Kai Schäfer, F. Wilhelm Bruns

Rechnergestützte Übergänge zwischen gegenständlichen und abstrakten Modellen produktionstechnischer Systeme

Dritter Zwischenbericht zum DFG Forschungsprojekt RUGAMS

artec-paper 67, Februar 1999

Antragsteller: Professor Dr.-Ing. F. Wilhelm Bruns,

DFG-Geschäftszeichen: BR 1556/2-3

Berichtszeitraum: 1.2.1998 - 31.1.1999

Unter Mitarbeit von: Volker Brauer
Kai Schmudlach
Jürgen Huyer
Martin Faust
Wolfgang Tieben

Abschlußbericht

Der vorliegende Abschlußbericht dokumentiert den Fortschritt des Projekts im dritten Projektjahr (Februar 1998 - Januar 1999).

1. Zielsetzung

Ziel des Projekts ist die Entwicklung eines rechnergestützten Modellierers, der es ermöglicht, mit stofflichen Bausteinen ein gegenständliches Modell eines technischen Systems aufzubauen und synchron dazu ein strukturell und funktional repräsentatives Modell im Rechner zu erzeugen, das dort leicht variierbar und analysierbar ist. Über einen Datenhandschuh erfaßte Hand- und Fingerbewegungen führen zu Operationen auf bereits vorhandenen rechnerinternen Modellfragmenten, die zu einem Gesamtmodell zusammengesetzt und in Simulationsexperimenten weiter verwendet werden können. Damit soll der Modellierungsprozeß um eine Dimension der Konkretheit erweitert werden, die für das Verständnis komplexer räumlicher und funktionaler Zusammenhänge hilfreich ist. Das nach diesem Leitbild entwickelte Konzept bezeichnen wir als *Real Reality*.

2. Ausgangssituation

Die Ausgangssituation der hier dokumentierten Arbeiten ist im Zwischenbericht 1 und 2 ausführlich dokumentiert (artec paper 56).

Während im ersten Antragsjahr die Untersuchung der Machbarkeit durch einen funktionalen Prototypen im Vordergrund stand, wurde dieser im zweiten Projektjahr durch eine neue Client-Serverarchitektur und die Entwicklung von Netzwerkprotokoll- und Dateischnittstellen zu einem vielseitig einsatzfähigen System zur gegenständlichen Modellierung erweitert. Verschiedene Beispielszenarien von produktionstechnischen Szenarien wurden modelliert und ihr Verhalten in Simulationen überprüft.

Durch Vormachen konnten Verzweigungsregeln von Fördersystemen programmiert und die resultierenden Steueralgorithmen in einem Simulationsmodell übernommen werden. Verschiedene Datenformate und -strukturen zur Speicherung und zum Austausch von Steuerprogrammen wurden untersucht. Hierfür wurden Petri-Netze als die am besten geeignete Darstellung identifiziert. Anbindungen zu Industriesteuerungen wurden realisiert.

Neben dem Modellaufbau und der Programmierung des Verhaltens durch Vormachen wurden Objektrelationen zum Interaktionskonzept hinzugefügt. Eine automatisch erkannte „On Top“ Relation ermöglicht das Aufstapeln und Beladen von Paletten und Werkstückträgern. Diese werden im virtuellen wie im realen Modell gemeinsam bewegt.

Die Funktionalität der Software wurde um die Möglichkeit der Mehrhändigkeit erweitert, um synchrone Vorgänge mit mehreren Akteuren vormachen zu können.

Als zur Verwendung als gegenständliche Modellkomponenten für produktionstechnische Systeme wurden aus Fischertechnik Bausteinsätze in verschiedenen Detaillierungsgraden angefertigt. Der abstrakteste Bausteinsatz, im kleinsten Maßstab, dient zur Modellierung komplexer Szenarien, wie dem Beispiel eines flexiblen Montagesystems aus EUROSIM mit 8 Stationen und zugehöriger materialflußtechnischer Verknüpfung durch ein Fördersystem. Der detaillierteste Bausteinkasten wurde vollständig mit Sensoren und Aktoren ausgestattet und kann, über eine SPS gesteuert, Modellpaletten durch das System fördern.

Im Umfeld des Projekts wurden besonders interessante Anwendungsfälle im Rahmen von studentischen Projekten und Diplomarbeiten untersucht:

- Die Modellierung von Montagevorgängen aus vorgefertigten Modulen (Diplomarbeit),

- die gegenständliche Modellierung von Freiformflächen an Schaum- oder Knetmassenmodellen für CAD-Anwendungen (Promotions-Stipendium),
- Telemodellierung und Conferencing in Real Reality Umgebungen (Diplomarbeit),
- gegenständliche computerunterstützte Modellierung pneumatischer Schaltungen zur Ausbildung von Berufsschülern (das laufende europäische ESPRIT-Projekt „BREVIE“ wurde auf der Grundlage der Ergebnisse von RUGAMS initiiert).

(Eine Auflistung der Arbeiten befindet sich in Anhang C)

Die sich aus den Anwendungsfällen und Beispielszenarien ergebenden Anforderungen waren die Grundlage zur Entwicklung der Fragestellungen des Antrags für das 3. Projektjahr, dessen Ergebnisse in diesem Bericht vorgestellt werden.

Auch die bei der Präsentation der Konzepte und Ergebnisse auf Tagungen und die in verschiedenen anderen Foren geführten Diskussion mit Fachleuten aus Industrie und Forschung wurden bei den aktuell bearbeiteten Fragestellungen berücksichtigt.

3. Ausgangsfragen / Projektziele

In der Antragstellung für das dritte Projektjahr werden 16 Arbeitspakete beschrieben, die die Ausgangsfragen für diesen Bericht darstellen. Leitfragestellungen sind:

1. Welche Möglichkeiten zur Interaktivität bietet das *Real Reality* Konzept in der Produktionstechnik?
    - Gestische Interaktion mit dem Computersystem
    - Integration projizierter Informationen in das gegenständliche Modell
    - Beeinflussung dynamischen Modellverhaltens durch Gesten und durch Vormachen am Modell
2. Welche Möglichkeiten bieten *Real Reality* Modelle bei der verteilten Modellierung, bei der nicht alle Teilnehmer physikalischen Zugriff auf das Modell haben?
    - Integration weiterer Medien wie Videobilder, remote 3d-Visualisierung, Audio Konferenzen, ferngesteuerte Zeiger
    - Wie können Teilnehmer aktiv bzw. interaktiv an der Lösung eines Problems beteiligt werden?
3. Läßt sich ein von uns entwickelter Telemodellierer bei der gemeinsamen Bearbeitung einer Aufgabe mit einem entfernten Partner, dem WBK in Karlsruhe, unterstützend einsetzen?
    - Modellierung, Programmierung, Simulation und Überprüfung an einem realen SPS-gesteuerten Modell bei einer exemplarischen Aufgabenstellung
    - Zusammenführung der Projektergebnisse durch gemeinsame Datenschnittstellen

Die Beantwortung dieser Fragen und die Entwicklung weiterführender Fragestellungen, Konzepte und Anwendungsmöglichkeiten für *Real Reality*, von denen einige in dem vorliegenden Antrag beschrieben sind, stellen die Ergebnisse der Arbeit im dritten Projektjahr dar.

4. Veränderung des Erkenntnisstandes

Im Bereich der gegenständlichen Eingabe und Interaktion und bei der Telekooperation sind zahlreiche Weiterentwicklungen vorgenommen worden. Auch Fragestellungen, die in der An-

tragstellung nicht auftauchen, wurden in diesen Bericht aufgenommen, sofern sie zu konkreten Ergebnissen geführt haben und zu einer schlüssigen Darstellung beitragen.

4.1 Weiterentwicklung der Sensorisierung der Hand und der Modellierung des Greifvorgangs

Die Sensorisierung der Hand und die Software zur Abbildung des Greifvorgangs stellen das Zentrum der *Real Reality* Anwendung dar. Durch sie werden Interaktivität und Modellgenauigkeit am stärksten beeinflußt. Je detaillierter und filigraner ein Modell ist, um so höher sind die Anforderungen an die Präzision (MacKenzie 94).

4.1.1 Verbesserung der Sensoranordnung

Bei Interaktionen auf Objekten spielen die Fingerspitzen eine wichtige Rolle. Es konnte beobachtet werden, daß bei allen praktisch eingesetzten Griff- und Zeigegesten die Spitze des Zeigefingers ein Objekt berührt. Anfänglich wurde der Trackingsensor, so wie es allgemein üblich ist und von der Handschuhhardware vorgesehen ist, auf dem Handrücken befestigt. Hierbei wurde die Fingertip-Position über die kinematische Kette der Zeigefingerglieder errechnet (Abb 1 links). Da für die drei Gelenke eines Fingers in dem verwendeten Datenhandschuh nur ein Biegesensor zur Verfügung stand und damit die Fingerspreizung (Abdation) und Winkelverteilung auf die 3 Gelenke vernachlässigt wurden, ergab sich eine zu ungenaue Extrapolation der Fingerspitzenposition. Bei der Modellierung nach dem *Real Reality* Konzept spielt aber die Position der Berührungspunkte zwischen Hand und Objekt eine wichtige Rolle. Ein weiteres Problem war die Befestigung des Sensors auf dem Handrücken. Eine relativ zu den Fingerhandknochen unveränderte Position am Datenhandschuh konnte mangels geeigneter Befestigungspunkte nicht hergestellt werden. Es wurde deshalb die Anordnung des Sensors verändert (Abb 1 rechts).

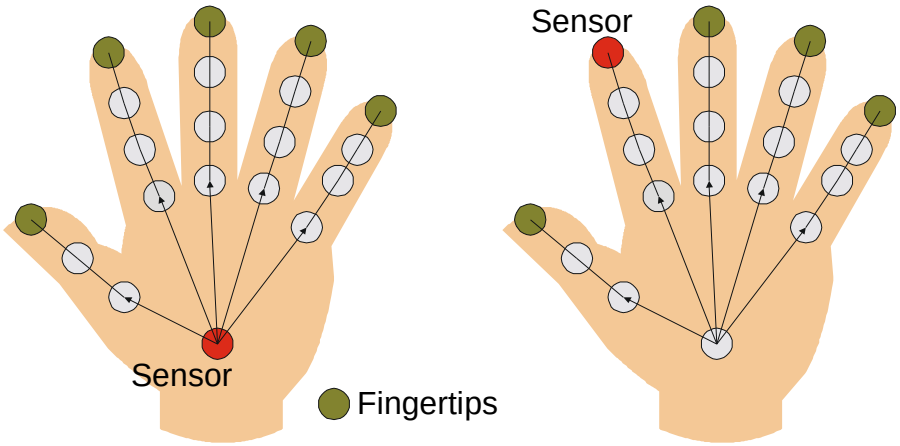

Abb 1: Veränderung der Sensoranordnung

Über einen *Grasptracker* (Abb 2), der als Ring auf die Spitze des Zeigefingers gesteckt wird konnte eine erhebliche Verbesserung der Grifferkennung erzielt werden. Beim Modellieren wird die flache Unterseite des *Grasptrackers* an das Objekt gepreßt, wodurch die relative Position des Sensors zum Objekt unverändert bleibt. Allein die Genauigkeit des Trackingsystems ist dann maßgeblich für das erstellte virtuelle Modell.

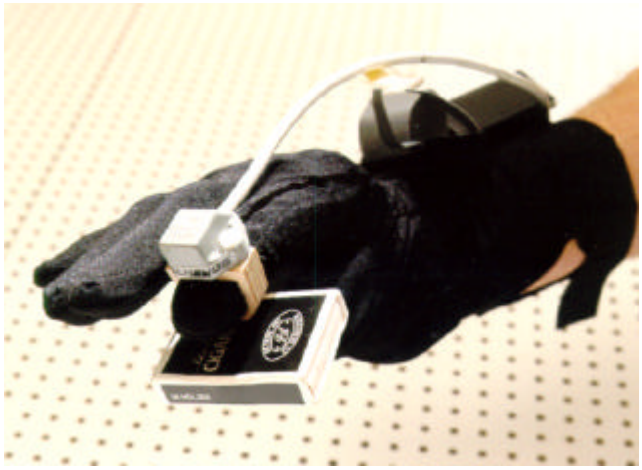

Abb 2: Grasptracker

Datenhandschuh und Trackingsystem haben bei diesem Modell eine sehr geringe Abhängigkeit voneinander. Der Handschuh dient neben der Visualisierung hauptsächlich zur Auslösung von Griff- und Gestenereignissen. Der *Grasptracker* wurde zusätzlich mit einem Drucksensor ausgestattet, so daß sich auch ohne Datenhandschuh Ereignisse auslösen lassen. Wird der Ring auf eine waagerechte Fläche gepreßt, wird dieses als Zeigegeste interpretiert, bei einer senkrechten Fläche als Griffgeste. Der Grenzwinkel kann in der Software eingestellt werden.

Die Modellierung mit kleineren und filigranen Objekten machte die Verbesserung der Grifferkennung erforderlich. Gerade das Modell aus EUROSIM (Krauth 91, 92, 95) stellt mit seinen 66 Förderbandobjekten und zahlreichen Paletten in einem Arbeitsbereich von weniger als einem Meter Länge (Reichweite des Trackingsystems) beim Modellaufbau und beim Vormachen von Anlagensteuerungen hohe Anforderung an die Grifferkennung.

4.1.2 Ausstattung eines Datenhandschuhs mit Drucksensoren

Wenn wir mit den Händen Gegenstände manipulieren, spielt das Feedback durch Druckwahrnehmung eine wichtige Rolle. Daß das Fehlen dieses Feedbacks eine schwerwiegende Einschränkung der Interaktion ist, wurde bei Untersuchungen im Bereich Virtual Reality festgestellt, bei denen dieses Feedback fehlt. Präzises Positionieren von Objekten ist unter diesen Voraussetzungen außerordentlich schwierig. Die Annahme liegt nahe, daß auch für die synchrone gegenständliche und virtuelle Modellierung der Druck auf die Fingerspitzen eine wichtige Rolle spielt, und hilfreich für die Erkennung und Verarbeitung von Ereignissen in der über Sensoren beobachteten realen Welt sein kann. Um die Vorteile, die sich hieraus ergeben, genauer zu untersuchen, wurde ein Datenhandschuh mit Drucksensoren angefertigt und an die bestehende Real Reality Software angeschlossen.

Ein einfacher Stoffhandschuh wurde mit Sensoren und Meßelektronik ausgestattet und zu einem voll funktionsfähigen prototypischen Datenhandschuh ausgebaut. Für jeden Finger werden zwei Meßwerte erfaßt, die Fingerkrümmung über einen Biegemeßstreifen und der Anpreßdruck über einen Drucksensor. Die Druck- und Biegesensoren liefern veränderte Leitwerte die über einen A/D-Wandler mit 8 Eingangskanälen und 10 Bit Auflösung (1024 verschieden Werte) gemessen und über eine serielle Schnittstelle an den Computer übertragen werden. Die acht Eingangskanäle des Wandlers erlauben die Sensorisierung von vier Fingern, der Kleine Finger blieb unsensorisiert (Abb 3).

Abb 3: Mit Drucksensoren ausgestatteter Datenhandschuh

Die Implementation eines Treibers für den neuen Handschuh wurde in C++ und Assembler durchgeführt. Über diesen konnte der Handschuh in die bestehende Software eingebunden und getestet werden.

4.1.3 Erweiterung der Gestenerkennung mit Druckparametern

Die verwendete Gestenerkennung verwendet ein statistisches Verfahren zur Gestenerkennung, die Multivariatenanalyse (Brauer 94). Die Eingangsdaten von Sensoren werden als Featurevektor aufgefaßt, der mit vorgegebenen Beispielvektorsätzen verglichen wird. Bei einer Ausreichenden Übereinstimmung mit einem Beispielvektorsatz wird dieses Feature als erkannt gemeldet. Das Verfahren, welches bisher die Erkennung von Griffposen aufgrund der 5 Krümmungswerte der Finger durchgeführt hat, läßt sich unverändert auf die Erkennung von Sensordaten, die sich aus 4 Krümmungswerten und 4 Druckwerten zusammensetzen, anwenden. Es wurden neue Beispielvektorsätze vorgemacht, die neben charakteristischen Gesten, auch Griffe, mit hierfür typischen Druckwerten an den Fingerspitzen enthalten.

4.1.4 Erfahrungen mit der drucksensorgestützten Grifferkennung

Zunächst stellen sich die Drucksensoren an den Fingerspitzen als Nachteil dar, da sie durch ihre Steifheit den Tastsinn der Benutzers behindern. Die Sensoren sind zwar flexibel, verhalten sich bei der Handhabung aber doch eher wie Plättchen. Dadurch, daß diese nicht immer gleichmäßig auf der gesamten Fläche auf das Objekt gepreßt werden, ergeben sich außerdem starke Streuungen in den Meßwerten für die Andruckkraft. Um diese stark streuenden Featurevektoren mit der verwendeten statistischen Gestenerkennung zu erkennen, ist die Vorgabe von vielen Beispielvektoren nötig, die diese Streuungsbreite abdecken müssen.

In Experimenten mit entsprechenden Beispielvektorsätzen war eine sichere Grifferkennung möglich. Dabei stellte sich als entscheidender Vorteil heraus, daß besonders das Lösen von Griffen erheblich schneller erkannt wird. Bereits beim Nachlassen der Griffstärke wird das Objekt gelöst. Diese Eigenschaft ist wichtig, weil beim natürlichen Loslaßvorgang die Hand häufig schon weggezogen wird, bevor die Fingerstellung soweit verändert wurde, daß dieses als Verlassen des Griffmusters erkannt werden kann. In der rein gestenbasierten Grifferkennung führt das leicht zu einem Verwackelungseffekt, der die Modellgenauigkeit negativ beeinflußt. Die Modellierenden haben sich deshalb eine etwas unnatürliche Vorgehensweise beim Modellieren angewöhnt, bei der die Hand nach dem Öffnen des Griffes noch einen Moment in der Position verweilt, bis durch das akustische Feedback das Loslassen signalisiert wird.

Die Ausstattung der Datenhandschuhe mit Drucksensoren kann als eindeutige Empfehlung ausgesprochen werden, weil hierdurch die Genauigkeit beim Modellieren in *Real Reality* erhöht wird und eine natürlichere Interaktion gefördert wird. Interessant wären Experimente mit professio-

nellen mit Drucksensoren ausgestatteten Datenhandschuhen wie z.B. dem TUB-Glove der TU-Berlin (Hoffmann 95).

4.1.5 Grifferkennung durch Kollisionscheck

Die Modellierung mit kleineren und filigranen Objekten machte die Verbesserung der Grifferkennung erforderlich. Gerade das Modell aus EUROSIM (Krauth 91) stellt mit seinen 66 Förderbandobjekten und zahlreichen Paletten in einem Arbeitsbereich von weniger als einem Meter Länge beim Modellaufbau und beim Vormachen von Anlagensteuerungen hohe Anforderung an die Grifferkennung.

Ein Griffereignis findet statt, wenn eine Griffpose vorliegt, die durch die Gestenerkennung identifiziert wird und sich ein passendes greifbares Objekt innerhalb dieser Griffpose befindet. Die Beobachtung, daß bei allen relevanten Griffen der Zeigefinger der Hand das Objekt berührt, wurde bereits oben erwähnt. Dieser Beobachtung wurde durch die Verlagerung des Positionssensors an die Zeigefingerspitze Rechnung getragen. Bei dieser Konfiguration kann ein Greifereignis durch die Kollision der Fingerspitze mit der virtuellen Repräsentation eines greifbaren Objekts beim Eintreten einer Griffgeste erkannt werden.

Es wird ein hierarchisches Verfahren zur schnellen Kollisionserkennung verwendet (Merkler 96). Durch eine wenig rechenaufwendige Kontrolle auf die Befindlichkeit eines Punktes in der Umgebungskugel eines Objekts (Sphären Check) können bereits die meisten Objekte von einer detaillierteren rechenintensiveren Kontrolle ausgeschlossen werden. Nur wenn dieser Test positiv ausfällt, wird das Objekt einer detaillierteren Kontrolle an der umgebenden Box (Boundary Check) unterzogen. Wenn auch dieses Ergebnis positiv ist, wird ein Griffereignis ausgelöst. Dieses zweistufige Verfahren erfüllt die Echtzeitanforderungen der Grifferkennung. Für die bei Merkler vorgeschlagene aufwendigere vierstufige Kollisionserkennung aus umgebender Kugel, umgebendem Zylinder, umgebender Box und umgebenden Boxen der Teilobjekte ergibt sich aus unseren Anforderungen zunächst kein Bedarf.

Der Einsatz einer Kollisionserkennung stellt die Voraussetzung für die Interaktion mit nicht sensorisierten Schaltern und Tastaturen dar, die im folgenden Abschnitt beschrieben wird.

4.1.6 Ergebnisse

Die minimale handhabbare Objektgröße liegt jetzt 30 x 30 x 10 mm. Mit dieser dürfte die minimale, mit einem Datenhandschuh sicher handhabbare Größe erreicht sein. Damit steht eine geeignete Lösung zur Interaktion für die weitere Arbeit mit *Real Reality* Anwendungen zur Verfügung. Die dargestellten Verbesserungen im Kernfeld von *Real Reality*, der Grifferkennung und Sensorisierung der Hand, leisten einen Beitrag zum sicheren und präzisen Umgang mit den *Twin-Objects*, die aus realem und virtellem Objektpart bestehen. Mit ihnen wurden die Voraussetzungen für die Arbeit an *Real Reality* Projekten mit detaillierten Objekten geschaffen.

4.2 Spezifikation der Interaktionen durch eine Gesten-Interaktionsgrammatik

Bei der gestenbasierten Interaktion mit einer gegenständlichen Benutzeroberfläche sind verschiedene Interaktionen möglich, die in einer eindeutigen Form zu beschreiben sind. Hierfür bot sich die Entwicklung einer Gesten-Interaktionsgrammatik an. Diese Form kann als Spezifikation für die Implementierung der einzelnen Interaktionen eingesetzt werden.

Zunächst sollen hier die Interaktionen beschrieben werden. Zunächst jene, die unabhängig von der Bedeutung der Modellbausteine sind. Auf dieser Ebene spielt es daher noch keine Rolle, ob mit Förderbändern, Pneumatikelementen, Elektronikbauteilen oder anderen Objekten modelliert wird. Mit der Interpretation der Modelle wird dann in Abschnitt *Vormachen von Steuerungsprogrammen* die kontextabhängige Weiterverarbeitung der Modellinformationen vorgestellt.

4.2.1 Interaktionen in der Real Reality Arbeitsumgebung

In diesem 3. Projektjahr von RUGAMS wurden verschiedene Arbeitspakete bearbeitet, die in Zusammenhang mit neuen kontextunabhängigen Interaktionen stehen:

1. Bedienung eines Simulators; allgemein die Bedienung von Anwendungssoftware (AP 1.1)
2. Virtual Touch Screen (AP 1.2)
3. Gegenständliche Dummy-Tastatur (Free Keyboard) (AP 1.3)
4. Relative Parameteränderungen (AP 1.4)
5. Aufruf von verschiedenen Hilfe- und Informationsfunktionen (Sprechende Modelle) (AP 1.6)

Zunächst wurden die Interaktionen, die bisher eher spontan und unter pragmatischen Aspekten für die Modellierung eingesetzt wurden systematisiert und um die oben genannten Interaktionen ergänzt, zu einem Gesamtkonzept vereinigt und in der *Real Reality* Software implementiert.

4.2.2 Modellieren

Das *Real Reality* Konzept basiert auf dem synchronen Aufbau realer und virtueller Modelle, also der Manipulation von *Twin Objects*. Eine Manipulation beginnt mit dem *Greifen* eines Objekts, darauf folgt eine *Bewegungsphase* und endet mit dem lösen des Griffs und der *Positionierung*. Die Objektbox hat bei der Modellierung eine besondere Bedeutung. Im 2. Projektjahr wurde ein Konzept entwickelt, wie durch eine Objektbox beliebig viele Instanzen einer Klasse im Modell verwendet werden können. In der Objektbox führt die Griffaktion sofort dazu, daß ein neues Objekt in die Box eingefügt wird. Objekte die sich in der Box befinden, sind hierfür mit dem speziellen Attribut „IsInBox“ ausgestattet, das beim ersten Greifen gelöscht wird.

4.2.2.1 Selektieren

In der Modellszene können mehrere Objekte selektiert werden. Diese erscheinen dann mit einem blinkenden Rahmen in der Visualisierung. Auf den selektierten Objekten können verschiedenen Aktionen wie z.B. Löschen oder Parameteränderung ausgeführt werden. Die Selektierung hat auch für die Kommunikation zwischen den Akteuren eine Bedeutung. Speziell bei der verteilten Modellierung, bei der den Teilnehmern individuelle Selektionsfarben zur Verfügung stehen, kann auf diese Weise deutlich gemacht werden, auf welche Objekte sich Äußerungen oder andere Informationen beziehen. Durch einfaches *Berühren* mit einer *Zeigegeste* wird ein Objekt selektiert, durch erneutes Antippen deselektiert.

4.2.2.2 Objektspezifische Kommandos

Bei der Interaktion in einem Modell können eine Vielzahl verschiedener Eingaben erforderlich sein. Es zeigte sich, daß es keine ausreichende Zahl wohlunterscheidbarer Gesten gibt, die diese Befehle angemessen, das heißt selbsterklärend und intuitiv, repräsentieren. In der Szene können statt dessen beliebige *Kommandoobjekte* eingesetzt werden. Wenn diese mit einer Zeigegeste berührt werden, findet keine Selektion statt, sondern es wird ein objektspezifisches Kommandoereignis ausgelöst. Diese Form der Interaktion ist sehr vielfältig verwendbar. Verschiedene Arbeitspakete in diesem Antragsjahr können mit Kommandoobjekten bearbeitet werden:

- Aufruf von Hilfe

  Im Modell steht ein spezielles *Hilfeklötzchen* zur Verfügung, das mit einem Fragezeichen gekennzeichnet ist (Abb 4). Dieses kann auf ein beliebiges Objekt gelegt werden. Durch Antippen des Klötzchens wird die Hilfe des Objekts, auf dem das Hilfeklötzchen liegt, ausgegeben.

