A Collaborative Multi-Scale Planning Platform: Concept and Implementation Approach

Matthias Flurl, Ralf-Peter Mundani, André Borrmann, Ernst Rank

Chair for Computation in Engineering
Technische Universität München, Munich, Germany

Abstract

The main goal of the DFG research project "3D Tracks" is to develop a collaboration platform for the interactive planning of inner-city carriageways. This platform should enable different planners to work synchronously and thereby using multi-scale planning data. An object-oriented database connected to the server facilitates complex spatial-temporal request concerning the planning state. Additionally, it will be possible to integrate Geo-Web-Services, which provide geodetic information regarding the planning process. In order to support the working process at the building-site, the collaboration platform will furnish possibilities to visualize the planning process on portable devices.

Keywords:
Collaborative planning, multi-scale geometry data, procedural geometry, synchronous working, levels of detail

1 The German Research Foundations project "3D Tracks"

In spring 2011, the German Research Foundation (DFG) approved the research project "3D Tracks". The main goal of the project is to develop a collaboration platform for the interactive planning of inner-city carriageways, a highly complex task that by nature involves plenty of different stakeholders. Thereby the collaboration platform should enable these different planners to work synchronously and with multi-scale planning data. An object-oriented database connected to the collaboration platform gives the planners the possibility to ask complex spatial-temporal requests regarding the planning process. In order to support the working process at the building-site possibilities to visualize the planning process on portable devices will be included. Furthermore, it will be possible to integrate Geo-Web-Services, which provide geodetic information that is necessary in this planning task. To accomplish these goals, the DFG founded five separate project teams. Each of these teams focuses on a particular aspect in the development of the collaboration platform:

The first team is responsible for the design of the actual collaboration server; the second team creates a procedural geometric model, which facilitates the multi-scale planning properties of the platform; the third team develops the spatial-temporal database. The forth team provides several Geo-Web-Services containing geodetic information, while the last team researches the possibilities of visualization tools on portable devices.

2 Collaboration tools for 3D modeling and planning

For a long time, software tools to support a collaborative engineering process have been subject to scientific research, particularly tools for collaborative geometric modeling. Kao and Lin [Kao et al., 1997] did some fundamental research in the late nineties regarding the development of a collaborative CAD/CAM system. Ramani, Agrawal, Babu, and Hoffmann [Ramani et al., 2003] provided an interesting prototype of a multi-client collaborative shape design system with a server-based geometry kernel; thereby a server hosts a central geometric model and forwards this central model to the different thin clients. One of these clients is the master-client who can change the model while the others at least view these changes in a synchronous manner. This idea is similar to the control token mechanism

in the Alibre Design 2D/3D CAD software. Bidarra, van der Berg, and Bronsvoort [Bidarra et al., 2002-1, -2] built a promising client server structure called WebSpliff supporting a true synchronous collaboration process. Kim, Mun, and Han [Kim et al., 2010] did some fundamental research in web services supporting a collaborative product development process. Li, Lu, Fuh, and Wong [Li et al., 2004] studied the development status of collaborative CAD software, thereby distinguishing horizontal and hierarchical collaboration functionality. Ku, Pollalis, Fischer, and Sheldan [Ku et al., 2007] studied the collaboration process and tools supporting this process regarding the construction of complex shaped buildings. Borrrmann [Borrmann, 2008] developed a collaboration server – the CoCoS platform – to incorporate simulation data in a collaborative planning process using CORBA as communication technology. Iacob [Iacob, 2011] did some very interesting research regarding patterns in the design of synchronous collaboration. Lee, Kim, and Banerjee [Lee et al., 2010] developed an intelligent CAD framework incorporating a design history-tracking algorithm in order to reduce redundant designs.

Obliviously, synchronous collaboration using a client-server architecture has been of high interest in scientific research during the last years, nevertheless in our project "3D Tracks" we want to add a new dimension to the synchronous planning process supporting the planners with multi-scale geometry data while working simultaneously on the same geometry model.

3 Main goals: Synchronicity and multi-scale data

Usually one distinguishes tools for a collaborative planning process whether they support asynchronous or synchronous planning processes.

If a planner decides in an asynchronous case to modify the geometric model, he asks the server for a local copy of the model and then solely works on this local data. Generally, one calls this process "checkout". While he is working with this local copy, the other stakeholders do not know anything about his changes. Server-internally the checkout process implies that the server creates a new branch in the history tree of the geometric model data that evolves completely separated from the original one.

