durch das Netzwerk, in dem das Anwendungssystem gestartet wird, standardmäßig auf den lokalen Standort voreingestellt. Nur wenn der Anwender von einem anderen Standort kommt, ist hier eine Eingabe erforderlich. Ebenso verhält es sich mit der Nebenstellenanlage. Diese muß nur in den Fällen geändert werden, bei denen der Anwender eine besondere Arbeitssituation durchspielt. Im allgemeinen ist für den Anwender bereits mit der Auswahl des Benutzernamens eine Voreinstellung der Nebenstellenanlage und der Nebenstellennummer gegeben.

Aufgrund der Anmeldung werden bestimmte Funktionen (sogenannte Schlösser) der weiteren Werkzeuge freigeschaltet oder sind nicht auswählbar (sichtbar).

Nach dem Anmelden werden die Autostart-Werkzeuge gestartet. Für jeden Anwender kann eine Reihe von Werkzeugen festgelegt werden, die beim Anmelden aufgestartet werden sollen. Im allgemeinen sind dies Werkzeuge wie z.B. das Palettenwerkzeug, mit dem weitere Werkzeuge gestartet werden können. Handelt es sich um einen Sachbearbeiter, werden hier die standardmäßig notwendigen Werkzeuge zur Anzeige der Arbeitsgruppe und Warteschlange sowie der Agentenmonitor spezifiziert.

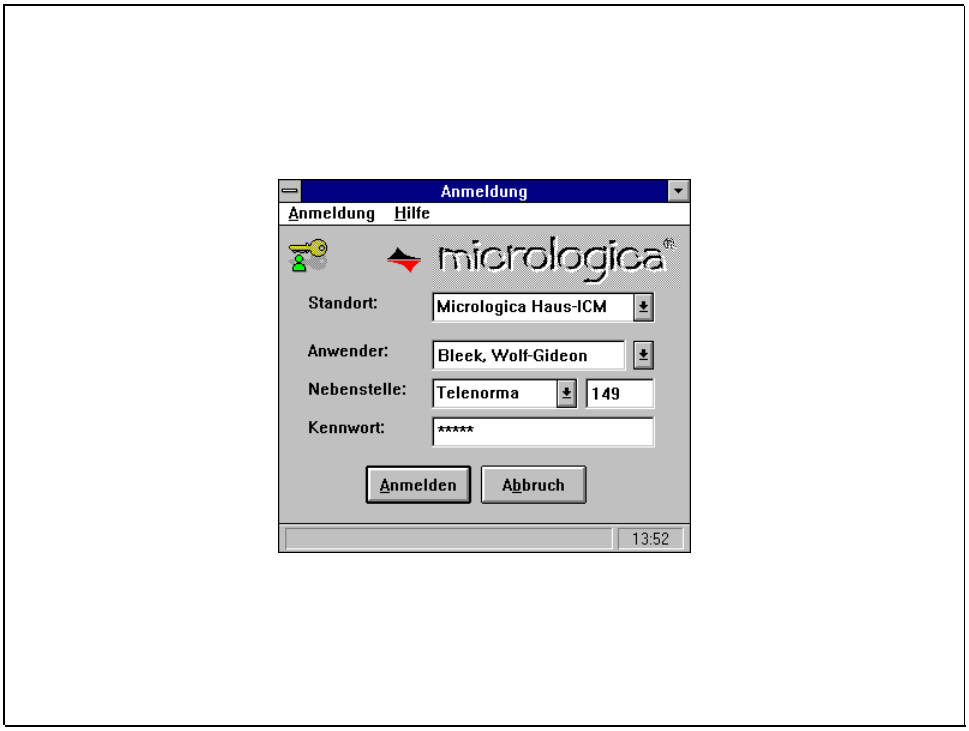

Abbildung 10 Das Anmeldewerkzeug

Die Anmeldung ist aus ihrer Sache heraus der einzige Vorgang, der dem Anwender bei der Benutzung des Online Config Tools in dieser Reihenfolge vorgeschrieben wird. Alle weiteren Werkzeuge können in einer freien Reihenfolge bedient werden. Der in der Vision erwähnte Autostart-Vorgang[9] dient lediglich der Arbeitserleichterung des Anwenders. Ihm bleibt es erspart, bestimmte Werkzeuge, mit denen er seine Arbeit standardmäßig vollführen wird, nachträglich von Hand zu starten. Der Autostart kann frei für jeden Anwender eingestellt werden und stellt somit keine Einschränkung oder Behinderung im Arbeitsablauf dar.

Die technische Möglichkeit, innerhalb eines Arbeitsplatzes durchaus Telefone vorzufinden, die an unterschiedlichen Nebenstellenanlagen angeschlossen sind, macht es notwendig, neben der Nebenstelle des Telefons auch noch die Anlage (in der Abbildung 10 „Telenorma“) anzugeben, wenngleich dieser Fall nur in wenigen Ausnahmefällen wirklich eintritt.

[9] Vgl. Windows, Windows 95/NT oder OS/2 bzw. entsprechende Batch-Dateien unter UNIX.

Die nächste Vision wird zeigen, wie der Anwender mit Hilfe des Palettenwerkzeugs den Zugriff auf die Werkzeuge des Online Config Tools angeboten bekommt.

Vision 2: Mit dem Palettenwerkzeug arbeiten

Das Palettenwerkzeug bietet den Zugang zu allen Werkzeugen des Online Config Tools. Für jedes Werkzeug findet der Anwender eine Möglichkeit, dieses über ein Symbol in der Toolbar oder über das Menü zu starten.

Der Anwender hat einen Überblick über alle verfügbaren Werkzeuge.

Um ein Werkzeug zu starten, wählt der Anwender das entsprechende Symbol in der Palette aus und betätigt den Knopf. Im Online Config Tool wird das Werkzeug sofort gestartet und erscheint auf dem Bildschirm.

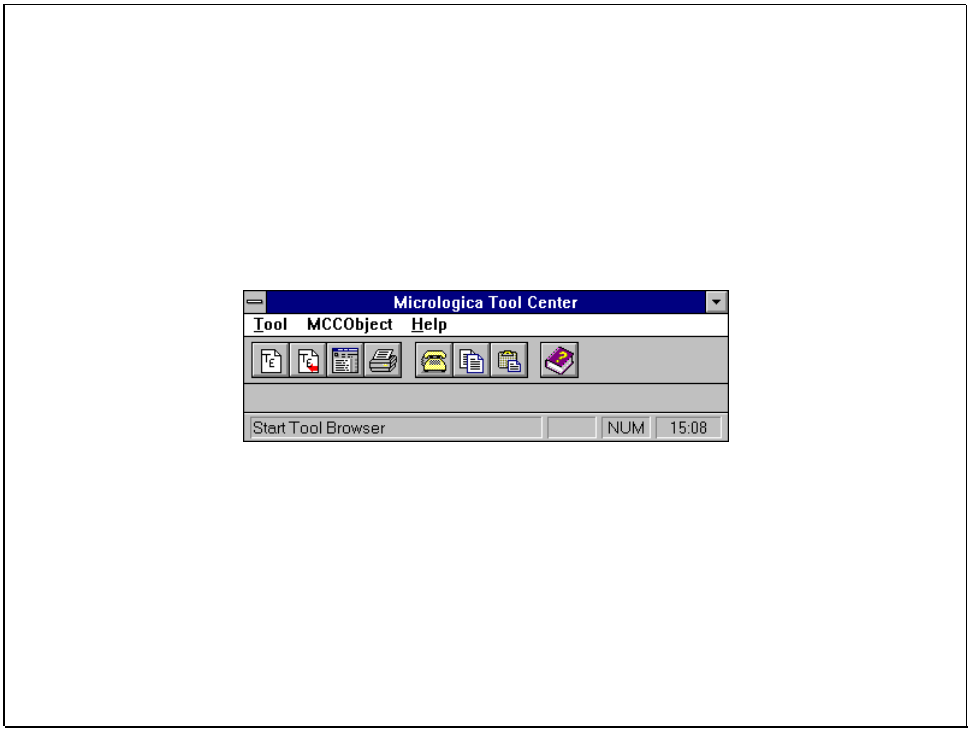

Abbildung 11 Das Palettenwerkzeug

Das Palettenwerkzeug bietet den Zugang zu allen Werkzeugen des Online Config Tools sowohl über Symbole in der Toolbar als auch über Menüpunkte. Im Gegensatz zu „normalen“ Einträgen in der Betriebssystem-Oberfläche, können hier die Symbole und Menüpunkte automatisch durch die Administration des Online Config Tools um neue Werkzeuge erweitert werden. Aus diesem Grund wurde das Palettenwerkzeug überhaupt in die Konzeption des Online Config Tools aufgenommen. Alle Programme lassen sich auch weiterhin über die standardmäßigen Wege des Betriebssystems starten. Sie sind darüber in gleicher Weise in das Anwendungssystem integriert.

Weil bei diesem Anwendungssystem die Konfiguration von Call Centern im Mittelpunkt steht, zeigt die nächste Vision, wie der Administrator sich ein Bild über das zu konfigurierende System macht.

Vision 3: Eine Konfiguration auswählen

Nach dem Starten des Konfigurations-Auflisters über die Palette erscheint daraufhin eine Auswahlbox mit einer Liste von bekannten Konfigurationen. Zu jedem Call Center kann es eine Reihe von Konfigurationen geben. Der Administrator wählt in der Liste eine bereits verfügbare Konfiguration aus und bestätigt den Dialog.

Alle in dieser Konfiguration eingetragenen MCC-Systeme werden daraufhin in einzelnen Fenstern dargestellt.

Die Anzeige des Konfigurations-Auflisters wird nach der Auswahl einer Konfiguration um eine Liste der Server, die an dieser Konfiguration beteiligt sind, ergänzt. In dieser Darstellungsform kann der Administrator einzelne Server hinzufügen oder aus der Konfiguration löschen. Mit einem Doppelklick auf einen Eintrag der Liste wird das Anzeigewerkzeug gestartet oder in den Vordergrund geholt, wenn es bereits läuft.

Es erscheint zusätzlich ein Fenster „Schablonen", in dem alle möglichen Objekte eines MCC-Systems aufgelistet sind.

Fachlich motiviert können für ein Call Center mehrere Konfigurationen existieren, wovon eine Konfiguration zur Zeit aktiv ist. Wird ein Call Center saisonal betrieben, kann es notwendig sein, beim Saisonwechsel auch auf eine vollständig andere Konfiguration zurückzugreifen. Dies wird dadurch unterstützt, daß Konfigurationen abgelegt und später wieder geladen werden können. Das hier beschriebene Werkzeug bietet alle gespeicherten Konfigurationen zur Auswahl an. Aus der Anwendungspraxis ergibt sich, daß zumeist nur mit aktiven Konfiguration gearbeitet wird.

Der Umgang mit dem Schablonenwerkzeug wird später im Zusammenhang beschrieben.

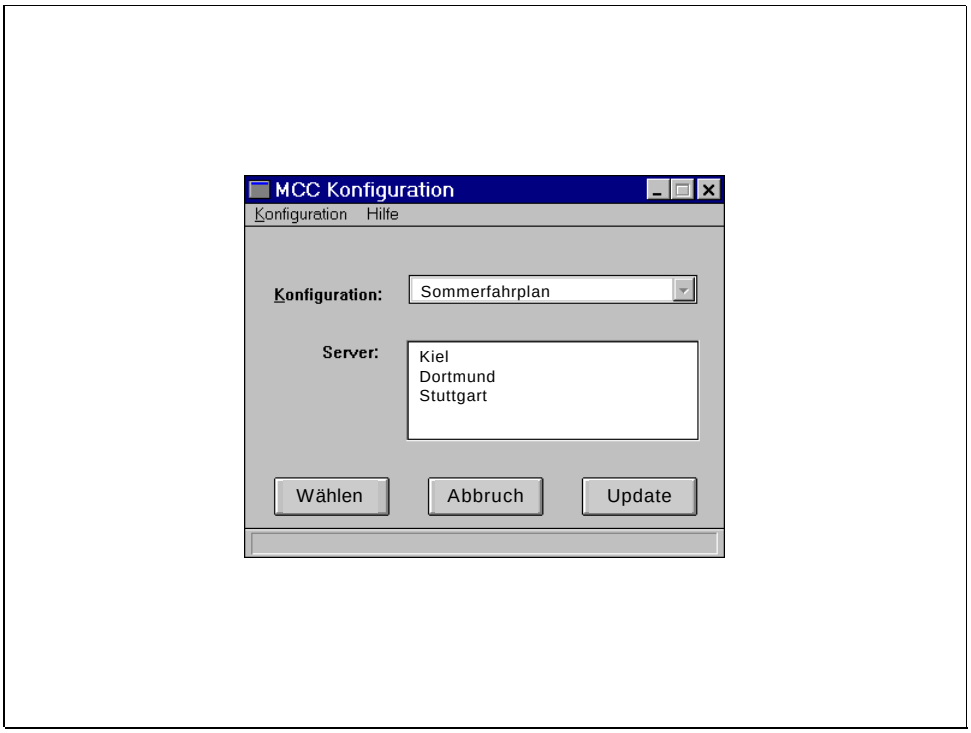

Abbildung 12 Das Werkzeug zum Auswählen der Konfiguration

Die Abbildung zeigt das Werkzeug zum Auswählen der Konfiguration. Dafür bietet die Dropdown Liste eine Auswahl von Konfigurationen an. Je nachdem, welche ausgewählt wurde, sind in der mit „Server“ bezeichneten Listbox die entsprechenden Server namentlich aufgeführt.

Die nächste Vision zeigt den Umgang mit den Werkzeugen zum Betrachten der einzelnen Call Center Server. Sie sind das zentrale Werkzeug zum Erstellen und Anpassen einer bestehenden Konfiguration.

Vision 4: Die verschiedenen Server betrachten

Aufgrund der Auswahl einer Konfiguration wird für jeden Call Center Server ein Exemplar des Objekt Auflister Werkzeugs gestartet. In der Titelzeile des Werkzeugs wird der Name des Call Center Servers angegeben. Im oberen Anzeigebereich findet sich neben dem Namen auch die technisch motivierte Prozeß-ID des Servers.

In einer Listbox zeigt das Werkzeug alle Agenten, die in diesem Server konfiguriert sind, an. Zu jedem Agenten können verschiedene Attribute in der Liste angezeigt werden. Der Administrator kann zwischen der technischen Agent-ID, dem Namen und dem Typ wählen.

Um die anzuzeigenden Attribute oder die Sortierung der Agenten Liste einzustellen, wird das Werkzeugfenster mit Hilfe des Knopfes „Erweitert“ aufgeblättert und bietet Schaltflächen für die Attribute und die Sortierung.

Mit Hilfe der rechts angebotenen Knöpfe können die ausgewählten Agenten in der Liste angesprochen werden. Mit „Anzeigen“ wird jeweils ein Fenster geöffnet, das die detaillierten Informationen zu

den Agenten präsentiert. Mit „Bearbeiten“ wird ein Editierwerkzeug geöffnet, daß es dem Administrator erlaubt, die einzelnen Attribute eines Agenten zu verändern. Und mit Löschen können die selektierten Agenten aus dem Server gelöscht werden.

Der Doppelklick auf einen Eintrag der Liste bedeutet das gleiche wie das Auswählen des Agenten und das Betätigen des Knopfes „Anzeigen“. Für alle selektierten Agenten steht zusätzlich auch ein Kontextmenü zur Verfügung, mit dem dieselben Funktionen aufgerufen werden können. Auch die Sortierung und die angezeigten Attribute können über das Kontextmenü eingestellt werden.

Alle Funktionen stehen ebenfalls über das Menü des Werkzeugs zur Verfügung, in dem auch die Tastenkombinationen zur tastaturgesteuerten Bedienung des Werkzeugs angezeigt werden.

In der Statuszeile des Fensters werden zusätzliche Informationen zur aktuelle ausgewählten Zahl der Agenten ausgegeben, wenn eine Auswahl stattgefunden hat. Damit hat der Administrator auch einen Überblick über die insgesamt selektierten Agenten, wenn die Auswahl über die anzeigbare Menge hinausgeht.

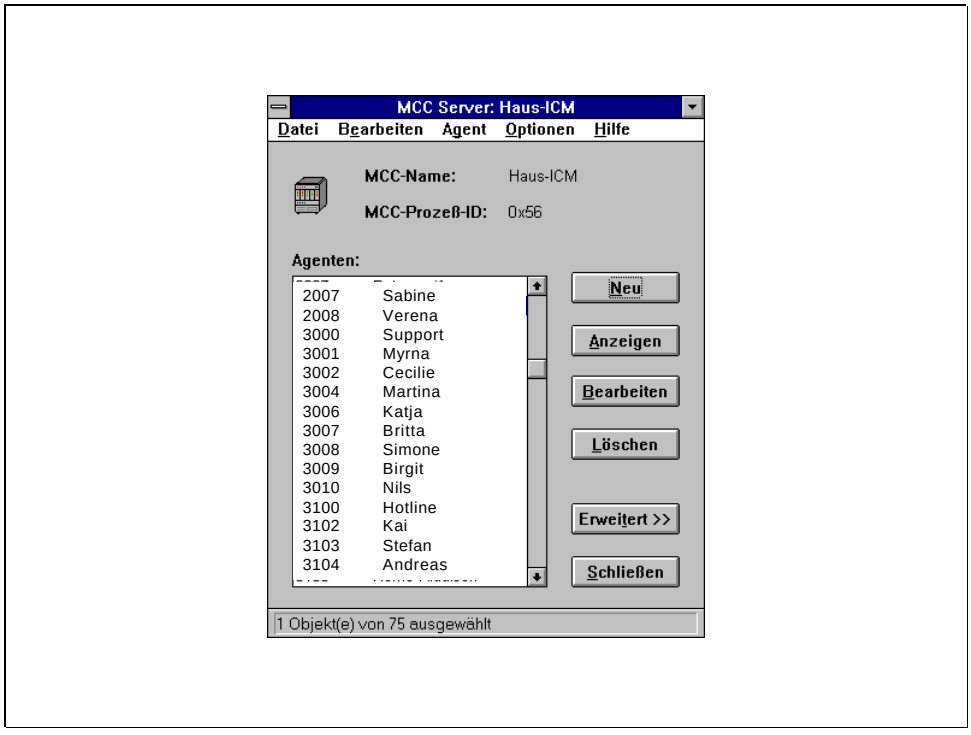

Abbildung 13 Objekte der Server ansehen

Für die Arbeit eines Administrators ist es nun hauptsächlich notwendig, die Einstellungen der relevanten Agenten und Gruppen zu betrachten und gleichzeitig die Möglichkeit zu besitzen, kleinere Änderungen sofort durchführen zu können.

Die folgende Vision stellt dar, wie der Administrator die Konfiguration einer Gruppe betrachtet.

Vision 5: Betrachten einer Gruppenkonfiguration

Um eine Gruppenkonfiguration zu betrachten, wählt der Administrator aus der MCC-Objekt Liste eine Gruppe aus und betätigt den Knopf „Anzeigen". Alternativ kann der Anwender im Menü „Agent" eine Gruppe aus der Liste auswählen.

Für jede ausgewählte Gruppe erscheint genau ein Fenster. Wählt der Anwender eine bereits geöffnete Gruppe erneut, wird das bereits offene Fenster in den Vordergrund geholt bzw. wiederhergestellt.

Die Darstellung einer Gruppe erfolgt wie bereits im Gruppenmonitor vorgelegt. Für jeden freien Platz einer Gruppe wird ein rechteckiger Rahmen dargestellt, in dem der Name des Agenten eingetragen wird, wenn sich ein Agent an diesem Platz mit der Gruppe verbunden hat. In dieser Darstellung werden Gruppen und Agenten nur durch die Schriftart unterschieden. Der Zustand jedes Agenten (jeder Gruppe) wird mit den von Micrologica definierten Zustandsfarben angezeigt.

Die Detailwerte einer Gruppe kann der Administrator in diesem Fenster darstellen, indem er eine horizontale Trennleiste in der rechten oberen Ecke des Gruppenfenster anfaßt und nach unten zieht. Der innere Fensterbereich teilt sich daraufhin so, daß im oberen Bereich die Detailwerte und im unteren Bereich die gewohnten Einträge der Gruppe dargestellt werden.

Die Anzeige der Zustandsfarben kann für jede Gruppe im Systemmenü ein- bzw. ausgeschaltet werden.

Zur genaueren Darstellung der Typen von Agenten auf den einzelnen Plätzen der Gruppen kann eine Icon-Darstellung hinzugeschaltet werden. Dies geschieht über das Systemmenü. Dann wird jeder Platz einer Gruppe nicht nur durch den Namen, sondern auch mit Hilfe eines Icons beschrieben. Für jeden Typ von Agenten gibt es ein spezielles Icon.

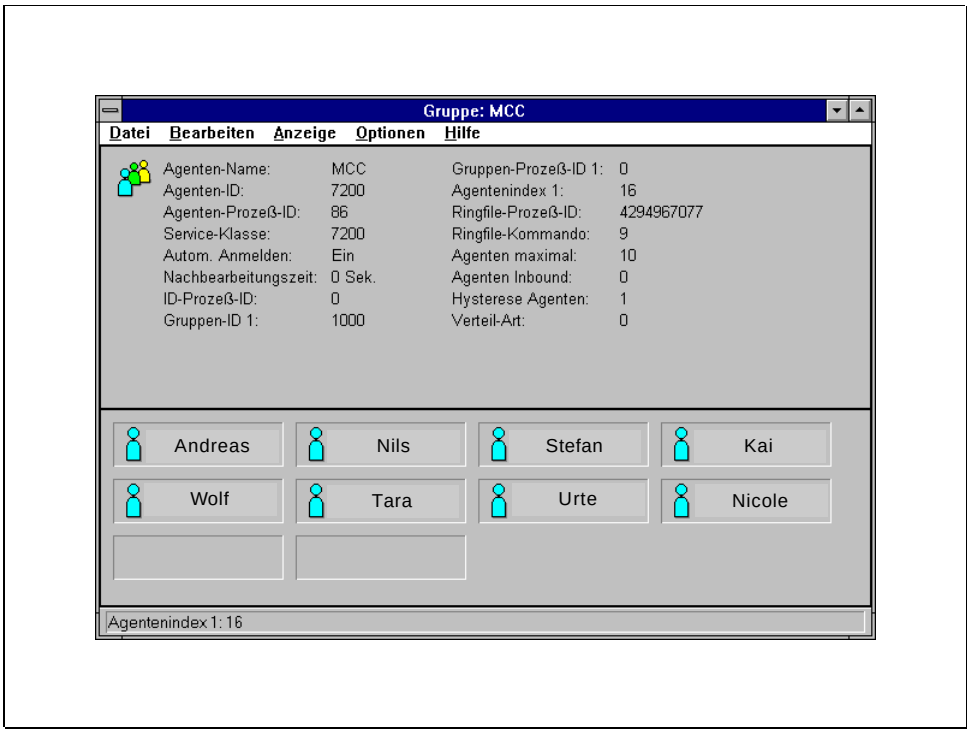

Abbildung 14 Einstellungen einer Gruppe einsehen

Die Abbildung 14 zeigt das Fenster des Gruppenwerkzeugs. Die vorangegangene Vision konnte nicht alle Aspekte dieses Werkzeugs beleuchten. Die Philosophie dieser Werkzeugfamilie, der Werkzeuge zum Anzeigen und Bearbeiten von Agenten, läßt sich in drei Sätzen beschreiben: Durch ein Symbol (hier in der linken oberen Ecke) ist der Agent (in diesem Fall eine Gruppe) für den Anwender wiederverwendbar, indem er das Symbol mit der Maus auf ein anderes Werkzeug zieht. Die Anzeige aller Parameter eines Agenten erfolgt aufgelistet im oberen Bereich des Fensters. Die komplexen zusätzlichen Werte der Agenten werden im unteren Teil des Werkzeugfensters dargestellt.

Bis zu dieser Stelle ist der Umgang mit dem System eine Vorbereitung für tägliche Arbeiten an der Call Center Konfiguration. Die folgende Vision soll einen typischen Fall darstellen, der immer dann auftritt, wenn ein Mitarbeiter die Firma verläßt und ein neuer Mitarbeiter seinen Platz einnimmt:

Vision 6: Die Konfiguration eines Agenten ändern

Um die Konfiguration eines Agenten zu ändern, wählt der Administrator aus der Liste der Agenten im Objekt-Auflister den gewünschen Agenten aus und betätigt den Knopf „Bearbeiten“. Ein Editierwerkzeug wird geöffnet, in dem die Parameter dieses Agenten angezeigt werden.

Der Administrator geht mit dem Mauspfeil auf den Wert, den er ändern möchte und klickt einmal: beispielsweise den Namen des Agenten. In das Eingabefeld kann er nun einen anderen Wert eintragen. Dies wiederholt er für alle Werte, die geändert werden sollen.

Sind alle gewünschten Änderungen durchgeführt, betätigt der Administrator den Knopf „Ändern“, der den Änderungswunsch an den Server sendet. Wurde die Änderung durchgeführt, wechselt das Fenster in den reinen Darstellungsmodus dieses Agenten und zeigt so die neuen Werte an, die ab diesem Zeitpunkt für den Agenten Gültigkeit haben.

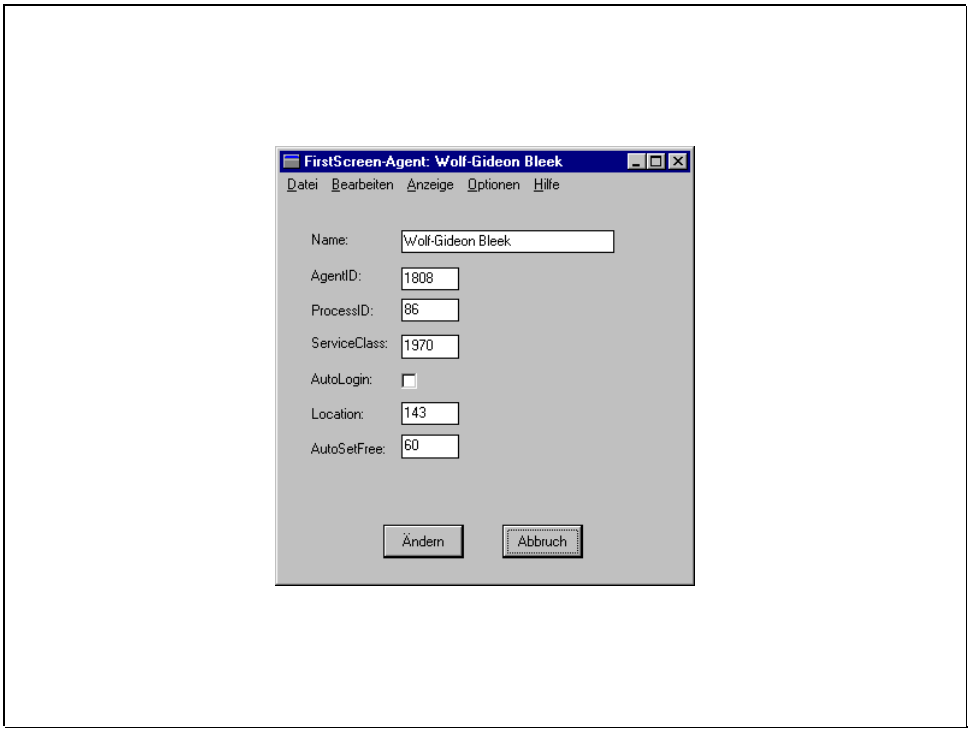

Abbildung 15 Daten eines bestehenden Agenten ändern

Das zweite Fallbeispiel soll auf eine komplexere Tätigkeit beim Konfigurieren eines Call Centers eingehen. Für ein neues Tätigkeitsfeld wird eine korrespondierende Gruppe benötigt, in die neue und alte Agenten eingeordnet werden können. Die folgenden Szenarios zeigen, wir der Administrator dazu vorgeht:

Vision 7: Den Schablonen-Auflister zum Einrichten einer Gruppe verwenden

Der Schablonen-Auflister kann über die Palette gestartet werden oder wird automatisch gestartet, wenn mit dem Werkzeug Konfigurations-Auflister eine Konfiguration ausgewählt wurde.

Das Fenster des Werkzeugs zeigt eine Liste aller verfügbaren Schablonen mit Namen an. Die Schablonen können für unterschiedliche Typen von Objekten im Systeme stehen.

Durch Doppelklick auf einen Eintrag in der Liste wird ein neues Objekt erzeugt. Der Typ des Objektes hängt von der ausgewählten Zeile ab. Es gibt standardmäßig Einträge für neue Agenten, neue Gruppen und neue MCC-Systeme. Anstelle eines Doppelklicks kann der Anwender auch den Knopf „ Nutzen" verwenden.

Durch diese Aktivität wird das Werkzeug zum Erstellen eines neuen Objektes gestartet. In dem Fenster des Werkzeugs sind alle Attribute des Objektes aufgelistet und mit den vorbelegten Werten der Schablone versehen.

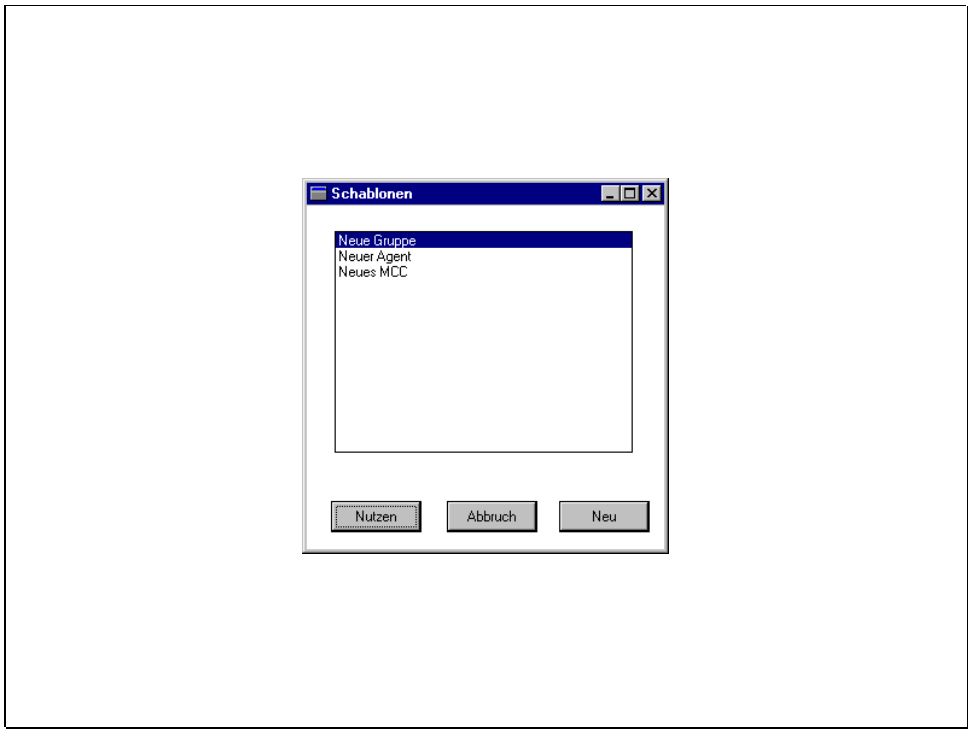

Abbildung 16 Der Schablonen-Auflister

Das Konzept des Schablonen-Auflisters wurde in das Online Config Tool eingeführt, um die Arbeit des Administrators deutlich zu erleichtern. Neben dieser Grundfunktion die Schablone als Vorlage für neue Agenten einzusetzen bietet der Auflister die Möglichkeit, Schablonen in der Art zu definieren, daß bereits existierende Agenten (gleich welchen Typs) in die Auflistung geschoben werden. Dadurch kann der Administrator bereits durchgeführte Konfigurationen einzelner Agenten in einer Form abspeichern, daß sie für später wiederverwendet werden können.

Die nächste Vision geht auf den Fall ein, in dem eine Gruppe mit neuen Agenten eingerichtet werden muß. Dieser Fall tritt immer dann ein, wenn das Call Center eine neue Aufgabe übernehmen soll, bzw. wenn sich die Strukturierung der Abteilung ändert.

Vision 8: Neue Gruppe mit neuen Agenten einrichten

Zum Einrichten einer neuen Gruppe mit neuen Agenten wird der Administrator zuerst eine leere Gruppe anlegen. Dazu wird er mit dem Schablonen-Auflister und einem Doppelklick auf den Eintrag „neue Gruppe“ das Gruppenwerkzeug aufrufen:

Zur Spezifikation der Gruppe trägt er einen Namen, die Gruppen-ID und die Anzahl der Plätze, die diese Gruppe haben wird, in das Werkzeug ein. Durch Betätigen des Knopfes „Erzeugen“ wird dem Server der entsprechende Auftrag erteilt. Das Werkzeugfenster

wechselt in die Anzeige der Gruppenparameter, wenn diese Gruppe erfolgreich erzeugt werden konnte.

Nun wiederholt der Administrator den Vorgang für einen Agenten. Das Werkzeug zum Erzeugen eines neuen Agenten erscheint und die entsprechenden Werte werden eingetragen. Dazu gehört der Name, die Agenten-ID und die Service-Klasse des Agenten. Weitere Parameter können eingestellt werden, müssen es aber nicht, da sie mit sinnvollen Standardwerten vorbelegt werden. Auch hier wechselt das Fenster nach dem Betätigen des Knopfes „Erzeugen“ in die Anzeige, wenn der Server den Auftrag erfolgreich ausführen konnte.

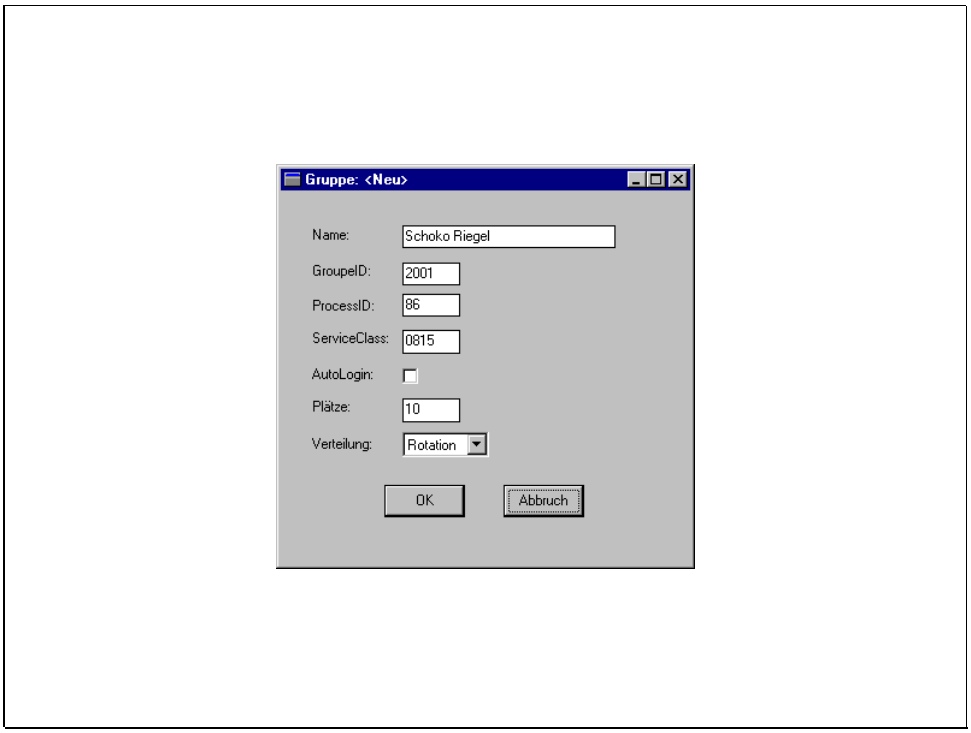

Abbildung 17 Eine neue Gruppe spezifizieren

Nun kann der Administrator den soeben erzeugten Agenten in der vorher angelegten Gruppe eintragen. Dazu zieht er mit der Maus (Drag and Drop) das Sinnbild des Agenten (Mensch-Ärger-Dich-Nicht Figur) auf einen freien Platz der Gruppe (im unteren Bereich des Fensters) und läßt es dort los. Damit bekommt der Server den Auftrag, diesen Agenten in der ausgewählten Gruppe einzutragen. Als Erfolgsmeldung der gelungenen Konfiguration wird das Sinnbild nun mit Namen des Agenten in dem Gruppenfeld angezeigt.

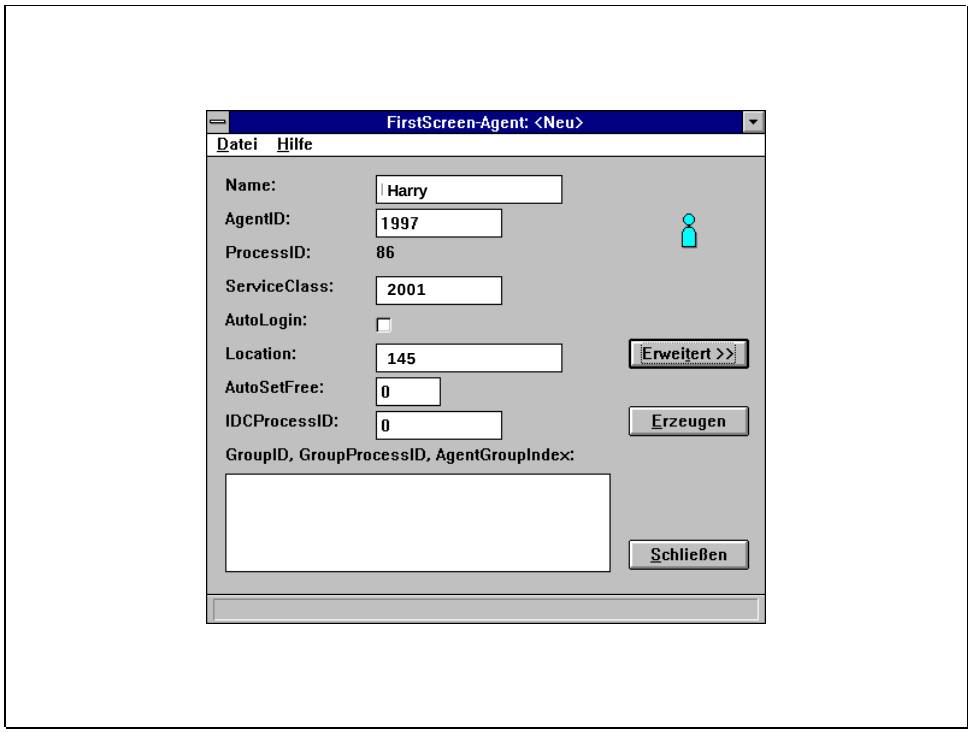

Abbildung 18 Einen neuen Agenten spezifizieren

Die hier mit den Visionen vorgestellten Tätigkeiten eines Administrators des Call Centers wurden lediglich exemplarisch ausgewählt. Sie gehören zu den Basistätigkeiten seiner Arbeit. Dazu kommen eine Reihe von weiteren Aufgaben, die hier den Rahmen der Arbeit sprengen würden. Es ist jedoch wichtig anzumerken, daß die hier vorgestellte Reihenfolge der Tätigkeiten nicht zwingend vorgegeben ist. Sie bietet sich im Rahmen einer Kurzvorstellung der Werkzeuge an. Im täglichen Einsatz des Online Config Tools und in bestimmten Einsatzformen sind durchaus andere Reihenfolgen denkbar. Zum Beispiel beim Konfigurieren eines neuen Systems wird man zuerst die entsprechenden Server einrichten müssen, um dann als nächstes Gruppen und Agenten neu zu erzeugen. Erst dann wird man die Möglichkeit haben, die Agenten in den unterschiedlichen Gruppen einzusetzen.

4.2 Integrationspotentiale

Eine wesentliche Anforderung bei der Erstellung dieses Online Config Tools, war die Möglichkeit einer Integration der Anwendung in bestehende Systeme. Hierbei ergeben sich insbesondere zwei Ebenen, die zentrale Beachtung finden müssen. Das ist einerseits die technische Integration der Benutzer- und Rechteverwaltung in bereits etablierte Konzepte. Netzwerk-Betriebssysteme geben hier durch Benutzerverwaltung und Rechteverwaltung Standards vor, in die neue Systeme eingebunden werden müssen. Obwohl bei der Implementation des hier vorgestellten Anwendungssystems noch eine proprietäre Umsetzung gewählt wurde, kann es in Zukunft notwendig sein, die Potentiale der bestehenden Betriebssysteme weiterzuverwenden. Andererseits ist das vorliegende Call Center System eine erweiterbare Anwendung. Zusätzliche Komponenten, wie z.B. der PowerDialler, können später ergänzt werden und machen es notwendig, die

Werkzeuge zur Bedienung dieser Komponente in das bestehende Online Config Tool zu integrieren.

