EINE VERTEILTE ARCHITEKTUR FÜR SYNCHRONES KOOPERATIVES ARBEITEN MIT EINER INTERAKTIVEN STRÖMUNGSSIMULATION

A. Borrmann, P. Wenisch, C. van Treeck

Lehrstuhl für Bauinformatik, Technische Universität München

borrmann@bv.tu-muenchen.de, http://www.inf.bv.tum.de

Kurzfassung: *Der Beitrag stellt eine CORBA-basierte Client-Server-Architektur für die synchrone Zusammenarbeit mehrerer verteilt arbeitender Ingenieure vor. Exemplarisch wird die Einsatz der entwickelten Architektur für ein Collaborative Virtual Environment (CVE) gezeigt, da sich gerade hier die Vorteile einer zentralen Verwaltung des abstrakten Geometriemodells in der damit verbundenen Möglichkeit zeigen, individuelle Views für jeden der Beteiligten anbieten zu können. Es wird die Integration eines Simulationsservers besprochen, der verteilte Fachapplikationen mit Daten einer Strömungssimulation versorgt, gleichzeitig aber auch Modifikationen am Geometriemodell als geänderte Randbedingungen an den CFD-Simulationskern weitergibt. Dadurch wird den Anwendern gemäß dem Computational-Steering-Paradigma ein interaktives Eingreifen in den Simulationsablauf ermöglicht.*

1 Einführung

1.1 Kooperatives Arbeiten

Die Planung von Bauwerken ist gekennzeichnet durch ein hohes Maß an Arbeitsteilung und Spezialisierung. Dies impliziert die Notwendigkeit zur intensiven Kooperation zwischen den an der Planung Beteiligten. Typisch für Planungsteams im Bauwesen ist deren geographische Verteilung; wegen der starken Fragmentierung der Branche in kleine und mittlere Unternehmen arbeiten die Planenden nur selten am gleichen Ort. Im allgemeinen wechseln sich Phasen der asynchronen mit Phasen der synchronen Kooperation ab, wobei synchron hier bedeutet, dass die Beteiligten zur gleichen Zeit (also simultan) miteinander arbeiten und Informationen austauschen, während asynchron heißt, dass die Zusammenarbeit nicht zeitlich abgestimmt ist [1]. Ein typisches Werkzeug der synchronen Kooperation ist eine Telefonkonferenz, ein Beispiel für asynchrone Kommunikation ist eMail-Verkehr.

Das hier vorzustellende Projekt widmet sich zunächst ausschließlich der synchronen Zusammenarbeit, soll aber später als Teil eines Systems fungieren, das auch die asynchrone Kooperation unterstützt.

1.2 Interaktive Strömungssimulationen

1.2.1 Computational Steering

Bei der Planung von modernen Bürogebäuden spielt deren Behaglichkeit bzw. das Komfortempfinden der späteren Nutzer hinsichtlich der Raumluftströmung eine immer größere Rolle. Um die Planung von Heizungs- und Lüftungssystemen und der Ausstattung von Innenräumen zu unterstützen, wird am Lehrstuhl für Bauinformatik unter anderem eine interaktive Strömungssimulation entwickelt [2]. Die Interaktivität bezieht sich dabei vor allem auf das Setzen bzw. Verändern von Randbedingungen zur Laufzeit der Simulation. So ist es möglich, in das Strömungsgebiet neue Hindernisse einzufügen, diese zu verschieben oder ggf. zu entfernen. Gleiches gilt für Ein- und Auslasse, wie Fenster, Türen, Belüftungsschlitze und ähnliches.

Herkömmlich werden große, berechnungsintensive Simulationen nicht interaktiv durchgeführt. Stattdessen werden die Anfangs- und Randbedingungen der Simulation in einer Datei beschrieben (Preprocessing) und dieses dem Simulationsprogramm übergeben. Erst nach dessen Durchlauf kann der Nutzer die Ergebnisse der Simulation auswerten (Postprocessing).

