GridCAD: A Collaborative CAD System Based on Grid Computing

He-Fu Shi[1], Min Tang[1,2], Shang-Ching Chou[2] and Jin-Xiang Dong[1]

[1]Zhejiang University, loocool@sina.com, tang_m@zju.edu.cn, djx@zju.edu.cn
[2]Wichita State University, chou@cs.wichita.edu

ABSTRACT

Current works on distributive and collaborative CAD systems often focus on collaborative request submitting, but not distributively executing. A new approach to integrating Grid Computing into Collaborative CAD system is proposed in this paper. A prototype system GridCAD based on this approach is implemented with Web Services based Grid Computing technology. XML/SOAP is adopted as the communications mechanism between Server and Clients. The Session Manager provides the solution of validity maintenance of product models under collaborative environments. In particular, as the core module, Task Manager enables GridCAD to dynamically split complex operations and to distributively perform subtasks. GridCAD also guarantees interoperability on heterogeneous systems so that different types of systems can communicate smoothly and perform the computation much faster than on a single machine.

Keywords: Web Services; XML; SOAP; Grid Computing; Dynamic Scheduling; Collaborative CAD.

1. INTRODUCTION

Due to the rapid development in network and computer technologies, new demands in *computer aided design technologies* arose. Complex system designs today are frequently done in a collaborative distributive environment. A lot of researches have been done on distributed and collaborative CAD applications. Many prototype systems with different technologies and architectures also have been experienced.

1.1 Current Approaches on Distributed CAD

Considering the network architecture, most of previous works on distributed CAD researches mainly put the focus on two kinds of models: *Server/Client Model* and *Equal Node Model*.

In a Server/Client Model, all the modeling operations like entities creating or editing at the model level are performed on the server side. Commonly, the server is composed of modeling kernel and geometric kernel; it maintains the validation of models and provides interactive interfaces for clients' requests during design processes. As a node exposed to the end-user, the client provides UI Module and Model Displaying Module. Experimental systems like WebSPIFF[6] and NetFeature[2] are well known in this direction.

Equal Node Model assures all computer nodes included are "equal" in functionality. Same modeling kernel and model sharing module are installed in each host in an equal model system. A cutting action made by a single host will cause all nodes performing the same operation under communication mechanism for maintaining consistency of the model. Current prototypes with Equal Nodes Model such as CollIDE[1], Cooperative ARCADE[3] and CSCW-FeatureM[4] and TOBACO[5] are ready for experimental use.

Progresses have been achieved in distributed CAD system, but most of the communication mechanisms of current works are based on Socket, CORBA, JAVA-RMI, or DCOM. These mechanisms limit the scalability and applicability of Internet based collaborative CAD systems. The network condition nowadays becomes quite complicated: multiple protocols, heterogeneous systems and various programming languages. An open and platform-independent distributed architecture needs to be developed urgently. Designers also need a more powerful and faster CAD system in a collaborative environment for functions like large-scale model comparison and high quality rendering. Since those time consuming operations generally processed by the server, multiple and continuous requests may cause the server to be overloaded and turn the design process too slow to be practical.

1.2 Web Services & Grid Computing

As a burgeoning technology, Web Services are software components that are exposed to a network via a statically defined interface, allowing other applications to leverage exported functionality. Web Services plus *"Universal Description, Discovery and Integration"* (UDDI), *"Web Services Description Language"* (WSDL)

and *"Simple Object Access Protocol"* (SOAP) present a complete solution for distributed applications. UDDI provides a means of discovering other services and registering new services for others to discover. WSDL offers a way to provide the needed information about a discovered service to allow a client to interact with it. SOAP is adopted as the interfaces of invoking between server and clients.

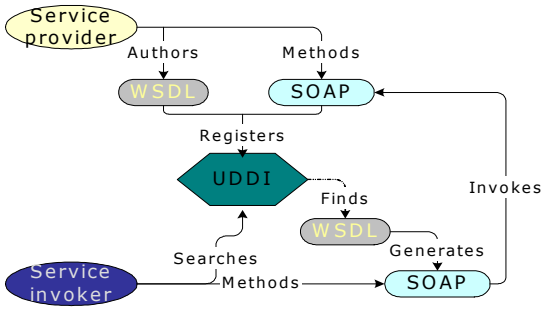


Fig. 1. Web Services process

Figure 1 shows how the Web Services process flows. At the beginning, a service is coded up. As long as the service is web-accessible, it's a candidate for a service registry with its own WSDL document. This is all part of the UDDI portion of the figure. UDDI also allows a user to search all registered services for a specific service name and the registry returns all the matching services.