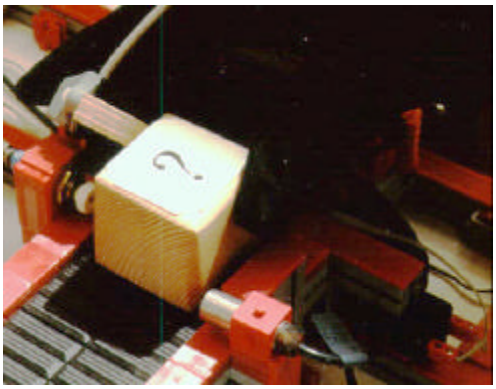

Abb 4: Hilfeklötzchen

- Sprechende Modelle

  Dadurch, daß einem Gegenstand eine Hilfe in Form einer Sprachausgabe als Attribut gegeben wird, lassen sich auch die „Sprechenden Modelle“ realisieren.

- Free Keyboard

  Die kabellose Tastatur für Real Reality, das „Free Keyboard“, ist eine Anwendung, bei der die Tasten durch Kommandoobjekte dargestellt werden (Abb 5). Das Berühren der Tasten löst entsprechende Ereignisse aus, z.B. das Ausgeben des auf der Taste gezeigten Zeichens an der Stelle eines aktiven Cursors.

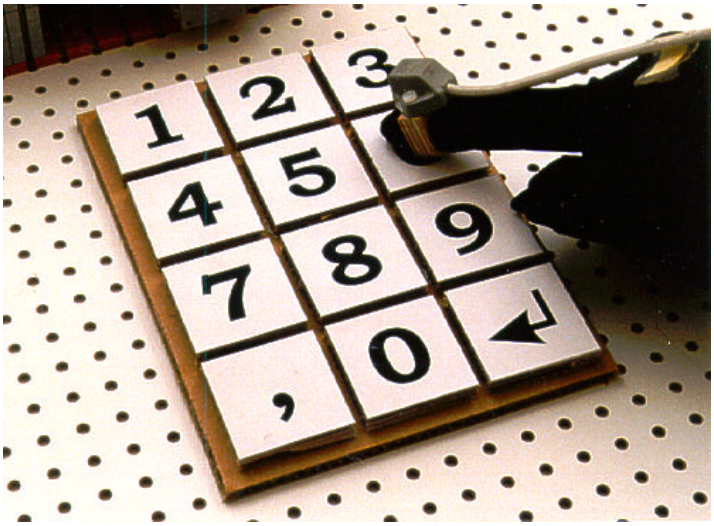

Abb 5: „Free Keyboard“ zur Eingabe von Realzahlen

- Relative Änderung von Parametern

Die Bearbeitung von Simulationsmodellen erfordert das häufige Verändern von Parametern, beispielsweise für Bearbeitungszeiten, Puffergrößen oder Simulationsgeschwindigkeit. Um die Ausdrucksfähigkeit der Hand zur Parameterveränderung zu benutzen, wurde ein Prototyp angefertigt, der das Prinzip des „Shuttle Jogs“ oder „Jog Dials“ aufgreift, welches von Videorecordern und anderen technischen Geräten bekannt ist: Eine beliebige Stellung eines Wählrades wird mit dem aktuellen Wert eines Parameters verknüpft. Durch Drehung des Wählrades kann der gesamte Wertebereich überstrichen werden. Dabei wird der Wertebereich je nach Genauigkeitsanforderungen auf eine sinnvolle Anzahl von Umdrehungen skaliert. Die Anfangsstellung des Wählrades ist irrelevant, da alle Wertänderungen relativ zum Anfangswert erfolgen.

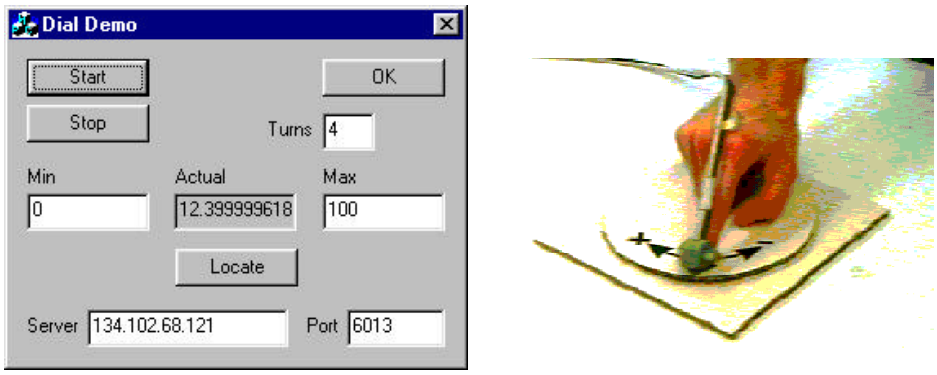

Abb 6: Relative Parameteränderung mithilfe einer Wählscheibe

Die Wählscheibe des Real Reality Prototyps arbeitet ohne Verdrahtung (Abb 6). Der Drehwinkel wird durch Vermessung der Mittelachse (Locate) und einen Kollisions-Check mit der Hohl-Zylinderbahn, den die Griffmulde beim der Rotation beschreibt, mithilfe des Trakkingsystems bestimmt. Die zugehörige Software erlaubt die Eingabe des Wertebereichs, der Umdrehungszahl zum Überstreichen desselben und eines Anfangswertes unter „Actual". Voraussetzung für die reale Verwendung der Dial Applikation ist die Offenheit des Zielsystems, das die Parameter empfängt. Von diesem muß der Wertebereich und der aktuelle Wert bezogen werden können. Ist dieses nicht möglich, können die Eingaben mit dem „Free Keyboard" (s.o.) durchgeführt werden.

Das Dial Demo veranschaulicht die Flexibilität von Real Reality bei der Gestaltung von Eingabemedien und ihre kostengünstige Realisierung mit einfachen Materialien.

- Kalibrierungsklotz

  Wenn mit projizierten Objekten interagiert werden soll, wie z.B. beim Virtual Touchscreen, muß die Projektionsfläche vermessen werden, damit reale und projizierte Position zur Dekkung kommen.

  Auch wenn neue Objekte in die Szene eingebracht werden, die nicht der Objektbox entnommen werden, muß ihre Anfangsposition im Modell werden. Das kann durch Auflegen und Antippen des Kalibrierungsklotzes geschehen. Je nach Anforderung können verschiedene Typen von Klötzen verwendet werden. Vorstellbar sind verschiedene Symbole für die Selektion von Ecken oder Mittelpunkten (Abb 7).

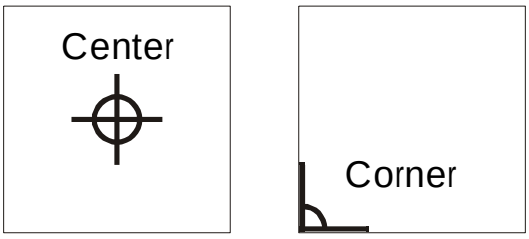

Abb 7: Kalibrierungsklötze zum Vermessen von Mittelpunkt und Ecke

Um Kommandos auf Objekten zu erkennen ist es nicht erforderlich, die Bedeutung des Kommandos zu kennen. Das Objektattribut „IsACommand" legt lediglich fest, daß beim Berühren mit einer Zeigegeste ein Kommando ausgeführt werden soll. Die Interpretation des Kommandos bleibt speziell entwickelter Software überlassen, die die Ausführung des Kommandos übernimmt.

4.2.2.3 Objektneutrale Kommandos

Die Interpretation von Handgesten spielt eine wichtige Rolle bei der Kommunikation, weil Gesten wichtige Bedeutungsinhalte übertragen. Es werden deshalb auch Möglichkeiten vorgesehen, in einer *Real Reality* Umgebung die Aussage von Gesten zu interpretieren, um hiermit Funktionen z.B. des Simulators zu steuern. Je nach Software sind verschiedene Befehle von Bedeutung. Bei Simulationsanwendungen sind Start und Stop wichtige Kommandos. Durch das Formen einer Faust können diese ausgelöst werden.

4.2.2.4 Virtual Object spezifische Kommandos

Die gegenständlichen Modelle dienen der Spezifikation von Situationen und Zuständen, die vom Rechner analysiert und variiert werden sollen. Hierfür steht eine große Vielfalt von Hilfsprogrammen wie z.B. Simulatoren, Dateiverwaltungsprogrammen, statistischen Analyseprogrammen zur Verfügung, die in der Regel nicht an die Eingabe mit einem Datenhandschuh angepaßt werden können. Den Benutzern von *Real Reality* soll die Möglichkeit gegeben werden, desktoporientierte Programme mit dem Handschuh zu bedienen. Hierfür kann eine Schnittstelle genutzt werden, die von den Programmen bereitgestellt wird, die Maus. Die wichtigsten Ereignisse die von einer Maus ausgehen sind linke und rechte Maustaste drücken, Maustaste loslassen und linker Doppelklick. Bei diesen Ereignissen spielt die aktuelle Position der Maus eine Rolle, bei gedrückter Taste auch die Bewegung, die ausgeführt wird.

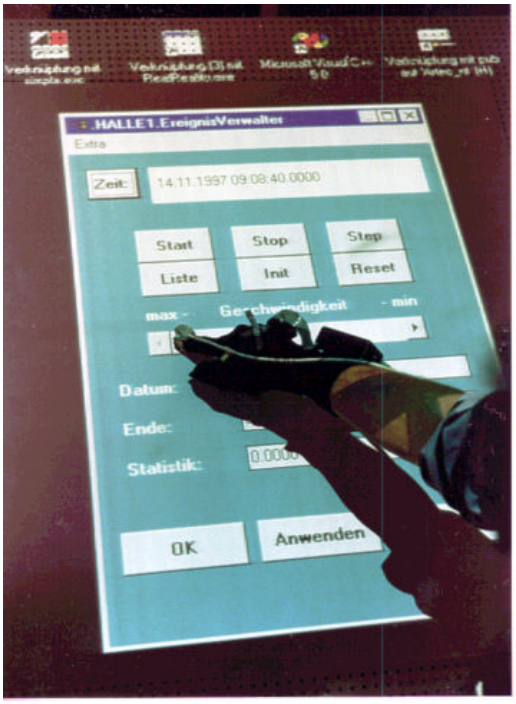

Bild 1: Interaktion mit einem projizierten virtuellen Windows Objekt auf der waagerecht angeordneten Modelltischplatte

Auf die Tischfläche projizierte Dialogelemente, wie Slider, Ikonen, Schalter etc. werden beim „Virtual Touchscreen“ mit dem Datenhandschuh bedient. Der Windows Desktop wird auf eine senkrechte oder waagerechte Fläche neben dem Modell oder direkt auf das Modell projiziert. Mit dem Trackingsystem wird die Ausdehnung der Desktopfläche vermessen, so daß der Mauszeiger innerhalb dieser Fläche mit der Fingerspitze des Zeigefingers zur Deckung gebracht und nachgeführt werden kann. Mit spezifischen Gesten lassen sich jetzt die Mausereignisse auslösen. Zeige- bzw. Mittelfinger für rechten oder linken Mausklick, Zeige- und Mittelfinger zusammen für einen linken Doppelklick. Auf diese Weise lassen sich eine Vielzahl von Windows-Programmen bedie-

nen. An den Stellen, an denen Tastatureingaben unumgänglich oder sinnvoll sind, kann das „Free Keyboard“ zum Einsatz gebracht werden, um z.B. Parameter in Eingabefeldern zu editieren.

Durch Vermessen des projizierten Desktops mit dem Trackingsystem oder mit Kalibrierungsklötzchen an zwei Eckpunkten wird ein virtuelles Objekt im virtuellen Modell erzeugt, das aber beim Rendern nicht dargestellt wird. Dieses hat die Form eines Quaders, dessen Oberseite der Projektionsfläche entspricht. Über eine Kollisionserkennung können die Maustastendrücke auch ohne einen Drucksensor an den Fingerspitzen des Datenhandschuhs erzeugt werden. Die relative Position der Fingerspitze zum virtuellen Desktop-Objekt gibt Auskunft über die Position des Mauszeigers auf dem Desktop.

Um die Interaktionen bei der Berührung des „Virtual Touchscreen“ richtig zu interpretieren, hat dieses Objekt das spezielle Attribut „IsDesktop“.

4.2.3 Spezifikation der Gesten- Interaktionsgrammatik

Die vorigen Abschnitt beschriebenen Konzepte wurden in eine formale Spezifikation gefaßt, um die Implementierung durchzuführen. Der Vergleich verschiedener Darstellungsformen in (Stary 94) hat die besondere Eignung des Ansatzes von (Reisner 81) für unsere Zwecke gezeigt. Dieser Ansatz zur Beschreibung von Human-Computer Interaction beruht auf kontextfreien Grammatiken. Durch eine Backus-Naur Form können diese formal, aber auch lesbar und nachvollziehbar beschrieben werden. Die terminalen Symbole werden durch Ereignisse die vom Benutzer mit dem Datenhandschuh oder durch den Computer ausgelöst werden, durch Abfrageergebnisse aus der Modelldatenbank oder durch das Senden von Ereignissen beschrieben. In Tabelle 1 sind alle terminalen Symbole nach Kategorien aufgeführt.

User Action	**ROMAN[1] Command**	**ROMAN Question**	**ROMAN Raise Event**	**DGC[2] Command**	**DGC Question**
<PointGesture>	<StartRecordPath>	<Collision>	<EvHitCommandObject>	<PlaySelectSound>	<ObjectIsNotSelected>
<DoublePointGesture>	<StopRecordPath>	<NoCollision>	<EvSetMousePos>	<PlayDeselectSound>	
<GraspGesture>	<SetObjectPos>	<IsACommand>	<EvLMButtonDown>	<PlayGraspSound>	
<CommandGesture>	<CreateObject>	<ObjectInBox>	<EvLMButtonUp>	<PlayReleaseSound>	
<NoGesture>		<IsDesktop>	<EvLMDoubleKlick>	<PlayMouseKlickSound>	
			<EvGraspObject>	<SelectObjectCommand>	
			<EvReleaseObject>	<DeselectObjectCommand>	
			<EvSelect>		
			<EvGestureStart>		
			<EvGestureStop>		
			<EvCommand>		
			(Ev = Event) (LM = left mouse button)		

Tabelle 1: Terminale Symbole der Gesten- Interaktionsgrammatik

Komplexe Interaktionen setzen sich aus diesen Interaktionen zusammen (Tabelle 2). Ausgangspunkt sind immer die Intentionen der Benutzer. Alle nichtterminalen Symbole sind deshalb vom Benutzer gesteuert und werden durch „Produktionen“ hierarchisch weiter spezifiziert. Alle Interaktionen sind schließlich auf die terminalen Symbole aus Tabelle 1 zurückzuführen.

Das oberste Symbol „Session“ ist das Einstiegssymbol. Geschweifte Klammern schließen eine Liste ein und zeigen an, daß unendlich oft versucht wird, die Grammatik anzuwenden, bis das

[1] **R**eal **O**bject **Man**ager: Dieses ist die Software, die das virtuelle Modell verwaltet und über Schnittstellen (s.u.) den Zugriff auf dieses ermöglicht.

[2] **D**aten und **G**esten **C**lient: Hierbei handelt es sich um die Software, die handbasierte Interaktionen auf der Grundlage der vorgestellten Grammatik verwaltet und daraufhin die Modelldatenbasis im ROMAN aktualisiert.

Programm vom Benutzer beendet wird. Von eckigen Klammern eingeschlossene Symbole sind optional.

Bei der Implementierung wurde die Grammatik in C++ Code übertragen. Für jedes Symbol wurde eine gleichnamige Methode definiert. Ein Thread[3] versucht immer wieder, die Methode „Session“ auszuführen. So werden die Aktionen des Benutzers quasikontinuierlich überwacht und die Interaktionen, die der Grammatik entsprechen, erkannt und interpretiert.

Session	→	{ TipInteraction | GraspInteraction | DoublePointInteraction | CommandInteraction }
TipInteraction	→	<PointGesture> <Collision> (DesktopInteraction | ((HitCommandObject | SelectObject | DeselectObject) (<NoGesture> | <NoCollision>)))
GraspInteraction	→	<GraspGesture> <Collision> [<ObjectInBox> <CreateObject>] <StartRecordPath> <EvGraspObject> <PlayGraspSound> (<NoGesture> | {<SetObjectPos>}) <StopRecordPath> <EvReleaseObject> <PlayReleaseSound>
DoublePointInteraction	→	<DoublePointGesture> <Collision> <IsDesktop> <EvLMDoubleKlick> <PlayMouseKlickSound> <PlayMouse- KlickSound> (<NoGesture> | <NoCollision>)
CommandInteraction	→	<CommandGesture> <EvCommand> <NoGesture>
DesktopInteraction	→	<IsDesktop> <EvSetMousePos> <EvMLButtonDown> <PlayMouseKlickSound> (<NoGesture> | <NoCollision> | {<EvSetMousePos>}) <EvLMButtonUp> <PlayMouseKlickSound>
HitCommandObject	→	<IsACommand> <EvHitCommandObject> <HitCommand- Sound>
SelectObject	→	<ObjectIsNotSelected> <SelectObjectCommand> <EvSelect> <PlaySelectSound>
DeselectObject	→	<DeselectObjectCommand> <EvSelect> <PlayDeselectSo- und>

Tabelle 2: Kontextfreie Grammatik zur Spezifikation von Interaktionen am realen Modell (nichtterminale Symbole)

4.2.4 Erfahrungen mit der formalen Spezifikation

Durch die *Gesten-Interaktionsgrammatik* wurde ein Werkzeug zur formalen Spezifikation von synchronen Interaktionen an realen und virtuellen Modellen geschaffen. Dieses hat bei der Entwicklung der Interaktionen, wie sie in den Arbeitspaketen beschrieben sind und in Diskussionen über die Eignung verschiedener Interaktionskonzepte wertvolle Dienste geleistet. Die „gute“

[3] ein asynchroner zyklischer Prozeß unter Windows

Lesbarkeit der Grammatiken spielt dabei eine wichtige Rolle. Auf der anderen Seite leisten sie als Sezifikationssprache die Grundlage für die Implementierung der Software und führen damit zu

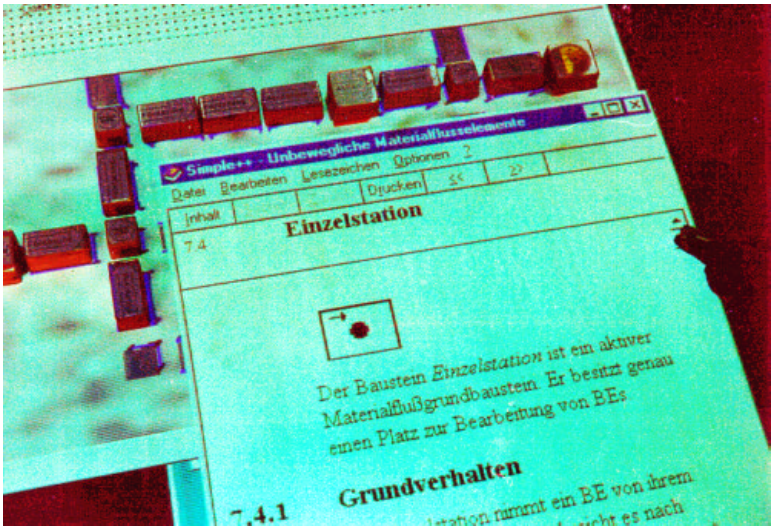

Bild 2: Einblendung von Hilfeinformation durch „Augmentation“ eines auf dem Tisch aufgebauten Modells

einer Verkürzung der Entwicklungszeiten. Die Grammatik wird in einer EBNF-Form[4] spezifiziert. Aufgrund der positiven Erfahrungen werden Gesten-Interaktionsgrammatiken in der weiteren Arbeit an *Real Reality* Projekten eingesetzt.

4.3 Integriertes Modellieren und Simulieren mit Augmented Real Reality

Aufbauend auf den Erfahrungen, die durch das Experimentieren mit Bildprojektionen gewonnen wurden, soll eine weitgehende Integration der Modellierung (Modellaufbau und Variation) und der Simulation (dynamischer Ablauf, Ergebnisdarstellung) erreicht werden. Dies wird u.a. durch Einblendung visueller Informationen in die gegenständliche Modellierungs-Umgebung erreicht. Es wird nicht nur eine Darstellung von grafischen Objekten angestrebt, sondern auch die Interaktion mit den eingeblendeten Bildern, unter Verwendung der bereits implementierten Algorithmen zur Griff- und Gestenerkennung. Interaktionen mit realitätsnahen Modellen spielen eine wichtige Rolle, speziell, wenn fachfremde Akteure an den Planungen beteiligt sind und eventuell vorhandene Mängel am System frühzeitig erkannt und behoben werden sollen (Drews 97).

4.3.1 Modellerweiterung durch Einblendung von Informationen

Bei der Bearbeitung einer Planungsaufgabe liegt das Ergebnis in Form des gegenständlichen Modells auf dem Modelltisch vor. In der nun folgenden Phase wird mit Hilfe des Computers das im Hintergrund erstellte virtuelle Abbild des Modells analysiert. Bei dem im Projekt RUGAMS als Applikation gewählten Fördersystemen spielt die Simulation des dynamischen Verhaltens eine wichtige Rolle. Idealerweise soll das Verhalten der Komponenten aus der Simulation in engem Zusammenhang mit dem gegenständlichen Modell präsentiert werden, um den hierdurch erzielten Erkenntnisgewinn direkt am realen Modell zu diskutieren und gegebenenfalls weitere Spezifikationen und Veränderungen am Szenario vorzunehmen. Durch die Vermeidung eines Bruchs zwischen gegenständlicher Eingabe und Feedback des Computersystems werden die Planer bei der Arbeit unterstützt, anstatt durch einen Wechsel der Präsentationsmodalität von der Aufgabe

[4] Erweiterte Bbackus-Naur Form (vgl. Stary 94). Diese Form läßt sich zur besseren Übersicht auch als Diagramm darstellen.

abgelenkt zu werden. Diese Ziel wird erreicht, indem das gegenständliche Modell um die vom Rechner gewonnenen Informationen vergrößert (augmented), bzw. erweitert wird. Die Technik der Augmented Reality stellt Methoden vor, wie durch Einblendungen über Spiegel im Zusammenhang mit Head mounted Devices (HMD) oder Projektionen, die natürliche Umwelt um Informationen erweitert wird (vergl. Streitz 98, Rekimoto 96). Wird das gegenständliche Modell in diese Erweiterungen einbezogen, bzw. als Grundlage hierfür verwendet, sprechen wir von *Augmented Real Reality*.

Die Arbeit beim Modellieren findet an einem feststehenden Tisch statt. Dieser Tisch läßt sich gut als Projektionsfläche nutzen. Mit einem Datenprojektor (Beamer) können die Informationen direkt ins Modell eingeblendet werden. Die Modellierenden benötigen keine zusätzlichen Geräte wie Shutter Glasses oder HMDs, um diese Einblendungen zu sehen. Die oben beschriebenen Anforderungen, daß die Informationen ohne einen Medienbruch wahrgenommen werden können, wird durch diese Anordnung erfüllt.

Zur technischen Realisierung dieser *Augmented Real Reality* Konfiguration kann das Bild des waagerecht strahlenden Beamers über einen Spiegel auf das Modell projiziert werden. Durch Hochklappen des Spiegels kann alternativ auch eine Vertikalprojektion genutzt werden (Abb 8).

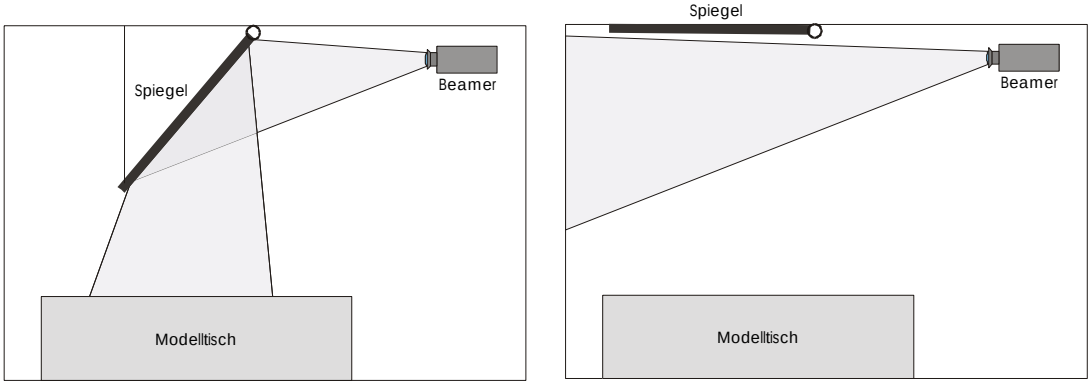

Abb 8: Anordnung von *Augmented Real Reality* zur Einblendung von Informationen ins Modell (links), Vertikalprojektion durch Hochklappen des Spiegels (rechts)

4.3.1.1 Geräteauswahl

Zur Realisierung wurde ein geeigneter Beamer ausgesucht. Zuvor durchgeführte Experimente mit einem lichtschwächeren Gerät (siehe RUGAMS Arbeitsbericht Nr. 2) hatten gezeigt, daß die Lichtleistung ein wichtiges Auswahlkriterium ist. Durch den Vergleich verschiedener Geräte unter Tageslichtbedingungen konnte eine Mindestleistung von 600 ANSI-Lumen ermittelt werden. Je nach Umgebungsbedingungen und Reflexivität der Projektionsfläche ist eine möglichst hohe Leistung von Nutzen. Ein anderer wichtiger Aspekt ist die Auflösung des Displays. Diese sollte möglichst hoch sein, um bei der Realisierung des Virtual Touchscreens eine sinnvolle Darstellung des Desktops zu ermöglichen und um eine ausreichend detaillierte Abbildung von Projizierten Modellkomponenten zu gewährleisten. Die Ausstattung mit einem Zoom-Objektiv und die Option der Rückprojektion zur Anzeige eines Seitenverkehrten Bildes für die Projektion über einen Spiegel sind weitere geforderte Ausstattungsmerkmale. Ausgewählt wurde schließlich ein Gerät von Polaroid, der Poaview 211E, der die Anforderungen innerhalb des finanziellen Rahmens optimal erfüllt. Dieser verfügt über eine Leuchtkraft von 650 ANSI-Lumen.