Im Gegensatz dazu folgt die hier vorgestellte Anwendung dem Computational-Steering-Paradigma, das besagt, dass durch eine schnelle Reaktion einer Simulation auf Änderungswünsche des Nutzers an den Randbedingungen kürzere Iterationszyklen für die Optimierung eines Problems möglich werden und dem Anwender ein intuitives Verständnis für die Zusammenhänge zwischen den gesetzten Randbedingungen und dem Ergebnis der Simulation ermöglicht wird [3].

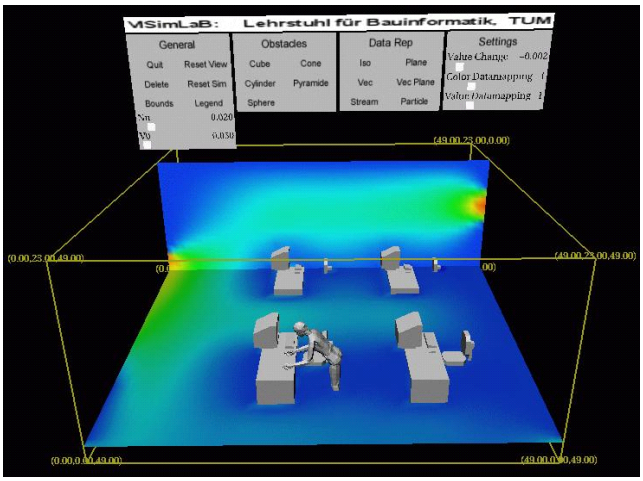

Abb. 1 Interaktive Strömungssimulation in einer VR-Umgebung

Sowohl die interaktiven Operationen wie das Erzeugen, Verschieben und Entfernen von Strömungshindernissen als auch die Visualisierung des Strömungsfeldes in Form von Stromlinien und Isoflächen können in einer Virtuellen Umgebung realisiert werden. Auf diese Weise kann das dreidimensionale Strömungsfeld komfortabel exploriert werden (Abb. 1).

1.3 Lattice-Boltzmann-Simulation

Es hat sich gezeigt, dass für interaktive Strömungssimulationen die Lattice-Boltzmann-Methode besonders geeignet ist [4]. Während bei klassischen Ansätzen die Navier-Stokes-Gleichungen mit Hilfe verschiedener Diskretisierungsmethoden (Finite-Differenzen, Finite-Volumen) gelöst werden (Top-Down-Vorgehen), stellen Lattice-Boltzmann-Verfahren ein Bottom-Up-Vorgehen dar. Hierbei geht man von einem diskreten mikroskopischen System aus, das per Konstruktion Massen- und Impulserhaltung erfüllt, um auf einer makroskopischen Skala hydrodynamisches Verhalten im Sinne der inkompressiblen Navier-Stokes-Gleichungen wiedergeben zu können. Die grundlegende Idee der LB-Methode ist es dabei, die Evolution von Partikeldichte-Verteilungsfunktionen auf einem Berechnungsgitter zu simulieren. Ausgehend von der hinsichtlich des mikroskopischen Geschwindigkeitsraumes diskretisierten Boltzmann-Gleichung mit linearisierten Kollisionsoperator verwendet die Methode ein Finite-Differenzen-Schema zur Diskretisierung in Raum und Zeit, womit ein einfaches raum-zeitliches Entwicklungsschema erhalten wird, dessen algorithmische Umsetzbarkeit für die betrachtete Problemklasse besonders vorteilhaft ist.

Dies betrifft insbesondere das interaktive Ändern von Randbedingungen: Bei Simulationen nach Top-Down-Ansätzen werden in der Regel komplexe Gleichungssysteme gelöst. Während dieses zeitaufwändigen Prozesses ist keine Änderung an den Randbedingungen möglich. Bei der LB-Simulation hingegen werden ausschließlich lokal wirksame Algorithmen verarbeitet, wodurch ein Eingreifen zur Laufzeit der Simulation im Sinne des Computational Steerings möglich wird. Wegen der äußerst schnellen Verarbeitung dieser lokalen Algorithmen ist es zudem vertretbar, ein einfaches kartesisches Berechnungsgitters zu verwenden, was die automatische und effiziente Generierung von Randbedingungen erlaubt [5].