Grid computing [7], most simply stated, is distributed computing taken to create the illusion of a simple yet large and powerful self managing virtual computer out of a large collection of connected heterogeneous systems sharing various combinations of resources. A distributed application integrated with grid computing can make use of all available system resources effectively, and can separately perform time consuming computations to achieve faster responses. Especially for a distributed CAD system, grid computing can greatly improve efficiency in the design processes of all end-users.

1.3 Integrating Web Services & Grid Computing in Distributed CAD System

GridCAD is a prototype collaborative CAD system integrated with grid computing technology. Server/Client architecture is adopted in GridCAD and Web Services is used as communication interfaces between the server and clients. In order for GridCAD to integrate grid computing functionality, we introduce Task Manager Module embedded in server. We use SOAP and *Extended Markup Language* (XML) as transferring interface and data format. These characteristics overcome the bottlenecks of traditional distributed systems to support simple porting to multiple platforms. The Task Manager subdivides complex operations into several sub-operations, and then submits them to grid services and executes them as a parallel system.

This paper is organized as follows. In Section 2, the distributed architecture of GridCAD is overviewed; the structure and functionality of the server and the clients, and the module of model maintenance are also described in this section. In Section 3, our main contribution, the Grid Computing mechanism is embedded into GridCAD, is presented. Especially, the Task Manager Module is detailed. Some implementation details and examples are given in Section 4. Finally, conclusions and the future work are discussed in Section 5.

2. DISTRIBUTED ARCHITECTURE OF GRIDCAD

2.1. Architecture of GridCAD

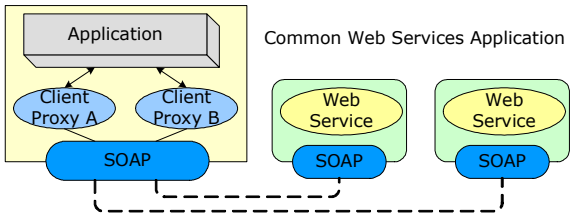


Fig. 2. A common Web Services based application programming model

We use Figure 2 to graphically demonstrate the client proxy application programming model [8]. A client proxy is similar to a "stub" that is local to the application and provides the same API (Application Programming Interface) as the actual web service. However, the client proxy does not implement any of the web service's business logic, instead brokers communicate between the application and the remote web service. Essentially, the application invokes methods of the client proxy, which then encapsulates the method calls together with any arguments as a SOAP message and transmits the message to the web service. The web service performs the desired functionality, and returns the result as a SOAP message to the client proxy. The method call of the client proxy then returns with the result.

Figure 3 depicts the main architecture of GridCAD and it is mainly based on the framework of Figure 2. It hides the complexity of network implementation because any invoking for a client web service simply contacts with the client proxies and all low level networking details will be transparent to application.

2.2 Server

The CAD kernel is implemented at the server side; it provides all of modeling operations and geometric functions; and the kernel does not have any distributive characteristics. For better programming structure, all functions of the kernel are exposed as APIs and these APIs have been classified and encapsulated into different Web Services by functionality (entity creating, entity editing, model viewing, scene rendering, etc.) APIs with similar function will be put into the same web service.

The server establishes one client proxy for each client. When a SOAP message from client arrives, the server decodes it and processes the request information into its corresponding Client Proxy. Unlike the common Web Services programming structure, client proxies in GridCAD do not directly submit the message to Web Services but refer to Session Manager for arbitration. The Session Manager performs the validation of requests from all clients. If a request is valid, then GridCAD processes it with Web Services, otherwise refuses this request.