4.3 Zusammenfassung

Aus den Visionen haben wir erkennen können, welche Werkzeuge für das beschriebene Anwendungssystem benötigt werden. Hier folgt noch einmal eine Aufstellung der Werkzeuge:

- Anmeldewerkzeug
- Palettenwerkzeug
- Konfigurations-Auflister
- Schablonen-Auflister
- Objekt-Auflister
- Gruppenwerkzeug
- Agentenwerkzeug
- Werkzeug zum Erzeugen einer neuen Gruppe
- Werkzeug zum Erzeugen eines neuen Agenten

Neben diesen Werkzeugen ist es notwendig, eine Reihe von Automaten in das Anwendungssystem zu integrieren. Das folgende Kapitel zeigt, mit welchen Konstruktionsprinzipien wir bisher Anwendungssysteme gebaut haben.

5 Konstruktion von Anwendungssystemen

Dieses Kapitel beschreibt, wie man von den identifizierten Werkzeugen, Materialien und Automaten zur Konstruktion konkreter Anwendungssysteme gelangt. Die Begriffe Werkzeug, Material und Automaten helfen auf der Metaphernebene, reale Arbeitssituationen zu beschreiben. Einzelne Bestandteile werden identifiziert. Bei der Umsetzung der hierbei gewonnenen Erkenntnisse wird eine Konstruktionstechnik benötigt, die gezielt auf diese Gegebenheiten eingeht.

Konkret müssen Werkzeuge in Objekte umgesetzt werden, die sowohl die äußerliche Erscheinung als auch deren intendierte Funktion widerspiegeln. Außerdem müssen Konventionen für das Zusammenwirken der Objekte entwickelt werden, die im Sinne von Konstruktionsvorschriften dem Programmierer Leitlinien an die Hand geben.

Dieses Kapitel untergliedert sich in drei Abschnitte. In der Einleitung wird der Musterbegriff eingeführt und in den Kontext der Werkzeugkonstruktion gesetzt.

Der zweite Abschnitt zeigt den technischen Entwurf von Anwendungssystemen. Die Bestandteile eines Anwendungssystems werden in Beziehung zueinander gesetzt. Ihre Aufgaben werden voneinander abgegrenzt.

Im dritten Abschnitt wird die Rolle von Automaten näher untersucht und in den Kontext des hier zentralen Einsatzfeldes gerückt. Automaten werden unter der Perspektive von realen technischen Maschinen diskutiert.

Eine Zusammenfassung rundet das Kapitel ab.

5.1 Einleitung

Mit den Metaphern des WAM Repertoires war es in den Visionen möglich, die Handlungsabsichten auszuformulieren. Um allerdings zu einem konkreten Software-Entwurf zu gelangen, ist es notwendig das System detaillierter zu beschreiben. Es wird ein Ausdrucksmittel notwendig, mit dem die Konstruktionsabsichten beschrieben werden können.

Zur Konstruktion von Anwendungssystemen und speziell zur Konstruktion von Werkzeugen und Automaten werden deshalb Entwurfsmuster verwendet. Muster, engl. Pattern, haben sich im Rahmen der objektorientierten Anwendungsentwicklung durchgesetzt.

Riehle definiert den Begriff des Musters folgendermaßen:

> *Ein Muster ist eine in einem bestimmten Kontext erkennbare Form. Es dient als Vorlage zum Erkennen, Vergleichen und Erzeugen von Musterexemplaren. Ein Muster ist die Essenz aus Erfahrung und Analyse immer wiederkehrender Situationen. Es besitzt eine innere Struktur und Dynamik. (vgl. [Rie95], Seite 20)*

Entwurfsmuster sind konkrete Muster zum Entwerfen von Software-Systemen. Gamma et al. (vgl. [GHJ+]) haben mit ihrem Buch eine allgemeine anerkannte Kollektion von Entwurfsmustern vorgelegt, die sich in vielen Anwendungsfeldern einsetzen lassen.

Sie formulieren die Anforderungen an ein Entwurfsmuster folgendermaßen:

> *A design pattern names, abstracts, and identifies the key aspects of a common design structure that make it useful for creating a reusable object-oriented design. The design pattern identifies the participating classes and instances, their roles and collaborations, and distribution of responsibilities. (vgl. [GHJ+95], Seite 3)*

Demnach wird durch die Einführung von Entwurfsmustern eine Abstraktionsebene oberhalb der konkreten Programmtexte eingezogen, die es erlaubt, die Metaphern des WAM Repertoires auf der Konstruktionsebene zu beschreiben.

Entwurfsmuster, wie wir sie verwenden, können grob in drei Kategorien unterteilt werden. Es gibt Muster zur Erzeugung von Objekten, Muster, die das Verhalten von Objekten beschreiben, und Muster zur Strukturierung von Objekten.

Auf der Ebene von Mustern können Absichten beschrieben werden, die auf der Visionsebene nicht oder nur schlecht beschreibbar waren. So kann z.B. mit dem Muster Singleton ausgedrückt werden, daß es nur höchstens ein Objekt einer Klasse geben kann. Dieser Aspekt wird auf der Visionsebene nicht berücksichtigt. Auf der Musterebene kann definiert werden, über welches Protokoll zwei Objekte miteinander verbunden werden. Das Observer-Muster beschreibt dafür einen geeigneten Mechanismus. Aspekte der Kopplung können auf der Visionsebene nicht beschrieben werden. Es ist allerdings auch gar nicht die Absicht. Auf der Musterebene werden die Begriffe der Visionsebene wieder aufgegriffen und weiterverwendet.

5.2 Technischer Entwurf von Anwendungssystemen

In diesem Abschnitt soll der prinzipielle Aufbau eines Anwendungssystems anhand von Aufgabenverteilungen erläutert werden. Dafür wird ein Entwurfsmuster angegeben, daß die Zusammenstellung von Werkzeugen und Automaten zu einem Anwendungssystem umsetzen hilft.

Die von dem Entwurfsmuster verwendeten Komponenten werden hier auf der Ebene der Szenarios und Visionen betrachtet. Erst im darauffolgenden Abschnitt werden auch Werkzeuge und Automaten in feiner granulare Bestandteile untergliedert.

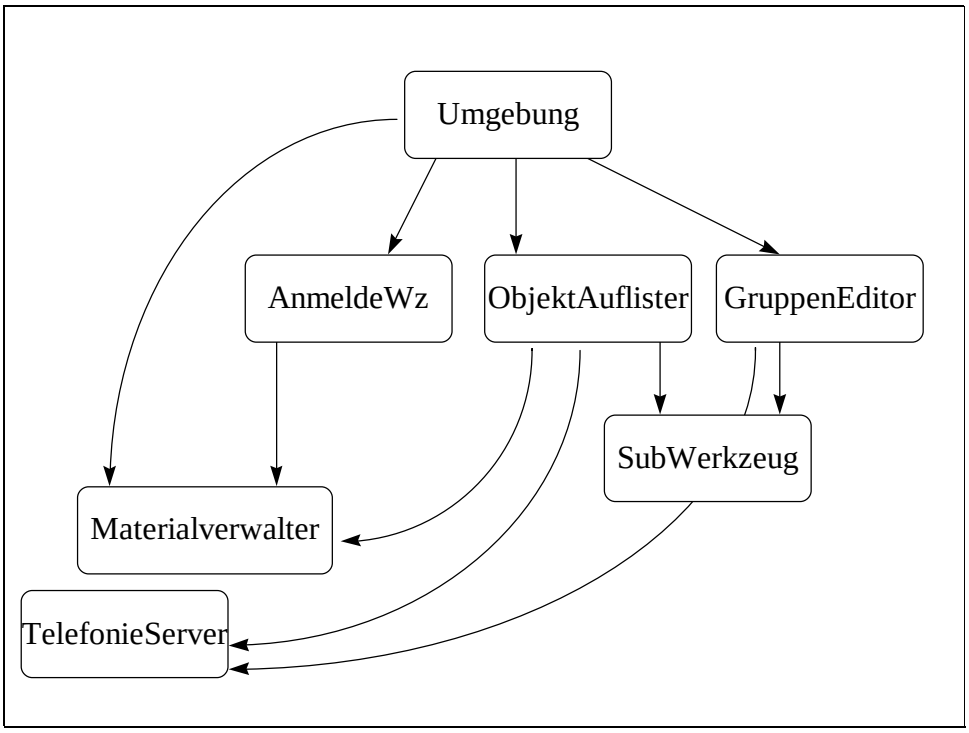

Abbildung 19 Die Komponenten eines Anwendungssystems

Die Abbildung 19 zeigt Komponenten eines Anwendungssystems auf der Granularität von Werkzeugen bzw. Automaten. Die Pfeile geben ein Beispiel für die Benutzung der Komponenten an. Ein Pfeil zeigt immer auf die Komponente, die vom Ausgangspunkt aus benutzt wird. Das bedeutet, das die Komponente am Ursprung des Pfeils die andere Komponente genau kennt. In der obersten Ebene der Abbildung steht die Umgebung. Sie erzeugt neue Werkzeuge und bindet diese in einen Kontext, den wir als Arbeitsplatz bezeichnen. Werkzeuge finden sich in der zweiten und dritten Ebene. Werkzeuge können gemeinsame Subwerkzeuge benutzen. Im linken unteren Bereich der Abbildung finden sich Automaten des Anwendungssystems. Der Materialverwalter stellt einen Automaten zur persistenten Lagerung und Verwaltung von Materialien dar. Der andere Automat wurde mit „Telefonie Server“ bezeichnet. Dieser Automat bildet die Schnittstelle zwischen realem technischen Gerät und den Werkzeugen, die sich seiner bedienen.

5.2.1 Trennung von Funktion und Interaktion

In diesem Abschnitt wird das grundlegende Prinzip der Trennung von Funktion und Interaktion eingeführt. Im Gegensatz zu Automaten haben Werkzeuge eine Interaktionskomponente, die es dem Anwender ermöglicht, mit dem Werkzeug zu arbeiten. Die Präsentation ist dafür zuständig, Werkzeugzustände angemessen zu visualisieren und Aktionen des Benutzers entgegenzunehmen. Wegen dieser wechselseitigen Funktion wurde die Bezeichnung „Interaktionskomponente“ gewählt.

Durch die Trennung von Funktion und Interaktion kann eine notwendige softwaretechnische Unterscheidung eingeführt werden. Aus der gewachsenen Erkenntnis, daß sich die Präsentation und damit die Interaktion wiederholt ändert, die Funktion aber relativ stabil bleibt, ist dies die logische Konsequenz. Änderungen an der Interaktionskomponente haben keinen oder nur geringen Einfluß auf die

Funktionskomponente. Es kann diverse Interaktionskomponenten zu einer Funktionskomponente geben. Zum Beispiel kann es eine textbasierte Interaktion geben und eine fensterbasierte.

Nachfolgende Abbildung zeigt die Unterteilung eines Werkzeugs:

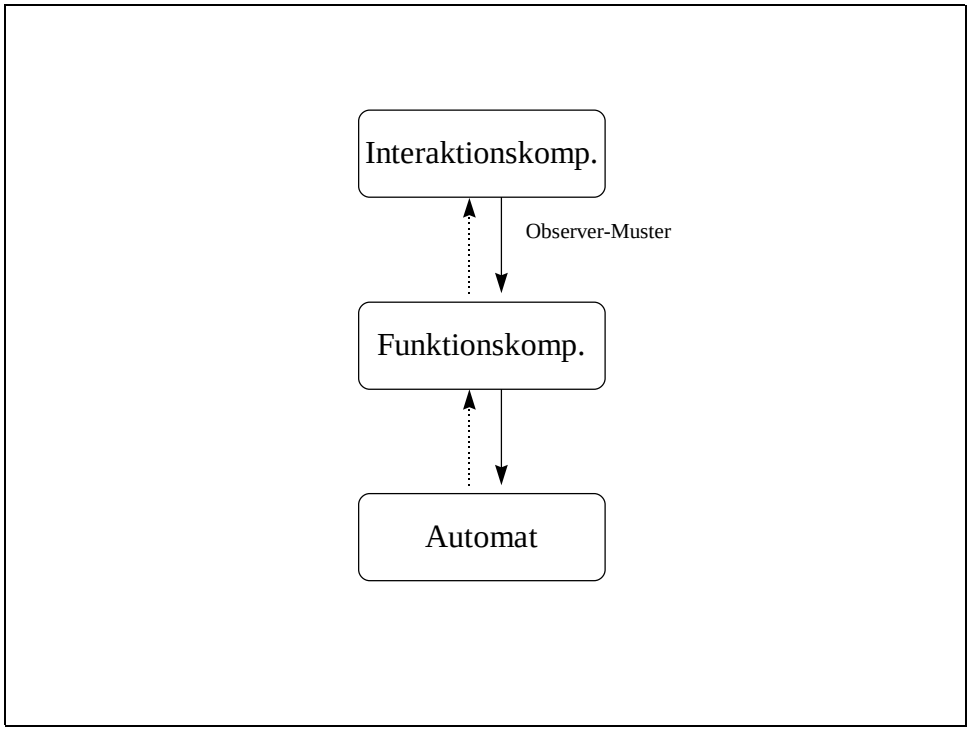

Abbildung 20 Beispiel für die Trennung von Funktion und Interaktion

Das Beispiel zeigt, wie ein Werkzeug aufgebaut ist, welches mit einem Automaten zusammenarbeitet. Das Werkzeug wird durch die beiden oberen Objekte verkörpert. Es ist zweigeteilt, um die Trennung von Funktion und Interaktion umzusetzen. Das obere Objekt beinhaltet die gesamte Funktionalität, die notwendig ist, um den Zustand des Werkzeugs darzustellen und Eingaben des Anwenders entgegen zu nehmen. Die Funktionskomponente stellt fachliche Funktionen zur Verfügung, die Daten manipulieren und in diesem Sinne auf Materialien arbeiten. In diesem Beispiel ist die Funktionskomponente mit einem Automaten verbunden.

Die Trennung der Funktion und Interaktion macht es andererseits aber auch notwendig, beide Objekte miteinander in der Art zu koppeln, daß diese ihre notwendigen Daten austauschen und Funktionen aufrufen können. Dazu wird ein Entwurfsmuster verwendet, daß die Asymmetrie in der Beziehung ausdrückt. Das Observer-Muster erlaubt die Kopplung zweier Objekte, wobei der Beobachter, hier die Interaktion, die Schnittstelle des Beobachteten, hier die Funktion, genau kennt. In umgekehrter Richtung kennen sich die Objekte nicht. Die unterschiedlichen Pfeile drücken diese asymmetrische Beziehung von Funktion und Interaktion aus. Ein Pfeil mit durchgezogener Linie steht für vollständige Kenntnis der Schnittstelle. Ein Pfeil mit gestrichelter Linie bedeutet, daß nur ein Teil der Schnittstelle verwendet wird.

5.2.2 Werkzeugkomposition

Bereits einfache Werkzeuge lassen sich nicht mehr oder nur unbefriedigend nach obigem Schema bauen, weil ihre Struktur weitaus komplexer ist, als es mit zwei Objekten beschrieben werden könnte. Werkzeuge werden dann in Subwerkzeuge unterteilt, die in unterschiedlichen Kontexten wiederverwendet werden können.

Auch für diese Konstruktionstechnik gibt es ein Entwurfsmuster, welches die Zusammenstellung von Subwerkzeugen zu Werkzeugen beschreibt. Die nachfolgende Abbildung zeigt ein Beispiel für die Konstruktion eines Werkzeugs, welches zwei unterschiedliche Subwerkzeuge benutzt.

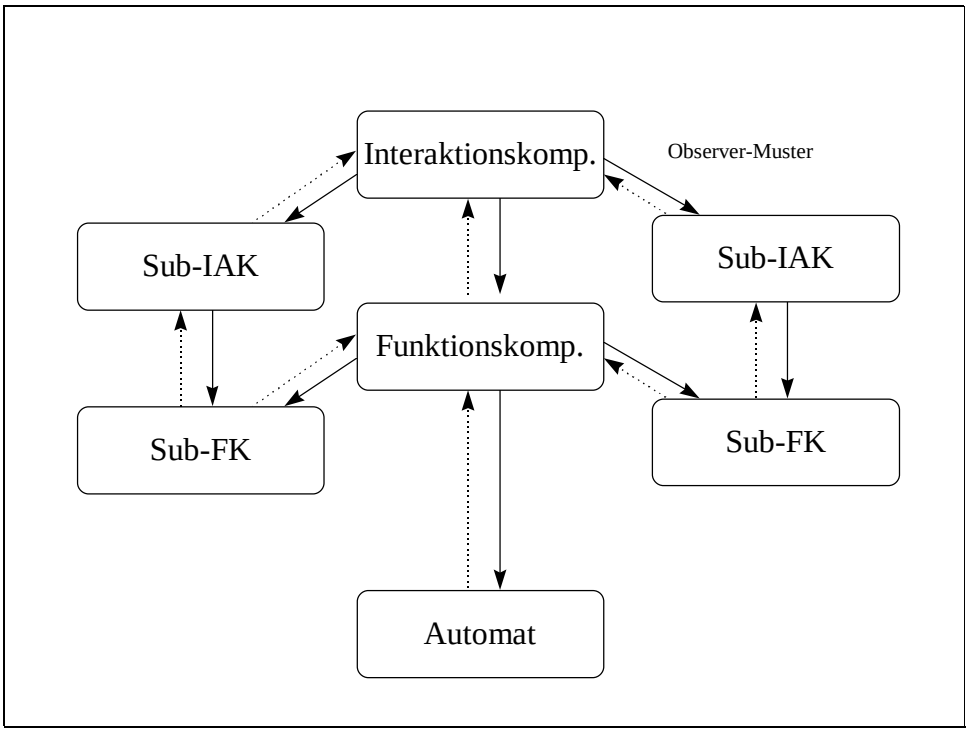

Abbildung 21 Trennung von Funktion und Interaktion mit Subkomponenten

Funktionskomponenten kennen ihre Subwerkzeuge und erzeugen deshalb bei Bedarf die entsprechenden Sub-FKs. Interaktionskomponenten kennen ebenfalls ihre Subinteraktionskomponenten und können auch diese bei Bedarf erzeugen. Entscheidend ist nun die Koordination dieses Vorgangs. Eine Möglichkeit besteht darin, daß die Funktionskomponente der Interaktion mitteilt, daß sie eine Subkomponente erzeugt hat. Die Interaktion kann dann die entsprechende Subkomponente erzeugen und dieser den Verweis auf die Subfunktionskomponente übergeben.

5.3 Automaten

Automaten werden auf der Metaphernebene für diejenigen Aufgaben verwendet, die routinisierbar sind und wiederholt durchgeführt werden. Auf der Ebene der Entwurfsmuster werden sie durch Objekte repräsentiert, die für klar abgegrenzte

Aufgabenbereiche den Werkzeugen zur Verfügung stehen. Werkzeuge benutzen Automaten, um Aufgaben an diese zu delegieren. Deshalb müssen die Entwurfsmuster einerseits die Delegation angemessen ausdrücken und gleichzeitig ein Verfahren anbieten, wie Automaten die beendete Bearbeitung eines Auftrags an den Auftraggeber signalisieren können.

Die Anbindung von Automaten wurde aus diesem Grund vielfach über das Observer-Muster durchgeführt. Die Verwendung dieses Musters wird im folgenden Kapitel ausführlich diskutiert werden. An dieser Stelle sei lediglich auf die Verwendung dieser speziellen Anbindung eingegangen.

Während das Werkzeug die Schnittstelle des Automaten vollständig kennt, besitzt der Automat kein Wissen darüber, welche Werkzeuge oder Automaten seine Dienstleistungen in Anspruch nehmen. Dem Automaten werden von Seiten der Klienten Aufträge über seine Schnittstelle übergeben. Der Automat benachrichtigt seine Klienten über das Observer-Muster. Benachrichtigungen werden durch den eigenen Auftrag oder durch andere Aufträge ausgelöst. Diese unterschiedlichen Aspekte werden im folgenden Kapitel ausführlich beleuchtet.

Im Kontext von Automaten als Abbild realer technischer Geräte kommen äußere Einflüsse hinzu. Ereignisse der realen Welt führen nun im Automatenmodell zu Ereignissen, die beispielsweise über die Benachrichtigungsschnittstelle an Klienten des Automaten weitergegeben werden. Der Automat wird zu einer aktiven Komponente im Anwendungssystem.

Im dritten Abschnitt wird die Rolle von Automaten näher untersucht und in den Kontext des hier zentralen Einsatzfeldes gerückt. Automaten werden unter der Perspektive von realen technischen Maschinen diskutiert.

5.4 Zusammenfassung

Das WAM Repertoire auf der Metaphernebene ist durch Entwurfsmuster auf der Konstruktionsebene ergänzt. Es gibt Entwurfsmuster für Anwendungssysteme, für Werkzeugkonstruktion und für die Anbindung von Automaten. Die Entwurfsmuster beschreiben die Aufgaben der Komponenten und wie sie miteinander verbunden sind. Entwurfsmuster setzen somit die Gestaltungsprinzipien in einen softwaretechnischen Entwurf um.

6 Konzeptionsmuster für technische Systeme

Dieses Kapitel zeigt die Übertragung der Metaphern der Methode WAM in ein technisch motiviertes Anwendungsfeld. Am Beispiel des Call Centers wurden die Metaphern Automat und Werkzeug dahingehend in Konzeptionsmuster umgesetzt, daß sie nun auch im Bereich technischer Systeme eingesetzt werden können.

Obwohl die Methode WAM Metaphern zur Verfügung stellt, die im Anwendungsfeld technischer Systeme einsetzbar scheinen, paßten die bisher verfügbaren Konzeptionsmuster nicht in das Anwendungsfeld. Es wurden deshalb neue Muster erarbeitet, die hier vorgestellt werden.

Das Kapitel untergliedert sich in vier Abschnitte. Zuerst wird ein Konzeptionsmuster für die Metapher Automat im den Bereich der technischen Geräte vorgestellt. Im zweiten Abschnitt wird dieses Konzept mit der Metapher Sonde vervollständigt. Der dritte Abschnitt wird dazu benutzt, den Begriff des Einstellwerkzeugs zu definieren und tiefer zu untersuchen. Im Anschluß daran findet sich eine Zusammenfassung.

6.1 Der technische Automat

In dem referenzierten Industrieprojekt wurde festgestellt, daß die Verwendung der Konzeptionsmuster von Automaten und Werkzeugen nicht ausreicht, um alle Formen des Umgangs mit Automaten genügend zu beschreiben. Beide Konzeptionsmuster, Automat und Werkzeug, müssen im konkreten Anwendungsfeld präziser gefaßt bzw. redefiniert werden. Dieser Abschnitt geht auf das Muster für den Automaten näher ein.

In dem bisherigen Repertoire sind Automaten dafür vorgesehen, Aufgaben ohne das Zutun von Anwender durchzuführen. Automaten werden für Handlungsaufgaben verwendet, die routiniert durchgeführt werden und deren formalisierbarer Anteil groß genug ist.

Im Gegensatz zu Werkzeugen sind Automaten pro aktiv, d.h. Automaten verändern sich inhärent. Während Werkzeuge nur als Reaktion auf eine Benutzeraktion aktiv werden,

kann dies bei Automaten auf zweierlei Arten geschehen: durch eigene Einstellungen, aber auch durch fremde äußere Einflüsse.

Definition 8. Automat

Ein Automat *ist die Komponente eines Anwendungssystems, die formalisierbare Aufgaben selbsttätig durchführen. Sie besitzt einen Zustand und kann eingestellt werden. Automaten benachrichtigen andere Komponenten des Anwendungssystems über ihre Aktivität, wenn diese sich dafür registrieren.*

Ein Anwendungsbeispiel für den Automaten ist ein Kopierer. Ein Kopierer ist aus der realen Welt bekannt, als ein Gerät, mit dem Dokumente vervielfältigt werden können. In einem Anwendungssystem nimmt ein Kopierer die gleiche Aufgabe wahr. Einstellbar an einem solchen Kopierautomaten ist z.B. die Anzahl der anzufertigenden Kopien. Dem Gerät wird nur das Dokument übergeben, welches vervielfältigt werden soll. Danach verrichtet der Kopierer seine Arbeit selbständig.

Neben dieser übertragenen Anwendung des Automatenbegriffs, findet sich ein weiteres Konzept des Automaten: Automaten, die bestimmte technische Geräte abbilden, stellen eine zusätzliche Ereignisquelle neben dem Benutzer dar. Dafür wird ein neues Modell benötigt, welches das Verhalten gegenüber dem Benutzer konsistent erklärt.

Während ein Kopierer nur auf Benutzeraktionen hin reagiert, kann z.B. eine Telefonanlage (siehe Beispiel des Call Centers) durch äußere Einflüsse, wie z.B. den Anruf eines Kunden, einen Zustandswechsel vollziehen. Diese neue Eigenschaft muß im Modell berücksichtigt werden.

Definition 9. technischer Automat (Konzeptionsmuster)

Technische Automaten *vollziehen Zustände von realen technischen Geräten nach, um sie in ein softwaretechnisches Anwendungssystem einzubinden. Sie benachrichtigen andere Komponenten des Anwendungssystems über Zustandsänderungen und stellen eine zusätzliche Ereignisquelle dar.*

Technische Automaten stellen eine konkrete Metapher für reale technische Geräte, wie z.B. Telefonanlagen oder Maschinen, zur Verfügung. Technische Automaten präzisieren den allgemeinen Begriff Automat aus dem WAM Repertoire.

An einem technischen Automaten können einstellende Methoden aufgerufen werden, um das Verhalten des Automaten zu verändern. Diese Einstellungen können den Zustand des Automaten verändern. Externe Ereignisse führen ebenfalls zur Veränderung des Automatenzustands. Damit kommt zum Fenstersystem eine zusätzliche zweite Ereignisquelle in ein Anwendungssystem.

6.2 Das Sondenkonzept

Zur Vervollständigung der Metapher des technischen Automaten wird eine weitere Metapher benötigt. Die fachlichen Zustände des Automaten lassen sich in zwei Kategorien unterscheiden. Automatenzustände bzw. Eigenschaften, die für einen Automaten charakteristisch sind und sich nicht ändern, sind die erste Kategorie. Sie können direkt am Automaten sondiert werden. In die zweite Kategorie fallen jene

fachlichen Zustände eines Automaten, die sich in regelmäßigen oder unregelmäßigen zeitlichen Abständen ändern. Sie drücken sich in Meßwerten aus.

Klienten eines technischen Automaten sind nicht immer am gesamten Automatenzustand interessiert. In vielen Anwendungsfällen reicht es aus, einen partiellen Zustand des technischen Automaten zu kennen. Deshalb wird die Metapher Sonde (vgl. [Per96]) neu in das Konzept des Automaten eingeführt.

Definition 10. Sonde

2) kleines Meßgerät oder Meßfühler (im einfachsten Fall eine Metallspitze) zur Messung physikalischer Verhältnisse (z.B. Geschwindigkeitsverteilungen einer Strömung, elektr. Potentiale in einer Gasentladung), ohne sie zu stören (vgl. [Fis81], Seite 5594)

Eine Sonde wird an einem Automaten angebracht, um dort Meßwerte eines Typs abzunehmen und sie dann anzubieten. Beim Design eines Automaten wird festgelegt, welche Sonden in seinem Einsatzkontext verfügbar sind. Der Automat kann nach den verfügbaren Sonden gefragt werden. Eine Sonde, oder Meßfühler, ist ein Objekt, welches einen vordefinierten Typ von Meßwert am Automaten abliest und in fachlich motivierten und technisch realisierbaren Zeitabständen liefert.

Definition 11. Sonde

Eine Sonde *ist die Metapher, die genau einen meßbaren Wert eines Automaten bereithält. An der Sonde kann eingestellt werden, mit welcher Regelmäßigkeit dieser Wert aktualisiert wird.*

Die Metapher Sonde erlaubt es nun, den Anwendungsbereich in der Weise zu strukturieren, daß einzelne fachlich relevante Meßwerte eines Automaten repräsentiert werden. Aus dieser Metapher läßt sich ein Konzeptionsmuster ableiten.

Definition 12. Sonde (Konzeptionsmuster)

Eine Sonde besitzt ihren Ort am Automaten und wird durch diesen mit Meßwerten versorgt. Benutzer der Sonde können sich an ihr für Ereignisse registrieren. Sie werden dann in den gewählten Abständen über den Meßwerte informiert.

Mit Hilfe von Sonden werden aus den Zuständen des realen technischen Gerätes ggf. Aggregationen beziehungsweise andere fachlich motivierte Werte berechnet. Der Automat kennt das Verfahren, nach dem diese Werte berechnet werden. Er schreibt sein Ergebnis in die dafür vorgesehene Sonde. Die Sonde kennt alle ihre Klienten und benachrichtigt diese über eine Zustandsänderung.

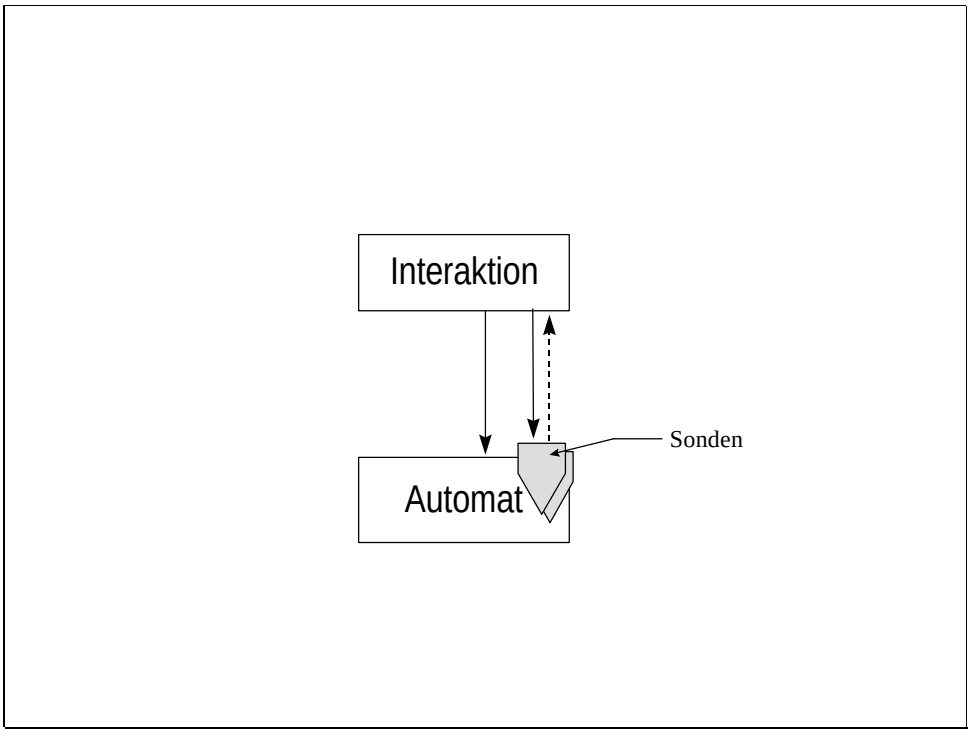

Abbildung 22 Das Sondenkonzept

Eine Komponente des Anwendungssystems kann sich bei einer Sonde registrieren, um einen Automaten nach einem technischenWert zu sondieren. Bei der Registrierung kann das Werkzeug angeben, wie häufig es über Änderungen informiert werden will (*Sampling-Rate*, bzw. bei jeder Änderung s.o.). Die Abbildung 22 zeigt eine Interaktionskomponenten, die einen Automaten und eine von zwei Sonden benutzt. Die Interaktion stellt an der Sonde die Häufigkeit ein und wird von dieser in der gewählten Frequenz benachrichtigt.

In einem Beispiel ausgedrückt, könnte man sich zwei softwaretechnische Automaten vorstellen, die mit realen technischen Bestandteilen eines Fahrzeugs verbunden sind. Der erste Automat repräsentiert den Motor eines Fahrzeugs und stellt eine Sonde für die Umdrehungszahl zur Verfügung. Außerdem kann man am Automaten den Hubraum des Motors sondieren. Während es sich bei der Umdrehungszahl um eine in der Zeit veränderliche Größe handelt, ist die Größe des Hubraums über die Lebensdauer des Motors unveränderlich. Ein zweiter Automat, der für die Benzinpumpe des Fahrzeugs steht, stellt eine Sonde zur Verfügung, mit der man über den Durchfluß des Kraftstoffs informiert wird. Bei beiden Sonden, Umdrehungszahl und Durchfluß, kann die Ereignisfrequenz eingestellt werden.

Zur Konstruktion eines Werkzeugs, welches darüber hinaus den Verbrauch des Motors anzeigen soll, würde man nun in der Funktionskomponente zwei Sonden benutzen, um aus den gelieferten Werten den Verbrauch des Motors zu berechnen.

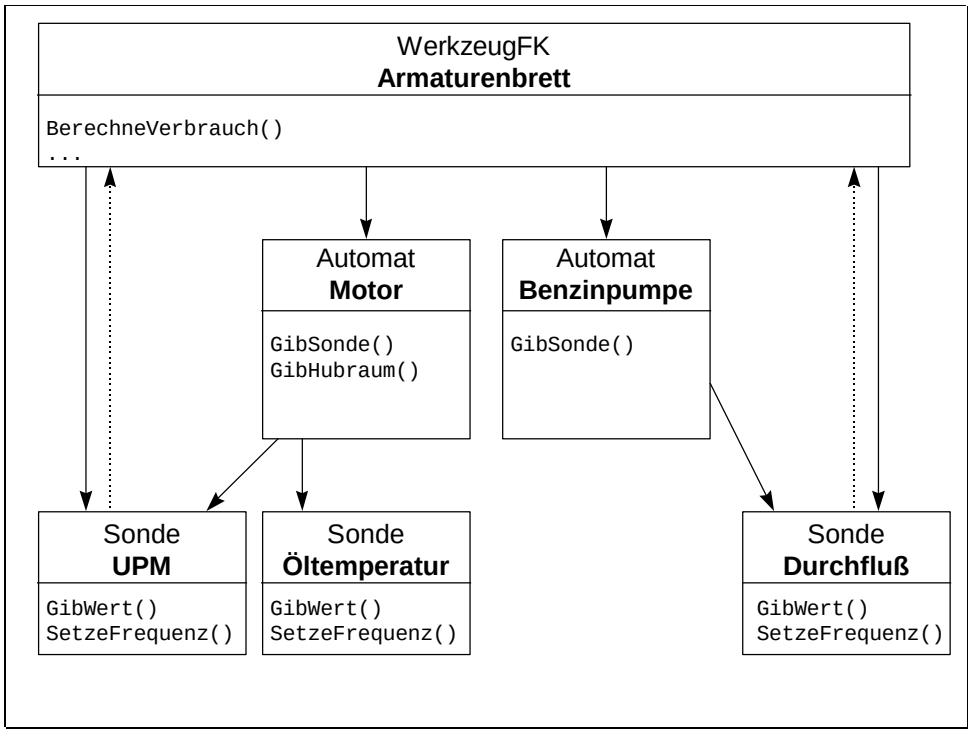

Abbildung 23 Beispiel Softwarewerkzeuge für Armaturenbretter[10]

Das in Abbildung 23 skizzierte Beispiel soll noch einmal das zuvor Beschriebene verdeutlichen. Ein Werkzeug soll technische Werte eines Fahrzeugs im Sinne eines Armaturenbretts anzeigen. Einige der Werte können nur durch Berechnung ermittelt werden (hier der Verbrauch). Dafür verwendet es zwei softwaretechnische Automaten, die reale Geräte im Fahrzeug repräsentieren. Unveränderliche Werte eines Automaten können an diesem direkt sondiert werden. Für über die Zeit veränderliche Größen kann die Funktionskomponente am Automaten eine Sonde anfordern. Diese Sonde liefert einen speziellen fachlichen Zustand. Dazu kann die Sonde eingestellt werden, in welcher Frequenz sie diesen Wert zur Verfügung stellt. Die Funktionskomponente kann nun aus den von beiden Sonden gelieferten Werten einen neuen fachlichen Zustand berechnen, der über die Interaktion angezeigt werden kann.

Technische Automaten und Sonden stellen die benötigten Metaphern für die softwaretechnische Umsetzung von technisch motivierten Anwendungssystemen bereit. Der technische Automat stellt Sonden für Klienten zur Verfügung. Technischer Automat und Sonden bilden als Einheit ein Gesamtkonzept. Diese Einheit ist eine neue Ereignisquelle im Anwendungssystem.

6.3 Einstellwerkzeuge

In dem bisherigen Repertoire sind Werkzeuge ausschließlich dafür vorgesehen, Materialien zu visualisieren und sie zu verändern. Material wird als passiv verstanden. Werkzeuge sind in dem Sinne reaktiv, daß sie auf Aktionen des Anwenders reagieren. Werkzeuge und Automaten sind die einzigen Bestandteile der Modellwelt, die Änderungen am Material durchführen können.

[10] UPM = Umdrehungen Pro Minute

Obwohl Automaten eine Interaktionskomponente besitzen können, ist dies nicht notwendigerweise immer möglich. Trotzdem ist es immer wieder notwendig, an ihnen Einstellungen, ähnlich wie an Werkzeugen, vorzunehmen. Dafür wurden spezielle Werkzeuge verwendet. Diese werden *Einstellwerkzeuge* genannt und in diesem Abschnitt eingeführt.

Mit Hilfe von Einstellwerkzeugen kann einerseits der aktuelle Zustand des Automaten angemessen repräsentiert werden und andererseits kann der Anwender geeignete Änderungen an den Einstellungen des Automaten vornehmen.

Dazu untersucht man zunächst noch einmal die Beziehung zwischen Automaten und Werkzeugen. Der Automat ist die Metapher des WAM Repertoires, die immer dann angewendet wird, wenn sich wiederholende Aufgaben ohne Zutun des Anwenders ausgeführt werden müssen.

Werkzeuge bleiben reaktiv. Allerdings reagieren Einstellwerkzeuge nun nicht ausschließlich auf Aktionen des Anwenders, sondern auch auf Zustandsänderungen eines angebundenen Automaten.

Aus den Projekten, an deren Fortschreibung ich beteiligt war, konnte man als Erfahrung herausziehen, daß alle dort konstruierten Automaten einer wiederholten, wenn auch teilweise seltenen Einstellung bedurften[11]. Es hat sich gezeigt, das in gewissen Abständen, deren Länge deutlich vom Anwendungsfeld abhängig ist, diverse Einstellungen am Automaten notwendig waren.

In den referenzierten Industrieprojekten hat man Arbeitsplätze kennengelernt, die durch einen softwaretechnisch realisierten Automaten mit ihren Arbeitsgegenständen kontinuierlich versorgt worden sind. Hier wurde mit Hilfe eines speziell dafür entwickelten Werkzeugs am Automaten eingestellt, welchen Charakter die Arbeitsgegenstände[12] haben sollten bzw. ob der Automat mit der Verteilung der Arbeitsgegenstände pausieren soll.

Unklar ist an dieser Stelle die Antwort auf die Frage der fachlichen Einordnung dieser neuen Werkzeugkategorie. Das entstandene Problem stellte sich wie folgt dar: Erstens sind technische Automaten zusätzlich mit einer Berechnungen durchführenden Funktionskomponente verbunden, um aus den von Automaten gelieferten Basiswerten neue fachliche Werte zu berechnen. Zweitens ist die räumliche Distanz zwischen dem den Automaten ausführenden Rechner und dem Benutzer, der die Einstellungen vornehmen will, üblicherweise so groß, daß keine „direkte" Einflußnahme möglich ist.[13]

Es wurde sich im Zusammenhang mit technischen Automaten eines weiteren Konstruktionsmusters bedient. Im Rahmen von Wartungsarbeiten, die in regelmäßigen Abständen oder bei Bedarf von qualifiziertem Personal an den betriebenen Automaten durchgeführt werden, finden hierfür spezielle Werkzeuge Verwendung, die einzelne Parameter eines Automaten einstellen können. Daraus wurde der Begriff des Einstellwerkzeugs abgeleitet. Das ist ein spezielles Softwarewerkzeug, das dazu benutzt wird, die Parameter eines Automaten zu visualisieren und dem Anwender die Möglichkeit zu geben, einzelne Parameter zu verändern.

[11] Projekt „Agentenmonitor“, Projekt „MCC Online Configuration“
[12] In diesem Anwendungsfall handelt es sich bei den Arbeitsgegenständen um Telefonate

Definition 13. Einstellwerkzeug

Ein Einstellwerkzeug *ist ein spezielles Werkzeug, das dazu dient, einerseits den (partiellen) Zustand eines oder mehrerer Automaten anzuzeigen und andererseits es ermöglicht, Dinge an diesen Automaten einzustellen.*

Automaten werden als weitestgehend unabhängige Bestandteile eines Anwendungssystems begriffen, deren Ergebnisse als Grundlage für weitere Arbeiten verwendet werden. Diese Automaten haben einen Zustand, der sich aus dem Verlauf der gerade bearbeiteten Routineaufgabe und den allgemeinen Einstellungen des Automaten zusammensetzt. Das Konzept des Automaten ist das eines Gerätes, das man einmal einstellt und dann wiederholt sondiert. Im synchronen Verständnis sind also diejenigen Bestandteile des Systems, die mit dem Automaten in Beziehung treten, gezwungen auf seine Ereignisse zu warten und dann erst zu reagieren.