4.3.1.2 Interaktion mit projizierten Modellelementen

Im Abschnitt *1 Interaktionen in der Real Reality Arbeitsumgebung* wurden bereits verschiedene Aspekte der Interaktion mit projizierten Informationen dargestellt. Darüber hinaus bieten sich weitere Möglichkeiten, mit projizierten Elementen zu interagieren.

Bewegliche Elemente wie Paletten oder Werkstücke, die von der Simulation erzeugt werden, sind im virtuellen Modell vorhanden. Diese können an der Kollisionserkennung ebenso teilnehmen, wie die realen Modellgegenstände auf dem Tisch. Das bedeutet, daß sie auch gegriffen und selektiert werden können. Durch die Selektion einer oder mehrerer Paletten, die dann in einer blinkenden Boundarybox dargestellt werden, kann ihre Bewegung durch das System beobachtet werden. Durch die Verwendung des Hilfeklötzchens (s.o.) können weitere Informationen über das projizierte Objekt abgerufen werden, wie z.B. Bearbeitungszustand, Gesamtzeit im System oder Bearbeitungsdauer.

4.3.1.3 Rekonstruktion von virtuellen Modellen

Nicht alle Planungsaufgaben können in einer Sitzung abgeschlossen werden. Häufig soll das Modell in verschiedenen Projektphasen herangezogen werden. Um den Arbeitstisch und die Modellkomponenten anderweitig nutzen zu können, ist eine Möglichkeit zur Rekonstruktion früherer Modelle wünschenswert. Hierfür bietet eine kalibrierte Projektion einen eleganten Weg. Die virtuellen Abbilder der Szenarien lassen sich als Datei speichern und archivieren. Später kann das Modell erneut geladen und auf den Tisch projiziert werden. Anhand der Projektion lassen sich die Komponentenanordnungen genau nachstellen und für Veränderungen des Modells, Diskussionen und Präsentationen nutzen. Es ist dabei nicht immer notwendig, das gesamte Modell nachzustellen, häufig reicht das Aufbauen im Fokus der Planung stehender Elemente aus.

Für die genaue Rekonstruktion spielt eine maßhaltige Projektion eine wichtige Rolle. In Abschnitt *4.3.2.2 Kalibrierung des Arbeitsraums zur Magnetfeldentzerrung* wird diese ausführlich beschrieben.

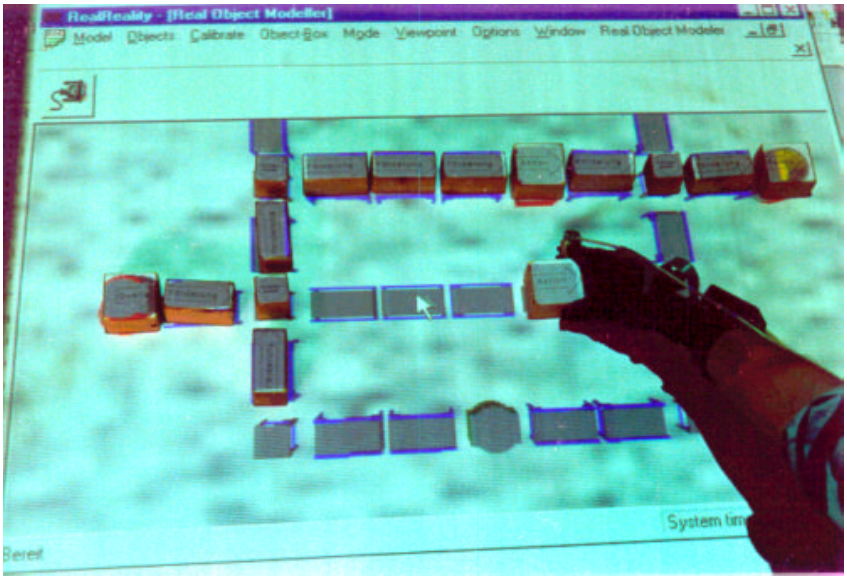

Bild 3: Rekonstruktion eines Modells auf dem Modelltisch

4.3.2 Kalibrierung des Arbeitsraums zur Magnetfeldentzerrung

Bei der Projektion von virtuellen Komponenten auf ein reales Modell kommt es zu einer Überlagerung. Projektion und Realität liegen direkt übereinander. Fehler, die sich in der Kette

reales Modell ⇒ Trackingsystem ⇒ virtuelles Modell ⇒ Projektion

ergeben, treten sehr auffällig zutage. Trackingsystem, virtuelles Modell und Projektion werden im Hinblick auf die gewünschte Genauigkeit korrigiert. Die Fehler die sich in der Kette ergeben, können dadurch reduziert werden, und treten vermindert in Erscheinung.

Erster Schritt zur Steigerung der Genauigkeit ist eine Verbesserung der Positionsdaten durch eine Kalibrierung des Trackingsystems. Skalierung, Positionierung und Verzerrungen können hierdurch ausgeglichen werden (Livingston 98).

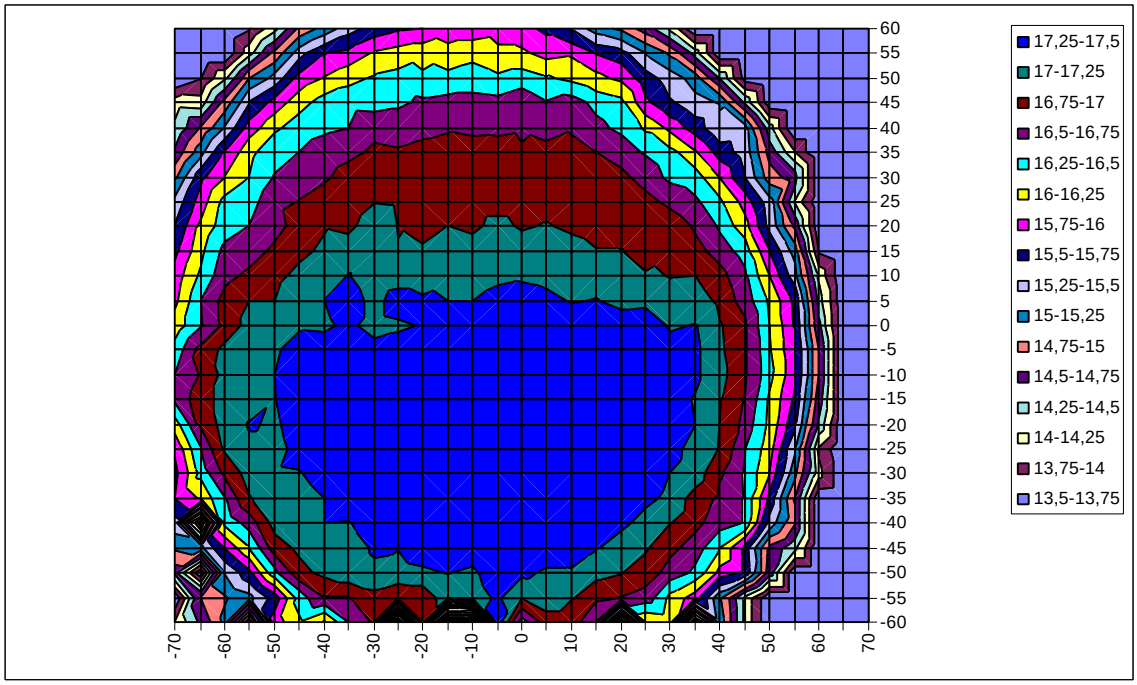

Abb 9: Diagramm des vom Trachingsystem gemessenen Höhenfehlers auf einer ebenen Fläche (h ≡ 10cm)

Eine mechanische Vermessung des Arbeitsraums wurde mit den von Trackingsystem gelieferten Daten verglichen. Je nach Entfernung zum Magnetfeldemitter und Position auf der Arbeitsplatte betragen die Abweichungen bis zu 70 mm. Das ist für eine sinnvolle Modellierung natürlich zuviel (Abb 9). Die Wiederholgenauigkeit der Trackingdaten ist erheblich besser, speziell, weil in unserer Arbeitsumgebung keine metallischen Objekte im Arbeitsraum bewegt werden. Diese liegt bei schneller Positionierungswiederholung bei 0,5 mm. Bei längerem Betrieb über mehrere Stunden ist eine Drift zu beobachten, die mit Temperaturschwankungen oder einer betriebsbedingten Erwärmung des Systems zusammenhängt. Die Abweichungen können hierdurch auf 5 mm steigen. Der Gewinn an Genauigkeit, der durch eine Kalibrierung zu erwarten ist, läßt sich je nach Dauer der Modellierung durch den Vergleich der absoluten Rohmeßwerte mit der Wiederholgenauigkeit abschätzen. Bei einem Rückgang von 70 mm auf 0,5 bis 5 mm entspricht das einer 14 bis 140 mal größeren Genauigkeit. Für die Kalibrierung müssen Referenzpunkte auf dem Modelltisch mechanisch genau vermessen, und mit den Werten des Trackingsystems verglichen werden, um Korrekturwerte zu erhalten. In Bereichen, die zwischen den netzartig angeordneten Referenzpunkten liegen, wird der Korrekturwert interpoliert (Abb 10).

Die Referenzpunkte 0 bis 8 liegen fest und sind auf der Arbeitsplatte markiert. Die anderen Punkte werden in Abhängigkeit von der gemessenen Abweichung automatisch bestimmt. Ausgehend von der Mitte wird der Sensor zu den äußeren Punkten auf einer Linie verschoben. Die Höhenabweichung zwischen innerem und äußerem Punkt wird skaliert und aufgeteilt. Wenn die beim Überfahren gemessene Abweichung ein Drittel der Gesamtabweichung überschreitet, wird automatisch ein Zwischenpunkt erzeugt. Durch diese fehlerabhängige Verteilung werden in Bereichen eines größeren Fehleranstiegs dichtere Punkte gesetzt. So kann mit einem geringen Zeitaufwand die bestmögliche Verbesserung der Genauigkeit erzielt werden. Zwischen den Stützpunkten wird linear interpoliert. Die verwendeten Algorithmen beanspruchen nur wenig Re-

chenzeit, so daß die Echtzeitanforderungen bei der Aktualisierung der Sensordaten nicht beeinträchtigt werden.

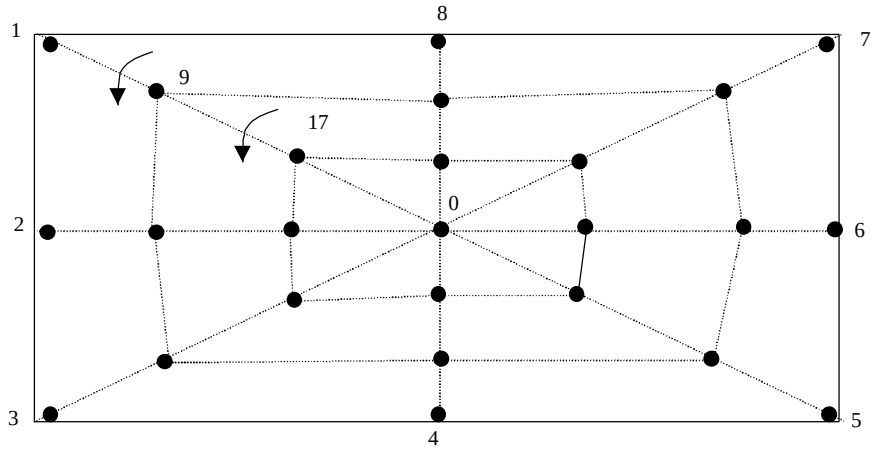

Abb 10: Netz von Kalibrierungspunkten

4.3.2.1 Vermessung des Desktop-Bereichs

In Abschnitt *4.2.1* wurde die Interaktion mit der Projektion des Desktops beschrieben. Bei der Vermessung der Projektionsfläche wird das virtuelle Desktopobjekt über zwei Eckpunkte an die Projektionsfläche angepaßt. Durch räumliche Verzerrungen bei den Tracking-Meßwerten kann es in Abhängigkeit zu diesen Punkten zu Ungenauigkeiten kommen. Die Kalibrierung des Trakkingsystems reduziert diese Verzerrungen deutlich, so daß eine gute Übereinstimmung zwischen Fingerspitze und projiziertem Mauszeiger erzielt wird. Das ist Voraussetzung für die Interaktion mit projizierten Desktopobjekten im „Virtual Touchscreen".

4.3.2.2 Vermessung des ROMAN-Fensters

Die genaue Modellierung mit *Real Reality* führt zu einem kongruenten virtuellen Modell im Rechner. Dieses sollte verändert werden, seine Genauigkeit ist durch die Kalibrierung des Trakkingsystems bereits optimiert. Über die Position der virtuellen Kamera kann das Modellfenster an das reale Modell angepaßt werden. Die ideale Kameraposition wird aus der Vermessung der Eckpunkte des Darstellungsfensters bestimmt. Abgesehen von optischen Verzerrungen, die sich durch die Projektion ergeben, kann so eine gute Übereinstimmung zwischen Modell und Projektion erzielt werden.

Die Lichtstrahlen der Projektors treffen nicht parallel auf die Arbeitsfläche. Je nach Brennweite des Projektionsobjektivs ergibt sich ein Öffnungswinkel, der in Abhängigkeit der Höhe über dem Tisch zu einer Verkleinerung des Bildes führt. Es zeigt sich jedoch, daß dieser Effekt dadurch kompensiert wird, daß auch die virtuelle Kamera einen Öffnungswinkel aufweist. Durch diese Kameraperspektive erscheint ein Objekt, daß durch seine Höhe näher am Objektiv liegt größer. Wenn Projektor und virtuelle Kamera den selben Öffnungswinkel aufweisen, hebt sich die Perspektive auf. Der Projektor verfügt über ein Zoom-Objektiv, so kann an dieser Stelle der Winkel angepaßt werden. Lediglich in größeren Höhen über dem Arbeitstisch ergibt sich durch die Defokussierung eine Unschärfe, die die Qualität der Projektion beeinträchtigt.

4.3.3 Verwaltung virtueller Bausteine

Für die Modellierung wurden verschiedenen Sätze von Twin-Objects angefertigt. Auf der gegenständlichen Seite bestehen diese aus einer stofflichen Repräsentation der Modellkomponenten aus Holz oder aus Baukästen entnommenen Kunststoffbausteinen wie z.B. Fischertechnik oder

Lego. Einige der gegenständlichen Bausteine sind mit Motoren, Pneumatikzylindern und Sensoren ausgestattet, so daß sie auch über eine reale Funktionalität verfügen. Der virtuelle Teil der Bausteine im Rechner wird durch verschiedene, im folgenden beschriebene Aspekte definiert und in 3 zugehörigen Dateien gespeichert.

SimulationModeling Language (SML) enthält:

- Eine Initialisierungsroutine
- Einen Verweis auf Geometrie
- Skalierung
- SensePoints[5]
- Bausteintyp
- Verweis auf Simulationsverhalten
- Verweis auf Basistyp für die hierarchische Spezifikation

Geometrien (Natural File Format (*.NFF), VRML2 (*.WRL), 3D Studio (*.3DS), AutoCad (*.DXF)) enthalten:

- Geometrie
- Verweise auf Textur-Bitmaps (nicht bei DXF)
- Facettenfarben

Simulationsverhalten (BST) enthält:

- Definition des Simulationsverhaltens
- Steuerprogramme

Diese Bausteininformationen erlauben die Modellierung, Visualisierung und die automatische Verknüpfung der Bausteine zu Simulationsmodellen.

Für eine Datenreduktion und zur besseren Verwaltung der Daten sind alle drei Kategorien hierarchisch aufgebaut (sofern VRML zur Beschreibung der Geometrie verwendet wird). Gemeinsame Eigenschaften können in Supertypen beschrieben werden. Zum Beispiel:

„LangesFörderband“ ist ein

„Förderband“ ist ein

„Materialflußelement“ ist ein

„Modellobjekt“ ist ein

„Objekt“.

Die Erstellung von Bausteinen wird hierdurch erheblich vereinfacht, weil bereits erstellte Bausteine als Ausgangsbasis für neue verwendet werden können und lediglich die Änderungen eingegeben werden müssen.

Der Simulator SIMPLE++ verwaltet einen eigenen Bausteinkasten. Simulationsbausteinklassen werden mit dem Editor von SIMPLE++ modelliert und dann als Bausteindatei (*.BST) gespeichert.

Der Simulator wurde um eine dynamische Bausteinverwaltung erweitert. Fehlende Bausteine werden automatisch aus den zugehörigen Dateien geladen, wenn sie nicht im Bausteinkasten vorhanden oder veraltet sind.

[5] Für die Modellierung und Modellanalyse relevante Positionen an den Objekten

4.4 Vormachen von Steuerungsprogrammen

Das Planen komplizierter Abläufe in Industrieanlagen erfordert Hilfsmittel zu ihrer Veranschaulichung. Bei der Planung einer Expreßgutsortieranlage wurden die Abläufe im System bei artec an einem Modell durchgespielt. Aus Videoaufzeichnungen konnte dieses Verhalten später rekonstruiert und auf ein Simulationssystem übertragen werden (Abb 11). Aus diesem Vorgehen entstand der Wunsch, ein System zu entwickeln, das Hilfestellung bei der Transformation der Bewegungen am Modell, hin zu Steueralgorithmen für Simulatoren oder Anlagensteuerungen leistet. Diese Idee erweitert die Ansätze des Programmierens durch Vormachen (Cypher 94) auf dreidimensionale Modelle und wurde mit dem *Real Reality* Konzept verwirklicht.

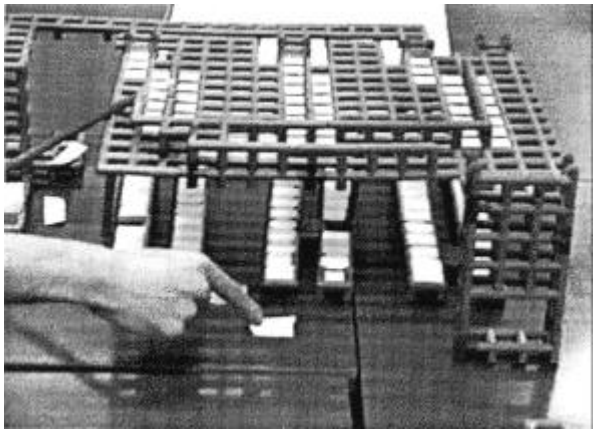

Abb 11: Modell einer Expreßguttransportanlage

Gegenständliche Modellbausteine werden mit der Hand so bewegt, wie es die spätere Anwendung erfordert und die Bewegung wird im virtuellen Modell aufgezeichnet. Durch die Interpretation dieser vorgemachten Dynamik durch den Computer kann die Aktivität des Systems in einer abstrakten Form als Programm ausgegeben werden, das später eine reale Anlage steuern kann. Im Vorfeld können mit den generierten Programmen Computersimulationen durchgeführt werden. Die Kombination des Aufbaus von statischen Modelltopologien mit dem Vormachen dynamischen Systemverhaltens erlaubt also eine sehr weitreichende Nachbildung realer Systeme mithilfe gegenständlicher Modelle.

Verhalten durch Vormachen ist eine Möglichkeit, Modelle und Steuerprogramme in sinnvoller und intuitiver Weise zu spezifizieren. Diese Arbeitsweise stellt einen relativ neuen Ansatz dar. Hieraus ergibt sich, daß bisher kaum Verfahren zur Interpretation von Verhalten in diesem Bereich existieren. Interpretation von Verhalten bedeutet, daß aus den zunächst *quantitativen* Bewegungsdaten *qualitative* Aussagen gewonnen werden. Erst diese qualitative und abstrahierte Repräsentation der Dynamik, die in eine geeignete Programmiersprache gefaßt werden kann, erlaubt die Übertragung auf reale und simulierte Systeme und die dortige Verifikation des Verhaltens.

Um dieses zu erreichen, wird ein wissensbasierter, auf logischer Programmierung beruhender Ansatz verwendet. Dieser ermöglicht es, das zur Interpretation erforderliche Kontextwissen in einem Computer zu speichern und automatisch auf das vorgemachte Verhalten anzuwenden. Hierdurch wird ein wichtiger Beitrag zur durchgängigen Modellierung von Fertigungsanlagen geleistet, der von großer Bedeutung für die Praxis ist.

4.4.1 Modellabstraktion in sieben Stufen

Zur grundlegenden Beschreibung der Informationsverarbeitung in *Real Reality* Systemen wurde ein Verfahrensmodell entwickelt. Dieses beschreibt die sequentielle Informationsverarbeirung

vom gegenständlichen Modell bis zum vollständig spezifizierten Simulationsmodell in sieben Stufen.

Von Stufe zu Stufe findet eine Abstraktion der Informationen statt. Alle Veränderungen des Modells gehen von der gegenständlichen Benutzungsschnittstelle aus. Je nach Bedarf können Informationen als Feedback visuell oder akustisch zurückgemeldet werden, und die Anwender bei der Modellierung unterstützen (Abb 12).

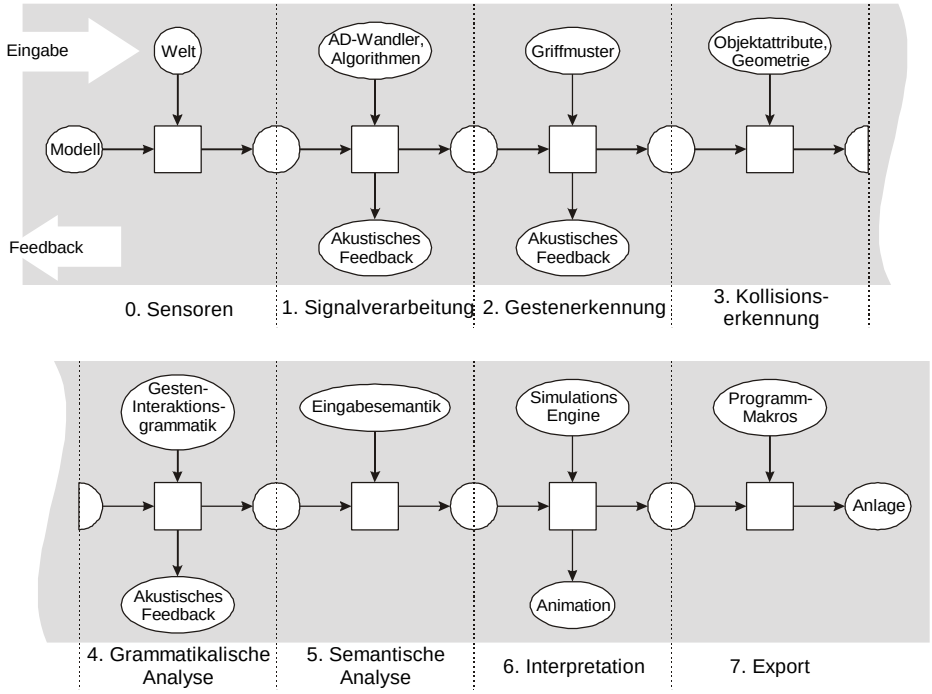

Abb 12: Stufenmodell zur Informationsverarbeitung an *Real Reality* Modellen

Stufe 0: Sensoren

Ausgangspunkt sind die Akteure und das gegenständliche Modell als Handlungs- und Interaktionsraum. Durch Sensorisierung der Hände werden hier die Aktionen aufgezeichnet, die zur Anpassung (Synchronisation) des virtuellen Modells notwendig sind.

Feedbacks vom Computer werden in diese reale Welt über Displays, Vertikalprojektion, Augmentation des Modells oder über akustische Kanäle eingeblendet. Sofern mit diesen Repräsentationen von Information interagiert werden kann, sind sie auch Teil der Modellumgebung.

Stufe 1: Signalverarbeitung

Die Sensoren liefern elektrische Signale, die zunächst digitalisiert und dann in ein geeignetes Datenformat transformiert werden. Dieses geschieht durch entsprechende Treiber, die darüber hinaus hardwarespezifische Korrekturen aufgrund von Kalibrierungen vornehmen. Die Sensorrohdaten können für Servicezwecke angezeigt werden.

Stufe 2: Gestenerkennung

Für einige Hardware, z.B. für Datenhandschuhe ist eine Mustererkennung erforderlich. Aus dem Datenstrom werden spezielle Griff- und Gestenmuster, deren Eintreten eine besondere Bedeutung hat, herausgefiltert.

Die Verwendung des oben beschriebenen *Grasp Trackers* erfordert die Interpretation der Drucksensordaten in Zusamenhang mit den Orientierungsdaten des Trackingsystems. Je nach Neigungswinkel des Fingers lassen sich Zeige- und Griffgesten voneinander unterscheiden.

Auf dieser Stufe hat sich ein akustisches Feedback bewährt, um erkannte Gesten anzuzeigen. Dieses ist der sicheren Handhabung der Interfacetechnologie zuträglich.

Stufe 3: Kollisionserkennung

Auf dieser Stufe werden die Interaktionen der Benutzer mit den Modellobjekten und zwischen Modellobjekten untereinander nachgebildet. Wichtig sind hier Kollisionen zwischen Hand und Objekten, die, in Verbindung mit Gesten, Modellveränderungen initiieren. Auch bestimmte Relationen zwischen Modellobjekten, wie z.B. Aufstapeln oder das Überfahren von als Sensoren markierten Positionen (Sensepoints) im Modell wird hier herausgefiltert.