2.3 Client

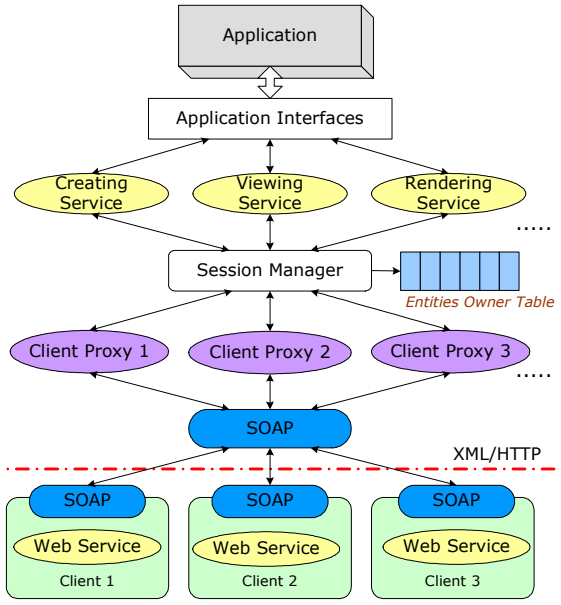


Fig. 3. Distributive architecture of GridCAD

The Client is composed of two main components: the display module providing model displaying functionality; and the user interactive module providing functionality to operate on model. The outgoing interfaces are also designed with Web Services. When a user request is processed, Web Services encapsulates it into a SOAP message and send to the server via the HTTP protocol. The display module updates the model view as soon as the result of request returns. In addition to the Web Services of request sending and receiving, a special Web Services is setup at each client end for receiving model update information, so any change of the model by other clients will be displayed at the client end in time.

2.4 Session Manager

The Session Manager stores all information of client requests and offers model validity maintenance functionality.

Due to the limitation of network condition, not all of operations on one model can be updated at every client end in real-time. So such situation like editing an entity that has already been deleted from one client could happen at another client. To avoid such illegal operations, the Session Manager Module is adopted. Session Manager is based on the *Entities Owner Rule*: any client that wants to operate on a certain entity should check out it at first. The implementation of Session Manager is based upon *Entities Owner Table*. Entities Owner Table stores all possible entities owner information of every model available in the current distributive environment. Any owner information of one entity can be found in this table. When an operation request arrives, Session Manager will check Entities Owner Table for the target entity. If the entity is not found in the table, that means the entity could have been deleted by another client, the request will be refused; or if the target entity is found but there is no owner information, then the request sender becomes the owner of the entity. If both the entity and owner information exist, then Session Manager compares the original owner with the request sender and determinates whether the request should be submitted or refused.

Any entities creating and deleting operations will cause Entities Owner Table updated to keep Session Manager working correctly. Some requests like model viewing that does not change the model data will skip Entities Owner Table and will be executed directly.

3. GRID COMPUTING MECHANISM

This section presents our main contribution: the Grid Computing mechanism is embedded into GridCAD. The Task Manager splits the task submitted by Session Manager into several sub-tasks, and processes them to Grid Services available over network. In addition, the Task Manager is also responsible for gathering all results generated by Grid Services and for assembling a final result of the original task. With such a layer at the server side of GridCAD, some complex or time-consuming operations will skip the CAD Kernel and will be executed distributively to keep the server running effectively and smoothly.

3.1 Task Definition

In GridCAD, each request from clients is wrapped as a Task object and each task corresponds to a certain API of the server. The data members of class Task are listed below.

```
Class Task {
        TaskID                  idTask;

        RequestID               idRequester;
        RequestTime             tmRequest;

        TaskType                tpTask;
        TaskName                nmTask;
        TaskObject              objTask;
        TaskParam               paraTask;

        TaskComplexity  cmplTask;
}
```

To identify a task uniquely, a global `TaskID` is assigned to bind every task received by the server. Task also records `idRequester`, to indicate the ID of Client that sends the request and `tmRequest` as when the request is submitted. Class member `tpTask` states as the type of the requested operation like Entity Creating, Editing and etc, for specifying the corresponding Web Service. `nmTask` here in class stores the name of a task and `objTask` points to the target object of the task, including the ID of model and ID of object. All parameters needed for operation are kept by data member `paraTask`. `cmplTask` specifies the complexity of the operation, each type of task inheriting from class Task will implement its own method to calculate `cmplTask` for itself. For examples, a model comparison operation weights the sum of entities and blocks in source models, the number of layers in models and etc. to compute the `cmplTask`; and `cmplTask` of a rendering operation depends on the size of the scenes, resolution of target picture, method of ray tracing and sum of entities.

All the above data members provides generic information of a base Task object. All those details about specific operations will be implemented in inherited classes. The capsulation of task not only makes the requests flow among modules more clearly but also results in simpler programming structure and in higher performance of server for executing different operations with classified Web Services.