When the above planner finishes his work, he checks in his modified local copy and the server tries to recombine the two different branches. If the other stakeholders did not modify the areas that the planner modified, the recombining process works without any problems. Otherwise, one manually has to recombine the different branches. Certainly, the server will try to support planner in this recombining process, but by nature, it cannot decide which modifications overrule other modifications on the same part of the geometric model. Obviously, manual recombination of the different branches is an undesirable and tedious task.

In a synchronous case, all participating planners see all changes made by another planner simultaneously. A possibility to realize this behavior is to maintain one central model residing on the server, while the different planners work on local copies, but in this case they notify every – even the smallest – change made on their copy immediately to the server, which then instantaneously informs the other clients. In details, this means when one planner starts a modification process he immediately notifies the server and the server itself instantaneously advises all the other stakeholders of this modification and prevents them from modifying the same part of the model. In particular, all the planners know about this modification process and more over about all single steps of the modification process. Therefore, by nature, the problems of recombining different local files do not exist, which is a true reason to work synchronously. A drawback of working synchronously is that permanent notification of the different planners about actions of the others entails both a high network traffic and a high number of updates on the client-side.

One of the main goals of the "3D Track" project is to provide a synchronous planning tool as described above. We will see later how our server facilitates these synchronous planning possibilities using a procedural geometric model.

During the process of planning, the perspective on the geometric model – the central source of the planning process – changes according to the state of planning. Thereby, it is clear that the later one planer is in a planning process the more details are of his respectively common interest. We call these different perspectives different levels of details (LOD). To illustrate this idea let us face the different steps of planning a tunnel – though we will do this in an absolute simple manner.

In the earliest stage, one will have to plan where the precise track takes course. We may represent this course in the geometric model as a simple line.

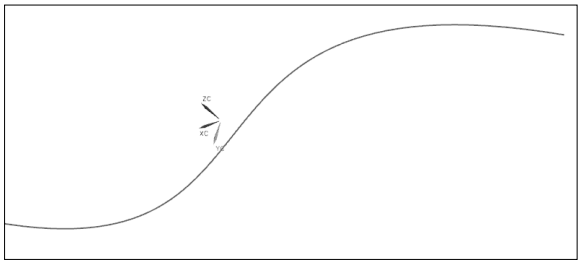

Figure 1.Planning at level of detail 1.

In a later step, one will plan the outer geometries of the tunnel sections. Therefore, perspective granularity switches to a higher (finer) level, i. e. the collaboration platform switches to LOD 2.

Figure 2.Planning at level of detail 2.

In a further step, the planner must make decisions about the air conditioning, the electric wires, the lighting, the emergency escapes and so on. We declare these decisions to belong to LOD 3.

Figure 3.Planning at level of detail 3.

One main goal of the collaboration server is to incorporate these different levels of detail: That means that different planners will be able to work on different levels of detail of the same model at the same time. In particular, the server will notify the clients about changes made in deeper (coarser) levels to planners working in higher (finer) levels in a synchronous manner.

Thus, in comparison to a usual collaborative planning tool, we have to add a new dimension of complexity to the history of our planning process: There is not only one model that has to be maintained during the entire planning process, but also must be maintained on different levels of detail. We face the above-mentioned example in detail:

Planner A works on LOD 1 and sketches the principle way of an inner carriage track (first step). Planner B models the tunnel-tube at LOD 2 (second step). After planner B already did some crucial planning efforts regarding the tunnel-tube, planner A changes the track again (third step). Planner B realizes this, since the server locks the "trail-object". A possibility to visualize this locking mechanism is that the client-planning tool displays the carriage track highlighted in a dashed line. Planner A fin-

ishes his modifications and unlocks the track-object (step 4). The server automatically notifies planner B and updates his view of the model.