Wie im Anwendungsfeld für mechanisch realisierte Automaten, so ist es auch im softwaretechnischen Anwendungsfeld notwendig, daß eine Person mit der Beaufsichtigung der automatisierten Tätigkeiten betraut ist. Diese Tätigkeit umfaßt regelmäßige Kontrollen, um eventuelle Fehlfunktionen, die sich abzeichnen können, zu vermeiden. Deshalb benötigt man für das Anwendungsfeld ein Softwarewerkzeug, welches den Zustand des Automaten sichtbar machen kann.

Andererseits nützt diese Möglichkeit der Kontrolle eines Automaten nur dann etwas, wenn seine automatische Abarbeitung von vorgegebenen Handlungsfolgen in gewisser Weise beeinflußt werden kann. Die Einstellungen des Automaten müßten verändert werden können. Diese Anforderungen werden folgendermaßen realisiert:

Für die Darstellung des fachlichen Zustands des Automaten stellt das Werkzeug eine Verbindung zum Automaten her, über die es mit Zustandsinformationen versorgt wird. Sowohl die Auswahl der Informationen, als auch die Regelmäßigkeit, in der die Zustandsbeschreibung geliefert wird, können ggf. vom Werkzeug bestimmt werden. Es mag aus fachlicher Sicht wünschenswert sein, in einem Fall nur einige wenige Teilzustände des Automaten zu visualisieren und in einem anderen Fall den „Gesamtzustand" darzustellen. Die zeitlichen Intervalle, in denen die Informationen an das Werkzeug übermittelt werden, können nicht allgemein vordefiniert werden. Unterschiedliche fachliche Anwendungsfelder werden ebenso unterschiedliche Anforderungen daran stellen. Typische Anforderungen an die Übermittlung einer Zustandsänderung eines Automaten an das Einstellwerkzeug sind: „bei jeder Änderung" und „in festen zeitlichen Abständen". Eher selten läßt sich im Anwendungsfeld der Telefonie fachlich motivieren, daß der Anwender explizit eine Zustandsabfrage durchführt.[14]

Zur Änderung der Einstellungen am Automaten bietet ein Einstellwerkzeug dem Anwender die Möglichkeit, Änderungswünsche zu formulieren. Diese werden dann an den Automaten als Wünsche übermittelt. Inwieweit der Automat die daraus resultierenden neuen Zustände einnimmt bzw. die Zustandsänderungen ablehnt bleibt

[13] Diese Eigenschaft trifft nicht auf jedes Anwendungsfeld zu.

[14] Automatische regelmäßige Zustandsabfragen werden deshalb bevorzugt, weil Anwender dann ohne Benutzung von Tastatur und Maus durch einfaches Ansehen des Bildschirms bestimmte Zustände aktuell „im Blick“ haben. Das entspricht der Arbeitsweise eines Call Center-Supervisors.

durch die Konzeption des konkreten Automaten vorgegeben und muß dort sowohl fachlich als auch technisch begründet werden.

Einstellwerkzeuge werden aus den Komponenten Umgebung, Interaktion, Funktion, Proxy, Sonde und Automat aufgebaut. Im folgenden wird erläutert, in welcher Beziehung diese zueinander stehen. Die Umgebung ist die für alle Werkzeuge umfassende Komponente. Sie hat das Wissen, welche Werkzeuge erzeugt werden können, welche laufen und sie kann neue erzeugen und bestehende wieder schließen.

Das Werkzeug selbst besteht aus einer Interaktion, der Funktion und benutzt das Automaten-Proxy. Die Interaktionskomponente ist ausschließlich für die Darstellung des Automatenzustandes vorgesehen und erlaubt das Absetzen von Änderungswünschen. Sie speichert keine Werte und ist deshalb auf eine Funktionskomponente angewiesen, von der sie jederzeit den fachlichen Zustand abfragen kann.

Die Funktionskomponente liefert der Interaktion den Zugang zum fachlichen Zustand und koordiniert die Verbindung zum Automaten. Obwohl jedes Proxy den Zustand des Automaten lokal abbildet, wird eine Funktionskomponente benötigt. Diese transformiert die technischen Werte eines Automaten in ihre fachliche Interpretation. Werden mehrere Automaten mit Hilfe eines Werkzeugs gesteuert, dann findet in der Funktionskomponente die fachlich begründete Kombination der technischen Ereignisquellen statt.

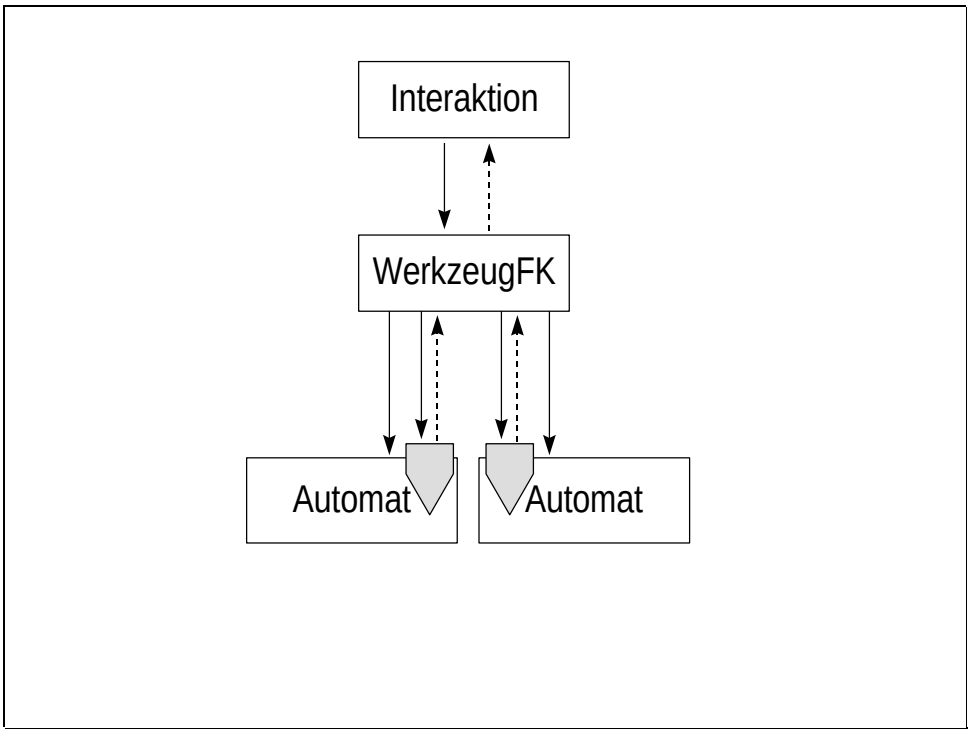

Abbildung 24 Der Aufbau eines Einstellwerkzeugs

Eine Funktionskomponente kann die Vorhaltung des fachlichen Zustands und seine Interpretation für die Interaktionskomponente übernehmen. Mit einer FK wird es

außerdem möglich, den fachlichen Zustand und die Einstellungen unterschiedlicher Automaten, für eine IAK zu kombinieren.

6.4 Zusammenfassung

In diesem Kapitel wurde gezeigt, daß die Metapher des Automaten auf ein technisches Anwendungsfeld ausgeweitet werden kann. Daraus ist der Begriff des technischen Automaten entstanden. Es wird mit Hilfe von Sonden realisiert, die es ermöglichen, partielle Automatenzustände verfügbar zu machen.

Der Begriff des Einstellwerkzeugs wurde als Ergänzung zur Werkzeugmetapher eingeführt. Einstellwerkzeuge sind ebenfalls im Anwendungsgebiet des technischen Automaten entstanden. Diese Werkzeuge ermöglichen das Visualisieren und Einstellen von Automaten. Sie werden notwendig, wenn fachliche Zustände von Automaten kombiniert werden sollen.

7 Vom Einzelprozeßsystem zum Mehrprozeßraum

Dieses Kapitel greift das bereits thematisierte Problem der Granularität objektorientiert entworfener Anwendungssysteme auf. Zwischen dem Objekt als Einheit und der Einheit Anwendungssystem gibt es keine natürliche Aufteilung, die Anwendungssysteme in vertretbare Einheiten untergliedern. Die Verwendung von Entwurfsmustern ist ein Ansatz, um bei der Konstruktion von Anwendungssystemen eine Gliederungsebene zwischen den bestehenden Ebenen einzuziehen.

In diesem Kapitel wird gezeigt werden, welche alternative Gliederungsmöglichkeit auf der Ebene von Werkzeugen und Automaten zur Verfügung steht. Begreift man diese als Komponenten eines Anwendungssystems und schafft die notwendigen Voraussetzung für den Umgang mit ihnen, dann können in dieser Infrastruktur Anwendungssysteme entworfen werden, die auf der Ebene der Werkzeuge und Automaten in Komponenten untergliedert werden können.

Ein Großteil der bisher nach der WAM-Methodik konstruierten Anwendungsysteme sind für den Einzelprozeßraum entwickelt. Genau ein Umgebungsobjekt koordiniert den Ereignisfluß zwischen den Werkzeugen (vgl. [Gry96], Seite 144), die im Rahmen der Arbeit des Benutzers vom Anwendungssystem zur Verfügung gestellt wurden. Jedes Werkzeug wird als Kontextwerkzeug aufgefaßt, das den Rahmen für Sub-Werkzeuge (vgl. [Rie95], Seite 55) bildet. Diese Konstruktionstechnik unterstützt noch keine autonomen Komponenten, wie man sie gerne haben würde.

Insbesondere setzt die mit WAM verknüpfte Konstruktionstechnik eine synchrone Kopplung von Objekten voraus. Dieses Kriterium wird z.B. vom Observer-Muster[15] (vgl. [Gam94], Seite 293) erfüllt, welches zur Trennung von Funktionskomponente (FK) und Interaktionskomponente (IAK) und auch zur Anbindung des Umgebungsobjektes verwendet wird. Läßt man allerdings mehr als einen Prozeß innerhalb dieser Konstruktionstechnik zu und verbindet beide Prozesse über ein Kommunikationsmedium

[15] Wenn wir uns auf Gestaltungsmuster auf der technischen Ebene beziehen, werden wir diese mit ihrem englischen Namen, wie er in der einschlägigen Literatur geprägt wurde, referenzieren.

(lokale Interprozeßkommunikation mittels *Signals* oder LAN/WAN Kommunikation mit Hilfe von Datenpaketen) muß man asynchrone Kopplung berücksichtigen.

Zum Beispiel möchte man Werkzeuge und Automaten so realisieren, daß die Automaten als eigenständige Komponenten auf separaten Maschinen laufen. Dann benötigt man die Fähigkeit, Prozesse miteinander zu koppeln. Dies ist auch wünschenswert, weil aus Sicht der Entwickler voneinander abgrenzbare Komponenten entwickelt werden.

In diesem Kapitel wird eine Anbindung vorgestellt, die asynchrone Kommunikation an den bereits vorhandenen synchronen Kontrollfluß anknüpft. Dies ermöglicht es, separate Prozesse miteinander zu koppeln. Dadurch ist es nunmehr denkbar, Werkzeuge in unterschiedlichen Prozessen für einen Rechner zu konstruieren und auch Werkzeuge mit Automaten in lokalen oder weit entfernten Netzen miteinander kommunizieren zu lassen. Es wird dabei auf Erfahrungen zurückgegriffen, die im Rahmen einer zweijährigen Kooperationstätigkeit gesammelt wurden.

Dieses Kapitel gliedert sich dafür in sechs Bereiche. Nach der Motivation wird im zweiten Bereich die Kopplung von Werkzeugen im singulären Prozeßraum untersucht. Das dritte Unterkapitel geht auf das Finden der Komponenten näher ein. Das vierte Unterkapitel beschäftigt sich mit den derzeit bekannten verschiedenen Varianten der Prozeßkopplung und stellt ihre Eigenschaften, sowie Vor- und Nachteile gegenüber. Im fünften Abschnitt wird das gewählte allgemeine Muster zur Kopplung von Werkzeugen und Automaten vorgestellt. Der sechste und letzte Abschnitt faßt die Ergebnisse dieses Kapitels noch einmal zusammen.

7.1 Motivation

Die Methodik WAM hat sich bisher auf die Konstruktion von Anwendungssystemen konzentriert, die als einzelner Prozeß eines Betriebssystems realisiert waren. Es konnten dabei vier Nachteile identifizieren werden:

Erstens große *Programmdateien* (Executables): Da alle Komponenten eines Systems innerhalb eines Prozesses laufen, ergeben sich daraus zwangsläufig große Programme. Insbesondere müssen in einer Sitzung unbenötigte Teile geladen und ggf. initialisiert werden.[16]

Zweitens lange *Turnaround-Zeiten*: Da zum Erstellen[17] eines Systems immer das gesamte System neu gebunden werden muß, ergeben sich lange Bindezeiten. Durch schwer zu vermeidende Abhängigkeiten kann außerdem das vollständige Übersetzen aller Werkzeuge notwendig werden.

Drittens schlechte *Konfektionierbarkeit*: Unterschiedliche Kundenansprüche können eine unterschiedliche Zusammenstellung von Werkzeugen für die kundenspezifischen Anforderungen ergeben. Bei einer Fehlerkorrektur bedeutet dies, daß für jeden Kunden ein separat zu konfektionierendes Anwendungssystem zu erstellen ist. (Dieser Aspekt wird im nächsten Kapitel wieder aufgegriffen.)

[16] Wenngleich sich dieses Argument dahingehend abschwächt, daß bei einer Aufteilung natürlich bestimmte Programmkomponenten dafür doppelt geladen werden müssen.
[17] Ob zum Testen oder zur Auslieferung

Viertens kein *Verteilungsmodell*: Es ist möglich, externe Komponenten mit Hilfe des Musters „Automat“ in ein Anwendungssystem zu integrieren. Jedoch ergibt sich daraus noch kein generelles Muster für die Verteilung aller Typen von Komponenten eines Anwendungssystems, wie z.B. für Werkzeuge. Jeder Server (siehe z.B. Anbindung einer Datenbank an die Materialverwaltung) wird „von Hand“ angebunden.

Zusammenfassend ist es die monolithische Struktur eines Anwendungssystems, die bei der Weiterentwicklung von großen Anwendungssystemen hinderlich ist. Im Rahmen der Arbeit bei der Firma Micrologica wurde versucht, das Modell und die Methodik von WAM auf den Mehrprozeßraum auszudehnen, indem Werkzeuge nach den Leitlinien der Konstruktionstechnik von WAM gebaut wurden, die als Einstellwerkzeuge für - bereits nach anderer Methodik konstruierte - Automaten dienten. Dazu war es notwendig, den synchronen Kontrollfluß innerhalb des Anwendungssystems über ein asynchrones Kommunikationsmedium mit einem anderen Prozeß zu verknüpfen.

Dieses Kapitel zeigt deshalb, welche Methoden zur Aufteilung und Verknüpfung von Prozessen gewählt wurden, und welche Vorteile aus dieser Art der Anbindung resultieren.

7.2 Kopplung im Einzelprozeßraum

Nach der Einführung in die Differenzierung des Werkzeugbegriffs im Zusammenhang mit Automaten im vorangegangenen Kapitel, widmet sich dieser Abschnitt dem Resümee der bisher in der Literatur ausgearbeiteten Kopplung von Werkzeugen mit Automaten im singulären Prozeßraum. Nur eine intensive Untersuchung der Kopplung führt zu einem Verständnis, mit dessen Hilfe eine fundierte Wahl der Komponenten in einem Anwendungssystem durchgeführt werden kann.

Als ein einfaches Beispiel für den synchronen Kontrolfluß kann die Anbindung des Materialverwalters genommen werden. Dieser Materialverwalter wird benötigt, um u.a. das Konzept von persistenten Materialien und das Konzept von Original und Kopie zu realisieren. Dieser Anlaß wird dazu genutzt, um die Konstruktion eines Anwendungssystems unter dem Paradigma des synchronen Kontrollflusses zu betrachten.

In diesem Beispiel wird der Fall betrachtet, daß ein Werkzeug ein Material anfordert, um es auf dem Bildschirm darzustellen. Nach ([WW94], Seite 8) ergibt sich folgende Beziehung zwischen den (notwendigen) Objekten[18]:

[18] Interaktionskomponenten werden an dieser Stelle nicht betrachtet, da sie die Darstellung unnötig komplizierter machen würden.

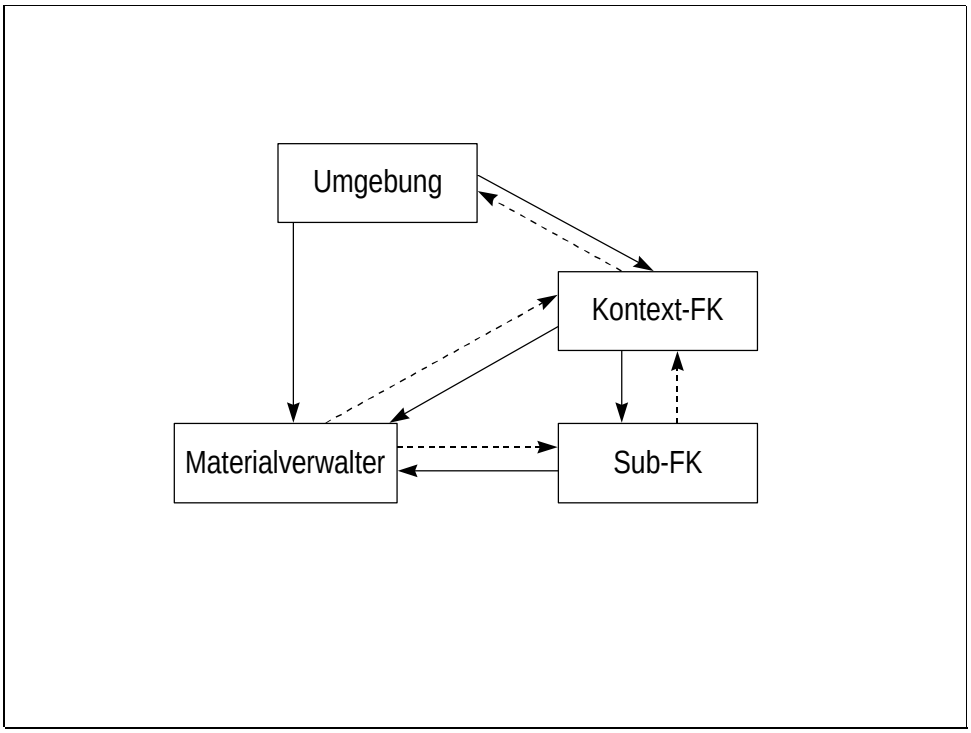

Abbildung 25 Das Zusammenspiel eines komplexen Werkzeuges mit dem Materialverwalter

Das Umgebungsobjekt wird benötigt, weil in ihm der Materialverwalter verankert ist: die Umgebung sorgt für dessen Erzeugung. Die Umgebung erzeugt ebenfalls die konkreten Werkzeuge (hier allgemein mit Kontext FK bezeichnet). Die Kontext-FK benutzt die Materialverwaltung ebenso wie ihre Sub-FK, um Materialien im Original oder als Kopie anzufordern. Über den Beobachter-Mechanismus werden sie über Ereignisse die Materialverwaltung betreffend benachrichtigt:

> *„... wir [verstehen] die Interaktion zwischen Werkzeug und Materialverwalter insbesondere auch als Beobachter-Beziehung...“ (vgl. [WW94], Seite 9)*

Der Mechanismus wird dazu benutzt, Werkzeuge über neue Materialien im Sinne von Erzeugung oder Veränderung zur informieren:

> *„Vorstellbar wäre etwa ein Editor, mit dem neue Dokumente erstellt werden können, die nach ihrer Bearbeitung und Übergabe an den Materialverwalter allgemeinen in der Umgebung zur Verfügung stehen. Dem Materialverwalter soll in diesem Fall die Möglichkeit gegeben sein, einen bereits gestarteten Dokumentenbrowser von der Existenz des neuen Dokuments zu benachrichtigen.“ (vgl. [WW94], Seite 9)*

Der Mechanismus wird für das Konzept von Original- und Kopie angewendet, welches es ermöglicht, über eine abgeschlossene Veränderung an einem Material informiert zu werden.

> *„Die Freigabe eines Materialoriginals nehmen wir daher auch in den Katalog möglicher Ereignisse auf, die ein Materialverwalter seinen beobachtenden Werkzeugen melden kann.“ (vgl. [WW94], Seite 10)*

Die hier beschriebene Kopplung zwischen Kontext-FK und Sub-FK und dem Materialverwalter beruht auf dem synchronen Observer-Muster. Das garantiert die Konsistenz der Daten während einer Benachrichtigungsphase. Synchronizität ist die Brandmauer bei der Konstruktion von Anwendungssystemen nach WAM.

Dies läßt sich an einem anderen Beispiel leicht verdeutlichen. Laufen in einem Anwendungssystem zur Zeit mehrere Werkzeuge, die dasselbe Material in Kopie benutzen und ein weiteres Werkzeug, mit dem genau dieses Material bearbeitet wird, dann sorgt die Materialverwaltung dafür (s.o.), daß alle anderen Werkzeuge benachrichtigt werden, wenn die Bearbeitung abgeschlossen ist. Innerhalb dieses Benachrichtigungszyklusses dürfen nur sondierende Operationen von den Werkzeugen aufgerufen werden. Damit ist garantiert, daß unabhängig von der Reihenfolge der Sondierung alle Werkzeuge mit demselben Zustand (des Automaten und des Werkzeugs) versorgt werden. Und nicht, während der Kontrollfluß der Benachrichtigung in das eine beobachtende Objekt wandert, von diesem Veränderungen durchgeführt werden, so daß andere, nachfolgende Objekte mit einem anderen Objektzustand konfrontiert werden.

Der in diesem Zusammenhang verwendete Begriff des Anwendungssystems verdient eine genauere Betrachtung, da immer dann auf ihn Bezug genommen wird, wenn es um die Zusammenstellung von Werkzeugen und Automaten gehen wird. Außerdem erlaubt dieser Begriff die fachliche Abgrenzung verteilter Systeme.

Definition 14. Anwendungssystem

Unter einem Anwendungssystem *wird die fachliche Zusammenfassung von Software-Werkzeugen, Automaten und Arbeitsplatzsystemen (Spezialfall ein Arbeitsplatz) zu einem Ganzen verstanden. Das Anwendungssystem unterstützt Anwender bei der Erledigung ihrer Aufgaben, indem es jedem - mit Hilfe einer Arbeitsumgebung - einen konsistenten Ort zur Verfügung stellt.*

Wird das Anwendungssystem vom Einzelprozeßraum in eine verteilte Umgebung gerückt, muß man bei der begrifflichen Unterscheidung einzelne Arbeitsplätze von einem Anwendungssystem abgrenzen können. Dabei hilft der Begriff des Arbeitsplatzsystems:

Definition 15. Arbeitsplatzsystem

Ein Arbeitsplatzsystem *ist ein Bestandteil eines Anwendungssystems, der genau für einen Arbeitsplatz konzipiert wurde. Er besteht aus Software-Werkzeugen und ggf. Automaten und stellt dem Anwender Möglichkeiten zum Austausch von Material und zur Kommunikation mit anderen Anwendern zur Verfügung. Ein Arbeitsplatzsystem realisiert genau eine Arbeitsumgebung für einen Anwender.*

Aus den beiden hier eingeführten Begriffen kann eine Begriffshierarchie abgeleitet werden, in deren Kontext sich die folgenden Feststellungen bewegen. Der übergeordnete Begriff des Anwendungssystems bezeichnet in diesem Kontext ein Gesamtsystem, das einer Gruppe von Anwendern zur Erledigung ihrer Aufgaben zur Verfügung gestellt wird. Mit Rücksicht auf die Anforderung einer gleichzeitigen Aufgabenerledigung wird es notwendig, im Rahmen dieses Anwendungssystems mehrere Arbeitsplätze anzubieten, an denen Mitarbeiter ihren Aufgaben zeitlich unabhängig voneinander nachgehen können. In diesem Zusammenhang spricht man von einem Arbeitsplatzsystem. Dieses ist immer in ein Anwendungssystem eingebettet und stellt deshalb zusätzlich Werkzeuge zur

Kooperation zur Verfügung. Außerdem kann es Werkzeuge enthalten, die das Einstellen von (entfernten) Automaten (s.u.) ermöglichen.

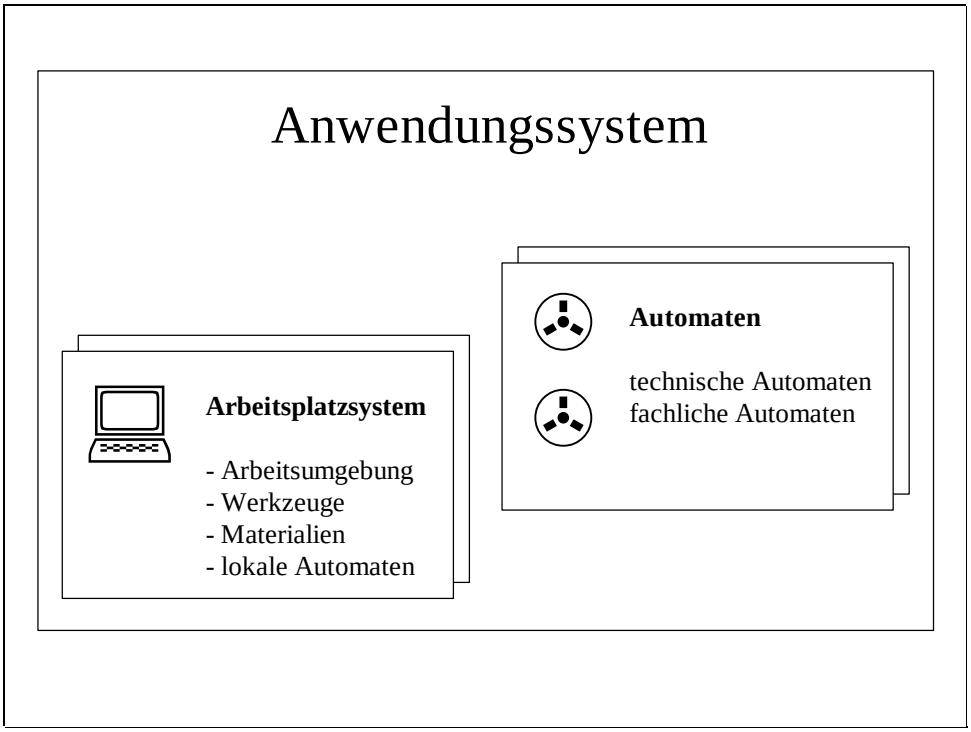

Abbildung 26 Beziehung zwischen Anwendungssystem und Arbeitsplatzsystem

In diesem Abschnitt wird näher auf die Konstruktion von Anwendungssystemen eingegangen, die für den Einzelprozeßraum konstruiert wurden, um dann im Laufe dieses Kapitels die Trennung in mehrere Prozesse zu verfolgen. Dabei wird wiederholt bezug auf Abbildung 25 genommen und die dort vorgefundene Konstellation wird näher betrachtet.

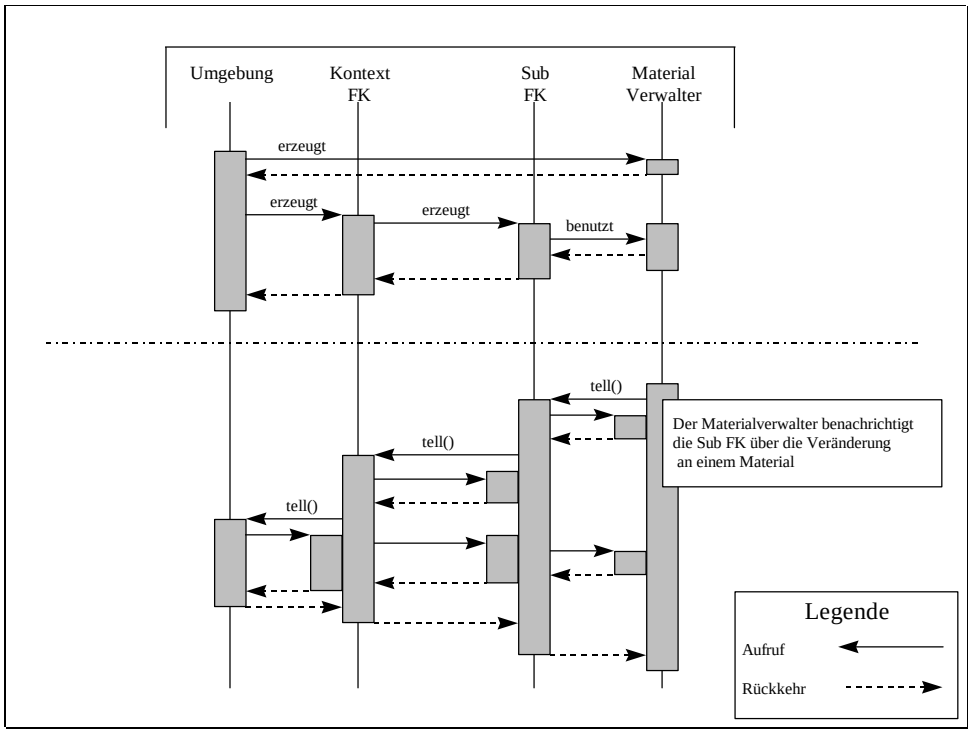

Abbildung 27 Interaktionsdiagramme für typische synchrone Abläufe in Anwendungssystemen (2 Beispiele)

Die Abbildung 27 zeigt, wie der Kontrollfluß (oberes Beispiel) von der Umgebung über die Kontext-FK und die Sub-FK in den Materialverwalter gelangt und auf genau dem umgekehrten Weg wieder zurückkehrt. Die gestrichelte Linie trennt zwei verschiedene Beispiele. In diesem Text werden die von Jacobson eingeführten und Lilienthal und Strunk erweiterten (vgl. [LS96]) Interaktionsdiagramme als Darstellungsmedium verwendet. Später wird gezeigt werden, wie diese Diagramme in ihrer Darstellung und Interpretation erweitert wurden, um sie für Interprozeßkommunikation zu verwenden. Bereits in diesem Diagramm angedeutet ist die Zusammengehörigkeit der Objekte zu einem Prozeß. Dies wird mit der oberen Klammer über den Objektlinien dargestellt.

Im unteren Beispiel von Abbildung 27 sieht man, wie die Benachrichtigung des Materialverwalters, angedeutet mittels tell(), bis in das Umgebungsobjekt wandert. Der Grund dafür können zum Beispiel aus der ersten Benachrichtigung ausgelöste weitere Ereignisse sein, die die Semantik der Veränderung betreffen. Bei der Verwendung der sondierenden Operationen wird vorausgesetzt, daß die Daten konsistent sind, weil dies durch die Synchronizität gewährleistet wird: Keine verändernde Operation darf während eines tell()s gerufen werden ([RW96], Seite 51) und keine anderen Objekte sind wegen des vorausgesetzten Einzelprozeßraums aktiv.[19]

Die aufgezeigten Rahmenbedingungen sind absolut notwendig, um bei der Konstruktion von Anwendungssystemen die Konsistenz der Daten zu garantieren. Deshalb werden die Bedingungen der Brandmauer zur Konstruktion von Prozessen mittels des Observer-Musters hier noch einmal zusammengefaßt:

[19] Nebenbedingung

- Beobachtete Objekte bieten drei verschiedene Arten von Methoden an (1). Testende Methoden verändern den Objektzustand nicht und gelten uneingeschränkt. Die sondierenden Methoden verändern den Zustand des Objektes ebenfalls nicht, sie sind allerdings partiell. Alle übrigen Methoden können den Zustand des Objektes ändern.
- Innerhalb eines Benachrichtigungszyklusses können vom benachrichtigen Objekt nur die sondierenden oder testenden Methoden aufgerufen werden (2).

Genau unter diesen Rahmenbedingungen können Objekte eines Anwendungssystems im Einzelprozeßraum mit Hilfe des Beobachter-Musters gekoppelt werden, ohne daß eine zu starke Abhängigkeit produziert wird.

Es wird deshalb wie folgt definiert:

Definition 16. Einprozeßsystem
Ein Einprozeßsystem *ist ein Anwendungssystem, welches in einem einzigen Prozeß eines Betriebssystem läuft. Die internen Objekte und extern angeschlossenen Komponenten (z.B. Datenbanken) sind durch synchrone Mechanismen miteinander verbunden.*

Faßt man die in diesem Abschnitt bereits erarbeiteten Erkenntnisse zusammen, so läßt sich feststellen, daß für die Konstruktion von Anwendungssystemen bestimmte Voraussetzung erfüllt sein müssen, die sich insbesondere auf die Kopplung der am Anwendungssystem beteiligten Objekte beziehen. Für die Übertragung der Konstruktionstechnik vom Einzelprozeßsystem zu einem Mehrprozeßsystem ist es deshalb von besonderem Interesse, die Charakteristika der Kopplung detailliert zu untersuchen.

7.2.1 Kopplung von Objekten

Die Schnittstelle zwischen zwei Objekten in einem objektorientierten System ist der Punkt, an dem der Gedanke an eine Verteilung ansetzen kann. An dieser Stelle werden deshalb die Eigenschaften einer solchen Kopplung näher betrachtet.

Betrachtet man zuerst den Begriff der Kopplung, der zwar schon mehrfach Verwendung gefunden, bisher aber noch kein direktes und vor allem konkretes Verständnis erlangt hat.

In objektorientierten Anwendungssystemen mißt man der Beziehungen zwischen Objekten auf der Ebene von Vererbung und Benutzung eine große Bedeutung zu. Vererbung wird ausschließlich zur Nutzung über abstrakte Schnittstellen eingesetzt und es wird über Vererbung keine Code-Wiederverwendung betrieben[20]. Ebensolche detaillierte Betrachtungsweise bedarf es beim Einsatz der Benutztbeziehung. Immer dann, wenn Objekte über eine Benutztbeziehung in Verbindung stehen, ist es interessant und von großer Bedeutung, diese Beziehung näher zu untersuchen.

[20] In WAM wird Vererbung nur zum Ausdruck der Beziehung Typ-Subtyp verwendet. Erlaubt die Programmiersprache dies nicht, muß auch hier darauf verzichtet werden.

Der Begriff der Kopplung gibt einem eine Kategorieliste an die Hand, mit der man die Benutztbeziehung bewerten kann:

Definition 17. Kopplung

Unter dem Begriff Kopplung *wird die Art der Verbindung verstanden, mit der zwei Objekte miteinander verknüpft sind. Hierbei werden generell drei Varianten unterschieden: Prozedur (Aufruf ohne direktes Ergebnis), Funktion (Aufruf mit direktem Ergebnis) und Beobachter (verändernde Prozeduren, Funktionen im mathematischen Sinn, Benachrichtigung).*

Benutzt ein Objekt ein anderes, indem es eine konkrete Methode aufruft (keinen Callback), die keinen Rückgabewert besitzt (Prozedur), dann ist die Intention dieser Kopplung die Änderung des Zustands am aufgerufenen Objekt. Als Kopplungsmethode wählt man die direkte Kenntnis der Schnittstelle, denn ein aufrufendes Objekt kennt immer eine konkrete Schnittstelle. Bezogen auf die Konstruktionstechnik handelt es sich hierbei zum Beispiel um die Beziehung IAK zur FK, FK zum Proxy usw.

Reaktion eines solchen Methodenaufrufs kann noch während des Aufrufs die Auslösung einer Signalisierung sein. Dieser Aspekt wird intensiver untersucht.

Benutzt ein Objekt ein anderes, indem es eine konkrete Methode mit Rückgabewert aufruft, dann ist die Intention dieser Kopplung die Sondierung eines Zustands vom aufgerufenen Objekt. Auch hier wird die Kenntnis der direkten Schnittstelle gewählt. Diese Technik findet ebenfalls bei den Beziehungen IAK zur FK und FK zum Proxy Anwendung.

Im Gegensatz dazu sind Kopplungen zu verstehen, die über abstrakte Schnittstellen arbeiten und Callback-Methoden verwenden. Dies findet sich z.B. beim Beobachter Muster wieder. Hier ist die Schnittstelle nur in einer Richtung bekannt. In der anderen werden anonyme Callback-Methoden verwendet.

Grundsätzlich wird man zwischen zwei Arten der Kopplung unterscheiden wollen: der festen Kopplung und der losen Kopplung. Beide Adjektive beziehen sich auf die Kenntnis der Schnittstelle des aufrufenden Objektes. Es wird für die weitere Diskussion angenommen, daß es zwei Objekte mit den Namen „A“ und „B“ gibt, wobei das Objekt A die Methode „m“ in Objekt „B“ aufrufen will.

Definition 18. Feste Kopplung

Unter der festen Kopplung *versteht man die Kopplung eines Objektes an ein zweites unter Ausnutzung einer konkreten Schnittstelle.*

```
class TypeB
{
public:
     void m() { /* do something useful */ };

     void o() {
            TypeA A;

            A.p();
            };
```

```
};

class TypeA
{
public:
     void n()   {
            TypeB B;   /* tightly coupled */

            B.m();
            };

     void p() { /* do something more useful */ };
};
```

Während bei der festen Kopplung das Objekt „A“ das Objekt „B“ kennt und dessen Methode direkt, daß heißt über Dereferenzierung am Objekt „B“, aufruft, wird dies bei einer losen Kopplung indirekt gewählt. Dazu kann z.B. das Objekt A eine Schnittstelle anbieten, die bei einer Methode als Parameter eine Referenz auf ein Objekt bestimmten Typs verlangt. Diese Objektreferenz wird gespeichert und für späteren Aufruf verwendet.

Innerhalb einer Kollaboration von Objekten ist die Nutzung der konkreten und vollständigen Schnittstelle eines Objektes in der einen Richtung wünschenswert. In der selben Beziehung wird allerdings gleichzeitig verlangt, daß die Schnittstelle in der anderen Richtung eine abstrakte ist, damit keine „feste Kopplung“ (s.o.) entsteht. In diesem Fall wäre die Designentscheidung zu überdenken und ggf. beide Objekte in ein (neues) Objekt zu legen.

Hingegen kann eine lose Art der Zusammensetzung von Objekten im Sinne der Wiederverwendbarkeit und Erweiterbarkeit von Softwarekomponenten befürwortet werden. Das Observer-Muster bietet dafür eine geeignete Semantik, aus der vor allem folgender Vorteil benutzt werden wird:

> *“When an object should be able to notify other objects without making assumptions about who these objects are. In other words, you don’t want these objects tightly coupled.“ (vgl. [Gam94], Seite 294)*

Es wird definiert:

Definition 19. Lose Kopplung

Unter einer losen Kopplung *versteht man die Kopplung eines Objektes an ein zweites unter Ausnutzung einer abstrakten (Teil-)Schnittstelle.*

Das nachfolgende Beispiel zeigt, wie ein Objekt vom Typ `TypeA` unter Verwendung der abstrakten Klasse `AbstractType` die Schnittstelle eines komplexeren Objektes nur teilweise kennen und benutzen kann.

```
class AbstractType
{
public:
     virtual void m() = 0;
};

class TypeB : public AbstractType
{
```

```
public:
     virtual void m() { /* do something useful */ };
};

class TypeA
{
public:
     void n( AbstractType *pAbstractObject )
            {
            pAbstractObject->m();
            };
};
```

Dies ist ein Beispiel, wie Objekte in einem System gekoppelt werden können. Gestaltungsmuster verwenden diese Technik, um gliedernde Abstraktionen einzuführen. Je nach Anwendungsfeld wird man unterschiedliche Muster wählen, um die beabsichtigte Beziehung ausdrücken zu können. *Frameworks* werden im allgemeinen in der Art angebunden, daß man nur abstrakte Schnittstellen der bereitgestellten Klassen verwendet, um sich möglichst unabhängig von deren Änderung zu machen. Dies ist ebenso im Sinne einer Erweiterbarkeit zu verstehen.

7.2.2 Intention der gewählten Objektkopplung

Grundsätzlich wird man die lose Kopplung bevorzugen, weil damit die Intention der Objektbeziehung deutlicher ausgedrückt werden kann. Außerdem ist dadurch explizit gemacht, was vom Entwickler beabsichtigt wurde. Schließlich erhöht die lose Kopplung die Wiederverwendbarkeit und Austauschbarkeit der Klassen und Objekte.

Das Observer-Muster bietet einen vielfach verwendbaren Mechanismus. Einerseits ist es für die Kopplung von mehreren Objekten gedacht, die sich alle auf ein konkretes Objekt beziehen. Aber auch die Verbindung von nur zwei Objekten, bei denen eine asymmetrische Beziehung besteht, wird damit unterstützt.

