Integrated Computer-Aided Engineering 13 (2006) 1–16 1
IOS Press

Collaborative computational steering: Principles and application in HVAC layout

André Borrmann, Petra Wenisch, Christoph van Treeck and Ernst Rank*
Computational Civil and Enviromental Engineering, Technische Universität München, Lehrstuhl für Bauinformatik, Arcisstrasse 21, 80290 Munich, Germany

Abstract. In this article, we present the Collaborative Computational Steering platform CoCoS. It enables geographically distributed engineers to use steerable simulation and analysis facilities during a collaborative design session. The environment is based on a distributed component architecture composed of a central Collaboration Server, an arbitrary number of Simulation Servers and an arbitrary number of clients. The Collaboration Server manages the shared model consisting of a three-dimensional geometric model and additional semantic data like boundary conditions for a certain simulation. The Simulation Servers provide simulation and analysis data for the engineer's front-end application and can be connected to the platform on demand. By using the explicitly available meta-model, the shared model can be dynamically adapted to the needs of simulation facilities that are not known a-priori. Exemplarily, the utilization of the CoCoS platform for the collaborative layout of a Heating Ventilation Air-Conditioning (HVAC) system is shown. In this context, the implementation and integration of an interactively steerable fluid simulator based on the lattice-Boltzmann method is discussed. This simulator allows obstacles to be inserted into the fluid domain, relocated, and removed from it during the simulation process and the impact of these modifications on the fluid flow to be seen immediately.

1. Introduction

1.1. Integration of design and simulation

Computer-based simulation and analysis tools play a major role in the engineering of buildings today. Among other tools, structural analysis programs are used for dimensioning the structural framework, and, for some special-purpose buildings, fluid simulations are used to design and dimension the air-conditioning system. Unfortunately, in most cases, these very useful programs still form "Isles of Automation" [2]. This involves the manual preparation of their input data and the manual porting back of design modifications to the design tool following simulation.

The integration of design applications and simulation facilities accordingly represents a major challenge to current research in the field of Computational Engineering [36]. The objective is to provide a seamless transition of data between design and simulation tools, so that the engineer barely notices which tool he is currently operating. Over the long term, this will increase the efficiency of today's engineering work flows.

1.2. Computational steering

A key concept on the way to integrating design and analysis is "Computational Steering". The term was introduced by Liere et al. and describes the interactive modification of parameters during the runtime of a simulation [21]. It enables the user to modify the input parameters of the simulated process and study its reaction 'on-the-fly'. The engineer using a steerable simulation obtains an intuitive understanding of its behavior, i.e. the correlation between the modifications and the reactions of the process.

There is indisputably a certain trade-off between interactivity and precision. Nevertheless, the results of the physically simplified interactive simulations are

*Corresponding author. Tel.: +49 89 289 23047; Fax: +49 89 289 25051; E-mail: rank@bv.tum.de.

ISSN 1069-2509/06/$17.00 © 2006 – IOS Press and the author(s). All rights reserved

able to show basic trends and thereby serve as a good platform for fundamental design decisions.

The classical method of conducting simulations and analysis for the design and engineering of buildings tends to be rather laborious: The geometry of the building – or parts of it – is taken from plans or building models and entered into the simulation or analysis tool. In most cases, this has to be done manually. After that, the geometric information is augmented by adding more details representing material parameters and boundary conditions for the simulation.

Afterwards, the simulation is started using the well-prepared input resulting from this 'pre-processing' stage. Depending on the computational effort of the simulation and the resources available, this can take several hours or even days, especially in the case of a three-dimensional fluid simulation.

When the simulation is finished, its results are visualized in order to make them easy to interpret for the engineer. This step is called 'post-processing'. If the engineer is not satisfied with the results, he has to reconfigure the simulation input, conduct the simulation and evaluate the results again. Since these three separate steps (pre-processing, simulation and post-processing) are frequently interrupted by manual data transfers between the different software tools, the optimization cycle often takes a long time and a high manpower investment. In order to reduce the amount of time needed for simulation-based optimization cycles, we propagate the Computational Steering approach.

Over the last decade, the context of computational engineering has shifted. Today very high-performance computing power is affordable even for medium-sized companies. The current efforts in Grid computing will fortify this trend by simplifying access to public and private high-performance facilities around the globe on a pay-per-use basis [10]. These changes, along with new numerical methods and parallelization strategies provide the prerequisites for implementing interactive simulations where parameters can be changed *while* the simulation is running. This enables the engineer to see how a simulation reacts to modifications and to optimize the corresponding parameters intuitively.

1.3. Collaborative engineering

Besides the integration of design and simulation facilities, another major challenge is the development of methods and tools that improve the collaboration between the engineers involved in the design process, regardless of whether they are from the same or from different engineering disciplines. Due to the on-going specialization in all engineering domains, collaboration plays an increasing role in industry today.

This is especially true for the *Architecture Engineering Construction* (AEC) sector, where the industry is heavily fragmented into small and medium-sized companies. In order to face the pressure from the globalized market, these companies will have to jointly form 'Virtual Enterprises' by improving communication and intensifying collaboration. Computer-supported collaborative work (CSCW) will play a crucial role in this process.

An important aspect of designing a collaborative platform is the fact that specialists from different domains make different demands on the application they are using while taking part in the collaborative session. More simple approaches like desktop or application sharing are often unsuitable. To analyze the indoor air flow, for example, an HVAC engineer will use a tool for computational fluid dynamics (CFD), while the interior designer might use a visualization software with advanced rendering capabilities in order to perform an illumination analysis.

1.4. The application scenario: Collaborative HVAC engineering

A fairly characteristic scenario for planning processes involves specialists from different domains coming together to find an optimum solution for a given problem. To support this collaboration we provide them with a virtual space representing the common topic of interest.

Exemplarily, our research project tackles the demands of engineers working together to design and dimension the HVAC system of a building. The common topic of interest here is an office with its surrounding walls including their voids, the HVAC devices and the office equipment inside the room.

In order to achieve optimal convenience for the future users of the office, both the inlets and outlets of an air ventilation system and the radiators have to be placed in the right positions. In addition, the air circulation inside a room or office also depends on the position of windows, doors and obstacles like furniture and partition walls.

While the location and dimensions of the building components are typically fixed at that stage of the design chain, the interior layout including that of the HVAC equipment is still subject to discussion. An interior designer or an architect is usually involved in the decision-making.

1.5. Outline

This paper will introduce CoCoS, a prototype platform for synchronous collaborative engineering in three-dimensional space with the ability to dynamically integrate steerable simulations. The availability of simulation facilities during a shared session increases the relevance of collaborative platforms for engineering practice: a specialist can use the simulation tool of his choice to obtain the data needed for his decisions.

We show how we applied the idea of Computational Steering to Computational Fluid Dynamics (CFD), leading to interactive fluid simulations. In our approach, the interactive modifications are not limited to simple input parameters, such as inflow velocity or outlet pressure, but mainly refer to the position and geometry of obstacles in the fluid domain. During the runtime of the CFD simulation, the engineer can insert new obstacles, change their position or remove them.

Section 2 provides a review of related work in the field of Computer Supported Cooperative Work (CSCW), Collaborative Virtual Environments (CVE) and Computational Steering. We discuss where the different approaches lack certain functionality for engineering practice and how we propose to fill these gaps by combining concepts and techniques from diverse domains in computer science.

Section 3 introduces the CoCoS platform, describes in detail its architecture and its features and takes a closer look at the communication techniques involved. We introduce the CoFluids application which is based on the CoCoS platform and enables the interactive collaborative steering of a CFD simulation.

In Section 4 we describe the simulation kernel, the discretization and parallelization techniques employed and give an overview of the lattice-Boltzmann method, which forms the numerical base of the CFD simulator.