Stufe 4: Grammatikalische Analyse

Die verschiedenen Interaktionsmöglichkeiten zwischen Händen, realen und virtuellen Objekten werden in einer Gesten-Interaktionsgrammatik (Abschnitt *4.2.3*, S.11) beschrieben. Diese verknüpft Gesten und topologische Relationen zu Interaktionen mit dem Modell.

Relevante Interaktionen wie Markieren oder Greifen von Objekten können durch akustische Signale gemeldet werden und die Modellierung unterstützen.

Stufe 5: Semantische Analyse

Je nach Anwendungskontext kann am Modell vorgemachtes dynamisches Verhalten als Vormachen interpretiert werden. Beispielsweise Montagevorgänge, Bewegungen von Robotergreifern oder Verzweigungsregeln können in dieser Stufe aus den dynamischen Modelldaten herausgefiltert und in anwendungsspezifischer Syntax gespeichert werden.

Stufe 6: Interpretation

Das nach Durchlaufen der vorherigen Stufen spezifizierte Modell und dessen Verhalten, definiert ein Simulationsmodell. Durch Simulationsläufe und Animation kann das abstrahierte Verhalten den Modellierenden dargestellt und von diesen beurteilt werden. Diese Stufe ermöglicht die Verifikation und somit die iterative Spezifikation von Verhalten.

Stufe 7: Export

Die Steuerprogramme, die den Ablauf der Simulation bestimmen, werden aus dem Modell extrahiert und in ein Format überführt, das von Industriesteuerungen gelesen werden kann.

4.4.1.1 Verwendetes Kontextwissen

In jeder Stufe wird Kontextwissen verwendet, um relevante Informationen aus dem Datenstrom zu extrahieren. Jeweils Ein- und Ausgabeformat der Stufe sind eindeutig definiert. Die Verarbeitung innerhalb der Stufen geschieht nach verschiedenen Verfahren. Die Kontextinformationen, die auf den verschiedenen Stufen herangezogen werden, sind folgende:

Stufe 0:

Realität, wie sie durch Physik und die Interaktionen zwischen Akteuren und Modell gegeben ist.

Stufe 1:

Das Verhalten der Datenerfassung und die auf die Daten angewendeten Algorithmen bestimmen die Vorgänge auf dieser Stufe.

Stufe 2:

Vorgegebene Griffmuster dienen zum Herausfiltern charakteristischer Gesten aus dem Datenstrom.

Stufe 3:

Spezifische Informationen zu den virtuellen Objekten werden herangezogen, um Relationen zu erkennen. Wichtigste Eigenschaften sind Greifbarkeit und Kollisionsgeometrie.

Stufe 4:

Die Gesten-Interaktionsgrammatik gibt die Informationsverarbeitung auf dieser Stufe vor.

Stufe 5:

Eine an den Anwendungskontext angepaßte Eingabesemantik spezifiziert, wie Verhalten in Regeln umgesetzt wird. Die stufen 1 bis 5 bereiten die Informationen vom Modell für die Behandlung als Spracherkennungsproblem vor.

Stufe 6:

Modellbausteinspezifische Verhaltensdefinitionen werden mit vorgemachen Regeln und Aspekten der erkannten Topologie zu einer Beschreibung des Gesamtverhaltens des Modells zusammengeführt. Eine entsprechende Verarbeitungs-Engine (Simulator) berechnet das so spezifizierte dynamische Verhalten.

Stufe 7:

Die für die gegebenen Zielsteuerungen relevanten Steuerungsalgorithmen werden aus dem Modell extrahiert und mit Sprachmakros zu lauffähigen Programmen verknüpft, die auf Industriesteuerungen lauffähig sind.

4.4.2 Eingabesemantiken zum Programmieren durch Vormachen

Eingabesemantiken beschreiben die Bedeutung von Symbolen und Symbolfolgen. Diese Verarbeitung ist der Stufe 5 zugeordnet. Indem reale Gegenstände, Gesten und Aktionen als Symbole begriffen und interpretiert werden, lassen sich beliebige Eingaben in beliebigen Kontexten als Ausdrücke interpretieren. Mit dieser sehr abstrakten Methodik zur Verarbeitung von dynamischen Daten aus gegenständlichen Modellierumgebungen wird eine große Flexibilität und Universalität erreicht. Mit wissensbasierten Programmiersprachen wie z.B. PROLOG lassen sich komplizierte Eingaben einfach interpretieren, wenn ihre Bedeutung zuvor natürlichsprachlich und eindeutig beschrieben werden konnte. PROLOG bietet mit seiner aus Fakten und Regeln zusammengesetzten Struktur die Möglichkeit, in natürlicher Sprache formulierte Zusammenhänge auf einfache Weise in eine formale Sprache zu überführen. Die Informationen liegen dann in Klartextform vor und werden erst zur Laufzeit vom System interpretiert. Dadurch wird eine maximale Offenheit, Supportfähigkeit und Erweiterbarkeit des Systems ermöglicht.

Mit einem in PROLOG implementierten Tool zur Datenanalyse wird an einem Szenario aus der Produktionstechnik die Effektivität dieses Verfahrens demonstriert. Als Herausforderung sollen an einem Modell verschiedenartige diskret und kontinuierlich arbeitende Einrichtungen, programmiert und simuliert werden. Mit einem Modell aus Robotern und speicherprogrammierbaren Steuerungen werden die verschiedenartigen Geräte über eine geschlossene Semantik durch Vormachen programmiert. Als Ergebnis liegen Programme für die Steuerungen vor, deren Lauffähigkeit mit verschiedenen Simulatoren gezeigt wird.

Dieser Ansatz führt die verschiedenen in RUGAMS entwickelten Eingabemethoden[6] zu einem handhabbaren und leicht zu wartenden System zusammen.

[6] Die wichtigsten Methoden gliedern sich in:

- Modellieren mit gegenständlichen Abbildern von Fertigungseinrichtungen
- Verwendung von Bausteinen mit symbolischer Bedeutung
- Vormachen dynamischen Verhaltens mit Modellen beweglicher Teile
- Interaktionen mit rein virtuellen über Projektion ins Modell eingeblendeten Informationen

4.4.3 Kontextwissen für produktionstechnische Prozesse

Die zur Modellierung eines Systems verwendeten Modellkomponenten haben bestimmte Funktionen zu erfüllen. Hierfür sind sie mit einem aktiven oder passiven Verhalten ausgestattet. Zunächst gilt es, dieses Verhalten zu beschreiben. Auch Beziehungen zwischen den verschiedenen Modellkomponenten spielen eine Rolle. Die Einzelkomponenten verfügen dafür über Schnittstellen, über die diese Beziehungen aufgebaut werden können.

Die Kombination der Eigenschaften und Verhalten zu einem Simulationsmodell geschieht nach definierten Regeln. Diese Regeln zusammen mit der Beschreibung des Verhaltens und der Schnittstellen der Einzelbausteine werden als *Kontextwissen* bezeichnet.

Wissen über mögliche Verknüpfungen

Das hierfür eingesetzte Verfahren der Topologieanalyse basiert auf logischen Konstrukten, die dem virtuellen Teil der Twin-Objects zugeordnet sind. Jedes Objekt erhält als Attribut eine Liste von sogenannten *SensePoints*[7], welche virtuelle Anschlußpunkte zu benachbarten Objekten darstellen. Bei der automatischen Topologieanalyse wird davon ausgegangen, daß ein nahe bei einem Ausgangs- SensePoint gelegener Eingangs- SensePoint mit diesem als verbunden gelten soll. Die Begründung hierfür liegt im Verhalten von realen Funktionselementen. Wird eine Palette von einem aktiven Materialflußelement (Förderband) ausgegeben und befindet sich in unmittelbarer Nähe ein aufnahmefähiges Materialflußelement, findet ein Übergang statt, wie z.B. bei zwei in gleicher Richtung laufenden Förderbändern, die mit ihrem Ende bzw. Anfang unmittelbar aneinandergrenzen (Abb 13). Für die Topologieanalyse wird dieses Prinzip genutzt, indem jeder Ausgangs- SensePoint mit allen Eingangs- SensePoints, die sich innerhalb einer definierten Umgebung befinden, verbunden wird. Die Topologieanalyse überführt also die Einzelkomponenten, die als Knoten eines Graphen betrachtet werden können, durch die Verbindung mit gerichteten Kanten in einen gerichteten Graphen, der als Grundlage für Simulationsmodelle dienen kann.

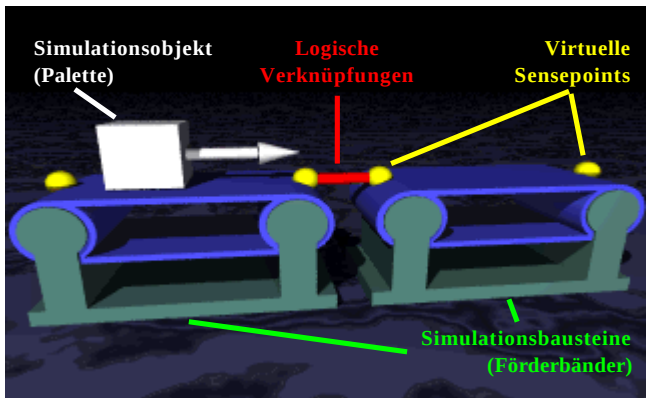

Abb 13: Verknüpfung von Materialfluß mittels *SensePoints*

Wissen über das Steuerverhalten

Die Koordination der Abläufe in Fertigungsanlagen wird von elektronischen Steuerungen übernommen. Über Signal- und Steuerleitungen sind die Komponenten miteinander verbunden. Auch dieser Aspekt soll in gegenständlichen Modellen abgebildet werden, denn er ist Voraussetzung für die Definition und den Test von Steueralgorithmen. Die Verbindungen mit Signalleitungen können nicht wie bei den Materialflußverknüpfungen aus der Nähe der Komponenten

[7] Der Name „Sensepoint“ ist zur Bezeichnung von „sensitiven“ Punkten entstanden. Die Annäherung mit einer Palette löst hier ein Ereignis aus. Inzwischen werden diese Punkte vielfältiger eingesetzt, mit dem Schwerpunkt der Topologieanalyse. In zukünftigen Berichten und Projekten werden deshalb die „Connectoren“ diese Positionen bezeichnen. Für die Definition von Sensoren zur Ereignisauslösung bietet das zukünftig eingesetzte VRML2 Format zur Objektbeschreibung eigene Methoden.

erkannt werden, weil die Verbindungsleitungen eine fast beliebige Länge haben können. Auch die gegenständliche Modellierung aller Verbindungsleitungen und -schläuche zwischen Steuerung und Aktoren und Sensoren ist nicht sinnvoll, da sich durch zu viele Elemente ein unübersichtliches Modell ergeben würde. Für die Signalverknüpfungen wird daher ein grundsätzlich anderes Prinzip genutzt, das neben der Topologie auch die am Modell vorgemachte Dynamik berücksichtigt.

Von der Verbindung einzelner Aktoren und Sensoren mit einer Steuereinheit, wie sie die realen Komponenten aufweisen, wird in dem hier beschriebenen Konzept zunächst abstrahiert. Die Verknüpfungen zwischen den Modellelementen wird auf der Ebene von Steuernachrichten vorgenommen, die Anweisungen und Abfragen über Zustände repräsentieren. Durch Vormachen wird eine Anlagensteuerung generiert, die die Steuerungen der Modellelemente zu einem funktional zusammenhängenden Gesamtsystem verbindet und die Aktionen der einzelnen Komponenten koordiniert (Abb 14).

Es wird ein objektorientiertes hierarchisches Steuerungsmodell verwendet. Objektspezifische Steuerungen sind zusammen mit den Verbindungen zu Aktor-Ein- und Sensor-Ausgängen innerhalb der Objekte definiert. Von den Objektsteuerungen werden definierte Schnittstellen zur Anlagensteuerung angeboten. Für die Steuerung einer Verzweigung reicht es beispielsweise aus, eine Methode oder Merker anzubieten, mit der zwischen dem Verhalten „Abzweigen“ und „Durchlassen“ ausgewählt werden kann. Wie der Abzweigungsvorgang im Einzelnen abläuft und wie die Sensoren und Aktoren miteinander in Verbindung stehen, ist für die globale Anlagensteuerung irrelevant und wird von der Objektsteuerung gekapselt.

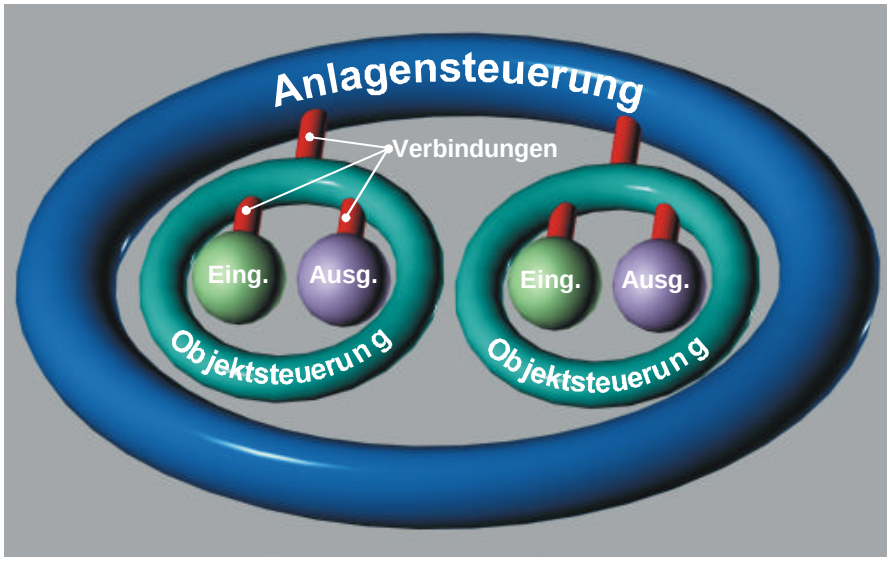

Abb 14: Hierarchisches Steuerungsmodell

4.4.4 Petri-Netze zur Darstellung von Steuerprogrammen

Zur Modellierung eines hierarchischen Steuerverhaltens durch Vormachen wird ein sehr flexibles Werkzeug benötigt. Zur Darstellung werden deshalb Petri-Netze verwendet, die diese Anforderungen erfüllen. Sie sind in der Lage, nebenläufige Prozesse hierarchisch abzubilden (Reisig 85, Kämper 91). Sie erlauben außerdem gleichermaßen eine anschauliche Repräsentation von Abläufen sowie die Definition einzelner Bedingungen, wie Sensoren und Aktoren zusammenarbeiten. Sie vereinigen damit Vorteile der Ablaufsprache (AS) und der Darstellung von SPS-Programmen in AWL, KOP, FUP oder ST, wie sie in (DIN 61131-3) definiert sind. Als Programmiersprache für reale Steuerungen sind Petri-Netze bisher wenig etabliert. Es besteht jedoch die Möglichkeit, Petri-Netze mit einfachen Methoden in Anweisungslisten zu überführen, die für die meisten

Steuerungen eingesetzt werden können (Abschnitt *4.4.6*). Diese werden verwendet, um aus der Simulation der Anlagensteuerung heraus Steuerprogramme zu exportieren (Abb 15) und zunächst auf einer Soft-SPS, die eine Siemens S5 Steuerung nachbildet, zu simulieren und später auf einer realen Steuerung zu testen.

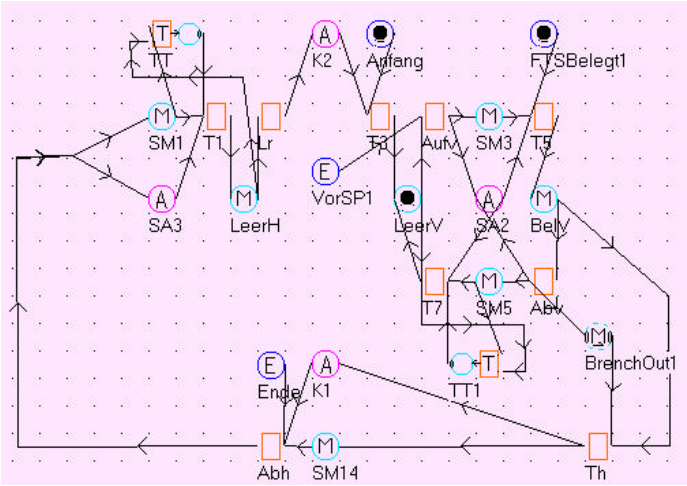

Abb 15: Petri-Netz zu Steuerung der Simulation eines Transportsystems

4.4.5 Programmierung von Petri-Netzen durch Vormachen

Auch die Programmierung von Verhalten durch Vormachen basiert auf Kontextwissen, nämlich der Art und Weise, wie Vorgemachtes Verhalten in Steuerprogramme überführt werden soll. Auf welche Weise Verhalten vorgemacht werden muß, um automatisch interpretiert werden zu können und wie die Interpretation von Verhalten technisch umgesetzt ist, wird durch eine *Eingabesemantik* beschrieben.

4.4.5.1 Eingabesemantik für Petri-Netze

Im Vergleich zum Vorjahr erlaubt die Modellierung von Steuerungen mit Petri-Netzen eine sehr abstrakte Beschreibung des Vormachens von Regeln. Das Verfahren ist leicht erweiterbar und auf verschiedene Anwendungskontexte projizierbar.

Ergebnis einer vorgemachten Eingabe ist immer die Verknüpfung von Voraussetzungen mit der Auslösung (Triggerung) von Aktionen. Im Petri-Netz wird eine vorgemachte Regel als Transition repräsentiert. Jede Eingabeaktion erzeugt also eine Transition im Steuerungsnetz und verbindet diese mit den Bedingungen und einem Trigger (Abb 16).

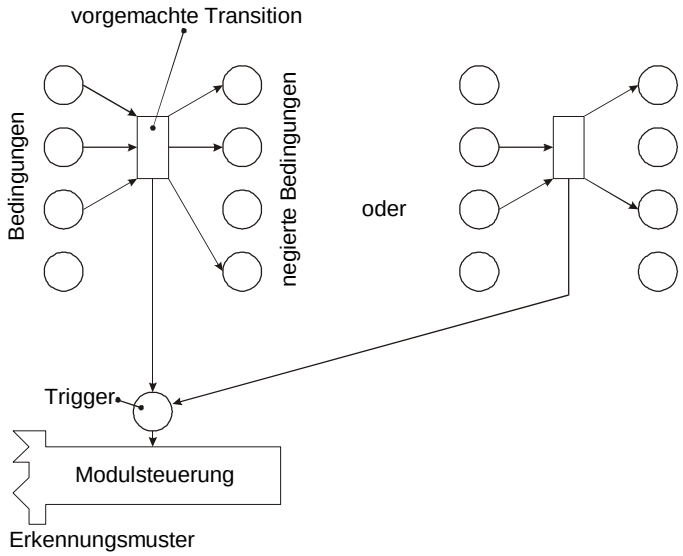

Abb 16: Vorgemachtes Steuerverhalten

Das Vormachen von Regeln geschieht in zwei Phasen:

1. Herstellen der Situation, in der ein Verhalten angewendet werden soll
2. Vormachen des Verhaltens

Die Dynamik im Modell wird einer Mustererkennung unterzogen. Dabei wird das vorgemachte Verhalten mit vorgegebenen Mustern verglichen, die einzelne Modulsteuerungen kennzeichnen. Erkennungsmuster können verschiedene Merkmale haben. Wichtigstes Merkmal sind die beim Vormachen überfahrenen *SensePoints*. Wenn diese zu einem Objekt gehören, ist das erste Kriterium erfüllt. Weiteres Kriterium ist, ob es an dieser Stelle etwas vorzumachen gibt. Nur wenn eine Objektsteuerung an dieser Stelle das Materialflußverhalten beeinflussen kann, ist das Vormachen von Aktionen sinnvoll. Relevantes Merkmal einer Objektsteuerung ist das Vorhandensein einer Triggerstelle (Abb 16). Diese zwei Kriterien reichen aus, vorgemachte Aktionen zu erkennen. Das Wechseln in einen besonderen Eingabemodus zum Beginn des Vormachens kann entfallen.

Wenn eine Eingabeaktion erkannt wurde, werden die Bedingungen untersucht, unter denen sie stattfand. Es wird davon ausgegangen, daß die Anwender die Aktion bei einem Modellzustand durchgeführt haben, der nicht zu unzulässigen oder unerwünschten Zuständen führt. Bezogen

auf Fördersysteme bedeutet das, der Belegungszustand von Ressourcen, also von Maschinen, Lagern oder Puffern, wurde zuvor mit Modellpaletten hergestellt. Darüber hinaus muß dem System mitgeteilt werden, an welchen Stellen die Zustände berücksichtigt werden sollen. Nicht jede Palette im System soll relevant sein, um einen Zustand zu klassifizieren. Relevante Positionen werden in ihrem Belegungszustand deshalb mit Token markiert. Token sind Modellobjekte, die bei der Modellinterpretation eine besondere semantische Bedeutung haben. Jeder Token wird einem abfragbaren und lokalisierbaren Anlagenzustand, z.B. einem Sensor oder einem Merker, zugeordnet. In der gegenwärtigen Konfiguration kann der markierte Zustand entweder belegt oder leer sein. Das entspricht der Abbildung von Lichtschranken, Näherungssensoren oder Endschaltern, welche die wichtigsten Sensortypen in realen Steuerungen sind.

> Für andere Anwendungen ist das Konzept auch für Positions- und Winkelsensoren geeignet, bei denen statt Werten im Bereich Boolean reelle Zahlenwerte geliefert werden. Um diese als Zustand abfragen zu können, muß zusätzlich eine zulässige Abweichung gegeben sein, damit ein Intervall bestimmt werden kann. Abfragen, ob ein Wert innerhalb eines Intervalls liegt, liefern wiederum True oder False als Ergebnis und können somit, als diskrete Zustände, die Transitionen von einfachen Petri-Netzen steuern.

Mit Token markierte Zustände, die den Wert True haben, werden als Bedingung vor der Transition eingefügt, Zustände mit False hinter der Transition als negierte Bedingung (Abb 16). Eine solche Transition mit verbundenen Stellen repräsentiert ein Zustandsmuster, das beim Anlagenbetrieb oder in der Simulation auftreten kann und eine Aktion der Steuerung, wie z.B. den Ablauf „Abzweigen“ an einer Verzweigung, auslöst. Die Verknüpfung von einem Zustand mit einer Aktion wird über eine Triggerstelle hergestellt, die als Auslöser für bestimmte Steuerungsabläufe dient.

Übertragen auf die Programmiersprachen für SPS wird beim einfachen Vormachen einer Aktion eine „UND“ Beziehung aus den Bedingungen hergestellt. Auch „ODER“ Beziehungen sind durch Vormachen zu programmieren. Sollen verschiedene Zustände die gleiche Aktion auslösen, werden sie nacheinander mit Paletten und Token hergestellt und die Aktion vorgemacht. So können beliebig viele Zustände erzeugt werden, die das selbe Verhalten auslösen (Abb 16 rechts).

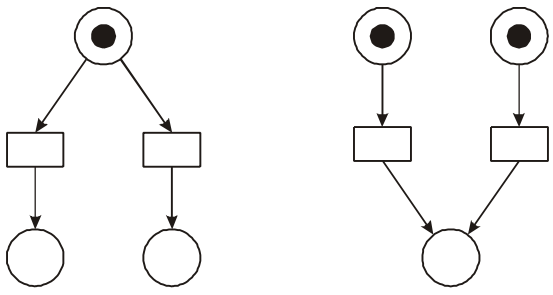

Abb 17: Konfliktsituationen

Beim Vormachen kann es passieren, daß ein Zustandsmuster verschiedene Aktionen oder verschiedene Zustandsmuster dieselbe Aktion triggern können, es liegt ein Konflikt vor (Abb 17). Die Modellierung als Petri-Netz gestattet neben dem Auffinden solcher Konflikte auch ihre Auflösung durch zufällige oder alternierende Auswahl der Möglichkeiten. SPS-Programme „lösen“ solche Konflikte auf einfachere Weise. Es wird immer die zuoberst in der Anweisungsliste stehende Aktion ausgeführt. Davon abweichendes Verhalten muß durch Veränderungen des Programmcodes bestimmt werden und ist damit von der verwendeten Sprache abhängig. Vorgemachtes Verhalten wird in der Reihenfolge, in der es vorgemacht wird, in die erzeugten SPS-Programme eingetragen. Diese liegen als ASCII-Text vor und können in dieser Form nachbearbeitet werden, oder in eine Programmierumgebung importiert und dort weiter verändert werden.

4.4.6 Umwandlung von Petri-Netzen in SPS-Programme

Petri-Netze stellen eine Möglichkeit zur SPS-Programmierung dar (Aspern 93). Leider stehen bisher wenig Systeme zur Verfügung, die die SPS-Programmierung mit Petri-Netzen unterstützen.

> In (DIN 61131-3) befindet sich die Beschreibung der Ablaufsteuerung (AS), die viele Parallelen zu Petri-Netzen aufweist. Diese ist jedoch zur Aktivierung von Programmodulen bestimmt und für die Sensor/Aktor-Ebene zu umständlich. Für die Programmierung durch Vormachen stellt sie jedoch eine interessante Basis dar.

Zunächst werden die durch Vormachen erzeugten Petri-Netze in Anweisungslisten überführt. Diese unterscheiden sich für die verschiedenen Steuerungstypen nur geringfügig und können z.B. für Siemens S5 Steuerungen mithilfe des Windows Clipboards leicht in das „S5 für Windows“ Programmier- und Simulationssystem von IBH Softec importiert und weiter bearbeitet werden. Aus dem programmiersystem heraus kann das Programm an die SPS übertragen werden. Vorgemachte Programme können auf diese Weise eine reale Anlage steuern.