Gamma et al. verweisen auf einen weiteren Aspekt:

> *„Because Subject and Observer aren’t tightly coupled, they can belong to different layers of abstraction in a system.“ (vgl. [Gam94], Seite 296)*

Wie bei der Konstruktion von Einstellwerkzeugen gezeigt wurde, werden die Werte des Automaten auf verschiedenen Ebenen erfaßt und weiterverarbeitet. Nimmt man an, daß diese Ebenen durch ein Observer-Muster miteinander verbunden sind, dann ist klar zu erkennen, daß die unterschiedlichen, über dieses Muster miteinander verbundenen Ebenen jeweils eine zusätzliche Interpretation bzw. Abstraktion einführen.

Genau diese Interpretation und Abstraktion ist auch die Begründung dafür, warum separate Objekte zur Identifikation der einzelnen Ebenen verwendet werden.

Wenngleich das Observer-Muster ein geeignetes Mittel zur Kopplung in vielen Fällen ist, so darf nicht vergessen werden, daß gerade durch seine universelle Einsetzbarkeit die Schärfe und Präzision in der Beschreibung der Beziehung zwischen Objekten durch dessen Verwendung verlorengeht. Insbesondere entfällt eine Trennung unterschiedlicher Anliegen in der Schnittstelle des Observers. Es ist deshalb angebracht, in den Anwendungsfällen darüber nachzudenken, ob nicht ein anderes strukturelles Muster eine bessere Aussagekraft erreichen könnte.

7.3 Komponentenfindung

Szenarien und Visionen helfen auf der fachlichen Ebene beim Finden von Arbeitsplätzen, Aufgaben und Tätigkeiten. Diese drei sind Ergebnis der Analyse. Damit liegt bereits ein fachliches Modell für Komponenten vor. Denn Werkzeuge sollen nicht lineare Tätigkeiten unterstützen, sondern für Teilaufgaben, die im Ermessen des Anwenders liegen, zur Verfügung stehen.

Dieser Abschnitt behandelt deshalb die unterschiedlichen Fragestellungen zur Komponentenfindung.

Was wird getrennt? Anwendungssysteme werden zu Komponenten zergliedert. Sie unterteilen sich in Werkzeug- und Automatenkomponenten. Als ein Beispiel dafür kann das Anwendungssystem „Online Config Tool“ dienen. Es wurde in eine Reihe von Werkzeugen getrennt, die in Kapitel 4 beschrieben wurden und Automaten, die sich aus den Anforderungen eines Call Centers ergeben (siehe Kapitel 3).

Wo wird getrennt? An der Verbindung zwischen Werkzeug und Automat kann getrennt werden. Ebenso kann an der Verbindung zwischen Werkzeug und Umgebung getrennt werden. Betrachtet man beispielsweise die Parameter eines Telefongesprächs als Material, dann kann das Werkzeug zum Editieren dieser Parameter und der Automat, der die Telefongespräche nachvollzieht in der Art aufgeteilt werden, daß Werkzeug und Automat voneinander getrennt werden. Das Material, die Parameter, werden vom Automaten an das Werkzeug gegeben und später zurückgesendet.

Warum Werkzeuge trennen? Dafür gibt es eine Reihe von Gründen. Zuerst können Werkzeuge dadurch in anderen Kontexten wiederverwendet werden. Zweitens bilden sie eine abgeschlossene Einheit genau wie auch Automaten (s.u.). Drittens stellen Werkzeuge einen fachlichen Abschluß über einem Handlungsspielraum dar. Viertens agieren Werkzeuge nicht direkt mit anderen Werkzeugen. Und schließlich können Werkzeuge zu bestehenden Materialien hinzukommen. Ein Beispiel dafür sind Automaten, die von Werkzeugen getrennt werden sollen, weil sie auf anderen Maschinen laufen müssen, die mit besonderer Hardware ausgestattet sind. Werkzeuge können in einem Netzwerk weit entfernt von den eigentlichen Automaten benutzt werden. Bestimmte Werkzeuge, wie z.B. das Anmeldewerkzeug, können unabhängig vom konkreten Einsatzfeld des Anwendungssystems verwendet werden.

Warum Automaten trennen? Automaten bilden eigenständige Einheiten eines Anwendungssystems. Außerdem können Automaten genau wie Werkzeuge auch wiederverwendet werden. Automaten müssen teilweise technisch bedingt auf speziellen Maschinen laufen, die für Anwender auf ungeeigneten Standorten stehen. Bestimmte Typen von Automaten kann es nur einmal in einem Unternehmen geben, wenn diese z.B. Datenbanken oder andere eindeutige Datenbestände verwalten.

Warum nicht Materialien als Komponenten wählen? Wichtigstes Argument dagegen ist die Feststellung, daß Materialien passiv sind. Material wird von vielen Werkzeugen und Automaten verwendet. Das erfordert eine enge Kopplung zum bearbeitenden Werkzeug oder Automaten. Der Materialverwalter stellt Material zur Verfügung, er ist in diesem Sinne ein Automat und damit eigenständiger Prozeß; Er müßte demnach Prozesse, die Material realisieren, verwalten. Beispiel: Die im Gegensatz zur Beziehung Automat-Werkzeug oder Werkzeug-Umgebung relative enge Kopplung von Material und

Werkzeug oder Material und Automat würde es enorm aufwendig gestalten, Materialien von diesen zu entkoppeln.

Es wurde auf dieser Grundlage untersucht, wie Werkzeuge und Automaten in separate Prozesse gegliedert werden können.

7.4 Kopplung im Mehrprozeßraum

Für die Aufteilung von Werkzeugen und Automaten in separate Prozesse wird eine Form der Kommunikation benötigt, die den fachlichen Hintergrund der WAM-Methodik ausreichend unterstützt. In diesem Abschnitt wird sich deshalb mit den unterschiedlichen Kommunikationsarten, die derzeit zur Verfügung stehen, beschäftigt. Es wird einerseits dargestellt werden, welche Charakteristika verfügbar sind und andererseits dies mit den Anforderungen in Beziehung setzen.

Grundsätzlich kann auf der technischen Ebene die Betriebsart der Kommunikation über die Art der Synchronisierung unterschieden werden (vgl. [Ker89], Seite 89): synchrone und asynchrone Übertragung. Auf der Programmierebene werden diese beiden Eigenschaften zur Kennzeichnung der Kommunikation verwendet. Auf diese Unterscheidung wird im folgenden noch detaillierter eingegangen.

Es wird an dieser Stelle eine Situation näher untersucht, bei der zwei Objekte in der Art miteinander „kommunizieren“, daß das erste Objekt eine Methode im anderen aufrufen will. Aufgrund von Verteilung sind die beiden Objekte in unterschiedlichen Prozessen untergebracht.

7.4.1 Synchroner Kontrollfluß

Beim synchronen Kontrollfluß wird die Kontrolle, die die Abarbeitung des einen Prozesses gesteuert hat, beim Kommunizieren virtuell an den Zielprozeß abgegeben. Der Kontrollfluß bleibt im sendenden Prozeß an der Stelle stehen und fährt erst dann fort, wenn das Ergebnis des Zielprozesses (ungeachtet eines eventuell auftretenden Fehlers) zurückgesandt wurde.

Definition 20. synchroner Kontrollfluß

Ein Kontrollfluß bzw. eine Interprozeßkommunikation wird als synchron *bezeichnet, wenn der Absender einer Nachricht so lange blockiert ist, bis der Empfänger die Verarbeitung bestätigt hat. Beispiel: Direkte Methodenaufrufe innerhalb eines Programms sind synchron.*

Ein Beispiel für die Umsetzung des synchronen Kontrollflusses für die Interprozeßkommunikation sind *Remote Procedure Calls (RPCs)*. Hierbei werden durch Stellvertreterfunktionen oder Stellvertreterobjekte Methoden angeboten, die nicht im selben Prozeßraum verfügbar sind. Die zur Verfügung gestellte Methode führte anstelle dessen eine Umwandlung der Parameter in ein internes Format durch und überträgt die Informationen über ein geeignetes Kommunikationsmedium zum Zielprozeß. Der Kontrollfluß des Prozesses, in dem die Methode aufgerufen wird, ist so lange angehalten, bis eine Antwort gesendet wurde. Dann kehrt die Methode, ggf. mit einem Rückgabewert, zurück.

Der Corba Ansatz ([OMG93], [OMG94], [OMG95]) verwendet RPCs zur Realisierung seiner Verteilung. Bedingt dadurch ist die Anbindung der beteiligten Prozesse primär synchron. Diese Feststellung hat grundlegende Konsequenzen für den Softwareentwurf. Einerseits können dadurch die Eigenschaften sequentieller Programmierung und damit gewisse Schutzbedingungen weiterhin angenommen werden, andererseits führt die synchrone Anbindung von verteilten Prozessen zu zusätzlichen Nachteilen, die im weiteren Text erarbeitet werden.

Ein Vorteil ist sicherlich, daß die geforderten Konsistenzbedingungen durch diese Art der Kommunikationsanbindung weiterhin erhalten bleiben. Wird z.B. die Benachrichtigung zwischen zwei Objekten in unterschiedlichen Prozessen mittels RPCs durchgeführt, kann während der Sondierung davon ausgegangen werden, daß alle sondierten Werte in einem konsistenten Zustand sind, denn der Kontrollfluß im benachrichtigenden Prozeß setzt erst fort - und hat somit die Gelegenheit zur Veränderung -, wenn die Kontrolle aus dem benachrichtigten Prozeß zurückkehrt.

Obwohl der synchrone Aufruf entfernt realisierter Methoden durch das von Corba zur Verfügung gestellte Rahmenwerk fast vollständig gekapselt wird, entsteht trotzdem zusätzlicher Aufwand bei der Programmentwicklung. Einher mit jeder Kommunikation gehen zusätzliche Fallunterscheidungen für die Verwaltung der kommunikationsbedingten Ausnahmesituationen. Der Kommunikationspartner muß ggf. ausgewählt und gefunden werden. Die Kommunikation kann unterbrochen sein und muß ggf. wieder aufgenommen werden. Diese Verwaltung auf der Metaebene muß in jedem Kommunikationsmodell berücksichtigt werden.

Aber gerade die Einfachheit der synchronen Anbindung stellt ebenso eine Schwäche dar. Zum einen spiegelt sich die Verteilung nicht in der Programmierung wider. Zweitens gibt es kein Verständnis über die Ausführungszeiten von bestimmten Methoden und schließlich sind externe *Fallback Mechanismen* notwendig.

1. Kein Widerspiegeln der Verteilung: Dem Programmierer wird nicht bewußt, welche Objekte im selben Prozeß laufen und welche in entfernten Prozessen existieren.
2. Kein Verständnis von Ausführungszeiten: Die Handhabung der Benutzerinteraktion (vgl. [Obe94], Seite 9,40) kann vom Programmierer nicht angemessen vorbereitet werden, da zum Entwicklungszeitpunkt kein Verständnis davon existiert, wie lange bestimmte Methoden benötigen. Das liegt vor allem daran, daß - wie unter Punkt eins genannt - nicht klar ist, wann eine Methode verteilt und wann direkt aufgerufen wird.
3. Externe *Fallback Mechanismen*: Da die Umsetzung der Verteilung ohne direkte Einbeziehung der konkreten Anwendungssituation geschieht, können auch nur allgemeine Ausfallmechanismen vorgesehen werden. Das bedeutet insbesondere, daß bei der Interaktion mit dem Benutzer im Fehlerfalle keine angemessene Unterstützung möglich ist.

Die synchrone Prozeßkopplung suggeriert direkten Zugriff, indem sie z.B. normale Methodenaufrufe durch automatisch erzeugte Kapseln ersetzt, in denen die Kommunikation abläuft. Jedoch sind aufwendige Mechanismen notwendig, wenn Kommunikation aus technischen Gründen versagt, weil sich innerhalb des vom Programmierer entwickelten Code keine dafür vorgesehenen Behandlungen befinden. Denn in der Programmierung spiegelt sich die Verteilung ja nicht wider.

Es ist ein Programmiermodell zu befürworten, in dem sichtbar gemacht wird, welche Objekte für lokale Exemplare stehen und welche aufgrund von Verteilung über Interprozeßkommunikation entfernte Exemplare vertreten. Es wird deshalb festgehalten:

Feststellung 1: Transparenz[21] im Programmiermodell

Es ist notwendig, daß das Verteilungsmodell im Programmiermodell durch geeignete Muster für den Programmierer deutlich wird. Eine explizite Handhabung von Verteilung muß im Programmcode sichtbar werden.

Nur durch eine klare Trennung der Arbeitsschritte, die im lokalen Bereich durchgeführt werden und derer, die nur durch Verteilung gelöst werden können, ermöglicht es, durch geeignete Visualisierungen auf eine Kommunikation mit einem entfernten System aufmerksam zu machen.

Es wird deshalb parallel zu der obigen Feststellung eine Konsequenz für das Benutzungsmodell abgeleitet:

Feststellung 2: Transparenz im Benutzungsmodell

Es ist notwendig, daß das Verteilungsmodell im Benutzungsmodell durch geeignete Metaphern für den Anwender deutlich wird. Eine explizite Handhabung von Verteilung muß durch Reaktionen und Anzeigen des Programms sichtbar werden.

Offenbar hat die Wahl des Verteilungsmodells nicht nur Einfluß auf das Programmiermodell, sondern hat seine Auswirkungen bis hin zum Benutzungsmodell. Wenn man deshalb die aufgestellten Forderungen bei der Auswahl des Verteilungsmodells berücksichtigt, kann es gelingen, eine adäquate Umsetzung der Verteilung im Programmiermodell und im Benutzungsmodell zu erreichen.

Wenn man diese Kriterien als Maßstab z.B. an eine synchrone Implementation von Corba anlegt, dann sieht man, daß im Design von Corba eine offensichtliche Schwäche angelegt ist: Dem Benutzer eines Dienstes wird nicht offenbar, mit welcher Distanz er auf diesen Dienst zugreift (selber Prozeß, selber Rechner, LAN oder WAN):

> *„Bei Corba ist es nicht vorgesehen, einen Dienst, mit der Einschränkung, daß er lokal zu dem Klienten läuft, anzufordern. Mit anderen Worten: Es ist nicht vorgesehen, den Raum, den der ORB für die Suche nach einem Dienst verwendet, in irgendeiner Weise (z.B. auf einen Rechner) zu beschränken.“ (vgl. [Fri96], Seite 15)*

Dies hat insbesondere Auswirkungen für die sogenannte lokale Verteilung, bei der versucht wird, ein Arbeitsplatzsystem in separate Prozesse aufzuteilen (dieser Punkt wird in diesem Kapitel später behandelt).

7.4.2 Asynchroner Kontrollfluß

Der asynchrone Kontrollfluß umfaßt all diejenigen Modelle, bei denen mehrere Prozesse quasi oder real nebeneinander ausgeführt werden. Beim asynchronen Kontrollfluß wird die Kontrolle bei der Kommunikation zwischen zwei Prozessen nicht an den jeweils

[21] Im Sinne von Nachvollziehbarkeit

anderen Prozeß abgegeben. Vielmehr läuft der Kontrollfluß im sendenden Prozeß ohne zeitliche Verzögerung weiter. Sowohl Sender als auch Empfänger unterhalten zusätzlich Ereignismechanismen, mit deren Hilfe sie über eintreffende Nachrichten informiert werden.

Definition 21. asynchron

Eine Kommunikation wird als asynchron *bezeichnet, wenn der Absender einer Nachricht seinen Kontrollfluß sofort nach dem Abgeben eines Versendeauftrags fortsetzen kann, ohne daß die Verarbeitung auf der Empfängerseite abgewartet werden muß. Beispiel: bei einem Telefongespräch können beide Partner jederzeit reden.*

Asynchroner Kontrollfluß wird von jedem modernen Betriebssystem angeboten und auch genutzt. So finden sich asynchrone Kontrollflüsse zum Beispiel bei *multi-threaded* Programmen, in denen Teile des Algorithmus parallel implementiert sind. Man spricht von einem *Thread* im Gegensatz zu einem Prozeß[22], wenn beide denselben Adreßraum benutzen, während Prozesse jeweils einen getrennten Adreßraum verwenden (vgl. [GS94], Seite 97ff). *Threads* können auf gemeinsame Variablen zugreifen; das ist bei Prozessen nicht möglich.

Im Bereich des *preemtive Multitasking*[23] wird der Kontextwechsel aber in beiden Fällen (*Thread* und Prozeß) direkt vom Betriebssystem nach Ablauf einer gewissen Zeitspanne durchgeführt oder vom Programm direkt vorgegeben. Die Kommunikation von *Threads* (wie auch zwischen Prozessen) kann nun einerseits über Semaphoren erfolgen, wodurch sich die Prozesse wenn erforderlich synchronisieren können oder sie verwenden, wie in Fenstersystemen üblich, eine Ereignisschlange, die von einem *Dispatcher* verwaltet wird, der die Kontrolle z.B. immer an denjenigen Prozeß übergibt, für den das aktuelle Ereignis bestimmt ist, wenn das vorhergehende abgearbeitet worden ist.

Bei einer Ereignisschlange stellt jeder *Thread* oder Prozeß ein Ereignis mit Ereignistyp, Absender, Empfänger und ggf. Ereignisdaten in eine Ereignisschlange ein. Verliert dieser Prozeß die Kontrolle, wird der Prozeß aktiviert, für den das nächste Ereignis der Schlange bestimmt ist. Dieses wird ihm zur Verfügung gestellt und kann ggf. abgearbeitet werden. Der Prozeß selbst kann neue Ereignisse in die Schlange einstellen. Diese werden allerdings erst bearbeitet, wenn alle anderen Ereignisse, die vorher in der Schlange standen, abgearbeitet worden sind. (Prioritätsorientierte Verfahren bevorzugen bestimmte Prozesse oder Ereignisse.)

Vorteil beider Konzepte, die hier vorgestellt wurden, ist, daß die Prämisse der Verteilung bereits im Programmierkonzept ihren Niederschlag findet: Der Programmierer muß sich an jeder Stelle seines Programm darüber klar werden, daß die Abarbeitung der Ereignisse zu einem (potentiell) späteren Zeitpunkt passiert, als der Code, der direkt hinter der Einstellung des Ereignisses steht.

Die Charakteristika eines asynchronen Kommunikationssystems sind ähnlich denen der oben beschriebenen Mechanismen. Kommunikation wird zwischen zwei Partnern auf die

[22] Threads werden auch als „light weight processes“ bezeichnet

[23] preemtive bedeutet, daß die Kontrolle einem Prozesse vom Betriebssystem entzogen werden kann (vgl. [GS94], Seite 134)

Weise aufgebaut, daß der Sender die Parameter einer Funktion in geeigneter Weise an das Kommunikationssubsystem übergibt und von diesem eine Quittung darüber erhält, daß diese entgegen genommen wurden. Der Kontrollfluß fährt im Sender fort. Zeitlich versetzt erhält der Empfänger die Parameter und kann sie (nun synchron) verarbeiten. Zu einem frei gewählten Zeitpunkt (z.B. zu Beginn oder am Ende der Verarbeitung) kann der Empfänger eine Antwort an den Sender schicken, die ebenso über das Kommunikationssubsystem vermittelt wird.

Auch hier wird nicht zwischen lokaler Verteilung und allgemeiner Verteilung unterschieden. Das Konzept erlaubt sogar die Zusammenführung eines für den Mehrprozeßraum entworfenen Systems zu einem Einzelprozeß, wenn z.B. eine Ereignisschlange für das Verteilen der Nachrichten verwendet wird.

Es ist festzuhalten, daß wegen der Trennung und Asynchronizität neue Synchronisationspunkte geschaffen werden müssen. Methodenaufrufe im Synchronen, die früher Rückgabewerte lieferten, sind jetzt zeitlich auseinandergerissen: Aufruf der Methode und Rückgabe eines Wertes sind zeitlich getrennt. Zwar wird durch das Subsystem die Beziehung zwischen den Kommunikationselementen hergestellt, für die Implementation bedeutet dies aber eine Fortentwicklung von der flußgeleiteten zur ereignisorientierten Struktur des Programms. Bei Fenstersystemen wurde diese Entwicklung bereits vollzogen: Werkzeuge sind im Bezug auf das Fenstersystem durch Ereignisse gesteuert. Nun erweitert sich dieser Gedanke auch in Richtung der Anbindung anderer Prozesse.

Die Konsequenzen der asynchronen Kopplung von Prozessen lassen sich folgendermaßen zusammenfassen:

- Weiche zeitliche Annahme über die Ausführung eines Requests
- Handhabung der Verbindung im Sinne von Aufbauen, Abbauen und Fehlerbehandlung ist notwendig
- Antworten können ausbleiben

Wenngleich die hier aufgeführten Konsequenzen im allgemeinen als Nachteile aufgefaßt werden und von den Verfechtern des synchronen Programmiermodells in der Verteilung als Schwächen dargestellt werden, kann diese Interpretation nicht akzeptiert werden. Vielmehr ist in der expliziten Handhabung der Verteilung ein Vorteil zu sehen, der sich in der Gestaltung der Software und ihrer Anwendungsfreundlichkeit widerspiegelt. Es werden deshalb in Ergänzung zu den oben genannten Konsequenzen an dieser Stelle auch die interpretierten Vorteile aufgezeigt.

Vorteile dieser Kommunikationsform:

- Im Programmiermodell wird bereits die Verteilung deutlich
- Asynchrone Handhabung ermöglicht (wenn auch ggf. eingeschränktes) Weiterarbeiten des Anwenders
- Asynchrone Prozeßkopplung ermöglicht parallele Ausführung der Prozesse ohne zwingendes Blockieren
- Fehlerbehandlung kann als Bestandteil der fachlichen und technischen Umsetzung eingebettet werden

In dem konkreten Industrieprojekt wurden mit Hilfe eines asynchronen Kommunikationsprotokolls die fachlichen Anforderungen umgesetzt. Durch die dabei gemachten Erfahrungen fühlt man sich in der oben aufgeführten Liste der Vorteile bestätigt.

Der nächste Abschnitt soll nun das firmeninterne Kommunikationsprotokoll darstellen und auf seine wesentlichen Eigenschaften und Konzepte hinweisen.

7.4.3 PSI (Micrologica)

Das Process System Interface (kurz PSI) (vgl. [Kra95]) ist eine von der Firma Micrologica entwickelte Middleware zur Interprozeßkommunikation. Es wird dazu einerseits ein Modell zur Verfügung gestellt, mit dem Prozesse betriebssystemunabhängig konstruiert werden können, und andererseits wird eine asynchrone Form der Kommunikation zwischen diesen Prozessen bereitgestellt. Die Prozeßkonstruktion erfolgt dabei mit Hilfe eines Modells von endlichen Automaten. Diese Möglichkeit soll hier nicht weiter beschrieben werden, da in den konkreten Projekten keinen Gebrauch davon gemacht wurde.

Das Kommunikationsmodell stellt eine Möglichkeit zur Verfügung, zwischen zwei Prozessen sowohl verbindungslos, als auch verbindungsorientiert (vgl. [Ker89], Seite 57) zu kommunizieren. Prozesse werden dabei durch ihre Prozeßidentifikatoren (kurz Prozeß-ID) unterschieden. Pro Adreßraum kann jede Prozeß-ID höchstens einmal vorkommen. Standardmäßig sind Prozesse nur im lokalen Prozeßraum, d.h. auf einem Rechner bekannt. Erst durch explizite Erweiterung des Adreßraums, durch sogenanntes Public-Erklären, können auch entfernte Prozesse die Kommunikation zu dem public-erklärten Prozeß aufnehmen.

Zwischen den Prozessen kann unter Angabe des Zielprozesses ein Datenpaket vom Sender zum Empfänger geschickt werden. Die Datenpakete werden einfach durch Speicherbereiche definiert. Als Empfänger stehen alle Prozesse zur Verfügung, die auf demselben Rechner oder in dem LAN aktiv sind. Dieses Kommunikationsmodell ist ausschließlich asynchron[24], d.h. der Programmablauf setzt sofort nach dem Aufruf der Versendefunktion fort und selbst eine sofort auftretende Fehlermeldung wird asynchron über den Meldungspuffer zur Verfügung gestellt. Deshalb spricht man bei diesem Kommunikationsmodell von *asynchroner* Ausführung.

Im allgemeinen wird ein wie oben beschriebenes Datenpaket vom Empfängerprozeß beantwortet. Man unterscheidet Datenpakete, die nicht beantwortet werden, Datenpakete, die einmal beantwortet werden und Datenpakete, die wiederholt beantwortet werden. Antworten werden - im Gegensatz zu initial versendeten Paketen - vom Kommunikationssystem automatisch an den richtigen Empfänger geleitet. Diese spezielle Designentscheidung wurde sich zunutze gemacht (s.u.)

In jedem Fall, also auch bei unbeantworteten Datenpaketen, wird der Absender darüber informiert, ob das Paket zugestellt werden konnte. Im Fehlerfall wird eine entsprechende Information mit Bezug auf das ursprüngliche Paket an den Absender übermittelt. Wird eine Antwort versendet, so ist auch hier der Bezug zum Ursprungspaket hergestellt.

[24] Es gibt auch eine Unterstützung für synchrone Kommunikation mittels RPCs, die aber (fast) gar nicht verwendet wird.

Außerdem wird dem Prozeß deutlich gemacht, ob diese Antwort die endgültige Antwort zu dem Datenpaket ist, oder noch weitere Antworten folgen. Denn in bestimmten Situationen ist es wünschenswert, daß neben der Antwort die z.B. ein Ergebnis trägt, vorher auch Antworten über den Status der Bearbeitung an den Urheber der Nachricht versendet werden.

Diese spezielle Form der Mehrfachantworten wurde dazu benutzt[25], um zwischen zwei Prozessen eine asymmetrische Verbindung herzustellen. Asymmetrisch deshalb, weil nur der eine Prozeß, der Klient, den anderen Prozeß, den Server, kennt. Interpretiert man ein Datenpaket als eine Anmeldung eines Klienten beim Server, im folgenden Anmeldepakete genannt, und nutzt die Möglichkeit aus, beliebig viele Antworten auf dieses Anmeldepaket zu schicken, dann hat man ein Verfahren gefunden, wie ein Server mit einem Klienten kommunizieren kann, ohne daß er dessen Prozeß-ID kennt. Nur der Klient muß die Prozeß-ID des Servers kennen, denn bei Antworten wird der Versand vom Kommunikationssystem PSI erledigt.

Das allgemeine Verfahren, Datenpakete eines Prozesses bei einem anderen Prozeß aufzubewahren, um die Kommunikation in die andere Richtung zu realisieren, obwohl hier keine public Prozeß-ID bekannt ist, wird „hängendes Paket“ genannt. Der Klient hängt beim Server ein Paket auf und nutzt die Fähigkeit des Kommunikationssystems, Antworten auf dieses Paket an den Absender zuzustellen.

Das Verfahren kann allgemein dazu benutzt werden, um sogenannte Monitorprozesse mit Servern zu verbinden. Unter einem Monitorprozeß wird bei Micrologica ein Werkzeug verstanden, welches den Zustand eines Automaten anzeigt. Beide stehen also in der oben beschriebenen Client-Server Beziehung. Diese spezielle Beziehung wird im nächsten Abschnitt näher untersucht.

7.5 Neues Programmiermodell

Die in der Literatur (vgl. [KGZ94], [RZ95], etc.) charakterisierten Anwendungssysteme bestehen aus einer Reihe von Werkzeugen, Materialien und Automaten. Dabei bearbeiten die Werkzeuge die Materialien und benutzen Automaten, um die Materialien abzulegen bzw. weiterzuverarbeiten. In diesem Abschnitt steht die Zusammenarbeit zwischen Werkzeugen und Automaten im Vordergrund. Insbesondere wird hierbei untersucht, wie eine Kopplung eines Werkzeugs an einen Automaten aussehen kann, wenn diese in getrennten Prozessen realisiert werden.

Anwendungssysteme wurden bisher in einem nicht nebenläufigen Prozeß zusammengefaßt. Im folgenden werden zwei Aufteilungen in separate Prozesse untersucht werden. Die Aufteilung von Werkzeugen und Automaten und die Aufteilung von Werkzeugen selbst. Zuerst betrachtet man, wie die Subsysteme eines Anwendungssystems, die zuvor in demselben Prozeß liefen, in unterschiedliche Prozesse auf demselben Rechner aufgeteilt werden.[26]

[25] Es handelt sich dabei um eine Standardtechnologie der Firma Micrologica

[26] Der Zwischenschritt über Threads wird hier nicht näher betrachten werden, weil sich die Erkenntnisse darauf leicht übertragen lassen. Multi-Threading bringt überdies keine weiteren Vorteile, wenn es darum geht, Verteilung zu betrachten.

Es wird ein Kommunikationsverfahren benötigt, mit dem die dann getrennten Prozesse in Verbindung treten können. Über diese Kommunikationsverbindung können sich die Prozesse dann finden (1), Ereignisse austauschen (2) und Zustände mitteilen (3).

Aus der oben beschriebenen Auswahl des Kommunikationsverfahrens (asynchron) und dem gewählten Kommunikationsprotokoll (PSI) kann man ein allgemeines Muster zur Kopplung von Bestandteilen eines Anwendungssystems im Mehrprozeßraum entwickeln, welches sich in die technische Umsetzung des Metaphernrepertoires der WAM-Methodik eingliedern läßt.

In diesem Abschnitt wird einerseits beschrieben, auf welche Weise Werkzeuge mit Automaten verbunden werden und welche Auswirkungen dies auf das Softwaredesign hat. Außerdem wird angegeben, wie man sich die Kopplung von unterschiedlichen Werkzeugen vorzustellen hat. Dazu wird ein Muster vorgestellt, von dem man annimmt, daß es Bestand hat und auch für weitere Anwendungssysteme genutzt werden kann.

Wird der Kopplungsbegriff in eine Modellwelt übertragen, in der mehr als ein Prozeß betrachtet werden muß, kommen weitere Dimensionen des Betrachtungsmodells hinzu. Objekte, die zuvor in einem Prozeßraum Synchronizität in der Kopplung ausnutzen konnten, müssen nun mit neuen Gegebenheiten zurechtkommen. Dies betrifft einerseits die zeitliche Dimension: Verändernde Funktionen haben nicht sofort ihre verändernde Wirkung, sondieren Funktionen können nicht sofort einen aktuellen Wert liefern. Ruft ein Objekt in einem entfernten Objekt eine verändernde Funktion auf, kehrt der Kontrollfluß noch vor der Durchführung der Veränderung zurück.

Andererseits kommt eine neue Dimension in bezug auf das mögliche Fehlverhalten hinzu. Jede Prozedur und jede Funktion kann potentiell zu einem Fehler führen, der nicht im fachlichen oder programmlogischen Bereich angesiedelt ist, sondern durch die Kommunikation verursacht wird.

Man nähert sich diesen zusätzlichen Dimensionen der Kopplung und deren technischer Lösung, indem zuerst der Begriff des Mehrprozeßraums definiert wird:

Definition 22. Mehrprozeßraum

Unter einem Mehrprozeßraum *versteht man den Raum, den ein Anwendungssystem aufspannt, wenn dessen Werkzeuge und Automaten in unterschiedlichen Prozessen laufen, die durch ein Kommunikationsmedium miteinander verbunden sind.*

Bezugnehmend auf die Definition des Anwendungssystems ist im Mehrprozeßraum also eine ganze Reihe von Prozessen relevant. Es wird allerdings reichen, direkt nur die Werkzeuge eines Anwenders zu betrachten. Hinzu kommen dann dynamisch die Automaten, die durch Werkzeuge des Anwenders benutzt werden. Obwohl die Definition des Mehrprozeßraums die Art des Kommunikationsmediums nicht näher bestimmt, wird implizit ein asynchrones Medium postuliert. Dies stellt eine allgemeinere Systemanforderung dar.

Durch die Berücksichtigung des Mehrprozeßraums ergibt sich eine neue Definition für die Kopplung im Asynchronen.

Definition 23. Kopplung (asynchron)

Unter dem Begriff Kopplung *wird die Art der Verbindung verstanden, mit der zwei Objekte über ein asynchrones Medium miteinander verknüpft sind. Hierbei werden generell drei Varianten unterschieden: Aufruf ohne Erwartung eines direkten Ergebnisses, Aufruf unter Erwartung eines (direkten) Ergebnisses, abstrakte Kommunikation auf Ereignisebene. In einem asynchronen Medium muß stets die Möglichkeit von Verbindungsfehlern berücksichtigt werden. Dies führt zu Kopplungsinformationen auf der Metaebene.*

In dieser Definition werden neue Aspekte einbezogen, die im folgenden erarbeitet werden sollen.

Werkzeuge wurden bisher mit den Sonden von Automaten über das Observer-Muster verbunden. Das Werkzeug kennt die Schnittstelle der Sonde (und die des Automaten) vollständig; die Sonde verwendet hingegen eine abstrakte Schnittstelle, die sich auf das Aussenden von vordefinierten Nachrichten erstreckt.

Gefordertes Ziel ist es nun, den Automaten und die Sonden in einem von dem Werkzeug potentiell separaten Prozeß zu betreiben. Dazu wurde das Proxy-Muster ([Gam94], Seite 207) gewählt. In diesem Abschnitt wird gezeigt, wie man motivieren kann, die technische Trennung auf die hier gezeigte Art durchzuführen.

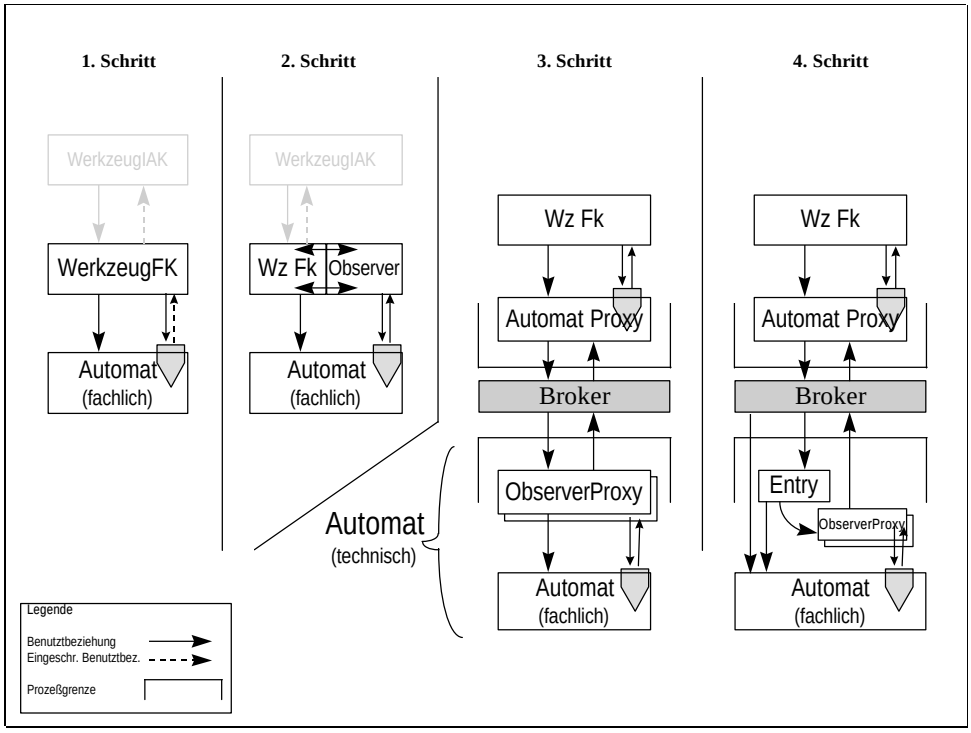

Abbildung 28 Genese der Trennung von Werkzeug und Automat in separate Prozesse

Die Abbildung 28 zeigt die vollzogene Entwicklung vom Einzelprozeß zu einem Mehrprozeßraum (von links nach rechts). Die grafische Darstellungsweise, eine Observer-Beziehung mit zwei Pfeilen darzustellen, bei denen der eine eine durchgezogene Linie besitzt und der andere eine gestrichelte, versucht auszudrücken,

daß in der einen Richtung die volle Schnittstelle benutzt wird und in der anderen Richtung nur ein Teil der Schnittstelle verwendet wird (Dieser Teil wird meist durch die Definition einer Klassenschnittstelle in einer Vererbungsbeziehung abgegrenzt).

Deshalb wird in der zweiten Phase das Verständnis präzisiert, indem die Werkzeug-FK in die zwei relevanten Objektteile aufgetrennt wird: die Implementation des Observer-Protokolls und die restliche Werkzeug-FK. Nun kann die Beziehung zum Automaten als zwei separate Benutztbeziehungen interpretieren werden.

In einem dritten Schritt wird die Prozeßtrennung über zwei Proxies modelliert. Jeweils das nicht vorhandene, weil im anderen Prozeß untergebrachte Objekt, wird durch ein Proxy ersetzt. Ein Automaten-Proxy mit einer Reihe von Sonden-Proxies befindet sich innerhalb des Werkzeugprozesses, da die Werkzeug-FK die volle Schnittstelle des Automaten und der Sonden benutzt. Und ein Observer-Proxy auf der Seite des Automatenprozesses, da die Sonden die volle Schnittstelle des Observers benutzen.

Die verwendete Ausprägung des Proxy wird von Gamma et al. *Remote Proxy* genannt:

> *A remote proxy provides a local representative for an object in a different address space. (vgl. [Gam94], Seite 208)*

In einem vierten Schritt (siehe Abbildung 28) wird die Erzeugung des Automaten und die Verknüpfung der Werkzeug-FK und der Sonden genauer betrachtet. Im Einprozeßraum erzeugt die FK den Automaten und registriert sich direkt bei den Sonden über das Observer-Protokoll (vgl. [WW94]). Im Mehrprozeßraum muß dieser Vorgang ebenfalls angemessen modelliert werden. Man unterscheidet dabei zwischen Erzeugung und Registrierung. Der Aspekt der Erzeugung des Automatenprozesses wird außerhalb des Betrachtungskontextes verlagert; man geht davon aus, daß der Prozeß bereits läuft.[27] Die Registrierung ist der Vorgang, bei dem sich das Werkzeug bei den Sonden dafür anmeldet, mittels Ereignissen über die Veränderung des Zustandes informiert zu werden.

Das Objekt, an welches sich ein Klient bei einem Automaten wenden muß, um sich für Ereignisse zu registrieren, wird *Entry*[28] genannt. Es ist notwendig, daß dieser Teil der Schnittstelle immer existiert, damit die Kommunikation aufgebaut werden kann. Dieser Eingang stellt einen geeigneten Mechanismus zur Verfügung.

Letztendlich - in einem fünften Blick auf die Beziehung zwischen Automat und Werkzeug - werden die einstellenden Funktionen und die Benachrichtigungen über Zustandsänderungen betrachtet. Alle einstellenden Funktionen der Werkzeuge werden mittels des Automaten-Proxy an den *Entry* des Automaten-Prozesses gesendet. Dieser ruft direkt eine Methode im Automaten auf, die zu einer Zustandsveränderung führen kann. Eine Rückmeldung über einen aufgetretenen Fehler kann direkt an den Absender geschickt werden. Allen Observern wird im Falle einer Zustandsänderung im Automaten eine Benachrichtigung zugeschickt. Das ist auch der Grund weswegen die Einstellungen prinzipiell anonym vorgenommen werden.

---

[27] Es wird wahrscheinlich eine übergeordnete Instanz benötigt, die - wie eine Sub-FK über Chain-of-Responsibility an die Kontext-FK meldet, daß ein neues Werkzeug erzeugt werden soll - das Starten von Automaten und Werkzeugen durchführen kann. Dazu mehr im Kapitel zum Umgebungsprozeß

[28] engl. für Eingang, Eingangstür

In diesem Modell steht das Automaten-Proxy für die Verbindung zwischen den beiden Prozessen auf der Seite des Werkzeugs. Es wird deshalb zur Reduzierung von Verbindungen und deren Überwachung keine einzelne Verbindung zwischen jeder Sonde und den korrespondierenden Proxies erzeugt. Das Automaten-Proxy bündelt diese Kommunikation.

Es wird noch einmal zusammengefaßt:

- Auftrennung der Objektbeziehung auf die benutzten Schnittstellen
- Ersetzung der Objekte durch ein Proxy auf jeder Seite
- Bereitstellung eines Entries zur initialen Verknüpfung
- Realisierung der Einstellungen im Entry und der Benachrichtigungen im Observer-Proxy

Damit konnte das Proxy-Muster in dem Sinne eingesetzt werden, daß die relevanten Objektbestandteile der an der Kopplung beteiligten Objekte in einen anderen Prozeß ausgelagert und durch lokale Objekte ersetzt werden konnten. Im weiteren Verlauf wird gezeigt werden, daß diese Anwendung des Proxy-Musters nicht vollständig ausreichend ist, um für den Einsatz in Realsystemen zur Kopplung von getrennten Prozessen eingesetzt werden zu können.