2 Architektur des Systems

2.1 Überblick

Das implementierte System ist als verteilte Mehrbenutzer-Anwendung konzipiert. Jedes Mitglied des Planungsteams arbeitet interaktiv mit dieser Anwendung über eine individuell konfigurierbare Mensch-Maschine-Schnittstelle. Das bedeutet vor allem, dass der View auf den gemeinsamen Planungsgegenstand Innenraum je nach den Bedürfnissen der einzelnen Teilnehmer unabhängig voneinander gewählt werden kann (Abb. 2).

Abb. 3 zeigt die Architektur des konzipierten kollaborativen Computational-Steering-Systems. Alle dort gezeigten Softwarebausteine können auf unterschiedlichen Rechnern laufen. Die zentralen Komponenten bilden zwei Server mit dedizierten Aufgaben: der Geometrie-Server und der Simulationsserver. Der Geometrieserver verwaltet die geometrischen Informationen der Strömungshindernisse und koordiniert die kooperative Arbeit mit Hilfe von Sperren. Modifikationen wie das Hinzufügen, das Entfernen oder das Transformieren eines geometrischen Objekts werden vom Client an den Geometrieserver weitergegeben.

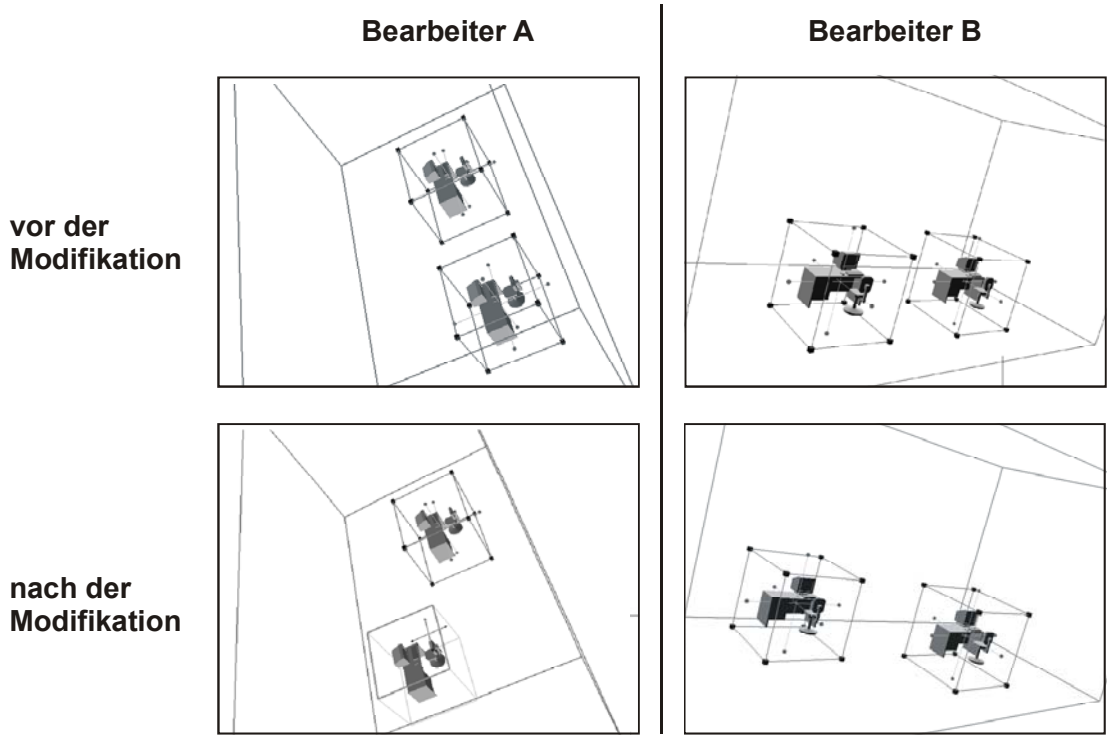

Abb. 2 Verschiedene Views der Bearbeiter

Der Simulationsserver hat die Aufgabe, eine Brücke zwischen dem verteilten System und dem Simulationskern herzustellen. Durch die Trennung zwischen Geometrie- und Simulationsserver ist es leicht möglich, Applikationen anderer Domänen in das System einzubinden, die zwar Zugriff auf die geometrischen Informationen erhalten sollen, aber nicht an den Ergebnisdaten der Strömungssimulation interessiert sind. Der Kern der Simulation wird wegen seinem enormen Bedarf an Rechenleistung in der Regel auf einem Höchstleistungsrechner wie der Hitachi SR 8000 oder einem Rechner-Cluster ausgeführt.

Die Clients dienen der Visualisierung der Strömungshindernisse und der interaktiven Steuerung der Simulation durch den Nutzer. Es kann sich hierbei um einfache Desktop-Anwendungen, aber auch um eine leistungsfähige VR-Umgebung mit stereoskopischer

Projektion handeln. Im letzteren Fall spricht man auch von sog. Collaborative Virtual Environments [7].

Die Client-Server-Kommunikation wird über eine CORBA[1]-Middleware abgewickelt. CORBA ist ein Vertreter der „Distributed Object Computing“-Technologie (DOC) und wurde von der Object Management Group (OMG) standardisiert [6]. Durch Verwendung dieser Technologie sind Prinzipien der Objektorientierung wie Kapselung, Vererbung und Exceptions auch über Rechner- und Betriebssystemgrenzen verfügbar. Dies führt ebenso wie beim Entwurf lokaler Softwaresysteme zu einer Reduzierung der Komplexität, einer leichteren Handhabbarkeit für den Programmierer und dadurch zu besserem, d.h. weniger fehleranfälligen und durch eine streng Modularisierung leichter wiederverwendbaren Code. CORBA hat sich in einer Vielzahl von Projekten zur Unterstützung kooperativer Arbeit als geeignete technologische Basis erwiesen [7] [8] [9].