Die Übertragung vom Petri-Netz in die Anweisungsliste geschieht mit einem sehr einfachen Algorithmus transitionsweise:

Alle Stellen vor der Transition werden als „UND“ Bedingung in folgendem Format in die Liste eingefügt (dargestellt im EBNF-Format zur Beschreibung von Grammatiken)

```
U    ( E | M | A ) X.Y \nl.
```

Alle Stellen nach der Transition werden als „UND NICHT“ Bedingung im Format

```
UN   ( E | M | A ) X.Y \nl
```

in die Liste eingefügt. Die Triggerstelle schließt die Anweisung ab

```
=    ( M | A ) X.Y \nl.
```

Automatisch generierten Programmcode zu lesen, ist immer eine mühsame und bei umfangreichen Programmen schwierige Aufgabe. Zu Erleichterung wird über jede Anweisung eine mit `;` beginnende Kommentarzeile gesetzt, die den vollständigen Pfad der Transition im Modell im Klartext angibt (z.B. `;.SPS_Halle.F1_FTS111.PetriSPS.T2`).

4.4.7 Verwaltung von Ein- und Ausgängen von Steuerungen

Auf der Hardwareebene sind die Ein- und Ausgänge mit einer Adressierung versehen über die die einzelnen Kabel-Anschlußklemmen auf verschiedenen I/O-Karten angesprochen werden. Um die Akteure nicht während der Eingabe von Verhalten mit der Vergabe von SPS-Adressen zu belasten, werden diese zunächst automatisch vergeben. Parallel dazu wird eine Symboltabelle angelegt. Diese ordnet den Adressen der Ein- und Ausgänge einen Namen zu, der sie innerhalb des Modells eindeutig identifiziert. Obwohl die Zuordnung von Sensoren- und Aktoren zu den Anschlußklemmen weitgehend beliebig ist, wird meistens angestrebt, die Adressen für die Anlagenteile zu gruppieren und einer bestimmten I/O-Karte zuzuordnen, um eine bessere Übersichtlichkeit und Wartbarkeit zu erzielen. Das für RUGAMS entwickelte System erlaubt deshalb, die Symboltabellen zu bearbeiten, um die Zuordnungen anforderungsgerecht vorzunehmen. Die Anschlüsse der SPS werden tabellarisch dargestellt. Durch einfaches Verschieben von Sensoren, Aktoren oder Merkern innerhalb der Tabelle können ihnen beliebige Adressen zugeordnet werden (Abb 18).

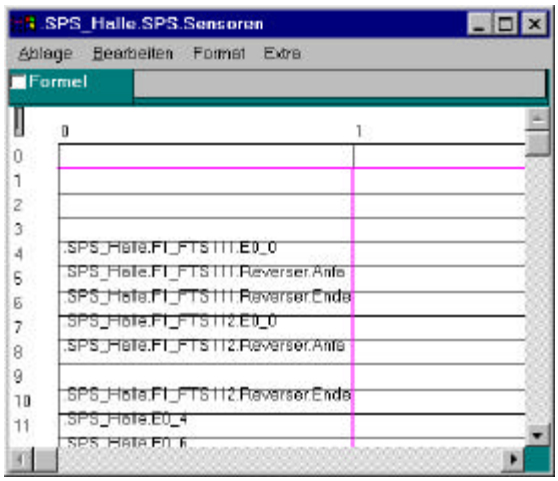

Abb 18: Verwaltung von Eingangsadressen von SPS

4.5 Simulation gegenständlich modellierter Szenarien

Bei der Modellierung von Anlagenkonfigurationen spielt die Simulation des Verhaltens eine wichtige Rolle. Besonders, wenn es um das vorgemachte Verhalten und die daraus resultierende Performance einer geplanten Produktionsanlage geht, stellen die Analyseergebnisse aus der Simulation ein wichtiges Feedback dar.

Wir haben bisher den

- Aufbau der Modelltopologie,
- die Verwaltung von Objektspezifischen Attributen,
- die Topologieanalyse,
- die Verwaltung von Simulationsbausteinen
- und die Spezifikation von Steuerverhalten durch Vormachen

beschrieben. Diese Informationen reichen aus, um aus der Szene vollautomatisch ein Simulationsmodell zu generieren, dieses zu simulieren und in 3d zu visualisieren. In diesem Abschnitt wird die Konvertierung vom virtuellen Modell in ein Simulationsmodell, die Beschaffenheit der Simulationsbausteine und die Konfiguration des Simulators beschrieben.

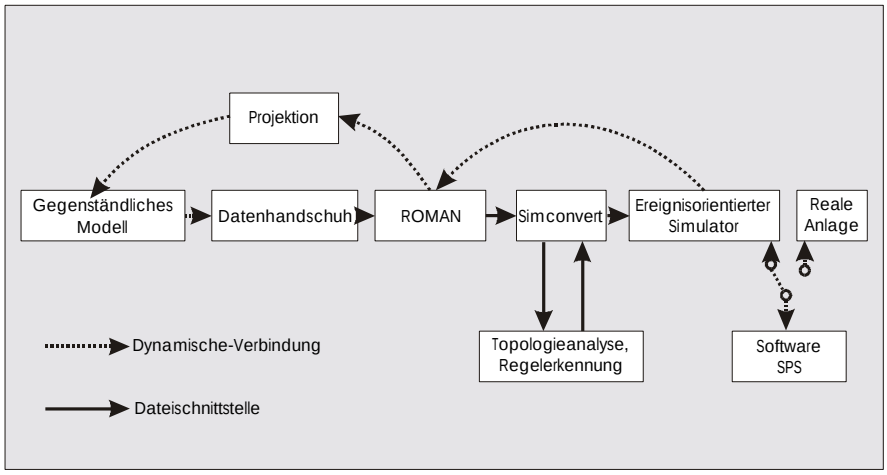

Abb 19: Konfiguration des Systems zur Modellierung und Simulation von Fertigungsanlagen in gegenständlichen Umgebungen

Abb 19 zeigt die Systemkonfiguration zur Modellierung und Simulation von gegenständlich modellierten Fertigungsanlagen. Schließlich wird die Möglichkeit aufgezeigt, durch die Verwendung einer Software-SPS das modellierte Steuerverhalten an einer Modellanlage aus Fischertechnik zu überprüfen. Durch die Verwendung einer im PC eingebauten digitalen I/O-Karte kann auf den Aufwand der Übernahme der Getesteten Anweisungsliste in ein SPS-Programmiersystem und die Übertragung auf die Steuerung verzichtet werden.

4.5.1 Erzeugen des Simulationsmodells

Wir gehen zunächst davon aus, daß nicht während der Modellierung und Programmierung gleichzeitig simuliert werden soll. Es ist zu befürchten, daß durch die Interaktion mit einem dynamischen Modell die Interaktionen nicht sicher ausgeführt werden können und sich für die Benutzer eine zu hohe Komplexität bei der Modellierung ergibt. Die Interaktion mit sich bewegenden Paletten wird zwar überlegt, bisher fehlen aber Konzepte für sinnvolle Interaktionen, obwohl unser Konzept zur Griffmodellierung das „Fangen“ animierter Paletten zuläßt. Zunächst kann also für die Simulation auf eine Dateischnittstelle zurückgegriffen werden. Das Modul „Simconvert“ liest die Szenenbeschreibung, speichert diese in einem Zwischenmodell, führt auf diesem die Topologieanalyse und Regelerkennung gemäß der oben beschriebenen Eingabesemantik durch und exportiert die Daten schließlich in ein vom Simulator lesbares Format.

Aufbau der Simulationsbausteine

Die Simulationsbausteine sind detailliert modelliert. Die Hardware wird bis auf jeden einzelnen Sensor und Aktor abgebildet, um Steuerprogramme an dieser virtuellen Hardware testen zu können. Innerhalb der Simulationsbausteine liegen Hardwarebeschreibung und Steuerprogramme in einer strikten Trennung vor. Diese Trennung ermöglicht es, das Steuerprogramm zu extrahieren und für die Verarbeitung in einer Steuerung zu exportieren. Für die Beschreibung des Verhaltens der Hardware wurden verschiedene standardisierte Beschreibungssprachen wie z.B. VHDL oder Modelica in Erwägung gezogen, bisher aber nicht verwendet, da Importmöglichkeiten für diese bei den Materialflußsimulatoren fehlen.

Simulation der SPS

Die Modellierung des Verhaltens der Steuerung wird *nicht* im Simulationsmodell abgebildet. Für diesen Zweck wird eine Software-SPS verwendet, die das Verhalten einer Siemens S5 SPS nachbildet (emuliert). Mit dieser Software-SPS lassen sich die Steuerprogramme schon bei der Simulation unter sehr realitätsähnlichen Bedingungen testen. Darüber hinaus ist der verwendete ereignisorientierte Simulator nur schlecht in der Lage, das zyklische Abarbeiten eines Steuerprogramms einer SPS zu simulieren (Schäfer 95, Fishwick 95). Der in C++ implementierte SPS-Emulator ist dazu erheblich besser in der Lage. Die im S5-Anweisungslistenformat definierten Elemente

- Eingang,
- Ausgang,
- Merker,
- die 5 Timertypen (Impuls, verlängerter Impuls, Einschaltverzögerung, speichernde Einschaltverzögerung, Ausschaltverzögerung)
- und Zähler

werden entsprechend der S5 Spezifikation für Anweisungslisten verarbeitet.

Über ein ereignisorientiertes Socket-Protokoll steht der SPS-Emulator mit der Simulation in Verbindung. Über dieses Protokoll kann auch die im Simulator generierte Anweisungsliste auf die Steuerung übertragen werden. Da der SPS-Emulator als Serveranwendung ausgeführt ist und

über ein Netztwerkprotokoll angesprochen wird, kann die SPS auf einem anderen Rechner ausgeführt werden als die Simulation. Versuche haben jedoch gezeigt, daß durch das hohe Aufkommen an kleinen Datenblöcken, die zur Synchronisation notwendig sind, die Performance sinkt. Wenn die Anwendungen auf einem Rechner laufen und lokal miteinander kommunizieren, sind Socketverbindungen unter Windows sehr schnell, so daß sich eine befriedigende Performance ergibt. Der Rechenaufwand zur Protokollverwaltung ist also höher, als der für die SPS-Emulation. Als Konsequenz dieses Ergebnisses ist zu fragen, ob die SPS als DLL implementiert werden sollte. Hierdurch würde sie einen Teil ihrer Flexibilität einbüßen, speziell was den plattformübergreifenden Einsatz betrifft, dafür ist eine Performancesteigerung zu erwarten.

Statt der Nachbildung der Hardware durch einen Simulator, könnten auch die Ein- und Ausgänge einer Modellanlage, die mit Sensoren und Aktoren ausgestattet ist (Abb 20), mit einer digitalen I/O-Karte und einem entsprechenden Treiber mit dem SPS-Emulator verbunden werden (Abb 19). Auf diese Weise erhält man eine Soft-SPS, die für das Experimentieren und Optimieren der Steuerprogramme eine gute Basis bildet. Dabei kann zwischen der Simulation und dem im Simulator integrierten Programmeditor, dem gegenständlichen Modell mit der Option zum Vormachen von Steuerverhalten und der realen Anlage beliebig hin und her geschaltet werden.

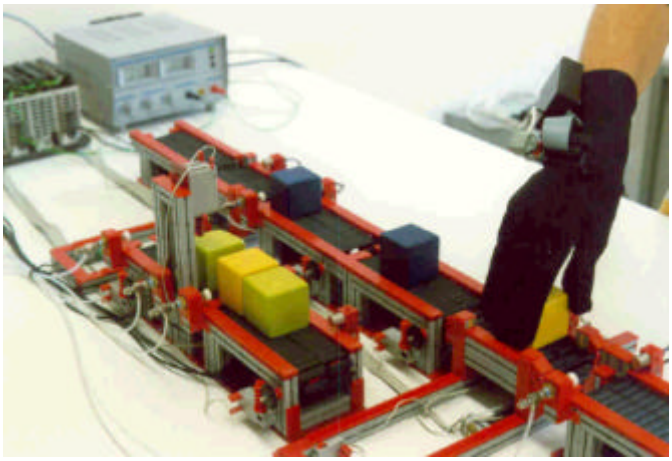

Abb 20: Mit Sensoren und Aktoren ausgestattete Fischertechnik Anlage zum Experimentieren mit vorgemachten Steuerprogrammen

Verwaltung der Bausteinsteuerungen

Das Steuerprogramm ist ein Bestandteil des beschriebenen Simulationsverhaltens von Förderbausteinelementen. Während ein Defaultverhalten bei allen Instanzen eines Typs gleich ist und die Simulation ohne Vormachen von Verhalten erlaubt, unterscheiden sich die Steuerungsteile, die das Entscheidungsverhalten in Konfliktsituationen bestimmen, nach dem Vormachen von Instanz zu Instanz. Aufgrund dieser Tatsache scheint eine objektorientierte Abbildung des Steuerverhaltens günstig. Die allen gemeinsamen Grundverhalten werden als Klassenattribute gespeichert. In den Klassen ist ein Defaultverhalten implementiert worden. Wenn die Instanzen beginnen, sich durch vorgemachte Verhaltenssteuerungen zu unterscheiden, wird das Defaultverhalten in den Instanzen überschrieben. Diese Arbeitsweise macht eine zuverlässige und effiziente Handhabung von Steuerprogrammen möglich. Aus verschiedenen Gründen wurden zur Abbildung des Steuerverhaltens Petri-Netze gewählt (vgl. Abschnitt 0). Diese können durch Vormachen oder mit grafischen Editoren (Abb 21) generiert und verändert werden. Die Verwaltung der Programme geschieht zusammen mit den Bausteinklassen und Instanzen im Simulator.

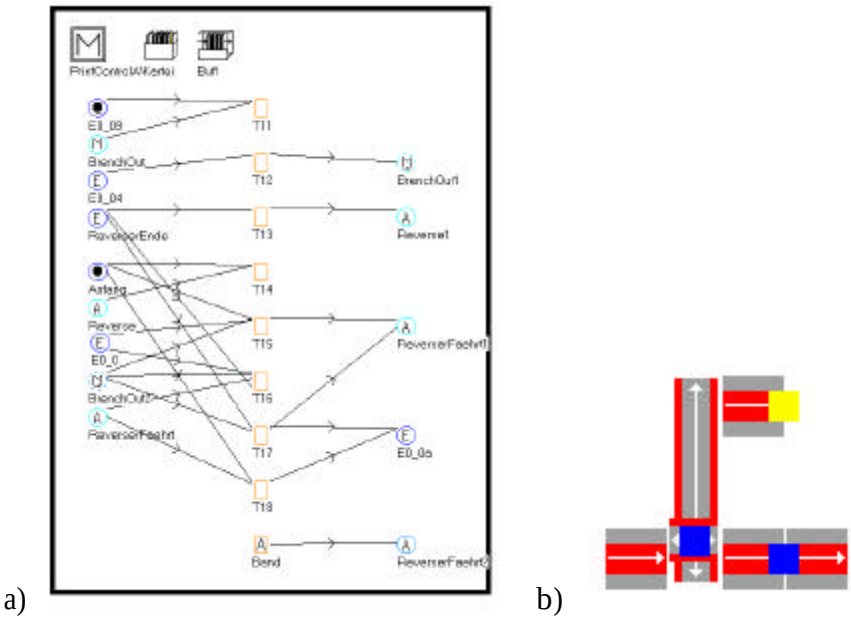

Abb 21: Petri-Netz zur Steuerung eines FTS

Der im Projekt eingesetzte Simulator SIMPLE++ bietet komfortable Methoden zum Editieren, animieren und Verwalten von Petri-Netzen und wird hierfür ebenso wie zur Verwaltung der Steuerprogramme eingesetzt.

In Abschnitt *4.4.6* wurde beschrieben, wie Petri-Netze mit einfachen Algorithmen in Anweisungslisten überführt werden können. Diese Algorithmen sind in SIMPLE++ mit der integrierten Programmiersprache SIMTALK implementiert worden.

4.5.2 Simulation des Modells

Die Simulationsbausteine müssen Schnittstellen aufweisen, über die die Kommunikation mit der SPS und der Visualisierung stattfindet. Der Simulationsbaustein, der einen Sensor abbildet, generiert Ereignisse, wenn eine Palette in seinen Bereich ein- oder austritt. Diese Ereignisse werden über die C-Schnittstelle und eine Socketverbindung an den SPS-Emulator weitergeleitet. Änderungen, die sich durch die Anwendung des Steuerprogramms an den Ausgängen der Steuerung ergeben, werden über die Socketverbindung und die C-Schnittstelle an den Simulator zurück übertragen. Die Aktoren im Modell werden durch diese Ereignisse von Außen gesteuert und bewegen die Paletten entsprechend der Hardware einer realen Anlage (Abb 22).

Zur Animation wird eine Netzwerkverbindung zwischen Simulator und ROMAN aufgebaut[8]. Über diese werden alle Ereignisse übermittelt, die sichtbare Veränderungen im Modell bewirken. In der Version 3.1 von SIMPLE++ werden durch die Bewegung von Paletten keine Ereignisse generiert. Ein „Generator“ erzeugt in einstellbaren Abständen Ereignisse, die bewirken, daß die Positionen aller beweglichen Elemente im ROMAN aktualisiert werden.

Für die verteilte Simulation und Visualisierung existieren verschiedene Ansätze. Zentrales Problem ist neben den Übertragungsformaten und der durch die Bandbreite von Netzwerkverbindungen begrenzten Performance die Synchronisation zwischen den verschiedenen Prozessen (Mehl 94). An der Universität Magdeburg werden zur Zeit intensiv die Möglichkeiten untersucht, die sich durch den vom Verteidigungsministerium der USA freigegebenen Standard der „High Level Achitecture“ (HLA) ergeben (Schulze 98). Die Relevanz dieser Arbeiten wird geprüft und

[8] Die Übertragung der Animationsdaten erfolgt über das DoxNATIVE Protokoll. Dessen Beschreibung findet sich in Abschnitt *4.8.3*

im Projekt berücksichtigt. Für die Implementierungen, die im Projekt RUGAMS durchgeführt wurden, wurden zunächst einfache Protokolle entwickelt und eingesetzt, die den gestellten Anforderungen entsprechen. Die Berücksichtigung weiterreichender Konzepte wie z.B. HLA könnte dazu beitragen, die Tragweite des Projekts RUGAMS zu erhöhen.

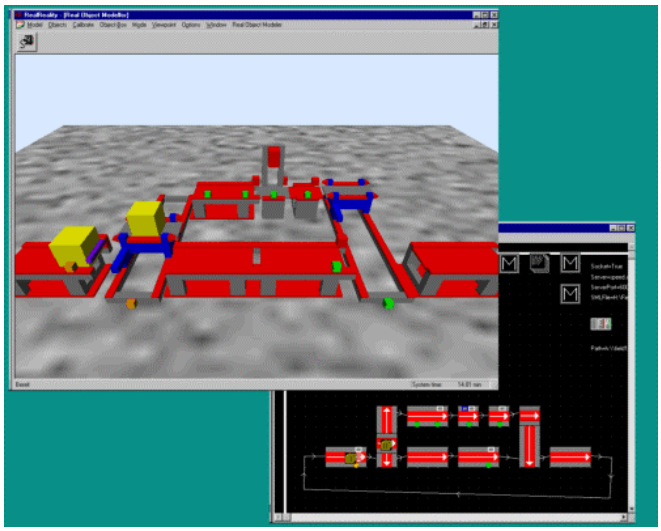

Abb 22: Simulation und Animation mit ROMAN und SIMPLE++

Synchron zur Simulation kann durch die Projektion der 3d-Visualisierung das animierte Modell beobachtet werden. Die Benutzer erhalten so ein sofortiges Feedback über das vorgemachte dynamische Verhalten der Anlage. Die Steuerung kann ohne Zeitverlust iterativ durch Vormachen programmiert und immer wieder verifiziert werden. Das System erzielt damit eine direkte Rückkopplung der Eingaben, wie sie bisher bei anderen Konzepten nicht erreicht wird.

4.6 Verteiltes Modellieren

Die gegenständliche Modellierung setzt voraus, daß sich alle beteiligten Personen zur gleichen Zeit in demselben Raum aufhalten. Alle TeilnehmerInnen können das reale Modell aus vielen Perspektiven betrachten, haben die Möglichkeit, den Kontext des Modellierungsprozesses wahrzunehmen, und können direkt (durch Aktionen mit Datenhandschuh) oder indirekt (durch Diskussionen oder Anweisungen) auf die laufende Modellierung Einfluß nehmen. Um einen Großteil dieser Bandbreite von kommunikativen und interaktiven Einflußmöglichkeiten anderen verteilten TeilnehmerInnen einer Modellierung zu erschließen, sind die Kommunikationstechniken der CSCW[9] von besonderem Interesse.

Die Anforderungen beinhalten die Verbindungen[10] zwischen den TeilnehmerInnen, sowie darauf aufbauend Verfahren, die den Nachrichtenaustausch (die Kommunikation) ermöglichen. Der Nachrichtenaustauch findet über Protokolle, mit deren Hilfe gemeinsam auf Daten (dem Inhalt der gemeinsamen Arbeiten) zugegriffen und diese modifiziert werden können und Computerapplikationen, die den TeilnehmerInnen als Kollaborationswerkzeug zur Verfügung stehen, statt.

[9] Computer Supported Cooperative Work

[10] Gemeint sind hier sowohl Protokoll-Verbindungen des Datenaustausches als auch die physikalischen Beziehungen, in denen die TeilnehmerInnen zueinander stehen. Es kann sich um zwei Personen handeln, die nebeneinanderstehend miteinander reden oder um mehrere Gruppen, die sich an verschiedenen Orten befindend, miteinander austauschen wollen.

4.6.1 Verteilte Konferenzen in einer Real Reality Umgebung

Um eine Konferenz zwischen verschiedenen Personen oder Personengruppen abhalten zu können, sollte eine möglichst reibungslos funktionierende und dem Konferenzinhalt angemessene Kommunikation zwischen den TeilnehmerInnen gewährleistet sein. Eine Konferenz besteht aus zwei oder mehreren TeilnehmerInnen, die über bestimmte Themen kommunizieren.

Die Kommunikation stellt in diesem Zusammenhang keinen Selbstzweck dar, sondern dient dem Austausch von Informationen oder Handlungsanweisungen während der Konferenz. Es wird *über* ein Thema konferiert. Die KonferenzteilnehmerInnen verfolgen bestimmte Ziele. Die Konferenz ist das Forum, in dem Diskussionen stattfinden können.

Bei einer Diskussion handelt es sich um einen Meinungsaustausch, der während einer Konferenz stattfinden kann. Dieser Meinungsaustausch dient idealtypischer Weise der Suche nach der besten Lösung für ein gemeinsames Problem oder der besten Vorgehensweise auf dem Wege zu einer Lösung. Um mögliche Mißverständnisse zu vermeiden oder Stimmungen der DiskussionspartnerInnen wahrzunehmen, ist es von Vorteil, ein oder mehrere Kommunikationsmedien in Anspruch zu nehmen, die für die beteiligten Personen und dem zu diskutierenden Thema angemessene Ausdrucksmöglichkeiten einräumen.

Im Sinne des *Real Reality* Konzepts folgt die Konferenz der Modellierung. In diesem Arbeitspakets wurde ein Konzept entwickelt, das die gegenständliche Modellierung als zentrales Element in eine verteilte Konferenz einbettet. Es sollen möglichst viele Elemente einer natürlichen Interaktion, mit der Einfachheit, Direktheit und Konkretheit, wie sie in einem direkten Dialog auftritt, in die Konferenz übernommen werden.

Das Thema des Konferenzierens in Real Reality Umbgebungen wurde in der Diplomarbeit von Kai Schmudlach ausführlich bearbeitet. Für Informationen, die über diesen Bericht hinausgehen, sei auf diese selbst verwiesen (Schmudlach 98).

4.6.2 Organisatorische Struktur einer verteilten Modellierung

Obgleich eine Kommunikation prinzipiell zwischen allen verschiedenen KooperationsteilnehmerInnen beabsichtigt ist, gibt es hier einen logischen und räumlichen Zentralknoten: das gegenständliche Modell. Hier fallen die Modellierungsdaten an und es ist die Tätigkeit des gegenständlichen Modellierens, die den Inhalt der geplanten Kommunikation bestimmt. Somit erscheint es sinnvoll, in dieses Zentralmodul, dem ROMAN, einen *Kommunikationsserver* zu integrieren, beziehungsweise diesen *Kommunikationsserver* in direkter logischer Nähe des gegenständlichen Modellierers anzusiedeln (Abb 23). Aus der Sicht des *Kommunikationsservers* stellt die Schnittstelle des ROMAN lediglich einen *Modellserver* dar. Die verteilten TeilnehmerInnen können sich über Clientapplikationen, den Telemodellierungsclients, an der verteilten Konferenz beteiligen.

Der *Kommunikationsserver* empfängt alle anfallenden Modellierungsdaten des Modellservers, dem ROMAN, und verteilt diese an die entsprechenden Telemodellierungsclients. Dieser Zwischenschritt bei der Verteilung der Modelldaten führt zu einer Entlastung des Modellservers, da lediglich *ein* "Modellierungsclient", der wiederum die Weiterverteilung übernimmt, von dem Modellserver bedient werden muß. Der Modellserver muß hierzu nicht modifiziert werden und wird zusätzlich von der Verwaltung jeder Form multimedialer Konferenzdaten entlastet. Ein weiterer Vorteil liegt in der Universalität des Konzeptes, das es erlaubt, den *Kommunikationsserver* an jedes beliebige Programm zu binden, das Daten über das TCP/IP Protokoll zur Verfügung stellt.