Wie man oben gesehen hat, gibt es drei grundsätzlich unterschiedliche Arten der Kopplung. Alle drei Varianten findet man in der Verteilung von Werkzeugen und Automaten wieder. Sie haben dabei grundlegende Elemente in der Nutzung des Kommunikationsmediums gemeinsam.

Zum technischen Aufbau der Kommunikation muß der initiierende Prozeß Kenntnis darüber haben, mit welchem anderen Prozeß er kommunizieren möchte, um einen gewissen Dienst nutzen zu können. Dazu wurde ein Dienstname spezifiziert. Da Prozesse im lokalen Prozeßraum nur über ihre Prozeß-ID bekannt sind - diese wird vom Kommunikationssystem vergeben -, gibt es einen speziell ausgezeichneten Prozeß, bei dem sich Prozesse unter einem Namen registrieren können, um den von ihnen angebotenen Dienst über diesen Namen öffentlich zu machen. Die Prozeßnummer des sogenannten Nameserver ist feststehend. Der initiierende Prozeß fragt den Nameserver nach einem Dienst, indem er dessen Namen angibt, und erhält dazu die Prozeß-ID des Prozesses, der sich unter diesem Namen registriert hat.

Definition 24. Nameserver

Mit Hilfe eines (lokalen) Nameservers werden vom System vergebene Prozeß-IDs durch Namen ersetzt.[29] *Prozesse können den Nameserver unter Angabe eines Namens nach einer Prozeß-ID fragen. Jeder Prozeß kann sich unter beliebig vielen, aber verschiedenen Namen mit seiner Prozeß-ID beim Nameserver registrieren.*

Durch das Konzept des Nameservers konnte eine weitere Ebene der Entkopplung in diesem Industrieprojekt eingeführt werden. Während in der vorangegangenen Entwicklungsphase die Prozeß-IDs der Prozesse als fix angenommen wurden, können

[29] Dieser Dienst ist notwendig, weil die vom System vergebenen Prozeßnummern bei jedem Start neu vergeben werden.

jetzt vom System frei zugewiesene Prozeß-IDs verwendet werden. Dadurch sind alle Nachteile, die durch fest gewählte Prozeß-IDs entstehen können (versehentliche Doppelbelegung, Notwendigkeit eines Neustarts des gesamten Rechners wegen eines fehlerhaften Prozesses, etc.) im lokalen Bereich entfernt worden.

Außerdem bietet sich jetzt der Vorteil, daß Kommunikation nun auf der Ebene von Diensten und nicht von Prozessen möglich ist. Diese Abstraktion erlaubt es, die Entscheidung, in welchem Prozeß ein Dienst implementiert ist, bis zur Laufzeit hinauszuzögern.

Das im konkreten Projekt verwendete Nameserver-Konzept wurde ausschließlich lokal, d.h. jeweils bezogen auf einen Rechner, verwendet. Es bestand lediglich die Notwendigkeit, auf einem Gerät die Prozesse in der oben genannten Weise zu entkoppeln. Es ist aber leicht aufzuzeigen, daß dieses Nameserver-Konzept ebenso im Bereich einer Domäne und sogar darüber hinaus Anwendung finden kann (vgl. [OMG94]).

7.5.1 Werkzeug - Automat

Es wird nun untersucht, wie die Ergebnisse der oben durchgeführten Ausarbeitung von Anforderungen in ein konkretes Anwendungsfeld übertragen werden können.

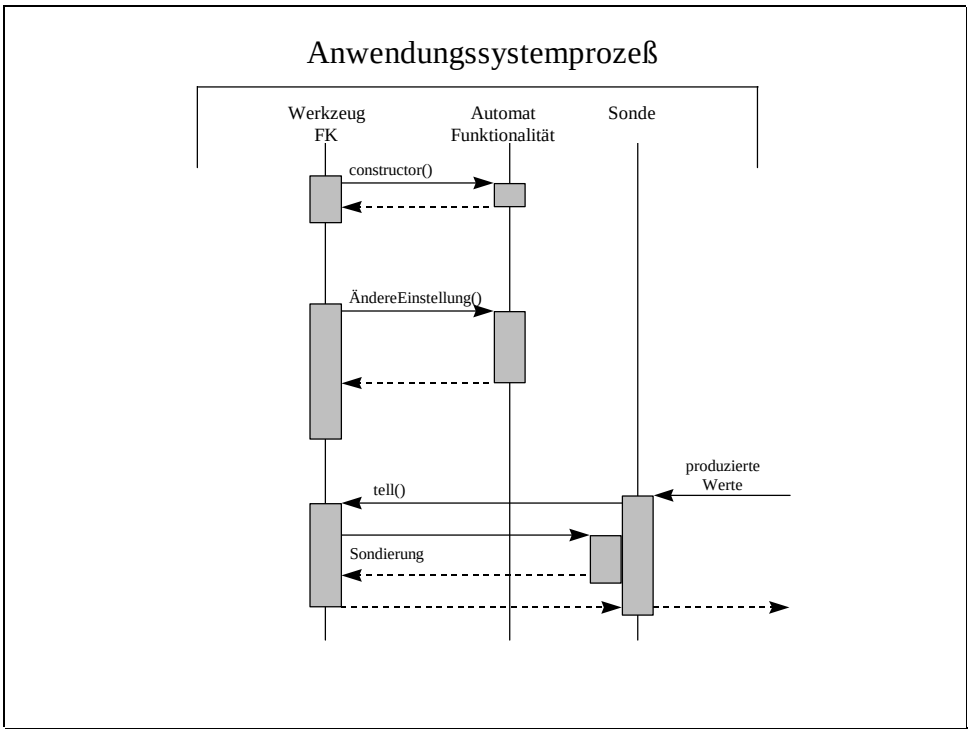

Abbildung 29 Synchrone Interaktion bei Kopplung von Werkzeugen und Automaten/Sonden im Einprozeßraum

Vorausgesetzt wird, daß der Automatenprozeß bereits gestartet wurde und sich beim Nameserver unter seinem Dienstnamen registriert hat. Das Werkzeug ermittelt nun nach oben beschriebener Art seinen Kommunikationspartner.

Das Werkzeug verwendet zur Kommunikation mit dem Automaten (und auch dem Nameserver) ein entsprechendes Automaten-Proxy, in dem die Kommunikation gekapselt wurde. Es ist daher nicht notwendig, die (technisch orientierte) Kommunikationsschnittstelle des Automaten zu kennen. Lediglich die Schnittstelle des Proxy ist relevant. Und diese ist nach der Idee eines Proxies identisch mit der fachlichen Automatenschnittstelle, wenn der Automat im selben Prozeß als Objekt realisiert worden wäre:

> *Proxy ... provides an interface identical to [the][30] Subject's so that a proxy can be[31] substituted for the real subject. (vgl. [Gam94], Seite 209)*

Allerdings wurde beim Umsetzen dieses Ansatzes klar, daß im Rahmen einer fachlich motivierten Unterstützung des Anwenders die Eins-zu-Eins-Umsetzung des Proxy-Konzeptes nicht zu einem zufriedenstellenden Ergebnis führte. Begründet ist diese Erkenntnis vor allem in einer fehlenden Transparenz (im Sinne von Erkennbarkeit) bzgl. des Verbindungszustandes. Dieser Problematik wurde durch eine Erweiterung der Schnittstelle des Proxies begegnet.

Bereits hier zeichnet sich ab, daß die bis jetzt übliche Implementationstechnik nicht mehr aufrecht erhalten werden kann. Bisher werden Methoden des Automaten und somit auch des Automaten-Proxy als Funktionen aufgefaßt, von denen im allgemeinen ein Rückgabewert erwartet wird. Nun, um deutlich zu machen, daß man zwischen demselben Rechner und einem entfernten nicht prinzipiell unterscheidet, wird diese funktionsorientierte Implementation zu Gunsten einer ereignisorientierten Schnittstelle geändert.

Auf der Grundlage der obigen Erkenntnisse, läßt sich die Schnittstelle des Proxy grob in drei Teile unterteilen: Der erste Teil sind die einstellenden Operationen. Als zweiten Teil gibt es sondierende Operationen. Diese beiden Teile ergeben sich durch eine einfache Untergliederung der üblichen Automatenschnittstelle. Der dritte Bestandteil der Schnittstelle sind die Benachrichtigungen (Events) über den Zustand des Proxies mittels des Observer-Musters. Hierüber werden Ereignisse zur Verfügung gestellt, die Auskunft über unterschiedliche Gruppen von Zuständen des Automaten geben.

Die Entwicklung des Proxy:

- Proxy im Sinne des von Gamma formulierten Stellvertreters als Substitut für ein Objekt in einem anderen Adreßraum
- Reduktion der Schnittstelle des Proxy auf die in diesem abgegrenzten Fall notwendige Funktionalität im Gegensatz zum vollständigen Funktionsumfang des Automaten
- Kopplung der Funktionskomponenten und des Automaten über das Observer-Muster mit lokaler Sondierung (lokales Abbild des Automatenzustandes)
- Einführung von zusätzlichen Meta-Ereignissen zur Signalisierung des Zustandes der Verbindung

[30] Durch den Autor ergänzt
[31] Im Original „by“: trotz seines schlechten Englisch ist der Autor davon überzeugt, daß hier „be“ stehen sollte

Besonders wichtig sind dabei die beiden letzten Punkte. Das Automaten-Proxy und die dazugehörigen Sonden bilden den Zustand des Automaten lokal ab. Dies erfolgt zur Sicherstellung eines konsistenten Sondierungsvorgangs. Dadurch, daß nur die registrierten Sonden abgebildet werden, wirkt sich dieses Verfahren nur minimal negativ auf den Speicherverbrauch aus.

Die Einführung von Meta-Ereignissen, in das Protokoll zwischen Automaten-Proxy und Funktionskomponente trägt dem Umstand Rechnung, daß im Rahmen der Kommunikation Verbindungsabbrüche auftreten können (siehe Abschnitt Fehlerbehandlung).

Aus der Gestaltung des Proxy leitet sich das folgende Interaktionsdiagramm ab, welches die Realisierung des Werkzeugs und des Automaten in unterschiedlichen Prozessen berücksichtigt:

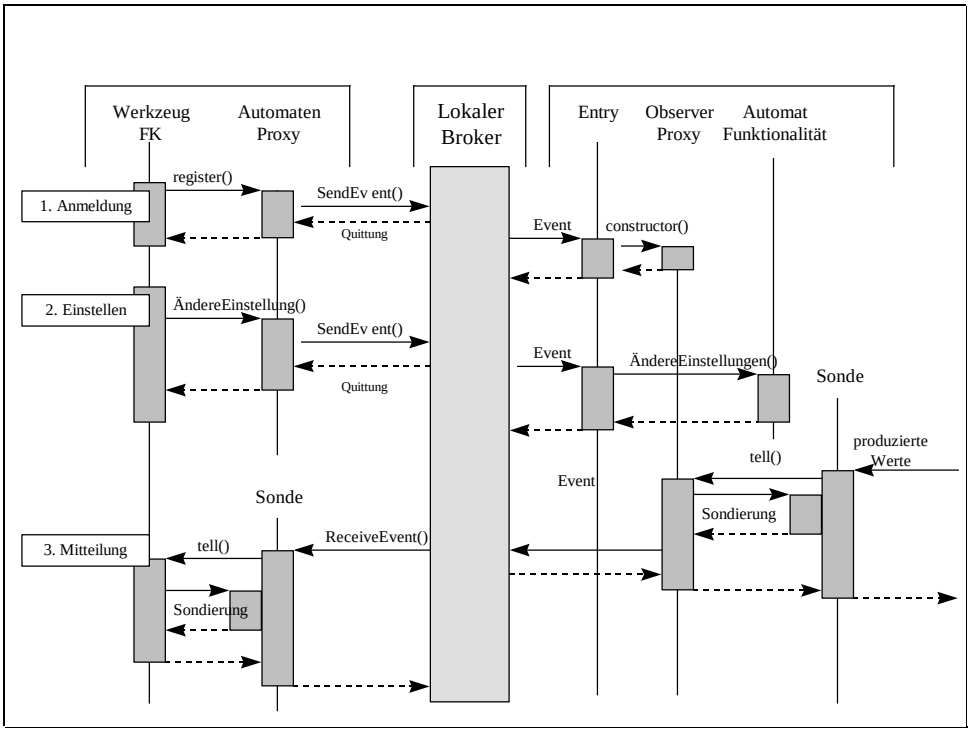

Abbildung 30 Einbeziehung der asynchronen Kommunikation in die Objektinteraktion

7.5.1.1 Interaktionsdiagramme für IPK

Die Grafik (Abbildung 30) verwendet als Grundlage für die Darstellung der Interprozeßkommunikation (IPK) die von [Jac94] eingeführten Interaktionsdiagramme. Als Erweiterung wurden die von [Lil95] ergänzten Grafikelemente verwendet. Zusätzlich sind zwei Elemente hinzugefügt worden: Erstens werden die Objekte eines Prozesses mit Hilfe einer oberen Klammer zusammengefaßt und als zusammengehörig gekennzeichnet. Es kann also mehrere Prozesse geben, die miteinander kommunizieren. Zweitens wird ein Broker-Objekt eingeführt, mit dessen Hilfe die Trennung durch Interprozeßkommunikation deutlich gemacht werden soll.

Definition 25. Broker

Unter einem Broker *(nächstmöglicher Meldungspuffer) versteht man eine Instanz, der von einem Prozeß Meldungen übergeben werden können, damit diese an einen anderen Prozeß weitergeleitet werden.*

Die Abbildung 30 zeigt im oberen Bereich, wie die Funktionskomponente sich mittels des Proxy beim entfernten Automaten registriert. Bereits die Registrierung ist ein asynchroner Vorgang, der mit einer zeitlichen Verzögerung durchgeführt wird. Der Automat empfängt eine Aufforderung zur Registrierung. Daraufhin erzeugt dieser dafür ein Observer-Proxy. Das geschieht nach einem einheitlichen Muster, welches anhand der Einstelloperation erläutert werden soll. Sollte die Registrierung technisch nicht möglich sein, weil z.B. die Netzwerkkommunikation zusammenbricht, wird der Klientprozeß durch ein Fehlerpaket benachrichtigt. Das geschieht nach Ablauf von durch das Kommunikationssystem bedingten Timeouts über den Fehler im Proxy-Objekt. Das Proxy setzt das in ein Event um. Hat sich die Funktionskomponente für Events bzgl. des Verbindungsstatus registriert, kann sie nun geeignet darauf reagieren.

Im 2. Fall „Einstellen“ der Abbildung 30 nimmt die Funktionskomponente eine Einstellung am Automaten vor. Es wird die einstellende Operation am Proxy aufgerufen. Das Proxy erstellt daraus ein Datenpaket und versendet dies über das Kommunikationssystem. Der Kontrollfluß kehrt in die FK zurück.

Zeitlich versetzt wird das Datenpaket an den Automaten zugestellt. Die Daten gelangen hier in das Entry und führen zu einem Aufruf der einstellenden Operation im Automaten. Dieser nimmt ggf. die geforderten Einstellungen vor. Der Kontrollfluß kehrt zurück in das Entry-Objekt und dann in das Kommunikationssystem. Kommunizieren auf diese Weise mehrere Klienten mit dem Automaten, werden die Aufforderungen zur Einstellungsänderung durch das Kommunikationssystem serialisiert. Jede Änderung führt ggf. zu Benachrichtigungen über Zustandsänderungen. Nur die Ausführung aller Änderungen hintereinander bestimmt den letzten Zustand, der jedem Klienten mitgeteilt wird.

In der anderen Richtung gibt es ebenfalls Kommunikation. Gesetzt den Fall, daß die Einstellung zur Änderung des fachlichen Zustands im Automaten führt, wird dieser die Werkzeuge über die Observer-Proxies benachrichtigen. Die Automatenfunktionalität ruft deshalb im Observer-Proxy eine Operation auf, die im Proxy zu Parametern umgesetzt wird, welche dieses an das Kommunikationssystem übergibt.

Zeitlich versetzt führt dies im Automaten-Proxy zu einem Aufruf, der den Empfang von Daten signalisiert. Über die vom Proxy benutzte objektorientierte Kapselung der Kommunikationsschnittstelle wird eine automatische Zuordnung zwischen empfangenen Daten und der Anmeldung am Automaten durchgeführt. Dieser Mechanismus stellt die notwendige Infrastruktur zur Verfügung, so daß davon ausgegangen werden kann, daß bereits der gewünschte Kontext für die Auswertung der empfangenen Daten hergestellt worden ist.

Das Automaten-Proxy interpretiert den Inhalt der Kommunikation und ändert den internen Zustand. Als Ergebnis wird eine Benachrichtigung durchgeführt an alle beobachtenden lokalen Objekte, für die sich z.B. die Werkzeug-FK registriert hat. Diese kann mit sondierenden Operationen den Zustand des Automaten am Proxy abfragen,

denn das aktuelle Abbild des Automatenzustandes ist im Proxy vorhanden. Im Sinne der Betrachtungsweise läßt sich diese Zustandssondierung als synchrone Kommunikation interpretieren, die alle geforderten Konsistenzkriterien gewährleistet. Der Kontrollfluß kehrt danach in das Proxy und danach in das Kommunikationssystem zurück. Hier findet die Bearbeitung der Kommunikation ihr Ende und damit wird die Kontrolle wieder an den Benutzer des Werkzeugs zurückgegeben.

Die Möglichkeit, in einem Prozeß sowohl aktiv, als mittelbare Reaktion auf Benutzereingaben zur Einstellungsänderung, Kommunikationspakete zu versenden, und gleichzeitig für den Empfang von Kommunikationspaketen offen zu sein, wie es für die Anzeige von neuen Automatenzuständen notwendig ist, ist eine Besonderheit des gewählten Kommunikationssystems. Der Empfang von Paketen wird hier durch eine besondere Integration in die Ereignisschlange des Fenstersystems auf dieser Ebene gleichwertig behandelt.

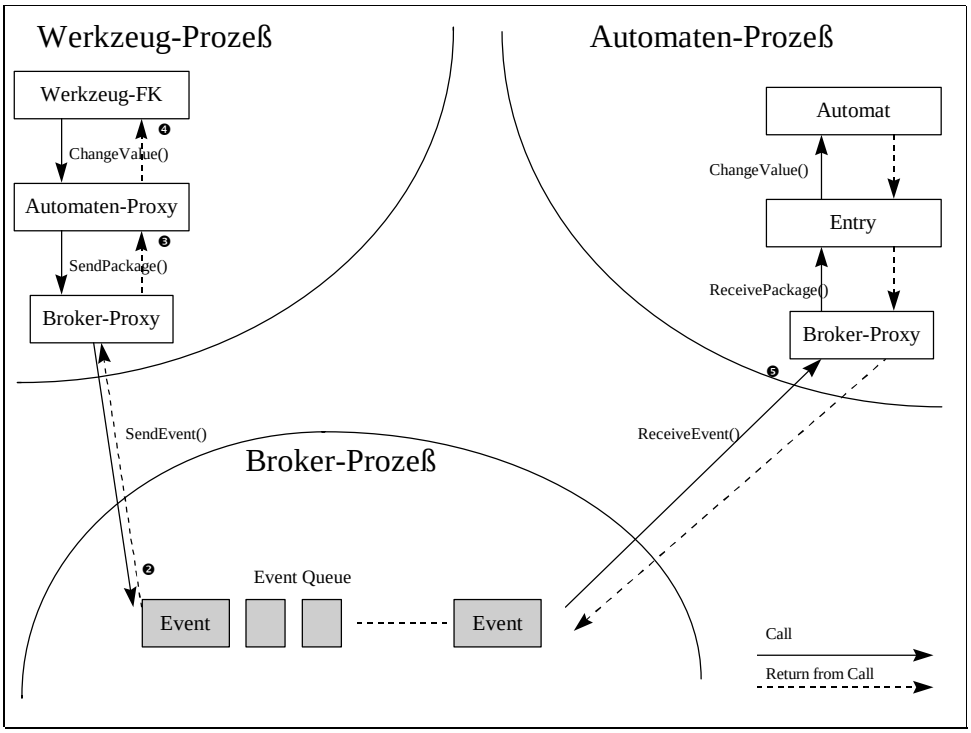

Abbildung 31 Asynchrone Prozeßkommunikation an einem Ablaufbeispiel

Das Beispiel in Abbildung 31 zeigt exemplarisch, wie der Kontrollfluß innerhalb der drei an der Kommunikation beteiligten Prozesse verläuft. Die mit runden Linien abgegrenzten Bereiche stellen jeweils separate Prozesse dar.

7.5.2 Werkzeug - Werkzeug

Die Kommunikation zwischen Werkzeugen innerhalb eines Anwendungssystems wird beim Design nach WAM immer indirekt durchgeführt (vgl. [Rie94], [WW94], [Gry96]). Dabei sind grundsätzlich zwei Ebenen zu unterscheiden. Bei der Beziehung zwischen Werkzeug und Subwerkzeug findet eine direkte Kopplung statt. Diese wird zum Beispiel (vgl. [Rie95], Seite 55) über einen Ereignismechanismus durchgeführt. Im Fall der

Beziehung zweier Kontextwerkzeuge, also nicht auf der Ebene von Teil und Ganzem, findet die Benachrichtigung über die Vermittlung einer dritten Instanz (den Ereignisverwalter (vgl. [Gry96], Seite 145)) statt.

Technisch konkret realisiert ist die Kommunikation zwischen der Kontext-FK und den Sub-FKs durch das Observer-Muster, welches die Kommunikation zwischen den Werkzeugen reguliert. In Abhängigkeit von dem Einsatzkontext bieten sich aber auch andere softwaretechnische Muster an, wie z.B. das Chain of Responsibility-Muster (Pattern in [GHJ+95], Seite 223) oder eine häufig auftretende Kombination aus Mustern, wie das Bureaucracy-Muster in [Rie96].

Die Kommunikation zwischen zwei (gleichberechtigten) Kontext-FKs wird ebenfalls über das Observer-Muster realisiert, allerdings unter der Zuhilfenahme des Ereignisverwalters (s.o.). Dieser tritt im Sinne eines Mediators (vgl. [GHJ+95], Seite 273) auf. Deshalb spricht man hierbei von einer indirekten Kopplung, weil ein vermittelndes Objekt zwischen die Kontext-FKs tritt.

In beiden Fällen ist zu überlegen, ob die Möglichkeiten der asynchronen Prozeßkopplung genutzt werden können. Sowohl die Beteiligten einer Beziehung Werkzeug zu Subwerkzeug können in unterschiedlichen Prozessen angesiedelt werden, wie die Kommunikationspartner einer Werkzeug zu Werkzeug Beziehung. Der Erfahrung nach ist die Prozeßtrennung von Kontext-FK und Sub-FK nicht sinnvoll. Dafür spricht nicht nur das Effizienzkriterium einerseits, sondern auch die enge fachliche Bindung. Im allgemeinen sind Kontext-FK und Sub-FKs so eng miteinander verknüpft, daß die Kontext-FK nicht ohne die Existenz der Sub-FKs arbeiten kann. Dies läßt sich im Mehrprozeßraum nicht garantieren. Während auf der Ebene der Kontext-FKs keine enge Verknüpfung stattfindet.

Man hat sich deshalb nur dafür entschieden, die Trennung von Werkzeugen, repräsentiert durch ihre Kontext-FKs, zu modellieren, während man ganz bewußt darauf verzichtet, technische Muster für die Trennung von Kontext-FK und Sub-FKs anzugeben. Im folgenden werden deshalb nur die Fälle angesprochen, in denen Werkzeuge miteinander in Kommunikation treten.

Eine indirekte Kommunikation von Werkzeugen kann über Materialien und Automaten durchgeführt werden. Werkzeuge stehen dann über ein Material in Kommunikation, indem ein Werkzeug das Material bearbeitet und ein anderes über dessen Veränderungen informiert wird. In diesem Fall löst ein Automat, der Materialverwalter, die Kommunikation auf (vgl. [WW94]). Allgemeiner formuliert findet ein Teilzustand eines Automaten im Rahmen der Beobachtung durch Werkzeuge deren Aufmerksamkeit. Ausgewählte Werkzeuge können Einstellungen am Automaten vornehmen und lösen damit eine Benachrichtigung anderer Werkzeuge aus.

Diese Art der Kommunikation wird von dem vorgestellten Modell unterstützt, denn die Verknüpfung von Werkzeug und Automat ist bereits realisiert. Dadurch ist auch die indirekte Kommunikation von Werkzeug über Automat zu einem anderen Werkzeug hergestellt.

Gryczan nennt in seiner Arbeit andererseits die Umgebung als vermittelnde Instanz bei der Kommunikation von Werkzeugen untereinander (vgl. [Gry96]). Sie wird als Mittler zwischen Ereignis aussendenden Werkzeugen und dem Ereignisverwalter selbst

eingestuft. Ereignisse, die von einem Werkzeug stammen und in der Umgebung nicht ausgewertet werden können, werden dem Ereignisverwalter übergeben, um diese an andere Werkzeuge weiter zu vermitteln.

Im Kontext des referenzierten Industrieprojektes wurden explizit Kommunikationsklassen herausgearbeitet, die durch Werkzeuge initiiert über die Umgebung weitergereicht wurden. Allerdings wurden dafür gänzlich andere Kopplungsmechanismen verwendet, als die von Gryczan vorgeschlagen wurden. Deshalb wird hier ein allgemeines Muster für die Realisierung eines Ereignisverwalters angegeben. Weitere Details zu diesem Aspekt finden sich im Kapitel über den Umgebungsprozeß.

7.5.3 Fehlerbehandlung

Bei der synchronen Kopplung von Komponenten ist die Fehlerbehandlung in klar getrennte Bereiche unterteilbar. Schwerwiegende Fehler stellen z.B. mißlungene Versuche dar, ein neues Objekt anzulegen (Speicherprobleme). Normale Fehler sind z.B. Funktions- oder Methodenaufrufe, bei denen die Ausführung der Funktion bzw. Methode aufgrund der Rahmenbedingungen (z.B. Vorbedingungen, bzw. Vertragsmodell (vgl. [Mey94])) abgelehnt werden mußte. Diesen Fehlern kann mit geeigneten und bereits untersuchten Strategien begegnet werden.

Durch die Einführung der Verteilung kommt eine weitere Fehlerdimension hinzu, die sich nicht auf dieselbe Weise statisch lösen läßt. Das Kommunikationsmedium kann die Verbindung zwischen zwei Prozessen, auch nachdem bereits erfolgreiche Kommunikation durchgeführt wurde, bei späteren Versuchen als gescheitert melden. Es gibt grundsätzlich keine Garantie dafür, das eine bisher als gesichert geltende Verbindung zwischen zwei Prozessen, beim nächsten Versuch noch funktionsfähig ist.

Die Ursachen für diese potentielle Instabilität liegen darin begründet, daß einerseits ein System aus Treibern, Prozessen und Libraries für die Verbindung verantwortlich ist, welches selbst fehlerträchtig und anfällig ist. Andererseits wird als Kommunikationsmedium eine Hardware verwendet, deren physische Verbreitung und Qualität nur bedingt kontrollierbar ist.

Diese Voraussetzungen, die Unberechenbarkeit und die physische Komponente, machen es erforderlich, daß eine andere Strategie für die Behandlung von Kommunikationsfehlern angewendet wird, als sie bei den bekannten Fehlerquellen möglich war.

Bei der funktionsorientierten Programmierung steht die Frage nach Fehlerbehandlung erst in zweiter Linie. Es kann als gesichert angenommen werden, daß alle Funktions- bzw. Methodenaufrufe durchgeführt werden können. Lediglich das Ergebnis eines Methodenaufrufs kann ggf. in Abhängigkeit vom Zustand oder den Eingabeparametern nicht erfolgreich berechnet werden. Beim Übergang von der synchronen Kopplung zu einer asynchronen und vom direkten Funktionsaufruf zu einer verzögerten Ausführung gekapselt durch Proxies entsteht eine neue Dimension der Fehlerbehandlung.

In einem Mehrprozeßraum kann es nun auftreten, daß die durch ein Proxy gekapselte entfernte Komponente nicht mehr erreichbar ist, weil sie selbst oder das Netzwerk über das der Prozeß mit ihr verbunden war, defekt ist. In diesem Fall wird der Broker einige

Versuche unternehmen, bevor er das Kommunikationspaket als „nicht erfolgreich verschickt“ an den Absender mit einem Fehlerstatus zurückgibt.

Bemerkenswertes Charakteristikum dieser Fehlerklasse ist die zeitliche Verzögerung, mit der die Fehler gemeldet werden. Während bei Speichermangel das Erzeugen eines neuen Objektes sofort mit einem Fehler gemeldet wird, entsteht hier die Notwendigkeit, den Kontext so lange aufzubewahren, bis eine positive Bestätigung, eine Fehlermeldung des Transportmediums oder ein Timer abgelaufen ist. Der Zeitgeber bzw. Timer wird deshalb notwendig, damit kein unzumutbares oder annähernd unendliches Warten bei der Kommunikation zweier Prozesse ermöglicht wird. Für jede Kommunikation muß ein *Schwellenwert* definiert werden, nach dessen Überschreitung die Kommunikation als gescheitert angesehen wird, auch wenn die theoretische Annahme möglich wäre, daß die Kommunikation noch zustande kommt.

Definition 26. Schwellenwert

Bei der Umsetzung von analogen Signalen in digitale Signale wird ein bestimmter Wert des analogen Signals als Schwellenwert definiert. Übersteigt die Amplitude des analogen Signals diesen Wert, so wird daraus ein digitales Signal erzeugt, das so lange anhält, bis das analoge Signal wieder unter den Schwellenwert gesunken ist. (vgl. [Sch89], Seite 2378)

Während man bei sondierenden Funktionen diese Fehlersituation vermeiden kann, da die Funktionen einen konkreten Wert als Rückgabewert lieferten, eröffnet das ereignisorientierte Programmiermodell diese neue Fehlerklasse. Insbesondere bei einstellenden Methoden am Automaten kann diese Fehlersituation nicht vermieden werden. An dieser Stelle wird deutlich, daß es sich um verteilte Komponenten handelt. Dies schlägt sich im Programmiermodell nieder und somit beim Design der Klassen und wird darüber hinaus für den Anwender sichtbar.

7.5.3.1 Technische Lösungsansätze

Die technische Fehlerbehandlung bei gekapselter Kommunikation kann auf zwei Stufen geschehen. Die erste Stufe ist der noch synchrone Abschnitt, bei dem der Auftrag an den Partner formuliert wird. Die zweite Stufe ist danach angesiedelt; hierzu wird jede Fehlersituation gezählt, die erst verzögert erkannt werden kann.

Im ersten Fall kann eine Rückmeldung sofort erfolgen. Bereits bei der Übergabe des Versandauftrages an den Broker steht fest, ob der Auftrag überhaupt entgegen genommen wird. In Bezug auf Abbildung 31 wird also bereits bei (2) festgestellt, daß der Partner nicht verfügbar ist. Deshalb liefert (3) einen Fehler zurück. Das kann vom Proxy (4) sofort an den Aufrufer signalisiert werden.

In jedem anderen Fall, d.h. der Versandauftrag wurde vom Broker entgegengenommen, wird vorgeschlagen, daß ein Timer verwendet wird. Dieser gibt einen Schwellenwert dafür an, wann eine Kommunikation nicht mehr erfolgreich beantwortet wurde. Das führt zu einem Abbruchauftrag an das Kommunikationssystem. Darüber hinaus wird im Proxy die Verbindung zum Partner als getrennt markiert und dies allen Beobachtern gemeldet.

Definition 27. Timer

Timer *ist ein Prozeß, der Weckaufträge entgegennimmt. Das heißt, er schickt dem auftraggebenden Prozeß zu einem bestimmten Zeitpunkt eine Nachricht, ein sogenanntes Weckpaket. (vgl. [Kra95], Seite 24)*

Es ist deshalb notwendig, daß die beobachtenden Komponenten sich auch für Ereignisse, die die Kommunikation betreffen, registrieren und darauf angemessen im Sinne der Benutzerfreundlichkeit reagieren.

An dieser Stelle mag man versucht sein, die Art der Verteilung in Kategorien zu unterscheiden: wie z.B. in Hochgeschwindigkeitsnetze und solche Netze, die mit langsamer Kommunikation arbeiten. Ebenso könnte man zwischen Verteilung auf einem Rechner, sogenannter lokaler Verteilung und Verteilung über mehrere Rechner unterscheiden.

Diese Trennung ist so leider nicht möglich, da auch lokale Netze keine akzeptablen Antwortzeiten bieten können, wenn die Netzlast über ein bestimmtes Maß wächst. Deshalb ist die Akzeptanz ausschließlich von der aktuellen Antwortzeit in jedem einzelnen Fall und nicht von der technischen Netzrealisierung abhängig. Genauso wie ein langsamer Prozessor mit einem komplexen Betriebssystem keine erträgliche Antwortzeit liefert, wird auch eine entsprechende Verzögerung bei der Netzkommunikation inakzeptabel sein. Grundsätzlich ist der Unterschied zwischen einem Funktionsaufruf im selben Adreßraum und einer Netzkommunikation um Größenordnungen unterschiedlich.

Der Lösungsansatz sind, wie oben angedeutet, Timer, die maximale Wartezeiten in Form von Schwellenwerten im Programm angeben. Diese Werte können entsprechend dem Einsatzkontext vom Entwickler eingestellt werden und sind in dem Sinne bekannt, als daß auch eine Visualisierung und Fehlerbehandlung vorgesehen ist.

Jeder Versand eines Kommunikationspaketes im Proxy führt zu einer Eintragung in einer Tabelle, bis zu welchem Zeitpunkt eine Antwort erwartet wird. In zyklischen Abständen wird diese Tabelle nach Einträgen durchsucht, die bereits abgelaufen sind. In diesen Fällen wird das Paket vom Kommunikationssystem zurückgefordert und das Überschreiten des Schwellenwertes dieses Paketes dem Absender mitgeteilt.

Geeignete Zeitüberbrückungsmechanismen geben dem Anwender ein Feedback über verteilte Kommunikation. Beispiele hierfür sind Statuszeileneinträge, die vermelden, daß von einem entfernten System Dienstleistungen angefordert werden; ein anderes Beispiel ist ein Symbol in der Statuszeile, welches eine bestehende Kommunikationsverbindung signalisiert und pro Kommunikationseinheit Aktivität visualisiert. Dagegen wird die Verwendung eines „schlafenden Mauszeigers“, der aus einer synchronen Interpretation der Verteilung stammt, als ungeeignet angesehen.

Aus Gründen der technischen Machbarkeit muß das Abfragen der Werte im Werkzeug lokal möglich sein. Für die Lieferung des Automatenzustandes verwendet das Proxy eine Reihe von Sonden, die an den Automaten angeschlossen werden.

7.6 Zusammenfassung

In diesem Kapitel wurden folgende Konzepte und Verfahren eingeführt:

Die Unterscheidung zwischen Anwendungssystem und Arbeitsplatzsystem wurde notwendig, um einer Aufteilung eines Anwendungssystems auf verschiedene gleichzeitig involvierte Arbeitsplätze gerecht zu werden. Auf dieser Grundlage läßt sich nun die Gesamtheit aller Werkzeuge und Automaten genauso gut ansprechen, wie z.B. die für einen Arbeitsplatz relevanten Werkzeuge.

Ein wichtiges Konzept ist die Leitlinie für die Aufteilung eines Anwendungssystems in Komponenten. Werkzeuge und Automaten werden als eigenständige Prozesse realisiert. Die Muster Proxy, Sonde und deren strukturelle Anbindung sind dafür das Verbindungsmaterial.

Die Untersuchung der Kopplung führte zu einem neuen Programmiermodell: Von der funktionsorientierten Programmierung zu einer ereignisgesteuerten Programmierung. Dieses allgemeine Programmiermodell eignet sich gleichermaßen für die Gestaltung von verteilten wie von herkömmlichen Anwendungssystemen.

Ein weiteres Ergebnis dieses Kapitels ist die Erkenntnis über die Gestaltung der Benutzungsschnittstelle gegenüber dem Anwender. Die Benutzungsschnittstelle muß die Kommunikation über geeignete Mittel verdeutlichen. Das kann mit Hilfe von Statusanzeigen geschehen, die den aktuellen Status der Verbindung zu entfernten Automaten visualisieren. Das kann ebenfalls mit Anzeigen durchgeführt werden, die über die Kommunikationsaktivität Auskunft geben. In jedem Fall sind aber Dialoge notwendig, die den Anwender über abgebrochene Verbindungen informieren.

Letztendlich, nimmt man die Ergebnisse des vorangegangenen Kapitels hinzu, lassen sich nun Einstellwerkzeuge auf von dem betreffenden Automaten unabhängigen Rechnern betreiben. Durch die Aufteilung in unterschiedliche Prozesse wird es möglich, Wartungsarbeiten von entfernten Arbeitsstationen durchzuführen.

8 Der Umgebungsprozeß

Im vorangegangenen Kapitel wurde gezeigt, daß die Kopplungen von Werkzeugen und Automaten mit erheblichen Vorteilen und geringem zusätzlichem Aufwand[32] in Mehrprozeßräumen konstruiert werden können. Ein allgemeines Muster war dazu als Grundlage entstanden. Die dabei notwendige Berücksichtigung des asynchronen Kommunikationsflusses bereits bei der Kommunikation zwischen unterschiedlichen Prozessen auf einem Rechner ist Bestandteil dieses Schrittes. In diesem Kapitel wird gezeigt werden, daß es notwendig ist, fachlich motivierte Werkzeuge eines Arbeitsplatzsystems, die in getrennten Prozessen laufen, durch eine zentrale Instanz pro Prozeßraum zu koordinieren. Dafür wird ein Umgebungsprozeß eingeführt, der einerseits die Koordination von Werkzeugen übernimmt und andererseits eine Reihe von Diensten für die Werkzeuge zur Verfügung stellt, die als eine Basis für eigenverantwortliche Arbeitsplätze angesehen werden können.

Ich werde deshalb zuerst in einer Diskussion zeigen, warum es sinnvoll ist, sich mit dem Umgebungsbegriff zu beschäftigen. Im zweiten Teil werde ich den Umgebungsbegriff fachlich interpretieren und in die Werkzeug-Automat-Material Methodik einordnen. Im dritten Abschnitt wird gezeigt werden, warum es sinnvoll sein kann und ist, Werkzeuge so zu konstruieren, daß sie in separaten Prozessen ablaufen (können). Als viertes wird der Umgebungsprozeß vorgestellt. Er ist die zentrale Instanz, die zur Verwaltung der Werkzeuge benötigt wird. Seine Eigenschaften und die von ihm angebotenen Dienste sind Bestandteil dieses Abschnittes. Im fünften Teil werde ich detailliert darauf eingehen, wie die Werkzeuge an den Umgebungsprozeß angebunden werden. Die technischen Aspekte der Konstruktion von Werkzeugen werden untersucht. Und im letzten Abschnitt schließt sich eine Diskussion an, die die verschiedenen Aspekte und Gesichtspunkte der Trennung von Werkzeugen und Automaten in separate Prozesse näher betrachten wird.

8.1 Motivation

Um einen eigenverantwortlichen Arbeitsplatz einer gewählten Anwendungsdomäne zu unterstützen, sind verschiedenartige Werkzeuge notwendig. Insbesondere hängt die Zusammenstellung der Werkzeuge von dem gewählten Arbeitsschwerpunkt des

[32] Lediglich ein Proxy muß angefertigt werden

Anwenders ab. Neben einer Reihe von Standardwerkzeugen, werden meist auf die jeweilige Arbeitssituation abgestimmte Zusammenstellungen von Spezialwerkzeugen benötigt.

Das Beispiel des Anwendungssystems zur Konfiguration eines Call Centers hat gezeigt, daß sich die Werkzeuge nach zwei Gesichtspunkten kategorisieren lassen. Dabei unterscheidet man Werkzeuge, die zur Konfiguration entwickelt wurden (Objekt-Auflister-Werkzeug) und solche, die auch in anderen Arbeitskontexten eingesetzt werden können (Palettenwerkzeug, Anmeldewerkzeug).

Neben dieser flexiblen Möglichkeit der Zusammenstellung von Werkzeugen ist die Personalisierung des Arbeitsplatzes wichtig. Alle Werkzeuge müssen einerseits in Abhängigkeit zu der benutzenden Person einen unterschiedlichen Umfang an Funktionalitäten zulassen (ein Administrator hat mehr Befugnisse als sein Stellvertreter) und zusätzlich die Präferenzen des Anwenders speichern und bei späterer Benutzung wieder anbieten.

Diese Grundanforderungen ließen sich mit Hilfe eines Umgebungsprozesses realisieren, der als lokale Instanz die Koordination der verschiedenen Werkzeuge übernimmt. Insbesondere zu diesem Aspekt ist keine Literatur bekannt. Autoren, die sich mit der Verteilung von Systemen beschäftigen, gehen von einer expliziten Regelung der Verteilungsaspekte in den Bestandteilen der Werkzeuge aus. Im Rahmen dieser Arbeit wird herausgestellt, daß durch das allgemeine Konzept eines Umgebungsprozesses insbesondere die Wiederverwendung im Vordergrund steht.