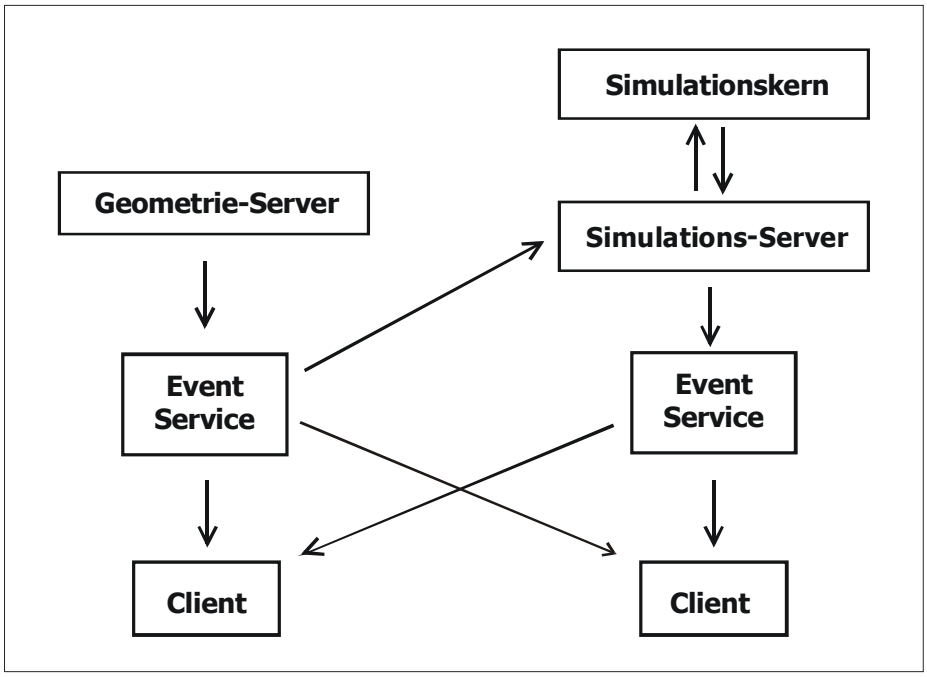

Abb. 3 Architektur des kollaborativen Computational-Steering-Systems

Die OMG hat des weiteren eine Reihe von horizontalen Services standardisiert, darunter einen Event Service, der eine sehr gute Grundlage für das Multicasting von Benachrichtigungen bietet. Der Event Service wird in der hier gezeigten Architektur dazu eingesetzt, Benachrichtigungen von den Servern über Veränderungen der Geometrie bzw. dem Eintreffen neuer Simulationsdaten an alle angeschlossenen Clients weiterzugeben.

[1] Common Object Request Broker Architecture

2.2 Der Geometrie-Server

2.2.1 Geometriemodell als Grundlage der Kooperation

Für die Implementierung eines kooperationsunterstützenden Systems ist es notwendig, ein gemeinsames Modell für die zentral verwalteten Daten zu verwenden. Es ist möglich, dafür die objektorientierte Beschreibung dieser Daten im Sinne eines Produktmodells zu nutzen. Wir haben uns in der ersten Phase des Projekts jedoch für den Austausch von reinen Geometriedaten des triangulierten Oberflächennetzes (Facettenmodell) nach dem vef[2]-Schema entschieden. Diese Entscheidung entspricht zum einen den Anforderungen der Clientapplikation hinsichtlich der Visualisierung und zum anderen dem Prozess der Diskretisierung der Strömungshindernisse. Dieser basiert im wesentlichen darauf, dass die Kuben des Berechnungsgitters auf Schnitt mit der facettierten Oberfläche getestet werden. Um die Effizienz der Diskretisierung zu erhöhen, wurde hierzu ein rekursives, oktalbaum-basiertes Verfahren entwickelt [5].