2. Related work

2.1. Computer-supported cooperative work

Johansen provides a time-based classification of cooperative work and distinguishes synchronous and asynchronous collaboration [16]. While synchronous collaboration indicates that the participants work together simultaneously, in asynchronous collaboration there is no explicit time coordination of the cooperative work. In our project we mainly concentrate on the support of synchronous collaboration phases.

The idea of synchronous collaboration between distributed users through so-called multi-user editors appeared in the late 1980s. The first applications were shared text editors [1,27], and shared drawing tools [12,25]. Later, systems for collaboration in two-dimensional CAD systems were developed [33].

Because CoCoS supports synchronous collaborative work in three-dimensional space it can be seen as a Collaborative Virtual Environment (CVE). CVEs are based on Virtual Reality (VR) techniques, and provide an immersive interface to the geographically distributed participants of a collaborative session [5]. The user can navigate through the virtual three-dimensional environment and interact with virtual objects. Usually, the other participants are represented in the virtual world as artificial embodiments, so-called avatars. Thanks to these avatars, an interaction between the geographically dispersed participants becomes possible.

There are three major systems which emerged as a result of the intense research on Collaborative Virtual Environments during the mid-Nineties: MASSIVE, DIVE, and VIVA.

MASSIVE is a distributed virtual reality system and provides facilities to support user interaction and cooperation via text, audio and graphics [13]. The communication architecture is based on processes communicating via typed connections which integrate Remote Procedure Calls (RPCs), attributes and streams in a common context.

The Distributed Interactive Virtual Environment (**DIVE**) is an internet-based multi-user VR system where participants navigate in 3D space and see, meet and interact with other users and applications [14]. It is a loosely-coupled heterogeneous distributed system, which combines audio, document handling and the web with VR. DIVE supports peer-to-peer network communication without a central server and provides a basic 3D user interface in which users are represented as simple avatars.

A CVE dedicated to collaborative work is **VIVA,** a virtual office application that combines different media, such as VR, audio, video, WWW and document handling in order to provide a transparent group work application [30].

The Shared Simple Virtual Environment (**SSVE**) developed by Linebarger et al. enables collaborative construction between dispersed engineers [22]. The SSVE application that has been instrumented for group collaboration experiments is a virtual environment for "design by assembly" prototyping. In the system, products can be collaboratively designed by assembly from

prefabricated component parts, virtually manufactured, simulated in the same construction environment and then modified on the basis of the simulation results.

Despite numerous research efforts, there has been only little impact of shared editors and CVEs on engineering practice so far. In our opinion, this is mainly due to the lack of integration of simulation and analysis tools engineers need to work with in order to make design decisions. The CoCoS platform aims to fill this gap, while applying the results from the CSCW and CVE research projects mentioned before.

In the field of asynchronous collaboration for building design, the SEMPER project is one of the most ambitious undertakings and closely related to the one presented here. SEMPER is a multi-domain, space-based building design environment with integrated performance simulation facilities [20]. Like CoCoS, the system relies on a centrally managed shared object model. In SEMPER, the various analysis modules (energy, air flow, HVAC, thermal comfort, lighting, acoustics) derive their domain object models from the shared model using a homology-based mapping process [20].

2.2. Computational collaboratories

The term 'Collaboratory' was coined by the scientific computing community and refers to a network platform that enables scientists to work remotely in the same virtual laboratory. As in a real laboratory, the 'Collaboratory' provides scientists with certain facilities with which to perform their scientific work. In the case of 'Computational Collaboratories', these are high-performance computing resources and the simulation and analysis software running on them.

Important developments in the area of 'Computational Collaboratories' are **DISCOVER** [38], and **CACTUS**. DISCOVER (Distributed Interactive Steering and Collaborative Visualization EnviRonment) provides a virtual shared workspace for scientists and researchers to steer and collaboratively interact with large parallel and distributed applications by providing collaborative web-based portals. In contrast, CACTUS is a programming framework that enables scientists from diverse disciplines to co-develop code and tools, and to share or interchange modules and expertise.

DISCOVER is the scheme most closely related to the one presented here. Nevertheless, there are some crucial differences. The DISCOVER approach to visualizing steering parameters is mostly generic: the parameters of simulation are displayed as simple text. By contrast, the architecture of CoCoS pays special attention to the visualization of three-dimensional geometry and simulation data. As depicted in Fig. 7, CoCoS client applications are able to display objects and results in a three-dimensional environment.

Another noteworthy undertaking is the **Distributed Laboratories** project [31] that aims at supporting collaboration between researchers or engineers in performing virtual experiments using computational instruments and interactive visualization. The project provides a framework that assists software engineers to develop application-specific interaction and collaboration functionality. The platform was used for the Computational Steering of an atmospheric global transport model which simulates the transport of chemical compounds through the atmosphere.

Forkert et al. describe the Distributed Engineering Framework **TENT**, a component-based framework for the integration of distributed technical applications [9]. It was used as a simulation environment for the analysis of turbo components in gas turbines, for the development of virtual automobile prototypes, for the simulation of the static aeroelastics of an aircraft, and for the simulation of combustion chambers. TENT was prepared for integration in a GRID environment [35], but does not allow for an interactive steering of the simulations involved.

CoViSe (Collaborative Visualization and Simulation Environment) is an extendable, distributed software environment for integrating supercomputer-based simulations, post-processing and visualization functions with cooperative work. Collaborative sessions are supported by CoViSE in a strict *What you see is what I see* (WYSIWIS) manner: All participants see the same screen representations at the same time on their local workstations. A tele-pointer is provided to locate objects in the display area.

The integrated VR renderer, called **COVER,** allows collaborative data analysis within immersive environments. COVER has been applied to the steering of an ongoing simulation in the area of water turbine optimization [32]. During the simulation, boundary conditions can be modified, such as altering the velocity of the incoming water stream by adjusting the angle of the turbine blades. With respect to collaborative features, COVER supports the visualization of avatars for remote participants inside the virtual environment.

2.3. Interactive computational fluid dynamics

The idea of applying interactive techniques to the field of computational fluid dynamics is not new. While

most of the scientific work was dedicated to the interactive exploration of the simulation results by means of interactively movable slice planes, seeding points for particle tracing or adaptable iso-surfaces combined with the utilization of immersive displays, a minority has worked on the realization of fluid simulations where parameters can be modified on-the-fly.

Luksch et al. present **OViD**, a platform for Online Visualization and Computational Steering of parallel and Distributed scientific applications [23]. The project aimed at making existing software packages interactive with only minor modifications to the source code. The system basically consists of a server which provides interfaces for parallel applications and for several visualization tools. OViD works as a layer between the simulation and an arbitrary number of connected visualization tools. For the sake of illustration, a parallelized CFD simulator was connected to this architecture. Unfortunately, Luksch does not point out which kind of interaction with the simulation is supported by the implemented prototype.

Kreylos et al. describe a system that supports the interactive visualization of computational fluid dynamics (CFD) simulations [18]. The system allows a user to position and manipulate visualization primitives, such as isolines and streamlines, during an ongoing simulation process. Furthermore, a user can interactively select and designate regions of the computational mesh for refinement as the simulation progresses and can observe the effects of the refinement on the simulation. In this approach, geometry and boundary conditions cannot be changed on-the-fly.

By contrast, in our approach to interactive fluid simulation the number, the position and dimension of the obstacles inside the fluid domain, as well as the position and dimension of the inlets and outlets are steerable parameters.

3. The CoCoS platform: Architecture & Features

3.1. Overview

The CoCoS platform was designed as a distributed multi-user application. Everyone participating in a collaborative session can work interactively with this application by means of an individually configurable human-machine interface.

This approach has two major advantages: On the one hand, the visual interface can range from desktop monitors to high-end visualization equipments, such as Virtual Reality environments (Fig. 1). On the other hand, it enables each participant's viewing and interaction facilities to remain completely independent of one another. In this way, typical phenomena which are familiar from collaborative environments based on shared desktop or shared application approaches, like "mouse wars", or sickness caused by remotely controlled viewing [4] can be avoided.