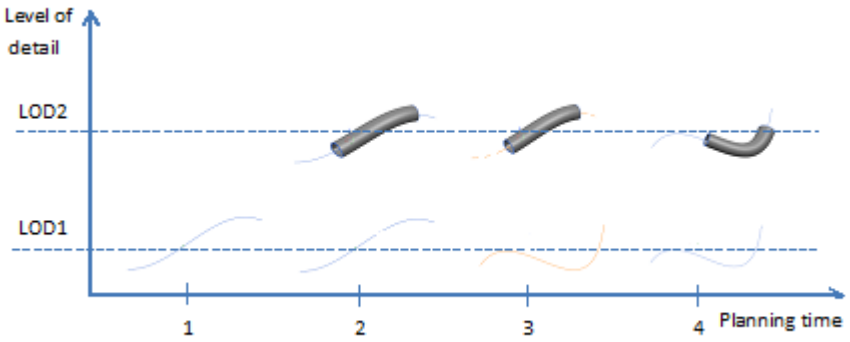

Figure 4.Synchronous working with multi-scale data.

Looking at these two key-features of the "3D Tracks" project one of the main question obviously is how one builds a proper geometric model. In principle, one can distinguish two kinds of geometric models: explicit and procedural ones. We characterise these two types in the following way: Within an explicit geometric model a body is described by its surface respectively the faces building its surface. A typical example is the BREP description. In this case, smoother descriptions are accomplished by approximating a body's surface by smaller facets, i. e. by a larger number of smaller triangles – a triangular mesh. This kind of geometry is fast and easy to handle for 3D visualization tools. However, in our case, i. e. the concrete model respectively the concrete data resides on the server, one would have to transfer an enormous amount of data via the network.

In a procedural geometric model, instead of describing a body by its surface, it is declared by a composition of subsequent (simple) geometric construction steps. For instance, the tunnel tube of the previous example is created as procedural model starting with a sketch representing the tunnel front, sweept along a spline to form the complete tunnel tube. To render the tube as 'real' model (using a BREP representation, e. g.) it is forwarded from the server to the visualization device. In this case, the data amount that needs to be broadcasted via the network is extremely low, but clients have to be more intelligent: Though these clients must be able to create an explicit model from a procedural one. The definition of a proper procedural geometric model is one of the main goals of the second project team.

Since we want to support a synchronous communication, reduction of the network-traffic is extremely important, hence a procedural model description is the right choice. Additionally, though we cannot guarantee that all clients have sufficient computing power to build a satisfying explicit model – think in particular about mobile devices or smart phones – the server will provide a possibility to create an explicit model from the procedural one by itself and send this model via the network to those clients.

4 The fundamental architecture

While building a prototypical implementation of the collaboration server we designed seven central software modules. In the following section, we will illustrate their basic functionality and study how these modules interact to achieve the goals presented above.

3.1. API-User interface

The API-User module is the concrete interface for the clients to communicate to the server. Therefore, in a first step, the API module provides the basic functions to login, logout, and manipulate the geometric model.

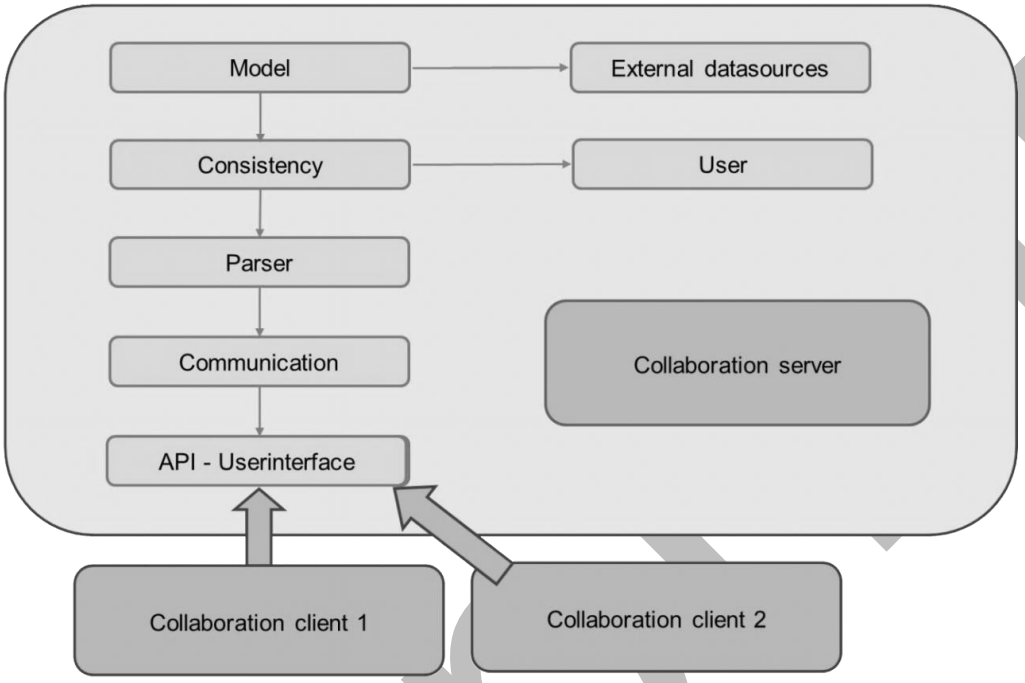

Figure5. The main server modules.