Von entscheidendem Vorteil gegenüber der Festlegung auf vordefinierte Objekttypen mit parametrisierter Geometrie ist, dass auf diese Weise Strömungshindernisse mit beliebiger Geometrie vorgehalten werden können. Außerdem erscheint eine Klassifizierung der vom Nutzer manipulierbaren Strömungshindernisse wie Schreibtische, Lampen etc. nach dem Paradigma der Objektorientierung nicht sinnvoll, da sie in der Regel über keine gemeinsamen (in diesem Kontext relevanten) Eigenschaften verfügen.

Zu jedem Strömungshindernis wird eine Transformationsmatrix vorgehalten. Diese beschreibt die konkrete Lage, Größe und Ausrichtung des Objektes durch ihre Translations-, Rotations- und Skalierungsanteile. Bei entsprechenden Modifikationen durch einen Nutzer muss nur die neue Transformationsmatrix kommuniziert werden.

2.2.2 Replikative Datenhaltung und Synchronisierung

Um eine schnelle Visualisierung der mitunter sehr komplexen dreidimensionalen Geometrie gewährleisten zu können, werden die Objekte repliziert vorgehalten, d.h. die geometrischen Daten liegen sowohl zentral beim Server als auch bei allen angeschlossenen Clients vor. Die Synchronisierung der replizierten Daten erfolgt mit Hilfe von Benachrichtigungen zu diskreten Zeitpunkten: nach dem Erzeugen, nach dem Löschen und nach der Modifikation eines Objektes.

Hierzu wird der Typed Event Service verwendet, der eine entkoppelte Übermittlung von Benachrichtigungen in Form von Methodenaufrufen erlaubt. Entkoppelt heißt hier, dass der Geometrie-Server nur genau einen Methodenaufruf an den Event Service senden muss, der zudem sehr schnell zurückspringt. Der Event Service übernimmt dann seinerseits die Übermittlung der Benachrichtigung an alle angeschlossenen Clients durch ent-

---

[2] vertex – edge – face

sprechende Methodenaufrufe (Abb. 4). Die Benachrichtigungsschnittstelle ist dabei symmetrisch zur Modifikationsschnittstelle: Zu jeder modifizierenden Methode gibt es genau eine zugehörige Benachrichtigungsmethode.