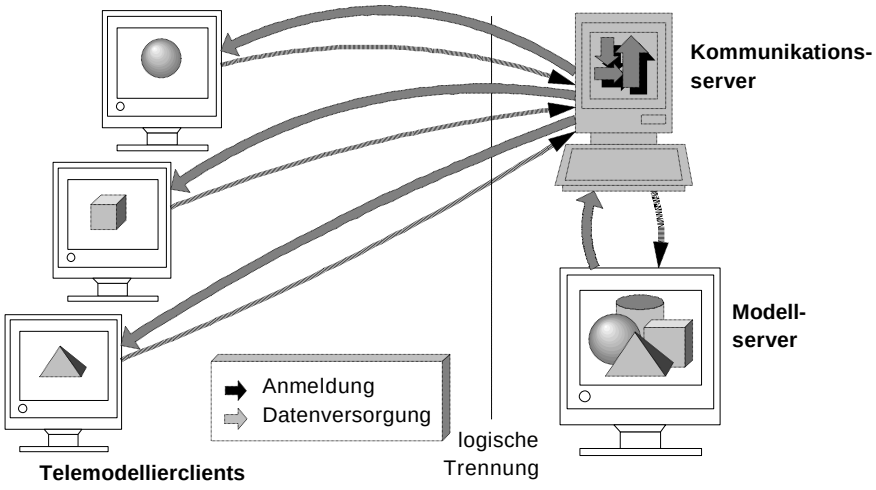

Abb 23: Logische und tatsächliche Positionierung der Module

4.6.3 Medienformen bei verteilten Konferenzsystemen

Die Austauschdaten bei einer Telekooperation können in zwei verschiedene funktionale Gruppen gegliedert werden. Zum einen gibt es *Basisdaten*, die eine Grundlage der Kooperation und den Inhalt der Diskussion bilden. In unserem Fall wird es sich zweckmäßigerweise um die vom ROMAN erzeugten Modelldaten handeln. Bei der anderen Art der Daten handelt es sich um reine *Kommunikationsdaten*, die aus dem Inhalt der Kommunikation *über* die Modellierung bestehen.

Basisdaten

Um es allen TeilnehmerInnen zu ermöglichen, das bestehende Modell während der Modellierung eingehend mit allen relevanten Details, aber auch mit dem nötigen Überblick zu untersuchen, werden jedem Modellierungsclient die vollständigen Modellierungsdaten zur Darstellung der gesamten Szene übermittelt. Hierbei handelt es sich um die Basisdaten. Im Gegensatz zu den originären Modelldaten, hier als *Referenzdaten* (Abb 24) bezeichnet, werden die Modellierungsdaten in den Clientapplikation als *Clientdaten* betitelt. Übertragen auf die Anwendung dieses Verfahrens auf eine gegenständliche Modellierung werden auch die Bezeichnungen *Referenzmodell* und *Clientmodell* verwendet.

Kommunikationsdaten

Neben den Basisdaten bestimmen auch reine Kommunikationsdaten den gegenseitigen Datenaustausch zwischen TeilnehmerInnen an der Tele-Modellierung (Abb 25). Diese werden zur direkt zwischen den Modellierungsclients ausgetauscht und nicht vom Kommunikationsserver verteilt.

Die hier zu realisierende Telekonferenz kann als *same time, different place* charakterisiert werden. Die interagierenden Personen treten gleichzeitig (alle oder nur zum Teil) von verschiedenen Orten aus miteinander in Kontakt. Es werden *synchrone mittelbare* Kommunikationskanäle verwendet. Die in einer Konferenz dieser Art verwendeten Medienformen sollten möglichst entsprechende Eigenschaften aufweisen. Durch die folgenden Medienformen wird dieses Kriterium erfüllt:

- **Videoconferencing**

Videoübertragung (*Videoconferencing*) bietet den KommunikationsteilnehmerInnen eine Ansicht der GesprächspartnerInnen. Dieses datenintensive Medium erlaubt durch die *face-to-face*-Begeg-

nungen die Übermittlung der Gesichtsmimik, die Darstellung von Teilen des Körperausdrucks und eventuell der Übertragung der Gesten der Hände.

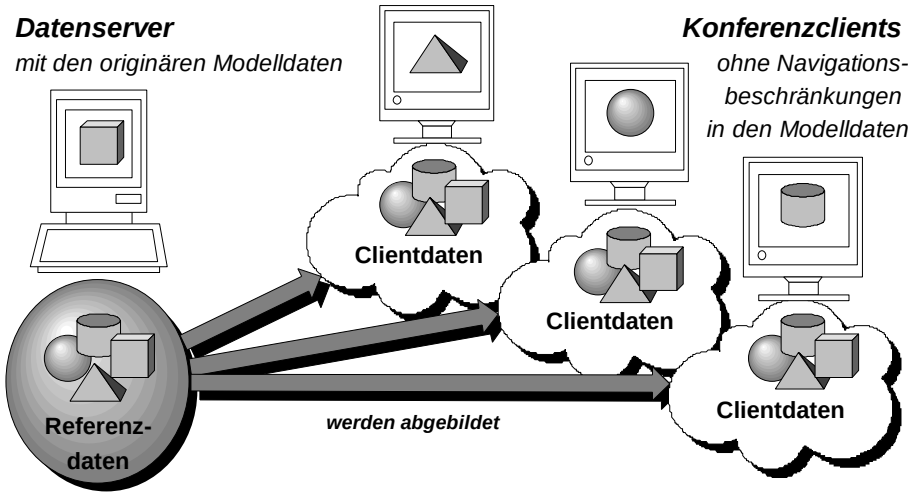

Abb 24: Abbildung der Referenzdaten auf Clients

- **Audioconferencing**

Telefonieren unter Beteiligung von zwei oder mehr Teilnehmern (*Audioconferencing*) stellt eine Möglichkeit dar, Diskussionen zu führen, ohne die Hände zu benutzen oder die Augen von etwas anderem abwenden zu müssen. Diese Eigenschaften macht es zu einem idealen Kommunikationsmedium bei einer gegenständlichen Modellierung. Die visuelle und handelnde Aufmerksamkeit der Modellierer ist auf den Modelliertisch und nicht auf einen Computerbildschirm gerichtet.

- **Chatting**

Textübertragung (*Chatting*) hat die Eigenschaft, eine stärkere Verbindlichkeit als verbale Gespräche zu vermitteln. Eine Textdiskussion ist schriftlich festgehalten und trägt damit die Eigenschaften eines Protokolls bereits in sich. Referenzen und Rückblicke auf bereits Geschriebenes sind ohne weiteres möglich.

- **Distributed Sketching**

Ein verteiltes Zeichenbrett (*Sketchpad*[11]) könnte ein einfaches Werkzeug zur Unterstützung verbaler (Audioconferencing) oder gestenbasierter (Videoconferencing) Erklärungen darstellen. Dieses Medium ersetzt das Papier und den Bleistift, die oft für kleine Skizzen oder beschreibende Illustrationen verwendet werden.

- **Distributed Clipboard**

Eine verteilte Zwischenablage (*Distributed Clipboard*) ermöglicht einen unkomplizierten Datenaustausch ohne Beschränkungen in der Art der verwendeten Daten. Alles, was sich in einer Zwischenablage[12] ablegen läßt, könnte auf diesem Wege auch allen anderen TeilnehmerInnen zu-

[11] Mit *Sketchpad* (engl. sketch: zeichnen, skizzieren; pad: Unterlage) wird eine Applikation bezeichnet, die es erlaubt, kleinere Zeichnungen oder Skizzen anzufertigen. Im Bereich der Telekooperation werden die so erzeugten "Kunstwerke" gleichzeitig von mehreren TeilnehmerInnen bearbeitet und gesehen. Das *Sketchpad* ist eine auf Zeichnungen spezialisierte Form eines Whiteboardsystems.

[12] Viele graphische Betriebsystemoberflächen beinhalten einen auf irgendeine Weise realisierten Datenaustausch zwischen lokalen Applikationen. In der Terminologie der MS Windows GUI wird dieses Verfahren *Zwischenablage* genannt. In der Regel sind mindestens folgende Operationen auf der Zwischenablage möglich: "Ausschneiden"

gänglich gemacht werden. Da eine Zwischenablage eine Datenschnittstelle zwischen unterschiedlichen lokalen Applikationen darstellt, bietet dieses Verfahren den BenutzerInnen eine einfache Einbettung konferenzfremder Anwendungen.

- **Datasharing**

Eine verteilter Datenaustausch *(Datasharing)* bietet mehreren KooperationsteilnehmerInnen die Möglichkeit, die gleichen Daten gemeinsam zu bearbeiten. Die einfachste Form bestünde aus einer Netzfreigabe einer Datei, die wechselseitig von unterschiedlichen Personen geöffnet, verändert und wieder geschlossen werden könnte. Interessantere Variationen erlauben eine gleichzeitige Bearbeitung mit feineren Synchronisationsmechanismen und Abstufungen zwischen reinen Lese- und Schreibzugriffen. Auch bei feinkörnigeren Synchronisationsverfahren kann zwischen gleichzeitiger und wechselseitiger Bearbeitung der Daten unterschieden werden. Besteht für mehrere (oder alle) TeilnehmerInnen die Möglichkeit, die Daten zu lesen und zu verändern, handelt es sich um ein *Shared Memory* oder ein *Whiteboard* System.

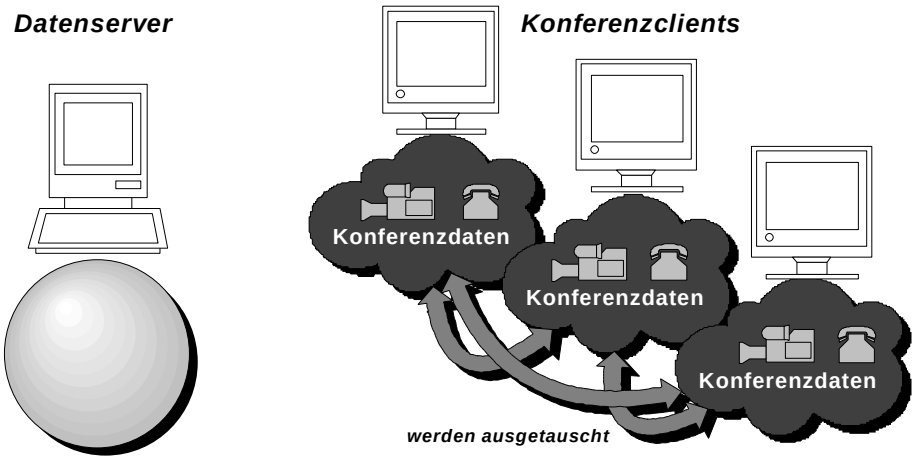

Abb 25: Konferenzdatenaustausch zwischen den Clients

4.6.4 Technische Realisierung

Zur Erkundung der Möglichkeiten des in diesem Projekt vorgestellten Konzeptes zum computerunterstützten Konferieren in einer gegenständliche Modellierumgebung wurde ein Prototyp einer Konferenzsoftware (*TeleROMAN*) mit einem dazugehörigen *Kommunikationsserver* implementiert. Die Aufgabe der Implementation ist es, erste Eindrücke über die entfernte Teilnahme an einer Modellierung zu erhalten und eine Grundlage für weitergehende Überlegungen, Konzepte und Experimente zu schaffen. Die Implementierungsdetails entsprechen der Vorgabe eines lauffähigen funktionalen Prototyps: sie erfüllen den geforderten Zweck, die gewünschte Funktionalität zu demonstrieren, halten sich aber nicht in allen Programm- und Protokollebenen an existierende zukunftsweisende Standards. Es wurde jedoch bei der Implementation in der Programmiersprache C++ darauf geachtet, funktionale Einheiten der Software sauber zu strukturieren und programminterne Schnittstellen für Erweiterungen oder Änderungen offen zu halten.

(*cut*) verschiebt Daten in die Zwischenablage, "Kopieren" (*copy*) stellt eine Kopie der Daten in die Zwischenablage, und "Einfügen" (*paste*) fügt eine Kopie der Daten von der Zwischenablage in die Applikation ein.

4.6.5 Netzprogrammierung

Das hier entwickelte Konferenzkonzept baut auf die Client-Server-Architektur des bestehenden Softwaresystems auf. Die Quelle der Modelldaten steht fest: der *Modellserver*.

DOXDisplay und DOXNative

Die Software des ROMAN besitzt die Fähigkeit, die Geometrie der aktuellen Szene an entfernte Clientapplikationen weiterzureichen. Verwendet werden die proprietären Protokolle *DOXDisplay* und *DOXNative*, die beide in der Version 1 vorliegen. Die Eigenschaften des *DOXDisplay* erlauben lediglich ein Beobachten und Visualisieren der Modellierung und haben keinerlei Konferenzcharakter. Dagegen ist *DOXNative* bidirektional ausgelegt und erlaubt ein aktives Abfragen und Verändern von Zuständen auf dem *Modellserver*. Ferner können Modellierungs-*events* als *DOXNative*-Nachrichten empfangen werden. In diesem Protokoll fehlen jedoch die geometrischen Attribute der *Twin Objects*. Für eine vollständige Integration der gegenständlichen Modellierung in eine Konferenz, ist die Verwendung beider Protokolle nötig, für eine einfache Darstellung der Geometrie des Modells, so wie es im Rahmen der Implementationen in diesem Projekt vorgesehen ist, reichen die Eigenschaften von *DOXDisplay* aus.

Für die Implementierung des Konferenzsystems wurden bei die Kommunikation mit dem ROMAN die im DFG Forschungsprojekt RUGAMS entwickelten C++-Bibliotheken *xSocket*, *Datagramm* und *CommandQueue* verwendet.

Verwendung mehrerer Sockets

Der *Kommunikationsserver* steht mit dem *Modellserver* auf Basis des Protokolls *DOXDisplay* in Verbindung. Die *TeleROMAN*-Clientapplikationen verwenden dieses Protokoll zur Darstellung der Modelldaten. Für andere multimediale Datenarten ist *DOXDisplay* jedoch ungeeignet. Die Struktur dieses Protokolls ist hierfür zu stark an die interne Datenstruktur des *Modellservers* angelehnt. Es ist also nötig, mehrere Datenprotokolle parallel zu verwenden. Da die verwendete Implementierung des *DOXDisplay* in bestimmten Zuständen keinerlei Fehlertoleranz aufweist, wurde für jedes Datenprotokoll ein separater Socket angelegt. Dies gilt sowohl für den *Kommunikationsserver* als auch für den *TeleROMAN*-Client.

Protokollarten und Sockets

Aus den oben genannten Gründen und zur Gewährleistung eines möglichst reibungslosen Nebeneinanders verschiedener Protokollarten mit sehr unterschiedlicher Datendichte, verwendet jede Clientapplikation insgesamt drei Sockets. Über einen TCP-Socket findet die Modelldatenkommunikation mittels *DOXDisplay* statt. Ein weiterer TCP-Socket dient dem Austausch von Synchronisationsnachrichten mittels des *TeleROMAN-Protokolls*. Bei dem dritten Socket handelt es sich um einen UDP-Socket zum Multimediadatenaustausch mittels *DataProtocol*. Nur die ersten beiden Sockets stehen mit dem *Kommunikationsserver* in Verbindung. Der *DataProtocol*-Socket arbeitet verbindungslos mit *Broadcast*-Nachrichten.

Broadcast

Broadcast bezeichnet in diesem Zusammenhang die Adressierung des Ziels der Datenpakete. Ein Datenpaket erreicht in einem LAN[13] bis zu Routergrenzen jeden Rechner, wird aber nur von dem Zielrechner angenommen. Router routen in der Regel keine *Broadcast*-Pakete. *Broadcast*-Nachrichten können von allen erreichbaren Rechnern angenommen werden. Mit diesem Verfahren können Daten an mehrere Rechner gesendet werden, ohne daß es zu einer relevanten zeitlichen Verzögerung bei dem Empfang der Daten zwischen dem ersten und dem letzten Ziel-

[13] Local Area Network

rechner kommt und ohne die gleichen Daten mehrmals für mehrere Adressaten versenden zu müssen.

Kommunikationsserver

Die Programmierung des *Kommunikationsservers* warf das grundlegende Problem der Clientverwaltung auf. Es müssen verschiedene Clientapplikationen mit genau den Daten versorgt werden, die für sie jeweils relevant sind. Der *Kommunikationsserver* muß also zusätzlich zur reinen Verwaltung der angemeldeten *TeleROMAN*-Clients und dem *Modellserver* die angeforderte Protokollart berücksichtigen und darf nur den Clientapplikationen Daten zukommen lassen, die sich für das entsprechende Protokoll angemeldet haben. In der Implementierung wurde dieses Problem durch die Verwendung der Klasse *CDataDispatcher* gelöst. Die Aufgabe dieser, von einer Liste abgeleiteten Klasse ist es, Daten an alle in der Liste verwalteten Clientverbindungen zu versenden. Der *Kommunikationsserver* enthält eine Instanz der Klasse *CDataDispatcher* für jedes unterstützte Protokoll. Die verwendeten Protokolle sind *DOXDisplay* zur Übertragung von Modellgeometrien, *TeleROMANProtocol* zur Synchronisation des Gesamtsystems und *DataProtocol* zum Versenden der Multimediakonferenzdaten. Die Klasse *CClientSocket* ist die Schnittstelle des Kommunikationsservers zu den Clientapplikationen. Eine Instanz dieser Klasse wird von der Klasse *CListenSocket* erzeugt, nachdem eine hereinkommende Verbindung akzeptiert wurde.

Generische Protokollwahl

Die Klasse *CClientSocket* beginnt mit den Protokollverhandlungen und verwendet dann, abhängig von dem Ergebnis der Verhandlung, Methoden der Klassen *CDOXDisplay*, *CTeleROMANProtocol* oder *CDataProtocol* zur Kommunikation mit den Clientapplikationen. Darüber hinaus nimmt *CClientSocket* die Registrierung in einer der oben genannten Instanzen der Klasse *CDataDispatcher* vor. Durch dieses System der getrennten Listen für unterschiedliche Protokollsprachen werden mehrere Probleme gleichzeitig gelöst:

- Es entfällt eine dedizierte Zuordnung bestimmter Protokolleigenschaften zu bestimmten Clientapplikationen sowie deren Verwaltung. Eingehende Daten können einfach an alle anderen in derselben Liste eingetragenen Verbindungen weitergeleitet werden.
- Neben flüchtigen Datenarten (Audio, Video, Text) gibt es bei den Modelldaten einen aktuellen Datenzustand, der zwischen allen TeilnehmerInnen konsistent gehalten werden muß. Diese Eigenschaft der Modelldaten wird besonders dann offensichtlich, wenn TeilnehmerInnen zu einer laufenden Konferenz neu hinzukommen. Der *Kommunikationsserver* hat die Aufgabe, die Änderungen an dem Modelluniversum zu verfolgen und "neuen" TeilnehmerInnen den aktuellen Modellzustand zu übermitteln. Das einfachste Verfahren wäre eine Protokollierung aller von einem bestimmten Anfangszustand ausgehenden Veränderungen. Der Anfangszustand und alle modellbezogenen Aktionen könnten dann an die neuen *TeleROMAN*-Clients übertragen werden. Implementiert wurde eine, in bezug auf diesen Ansatz, verbesserte Lösung: Der *Kommunikationsserver* analysiert die Modellveränderungen und versucht möglichst alle redundanten (zum Beispiel das Erzeugen und spätere Löschen von Objekten) Befehlskombinationen im Modellierungsprotokoll zu entfernen.
- Das System läßt sich einfach um neue Protokollarten erweitern.

4.6.6 Reintegration der Datenarten

Ausgehend von den beschriebenen Kooperationsverfahren können diese durch Kombinationen der verschiedenen Medien zur Übersichtlichkeit und Handhabbarkeit der Kooperation beitragen. Werden Elemente der Modelldaten in Bereiche der Kommunikationsdaten integriert, lösen sich die Grenzen zwischen Basis- und Kommunikationsdaten teilweise auf. Diese Kombinationen steigern die Redundanz der Informationen auf den verschiedenen Kommunikationsmedien und

steigern dadurch die Netzlast. Andererseits erlaubt dieses Konzept den KonferenzteilnehmerInnen, sich ein genaueres Bild vom Modellierungsvorgang zu machen.

Für die Aufgaben dieses Projekts wird eine Auswahl verschiedener Interaktionskopmponenten getroffen, die in folgender Kombination verwendet werden:

- eine Unterstützung von textbasierter Kommunikation in der Art eines Chatclients;
- Vollduplex Audiokonferenzunterstützung zwischen allen KonferenzteilnehmerInnen gleichzeitig;
- Videokonferenzunterstützung für jeweils einen Sender, der von allen TeilnehmerInnen empfangen werden kann;
- Zuordnung von Farbattributen zur Identifikation zu den Benutzerdaten;
- Modellansicht mit freier Perspektivenwahl aller TeilnehmerInnen;
- *Remote Pointer* für jede TeilnehmerIn, der für alle anderen KonferenzpartnerInnen sichtbar ist;
- eine für alle sichtbare *Remote Selection*, die mit den Farbattributen der selektierenden BenutzerIn hervorgehoben ist;
- Aufzeichnungsverfahren der Kommunikationsdaten einer Konferenz;
- eine verteilte globale Zwischenablage zum unkomplizierten Datenaustausch.

Die Anordnung der Komponenten zeigt Abb 26.

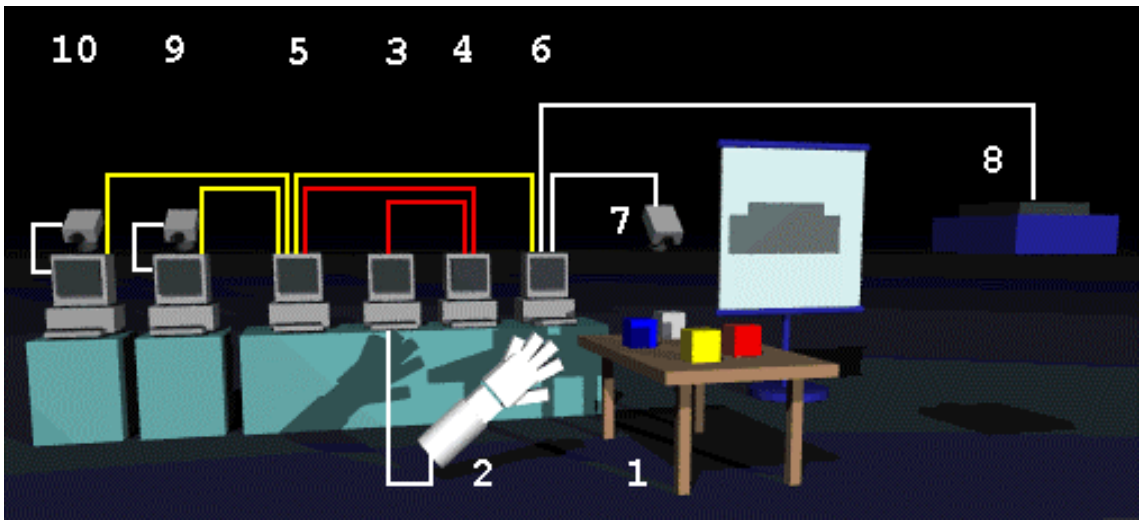

Abb 26: Aufbau des TeleROMAN-Szenarios

An dem Modelltisch (1), auf dem sich die kalibrierten *Real Objects* befinden, arbeiten zwei Modellierende am gegenständlichen Modell. Sie befinden sich in direktem Kontakt miteinander. Die Bewegungen der Hände der Modellierenden werden durch einen Datenhandschuh (2) erfaßt und an den DGS (3) weitergeleitet. Hier werden aus den Bewegungsdaten Gesten und Aktionen extrahiert und an den ROMAN (4) übermittelt. Der ROMAN verwaltet den virtuellen Anteil der Modellierung und verbindet die im DGS erkannten Aktionen mit den *Twin Objects*. Entsprechend der Veränderung der *Real Objects* auf dem Modelliertisch wird die Position, die Orientierung und der Zustand der korrespondierenden *Twin Objects* verändert. Der *Kommunikationsserver* (5) ist als Client an den ROMAN angemeldet und bekommt auf diese Weise alle Veränderungen des virtuellen Modells übermittelt. Sie werden an alle angeschlossenen *TeleROMAN*-Clients (6,9 und 10) weitergeleitet. Einer der *TeleROMAN*-Clients (6) nimmt in diesem Zusammenhang eine Sonderposition als Client für die gegenständlichen Modellierenden ein. Die Kamera (7) dieses Clients ist aus einer Vogelperspektive auf den Modelliertisch gerichtet und erfaßt so die *Real Objects*, sowie die Hände der Modellierenden. Die Bildschirmausgaben (Videobild der KonferenzteilnehmerInnen und die Modellansicht) werden mittels eines Videoprojektors (8) auf eine für

die Modellierenden sichtbare Projektionsfläche geworfen. Die zwei weiteren *TeleROMAN*-Clients (9 und 10) befinden sich nicht mit dem Modelliertisch im Labor, sondern in entfernten Räumen.

Die zahlreichen medialen Ebenen sind sowohl bei der gegenständlichen als auch bei der bildschirmgestützten Teilnahme in die Arbeitsumgebung integriert. Dadurch wird die Medienvielfalt als eine integrierte Umgebung, in der natürliche Kommunikation stattfinden kann, wahrgenommen. Inwieweit die hiermit zur Verfügung stehenden Konfigurationen praxisgerecht sind soll in verschiedenen Versuchen geprüft werden. Eine erste Testanwendung ist die Kooperation zwischen der Universität-Bremen und dem WBK Karlsruhe (Abschnitt *4.6.6*).

4.7 Kooperation zwischen der Universität Bremen und dem WBK Karlsruhe

Das *Institut für Werkzeugmaschinen und Betriebstechnik* (WBK) der Universität Karlsruhe beabsichtigt im Rahmen des Projektes "**Planung komplexer Produktionssysteme durch Techniken der Virtuellen Realität**" mit der am Forschungsinstitut artec durchgeführten Projektreihe zu kooperieren. Beide Institute verfolgen zum Teil sich ergänzende Zielsetzungen mit unterschiedlichen Lösungsansätzen.