Der Umgebungsprozeß wird für jeden Anwender auf technischer Basis „den Ort für die anwendungsfachliche Zusammenstellung von Werkzeugen und Materialien“ ([Gry96]) darstellen. Aus diesem Grund wird der Umgebungsprozeß auch Kenntnis über den Anwender besitzen, dessen Arbeitsplatz modelliert wird. Durch die in ihm angesiedelte Verwaltung der Benutzeridentität können alle Werkzeuge im Kontext dieses einen Umgebungsprozesses mit Anwenderinformationen und den ihnen zugeordneten Berechtigungen versorgt werden. Auf diesen Aspekt wird später noch eingegangen werden.

8.2 Fachliche Interpretation

Bevor eine technische Umsetzung eines Umgebungsbegriffs durchgeführt werden kann, ist zuerst die fachliche Interpretation dieses Begriffs in der Art zu festigen, daß sie die klaren Anforderungen für die technische Implementation vorgibt.

In [Gry96] wird die Arbeitsumgebung fachlich als

> *„[...] der Ort für eine anwendungsfachlich motivierte Zusammenstellung von Werkzeugen, Automaten und Materialien [...]"*

beschrieben.

Gryczan benutzt den Ort, an dem Werkzeuge, Automaten und Materialien angesiedelt sind, als Begründung dafür, wo und wie der eigenverantwortliche Arbeitsplatz für eine qualifizierte Tätigkeit zu gestalten ist. Die spätere Software muß dem Anwender einen Ort bieten, der Werkzeuge und Materialien beinhaltet. Werkzeuge werden vom

Anwender stets in einem sinnvollen Zusammenhang verwendet. Insofern obliegt es dem Anwender, die eingesetzten Werkzeuge auszuwählen und zu verwenden.

Aus dieser fachlichen Definition läßt sich interpretieren, daß eine Sammlung von Software-Werkzeugen, Automaten und Materialien, die in einer Software-Umgebung für den Anwender verfügbar gemacht werden, auszuarbeiten ist.

Hierbei werden die Werkzeuge dem Anwender direkt zur Hand gegeben, während Automaten und Materialien im konkreten Fall nur durch Werkzeuge bedient und benutzt werden können. Um einen Automaten z.B. einstellen zu können, wird ein passendes Einstellwerkzeug benötigt. Um ein Material auswählen zu können, wird ein geeignetes Auflistwerkzeug benötigt.

Die Umgebung bildet den fachlichen Rahmen für diese Werkzeuge. Stärker in einen Bezug zur Software gesetzt:

> *„An integrated environment is a collection of software tools that work together, freeing the user from coordinating them manually." ([SN92], S229).*

Das heißt, die Software-Werkzeuge bilden durch ihre Zusammenstellung die Umgebung, in der ein Anwender arbeitet und entbinden ihn gleichzeitig von ihrer Koordination.

Es wird als Ergebnisse der fachlichen Interpretation folgendes festgehalten:

- Die Umgebung ist Ort für die Zusammenstellung
- Sie bietet Werkzeuge zum Sondieren und Verändern von Materialien
- Sie enthält Einstellwerkzeuge für Automaten

Darüber hinaus kann die Anwendungssituation weitere Werkzeuge erfordern. Die praktische Erfahrung hat aber gezeigt, daß die o.g. Werkzeugtypen charakteristisch für einen eigenverantwortlichen Arbeitsplatz sind. Das heißt, man kommt bei der Zusammenstellung von Arbeitsplätzen immer wieder auf Werkzeuge, die entweder Automaten einstellen oder Materialien nur sondieren bzw. auch verändern können.

8.3 Technische Interpretation

Als Grundlage für die Umsetzung eines Anwendungssystems in einzelne Prozesse muß zuerst die technische Realisierung der Umgebung im Einzelprozeßraum untersucht werden, bevor dieses Konstruktionsprinzip auf den Mehrprozeßraum übertragen werden kann.

Gryczan bildet in seiner Arbeit aus dem fachlichen Begriff der Arbeitsumgebung den technischen Begriff der Umgebung. Auf diesen technischen Begriff Umgebung möchte ich mich im folgenden beziehen.

Die dortige Realisierung der Umgebung erfolgt in Form einer Klasse, die ein im System einmalig vorkommendes Objekt definiert (Singleton, vgl. z.B. [GHJ+95], Seite 127). Dieses Objekt erzeugt alle im Anwendungssystem laufenden Werkzeuge und verwaltet diese (siehe Abbildung 32). Gryczan schlägt deshalb in seiner Arbeit die Umgebung als Beobachter aller Kontext-FKs vor. Dieses Muster reicht aus, das zu realisieren, was er beabsichtigte: Die Weitergabe von Ereignissen an benachbarte (d.h. im selben Prozeß realisierte) Werkzeuge. Im Rahmen dieses Kapitels wird dazu eine Alternative erarbeitet.

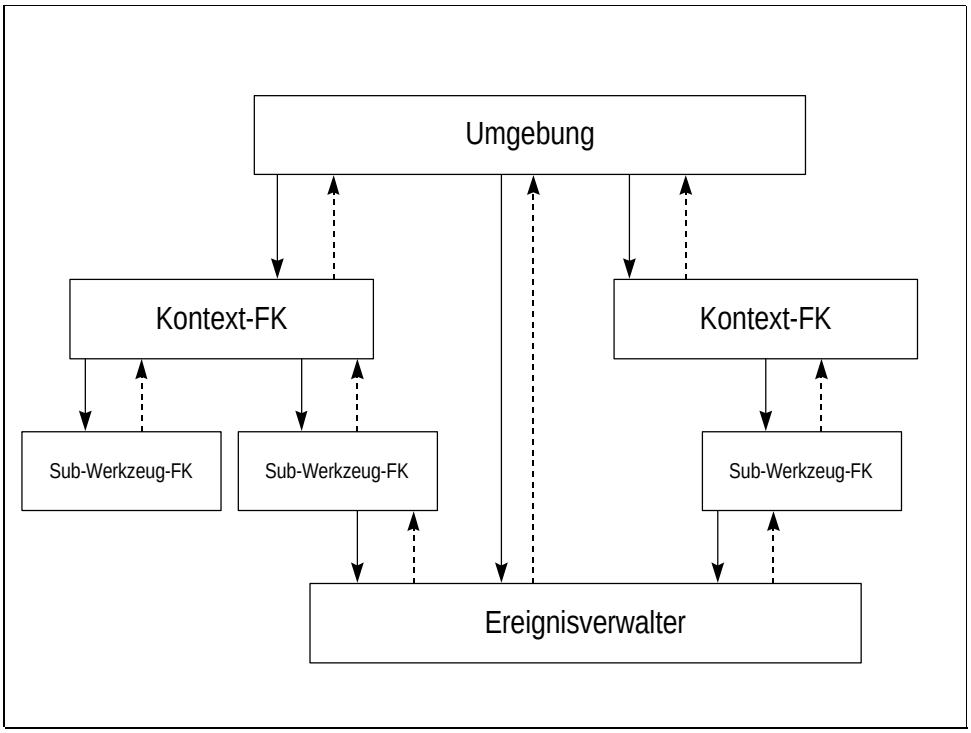

Abbildung 32 Gryczans Interpretation des Zusammenhangs der Komponenten

Zuständsänderungen, die z.B. in der Sub-FK erzeugt werden, werden über die Kontext-FK ggf. an die Umgebung weitergereicht. Diese kann das Ereignis über den Ereignisverwalter an andere Werkzeuge weitergeben, wodurch eine synchrone Benachrichtigung aller Werkzeuge über Zustandsänderungen durchgeführt wird.

Feststellung 3 Materialkonsistenz

Durch die Kopplung von Werkzeugen über einen synchronen Ereignismechanismus wird eine konsistente Sicht auf Materialien gegeben. Alle Werkzeuge werden über die Veränderung eines Materials informiert, bevor sie selbst eine weitere Veränderung durchführen können.

Der zur Kopplung der Funktionskomponenten, Subkomponenten und der Umgebung verwendete Ereignismechanismus ermöglicht zwar die technische Realisierung einer losen Kopplung, ist aber nach eingehender Untersuchung des Einsatzfeldes nur bedingt geeignet. Dies zeigt sich daran, daß der Ereignismechanismus sowohl zur Benachrichtigung von Zustandsänderungen eingesetzt wird (Material wurde modifiziert), als auch für Aufgaben, die an übergeordnete Instanzen delegiert werden sollen. (Sub-FK reicht eine Aufgabe an die Kontext-FK weiter, diese gibt diese ggf. an die Umgebung.)

Sowohl aus fachlicher, als auch aus technischer Sicht würde man nach heutigem Stand der Konstruktionstechnik unterschiedliche Kopplungsmechanismen (siehe Kapitel 6) wählen und so kombinieren, daß sich die intendierten Zusammenhänge der Objekte ebenfalls in den jeweils gewählten Kopplungsmechanismen widerspiegeln.

Zur Vorbereitung der Werkzeugkonstruktion für Systeme, die in unterschiedlichen Prozessen laufen, ist es deshalb notwendig, die Beziehungen zwischen den einzelnen Objekten detaillierter zu untersuchen.

In dem hier betrachteten Fall steht die Kontext-FK zur Umgebung mit einem Observer-Muster in Beziehung. Dieses wird dazu verwendet, Zustandsänderungen an die Umgebung weiterzugeben. Zustandsänderungen sind all jene Veränderungen, die sich an einem Werkzeug ergeben können: Z.B. Änderungen an den Einstellungen, ein anderes Material, was von diesem Werkzeug bearbeitet wird, oder ein fachlicher aber nicht persistenter Wert, der sich durch Eingaben des Anwenders geändert hat.

Eine andere Beziehung, in der Kontext-FK und Umgebung stehen, ist mit demChain-of-Responsibility-Muster beschreibbar (vgl. [GHJ+95], Seite 223). Hierbei geht es um die Weitergabe von Startaufträgen. So kann z.B. ein Doppelklick in einer Liste, die über eine Sub-IAK realisiert wurde, als der Wunsch interpretiert werden, ein weiteres Werkzeug zu starten. Diese Aufgabe muß von der Sub-FK an die Kontext-FK weitergegeben werden. In einigen Fällen ist sie aber selber nicht in der Lage, diesen Startauftrag auszuführen. Mit dem Muster Chain of Responsibility wird der Auftrag an die Umgebung weitergeleitet. Diese sollte im allgemeinen zur Ausführung des Auftrags in der Lage sein.

Es ist deshalb aus konzeptioneller Sicht wünschenswert, das Konstruktionsmuster „Beobachter" nur für Zustandsänderungen zu verwenden und das Konstruktionsmuster „Zuständigkeitskette" immer dann einzusetzen, wenn ein Objekt nicht in der Lage ist, einen Auftrag auszuführen. Dies führt zu einer saubereren Trennung der Tätigkeitsfelder und der Beziehungen, in der Objekte zueinander stehen.

Technisch ebensowenig benannt, ist das Verfahren, mit dem neue Werkzeuge erzeugt werden. Es hat sich hierbei angeboten, in der Umgebung den Konstruktionsprozeß anzusiedeln, damit sie in der Lage ist, neue Werkzeuge herzustellen (zu erzeugen). Dazu werden austauschbare Verfahren verwendet, die jeweils die konkrete Erzeugung eines Werkzeuges übernehmen. Mit Hilfe dieser Abstraktion kann wiederum ein allgemeines Verfahren angeboten werden, in das spezielle Realisierungen eingebettet werden können.

Konkret bedeutet dies für die Einbindung eines neuen Werkzeugs, daß um einen neuen Typus von Werkzeug anzubieten, folgende Konstruktionstechnik verwendet wird: Jedes Werkzeug, welches in einem Anwendungssystem realisiert werden soll, muß zusätzlich mit einem konkreten Builder (vgl. [GJH+95], Seite 97) versorgt werden, in dem die Erzeugung eines neuen Exemplars dieses Werkzeugs implementiert ist. Builder Objekte werden zur Laufzeit bei der Umgebung registriert. Jedes der Builder Objekte kann gefragt werden, welches Werkzeug es erzeugen kann. Als Ergebnis eines Erzeugungsauftrags liefert ein Builder ein Tool-Objekt zurück, welches als Referenz auf FK und IAK angesehen werden kann. Die Entscheidung fiel auf diese Erzeugungsmethode, weil z.B. die Verwendung des Musters „Prototyp“ (vgl. [GJH+95], Seite 117) für einige Werkzeuge eventuell zu sehr hohem Speicherbedarf geführt hätte. Der Einsatz einer „Fabrik“ (vgl. [GJH+95], Seite 87) ist hingegen zu kurz gegriffen, weil es notwendig sein kann, daß in den Konstruktionsprozeß eingegriffen werden muß[33].

Die dritte herauszustellende Beziehung zwischen der Umgebung, den FKs und den Sub-FKs ist das *Composite Pattern*. So besteht das Arbeitsplatzsystem, repräsentiert durch die Umgebung, aus Werkzeugen und jedes Werkzeug kann wiederum aus Sub-Werkzeugen bestehen. Das Ändern der Benutzeridentität zur Laufzeit des

[33] Es kann z.B. notwendig werden, daß eine Funktionskomponente vor dem Erzeugen ihrer Interaktionskomponente bereits Werte zur Verfügung stellt, von denen die Erzeugung der IAK abhängt.

Arbeitsplatzsystems ist eine Anwendungssituation, in der die Objekte in der Beziehung eines *Composite Pattern* stehen. Jedem untergeordneten Objekt muß mitgeteilt werden, welcher Benutzer nun angemeldet ist und welche Berechtigungen er besitzt.

Die nachfolgende Abbildung versucht, die Konstruktion von Werkzeugen zusammenfassend darzustellen.

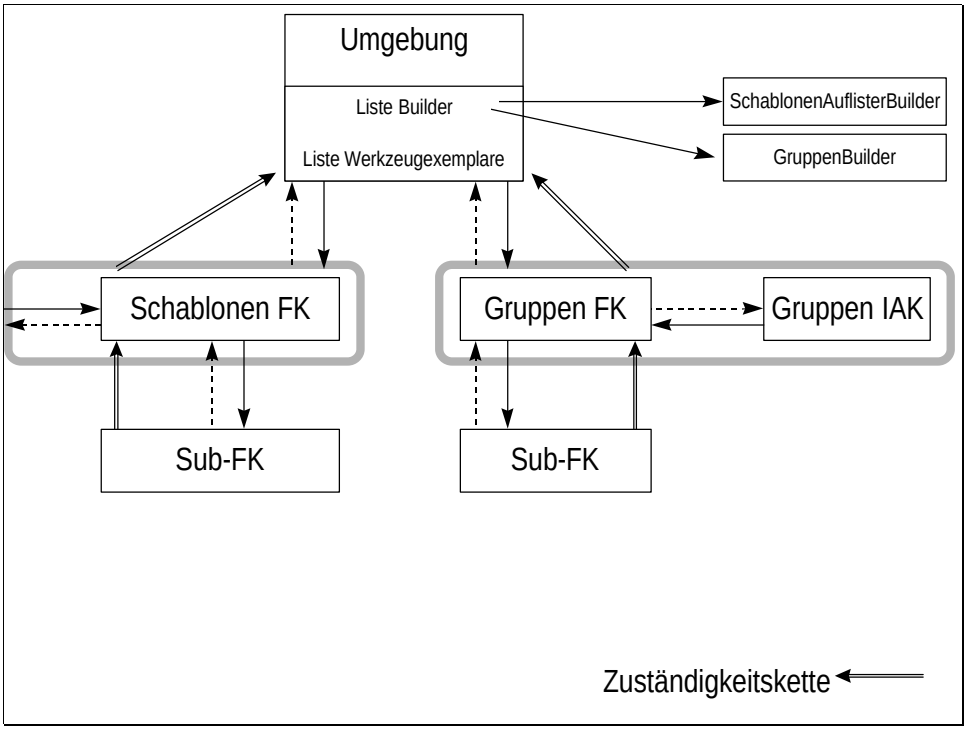

Abbildung 33 Beziehung der Objekte eines Werkzeugs

Anhand der Abbildung 33 wird offensichtlich, daß sich eine derart komplexe Beziehung zwischen nur wenigen Objekten bereits nicht mehr geeignet mit herkömmlichen Mitteln darstellen lassen kann.

Reenskoog und auch Riehle beschreiben diese unterschiedlichen Beziehungen mit Hilfe von Rollendiagrammen. Diese Darstellungsweise bietet gegenüber der obigen den Vorteil, deutlich übersichtlicher zu sein, weil Objekte nur unter einem Blickwinkel betrachtet werden.

8.4 Rollendiagramme

Die von Reenskoog ([Ree96], Seite 33ff) vorgeschlagene Betrachtungsweise auf die Beziehungen von Objekten kann etwa folgendermaßen beschrieben werden: Objekte in einem komplexen Anwendungssystem stehen in unterschiedlichen Beziehungen zueinander. Bei der Modellierung der Objekte wird jeweils nur eine semantisch abgrenzbare Beziehung zwischen einer Gruppe von Objekten herausgegriffen. Diese heißt dann Rolle.

Als Beispiel soll ein Mitarbeiter aus einem Arbeitsteam dienen. Er erfüllt unterschiedliche Rollen bei seiner Arbeit. Neben seiner Rolle als Softwareentwickler hat er z.B. in

Projektsitzungen die Aufgabe eines Diskussionsleiters und bei Projektvorstellungen die Aufgabe eines Vorführers. Obwohl also die Rolle in Abhängigkeit zur Situation wechselt, so werden dennoch verschiedene Aufgaben von ein und derselben Person übernommen.

Diese Betrachtungsweise kann auf Objekte in Softwaresystemen übertragen werden. Wie die Person aus dem oben aufgeführten Beispiel nimmt auch ein Objekt in unterschiedlichen Kontexten unterschiedliche Aufgaben wahr. Bei der Modellierung eines Objektes vereinigt man diese unterschiedliche Aufgaben aus unterschiedlichen Einsatzkontexten in Form von Rollen.

Durch die Modellierung mit Hilfe von Rollen hat man ein Ausdrucksmittel bekommen, welches es erlaubt auf einer höheren Ebene als der Klassenschnittstelle die Zusammenarbeit von Objekten zu beschreiben. Denn innerhalb eines betrachteten Zusammenhangs nimmt jedes Objekt wieder nur genau eine Rolle ein, die man aus den Gestaltungsmustern kennt.

Die nachfolgende Abbildung soll die wesentlichen Elemente dieses Darstellungsmediums anhand der beispielhaft ausgewählten Rollen von Mustern vorstellen:

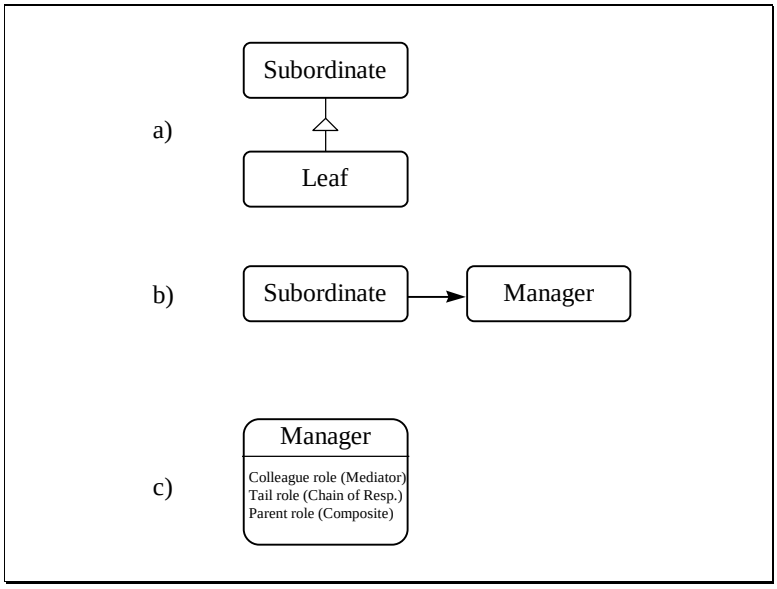

Abbildung 34 Darstellungselemente der Rollenmodellierung

Rollen werden durch Kästen mit abgerundeten Ecken dargestellt. Im Kasten wird der Name des Objektes, das diese Rolle einnimmt, angegeben. (Werden Muster der Zusammenarbeit abstrakt beschrieben, können die konkreten Rollenteilnehmer auch durch abstrakte Bezeichnungen aus den Musterbeschreibungen ersetzt werden.) Die Abbildung 34 zeigt unter a), daß es durchaus möglich ist, die Vererbungsbeziehung in Rollendiagrammen zu verwenden. Hier wird damit angedeutet, daß Leaf eine Spezialisierung von Subordinate ist.

Unter b) sieht man, daß in dieser Situation das Subordinate einen Manager benutzt. Und zwar genau ein Objekt von jedem Typ ist an dieser Rollensituation beteiligt. Würden mehrere Objekte desselben Typs beteiligt sein, würden diese Kardinalitäten am Pfeil kenntlich gemacht werden.

Mit dem Darstellungsmittel c) kann die Gesamtheit der Rollen, die ein Objekt einnimmt angegeben werden. Es wird jeweils die Bezeichnung der Rolle und das Muster angegeben, aus dem diese Rolle stammt.

Im weiteren Verlauf der Konstruktion wird untersucht, welches konkrete Objekt des späteren Laufzeitsystems welche herausgearbeiteten Rollen in sich vereinigen wird.

Auch Riehle ([Rie96], Seite 11ff) beschreibt diese unterschiedlichen Beziehungen mit Hilfe von Rollendiagrammen. Er unternimmt den Versuch, ein Verfahren anzubieten, mit dem die optimale Kombination von Rollen in einzelnen Objekten für den gewünschten Anwendungsfall bestimmt werden kann.

Die nachfolgende Abbildung 35 zeigt diese Rollendiagramme. In jedem Kasten ist eine Rolle dargestellt, die entsprechend betitelt wird. In diese Kästen geschrieben werden die Objektnamen, von den Objekten, die diese Rollen einnehmen. Zwischen den Rollen kann es wie in Klassendiagrammen auch, Vererbungs- oder Benutztbeziehungen geben. Somit kann in einem Rollendiagramm ausgedrückt werden, welche Objekte in der betrachteten Situation welche Rolle übernehmen.

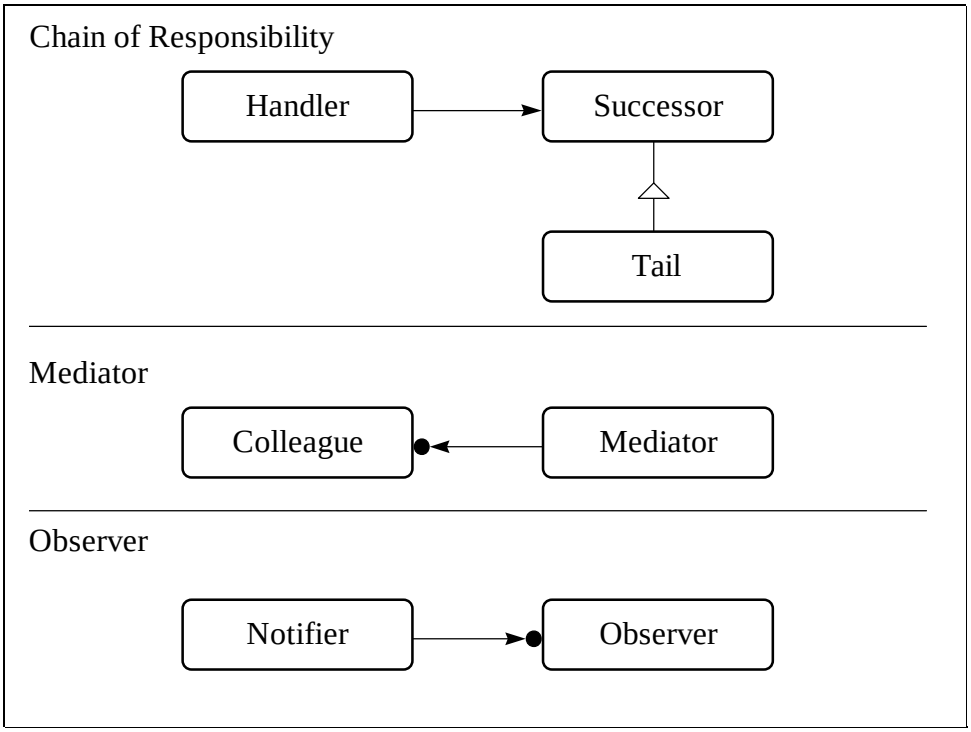

Abbildung 35 Allgemeine Gestaltungsmuster als Rollendiagramme

Die Abbildung 35 zeigt, wie allgemeine Gestaltungsmuster mit Hilfe von Rollendiagrammen dargestellt werden können. Die Bezeichnungen für die Rollen werden im Einsatz durch die konkreten Beteiligten ersetzt.

Wendet man die Darstellungsmethode auf das hier behandelte Einsatzfeld an, entsteht die nachfolgende Abbildung, in der die Rollenbezeichnungen durch die konkreten Klassennamen ersetzt wurden.

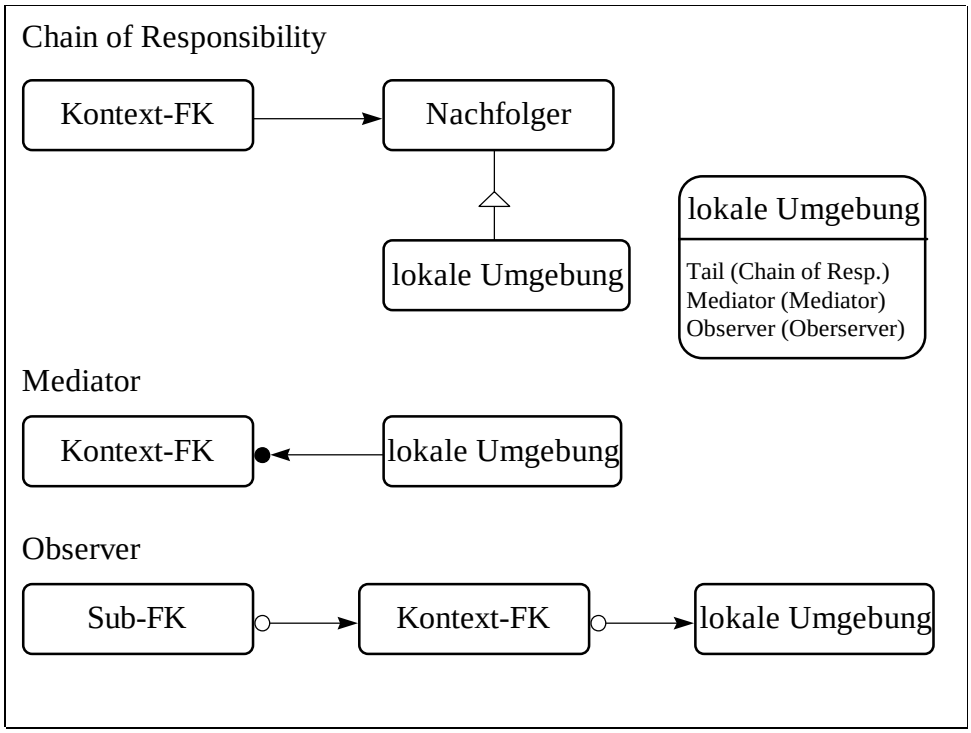

Abbildung 36 Rollen der lokalen Umgebung dargestellt mit Rollendiagrammen

Die lokale Umgebung nimmt in einem Prozeß eines Arbeitsplatzsystems die drei bereits oben ausgeführten zentralen Rollen ein. Die Abbildung 36 zeigt, in welchem Verhältnis die lokale Umgebung in jeder Rolle zu anderen Objekten des Prozesses steht.

8.5 Konfektionierbarkeit

Die in dieser Arbeit dargestellte modulare Konstruktionstechnik für Anwendungssysteme ermöglicht eine Kombination von Werkzeugen zu unterschiedlichen Anwendungssystemen. Diese freie Konfektionierbarkeit wird zum Beispiel dann notwendig, wenn man während und nach der Softwareentwicklung entstehenden Bedürfnissen gerecht werden möchte. Das könnten z.B. neue Werkzeuge sein, die in das System eingebunden werden müssen. Durch die Einführung eines Anwendungssystems können neue Aufgaben geschaffen werden, für die angemessene Werkzeuge in das Anwendungssystem ergänzt werden müssen.

Daraus ergibt sich die Notwendigkeit, ein allgemeines Muster zu entwickeln, mit dem unterschiedliche Werkzeuge in einem Arbeitsplatzsystem verfügbar gemacht werden können (vgl. [Rie95]). Durch Bereitstellen eines Buildersfür jedes Werkzeug kann diese Problematik technisch angemessen gelöst werden. Dadurch bekommt die Umgebung geeignete Informationen zur Verfügung gestellt, um zur Laufzeit neue Werkzeugexemplare erzeugen zu können. Die erzeugten Objekte sind der Umgebung nur als Tools unter einer abstrakten Schnittstelle bekannt. Für jedes Werkzeug muß demnach nicht nur eine Funktionskomponente und eine Interaktionskomponente (mit ihren jeweiligen Subkomponenten) zur Verfügung gestellt werden. Es ist auch nötig, einen konkreten Builder bereitzustellen, der die Erzeugung und Verknüpfung übernimmt.

Jedes Werkzeug (Tool) kann sowohl als Schablone, als auch als Exemplar nach seinem Typ, einer Identifikation, gefragt werden. Exemplare von Werkzeugen können zusätzlich nach dem aktuell von ihnen bearbeiteten Material gefragt werden.

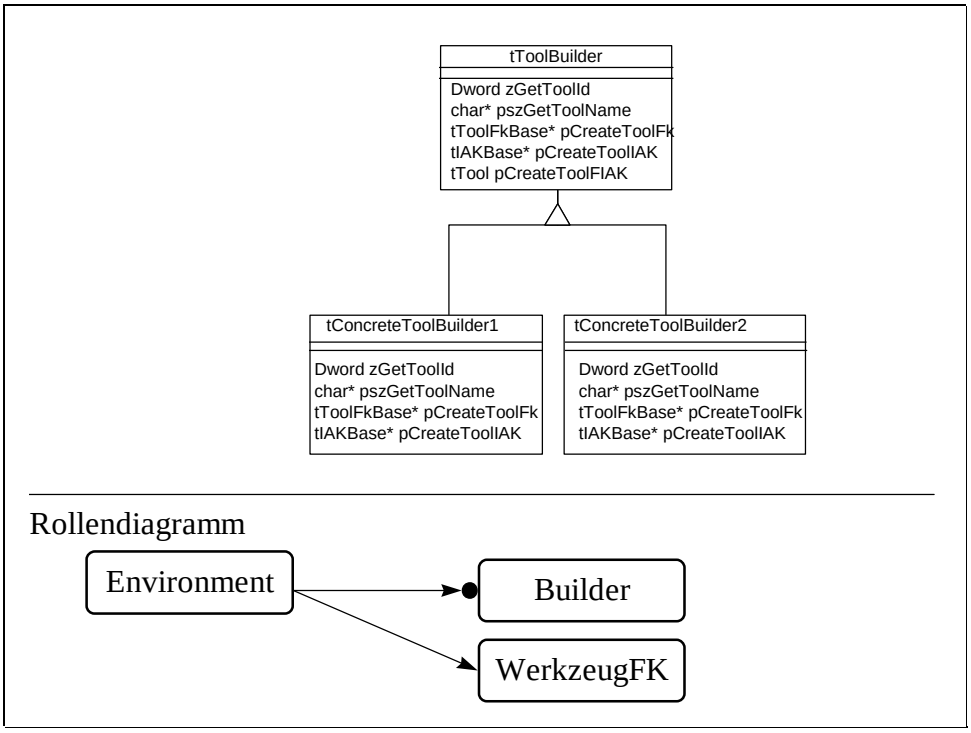

Abbildung 37 Konkretisierung eigener Werkzeuge

Beim Starten des Anwendungssystems werden alle Schablonen für Werkzeuge, die durch Builder implementiert sind, in der Umgebung registriert. Sie stehen danach zur Erzeugung einzelner oder mehrerer Exemplare zur Verfügung. Das hängt von der Implementation (Einschränkung durch Singleton) ab.

Eines dieser Werkzeuge, das Rahmenwerkzeug oder ein Palettenwerkzeug, wird initial gestartet und bildet für den Anwender den Ausgangspunkt für seine Arbeit. Es mögen aber auch Anwendungsfälle denkbar sein, bei denen eine ganze Reihe von Werkzeugen initial gestartet werden. Ebenso denkbar ist das Starten der Werkzeuge, die bei der letzten Arbeitssitzung geöffnet waren. Solche Unterscheidungen ließen sich ggf. mit dem Strategy-Muster (vgl. [GJH+95], Seite 315) realisieren.[34]

Ein Palettenwerkzeug oder ein anderes Werkzeug kann durch eine Benutzerauswahl oder den Aufruf einer Funktion das Starten eines anderen Werkzeugs veranlassen. Dieser Startauftrag wird über eine Zuständigkeitskette an die Umgebung weitergereicht.

Alle semantisch abgeschlossenen Werkzeuge werden über eine eindeutige Bezeichnung (z.B. Nummerncode) identifiziert. Diese Identifikation wird an die Factory weitergeleitet.

[34] Es könnten dann drei verschiedene Strategien zur Verfügung gestellt werden. Bei der ersten werden keine Werkzeuge gestartet. Die zweite startet immer einen „aufgeräumten“ Arbeitsplatz und die dritte Strategie würde den Zustand der letzten Arbeitssitzung wieder herstellen.

8.6 Werkzeuge in separaten Prozessen

Bisher wurde gezeigt, das Anwendungssysteme leicht in der Weise modular gestaltet werden können, daß zur Übersetzungszeit vom Entwickler entschieden werden kann, welche Werkzeuge Bestandteile des Systems sind und welches Rahmenwerkzeug dem Anwender zur Verfügung gestellt wird.

Obwohl die zuvor eingeführte Konstruktionstechnik das Zusammenstecken von Werkzeugen zu Anwendungssystemen auf modulare Weise ermöglicht, bleibt eine Reihe von entscheidenden Nachteilen bestehen:

- Die Werkzeuge können nur zur Compile-Zeit zusammengesteckt werden.
- Komponenten können nicht zur Laufzeit ausgetauscht werden.[35]
- Die Verbindung unterschiedlicher Anwendungssysteme kann nur durch Neugestaltung eines neuen Anwendungssystems geschehen.
- Korrektur oder Veränderung eines Werkzeugs hat die vollständige Neuzusammenstellung des Anwendungssystems zur Folge.
- **Zur Laufzeit wird immer das gesamte Anwendungssystem geladen. Insbesondere auch Werkzeuge, die während einer Arbeitssitzung nicht benötigt werden.**

Warum sind die hier genannten vier Kritikpunkte für die Gestaltung von Anwendungssystemen für eigenverantwortliche Arbeitsplätze qualifizierter Arbeit relevant?

Das Übersetzen und Binden eines Anwendungssystems ist ein zeitaufwendiger Prozeß.[36] Er wird im allgemeinen um so aufwendiger, je mehr Komponenten (Werkzeuge) und je mehr Entwickler an der Entstehung des Anwendungssystems beteiligt sind. Bei einer individuellen Zusammenstellung für unterschiedliche Kunden kommt eine weitere Dimension der Konfektionierung hinzu.

Der erste Kritikpunkt bezieht sich auf das Zusammenstellen des Anwendungssystems. Sprechen fachliche Gründe für eine andere Zusammenstellung eines Anwendungssystems, kann das nicht in kurzen Testphasen experimentell ermittelt werden, sondern muß für jede Zusammenstellung neu übersetzt und neu gebunden werden.

Werden für unterschiedliche funktionelle Rollen (vgl. [Flo94b]) unterschiedliche Anwendungssysteme separat entwickelt, besteht derzeit nur eine eingeschränkte Möglichkeit zum Datenaustausch zwischen den Anwendungssystemen bzw. Werkzeugen. Insbesondere ist für den Anwender eine Trennung der Anwendungssysteme durch zwei getrennte Rahmenwerkzeuge sichtbar, wenn man davon ausgeht, daß jedes Anwendungssystem seinen eigenen Desktop realisiert. Besteht die Möglichkeit, auf den Quellcode zurückzugreifen, können beide Anwendungssysteme ggf. zu einem neuen zusammengefügt werden. Dies ist aber nur unter Koordination der Entwicklungsteams

[35] Die DLLs ermöglicht zwar auf einigen Betriebssystemen das Austauschen von Komponenten auch zur Laufzeit, diese Technologie steht aber nicht unter allen Betriebssystemen zur Verfügung.

[36] Im konkreten Industrieprojekt wurden die aktuelle Versionen der C++ Compiler eingesetzt. Diese benötigten deutlich mehr als 20 Minuten, um den gesamten Quellcode eines Programms zu übersetzen.

beider Anwendungssysteme möglich und erfordert bei jeder Änderung eines Anwendungssystems die erneute Koordination und Zusammenstellung des gemeinsamen Anwendungssystems.

Dies spricht eben auch gleich den dritten Kritikpunkt an, der sich auf die Korrektur von Werkzeugen bezieht. Ein Anwendungssystem besteht in der Regel aus einer großen Anzahl kundenspezifischer Werkzeuge und einer kleinen Menge von Werkzeugen, die die entwickelnde Firma in verschiedenen Anwendungssystemen einsetzt. Die Korrektur von Fehlern bzw. Anpassung der Werkzeuge an neue Gegebenheiten kann bei einem monolithischen Anwendungssystem immer nur durch Austausch des Gesamtsystems erfolgen. Einzelne Werkzeuge können nicht durch korrigierte oder angepaßte Versionen ersetzt werden.

Der vierte Kritikpunkt bezieht sich auf die tägliche Arbeit des Anwenders. Hierbei handelt es sich weniger um ein konzeptionelles Problem, denn um praktische Erwägungen der täglichen Arbeit. Umfangreiche Anwendungssysteme führen zwangsläufig auch zu großen Programmen, die als Ganzes im Computer geladen werden müssen. Das bedeutet eine unnötig hohe Auslastung des Systems, die dem Anwender durch träges Reaktionsverhalten oder knappe System-Ressourcen die Erledigung seiner Arbeit schwer machen kann und deshalb deutlich ins Gewicht fällt.

Faßt man die hier genannten Kritikpunkte zusammen, läßt sich feststellen, daß mit zunehmender Größe des Anwendungssystems die angesprochenen Aspekte Konfektionierung, Wartung und Integration ebenfalls schwerer werden. Durch diesen Umstand motiviert, läßt sich folgende These formulieren:

These

Teilt man Anwendungssysteme in separate Prozesse auf, wobei die kleinste Einheit ein Werkzeug bzw. Automat bildet. Dann erleichtert diese Aufteilung die Konfektionierbarkeit, die Wartung und die Integration von inhaltlich unterschiedlichen Anwendungssystemen.

In dem referenzierten Industrieprojekt wurde untersucht, unter welchen Umständen die Trennung von Anwendungssystemen in Teilkomponenten sinnvoll ist, inwieweit das Schlagwort Softwarebus (vgl. [Maf96]) auf diesen Sachverhalt anwendbar ist und mit welcher Bedeutung, und welche strukturellen Vorteile aus der Trennung in separate Executables gezogen werden können. Es wurde sich ebenso mit der Kritik auseinandergesetzt, die einerseits den zusätzlichen Kommunikationsaufwand zwischen den Teilen des Anwendungssystemen nennt und andererseits auf die sich ändernden Kommunikationsformate hinweist.

In dem entwickelten Modell konnte tatsächlich die o.g. Problematik dadurch beseitigt werden, daß Werkzeuge (potentiell) für getrennte Prozesse entwickelt wurden. Die Architektur eines Anwendungssystems zerfällt dabei in separate Prozesse, die jeweils eine Menge von Werkzeugen realisieren.

Es muß aber bereits an dieser Stelle unterstrichen werden, daß das Ziel dieser Entwicklung nicht die grundsätzliche Trennung von Werkzeugen und Automaten in separate Prozesse ist. Vielmehr wird das Ziel dahingehend definiert, daß zu jedem Zeitpunkt der Systementwicklung die Entscheidung für die Trennung bzw. das Herauslösen einzelner Komponenten gefällt werden kann.

Bei der Gestaltung von Anwendungssystemen können nun neue Gesichtspunkte in den Vordergrund rücken.

- Werkzeuge können nach anwendungsfachlichen Gesichtspunkten in separate Prozesse gegliedert werden.
- Allgemeine Werkzeuge, die zur Administration des Systems benötigt werden, können als solche zur Verfügung gestellt werden.
- Handhabungen von speziellen Materialien ergeben sich als neue Gliederungskriterien für Werkzeuge.