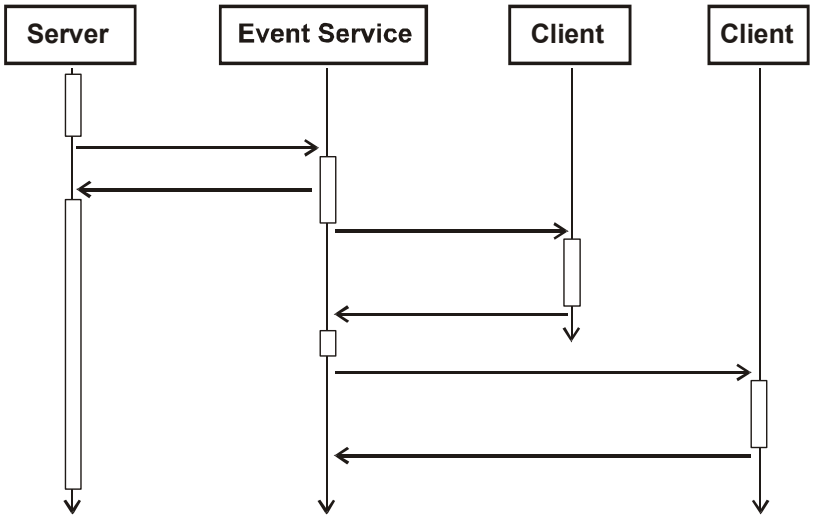

Abb. 4 Entkoppelte Übermittlung von Benachrichtigungen mit dem Event Service

2.2.3 Verteilte Objekte vs. Desktop/Application Sharing

Die Verwendung einer Client-Server-Umgebung mit verteilten Objekten und einer zentralen Datenhaltung war gegenüber technisch simpleren Verfahren wie Desktop bzw. Application Sharing abzuwägen. Beim Application bzw. Desktop Sharing wird in der Regel ein deutlich höheres Datenvolumen zwischen den Rechnern ausgetauscht, da hier der Inhalt aller Pixel eines bestimmten Bildschirmausschnitts übertragen werden muss. Das führt zu einer oftmals unbefriedigend Responsivität und dadurch zu einer geringeren Akzeptanz des Systems bei den Nutzern. Häufig resultieren Akzeptanzprobleme auch daraus, dass sich mehrer Teilnehmer quasi ein Eingabegerät teilen müssen, was zu Effekten wie dem sogenannten „mouse war“ führen kann.

Außerdem ist es als zu stark einschränkend einzustufen, wenn der View der Teilnehmer immer identisch ist. Gerade bei Verwendung von Virtual-Reality-Umgebungen als Clients ist es wichtig, dass die Teilnehmer der kooperativen Session ihren View eigenständig und unabhängig voneinander wählen können. Das kommt vor allem bei Head-Tracking-Verfahren zum Tragen, bei denen Position und Ausrichtung vom Kopfes des Betrachters aufgezeichnet und dazu verwendet werden, einen dem Blickwinkel entsprechenden View zu berechnen. Dadurch wird dem Betrachter die Möglichkeit gegeben, durch einfaches Drehen seines Kopfes Objekte im virtuellen Raum von verschiedenen

Seiten zu betrachten, was die sog. Immersivität[3] steigert. Diese wird drastisch eingeschränkt, wenn die Teilnehmer einer kooperativen Sitzung gezwungen werden, den gleichen Betrachtungswinkel einzunehmen.

Ähnliches gilt für die Verwendung von Stereoprojektionen zum Erzeugen eines dreidimensionalen Bildes. Bei Verwendung von Desktop Sharing ist es im allgemeinen nicht möglich, allen Teilnehmer ein Stereosignal zur Verfügung zu stellen.