The basic architecture of the conceived collaboration platform consists of the central collaboration server, an arbitrary number of simulation servers and an arbitrary number of clients. Figure 2 shows these components and the communication paths between them. Each of the components can be run on different machines.

The modularity of the platform provides a great amount of flexibility: First of all, clients can join and leave the collaborative session at any time. This also holds true for the simulation servers: They can be integrated into the platform when a certain simulation facility is required and released when the work is done. So the collaborating engineers can start to discuss a problem within the collaborative environment without using a simulation.

Moreover, different clients can receive data from different simulation servers, or not receive any simulation data at all, such as a construction engineer who participates in the collaborative session by means of a CoCoS-enabled CAD application.

For the communication between the components of the distributed system, a middleware based on the CORBA standard is utilized. CORBA (Common Object Request Broker Architecture) is a representative of the Distributed Object Computing (DOC) paradigm and was developed and standardized by the Object Management Group [29].

The major advantage of using CORBA is the ability to define semantically rich object-oriented interfaces for the communication between distributed software components. This drastically speeds up the development and distinctively improves the maintainability of such systems.

3.2. The centrally managed model

The shared model is the heart of CoCoS. It is managed by the central collaboration server and reflects the current state of the geometrical and non-geometrical boundary conditions, as well as input and steering parameters of the simulations.

In modern data exchange theory and practice, a so-called "product model" is used to structure the infor-

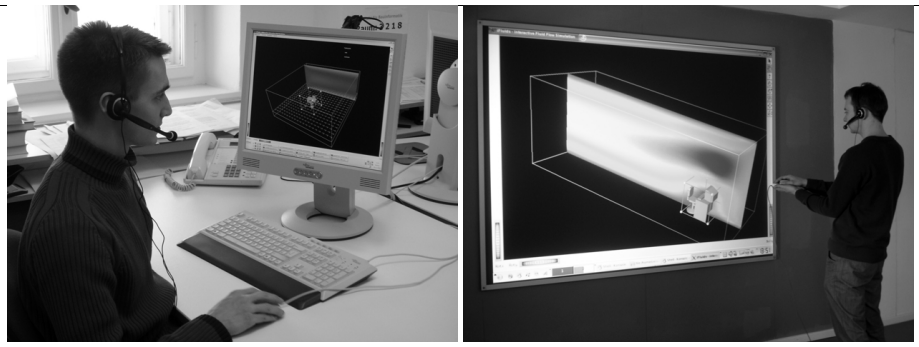

Fig. 1. HVAC engineers taking part in a distributed collaborative session using different human-machine interfaces.

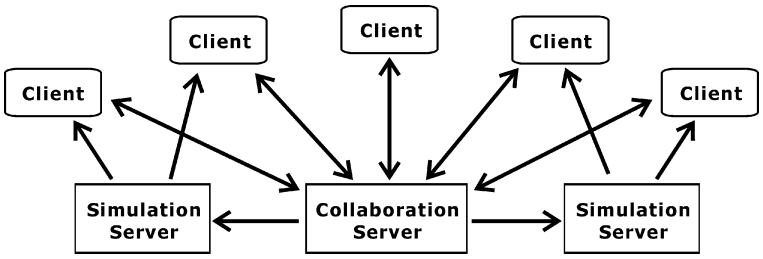

Fig. 2. The CoCoS architecture consists of the central collaboration server, an arbitrary number of simulation servers and an arbitrary number of clients. Not every client has to receive simulation data, and different clients can receive data from different simulation servers.

mation regarding the common subject of interest [8, 15]. In the AEC sector, product models are also referred to as Building Information Models (BIMs). A BIM represents the building by means of its semantic entities, e.g. a wall, a window, a ceiling, and the relations between them, e.g. a window fills the void in a wall. The geometry of the building components is either generated from the objects' attributes and relations ("attribute-driven geometry") or attached to it as an explicit shape description. So the core of a product model is formed by the semantics.

Generating geometry out of attributes has several advantages, for instance when different representations of a physical object are required in 2D plans and 3D renderings. It is important to keep in mind that major parts of the AEC industry are still strongly dependent on 2D plans.However, for fluid simulations and structural analysis, usually a precise three-dimensional description of the shape of the building components is needed. On that account, we follow a different approach for the shared model used in CoCoS than the one known from product modeling. In our approach, the explicit shape description of the physical objects forms the core of the model.

In building engineering, geometry is one of the most important aspects of modeling the product on the computer. Geometry is where the design process starts: functionality, usability and aesthetics are defined by the shape of the product. For simulation tasks like structural analysis or computational fluid dynamics, it is the major source for defining boundaries and boundary conditions.

While it cannot be expected that product models for the various simulation types will become fully standardized in the near future, there is only a limited number of geometry representations used in practice: the constructive solid geometry (CSG) approach based on the concatenation of Boolean operations on simple shapes, and several boundary-representation (B-rep) models including those which represent the surface of a solid by means of plane facets and those which use parameterized patches [24,26].

Nevertheless, besides geometry there is usually some additional data (such as machine and material parameters) to be shared among the participating engineers. A pure geometric model would not fulfill the demands of collaborative engineering. We therefore decided to use a hybrid solution with the ability to attach non-geometric information to geometric objects in a generic way (Fig. 3). The emphasis of the hybrid model is placed on the geometric part: the geometry of an object is the information that all client applications must be able to interpret.

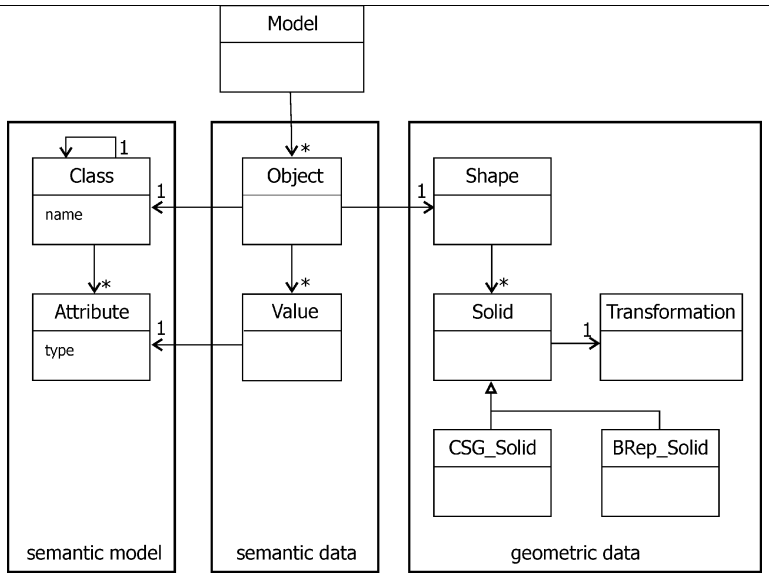

Fig. 3. The centrally managed, shared model combining geometric and semantic data. The explicitly available meta-model on the left describes the structure of the semantic model. The instance of this model, the semantic data of an object, is stored and accessed using the classes in the middle. The geometry and the position of an object is described by means of the classes on the right-hand side.

It is almost impossible to define the data structure of the non-geometric information independent of the concrete application domain. This is especially true when the resulting model has to represent the input and steering parameters for a broad variety of different simulations. Moreover, the ongoing enhancements of the simulation facilities available with respect to the underlying physical models also have to be borne in mind. Therefore, a meta-model depicted on the left hand side of Fig. 3 was integrated in the shared model that makes it possible to define and to query a data model for non-geometric data.

The meta-model supports three major aspects of the object-oriented modeling paradigm [3]: encapsulation by providing classes as containers for attributes, inheritance by making it possible to derive a class from a super-class, and the option of declaring a class as abstract to prevent it from being instantiated.