Es ergibt sich durch den Besitz einer anwendungsfachlichen Leitlinie für die Zusammenstellung von Werkzeugen in separaten Prozessen die Möglichkeit, nicht nur Code-Wiederverwendung (vgl. [Mey88], Seite 5), sondern auch Code Sharing (vgl. [GS94], Seite 252,281ff) in den geeigneten Fällen zu betreiben.

Der zweite Aspekt ermöglicht es, die geschaffene Infrastruktur dieser Philosophie eines Anwendungssystems sofort und ohne jeden Aufwand in weitere Projekte und Systeme zu übernehmen. Werkzeuge, die zur Verwaltung dienen und nicht an ein spezielles Anwendungsfeld gebunden sind, können als ein spezieller Satz von Werkzeugen jedem Anwendungssystem beigegeben werden.

Interessant zu bemerken ist der dritte Gestaltungsgesichtspunkt. Es hat sich gezeigt, daß insbesondere diejenigen Werkzeuge, die auf einem speziellen Material arbeiten, wie z.B. Werkzeuge zum Anlegen, Editieren und Auflisten von einem Materialtyp, in einem Prozeß zusammengefaßt werden. Speziell zu diesem Punkt folgt ein Schwerpunktabschnitt (Kapitel 8.8).

8.7 Technische Gesichtspunkte

Dieser Abschnitt beschäftigt sich mit der technischen Werkzeugkonstruktion. Besitzt man die technischen Voraussetzungen, Werkzeuge in getrennten Prozessen zu realisieren, stellt sich die Frage, wie die Werkzeuge eines Anwendungssystems in die unterschiedlichen Prozesse sinnvoll aufzuteilen sind. Dabei ist einerseits die Koordination der Werkzeugprozesse zu klären. Andererseits muß entschieden werden, welche Werkzeuge zusammen in einen Prozeß gelegt werden.

Nimmt man zunächst an, jeweils eine Teilmenge von Werkzeugen würde in einem Prozeß realisiert werden und mit Hilfe der oben von Gryczan angebotenen Umgebung koordiniert werden. Es ergibt sich dann etwa folgendes Bild:

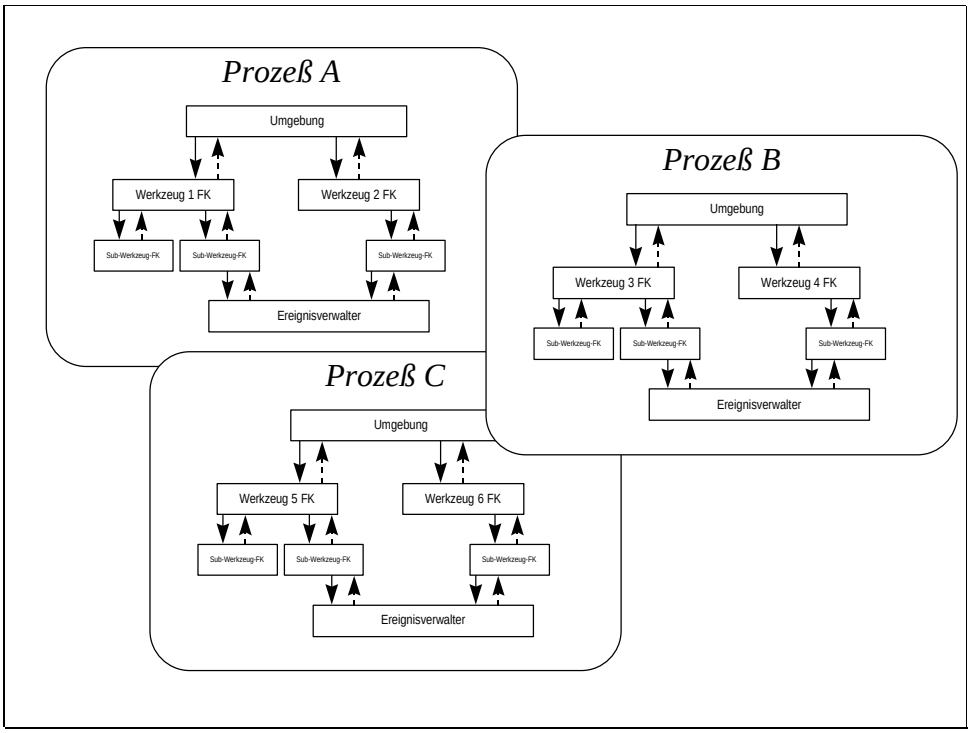

Abbildung 38 Schlußfolgerung für mehrere Prozesse mit jeweils einer Umgebung und mehreren Werkzeugen

Man erkennt, daß in jedem Werkzeugprozeß genau eine (lokale) Umgebung vorhanden ist. Innerhalb des Prozesses sind verschiedene (sinnvoll zusammengehörige) Werkzeuge realisiert.

Wäre die technische Umgebung Bestandteil eines jeden Prozesses, dann müßte entschieden werden, welcher technischen die fachliche Umgebung (und damit für den Arbeitsplatz relevante) zugeordnet werden soll. Da die Prozesse, in denen Werkzeuge realisiert sind, freien kombiniert werden können ist aber nicht entscheidbar, welchem Prozeß die Umgebung mitgegeben werden kann. Man wird deshalb, unter der Prämisse, daß die Umgebung mindestens bei einem Werkzeug angesiedelt sein muß, daraufhin allen Prozessen eine Umgebung geben. Es verbleibt ein neues Problem, nämlich das der Entscheidung, welche der vielen Umgebungen, die es gibt, wenn mehrere Werkzeuge gestartet sind, diejenige ist, die die Verwaltung durchführt.

Insbesondere ergeben sich Folgeprobleme, wie z.B. die Fragestellung, was beim Schließen desjenigen Werkzeugs passiert, welches die koordinierende Umgebung beinhaltet hatte.

Es kann deshalb die oben gemachte Annahme wie folgt geändert werden: Jeweils eine Teilmenge von Werkzeugen wird in einem Prozeß realisiert. In jedem Prozeß gibt es eine koordinierende Instanz für die in ihm implementierten Werkzeuge. Diese Instanz wird lokale Umgebung genannt. Können Aufgaben nicht innerhalb dieser lokalen Umgebung gelöst werden, delegiert diese mit Hilfe eines Proxies die Aufgabe an den Umgebungsprozeß, der alle Werkzeugprozesse verwaltet.

Demnach bleibt es bei einer Umgebung pro Arbeitsplatz; diese wird als Prozeß implementiert, jeder andere Werkzeugprozeß auf demselben Arbeitsplatz hat ein Proxy zu genau dieser Umgebung.

Für die folgende Diskussion wird ein Begriffsgerüst zurecht gelegt. Damit kann zwischen der fachlichen und den verschiedenen technischen Ebenen im Zusammenhang mit der Umgebung unterschieden werden.

Definition 28. Umgebung

Unter der Umgebung *versteht man die fachliche Repräsentation eines Ortes für einen Arbeitsplatz. Die Umgebung repräsentiert darüber hinaus eine Benutzeridentität und die mit ihr verknüpften Einstellungen. Eine Umgebung ist für die Koordination von Werkzeugen zuständig.*

Im Gegensatz zum Begriff der Arbeitsumgebung wird hier die Umgebung stärker eingegrenzt. Direkt davon abgeleitet ist der erste technische Begriff:

Definition 29. Umgebungsprozeß

Der Umgebungsprozeß *ist die technische Implementierung einer Umgebung auf einem Rechner zur Realisierung eines Arbeitsplatzsystems.*

Auch hier trägt der Umgebungsprozeß (wenngleich er aus einer technischen Begriffswelt stammt) zur Eingrenzung bei. Im Sinne des Begriffspaares Anwendungssystem und Arbeitsplatzsystem bezieht sich der Umgebungsprozeß nur auf letzteres.

Für die technische Umsetzung der Umgebung in den Werkzeugprozessen, werden die folgenden zwei Begriffe definiert:

Definition 30. lokale Umgebung

Die lokale Umgebung *bezeichnet das softwaretechnische Objekt in einem Prozeß, das den Umgebungsprozeß auf Prozeßebene vertritt. (Da Klassennamen in Englisch gewählt wurden, wird stellvertretend “Local Environment“ verwendet.)*

Definition 31. Umgebungsproxy

Das Umgebungsproxy *ist die technische Realisierung eines Proxies für den Umgebungsprozeß nach dem oben charakterisierten Muster. Mit Hilfe des Umgebungsproxy nimmt die lokale Umgebung Kontakt zum Umgebungsprozeß auf. (Da Klassennamen in Englisch gewählt wurden, wird stellvertretend “ EnvironmentProxy“ verwendet.)*

Damit ist die Begriffsunterscheidung verfügbar, die für die Darlegung der Entwurfsentscheidungen notwendig ist. Die folgenden Abschnitte werden darauf Bezug nehmen. Alle vier Begriff finden ihre Anwendung im Arbeitsplatzsystem. Während der Begriff „Umgebung“ auf der fachlichen Ebene liegt, sind „Umgebungsprozeß“, „lokale Umgebung“ und „Umgebungsproxy“ rein technisch motivierte Begriffe. Im Mehrprozeßraum ist der Umgebungsprozeß Repräsentant und technische Umsetzung einer Umgebung. Mit Hilfe der lokalen Umgebung können innerhalb von Prozessen realisierte Werkzeuge koordiniert werden. Durch das Umgebungsproxy wird von diesem

Prozeß Kontakt zum Umgebungsprozeß aufgenommen, wenn eine Aktion außerhalb des Prozeßkontextes liegt.

8.7.1 Technische Realisierung des Aufrufs zum Werkzeugstarten

Das Starten eines Werkzeugs untergliedert sich in zwei Schritte. Die Startaufforderung und die Umsetzung des Startauftrags. Die Aufforderung zum Starten kann von unterschiedlichen Urhebern kommen. Ein im selben Prozeß realisiertes Werkzeug kann ein anderes Starten. Die lokale Umgebung kann z.B. beim Hochfahren ein Werkzeug starten. Oder ein in einem anderen Prozeß laufendes Werkzeug kann im betrachteten Prozeß ein Werkzeug starten.

An dieser Stelle wird lediglich das Starten eines Werkzeugs ausgehend von einer Funktionskomponente betrachtet. Dafür bietet die Klasse `tStartToolHandler` eine entsprechende Schnittstelle an. Das Erzeugen eines konkreten Werkzeugexemplars ist Thema des nächsten Abschnitts.

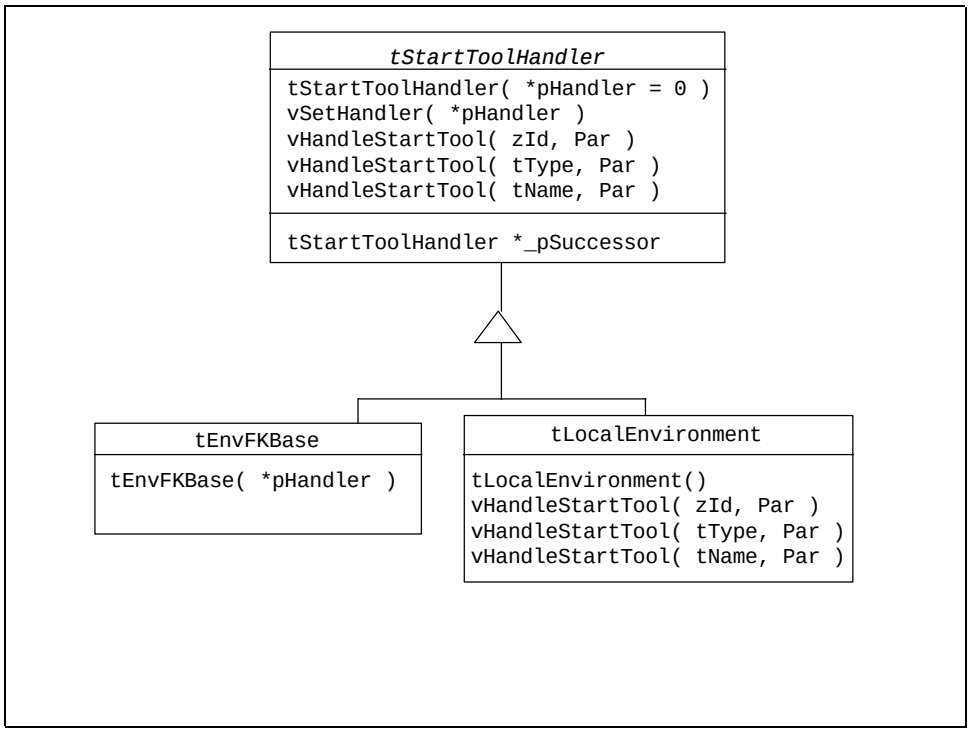

Abbildung 39 Der Einsatz der Klasse `tStartToolHandler` als abstrakte Oberklasse

Die Klasse `tStartToolHandler` ist Oberklasse für alle Werkzeug-Kontext-FKs und für die lokale Umgebung (siehe Abbildung 39). Durch Aufrufen einer der drei angebotenen Methoden ist eine Funktionskomponente in der Lage, ein anderes Werkzeug zu starten. Die Beziehung der Funktionskomponenten, Sub-Funktionskomponenten und der lokalen Umgebung ist durch das Gestaltungsmuster „Zuständigkeitskette“ ausgedrückt.

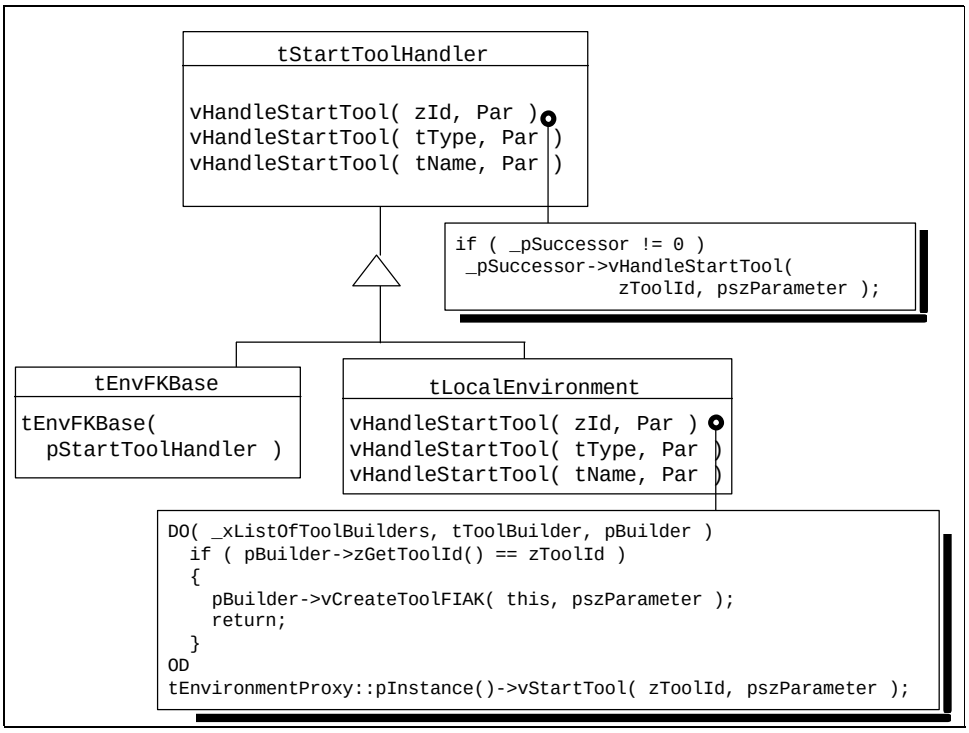

Abbildung 40 Die Verantwortlichkeitsübergabe von Funktionskomponenten zur lokalen Umgebung

Jede Sub-Funktionskomponente, die das Starten eines anderen Werkzeugs aufruft, übergibt den Auftrag der übergeordneten Instanz, wenn sie selbst nicht weiß, wie dieser Auftrag auszuführen ist. In der Klasse tLocalEnvironment ist eine andere Implementation (siehe Abbildung 40) vorgesehen. Die lokale Umgebung sucht in der Liste der verfügbaren Werkzeuge und erzeugt ein neues Exemplar, wenn dies möglich ist. Nur im anderen Fall wird der Auftrag an das Umgebungsproxy und somit an den Umgebungsprozeß weitergegeben.

8.7.2 Technische Realisierung der Werkzeugerzeugung

Kommt der Auftrag zum Erzeugen eines Werkzeugs in der lokalen Umgebung an, dies kann durch ein im selben Prozeß realisiertes Werkzeug oder durch Vermittlung des Umgebungsproxy auch durch ein Werkzeug eines anderen Prozesses geschehen, ist die lokale Umgebung dafür zuständig, die Erzeugung eines neuen Werkzeugexemplars auszuführen.

Dafür führt die lokale Umgebung eine Liste der bekannten Werkzeuge. Der zum gewünschten Werkzeug passende Builder wird ausgewählt und aufgerufen.

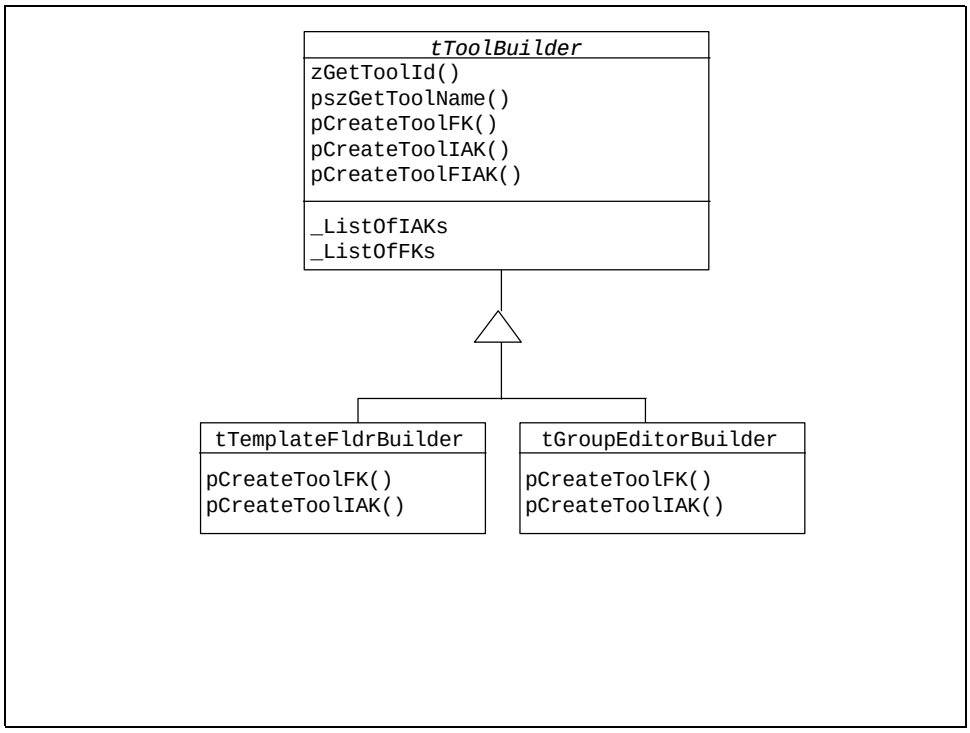

Abbildung 41 Der Einsatz der Klasse `tToolBuilder` als abstrakte Oberklasse

Durch die Klasse `tToolBuilder` wird eine abstrakte Oberklasse definiert, die für die Erzeugung eines Werkzeugs den Rahmen bildet (siehe Abbildung 41). Jedem konkreten Werkzeug muß eine entsprechende Implementation mitgegeben werden, damit dieses von der lokalen Umgebung über den Builder erzeugt werden kann. Dazu sind (in fast allen Fällen) lediglich die Methoden `pCreateToolFK()` und `pCreateToolIAK()` geeignet zu überschreiben.

8.7.3 Technische Realisierung der Werkzeug-Kontext-FKs

Die Kontext-FK und jede Funktionskomponente, die im Rahmen der Dienstleistungen der Umgebung partizipieren möchte, muß von der Oberklasse `tToolFKBase` abgeleitet werden.

Diese basiert auf der Klasse tEnvFKBase die bereits als Schnittstelle für die lokale Umgebung (Singleton) zur Verfügung steht. Deshalb wird hier zunächst die Vererbungshierarchie dieser Klasse aufgezeigt. An den Seiten sind die Klassen angetragen, die jeweils von der entsprechenden Schnittstelle Gebrauch machen.

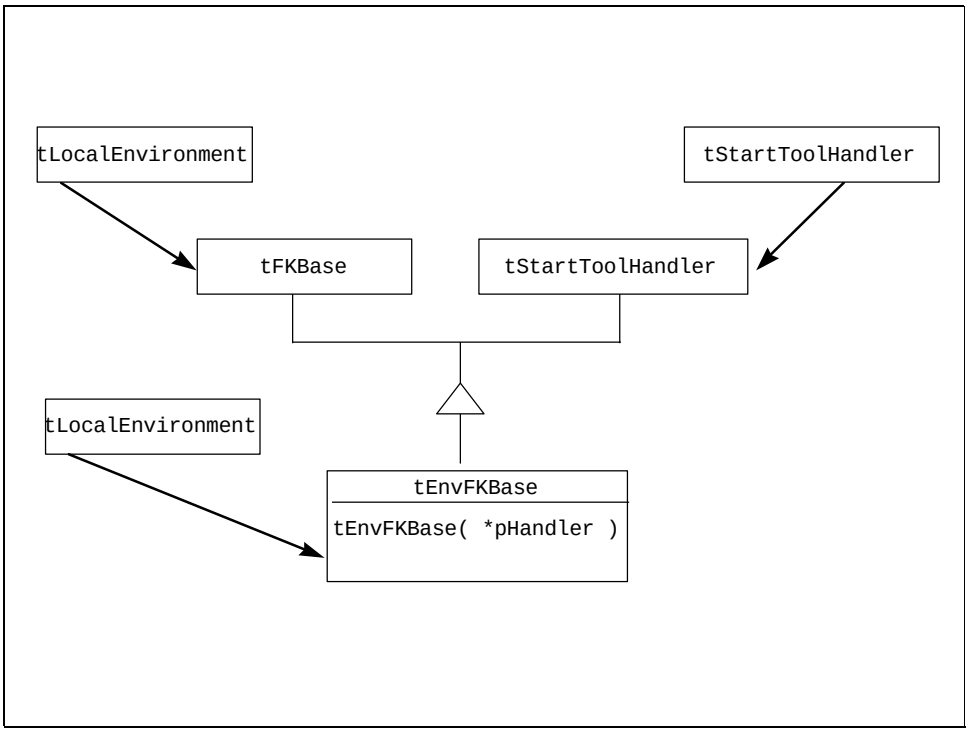

Abbildung 42 Die Vererbungsbeziehung der Klasse `tEnvFKBase` zu den Klassen `tFKBase` und `tStartToolHandler`

Im Rahmen der Beziehung *Chain of Responsibility* benutzen sich die Funktionskomponenten und die lokale Umgebung unter der Klassenschnittstelle `tStartToolHandler`, wenn es um die Benachrichtigung über Zustandsänderungen geht, wird die Klassenschnittstelle `tFKBase` von der lokalen Umgebung verwendet und wenn die lokale Umgebung auf die bereits laufenden Exemplare der Werkzeuge zugreifen will, so nutzt es dafür die Klassenschnittstelle `tEnvFKBase` (siehe Abbildung 42).

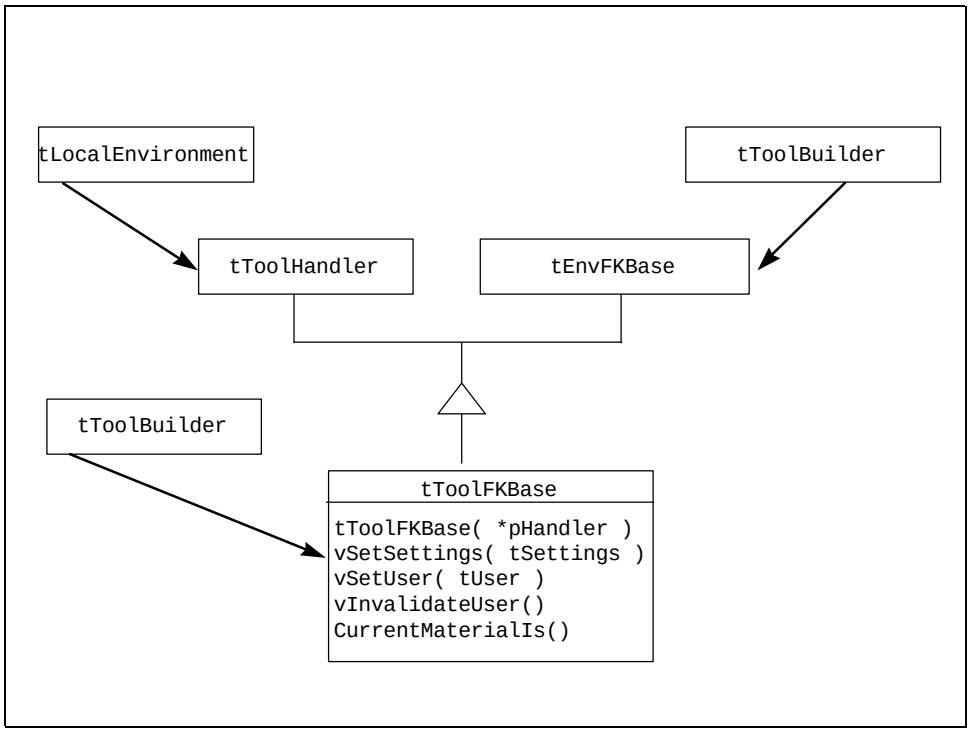

Abbildung 43 Die Vererbungsbeziehung der Klasse `tToolFKBase` zu den Klassen `tToolHandler` und `tEnvFKBase`

Ein bereits laufendes Werkzeug (d.h. ein Exemplar dieses Werkzeugs) wird von der lokalen Umgebung unter der Klassenschnittstelle tToolHandler angesprochen (vgl. Abbildung 43). Der Builder nutzt einerseits die Schnittstelle tEnvFKBase zur Speicherung der gestarteten Funktionskomponente und greift auf tToolFKBase zurück, wenn Einstellungen delegiert werden sollen.

8.7.4 Technische Realisierung der lokalen Umgebung

Das Singleton Objekt für die lokale Umgebung der Klasse `tLocalEnvironment` erbt von `tStartToolHandler` , weil es das letzte Glied dieses Prozeßraumes im *Chain of Responsibility* ist. Es erbt zusätzlich von Notifiable, weil es wie bereits von Gryczan motiviert (vgl. [Gry96]), in einer Beziehung zu den Kontext-FKs steht.

Werden die Werkzeuge des Prozesses mit dem Umgebungsprozeß konstruiert, wird zusätzlich eine Verbindung zwischen den Prozessen benötigt. Dafür tritt das Umgebungsproxy (technisch Realisierung durch die Klasse `tEnvironmentProxy`) als Stellvertreter in Erscheinung (siehe Abbildung 44). Es ist ebenfalls ein Singleton. Die Klassen `tLocalEnvironment` und `tEnvironmentProxy` stehen in einer Beobachter-Beobachtbar Beziehung, wobei das Objekt vom Typ `tLocalEnvironment` das Objekt des Typs `tEnvironmentProxy` benutzt, um in Kontakt mit dem Umgebungsprozeß zu treten. Das Umgebungsproxy benachrichtigt hingegen die lokale Umgebung über Änderungen in seinem Zustand.

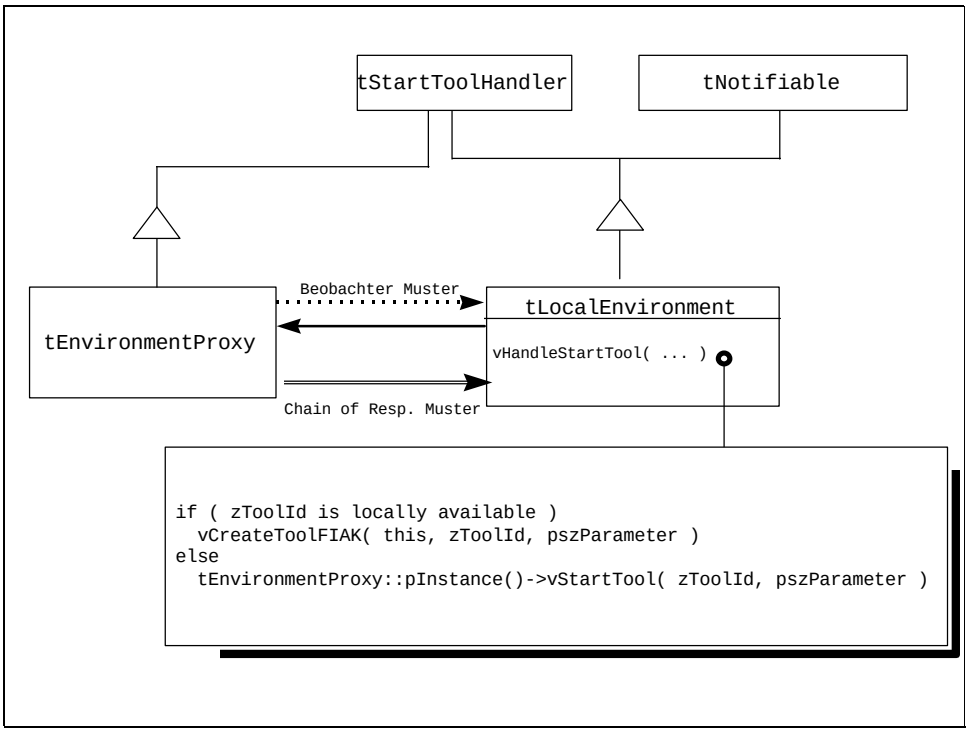

Abbildung 44 Die Beziehung der Klasse `tLocalEnvironment` zu der Klasse `tEnvironmentProxy`

Für die Kommunikation mit dem Umgebungsproxy benötigt die lokale Umgebung eine Reihe von *Callback* Funktionen, die hier nur unkommentiert aufgelistet werden.

```
class tEnvironmentProxy
{
...
protected:
     //
     // callback functions
     //
     virtualvoid vHandleEnvError();
     virtualvoid vHandleEnvConnectionLost();
     virtualvoid vHandleEnvConnected( );
     virtualvoid vHandleEnvNewUserInfo();
     virtualvoid vHandleEnvNoUser();
     virtualvoid vHandleEnvNewSettings();
     virtualvoid vHandleEnvUndefined();
...

};   // End of class tEnvironmentProxy
```

Wenn der Einsatz des Umgebungsprozesses für Werkzeuge wünschenswert ist, registriert sich die lokale Umgebung beim Umgebungsproxy für eine Reihe von Ereignissen. In diesem Fall der Implementation ist jedem Ereignis eine Callback Funktion zugeordnet. Eine Reihe von sondierenden Operationen wird vom Umgebungsproxy angeboten, um die signalisierten Zustandsänderungen abzufragen.

8.8 Leitlinien für das Zusammenfassen von Werkzeugen in Prozessen

Eine wichtige Frage bei der Zusammenstellung von Werkzeugen zu Prozessen in einem Anwendungssystem ist die nach der Entscheidung über die Werkzeugkombination. Welche Werkzeuge werden am besten in einem Prozeß kombiniert? Welche Werkzeuge werden besser in getrennte Prozesse gelegt?

Diese Frage ist die Frage nach den Schnitten im Anwendungssystem. Unter einem Schnitt soll die Trennung in unterschiedliche Prozesse verstanden werden. Nach den im Industrieprojekt gemachten Erfahrungen läßt sich dafür eine einfache Regel formulieren:

> *Werkzeuge, die auf dem gleichen Material arbeiten, sind vorzugsweise in einem Prozeß zu kombinieren.*

Die Werkzeugtypen, die für einen Materialtyp erstellt werden müssen, lassen sich auf einer abstrakten Ebene charakterisieren. Es sollte immer ein Werkzeug geben, mit dem man ein leeres Material erzeugen kann. Dieses Werkzeug wird Creator genannt. Zusätzlich dazu sollte ein Werkzeug zur Verfügung gestellt werden, welches das Material visualisieren kann. Dieses Werkzeug wird Viewer genannt. Der Editor ist das Werkzeug, welches die Veränderung der fachlichen Werte des Materials erlaubt. Da die drei Werkzeuge Tätigkeiten bezüglich eines Materials bereitstellen, läßt sich hier besonders in Bezug auf den Aspekt Code-Sharing sehr intensiv von einer Kombination in einem Prozeß profitieren.

8.9 Der Umgebungsprozeß

Genauso, wie bisherige Anwendungssysteme genau eine Umgebung für einen Anwender realisiert haben, wird auch mit Hilfe des Umgebungsprozesses pro Arbeitsplatz genau eine Umgebung repräsentiert. Der Umgebungsprozeß wird pro Arbeitsplatzrechner genau einmal gestartet und steht als verwaltende Instanz zur Verfügung.

Sinnvolle Ansammlungen von Werkzeugen werden in Prozessen realisiert und über den Umgebungsprozeß koordiniert. Die Notwendigkeit für eine koordinierende Instanz war bereits bei Anwendungssystemen, die in einem Prozeß realisiert wurden, gegeben. Es ist eine übergeordnete Instanz notwendig, die z.B. in der Lage ist Werkzeuge zu starten.

Durch die Wahl eines separaten Prozesses wurde das Problem umgangen, die Frage zu klären, in welchem Werkzeugprozeß die Umgebung angesiedelt sein soll. Außerdem ist der Umgebungsprozeß unabhängig von Werkzeugen weiter zu entwickeln und somit auch wiederverwendbar.

Wie ein Anwendungssystem pro Arbeitsplatz verwendet wurde, um einem Anwender die Software-Werkzeuge für seine Arbeit zur Verfügung zu stellen, wird jetzt pro Arbeitsplatz eine Sammlung von Werkzeugprozessen angeboten, die durch einen Umgebungsprozeß pro Arbeitsplatz koordiniert werden. Diese bilden ein Arbeitsplatzsystem.

Nachfolgend sollen einige wesentliche Aufgaben des Umgebungsprozesses betrachtet werden. Der Startvorgang gehört mit zu diesen Aufgaben.

8.9.1 Aufbau des Umgebungsprozesses

Der Umgebungsprozeß besteht aus einer Funktionskomponente, deren Aufgabe es ist, die Basiskonfiguration des Umgebungsprozesses vorzunehmen, und einem Command-Handler, der von außen eintreffende Kommunikationsanforderungen bearbeiten soll.

Die Funktionskomponente wurde dazu degradiert, lediglich die Konfiguration des Umgebungsprozesses vom einem entsprechenden Datenbanksystem anzufordern und auszuwerten. Ggf. davon abzuleitende Aktionen werden von ihr durchgeführt. Ist die Konfiguration einmal erfolgreich bearbeitet, hat die Funktionskomponente keine weiteren Aufgaben.

Der Command-Handler ist das Eingangstor des Umgebungsprozesses. Alle anderen Prozesse, die Werkzeuge realisieren, wenden sich mit Hilfe von Kommunikation an den Umgebungsprozeß. Diese Anforderungen werden im Command-Handler ausgewertet.

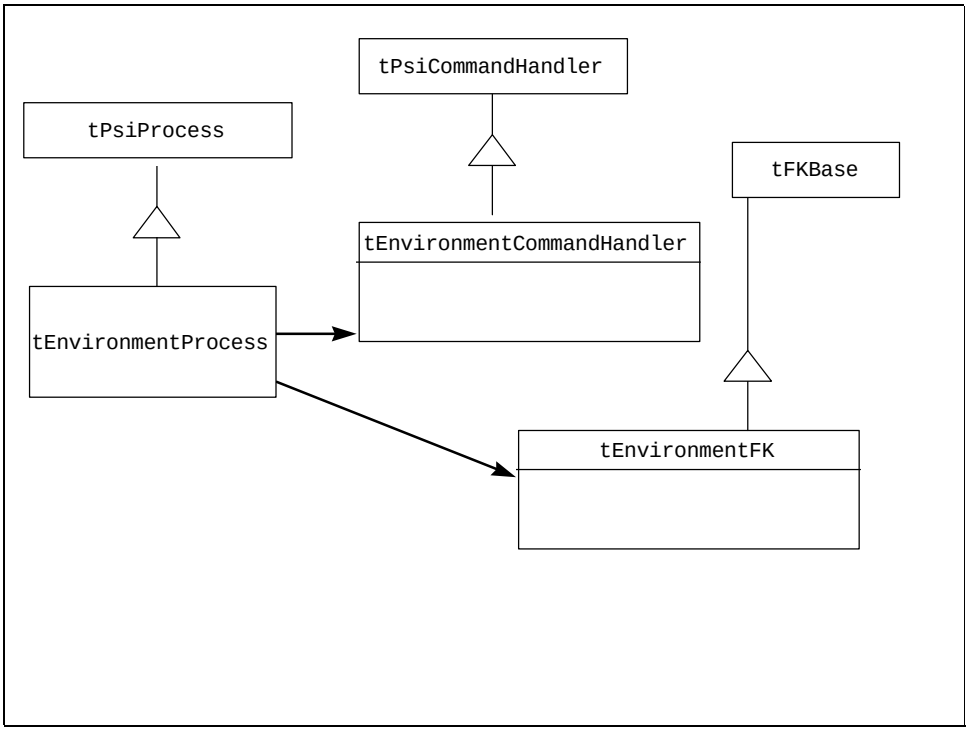

Abbildung 45 Die Beziehung der Klasse `tEnvironmentProcess` zu den Klassen `tEnvironmentCommandHandler` und `tEnvironmentFK`

Das verwendete Kommunikationssubsystem ist das PSI, deshalb wird im main() des Programms ein Objekt einer Spezialisierung vom Typ `tPsiProcess` erzeugt: `tEnvironmentProcess` (siehe Abbildung 45). Für den Command-Handler ist ebenfalls eine Oberklasse vorbereitet, die das Auswerten der Kommandos vorbereitet.

Dieser Command-Handler wartet nun auf eintreffende Kommunikation. Er kennt fünf verschiedene Klassifizierungen:

1. Tool Connect
2. User

3. Start und Stop
4. Settings
5. Service

Unter *Tool Connect* werden die Anmeldungen der Werkzeuge verstanden. Diese Anmeldung wird mit dem Verfahren des hängenden Paketes realisiert. Daraufhin kann der Werkzeugprozeß vom Umgebungsprozeß über Veränderungen informiert werden.

Unter *User* werden An- und Abmeldungen des Benutzers verstanden. Diesen Eingang der Schnittstelle verwendet ausschließlich das Anmeldewerkzeug.

Start und Stop bezeichnet die Dialoge, die sich auf das Starten und Schließen von Werkzeugen beziehen.

Die *Settings* sind für die Änderung von Einstellungen des Benutzers vorgesehen. Werkzeuge können hier Einstellungen des Benutzers absetzen, um diese vom Umgebungsprozeß abspeichern zu lassen. Damit wird ein allgemeines Verfahren zur Verfügung gestellt, welches von Seiten des Umgebungsprozesses garantiert, daß Einstellungen eines Benutzers bei der Arbeit mit einem speziellen Werkzeug an einem speziellen Material später in diesem Kontext wieder abrufbar sind.

Unter *Service* wird Kommunikation auf der Ebene von Fehlerbehebung und Fehlerortung des Umgebungsprozesses verstanden. Diese Kommunikation ist besonders wichtig beim Weiterentwickeln des Funktionsumfangs.