To manipulate the geometric model we offer only three methods: One can add procedural operations, modify, and delete them. In order to add a procedural operation the user respectively the client module creates an operation object, i. e. an instance of the operation class or one of its subclasses, resp. Then he calls the *AddOperation* method providing the operation object as a parameter, for instance the users creates a new sketch – in our definition an atomic step – and saves this sketch, before the software immediately creates a *NewSketchOperation* object and calls the method in question. In a further step, the user may extrude this sketch, so that the program will create an *AddExtrusionOperation* object and call the *AddOperation* method again thereby providing this object as a parameter. In this way, the complete procedural geometric model can be built by only calling these three methods providing the appropriate parameter objects.

3.2. Communication

From a technical point of view, to establish the API-User interface one needs a concrete way of communication with the server. Naturally, there are several technologies one can use building the communication module, for instance Web Services, CORBA, RMI, or sockets. All these technologies have different advantages and drawbacks and they all have already been used building client-server architectures. While technologies such as Web Services, RMI, and CORBA are easy to handle, plain socket communication gives full control and allows the maximum of speed for data transfer.

3.3. Parser

Since we decided to use sockets, the concrete communication consists of sending byte-streams from the client to the server and vice versa. The parser module reads these strings and interprets them, i. e. building certain command respectively operation objects. The parser forwards these objects to the responsible module, for instance after receiving a *Login()* string the parser builds an appropriate *Log-*

in-object and forwards this object to the user module, which is responsible for the login and logout process.

3.4. Consistency

Since many stakeholders are synchronously involved in the planning process, the server has to handle concurrency problems, i.e. different users try to modify the same part of the model at the same time. We described these problems in detail above. Without an elaborate consistency-mechanism, the synchronous work easily leads to an inconsistent geometric model. Hence, the server provides a locking mechanism: as soon as one user decides to change a certain part of the model, the server locks this part for the other users and notifies the other clients. This notification process means raising events in the client-software tools containing all necessary information to keep the model consistent. Let us face a concrete example: planner A created a sketch and afterwards planner B wants to modify this sketch. When planner B enters the modification mode, the server immediately locks this sketch (dashed red line) for planner A as depicted in the screen shots below.

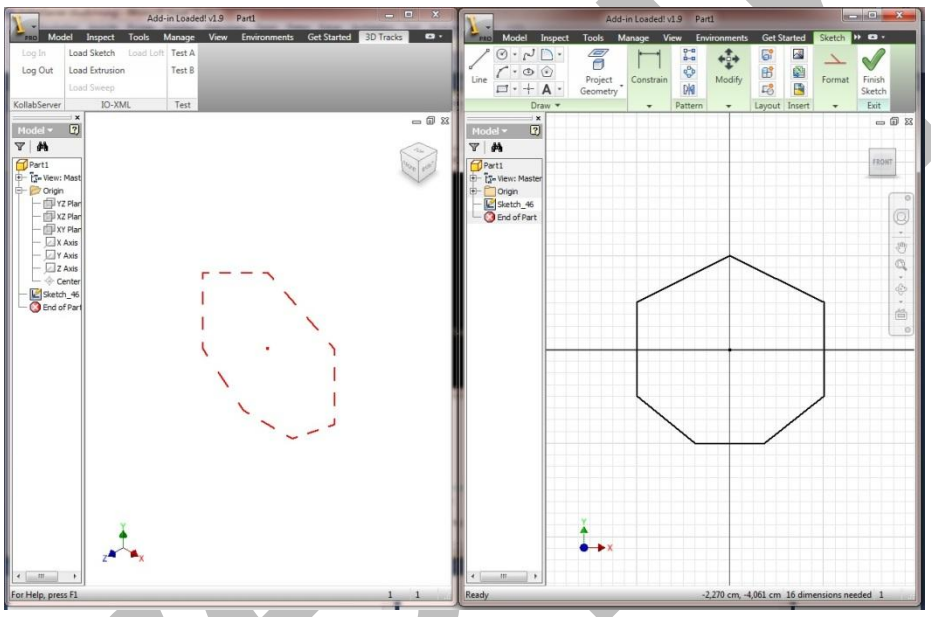

Figure 6.Two planners working synchronously using Autodesk Inventor as collaboration clients.