Utilizing a meta-model to describe the start-up and steering parameters allows for an ad-hoc integration of simulation servers in a plug‘n’play-like manner. As soon as a simulation server becomes available within the CoCoS platform, it sends the model containing the simulation parameters to the collaboration server. The model is then immediately accessible to all clients within the CoCoS platform.

Why not using simple name-value pairs for describing simulation parameters? Name-value pairs do not involve typing of any kind. A user can add any pair regardless of whether its name can be interpreted by the simulation facility or the data type of the value is correct. By contrast, strong typing prevents the user from erroneous inputs. The explicitly available meta-model allows all components of the CoCoS platform to know which parameters exist, and what type they are.

Furthermore, using the meta-model the shared model can be easily adapted to the needs of a specific simulation. When a newly available simulation server joins the CoCoS platform, the model is defined using the meta-model, and it will remain constant during a number of simulations. But, if necessary, it can grow with the simulation’s capabilities without any recompilation. As an example, an instance of the meta model is shown in Fig. 4: the model used by the CFD simulator discussed in Section 4. A CoCoS client can use a generic interface to query a solid with regard to its non-geometric data. Typically, it will display them in a generic manner and provide means for modifications. In most cases, this will be simple tables with the name of an attribute on the left-hand side, and the editable value on the right-hand side (Fig. 5).

Two ways of describing the shape of an object have been incorporated in the shared model. On the one hand, we implemented a boundary representation by defining an object through it’s triangulated surface. This representation is especially useful for objects with complex but static geometries, like dummies or office furniture .

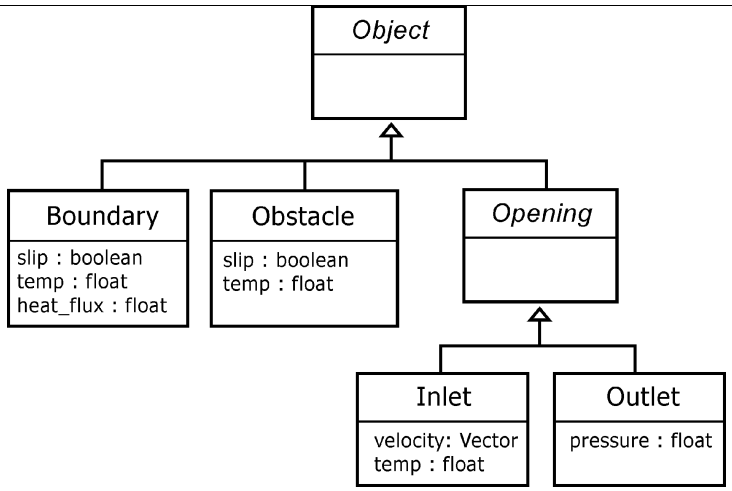

Fig. 4. The model used by the CFD simulator and its clients. It is defined and accessed using the meta-model.

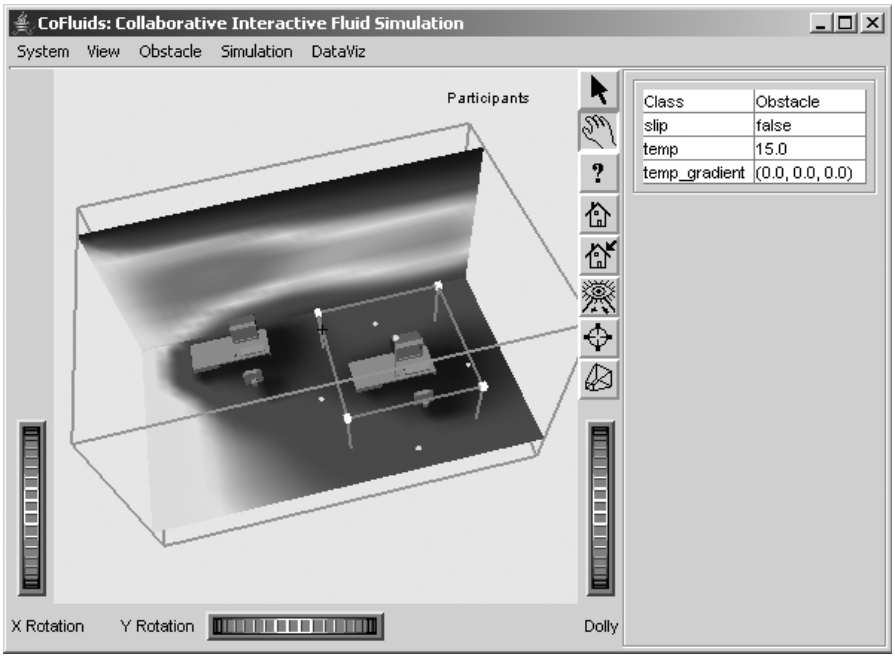

Fig. 5. Non-geometric data that is attached to a geometric object is displayed in a table.

On the other hand, a B-Rep structure is not suitable for inlets or outlets which can be modified during the simulation. The repositioning of a void or hole in a wall would involve a disproportionately time-consuming effort with respect to the recalculation of the triangulated surface. The shared model therefore offers a second geometric representation: the Constructive Solid Geometry (CSG) approach, where solids can be combined using Boolean operators, such as union, intersection or difference.

3.3. Concurrency control

Concurrency control is the activity of arbitrating between potentially conflicting parallel events. In a shared virtual environment, the goal is to maintain consistency of the environmental state that is replicated at each distributed session. This can be done either by preventing inconsistency or by converging to consistency. Concurrency control is a well-researched problem in the fields of distributed operating systems and database

management. Standard textbooks, such as [37] and [4], discuss this subject under various headings such as concurrency control, consistency maintenance, synchronization, or transaction processing.

In collaborative virtual environments, the most common application for concurrency control is to adjudicate between simultaneous attempts to access the same shared object. The need for concurrency control in synchronous collaboration environments was recognized at an early stage and discussed in detail by [11] and [6].

Concurrency control can be classified into optimistic and pessimistic methods. Optimistic algorithms assume that conflicts are relatively rare and undo a change if a conflict is discovered. Pessimistic algorithms prevent conflicts from ever occurring.

An optimistic strategy runs the risk of being inconsistent with other processes until mistakes are discovered and corrected by the serialization algorithm. A pessimistic strategy will introduce delays required to be sure of the global order at the time of the local action, at the expense of the interactivity of the local system. This trade-off between consistency and interactivity has been extensively discussed in the context of distributed virtual environments [34] and groupware [11].

For the CoCoS platform, we implemented a locking-based, centrally managed, pessimistic concurrency control. Because the clients are working with replica of the shared model, this approach is also referred to as "centralized locking with complete replication" [4].

The collaboration server serializes and synchronizes all modification operations by using locks. Before any of the participants can modify an object, the client application he is operating on obtains a lock for that object from the server. When the user starts to move an object, for example, the object is locked for all other users. When the modification operation is finished, the lock is released. Both operations happen implicitly, i.e. the user is not aware of the locking mechanism.

All other participants are notified about the object being locked. Their user interface will provide means to identify the locked objects and to prevent them from being modified. In the prototype application CoFluids, obstacles locked by other users are depicted in a different color and cannot be selected by the user. Because the calls to obtain a lock are synchronized at the server side, it is not possible for two users to access a given object at the same time. So locks are granted following the "*first come, first served*" principle.

The granularity of the locks has a major impact on the degree of concurrency [4]. The coarser the granularity of the locks, the slighter the likelihood of task concurrency. In our case, the smallest lockable unit is a geometric object which, in our opinion, perfectly meets the demands of collaborative engineering.