2.3 Nebenläufigkeitskontrolle

Um zu verhindern, dass mehre Bearbeiter gleichzeitig ein Objekt manipulieren, wurde eine Nebenläufigkeitskontrolle (engl. concurrency control) in das System integriert, die mit Hilfe von Sperren (engl. locks) implementiert wurde. Sperren sind per Definition Marken dafür, dass ein Prozess exklusiven Zugriff auf eine gemeinsam genutzte Ressource erlangt hat. Prozesse erwerben eine Sperre bevor sie auf die gemeinsam genutzte Ressource zugreifen und geben sie anschließend wieder frei [10].

Die OMG hat zwar einen Concurrency Control Service definiert [12], der die Nebenläufigkeitskontrolle als einen horizontalen (d.h. domänenunabhängigen) Dienst über eine standardisierte Schnittstelle anbietet. Wegen der im allgemeinen engen Verflechtung mit einer Datenbank sind jedoch nur wenige Implementierungen des reinen Concurrency Control Service verfügbar. Wir haben uns daher dafür entschieden, diese Funktionalität selbst zu implementieren und in die applikationsspezifische Schnittstelle zu integrieren.

Eine wesentliche Charakteristik des hier vorgestellten Systems liegt darin, dass es mit verteilten Repliken von Objekten arbeitet, die wie beschrieben über die Propagation von Modifikationen synchronisiert werden. Würde das Setzen bzw. Aufheben von Sperren ebenfalls lediglich propagiert werden, ergäbe sich aus der Latenz der Übermittlung die Möglichkeit, dass ein zweiter Anwender eine Modifikation an dem selben Objekt beginnt. Um dies zu vermeiden, ist es notwendig, dass vor Beginn einer jeden Modifikation explizit eine Sperre angefordert wird. Die Benachrichtigungen über gesetzte und freigegebene Sperren dienen daher nur dazu, dem Anwender zu visualisieren, ob ein Objekt gesperrt oder modifizierbar ist.

2.4 Der Simulationsserver

Der Simulationsserver verbirgt auf der einen Seite den Berechnungskern vollkommen vor dem Programmierer der Anwenderschnittstelle. Auf der anderen Seite verbirgt er die Details einer mehrbenutzerfähigen Computational-Steering-Umgebung vor dem Pro-

[3] Immersivität ist das Maß dafür, wie sehr der Nutzer einer VR-Umgebung die künstlich erzeugte virtuelle Welt als real wahrnimmt.

grammierer des Simulationskerns. Dadurch ist ein leichter Austausch des verwendeten Simulationskerns und eine unabhängige Entwicklung von Nutzerschnittstellen möglich.

Der Simulationsserver lauscht ebenso wie die Clients am Event Service des Geometrieservers und gibt bei einer entsprechenden Benachrichtigung die Änderungen an den Simulationskern weiter. Gleichzeitig empfängt er Simulationsdaten vom Simulationskern in Form von Wertefeldern und sendet diese über den eigenen Event Service an die Visualisierungsclients. Durch die oben beschriebene entkoppelte Kommunikation kann zum einen gewährleistet werden, dass der Simulations-Server nur kurze Zeit für die Übertragung der Simulationsdaten blockiert wird, und zum anderen, dass die Architektur sehr gut skaliert: Für den Simulationsserver ist es unerheblich, wie viele Clients angeschlossen sind, er überträgt die Daten immer nur durch einen einzigen Aufruf. Besondere Bedeutung erlangt dieses Konzept, wenn der Event Service auf einer anderen Maschine läuft als der Simulationskern, da dann voneinander vollkommen unabhängige Aufgaben auf verschiedenen Prozessoren ausgeführt werden, ein nach dem Parallelisierungsparadigma optimaler Zustand. Es bietet sich an, den Simulationsserver in netzwerk-topologischer Nähe zum Simulationskern zu betreiben, da eine geringe Latenz und ein hoher Datendurchsatz bei der Übertragung dazu beitragen können, dass der Simulationsserver nur kurze Zeit mit der Kommunikation beschäftigt ist.