Es wurde eine gemeinsame Bibliothek benötigter Geometriedaten von Betriebsmitteln, die als Quelle für eine geometrische und funktionale Modellgenerierung herangezogen werden, angelegt. Die erzeugten Modelle dienen als Basis für verschiedene weitergehende Bearbeitungen im Bereich der Materialflußsimulation, Simulation des Zeitverhaltens, Untersuchung von Kollisionssituationen und zur Generierung von SPS-Programmen.

Als Beispielszenario wurde ein Fertigungssystem mit einer CNC-Fräse und einer NC-Bohrmaschine, die mit Förderbändern materialflußtechnisch verkettet sind, realisiert.

Der Austausch der Daten fand über gemeinsam zugängliche Datenbasen statt, von denen über das Internet neue und geänderte Modelle und Programme bezogen werden konnten (Abb 27). Unterstützend wurden E-Mails und Audiokonferenzen genutzt, um den hohen Abstimmungsbedarf zu decken. Speziell in der Anfangsphase dienten bilaterale Treffen zum Abstecken der Projektziele und zur Vereinbarung der Kooperationsform.

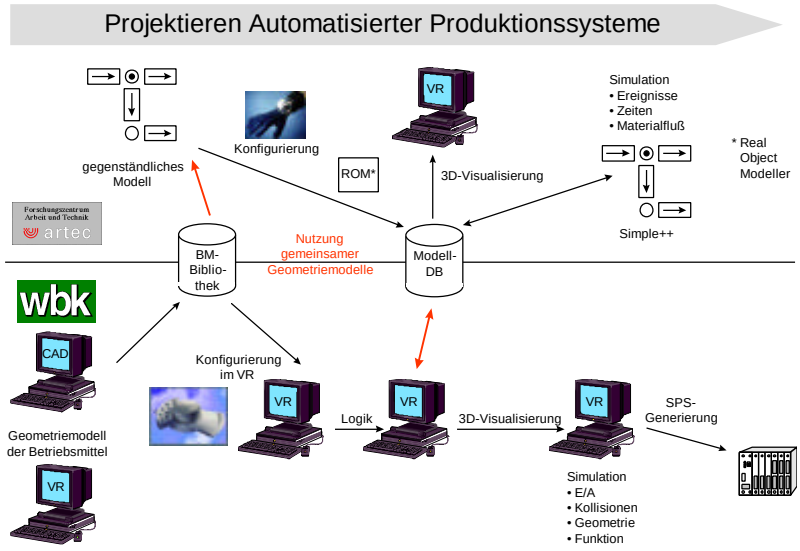

Abb 27: Gemeinsame Datennutzung bei der Kooperation zwischen WBK und artec

Zur Bearbeitung dieser Aufgaben wurde die im vorigen Abschnitt beschriebene Konferenztechnik verwendet. Die gemeinsame Entwicklung stellte ein reales Testszenario dar, das die Möglich-

keiten zur verteilten Arbeit im Zusammenhang mit gegenständlichen und virtuellen Modellen demonstriert hat.

Abb 28 Zeigt Beispiele aus der in Karlsruhe entwickelten Btriebsmittel-Bibliothek. Diese liegen im VRML2 Format zur gemeinsamen Nutzung vor. Mit gegenständlichen Repräsentaten der Objekte können im nächsten Schritt mithilfe der *Real Reality* Technik Szenarien auf dem Modelltisch aufgebaut, diskutiert und variiert werden. Diese Szenarien werden via Internet zum WBK übermittelt. Die dort entwickelte VR-Applikation erlaubt die SPS-Programmierung der gegenständlich konfigurierten Anlage und die Verifikation der Steuerprogramme durch Simulationsläufe am Virtuellen Modell.

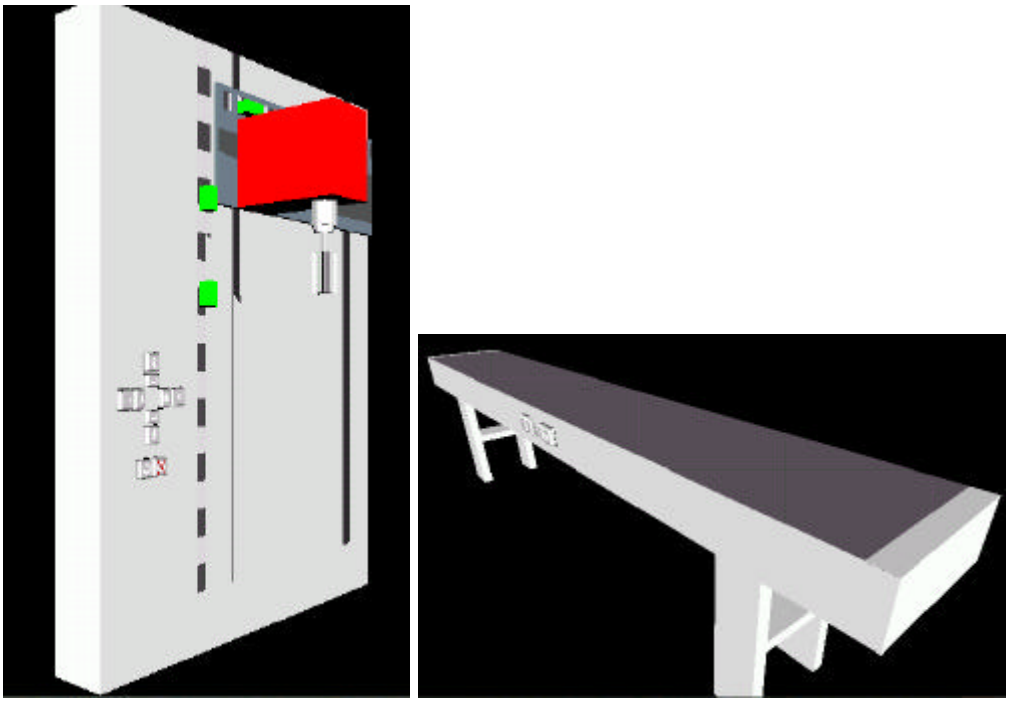

Abb 28: Objekte aus der Karlsruher Betriebsmittel-Bibliothek

Die Steuerprogramme werden ebenfalls in der Modell-DB abgelegt. Der bei artec zur Verfügung stehende Simulator SIMPLE++, der um ein Modul zur SPS-Simulation ergänzt wurde (s.o.), importiert die Modelle und Steuerprogramme aus dieser Datenbasis und simuliert das Materialflußverhalten, um die Performance der geplanten Anlage zu bestimmen.

Das vorgestellte Konzept erlaubt *entfernt*[14] arbeitenden Experten die gemeinsame Entwicklung von Szenarien, die iterative Verfeinerung und die Bewertung mit unterschiedlichen Simulationstools, durch deren Ausprägungen verschiedene Aspekte des Projekts untersucht werden können.

Zur Bearbeitung dieser Aufgaben wurde die im vorigen Abschnitt beschriebene Konferenztechnik verwendet. Die gemeinsame Entwicklung stellte ein reales Testszenario dar, das die Möglichkeiten zur verteilten Arbeit im Zusammenhang mit gegenständlichen und Virtuellen Modellen demonstriert hat.

Folgende Prozeßkette wurde im Rahmen der Kooperation realisiert:

Virtuelle Bausteinbibliothek → Anfertigung entsprechender Modellbausteine aus Fischertechnik → Modellieren der Anlage und Export → *Programmierung der SPS in der Karlsruher virtuellen Umgebung*

[14] Der englische Begrff *remote* ist in dem Bereich der Netzanwendungen fest etabliert und trifft in seinen Konnotationen besser zu als der Begriff *entfernt*. Gemeint sind Applikationen, Zugriffe, Prozesse oder Aktionen, die auf einem anderen Rechner im Netz stattfinden, als jenem, auf dem sie initiiert wurden. Die eigentliche *Entfernung* spielt hier keine Rolle.

(Erstellung von Anweisungslisten) → Verarbeitung der Anweisungslisten im SPS-Simulator und SIMPLE++, Animation im virtuellen und realen Modell → *Anweisungsliste auf die reale Anlage in Karlsruhe übertragen*.

(Kursiv gedruckte Arbeitsschritte werden vom WBK durchgeführt)

4.8 Softwarearchitektur

Auf der Suche nach einem tragfähigen Konzept für die synchrone Modellierung im Realen und Virtuellen, spielt die Architektur der Software eine bedeutende Rolle. Für Entwicklungsarbeiten ist eine große Flexibilität notwendig, da neue Ideen ohne einen großen Overhead implementiert und getestet werden müssen. Dazu ist eine modulare Softwarearchitektur am besten geeignet, weil die Komponenten besonders einfach ausgetauscht werden können. Die Komponenten wkkeln den dynamischen Datenaustausch über Protokolle, die auf Sockets basieren, den eher statischen Datenaustausch über Dateien, welche die Informationen im ASCII-Format spächern, ab.

Zur Laufzeit der Anwendung werden Verbindungen zwischen den verschiedenen Komponenten aufgebaut. Dabei ist die Reihenfolge der Verbindungsherstellung konzeptionell wichtig. Eine Verbindung wird immer von einem Client aufgebaut. Das setzt voraus, das bereits vorher ein Server gestartet wurde. Auf die Metapher der Socket-Verbindungen übertragen heißt das, daß zunächst eine Steckdose (dem Server zugeordnet) vorhanden sein muß, bevor ein Stecker (dem Client zugeordnet) in diese hineingesteckt werden kann.

Die Socket Architektur vereint mehrere Vorteile. Durch das verwendete TCP/IP Protokoll kann die Anwendung weltweit auf verschiedene Rechner verteilt werden. Dabei wird nicht ausgeschlossen, verschiedene Komponenten auf demselben Rechner laufen zu lassen, wodurch der Datentransfer sehr schnell wird oder die Komponenten über ein Lokales Netz z.B. über ein Ethernet zu verbinden. Socketverbindungen sind system- und sprachunabhängig. In unseren Szenarien kommt diese Eigenschaft zum Tragen, da wir in einem anderen DFG-Projkt (EUGABE[15]) einen plattformunabhängig in Java programmierten Simulator für elektronische und pneumatische Schaltungen entwickelt haben.

Für dieses Projekt wurde ein spezielles Protokoll entwickelt, das die Anforderungen der *Real Reality* Modellierung erfüllt. Eine Eigenentwicklung bietet für die Entwicklung große Vorteile, weil die Komplexität an die Bedürfnisse angepaßt, bzw. mit ihnen entwickelt werden kann. Schließlich können aus der entstandenen Spezifikation die Requirements abgelesen werden. Diese können dann als Grundlage zur Auswahl eines Standardprotokolls herangezogen werden. Das von uns entwickelte Protokoll heißt *DOXNative* (s.o.). Es dient zur Übertragung und Veränderung von Modellen sowie zum Austausch von Ereignissen zwischen verschiedenen Anwendungen, dem Event-Mapping.

In den folgenden Abschnitten wird die Implementierung und Verknüpfung der im Projekt entwickelten Softwarekomponenten beschrieben.

4.8.1 Verschiedene Eingabetechniken / Multimodale Anwendungen

Im Projekt BREVIE wird zur Zeit an einer Synchronisation zwischen realem und virtuellen Modell gearbeitet, die auf Bilderkennung beruht. Diese Interfacetechnologie bietet den Vorteil, in metalhaltigen Umgebungen arbeiten zu können und daß die Akteure keine Datenhandschuhe tragen müssen. Für statische zweidimensionale Anwendungen im Lowcost Bereich wird ein System für den Einsatz in Berufsschulen zur Serienreife entwickelt. Diese Modellerkennung deckt die Stufen 1 bis 5 (siehe Abb 12 S. 20) ab. Durch den Austausch dieser Stufen soll das System

[15] Erfahrungsorientierte Übergänge zwischen gegenständlichen und abstrakten Modellen technischer Systeme zur beruflichen Qualifizierung (DFG BR 1556/3-2).

auch für die Modellierung produktionstechnischer Systeme in diesem Projekt nutzbar gemacht werden.

Für die Synchronisation von realem und virtuellem Modell sind verschiedene andere Ansätze in unterschiedlichen Entwicklungsstadien vorhanden. Hierzu gehören 3d Bilderkennung, sensorisierte Tische oder Stecktafeln, und getrackte Objekte. Durch das Stufenmodell wird die *Real Reality* Software für diese Technologien geöffnet.

4.8.2 Client- Server Konzept

Seit nunmehr 2 Jahren wird am Real Object Manager (ROMAN) gearbeitet. In dieser Zeit hat sich vieles am Design und der Implementierung geändert. Dabei ist vor allem die Komplexität der Software gestiegen. Der aktuelle Sourcecode besteht aus 217 Dateien.

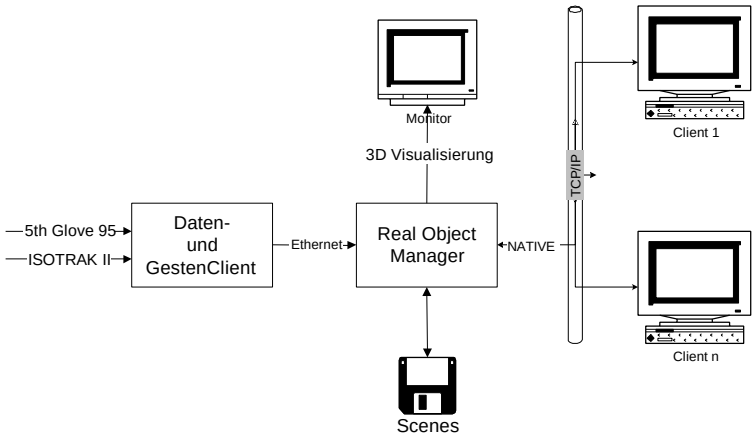

Abb 29: Schematischer Aufbau des ROMANs

Um mit dieser Komplexität umzugehen, wurde die Software für *Real Reality* in mehrere Komponenten aufgeteilt, die über ein Netzwerk verbunden werden. Abb 29 zeigt die Verknüpfung der Komponenten während der Modellierung. Dabei lassen sich drei große Kategorien erkennen:

- Real Object Manager (ROMAN) und
- Daten- und Gestenclient (DGC)
- Netzwerk-Clients.

Der Real Object Manager

Serveranwendung und damit zentrale Komponente in unserer Softwarearchitektur ist der Real Object Manager (ROMAN). Seine Aufgabe ist die Verwaltung des virtuellen Modells als Datenbasis, die Bereitstellung von Methoden zur Abfrage und Veränderung von Modelldaten und die Vermittlung des Datenaustauschs zwischen den Clients. Hierfür stellt der ROMAN ein Protokoll zur Verfügung, über das alle Anwendungen Verbindungen herstellen und Informationen austauschen können. Er verfügt über eine grafische Benutzerschnittstelle, um die Funktionen zur Verwaltung des virtuellen Modells und zur Visualisierung bedienen.

Zur Implementierung des ROMANs wurde die Programmiersprache C++ verwendet. Als Betriebssystem dient Windows-NT. Zur Programmierung der Oberfläche wurde die MFC (Microsoft Foundation Classes) Bibliothek benutzt. Diese bietet die Möglichkeit grafische Bedienungsoberflächen mit einem verhältnismäßig geringen Zeitaufwand zu entwickeln.

Abb 30 zeigt die wesentlichen Komponenten des ROMANs. Die Anwendung ist ein „Multiple Document Interface Architecture“ (MDI) Programm. Alle Fenster sind von einem gemeinsamen

Rahmen umgeben. Jedes Fenster kann in diesem Rahmen formatfüllend dargestellt werden, was für die grafische Visualisierung häufig verwendet wird. Für Objekte und Klassen stehen Browser zur Verfügung, über die die hierarchische Struktur der Objekte eingesehen werden kann. Das Consolenfenster, das links unten geöffnet ist, liefert Statusmeldungen des ROMAN. Der „Animation Controller“ ist ein Player, der die Animation aller aufgezeichneten Pfade ermöglicht. Hier kann der gesamte Modellierungsprozeß nachvollzogen werden. Je nach aktivem Fenster werden die Menüs im Hauptfenster dynamisch angepaßt, um die zugehörigen Einträge bereitzustellen.

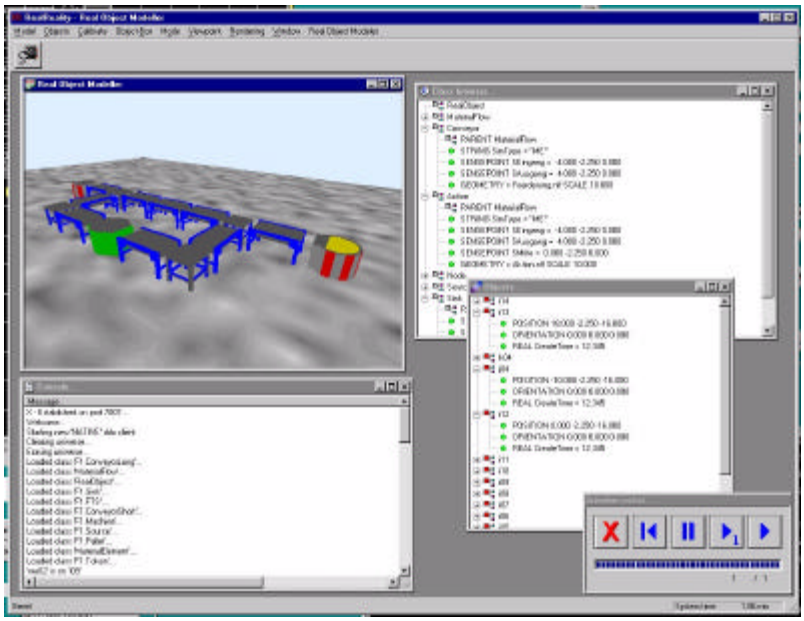

Abb 30: Grafische Oberfläche des ROMAN

Daten- und Gesten Client

Der Daten- und Gestenclient (DGC) synchronisiert das virtuelle mit dem realen Modell. Hierfür verwaltet er auf der einen Seite die Eingabegeräte, auf der anderen wird über eine Netzwerkverbindung mit dem ROMAN das virtuelle Modell verändert. Die Mechanismen zur Verarbeitung der Sensordaten und wie durch diese das virtuelle Modell synchronisiert wird, wurden bereits im Abschnitt *0 Spezifikation der Interaktionen durch eine Gesten-Interaktionsgrammatik* dokumentiert. Im DGC läuft ein Thread zur Interpretation der Gesten-Interaktionsgrammatik. Zur Bedienung verfügt der DGC über ein Bedienfeld, in dem die verwendeten Komponenten konfiguriert werden können (Abb 31).

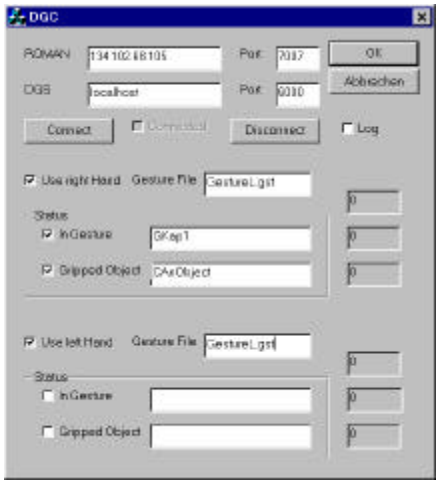

Abb 31: Oberfläche des Daten- und Gesten Clients

Der DGS verfügt nur über ein minimales Weltwissen. Dieses gibt er auf seiner Oberfläche preis. Er informiert über die

- aktuelle Geste, über das
- gegriffene Objekt und über die
- Position der Hand.

Ein DGC kann jeweils einen linken und einen rechten Handschuh mit Trackingsystem verwalten. Da sich beliebig viele DGS mit dem ROMAN verbinden können, ist das Modellieren mit mehreren Akteuren möglich.

Weitere Netzwerk-Clients

Beim ROMAN können sich beliebig viele Clients anmelden und über das NATIVE-Protokoll Informationen austauschen. Für verschiedenen Zwecke wurden bereits Clients entwickelt:

View-Client

Zum verteilten Arbeiten im Netz können auf beliebigen Rechnern View- Clients gestartet und mit dem ROMAN-Server verbunden werden. Diese Laden die Szenenbeschreibung vom ROMAN herunter und werden über alle Änderungen in der Szene informiert. Damit kann die Modellierung an der *Real Reality* Umgebung an verschiedenen Plätzen im Internet beobachtet werden.

Mit den View-Client steht ein Baustein für Conferencing-Systeme, die in Kapitel *4.4.6* beschrieben werden, zur Verfügung. Neben der Visualisierung können Objekte im Modell mit benutzerspezifischen Farben markiert werden, sowie Videobilder, Ton, Text usw. übermittelt werden.

Hilfe-Client

Über Gesten und Interaktionen kann objektspezifische Hilfe angefordert werden. Möglicherweise kann zwischen verschiedenen Darstellungsformen ausgewählt werden. Für die Sprechenden Modelle ist die Möglichkeit vorgesehen, akustische Informationen, die im Windows WAV-Format vorliegen, abzuspielen. Dieser Client ist in C++ implementiert.

Mouse-Client

Dieser Client empfängt Informationen vom ROMAN, die die Maussteuerung betreffen. Diese werden interpretiert und an das Betriebssystem Windows weitergeleitet, wo sie die Anwendungen auf dem Desktop steuern.

Free Keyboard-Client

Selektionsaktionen auf Objekten, die das Attribut „IsACommand“ tragen, werden von diesem Client empfangen. Hier werden die Aktionen, die mit dem Tastendruck verknüpft sind, ausgeführt.

Simulatoren

Durch Simulationen des Modellverhaltens sollen die Modellierenden unterstützt werden. Je nach Anwendungsbereich können verschiedene Simulatoren eingesetzt werden.

- **SIMPLE++**

SIMPLE++ ist ein Standardsimulator für Materialfluß und Logistik. Dieser ist speziell für die Beispielanwendungen in diesem Projekt aus dem Bereich der Fertigungstechnik geeignet. Das Einlesen der Simulationsmodelle geschieht über die Dateischnittstelle des ROMAN. Das für dieses Projekt entwickelte Programm *Simconvert* analysiert die Szene und erzeugt daraus ein SIMPLE++ Model. Der Simulationslauf kann vom ROMAN visualisiert werden. Über die C-Schnittstelle von SIMPLE++ werden die Animationsereignisse an den ROMAN übertragen. Hierfür wird eine hierfür speziell entwickelte DLL von SIMPLE++ geladen. Diese stellt dem

Simulator Funktionen zur Verfügung, mit denen eine Verbindung mit dem ROMAN hergestellt und von dort das Modell verändert werden kann. Das Verhalten des Modells kann in einer dreidimensionalen gerenderten Ansicht beobachtet oder aber auf das Modell projiziert werden (Abb 22 S. 33).

- **DigSim**

DigSim ist ein frei verfügbarer Simulator für digitale Schaltungselemente, dessen Java Sourcecode öffentlich zugänglich ist. Dieser Simulator wurde um die Möglichkeit erweitert, eine Netzwerkverbindung mit dem ROMAN aufzubauen. Das DOXNative-Protokolls stellt die Schnittstelle zwischen der Implementierung des Simulators in Java und dem in C++ implementierten ROMAN über das Netzwerk. Durch die Verwendung von Java ist DigSim plattformunabhängig. Anders als bei der SIMPLE++ Anbindung wird auch das Simulationsmodell über das Netzwerk geladen, wodurch der Dateiaustausch entfällt. Das Modell wird laufend aktualisiert, wenn am realen Modell gearbeitet wird. DigSim ist in der Lage, Benutzeraktionen zu empfangen und z.B. Hilfe zu den Elementen zur Verfügung zu stellen oder über Gesten gestartet und gestoppt zu werden.

- **PneuSim**

PneuSim wurde aus DigSim entwickelt, indem die Bausteinsätze für pneumatische Schaltungen integriert wurden. Berufsschüler können hiermit lernen, Schaltungen im Realen aufzubauen, in der Simulation das Verhalten ihrer Schaltung genauer untersuchen und hinterher auch am realen Modell zu überprüfen. Die Anbindung an den ROMAN ist auf die gleiche Weise wie bei DigSim realisiert.

- **FluidSim**

FluidSim ist ein professioneller Simulator für Pneumatik und Hydraulik, der von der Firma Festo-Didactic vertrieben wird. Er wird in Berufsschulen und Industrie eingesetzt. Strömungen in Rohren und Schläuchen werden über Differentialgleichungen detailliert beschrieben, so daß die simulierten Ergebnisse auf die Praxis übertragbar sind. In Zusammenarbeit mit Festo-Didactic und der Universität Paderborn wurde eine Dateischnittstelle erarbeitet, mit der gegenständlich modellierte Szenarien geladen und experimentiert werden können.

4.8.3 Das DoxNATIVE Protokoll

Die Basis für den Datenaustausch zwischen den Anwendungen stellt das DoxNATIVE Protokoll dar, das im folgenden beschrieben wird.

Design

Das Design des Protokolls entspricht dem Schichtenmodell. Es wurden verschiedene Ebenen mit bestimmter Funktionalität entworfen, die aufeinander basieren. Je höher man in der Hierarchie aufsteigt, desto mehr Funktionalität bieten die Schichten. Insgesamt lassen sich drei Schichten ausmachen:

- Verbindungsschicht (Schicht 1),
- Datenabstraktionsschicht (Schicht 2),
- Protokollschicht (Schicht 3).

Die Verbindungsschicht kapselt Befehle für die Interprozeßkommunikation in einem Modul. Alle Daten werden hier als unstrukturierte Byte-Folgen angesehen.

In Schicht 2 werden die Byte-Arrays zu bestimmten Datentypen strukturiert. Die Datentypen sind unabhängig von einer speziellen Programmiersprache oder Betriebssystem, da sie auf dem ASCII-Standard der Schicht 1 aufsetzen.

Auf der zweiten Schichte setzen die eigentlichen Protokolle des ROMAN auf. Sinn dieser 3. Schicht ist es, Befehle zu definieren, mit denen die Anwendungen ihre Daten austauschen können.