3.4. Awareness of the collaborators' activities

As opposed to classical, transaction-based database management systems that isolate concurrent users from each other, an environment for synchronous collaborative work has to make the participants aware of each other's activities [7]. Between absolute isolation and totally coupled human-machine interfaces (also denoted as WYSIWIS – "What you see is what I see") there are several levels of dependency, all suitable for different phases in collaborative work.

CoCoS provides three levels of awareness:

On the first, lowest level, only the geometric model is equal. The visualization of simulation data is independent, as well as the viewpoints and pointing devices. This level is suitable for collaboration of engineers from different domains or collaboration with little inter-dependency between the tasks of the collaborators.

The second level is equal to the first level, except for the visualization of simulation data, which is the same for all client applications. This level is suitable for collaboration between engineers from the same domain, where each participant keeps the freedom of choosing an appropriate view point.

On the third, highest level, also the view points and directions, as well as the (virtual) pointing devices are coupled. This level, also denoted as Leader-Follower pattern, is intended for very intense collaboration, where one participant guides another to a particular point-of-interest in order to discuss a certain phenomena.

On all levels, the CoCoS platform supports the principle of awareness in two ways: by tracking the modification actions of each single participant, and by providing information on the viewpoints and view directions of all participants. On the one hand, this information can be used to visualize so-called avatars, which are representations of the real participants in the virtual environment (Fig. 7). On the other hand, this information is utilized for implementing the Leader-Follower pattern as discussed above.

3.5. The collaboration server

The central collaboration server is the most important component on the CoCoS platform. It has to perform the following tasks:

– management of the shared model,
– management of users, their current view points and directions, their roles and rights, and the visualization means currently activated by them,
– concurrency control and
– management of the simulation servers, their locations, their start-up and steering parameters, as well as the structure of the simulation data they are producing.

Each of these tasks corresponds with a dedicated module in the collaboration server, as shown in Fig. 6.

The core of the collaboration server is the model management module. As discussed in Section 3.2, a hybrid model joining the geometric and the semantic model is used. In the HVAC example scenario, the model represents the obstacles in the fluid domain and the fluid domain hull. Boundary conditions are managed as semantic data attached to geometric objects.

Modifications like adding, removing or transforming obstacles are communicated from the performing client to the collaboration server. In order to avoid conflicts between the participants, the collaborative work is coordinated by means of locking mechanisms. As long as an object is locked by a certain user it cannot be modified by any other user. See Section 3.3 for an overview of the implementation of concurrency control in CoCoS.

The simulation management module keeps track of the simulators available, their locations and current state. When a simulation server joins the CoCoS platform, it registers with the collaboration server and provides all the required data. A client queries the collaboration server in order to connect to a specific simulation facility.

Each Simulation Server announces the structure of the numerical results it produces. This involves the type of grid, the denomination of each single field variable and the identification of scalar and vector fields. Up to now, only grids (uniform, recti-linear and structured) are supported, but the concept is open for unstructured meshes to be added in the future. The information is used by the clients to provide suitable visualization techniques and display the results accordingly. For vector fields for example, streamline and vector plane visualization are provided whereas scalar fields can be visualized by simple value planes or isometric surfaces.

The user management module provides information about the users currently participating in the collaborative session. This is required to support the several levels of awareness as discussed in Section 3.4. Besides information about his view point and direction, the visualization method currently utilized by the participating engineers is stored at the Collaboration Server. There are three visualization methods supported by CoCoS: vector planes, value planes, streamlines and isometric surfaces. The parameters of each visualization object are stored at the Collaboration Server and changes are passed to the clients by a notification mechanism.

3.6. The clients

The CoCoS clients serve as visualization and interaction interfaces for the engineers taking part in the collaborative session. The following basic services have to be provided by each client application:

– logging into/logging out of the collaboration server,
– visualization of the geometric objects, interaction facilities for transforming them, support for locking mechanisms,
– display of semantic data attached to the geometric object (at least via attribute-value tables),
– display of the current participants, notification of participants entering and leaving the session.

For the HVAC example scenario, the prototype client CoFluids was implemented. It enables a user to transform the obstacles inside an office and to visualize the CFD simulation data in the form of vector planes, isosurfaces or streamlines. The client can be run in single-window mode capable of stereoscopic rendering for use in Virtual-Reality environments, or in multi-viewer mode for use on desktop computers. Obstacles that are modified by another participant are shown in a different color and cannot be modified, i.e. they are locked (see Section 3.3 for an overview of concurrency control in CoCoS).

In CoFluids, awareness of the activities of the other participants is provided by a list of currently registered users, and by means of messages that signal whenever a participant joins or leaves the collaborative session. Furthermore, CoFluids can display the viewpoint and the view direction of the other participants using avatars. See Section 3.4 for more details on awareness support in CoCoS.

3.7. The simulation server

The main task of a simulation server is to build a bridge between the distributed collaborative system and a particular simulation kernel (Fig. 12). Like the clients, the simulation server is listening to the event

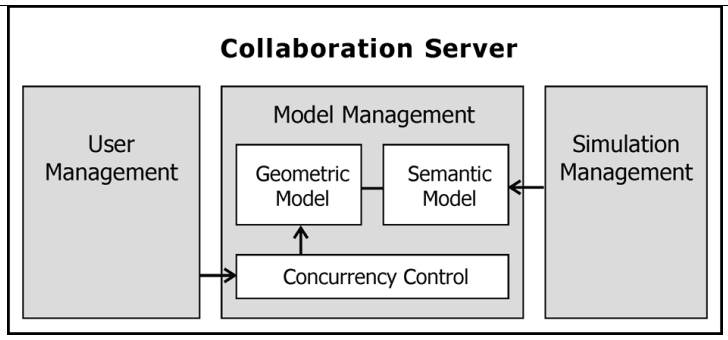

Fig. 6. The modules of the collaboration server: The model module manages the shared model and is responsible for the concurrency control, the user module manages the names, rights and viewpoints of the participants, and the simulation module keeps track of the simulators available, their locations and current state.

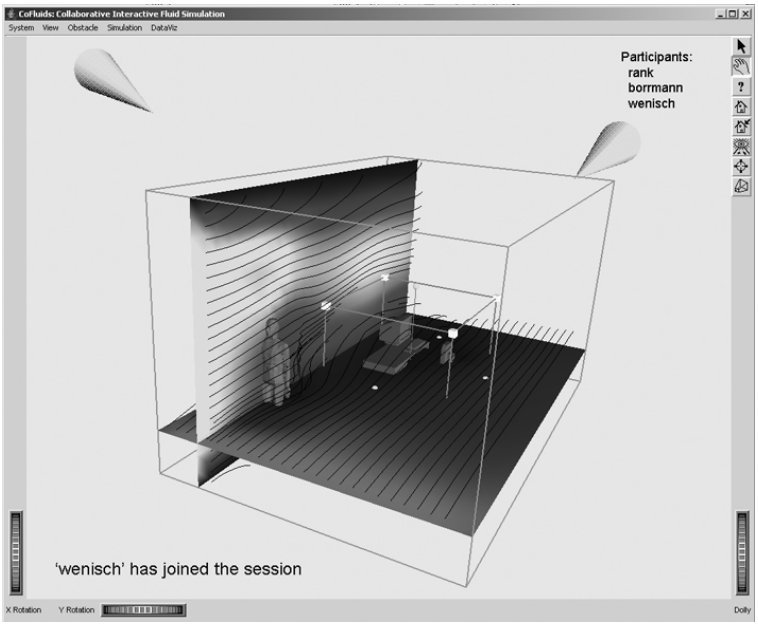

Fig. 7. Screenshot of the prototype client application CoFluids showing the movable obstacles in an office, streamline visualization of the resulting fluid flow, and avatars that represent the viewpoints of the other participants.

channels provided by the collaboration server. It is accordingly notified of any changes in the geometry and the corresponding boundary conditions and forwards this information to the simulation kernel.