3 Zusammenfassung und Ausblick

3.1 Zusammenfassung

In diesem Beitrag wurde eine sowohl flexible als auch stabile Architektur für ein *Collaborative Computational Steering* - System mit sehr guten Skalierungseigenschaften vorgestellt. Sie verwendet eine Client-Server-Struktur mit einem Simulations- und einem Geometrieserver. Die Kommunikation beruht auf einer CORBA-Middleware und dem Einsatz des Event Services für eine entkoppelte, nachrichtenorientierten Kommunikation.

3.2 Ausblick

Zunächst soll die absolute Unabhängigkeit der Views der einzelnen Bearbeiter relaxiert werden: so soll es auf Wunsch möglich sein, den eigenen View an den eines anderen Teilnehmers zu koppeln. Weiterhin sind kooperationsunterstützende Mechanismen wie eine Audioverbindung und Zeigerhilfsmittel in das System zu integrieren. Ferner ist geplant, ein *explizites* Sperren von Objekten durch einen Nutzer zuzulassen. Auf diese Weise kann ein fließender Übergang zwischen synchronem und asynchronem Arbeiten gewährleistet werden. Für die Zukunft ist vorgesehen, das System auch für domänenübergreifende Kollaboration einzusetzen. Dafür muss die Kopplung des Geometriemodells an ein Produktmodell erwogen werden. In diesem Zusammenhang soll eine Anwendung des Locking-Konzepts auf topologische Container-Strukturen, wie Räume,

Hallen usw. geprüft werden. Grundlage dafür wird der von van Treeck entwickelte Algorithmus für die Extraktion von Raumluftvolumen aus Produktmodellen sein [13].

Literatur

[1] R. Johansen: Groupware: Computer Support for Business Teams. The Free Press - Macmillan, New York. 1988.

[2] P. Wenisch, C. van Treeck, E. Rank: Interactive Indoor Air Flow Analysis using High Performance Computing and Virtual Reality Techniques, Roomvent, Portugal, 2004.

[3] R. V. Liere, J. Mulder and J. V. Wijk: Computational Steering. Future Generation Computer Systems, Vol. 12, nr. 5, 1997.

[4] Krafczyk, M: Gitter-Boltzmann-Methoden. Habilitation. Lehrstuhl für Bauinformatik. TU München, 2001.

[5] P. Wenisch and O. Wenisch: Fast octree-based voxilization of 3D BRep-objects. Technical Report. Lehrstuhl für Bauinformatik. TU München, 2004.

[6] OMG. The Common Object Request Broker: Architecture and Specification. Revision 3.0. OMG Document formal/2004-03-12. Framingham, MA. 2004.

[7] St. Louis Dit Picard, S. Degrand, C. Gransart: A CORBA based Platform as communication support for synchronous collaborative virtual environment. Electronic Proceedings of the ACM Multimedia Conference 2001. Ottawa, Canada. 2001.

[8] Bretschneider, D.: Modellierung rechnergestützter, kooperativer Arbeit in der Tragwerksplanung. Dissertation. Ruhr-Universität Bochum. VDI-Fortschrittsberichte Reihe 4 Nr. 151, VDI-Verlag GmbH. Düsseldorf, 1998.

[9] Hauschild, Th.: Computer Supported Cooperative Work - Applikationen in der Bauwerksplanung auf Basis einer integrierten Bauwerksmodellverwaltung. Dissertation. Bauhaus-Universität Weimar. 2003.

[10] A. Tanenbaum, M. van Steen: Distributed Systems: Principles and Paradigms. Prentice Hall. 2002.

[11] OMG. The CORBA Event Service. Revision 1.1. OMG Document formal/2001-03-01. Framingham, MA. 2001.

[12] OMG. The CORBA Concurrency Service. Revision 1.0. OMG Document formal/200-06-14. Framingham, MA. 2000.

[13] C. van Treeck , E. Rank: Analysis of Building Structure and Topology Based on Graph Theory, Proceeding of the 10th International Conference on Computing in Civil and Building Engineering. Weimar, 2004.