4.8.3.1 Event-Mapping

Der Informationsaustausch über Protokolle läßt zwei Konzepte zu, Pushing und Polling. Clients, die über das DOXNative-Protokoll mit dem ROMAN verbunden sind, können Informationen über das gespeicherte Universum erfragen (Polling), und auf diese Weise eine Kopie von Teilen oder des gesamten Universums herstellen. Wenn sich das Universum jedoch verändert, veraltet die erhaltene Kopie. Es ist sehr ineffizient, ständig die gesamten Universen für die Aktualisierung neu zu Übertragen.

Clients des ROMAN können Ereignisse (Events) auslösen. So kann z.B. der Daten- und Gesten Client durch Gesten und Interaktionen am Modell Events auslösen. Viele dieser Events werden nicht direkt vom ROMAN verarbeitet, sondern an bestimmte andere Clients weitergeleitet.

Durch das Event-Mapping wird bestimmt, welche Clients an welchen Events Interesse haben. Der Übertragungsaufwand über das Netzwerk läßt sich erheblich reduzieren, wenn die Events nicht immer an alle Clients verschickt werden sondern gezielt an die Clients, die diese bestellt haben. Es wird auch ausgeschlossen, daß Clients unbekannte Events erhalten, die sie nicht verarbeiten können.

4.8.4 Erfüllung der Arbeitsaufgaben

Die hier beschriebene Software stellt die Basis der im dritten Projektjahr durchzuführenden Untersuchungen dar. Alle durch Arbeitspakete geforderten Funktionalitäten konnten durch diese Implementierung erfolgreich umgesetzt werden. Die Module können jederzeit bei artec vorgeführt werden und dienen als Grundlage für die Softwareentwicklung in weiteren Projekten.

5. Arbeitserfahrungen

Das *Real Reality* Konzept und dessen Realisierung wurden einer breiten Öffentlichkeit vorgestellt und mit Vertretern verschiedener Disziplinen diskutiert. Im folgenden werden die wichtigsten Aussagen zusammengefaßt.

Allgemein stößt unser Anliegen, die Anschaulichkeit und Begreifbarkeit zu erhöhen, auf positive Resonanz. Sie liegt damit auch im Trend des Visual Engineering und der VR, geht aber in Hinblick auf das haptische Feedback und die sich aus der Realität ergebenden Restriktionen weiter. Von Vertretern der VR-Technologie wird der Aufwand der Modellfertigung und die Einschränkungen beim Umgang mit sehr komplexen Systemen angemerkt. Dem stehen die Vorteile ungehinderter Kommunikation, intuitiver Benutzbarkeit der Gegenstände und der im Vergleich zu VR-Technologien um ein Vielfaches geringere Hardwareanforderungen gegenüber.

Die Programmierung durch Vormachen, speziell die Erzeugung von Steuerprogrammen stieß anfänglich häufig auf Unverständnis. Die Restriktionen durch die Eingabesematik, die für eine kontrollierte Programmgenerierung unerläßlich ist, schienen der Idee des freien Vormachens zu widersprechen. Die Vermittlung des Vormachens als symbolische Programmiersprache brachte besseres Verständnis bei den Zuhörern. Die Besonderheit liegt bei dieser in Unmittelbarkeit der Symbole im Zusammenhang mit den situativen Komponenten der Aufgabe und des Modells. Bei dieser Darstellung wird die Programmierung am Modell als Erleichterung gesehen und könne zur schnellen und fehlerfreien Programmierung von Anlagen beitragen. Dazu wird auch durch die Einbindung von Simulationsläufen als Beitrag gesehen.

Bisher wurden sämtliche Einreichnungen von Proposals für Tagungen im Bereich Multimedia, HCI, Simulation und Pädagogik angenommen und führten zu zahlreichen Präsentationen und Veröffentlichungen des Konzepts (siehe Anhang A).

Die Akquisition eines ESPRIT-Projekts, das die gegenständliche Modellierung von Pneumatikschaltungen und deren parallele Simulation zum Inhalt hat, kann ebenfalls als positives Feedback auf die Idee der synchronen gegenständlichen und virtuellen Modelle gesehen werden.

Mit der vorhandenen Hardware wurden umfangreiche Erfahrungen gesammelt. Die vorhandenen Datenhandschuhe vom Typ 5th GLOVE ´95 von Fifth Dimension Technologies sind, neben dem PowerGlove von Nintendo (ca. $139,-), mit einem Preis von etwa DM 1200,- mit großem Abstand die günstigsten Produkte auf dem Markt. Der geringe Preis wird jedoch mit Mängeln der Qualität bezahlt, bereits mehrfach waren Reparaturen erforderlich. Die drei vorhandenen Handschuhe weisen untereinander sehr breite Streuungen der Meßwerte auf, so daß die Gestenerkennung mit jedem einzeln trainiert werden muß. Während zwar linke und rechte Handschuhe verfügbar sind, gibt es nur eine Größe (One fits most). Dies führt bei den Benutzern im artec zu überstehenden Fingerspitzen. Daraus resultieren Probleme beim Greifen kleiner Gegenstände, zumal die in die Finger eingearbeiteten Lichtleiterschlaufen relativ steif sind. Eine mit Drucksensoren ausgestattete Version ist nicht verfügbar.

Entgegen unserer Erwartung, daß die Handschuhe während der Projektlaufzeit billiger und besser würden, gab es kaum Veränderungen auf dem Markt, obwohl das schnelle Wachstum der VR-Branche dieses hätte erwarten lassen. Der Grund wird darin vermutet, daß VR-Anwendungen immer noch eine FuE-Domäne sind, was Preise zwischen 10 und 26 TDM erlaubt. In diesem Preissegment sind

- der CyberGlove von VTi mit $9.800,- bis $14.500,-,
- der TubGlove der TU-Berlin mit etwa DM 10.000,-
- und der HumanGlove von Humanware Pisa (I) mit ebenfalls etwa DM 10.000,- die Hauptvertreter.

Für den TubGlove wird eine mit Drucksensoren an Fingern und Handflächen ausgestattete Version angeboten.

Die Konsequenz für die kurzfristige hardwareseitige Weiterentwicklung der *Real Reality* Technologie ist die Konzentration auf Entwicklungen, die ohne Datenhandschuh auskommen. Ein interessanter Ansatz hierzu ist der in diesem Bericht vorgestellte Grasptracker. Auch die Kameragestützte Griffverfolgung bietet ein interessantes Entwicklungspotential (Rigoll 97).

Das im Projekt RUGAMS eingesetzte Trackingsystem Polhemus Isotrack II erwies sich im Gegensatz zu den Handschuhen als sehr robust und zuverlässig. Größtes Manko bleibt die Empfindlichkeit gegen Metallgegenstände in der Umgebung. Durch die Kalibrierung des Systems kann der Einfluß von unbewegten Metallobjekten größtenteils eliminiert werden und die Ungenauigkeit bei normalen Räumen auf einem Holztisch unter 5 mm gehalten werden. Zusammen mit der hohen Wiederholgenauigkeit ist das System für die gegenständliche Modellierung gut geeignet.

Neu auf dem Markt ist ein opto- akustisch- accelerativ arbeitendes System, das Intersense IS-600, das für etwa DM 22.000,- in Deutschland vertrieben wird. Die für accelerative Systeme charakteristische Drift wird bei diesem System durch optoakustisch arbeitende drahtlose Transponder-Beacons (SonicDisks) korrigiert. Auch wenn diese vorübergehend verdeckt sind, werden Positions- und Orientierungsdaten geliefert. Problematisch für den Einsatz an einem Datenhandschuh ist der relativ große Beschleunigungssensor (I-Cube), der nicht, wie oben vorgeschlagen, auf der Spitze des Zeigefingers getragen werden kann.

Ein Großteil der Entwicklungen im Projekt RUGAMS lag im Softwarebereich. Das virtuelle Abbild der Realen Szene wurde an die neuen Bedürfnisse (z.B. hierarchische Modelle) angepaßt. Die gesamte Software wurde auf eine neue Basis gestellt.

Das zuvor eingesetzte World-Tool-Kit wird zugunsten einer höheren Flexibilität und der Lizenzfreien Weitergabe durch eine selbstentwickelte Datenbasis und das als Freeware erhältliche ActiveX-Control des Cosmo-Players ersetzt.

Die Software wurde vollständig modularisiert und verwendet nun auf lokaler Ebene dynamische Bibliotheken sog. Plugins, und bei verteiltem Einsatz einen Netzwerkclient, der ebenfalls Plugins laden kann. Erst durch diese Anpassung wird die Software so flexibel, das die verschiedenen Einsatzsituationen ohne Änderungen im Sourcecode bewältigt werden können.

Durch die Modularisierung kann auch eine Stabilitätssteigerung erreicht werden. Programmabstürze sind durch die Einführung definierter und dokumentierter Schnittstellen selten geworden.

Neben Ergebnissen zu den gestellten Fragen sind zahlreiche neue Fragestellungen aufgetreten, an denen wir in Zukunft arbeiten werden. Diese reichen von der Konfiguration von Hard- und Software über Einsatzmöglichkeiten, Integrationskonzepte für Simulation bis hin zur Untersuchung von Vermarktungspotentialen.

6. Literatur

artec-paper 56 (1998): V. Brauer, W. Bruns, K. Schäfer: Rechnergestützte Übergänge zwischen gegenständlichen und abstrakten Modellen produktionstechnischer Systeme. Erster und zweiter Zwischenbericht zum DFG Forschungsprojekt RUGAMS. artec, Universität Bremen.

Aspern, J. (1993): SPS-Softwareentwicklung mit Petri-Netzen. Hüthing Verlag, Heidelberg.

Brauer, V. (1994): Feature-basierte Erkennung dynamischer Gesten mit einem Datenhandschuh. Diplomarbeit, Universität Bremen.

Cypher, E. (Ed.) (1994): Watch What I Do - Programming by Demonstration. MIT Press, Cambridge, Massachusetts.

DIN/EN 61131-3 (1994): Speicherprogrammierbare Steuerungen. Teil 3: Programmiersprachen. (IEC 1131-3) DIN im Beuth Verlag, Berlin.

Drews, P. & Weyrich M. (1997): Produktionsplanung und Anlagensimulation mit Methoden der „Virtual Reality“. Industrie Management 13 (1), S. 18-22.

Fishwick, P. A. (1995): Simulation model design and execution: Building digital worlds. Prentice Hall Inc., Englewood Cliffs, New Jersey.

Hoffmann, F. (1995): Der TU-Berlin Sensorhandschuh (TUB-SensorGlove). Technische Beschreibung. TU Berlin, (http://pdv.cs.tu-berlin.de/forschung/SensorGlove2_dt.html).

Hornecker, E; Schäfer, K. (1999): Software Ergonomie ´99, U. Arend, E. Eberleh, K. Pitschke (Hrsg.), Teubner, Stuttgart.

Kämper, S. (1991): PEGROS. Ein Konzept zur Entwicklung eines graphischen objektorientierten Modellbildungs- und Simulationswerkzeugs auf der Basis von Petri-Netzen. Verlag Dr. Kovac, Hamburg.

Krauth, J. (1991): Comparison 2, Flexible Assembly System. EUROSIM Simulation News 1, p. 28, March 1991.

Krauth, J. (1992): Comparison 2, Flexible Assembly System. EUROSIM Simulation News 2, May 1992.

Krauth, J.; Meyer, R. (1995): Comparisons of Simulation Systems based on a Test Model. OPA, Amsterdam.

Livingston, M.A. (1998): UNC Magnetic Tracker Calibration Research, Computer-augmented Vision Technology. The University of North Carolina at Chapel Hill, College of Arts and Sciences, Department of Computer Science, (http://www.cs.unc.edu/~us/magtrack.html).

MacKenzie, C. L.; Iberall, T. (1994): The Grasping Hand. Elsevier Science Publishers, Amsterdam.

Mehl, H. (1994): Methoden verteilter Simulation. Viehweg, Braunschweig.

Merkler, M.; Wachsmuth, I. (1996): Projekt VIENA. Ein hierarchisches Verfahren zur effizienten Kollisionsvermeidung in einer interaktiven graphischen Entwurfsumgebung. Forschungsnetz Anwendungen der KI in NRW. Arbeitsgruppe Wissensbasierte Systeme. KI-NRW 96-03, Universität Bielefeld.

Reisig, W. (1985): Systementwurf mit Netzen. Springer, Berlin.

Reisner, P. (1981): Formal Grammar and Human Factors. Design of an Interactive Graphics System. In: IEEE Transactions on Software Engineering, Vol. SE-7, No. 2, pp. 229-240, March 81.

Rekimoto, J. (1996): TransVision: A Hand-held Augmented Reality System for Collaborative Design. Virtual Systems and Multi-Media (VSMM)'96.

Rigoll, G.; Kosmala, A.; Eickeler, S. (1997): High Performance Real-Time Gesture Recognition Using Hidden Markov Models. In: Gesture and Sign Language in Human-Computer Interaction. Proceedings of the International Gesture Workshop, Bielefeld, Germany. Wachsmuth, I.; Fröhlich, M. (Eds.). Springer, Berlin.

Schäfer, K. (1995): Objektorientierte Modellierung zur Simulation des Steuerungsverhaltens von modularen Transfersystemen. Diplomarbeit, Universität Bremen

Schmudlach, K. (1998): Computerunterstütztes Konferieren in einer gegenständlichen Modellierumgebung. Diplomarbeit, artec, Universität Bremen.

Schulze, Th.; Klein, U.; Ritter, K.C.; Lorenz, P. u.a. (1998): Web-basierte und verteilte Simulation und Visualisierung. In: Simulation und Visualisierung ´98, Tagung der Otto von Guericke Universität Magdeburg. Lorenz, P.; Preim, B. (Hrsg.). SCS, Delft, Erlangen, S. 19 ff.

Stary, C. (1994): Interaktive Systeme. Softwareentwicklung und Software-Ergonomie. Vieweg, Braunschweig.

Streitz, N. A.; Geißler, J.; Holmer, T. (1998): Roomware for Cooperative Buildings: Integrated Design of Architectural Spaces and Information Spaces. Cooperative Buildings – Integrating Information, Organisation and Architecture. In: Proceedings Of The First International Workshop on Cooperative Buildings (CoBuild´98), Darmstadt, Germany, February 25-26, ´98, Springer, Heidelberg.

Anhang A: Projektbezogene Veröffentlichungen

artec-paper 43 (1996): Bruns, F. W.; Müller, D.: Lernförderliche Übergänge zwischen gegenständlichen und abstrakten Modellen technischer Systeme. artec, Universität Bremen.

artec-paper 56 (1998): V. Brauer, W. Bruns, K. Schäfer: Rechnergestützte Übergänge zwischen gegenständlichen und abstrakten Modellen produktionstechnischer Systeme. Erster und zweiter Zwischenbericht zum DFG Forschungsprojekt RUGAMS. artec, Universität Bremen.

artec-paper 59 (1998): Vom Bildschirm zum Handrad - Computer(be)nutzung nach der Desktop-Metapher - Workshop, 6.-7. Oktober 1997. artec, Universität Bremen.

artec-paper 60 (1998): RUMpv, Rechnergestützte Übergänge zwischen Modellen physikalischer und virtueller Realität, ein studentisches Projekt. artec, Universität Bremen.

artec-paper 67 (1999): K. Schäfer, W. Bruns: Rechnergestützte Übergänge zwischen gegenständlichen und abstrakten Modellen produktionstechnischer Systeme. Dritter Zwischenbericht zum DFG Forschungsprojekt RUGAMS. artec, Universität Bremen (dieses Papier).

Brauer, V. (1996): Simulation Model Design in Physical Environments, ACM SIGGRAPH, Computer Graphics, Vol. 30, No. 4, November 1996.

Brauer, V.; Bruns, F. W.; Schäfer, K. (1997): Eine Synthese aus Real Reality und Virtual Reality mit Anwendungsbeispielen aus der Produktionstechnik. Tagungsband der Virtual World ´97, Köln.

Brauer, V. (1998): A Collaborative Environment for Learning Pneumatics in Real and Virtual Reality. 5th Educational Multimedia Task Force Concertation Meeting, Brussels, 17.-18. June 1998.

Bruns, F. W.; Brauer, V. (1995): Greifendes und begreifendes Modellieren im Realen und Virtuellen. Erschienen im FORWISS-Report zum 7. Workshop Hypermedia & KI, Hannover, Nov. 1995, Hrsg. E. Neugebauer u. S. Wiesener.

Bruns, F. W.; Brauer, V. (1996): Bridging the Gap between Real and Virtual Modeling - A New Approach to Human-Computer Interaction, presented at the 2nd IFIP 5.10 Workshop on Virtual Prototyping, May 4-6 1996, Arlington Tx, USA.

Bruns, F. W. (1996): Grasping, Communicating, Understanding – Connection Reality and Virtuality. In: AI & Society Nr.10, S.6-14.

Bruns, F. W. (1997): Sinnlichkeit in der Technikgestaltung und Technikhandhabung. Ein konstruktiver Ansatz. In: Technik und Subjektivität. Schachtner, C. (Hrsg.). Suhrkamp, Frankfurt.

Bruns, F. W. (1998): Using Real and Virtual Reality in Training Production Engineers. Telematics Applications (TAP) Concertation Meeting, Barcelona, 4.-7. February 1998.

Bruns, F. W. (1998): Bridging Real and Virtual Reality in a Complex Learning Environment. IFIP '98, 15th IFIP World Computer Congress: The Global Information Society on the Way to the Next Millennium. August 31st to September 4th 1998, Vienna and Budapest.

Bruns, F. W. (1998): Integrated Real and Virtual Prototyping. IECON '98, The 24th Annual Conference of the IEEE Industrial Electronics Society. August 31st to September 4th 1998, Aachen, Germany, (Presented by E. Hornecker).

Grund S. (1998): Evaluation of the BREVIE Learning Environment. 6th Educational Multimedia Task Force Concertation Meeting, Brussels, 19.-20. November 1998

Grabsch, N. (1997): Freiformmodellierung im Realen – Entwurf und Implementierung eines Prototypen. Diplomarbeit, Universität Bremen.

Hornecker, E. (1998): Coupling Physical Artifacts and Abstract Representations. ED-Media 98, Freiburg, Germany, March 1998.

Hornecker, E; Schäfer, K. (1999): Software Ergonomie ´99, U. Arend, E. Eberleh, K. Pitschke (Hrsg.). 9. Fachtagung Software-Ergonomie '99 Design von Informationswelten 8. bis 11. März 1999 in Walldorf/Baden, Teubner, Stuttgart.

Karras, U. (1998): Bridging Industrial and Virtual Reality. 6th Educational Multimedia Task Force Concertation Meeting, Brussels, 19.-20. November 1998.

Kusch, G. (1997): Aspekte der Orientierung, des Handelns und der Selbstwahrnehmung in einer Virtual Reality-Testumgebung. Magisterarbeit, Universität Bremen.

Meier, E. (1998): Synchrones Generieren von Modellen in realen und virtuellen Räumen. Diplomarbeit, Universität Bremen.

Müller, D. (1998): Simulation und Erfahrung. Ein Beitrag zur Konzeption und Gestaltung rechnergestützter Simulatoren für die technische Bildung. Dissertation, Universität Bremen.

Robben, B.; Hornecker, E. (1998): Gegenständliche Modelle mit dem Datenhandschuh begreifen – Eine Lernumgebung für den Technikunterricht. In: Claus, V. (Hrsg.): Informatik und Ausbildung 98, GI-Fachtagung, Stuttgart, Springer.

Rügge, I; Robben, B.; Hornecker, E; Bruns, F. W. (Hrsg.)(1998): Arbeiten und Begreifen: Neue Mensch-Maschine-Schnittstellen. Lit Verlag, Münster.

Schäfer, K.; Brauer, V; Bruns, F. W.(1997): A new Approach to Human-Computer Interaction - Synchonous Modeling in Real and Virtual Spaces. Proceedings of the DIS (Designing Interactive Systems), Amsterdam. ACM Press, New York.

Schäfer, K. (1998): Real Reality - Simulationsunterstützung durch gegenständliche Modelle - In: Simulation und Visualisierung ´98, Tagung der Otto von Guericke Universität Magdeburg. Lorenz, P.; Preim, B. (Hrsg.). SCS, Delft, Erlangen.

Schmudlach, K. (1998): Computerunterstütztes Konferieren in einer gegenständlichen Modellierumgebung. Diplomarbeit, artec, Universität Bremen.

Webdarstellung:

http://www.artec.uni-bremen.de/field1/rugams/index.html

Präsentationen:

Demonstrationsstand auf der Hannovermesse 96: Anlagenprogrammierung durch gegenständliches Vormachen, 22.-27. April 1996.

2nd IFIP 5.10 Workshop on Virtual Prototyping, May 4-6 1996, Arlington Tx, USA: Bridging the Gap between Real and Virtual Modeling - A New Approach to Human-Computer Interaction

3Sat Fernsehbeitrag in der Sendung HeiTec, Oktober 96.

Präsentationsstand auf der Forschungsausstellung am Tag der Forschung der Universität Bremen, 26. Oktober 1996.

DIS (Designing Interactive Systems), Amsterdam, 18.-20. Aug. 1997: A new Approach to Human-Computer Interaction; Synchronous Modeling in Real and Virtual Spaces.

Vom Bildschirm zum Handrad - Computer(be)nutzung nach der Desktop-Metapher - Workshop, 6.-7. Oktober 1997.

Tag der Forschung im Forschungszentrum artec am 23. Januar 1998 in Bremen.

Telematics Applications (TAP) Concertation Meeting, Barcelona, 4.-7. February 1998: Using Real and Virtual Reality in Training Production Engineers.

Fachtagung Simulation und Visualisierung, 5.-6. März 1998, Otto-von-Guernicke-Universität Magdeburg: Real Reality - Simulationsunterstützung durch gegenständliche Modelle -.

The Global Information Society on the Way to the Next Millennium. August 31st to September 4th 1998, Vienna and Budapest: Bridging Real and Virtual Reality in a Complex Learning Environment.

IECON 98, The 24th Annual Conference of the IEEE Industrial Electronics Society. August 31st to September 4th 1998, Aachen, Germany, (Presented by E. Hornecker): Integrated Real and Virtual Prototyping.

SIMPLE++ Benutzertreffen, 7.-8. Okt. 1998: Objektorientierte SPS-Programmierung in SIMPLE++.

9. Fachtagung Software-Ergonomie '99 Design von Informationswelten 8. bis 11. März 1999 in Walldorf/Baden: Gegenständliche Modellierung virtueller Informationswelten.

Anhang B: Kooperationspartner

Es findet eine Zusammenarbeit u.a. mit folgenden Instituten und Firmen statt:

- Institut für Werkzeugmaschinen und Betriebstechnik (WBK) der Universität Karlsruhe
- Institut für Wirtschaftsinformatik Berlin (AEDV)
- Technologiezentrum Informatik (TZI) der Universität Bremen
- Fachgebiet Fertigungseinrichtungen der Universität Bremen
- Festo Didactic KG Esslingen
- Bremer Institut für Betriebstechnik und angewandte Arbeitswissenschaft (BIBA)
- Fraunhofer Institut für Produktionstechnik und Automatisierung Stuttgart (IPA)
- AESOP GmbH Stuttgart
- Virtual Presence Ltd Cromwell H, GB
- Superscape Ltd London, GB

Anhang C: Qualifikation wissenschaftlichen Nachwuchses

Diplomarbeiten:

Nicole Grabsch: Freiformmodellierung im Realen – Entwurf und Implementierung eines Prototypen. Universität Bremen, 1997.

Joachin Hinrichs und Claus Holm: Konzeption der Planungssysteme MOPPS und VOPPS auf der Basis gegenständlicher Arbeit und abstrakter Systemplanung. Universität Bremen, 1997.

Eckhard Meier: Synchrones Generieren von Modellen in realen und virtuellen Räumen. Universität Bremen, 1998.

Martin Faust: GRACEland – Ein 3D-Editor und Interpreter für die graph- und regelbasierte Sprache GRACE. Universität Bremen, 1998.

Kai Schmudlach: Computerunterstütztes Konferieren in einer gegenständlichen Modellierumgebung. Universität Bremen, 1998.

Promotionen:

Dagmar Cords: Beteiligungsorientierte Systemanpassung vom Basissystem zur Arbeitsumgebung, Universität Bremen, 1997.

Dieter Müller: Simulation und Erfahrung. Ein Beitrag zur Konzeption und Gestaltung rechnergestützter Simulatoren für die technische Bildung.

Volker Brauer: Gegenständliche Benutzungsschnittstellen (laufend)

Ingrid Rügge: ERBEN - Entwicklung einer realitätsorientierten Benutzungsschnittstelle für Werkstattrechner (laufend).

Kai Schäfer: Programmieren und simulieren produktionstechnischer Einrichtungen durch Vormachen an gegenständlichen Modellen (laufend)

Bernd Robben: Übersetzungen zwischen Notationssystemen - Nach graphischen Benutzungsoberflächen (laufend)

Eva Hornecker: Kooperatives Modellieren mit gegenständlichen Computeschnittstellen (laufend)

Kai Schmudlach: Telekonferieren in gegenständlichen und virtuellen Umgebungen (laufend)

Hauke Ernst: Kooperation in verteilten Real Reality Systemen (laufend)

Magisterarbeiten:

Gabriele Kusch: Aspekte der Orientierung, des Handelns und der Selbstwahrnehmung in einer Virtual Rality-Testumgebung. Universität Bremen, 1997.

Holger Buchmann: Begreifendes und abstraktes Lernen – Entwurf eines Prototypen für empirische Untersuchungen. Universität Bremen, 1998.

Studentische Projekte:

RUMpv: Rechnergestützte Übergänge zwischen Modellen physikalischer und virtueller Realität (artec-paper 60).

Theater der Maschinen: Modellieren im Realen und Virtuellen (laufend).