At the start-up of the simulation server, it registers with the collaboration server and defines the model it uses for the start-up and steering parameters by means of the meta-model (see Section 3.2). Additionally it announces the structure of the numerical results that it produces (see Section 3.5). This information is gained by the client via the collaboration server in order to connect to the simulation server and to display the simulation results as well as the available steering parameters accordingly.

The minimum service that a simulation server must provide is an interface to start and stop the simulation and to pass on steering parameters. It has to notify the connected clients when the simulation has been started or stopped, and when a steering parameter has been changed.

For the transmission of the simulation data from the simulation server to the clients, we experimented with two different configurations: a push and a pull communication mode.

There are two main criteria for estimating the efficiency of the configuration: First, the simulation should not be slowed down by the communication overhead. Secondly, the transmission should be as fast as possible

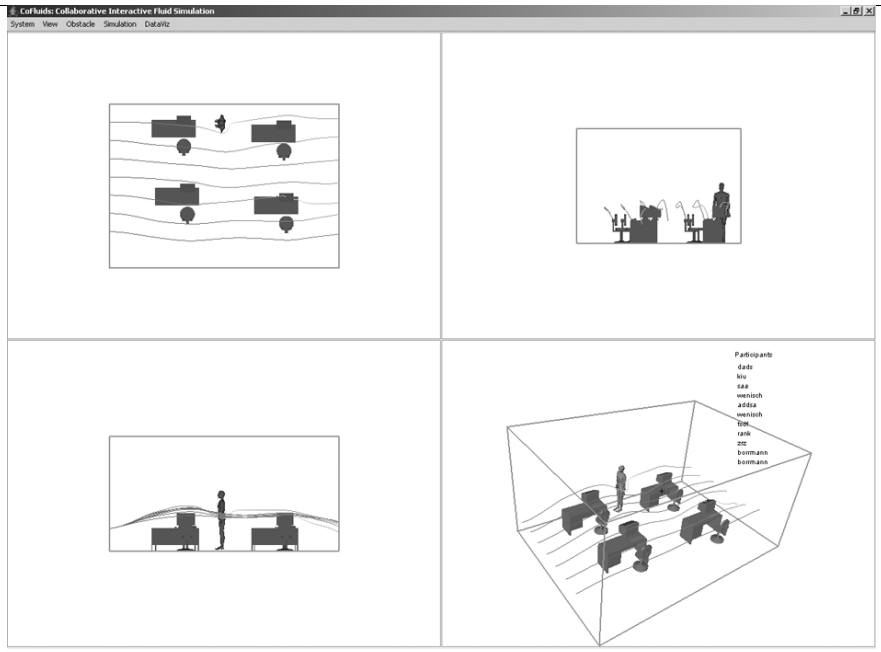

Fig. 8. The collaborative client application in multi-view mode for use on ordinary desktop computers. This mode provides three orthographic and one perspective mapping for easy navigation in three-dimensional space.

and as flexible as required. As opposed to streaming-based communication solutions as applied in video and audio broadcasting, for a Computational Steering client it is not necessary to receive every single 'frame' of the simulation results, but it is more important that it receives the most current simulation data.

In the first configuration, we used a CORBA Event Service implementation (see also Section 3.2) in order to distribute the simulation data. In this case, the data is pushed from the simulation server to the event service, which in turn pushes the data to the listening clients. The advantage of this configuration is that the distribution of the simulation results is completely decoupled from the simulation process. The major disadvantage is that each and every result set is transmitted, even if there are more up-to-date sets of result available on the simulation server (Fig. 8). So occasionally the event channel becomes choked by obsolete simulation data, if one of the clients is not fast enough to receive and process it.

The use of pull communication is more appropriate here. In this mode, the client pulls the data whenever it is ready to process it (Fig. 10). This method ensures that a client always receives the latest simulation data available. This should be regarded as the highest priority for a Computational Steering application.

Although a pull communication mode is also available using the CORBA event channel, its specification does not provide 'last in, first out' semantics, which are required here. For this reason, and due to the increased communication overhead when using an additional process, the event service is not employed for the distribution of simulation data in CoCoS. The advantage of decoupled communication as provided by the event service has been given up.

4. The lattice-Boltzmann simulation server

4.1. The lattice-Boltzmann method

The numerical method involved is of great importance for the feasibility of Computational Steering. The CFD simulation kernel developed by our research group for a Computational Steering prototype is based on the lattice-Boltzmann method [19,42].

During the past decade, lattice-Boltzmann methods (LBMs) have emerged as a complementary technique for the computation of fluid flow phenomena [17,40]. Common numerical methods for solving the Navier-Stokes equations are based on the discretization of the non-linear partial differential equations by applying finite volume or finite difference techniques, for instance. By contrast, LBM represent a bottom-up approach which begins with a discrete microscopic model

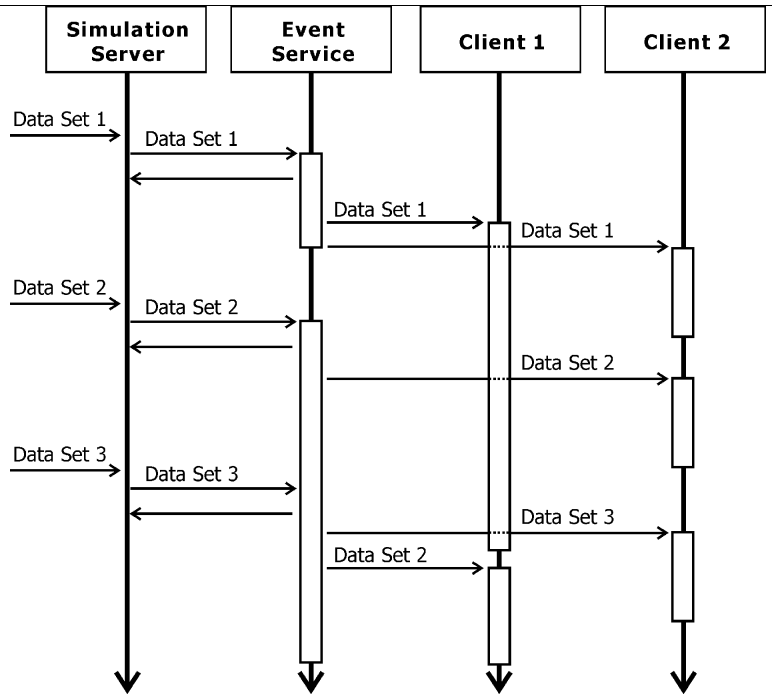

Fig. 9. Push communication mode for distributing simulation data. The simulation server sends the data to an event service as soon as it becomes available, and the event service is responsible for broadcasting it. Occasionally, the event service can get choked by obsolete simulation data.

preserving the desired quantities, such as mass and momentum, by construction in order to obtain hydrodynamic behavior on a macroscopic scale corresponding to the incompressible Navier-Stokes equations.

LBMs perfectly meet the demands of Computational Steering, especially in terms of the possibilities for an interactive modification of geometric boundary conditions. They use Cartesian grids, explicit time-stepping schemes and a marker-and-cell-like approach to define boundaries and obstacles.

Recently, LBM has been extended to include the simulation of turbulent convective flows. Although this has not yet been integrated into the simulation kernel used for the study in progress, the software will be upgraded soon using a Bousinesq approach, as discussed in [39]. For more detailed information on LB methods applied to indoor air flow simulations, see [40] and the references therein.

4.2. The discretization process

As already mentioned in Section 1.2, Computational Steering comprises the modification of boundary conditions and the visualization of the corresponding results of a simulation. In our case, the modification of the geometry of the simulated scene is particularly favored. An efficient grid generator is needed to achieve a fast response to such user interaction. Fortunately, the LBM used for our fluid simulations works on uniform Cartesian grids. To map the supported CAD-generated geometry (see Section 3.2) onto the computational grid (Fig. 11), a hierarchical space-partitioning concept based on octree structures was implemented [41].