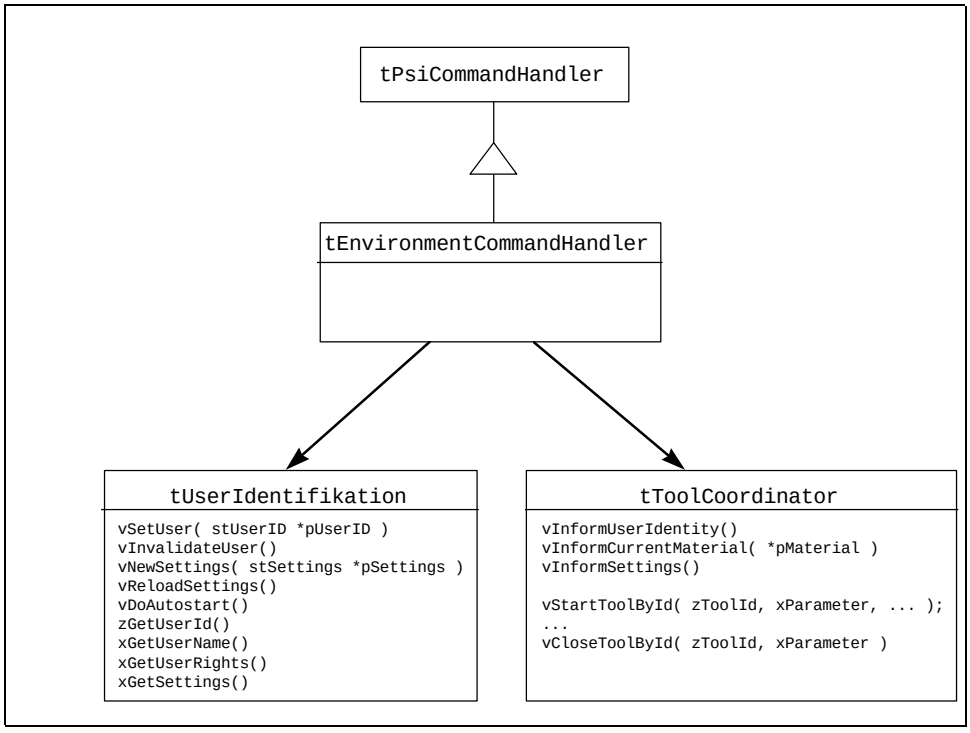

Abbildung 46 Die Beziehung der Klasse `tEnvironmentCommandHandler` zu den Klassen `tUserIdentifikation` und `tToolCoordinator`

Der Command-Handler benötigt für seine Arbeit zwei weitere Objekte, an die er die empfangenen Dialoge, die als Aufgaben interpretiert werden, delegieren kann. Das ist einerseits ein Objekt vom Typ `tUserIdentifikation` und andererseits ein Objekt vom Typ `tToolCoordinator`. Beide sind als Singleton realisiert. Die User Identifikation verkörpert die Identität des aktuell angemeldeten Benutzers. Der Tool-Coordinator führt eine Liste aller Werkzeuge, die bereits am Umgebungsprozeß angemeldet sind. Er ist außerdem dafür zuständig zu entscheiden, ob neue Werkzeugexemplare anzufordern sind bzw. ein neuer Prozeß gestartet werden muß, um ein Werkzeug zu öffnen.

8.9.2 Der Startvorgang

Damit Werkzeuge auf einer Maschine von einer Umgebung koordiniert werden können, muß diese Umgebung vorhanden sein. Konkret bedeutet dies, daß der Umgebungsprozeß aktiv sein muß, wenn ein Werkzeug- oder Automatenprozeß gestartet wird.

An dieser Stelle offenbart sich die Frage, wie das Starten der Umgebung gelöst werden soll. Einerseits könnte der Umgebungsprozeß explizit vom Anwender oder System gestartet werden. Andererseits könnte ein Werkzeug oder Automat prüfen, ob der Umgebungsprozeß bereits läuft und wenn nicht, diesen ggf. nachstarten.

Die erste Lösung besitzt den Vorteil, daß nicht jedem Werkzeug und Automaten das Wissen gegeben werden muß, wie man eine Umgebung startet. Die zweite Lösung bietet den Vorteil, daß im Fall einer unerwarteten Beendigung des Umgebungsprozesses ein Werkzeug die Umgebung wieder aktivieren könnte.

Im konkreten Anwendungsfeld wurde sich dafür entschieden, den Umgebungsprozeß explizit vom Anwender respektive dem Betriebssystem des konkreten Anwenderrechners zu starten. Dafür gibt es die folgenden Argumente:

Erstens ist es vertretbar, denn das Starten eines Anwendungssystems (wie z.B. WinWord) ist kein ungewohntes Vorgehen. Zweitens erfordert das Starten eines Prozesses wie der Umgebung, eine Reihe von Informationen (Pfad, Name des Executables, Parameter, ggf. Hilfsprozesse)

Drittens müßte der Mechanismus in jedem Werkzeug oder Automaten implementiert sein.

Und schließlich viertens könnte ein erneutes Starten zu vielfachem Starten führen.

Trotzdem möchte ich hier eine alternative Lösung darstellen, die zwar komplexer in ihrem Ablauf ist, es aber ermöglichen könnte, daß der Anwender von der Organisation in separaten Prozessen keine Kenntnis bekommt.

> *In dem Umgebungsprozeß könnte ein Mechanismus implementiert sein, der überprüft, ob es einen lokalen Nameserver gibt. Wenn nicht wird versucht, diesen nach einem allgemeinen Schema zu starten. Die Registrierung beim Nameserver mit der o.g. Überprüfung wird wiederholt versucht.*
>
> *In jedem Werkzeug könnte ein Mechanismus implementiert sein, der zuerst prüft, ob es einen Nameserver [Ble97a] gibt. Wenn nicht, kann es auch keinen Umgebungsprozeß geben, denn dieser hätte sich beim Nameserver registrieren*

lassen. Demnach wird versucht, den Umgebungsprozeß nach einem allgemeinen Schema zu starten.

Wird in dieser gedachten Konstellation ein Werkzeug gestartet, obwohl der Umgebungsprozeß nicht läuft, wird folgendes Schema ablaufen:

1. Das Werkzeug findet keinen Nameserver und startet deshalb den Umgebungsprozeß. 2. Die gestartete Umgebung findet keinen Nameserver und startet deshalb den Nameserver. 3. Die Umgebung registriert sich beim Nameserver. 4. Das Werkzeug erfragt beim Nameserver den Prozeß der Umgebung. 5. Das Werkzeug verbindet sich mit der Umgebung.

8.9.2.1 Starten von Werkzeugen

Das Starten von Werkzeugen ist die zentrale Aufgabe des Umgebungsprozesses neben deren Koordination. Werkzeuge werden einerseits durch das Anmelden des Anwenders gestartet, sogenanntes Autostarten. Andererseits starten Werkzeuge selber wieder Werkzeuge.

Gryczan schreibt dazu

„ Die Umgebung ist darüber hinaus für die Erzeugung der Werkzeuge zuständig. [...] Die Umgebung kann diese Funktionalität auch an einen Werkzeugverwalter delegieren." ([Gry96], Seite 146)

In einem Executable können mehrere Werkzeuge im Sinne des Werkzeugbegriffs implementiert sein. Selbst das Starten eines Werkzeuges, welches im selben Executable implementiert ist, wird durch den Umgebungsprozeß koordiniert.

Werkzeuge werden auf verschiedene Arten identifiziert. Eine Möglichkeit besteht darin, ein Werkzeug über die Werkzeug-ID, welche eindeutig ist, zu starten. Eine andere Möglichkeit besteht durch die Angabe eines Materialtyps und eines Verwendungszwecks.

Das Starten von Werkzeugen über eine eindeutige Identifikation besitzt den Vorteil, daß dem startenden Werkzeug „klar ist“, welches andere Werkzeug gestartet wird. Außerdem kann diese Variante eingesetzt werden, wenn eine Spezifikation über Material und Verwendungszweck nicht möglich ist.

Eine Identifikation des Werkzeugs über einen Werkzeugnamen entkoppelt den Starter vom Gestarteten insofern, als daß unter dem Namen eine andere Werkzeugvariante verfügbar sein könnte, was bei der eindeutigen ID nicht der Fall sein kann.[37]

Die Möglichkeit, ein Werkzeug über den Typ des Materials und den intendierten Verwendungszweck zu starten, ermöglicht es, die verfügbaren Werkzeuge im Rahmen der Weiterentwicklung des Anwendungssystems zu ergänzen oder zu ersetzen. Insbesondere bietet es sich an, zuerst rudimentäre Werkzeuge mitzuliefern, die nur einen geringen Umfang von Tätigkeiten zulassen. Später können komplexe und besonders an den Anwendungsfall angepaßte Werkzeuge geliefert werden. Dadurch, daß in anderen

[37] Das dadurch möglicherweise entstehende Typproblem liegt in der Verantwortung der Systementwickler.

Werkzeugen kein direkter Bezug genommen wird, kann das Austauschen ohne Änderung der bestehenden Werkzeuge geschehen. Der Umgebungsprozeß wird nach der Installation einfach auf die neuen Werkzeuge zugreifen.

Eine Erweiterung dieser Variante besteht darin, dem Anwender eine Liste von Werkzeugen anzubieten, die im Rahmen seiner beabsichtigten Bearbeitung zur Verfügung stehen. Dies kann zum Beispiel mit Hilfe eines Kontextmenüs geschehen. Ein Werkzeug sollte dann allerdings als Standardwerkzeug für die Bearbeitung gekennzeichnet sein, damit die Auswahl des Werkzeugs nicht immer notwendig ist.

Es soll noch einmal zusammengefaßt werden:

Das Starten eines Werkzeugs kann auf dreierlei Arten unterschieden werden.

1. Das Werkzeug oder der Automat, der das Starten eines Werkzeugs anfordert, weiß genau um welches Werkzeug es sich handelt. In diesem Fall wird das Starten durch eine eindeutige Werkzeug-ID spezifiziert.

2. Das Werkzeug oder der Automat kennt den Namen des Werkzeugs. Namen können mehrfach vergeben werden. Es kann also zweimal ein Anmeldewerkzeug „LoginTool" geben aber jedes besitzt genau eine eindeutige Identität.

3. Das Werkzeug oder der Automat kennt nur den Typ des Werkzeugs und den Typ des Materials, wobei beide Typen frei definierbar sind. Dann sucht der Umgebungsprozeß nach einer Werkzeugbeschreibung, die zu dem angeforderten Werkzeugtyp und gleichzeitig zu dem bearbeiteten Materialien paßt.

Diese Dienste bietet die Umgebung an. Das beauftragende Werkzeug wird über den Erfolg des Startauftrags informiert.

8.9.2.2 Starten von Automaten

Während das Starten von Werkzeugen bereits in herkömmlichen Anwendungssystemen einen normalen Vorgang darstellt, ist auch hier das Starten von Automaten jeweils individuell gelöst worden. Grundsätzlich ergeben sich eine Reihe von offenen Fragen und Problemen bei der Betrachtung der Frage, wie und wann man Automaten starten kann.

Einerseits ist die Frage nach dem Zieladreßraum ungeklärt. Werkzeuge, die vom Anwender gestartet werden, sollen immer auf derselben Maschine laufen, auf der auch der Anwender arbeitet. Dies ist bei Automaten nicht unbedingt der Fall. Soll z.B. eine leistungsstarke Maschine verwendet werden, muß diese erst genau spezifiziert werden.

Wenn ein Automat erst einmal gestartet wurde, ist nicht klar, wer für das Beenden eines Automaten zuständig ist. Da Automaten teilweise auch von mehreren Anwendern genutzt werden können, kann nicht festgelegt werden, daß der Automat z.B. nur so lange läuft, wie der Anwender angemeldet ist, der ihn gestartet hat.

Untersucht man die Frage des Beendens von Automaten näher, stellt sich heraus, daß viele Automaten gar nicht dafür konstruiert wurden, daß man sie wieder „normal“ beendet. Viele Automaten werden einfach mitten im Betrieb beendet oder sollen gar nicht beendet werden. Außerdem ist es in vielen Fällen sehr aufwendig zu ermitteln, ob die Kritierien für ein geordnetes Beenden erfüllt sind. Außerdem wird, wenn das

Beenden eines Automaten möglich sein soll, ein Werkzeug benötigt, mit dem man den Automaten ausschalten können muß.

Grundsätzlich ungeklärt ist die Frage, wie man in entfernten Adreßräumen überhaupt einen Automaten durch ein Werkzeug auf dem Arbeitsplatzrechner starten kann. Dafür ist im allgemeinen ein Stub notwendig, das diesen Prozeß dort starten kann. Dieses Stub kann man wieder als Automat auffassen, der dort eigentlich erst gestartet werden muß.

Es wurden deshalb folgende Prämissen formuliert:

- Automaten laufen auf entfernten Maschinen
- das Starten von Automaten liegt nicht im Bereich eines Anwendungs(!)systems
- Werkzeuge können nur über das Erreichen bzw. Nichterreichen eines Automaten über das Kommunikationsmedium informieren

Es ist deshalb bewußt außerhalb des realisierten Umgebungsprozesses gelegt worden, Automaten, die Bestandteil des Anwendungssystems sind, durch diesen zu starten. Diese Entscheidung ist vor dem Hintergrund des Anwendungsfeldes Call Center deswegen zu begründen, weil die Automaten des Einsatzgebietes ausschließlich technische Automaten darstellen. Würden auch Automaten im lokalen Anwendungsfeld eines Arbeitsplatzsystems möglich sein, könnte die oben skizzierte Problematik auf folgende Weise gelöst werden: Automaten werden grundsätzlich lokal gestartet.

8.9.3 Anbindung der Werkzeuge

Die Anbindung der Werkzeuge an den Umgebungsprozeß erfolgt nach einem allgemeinen Muster, das im Rahmen einer Musterarchitektur realisiert wurde. Das heißt, daß mit Hilfe dieses Musters eine Reihe von beliebigen erst später entworfenen Werkzeugen an eine Umgebung angebunden werden kann.

Ziele des Musters:

- Wiederverwendbare Anbindung
- Wahl ob Mehrprozeß- oder Einprozeßraum
- Unabhängige Konfektionierung der Werkzeuge

Es war besonders wichtig, daß bei der Erarbeitung der Anbindung der Werkzeuge an die Umgebung eine Struktur geschaffen wird, die sofort für weitere Werkzeuge verwendet werden kann.

Während bei einem Anwendungssystem die Umgebung als übergeordnetes Objekt für alle Kontext-FK Objekte anzusehen ist, wird hier anstelle dessen von der lokalen Umgebung ein Umgebungsproxy verwendet. Dieses Proxy hält über asynchrone Kommunikation Verbindung mit dem Umgebungsprozeß.

Der Vorteil dieser Lösung besteht darin, daß bei der Entwicklung eines Werkzeugs noch nicht entschieden werden muß, ob dieses nun in einem separaten Prozeß alleine implementiert wird, oder vielleicht später mit einer ganzen Kollektion von ähnlichen Werkzeugen in einem gemeinsamen Prozeß laufen wird. Ebenso kann das gesamte Anwendungssystem immer noch innerhalb eines Prozesses zusammengestellt werden, wenn besondere Rahmenbedingungen die Trennung verbieten.

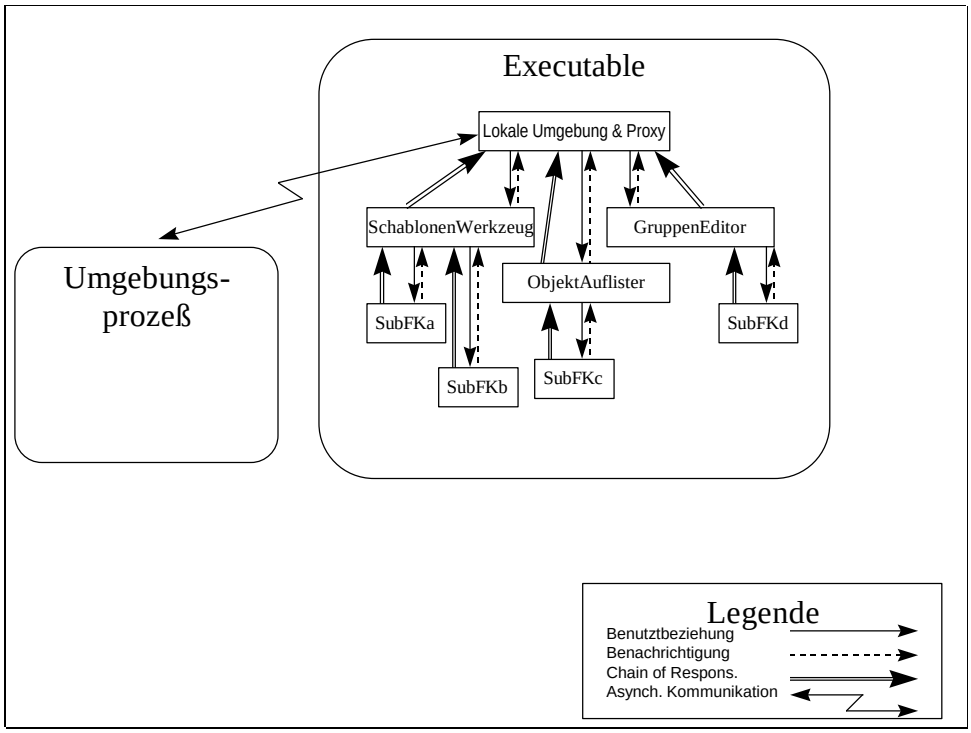

Abbildung 47 Beziehung Werkzeug-FK und Umgebungsproxy

In der Abbildung 47 ist zu sehen, wie die Beziehungen der einzelnen Objekte zueinander sind. Die Sub-FKs sind mit der jeweiligen Kontext-FK und diese wiederum mit dem Umgebungsproxy über das Chain of Responsibility Muster miteinander verbunden. Dieses wird zum Starten von weiteren Werkzeugen verwendet.

Kann eine Verantwortlichkeit nicht erfüllt werden, gibt das Umgebungsproxy den Auftrag über asynchrone Kommunikation an den Umgebungsprozeß weiter.

Implementiert ein Executable ein Werkzeug z.B. zum Betrachten eines Materials, dann muß dieses Werkzeug dafür vorgesehen sein, in mehreren Exemplaren zu laufen (s.o.). Dies wird o.B.d.A. aus Effizienzgründen gefordert. Nur so wird es vermieden, daß ein Werkzeug mehrfach gestartet und somit im Speicher verwaltet wird. Für die Verwaltung dieser Exemplare wird im Executable ebenfalls das Umgebungsproxy verwendet. Dieses hält eine Liste der derzeit aktiven Werkzeugexemplare vor.

8.9.4 Konzept Benutzeridentität

Im Rahmen der Entwicklung hat sich gezeigt, daß für die Arbeit mit komplexen Anwendungssystemen zur Unterstützung von qualifizierten Arbeitsplätzen auch die Bereitstellung einer Benutzerunterscheidung, die Vergabe von benutzerspezifischen Rechten und die Speicherung von benutzerbezogenen Einstellungen notwendig ist.

An dieser Stelle sei erwähnt, daß jeweils ein Umgebungsprozeß genau einem Anwender zugeordnet ist. Somit trägt der Umgebungsprozeß als eine Einstellung die Anwenderidentität in sich und kann diese an die Werkzeuge weitergeben. Pro Arbeitsplatzsystem gibt es genau einen Umgebungsprozeß. Demnach kann es pro Arbeitsplatzsystem genau einen Benutzer geben.

Es wird davon ausgegangen, daß die Werkzeuge nur dann verwendet werden können, wenn dem Arbeitsplatzsystem die Identität des Benutzers bekannt ist. Das läßt sich fachlich folgendermaßen motivieren: Nur mit Hilfe der Benutzeridentität kann entschieden werden, welche personenspezifischen Befugnisse vorhanden sind, damit bestimmte Werkzeugfunktionalitäten freigegeben werden können. Einstellungen werden benutzerbezogen abgelegt.

Es gibt ein Anmeldewerkzeug, welches dem Benutzer erlaubt, dem System seine Identität mitzuteilen (siehe Abbildung 48, vergleiche auch Kapitel 4). Diese Information wird dem Umgebungsprozeß weitergegeben, um sie für alle Werkzeuge verfügbar zu machen. Die Umgebung besorgt sich aufgrund des angemeldeten Anwenders dessen Benutzerrechte und reicht auch diese an alle Werkzeuge weiter.

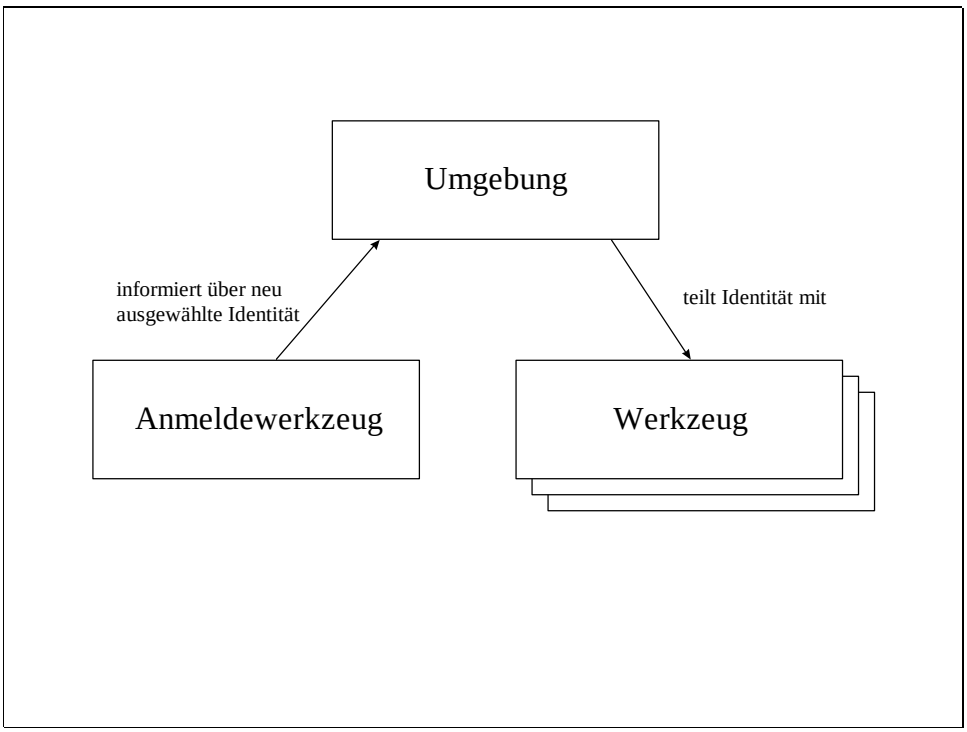

Abbildung 48 Verwendung eines Anmeldewerkzeugs

8.10 Zusammenfassung

Die Koordination von Werkzeugen, die in getrennten Prozessen laufen, und die Bereitstellung von Benutzereinstellungen ist notwendig, wenn ein modulares verteiltes Anwendungssystem erstellt werden soll.

Wenn der Anwender die Möglichkeit besitzt, seinen physikalischen Arbeitsplatz innerhalb eines Netzwerkes zu wechseln und gleichzeitig die Werkzeuge, die für seinen Arbeitsplatz angeboten werden, als Komponenten im Netzwerk zur Verfügung stehen, dann ist es notwendig, diese Dienstleistungen gezielt zu unterstützen. Benutzerspezifische Informationen müssen so verwaltet werden, daß sie an jedem Arbeitsplatz des Netzwerkes verfügbar sind.

Zur Koordination von prozeßmäßig getrennten Werkzeugen wird ein lokal laufender Umgebungsprozeß benötigt, der es den Werkzeugen ermöglicht, die Kommunikation untereinander aufzunehmen und gleichzeitig Basisdienste, wie die Bereitstellung der Benutzeridentität, zur Verfügung stellt.

Mit Hilfe eines jeweils lokal laufenden Umgebungsprozesses, wird es möglich, wechselnde Arbeitsplätze und benutzerbezogene Einstellungen innerhalb eines Netzwerkes anzubieten. Der Umgebungsprozeß kennt den Ort, an dem die benutzerspezifischen Informationen abgelegt werden.

8.11 Ausblick

Im Rahmen der Entwicklung des Umgebungsprozesses und der Integration von unterschiedlichen Teilanwendungen zu einem Anwendungssystem sind bereits Fragestellungen aufgetaucht, die leider nicht im Zusammenhang dieser Arbeit geklärt werden konnten. Der Vollständigkeit halber sollen diese Punkt hier aber erwähnt werden, da sie auch für weitere Forschung an diesem Thema interessant sein können.

Die Einbindung der Desktop-Metapher der unterschiedlichen Betriebssysteme in die Umgebung stellt ein ganz wesentliches Vorhaben im weiteren Vorgehen dar. Bereits im jetzigen Stadium gibt es Verwandtschaften beim Schablonenkonzept der Betriebssysteme und dem des hier vorgestellten Anwendungssystems. Ebenso könnten Auflister- und Ordnerkonzept auf Gemeinsamkeiten untersucht werden.

Es besteht bereits seit längerem die Idee (vgl. [Ble95]), den jeweils durch das Betriebssystem zur Verfügung gestellten Desktop durch einen separaten und noch zu entwickelnden Prozeß an das Anwendungssystem anzubinden. Dadurch würde die notwendige Unabhängigkeit des Anwendungssystems gewahrt und es wäre nicht erforderlich, betriebssystemspezifische Anpassungen des gesamten Anwendungssystems vorzunehmen. Vielleicht wäre es in diesem Zusammenhang notwendig, die vermittelnden Aufgaben des Umgebungsprozesses zu erweitern.

In jedem Fall könnte man sich aber vorstellen, einen Desktop, wie man ihn von den bekannten Betriebssystemen her kennt, durch ein Desktop-Werkzeug auch für das hier skizzierte Anwendungssystem zu realisieren. Die Integration in bestehende Desktops wäre ein weiterer Schritt. Hierbei würde die Sichtbarkeit in den Hintergrund treten und der Werkzeug-Charakter des Prozesses in einen Automaten-Charakter übergehen.

9 Zusammenfassung und Ausblick

Mit diesem Kapitel soll die vorliegende Arbeit abgeschlossen werden, indem die wesentlichen Ergebnisse noch einmal zusammengefaßt werden. Offene Punkte, die sich vielleicht auch als Themen für weiterführende Arbeiten eignen, werden anschließend aufgezeigt.

Ziel dieser Arbeit war die Konkretisierung des Umgebungsbegriffs der Software-Entwicklungsmethode WAM und die Erweiterung der Konstruktionstechnik um Konzepte zur Modellierung verteilter Anwendungssysteme. Als Erfahrungshintergrund diente dabei ein Industrieprojekt im Bereich der Telefonieanwendung, in dessen Rahmen Software-Werkzeuge mit Hilfe der Methode WAM entwickelt wurden.

Zu Beginn dieser Arbeit wurde das Anwendungsgebiet Call Center eingeführt und gezeigt, welche besonderen Anforderungen dieses Einsatzfeld an die Konstruktion von Anwendungssystemen stellt.

Im Anschluß daran wurden die bestehenden Metaphern Automat und Werkzeug dahingehend um Konzeptionsmuster erweitert, daß sie für den Einsatzkontext des verteilten, technisch eingebetteten Anwendungssystems die notwendige Grundlage bieten, reale technische Geräte zu modellieren. Das Konstruktionsmuster technischer Automat bildete dafür die Ausgangsbasis. Mit dem Konzept Sonde konnte die Problematik adressiert werden, nur partielle Zustandsräume von Automaten angemessen zugänglich zu machen. Durch die Erweiterung des Werkzeugs um das Konzeptionsmuster Einstellwerkzeug ist es gelungen, anwendungsfachlich zu motivieren, warum es spezielle Werkzeuge für Automaten gibt, und wie diese konstruiert werden können.

In einem weiteren Schritt wurde die in den Konstruktionsmustern der Methode WAM zugrunde gelegte Kopplung untersucht. Sie stellt den Schlüssel für die Übertragung von Anwendungssystemen in verteilte Umgebungen dar. Es wurde festgestellt, daß die bisher praktizierte funktionsorientierte Kopplung nicht angemessen in den Mehrprozeßraum übertragen werden kann. Sie sollte deshalb durch eine ereignisorientierte Kopplung

ersetzt werden. Diese kann dann leicht zu einer Verteilung von Komponenten des Anwendungssystems in den Mehrprozeßraum genutzt werden.

Die Kopplung von Werkzeugen und Automaten steht thematisch im Mittelpunkt bei der Überführung von Einprozeßraum zum Mehrprozeßraum. Es wird gezeigt, wie mit Hilfe von Proxies die ereignisorientierten Schnittstellen über ein Kommunikationsmedium abgebildet werden können. Zusätzliche Ereignisse werden dazu notwendig.

Der Einfluß auf die Benutzungsschnittstelle darf dabei nicht vernachlässigt werden. Bereits im Benutzungsmodell muß die Verteiltheit berücksichtigt werden. Nur so kann dem Anwender ein angemessener Eindruck vom Verhalten des Arbeitsplatzsystems gegeben werden.

Der Umgebungsprozeß wurde als Instanz eingeführt, die die Koordination von getrennten Werkzeugen übernimmt. Dienste wie Identitätsverwaltung, Bereitstellung von Einstellungen und Rechteverwaltung wurden eingeführt.

Ansatzpunkte für weiterführende Arbeiten stellen sich wie folgt dar:

Die etablierte Aufgliederung von Automaten in separater Prozesse hat die Frage aufgeworfen, wie Automaten entfernt gestartet werden können. Es könnte untersucht werden, inwiefern es fachlich in die Methode WAM eingegliedert werden kann, solche Fähigkeiten zu realisieren, und die damit verbundenen Konsequenzen abzuschätzen.

Erneut unbeantwortet geblieben (vgl. [Ble95]) ist die Frage nach der Integration eines Anwendungssystems, das nach WAM konstruiert wurde, mit bestehenden Anwendungssystemen, die nicht nach dieser Methode entwickelt wurden. Dies betrifft sowohl die Einbindung von externen Werkzeugen, als auch der Integration von Bestandteilen, die bereits vom Betriebssystem zur Verfügung gestellt werden. Hierbei ist insbesondere die Integration des Desktops zu nennen.

Letztendlich ist offen geblieben, welche Notwendigkeit einer direkten Kommunikation zwischen Werkzeugen unterstützt werden muß. Im jetzigen Stadium erlaubt das Modell nur die indirekte Kommunikation über den Umgebungsprozeß.

10 Literaturverzeichnis

[BAS89] Bundesanstalt für Arbeitsschutz Hrsg., „Arbeitswissenschaftliche Erkenntnisse Nr. 2/79 Bildschirmarbeitsplätze“, 2. Überarbeitete Auflage, Bundesanstalt für Arbeitsschutz Dortmund 1989

[Ber96] Philip A. Bernstein, „Middleware A Model for Distributed System Services“, February 1996, Volume 39, No. 2 CACM

[BK92] M. Beaudouin-Lafon, A. Karsenty, „Transparency and Awareness in a Real-Time Groupware System“, UIST’92 ACM

[Ble95] Bleek, Wolf-Gideon, „Oberflächengestaltung für nach WAM entwickelte Anwendungssysteme“, Studienarbeit, Universität Hamburg - Fachbereich Informatik - Arbeitsbereich Softwaretechnik, 1995

[Ble96] Bleek, Wolf-Gideon, „Technische Realisation des Umgebungsprozesses“, Micrologica 1996

[Ble97a] Bleek, Wolf-Gideon, „Vom Einprozeßsystem zum Mehrprozeßraum“, Universität Hamburg, 1997, Bestandteil der Diplomarbeit

[Ble97b] Bleek, Wolf-Gideon, „Der Environment-Prozeß - Koordinierung getrennter Werkzeuge“, Universität Hamburg, 1997, Bestandteil der Diplomarbeit

[Ble97c] Bleek, Wolf-Gideon, „Benutzerbezogene Einstellungen“, Universität Hamburg, 1997, Bestandteil der Diplomarbeit

[Bud91] R. Budde, H. Züllighoven, „Software Tools in a Programming Workshop“ in C. Floyd et al. (Hrsg.) „Software Development and Reality Construction“, Springer 1991

[But95] K. Butler, „Designing Deeper: Towards a User-Centered Development Environment“, University of Michigan, DIS’95 ACM

[BZ90] R. Budde, H. Züllighoven, „Softwarewerkzeuge in einer Programmierwerkstatt“, München, Wien, R. Oldenbourg Verlag, 1990

[Chi95] „Collaborative Realtime Process Management“, SIGCHI Bulletin Volume 27, Number 3, ACM Press July 1995

[Chi96] „The Denver Model for Groupware Design“, SIGCHI Bulletin Volume 28, Number 1, ACM Press January 1996

[Dah95] van Dahle, H., "Neue Call Center-Technik für Direktmarketing-Agentur", TeleTalk 11/95, S. 20

[DC91] P. Dewan, R. Choudhary, „Primitives for Programming Multi-User Interfaces“, UIST’91 ACM

[Dud90] „Der Duden: in 12 Bänden Fremdwörterbuch“, Bibliographisches Institut, Mannheim, Leipzig, Wien, Zürich, Dudenverlag, 1990

[Dud91] „DUDEN Die deutsche Rechtschreibung“, Bibliographisches Institut, Mannheim, Leipzig, Wien, Zürich, Dudenverlag, 1991

[Erg94] „Ergonomie & Informatik“ Mitteilungen des GI Fachausschusses 2.3 „Ergonomie in der Informatik“, Nr. 21 ISSN 0940-1210, März 1994

[Fis81] Das neue Fischer Lexikon in Farbe, Fischer Taschenbuch Verlag, Frankfurt am Main 1981

[Flo84] C. Floyd: „A Systematic Look at Prototyping“, in Approches to Prototyping, Edited by R. Budde, K. Kuhlenkamp, L. Mathiassen, H. Züllighoven, Berlin, Heidelberg, Springer Verlag 1994, 1-18

[Flo94a] C. Floyd, „Evolutionäre Systementwicklung und Wandel in Organisationen“, Der GMD-Spiegel 3/94 (September 1994), 36-40

[Flo94b] C. Floyd, „Einführung in die Softwaretechnik - Skript zur Vorlesung“, Universität Hamburg, Fachbereich Informatik, Version vom Oktober 1994

[Fri96] N. Fricke, „Das Projekttagebuch - Ein Fallbeispiel für eine verteilte Anwendung nach WAM“, Studienarbeit, Universität Hamburg - Fachbereich Informatik - Arbeitsbereich Softwaretechnik, Oktober 1996

[GHJ+95] E. Gamma, R. Helm, R. Johnson, J. Vlissides, „Design Patterns - Elements of Reusable Object-Oriented Software“, Addison-Wesley 1995

[Gry96] G. Gryczan, „Prozeßmuster zur Untestützung kooperativer Tätigkeit“, DUV 1996

[GS94] Galvin, A. Silberschatz, „Operating System Concepts“, 4th Edition, Addison-Wesley Publishing Company, 1994

[Jac91] I. Jacobson, „Object-Oriented Software Engineering“, Addison-Wesley 1992

[Ker89] H. Kerner (Hrsg.), Rechnernetze nach ISO-OSI, CCITT, selbstverlag 1989

[KGZ94] K. Kilberth, G. Gryczan, H. Züllighoven, „Objektorientierte Anwendungsentwicklung, 2. Auflage, Vieweg 1994

[Kos95] Koschek, H., "Micrologica Programmierrichtlinien für C++", Version 1.0-3, Micrologica GmbH, interne Arbeitsanweisung QS-07-10, 1995

[Kra95] C. Kracke, „Prozeß System Interface“, Micrologica, Bargteheide 1995

[Kuh96] Kuhn, K., "Call Center für die Börseneinführung der Telekom-Aktie", Midrange MAGAZIN, November 96, S. 70ff

[Kuh97a] Kuhn, K., "Deutsche Telekom setzt auf Call Center: Das Telefon wiederentdeckt", Gateway 1/97, S. 62

[Kuh97b] Kuhn, K., "Deutsche Telekom AG: Das Call Center für die Telekom-Aktie", ISDN-Report, 1/97, S. 23ff

[Lil95] C. Lilienthal: „Diplomarbeit: Konzeption und Realisierung eines an der Anwendungssprache orientierten Hilfesystems nach der Werkzeug-Material Metapher“, Universität Hamburg 1995

[LS96] C. Lilienthal, W. Strunk, „Documenting Frameworks by visualizing dynamics“, Beitrag zur TOOLS 96

[Maa91] S. Maaß: „Computergestützte Kommunikation und Kooperation“ in H. Oberquelle (Hrsg.) „Kooperative Arbeit und Computerunterstützung - Stand und Perspektiven“, Verlag für Angewandte Psychologie, Göttingen, Stuttgart 1991

[Maa94] S. Maaß: „Maschine, Partner, Medium, Welt ... Eine Leitbildgeschickte der Software-Ergonomie“, H.D. Hellige (Hrsg.) „Leitbilder der Informatik- und Computer-Entwicklung“, Bremem 1994

[Maf96] S. Maffeis, „On Building Highly Available CORBA Applications“, Component User’s Conference, München 1996

[Mey88] B. Meyer, „Objektorientierte Softwareentwicklung“, Hanser München 1988

[Obe87] H. Oberquelle: „Sprachkonzepte für benutzergerechte Systeme“, Springer, Berlin, Heidelberg, New York, 1987

[Obe91a] H. Oberquelle: „Kooperative Arbeit und menschengerechte Groupware als Herausforderung für die Software-Ergonomie“, in H. Oberquelle (Hrsg.) „Kooperative Arbeit und Computerunterstützung - Stand und Perspektiven“ , Verlag für Angewandte Psychologie, Göttingen, Stuttgart 1991

[Obe91b] H. Oberquelle: „CSCW- und Groupware-Kritik“, in H. Oberquelle (Hrsg.) „Kooperative Arbeit und Computerunterstützung - Stand und Perspektiven“ , Verlag für Angewandte Psychologie, Göttingen, Stuttgart 1991

[Obe94] H. Oberquelle, E. Eberleh, R. Oppermann: Einführung in die Software-Ergonomie, de Gruyter 1994

[OffSys95] Offene Systeme 1995, Ausgabe 4, 217-223

[OMG93] Object Management Group, Object Management Architecture Guide, John Wiley & Sons, Inc., 1993

[OMG94] Object Management Group, Common Object Services Specification, Volume 1. Revision 1.0, OMG technical document, 94-01-01, 1994

[OMG95] Object Management Group, Common Object Request Broker: Architecture and Specification. Revision 2.0, OMG technical dcoument 96-03-04, 1995

[Pat91] J. Patterson, „Comparing the Programming Demands of Single-User and Multi-User Applications“, UIST’91 ACM

[Per96] R. Perry, „The Displayer/Data Pattern-Language: A Software Engineering Technique for GUI/Application Decoupling“, Tel-Aviv University Department of Computer Science, November 1996

[Ree96] T. Reenskaug, „Working with Objects“, Manning Publications, 1996

[Rie95] D. Riehle, „Muster am Beispiel der Werkzeug und Material Metapher“, Diplomarbeit, Universität Hamburg - Fachbereich Informatik - Arbeitsbereich Softwaretechnik, 1995

[Rie96] D. Riehle, „Bureaucracy - A Composite Pattern“, Ubilab, Union Bank of Switzerland, 1996

[Rie96] Riede, S., "DSC-Prozep - Feinkonzept, Version 1.1", Micrologica GmbH, interne Dokumentation, 1996

[RSB+96] D. Riehle, W. Siberski, D. Bäumer, D. Megert, H. Züllighoven, „The Atomizer - Efficiently streaming objects“, Ubilab, Union Bank of Switzerland, 1996

[RW96] S. Roock, H. Wolf: „Konzeption und Implementierung eines ‘Reaktionsmusters’ für objektorientierte Softwaresysteme“, Studienarbeit Universität Hamburg 1996

[RZ95] D. Riehle, H. Züllighoven, „A Pattern Language for Tool Construction and Integration Based on the Tools and Materials Metaphor“, In J. Coplien et al. (hrsg.): „Pattern Languages of Programs“, Addison Wesley 1995

[RZ96] D. Riehle, H. Züllighoven, „Understanding and Using Patterns in Software Development“, in K. Lieberherr, R. Zicari, „Theory and Practice of Object Systems“, Special Issue Patterns, Guest Editor S. Berczuk, Volume 2, Number 1, 1996, pp. 3-13

[Sch89] H. H. Schulze, „Computer Enzyklopädie: Lexikon und Fachwörterbuch für Datenverarbeitung und Telekommunikation“, Band 1-6, Rowohlt Taschenbuch Verlag 1989, Reinbek bei Hamburg

[Sch95] D. Schmidt, „Using Design Patterns to Develop Reusable Object-Oriented Communication Software“, October 1995, Volume 38, No. 10 CACM

[Sch96] Schneegans, M., "Kosten/Nutzen-Analyse von Call-Centern: Mehr Gewinn am Telefon", Gateway 10/96, S. 50

[Sch97] Schneegans, M., "Gleiche Daten für alle - Computertelefonie im Call Center", Gateway 3/97, S. 142 - 143

[SN92] K. Sullivan, D. Notkin, „Reconciling Environment Integration and Software Evolution“, ACM Transactions on Software Engineering, 1 (1992) 3, S. 229-268

[Tan97] A. S. Tanenbaum, „Computernetzwerke“, Prentice Hall 1997

[Thi96] Thießen, S., "FSM-Klassen - Dokumentation, Version 1.0-1", Micrologica GmbH, interne Dokumentation, 1996

[Tre96] M. Tresch, „Middleware: Schlüsseltechnologie zur Entwicklung verteilter Informationssysteme“, Informatik-Spektrum 19:249-256 (1996), Springer Verlag

[Wag96] B. Wagner, „Black-Box Reuse within Frameworks based on Visual Programming“, Component User’s Conference, München 1996

[Wul95] M. Wulf: „Konzeption und Realisierung einer Umgebung zur Koordination rechnergestützter Tätigkeiten in kooperativen Arbeitsprozessen“, Diplomarbeit Juli 1995, Universität Hamburg, FB Informatik

[WW94] M. Wulf, D. Weske, „Konzepte zur Materialversorgung...“, Studienarbeit März 1994, Universität Hamburg, FB Informatik