Additionally, the consistency manager is responsible for making the model persistent to the object-oriented database provided by the third project team. That means the server forwards the new procedural steps in certain intervals to the database and reloads the model from the database when needed.

3.5. Model

The center of the planning process is the geometric model, in our case a procedural model. The model module maintains this model. Therefore, it is responsible for interpreting the objects created by the parser and then doing the appropriate actions. In particular, this includes notifying the several clients about these modifications thereby using the user module.

3.6. User

In the first instance, the user module is simply responsible for the login and logout-mechanism, so that different planners can participate in the collaboration process. According to their rights and rules in the real world planning process, the user module assigns different rights and roles to these clients.

One further responsibility is to determine the update intervals of the single clients: in principle, in a synchronous case the server forwards all modifications of the geometric model immediately to all clients as said above. This behavior may lead to a disturbing high rate of updates. In particular, this is

disturbing when the updates affect parts of the model that are not of one users interest. To avoid these problems the user can unsubscribe from certain update events, i. e. updates concerning an area of the model that does not affect him. Thereby, one should keep in mind that it is sufficient in order to keep the model consistent that the server immediately locks objects when one planner begins to modify them.

3.7. External data sources

One main goal of the collaboration platform is to integrate external data sources that might be useful in the planning process; in particular, we want to include simulation data in the planning process. Let us face an example to make things clear. One stakeholder in our carriageways planning process is planning a subway station and thereby it is his responsibility to determine the right location for the escalators. If he changes the position of one escalator – even if it is only a small change – it could have a significant impact on the pedestrian movement on the surface. Directly including a pedestrian simulation into the planning tool therefore is a highly interesting goal.

5 The collaboration clients

In the recent years, some CAD tools became standards in the engineering industry, such as Autodesk Inventor and Siemens NX. Both tools provide some *real* collaboration possibilities, but these collaboration features are limited to one software product, i.e. one NX user can communicate via the network with other NX users and Inventor planners with different Inventor planners. Additionally, these tools offer the possibility to save their geometric data in a standard file format such as IGES or STEP, so that other software tools are able to load these data in order to work with it. However, to the best of our knowledge there is no possibility that CAD products of different vendors communicate in a synchronous way as described above.

Therefore, one additional benefit of our collaboration platform and our procedural geometry model is the ability to offer different products the possibility to communicate with each other via our platform. To achieve this, we built a library that one can integrate using the programming interfaces of the software tools. We already did some very promising experiments using the two above mentioned software tools Siemens NX and Autodesk Inventor as collaboration clients.

Additionally, we provide our own collaboration client able to do the basic geometric operations. At the momentary stage, this client is very simple, but it already provides all key features to show a synchronous planning case. For example – using our client software – let us face two clients who built a very simple model of a station. An easy geometric operation is to translate objects. However, already in the case of translating an object we have to do all the things regarding the consistency of the model: Locking objects, notifying the other users, and unlocking the objects again. In details: Planner A decides to move one pillar to the left-hand side while at the same time planner B decides to move it to the right-hand side. If the server would allow this situation there would immediately arouse a contradiction in the two local copies of the geometric model that make a later recombining of the two copies impossible. In the screenshot below, we show this situation.

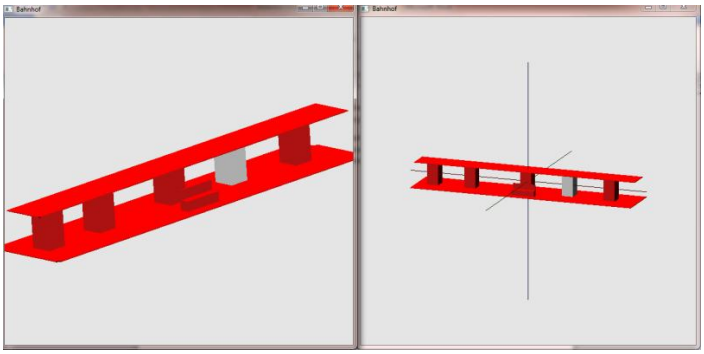

Figure 7.Synchronous working: Client 1 (left) modifies a pillar; the pillar is immediately locked for client 2 (right).