The algorithm starts with the bounding cube of an object, region or the whole simulation scene, which is recursively subdivided into eight congruent sub-cubes as long as the currently tested cube intersects with at least one triangle (see Fig. 11). With each subdivision step, the so-called refinement level increases by one. When the refinement level has reached the resolution of the given computational grid, it is possible to identify the voxels to which the boundary condition has to be set.

5. Conclusion

This article presents the concept of the CoCoS platform, a highly flexible distributed system for multidisciplinary collaboration. In order to provide the collaborating engineers with suitable analysis and simulation capabilities, we illustrate the flexible integration

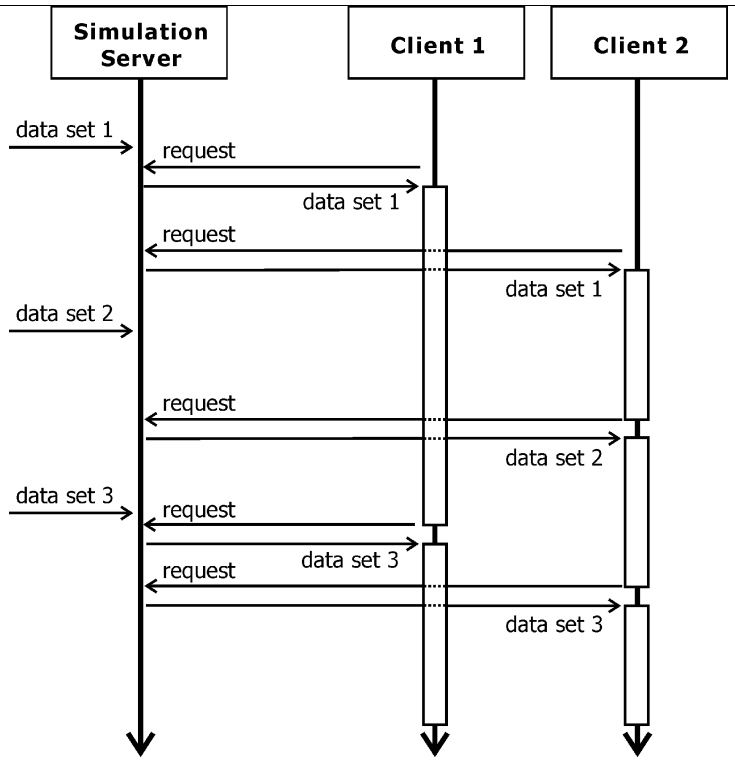

Fig. 10. Pull communication mode for distributing simulation data. Each client requests simulation data as soon as it is ready to process it.

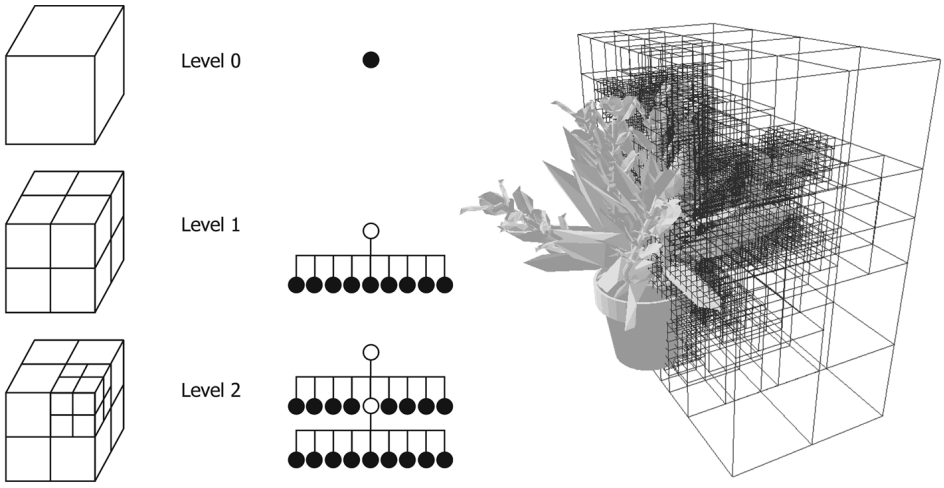

Fig. 11. The octree-based voxelization of an object. The algorithm starts with a root octant, which could be the bounding box of a given object. It is then recursively subdivided as long as it intersects with the object's geometry.

of simulation servers into the distributed system. The suitability of the concept has been proved by the implementation of clients and servers for a collaborative HVAC engineering scenario. It demonstrates the integration of a CFD simulation server whose kernel is based on the lattice-Boltzmann method, thus providing an interactive fluid simulation.

The basis of the collaboration platform is formed by a centrally managed, shared model joining an explicit geometric and a variable semantic model. Its emphasis lies on the geometric model, because of its vital importance for analysis and simulation tools and its

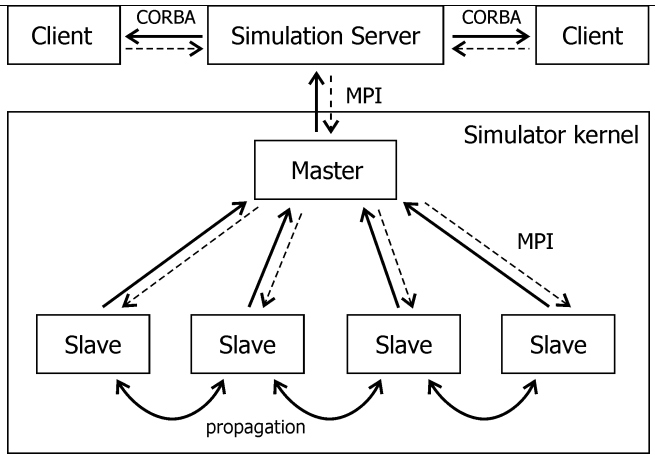

Fig. 12. Internal structure of the simulator kernel. Dashed lines denote the flow of start-up and steering parameters, continuous lines denote the flow of simulation results. The kernel consists of several processes, one of which is designated as the master process. The slaves perform the LB computation in parallel. A specific part of the fluid domain is assigned to each slave process. Where these parts meet, data has to be exchanged at each propagation step of the LB algorithm. The master gathers the results from the slaves at certain intervals and forwards them to the simulation server. All internal communication is realized by means of MPI, whereas communication between the distributed components of CoCoS is realized by means of CORBA.

domain-independent validity.

By separating the functionality of the collaboration server and the simulation servers, it possible to use different simulation facilities without the need to re-implement the generic functionality of geometry-focused, collaborative engineering. In addition, it is possible to use simulators for different physical phenomena at the same time, thus providing the basis for multi-disciplinary synchronous collaboration. CoCoS client applications are able to take part in the collaborative session by sending and receiving geometric data, but are free to receive different simulation data or no simulation data at all.

6. Outlook

Our on-going research will continue to focus on the integration of design and simulation tools and on supporting multi-disciplinary collaborative engineering.

A key aspect of the integration is the extraction of partial models with relevant information for simulation purposes from the overall Building Information Model. In a next step, we will therefore develop algorithms and tools for questioning BIMs using geometric and topological conditions.

Another important aspect of our future work will be to embed Grid protocols in the CoCoS platform. This will enable the engineer using a Computational Steering application to choose between different high-performance computing facilities available within the Grid network.

Finally, we will continue to improve the interactive fluid simulator in terms of performance, stability and complexity of the physical model implemented.

Acknowledgements

We would like to thank Mr. Oliver Wenisch from the Leibniz Rechenzentrum (LRZ) in Munich for his support during the development of the Computational Steering application.

The project presented here is partly sponsored by Siemens Corporate Technology and the Bavarian Competence Network of Scientific High Performance Computing (KonWiHR).

References

[1] R.M. Baecker, D. Nastos, I.R. Posner and K.L. Mawby, *The User-Centered Iterative Design of Collaborative Writing Software*, Proc. of SIGCHI conference on Human factors in computing systems, 1993.

[2] B.-C. Björk, Basic structure of a proposed building product model, *Computer-Aided Design* **21** (1989), 71–78.

[3] G. Booch, *Object-oriented Analysis and Design with Applications*, Benjamin-Cummings Publishing, Redwood City, CA, USA, 1994.

[4] U.M. Borghoff and J.H. Schlichter, *Computer-Supported Cooperative Work: Introduction to Distributed Applications*, Springer, Berlin, New York, 2000.

[5] E.F. Churchill, D.N. Snowdon and A.J. Munro, *Collaborative Virtual Environments: Digital Places and Spaces for Interaction*, Springer, London, 2002.

[6] P. Dewan and H. Shen, Controlling access in multiuser interfaces, *ACM Transactions on Computer-Human Interaction* (*TOCHI*) **5** (1998), 34–62.

[7] P. Dourish and V. Bellotti, *Awareness and Coordination in Shared Workspaces*, Proc. of ACM conference on Computer-supported cooperative work, 1992.

[8] C.M. Eastman, *Building Product Models: Computer Environments Supporting Design and Construction*, CRC Press, Baco Raton, Florida, 1999.

[9] T. Forkert, H.-P. Kersken, A. Schreiber, M. Strietzel and K. Wolf, *The Distributed Engineering Framework TENT*, Proc. of VECPAR2000: 4th International Conference on Vector and Parallel Processing, 2000.

[10] I. Foster and C. Kesselman, *The Grid: Blueprint for a New Computing Infrastructure*, Morgan Kaufmann Publishers Inc., 1999.

[11] S. Greenberg and D. Marwood, *Real time Groupware as a Distributed System: Concurrency Control and its Effect on the Interface*, Proc. of the ACM conference on Computer supported cooperative work, 1994.

[12] S. Greenberg, M. Roseman, D. Webster and R. Bohnet, Human and technical factors of distributed group drawing tools, *Interact. Comput.* **4** (1992), 364–392.

[13] C. Greenhalgh and S. Benford, MASSIVE: A collaborative virtual environment for teleconferencing, *ACM Transactions on Computer-Human Interaction* (*TOCHI*) **2** (1995), 239–261.

[14] O. Hagsand, Interactive MultiUser VEs in the DIVE System, *IEEE Multimedia Magazine* **3** (1996).

[15] International Alliance for Interoperability (IAI), Industry Foundation Classes, 2004.

[16] R. Johansen, *Groupware: Computer Support for Business Teams*, Groupware: Computer Support for Business Teams, New York, NY, USA, 1988.

[17] M. Krafzcyk, *Gitter-Boltzmann-Methoden: Von der Theorie zur Anwendung*, Professoral thesis, Technical University of Munich, 2001.

[18] O. Kreylos, A.M. Tesdall, B. Hamann, J.K. Hunter and K.I. Joy, *Interactive Visualization and Steering of CFD Simulations*, Proc. of the Symposium on Data Visualisation (VISSYM), 2002.

[19] S. Kühner, *Virtual Reality – basierte Analyse und interaktive Steuerung von Strömungssimulationen im Bauingenieurwesen*, Ph.D. dissertation, Technical University of Munich, 2003.

[20] K.P. Lam, A. Mahdavi, S. Gupta, N.H. Wong, R. Brahme and Z. Kang, Integrated and distributed computational support for building performance evaluation, *Advanced Engineering Software* **33** (2002), 199–206.

[21] R. Liere, J. Mulder and J.V. Wijk, Computational Steering, *Future Generation Computer Systems* **12** (1997), 441–450.

[22] J.M. Linebarger, C.D. Janneck and G.D. Kessler, *Shared Simple Virtual Environment: An Object-Oriented Framework for Highly-Interactive Group Collaboration*, Proc. of the Seventh IEEE International Symposium on Distributed Simulation and Real Time Applications, 2003.

[23] P. Luksch, S. Rathmayer and V. Sunderam, *Internet-Based Collaborative Simulation in Computational Prototyping and Scientific Research*, Proc. of the International Conference on Parallel and Distributed Processing Techniques and Applications, 2000.

[24] M. Mäntylä, *An Introduction to Solid Modelling*, Computer Science Press, College Park, MD, USA, 1988.

[25] T.P. Moran, K. McCall, B. van Melle, E.R. Peddersen and F.G. Halasz, *Some Design Principles of Sharing in Tivoli, a Whiteboard Meeting Support Tool*, In: Groupware for Real-time Drawing: A Designer's guide, McGraw-Hill, London, GB, 1995.

[26] M. Mortenson, *Geometric Modeling*, Wiley, New York, USA, 1985.

[27] R.E. Newman-Wolfe and H.K. Pelimuhandiram, MACE: a fine grained concurrent editor, *ACM SIGOIS Bulletin* **12** (1991), 240–254.

[28] Object Management Group, *The Common Object Request Broker: Architecture and Specification*, Revision 3.0, Framingham, MA, USA, 2004.

[29] Object Management Group, *The CORBA Event Service Revision 1.1*, Framingham, MA, USA, 2001.

[30] S. Pekkola, M. Robinson, J. Korhonen, S. Hujala, T. Toivonen and M.-J. Saarinen, *An Architecture for Virtual Reality*, audio, video, text, and document handling in applications supporting multi-person interactions, Proc. of 26th Euromicro Conference, 2000.

[31] B. Plale, V. Elling, G. Eisenhauer, K. Schwan, D. King and V. Martin, Realizing Distributed Computational Laboratories, *Int. Journal of Parallel and Distributed Systems and Networks* **2** (1999).

[32] D. Rantzau and U. Lang, A scalable virtual environment for large scale scientific data analysis, *Future Generation Computer Systems* **14** (1998), 215–222.

[33] M.A. Rosenman and J.S. Gero, Modelling multiple views of design objects in a collaborative CAD environment, *Computer-Aided Design* **28** (1996), 193–205.

[34] S. Singhal and M. Zhyda, *Networked Virtual Environments: Design and Implementation*, Addison Wesley Longman, Reading, MA, 1999.

[35] A. Schreiber, The integrated simulation environment TENT, *Concurrency and Computation: Practice and Experience* **14** (2002), 1553–1568.

[36] M.S. Shephard, M.W. Beall, R.M. O'Bara and B.E. Webster, Toward simulation-based design, *Finite Elements in Analysis and Design* **40** (2004), 1575–1598.

[37] A.S. Tanenbaum and M. van Stean, *Distributed Systems*, Principles and Paradigms, Prentice Hall, Upper Saddle River, NJ, USA, 2002.

[38] V. Mann, V. Matossian, R. Muralidhar and M. Parashar, DISCOVER: An Environment for Web-based Interaction and Steering of High-Performance Scientific Applications, *Concurrency and Computation: Practice and Experience* **13** (2001), 737–754.

[39] C. van Treeck, *Gebäudemodell-Basierte Simulation von Raumluftströmungen*, Ph.D. dissertation, Technical University of Munich, 2004.

[40] C. van Treeck, E. Rank, M. Krafczyk, J. Tölke and B. Nachtwey, Extension of a hybrid thermal LBE scheme for Large-Eddy simulations of turbulent convective flows, *Computers and Fluids* (2005).

[41] P. Wenisch, O. Wenisch and E. Rank, *Optimizing an Interactive CFD Simulation on a Supercomputer for Computational Steering in a Virtual Reality Environment*, In: High Performance Computing in Science and Engineering, Springer, Berlin, Heidelberg, New York, 2005.

[42] P. Wenisch, C. van Treeck and E. Rank, *Interactive Indoor Air Flow Analysis using High Performance Computing and Virtual Reality Techniques*, Proc. of Roomvent, 2004.