Now client A moves the locked pillar to the left-hand side. Client B immediately sees this operation. According to the login parameters, he (in real time) sees client A moving the pillar or realizes the updates only when client A has confirmed the new position of the pillar, resp.

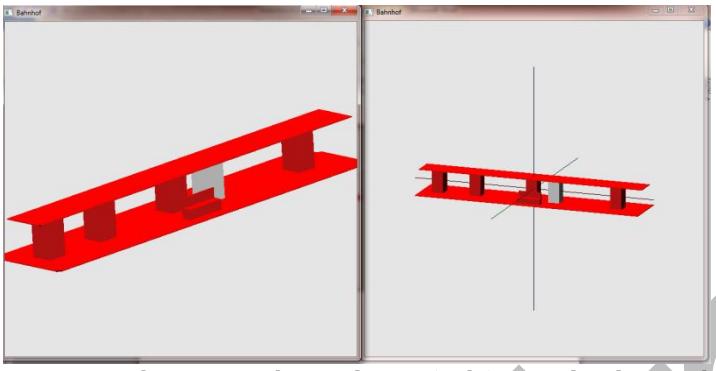

Figure 8: Synchronous working: Client 2 (right) immediately sees the modified position of the pillar.

6 Conclusion and future goals

The main goal of the "3D Track" project is to build a collaboration platform that supports collaborative planning processes while planning inner-city carriageways thereby supporting multi-scale geometry data, complex temporal-spatial request about the planning state, integrating Geo-Web-Services and providing visualization tools on mobile devices. We already did some very promising test with our prototypical server, thereby working with different client tools.
In the nearest future, we will extend our simple client to do some more complicated synchronous work with the mentioned software tools Siemens NX and Autodesk Inventor. Therefore, we have to improve the visualization abilities of our client and extend the concrete dynamic link libraries that one can integrate into the common software tools. We also will expand our set of operation objects, so that we can build up truly sensible geometric models. Thereby we will integrate also geodetic information, such as subterranean information that is necessary for planning a concrete inner city carriageway.

BIDARRA, R., VAN DE BERG, E. AND BRONSVOORT, W. F., 2002.A Collaborative Feature Modeling System, Journal of Computing and Information in Engineering, Vol. 2, Issue 3(2002), 192 et seq.

BIDARRA, R., KRANENDONK, N., NOORT, A. AND BRONSVOORT, W.F., 2002. A Collaborative Framework for Integrated Part and Assembly Modeling, 7th ACM Symposium on SOLID MODELING ANDAPPLICATIONS, Germany (2002)

BORRMANN A., 2008.Computerunterstützung verteilt-kooperativer Bauplanung durch Integration interaktiver Simulationen und räumlicher Datenbanken,35-56, Aachen: Shaker Verlag

IACOB, C., 2011. Mining for patterns in the Design of Systems for Synchronous Collaboration. AsianPLoP2011

KAO, Y.C. AND LIN, G.C.I., 1997. Development of a collaborative CAD/CAM System, Robotics and Computer-Integrated Manufacturing, 14(1998), 55-68

KIM, BC., MUN, D. AND HAN, S., 2010. Retrieval of CAD model data based on Web Services for collaborative product development in a distributed environment, The International Journal of Advanced Manufacturing Technology Vol. 50, Numbers 9-12, 1085-1099

KU, K., POLLALIS, S.N., FISCHER, M.A. AND SHELDEN, D.R., 2007.3D Model-Based Collaboration in Design Development and Construction of Complex Shaped Buildings, Journal of Information Technology in Construction, Vol. 13(2008), 485 et seq.

LEE, H., Kim, J. AND BANERJEE, A. 2010.Collaborative intelligent CAD framework incorporating design history tracking algorithm, Computer-Aided Design 42 (2010), 1125-1142

LI, W.D., LU, W.F., FUH, J.Y.H., WONG, Y.S., 2004. Collaborative computer-aided design-research and development status.

RAMANI, K., AGRAVAL, A., BABU, M. AND HOFFMANN, C., 2003.CADDAC: Multi-Client Collaborative Shape Design System with Server-based Geometry Kernel, Journal of Computing and Information in Engineering, Vol. 3(June 2003), 173 et seq.