3.2 Grid Services [9]

Grid Service is the smallest computational unit of Grid Computing mechanism in GridCAD. Grid Services is a set of particular Web Services places at clients distributively and randomly. Unlike the Web Services described in section 2, Grid Services response only when Task Manager invoking them. Each Grid Service is responsible for a specific operation like model comparison, scene rendering, etc.

Grid Services could be placed randomly on network and every Grid Service will register itself in UDDI when it is activated, Task Manager could find and invoke the required Grid Services by searching UDDI with WSDL documents.

3.3 Task Manager

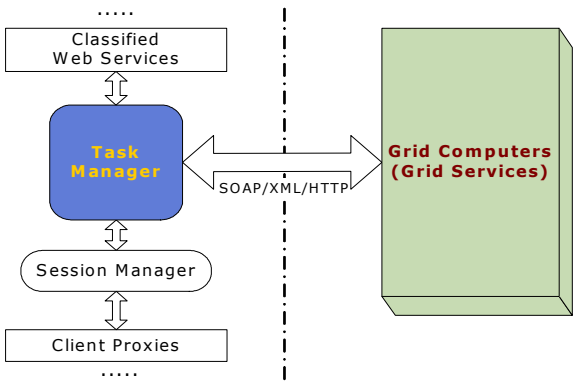


Fig. 4. Task Manager Module in GridCAD

Task Manager is the key component that enables GridCAD to have Grid Computing functionality. Figure 4 shows the position where Task Manager locates. Compared with Figure 3, Task Manager is a relatively independent module in GridCAD. Unlike the framework shown in Figure 3, Session Manager here submits tasks to Task Manager, not to local Web Services. Task Manager analyses the task and chooses a proper way to handle the task: Grid Services or local Web Services.

3.3.1 The Task Manager Process

As shown in Figure 5, Task Manager maintains a Task Queue. All valid requests checked by Session Manager are stored in Task Queue before being actually executed. Following the "FIFO" rule, Task Manager picks the first task from Task Queue, and then queries the type (tpTask), name (nmTask), and parameters (paraTask) of the task. By these properties, Task Manager searches the UDDI for available Grid Services. After that, Task Manager checks the task if it is divisible, and if it is, splits it into several subtasks by the properties of the task and available Grid Services returned by UDDI as searching result. A client proxy will be constructed for calling a Grid Service remotely. When all calls to Grid Services complete and the results of each subtask return to Task Manager, the task is accomplished.

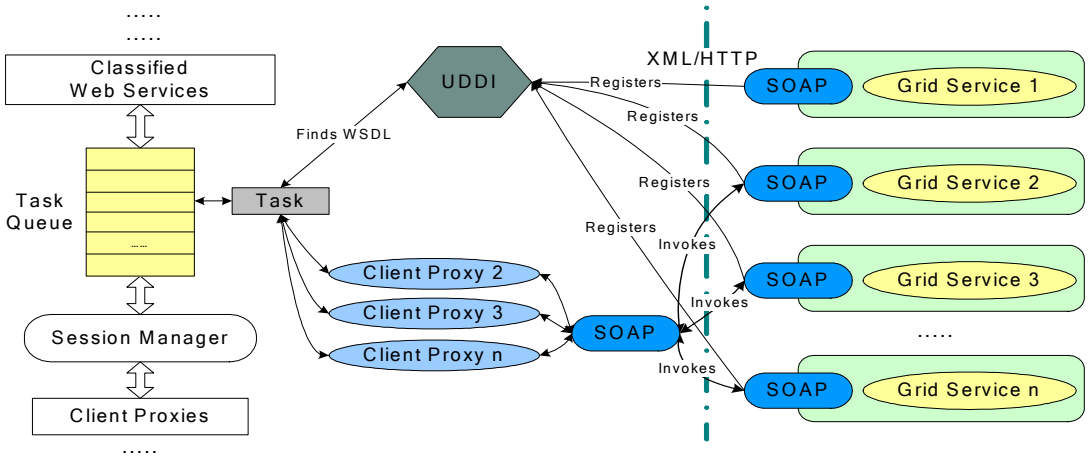


Fig. 5. The Task Manager process

As mentioned above, a special UDDI is setup in GridCAD for registering and searching Grid Services dynamically. UDDI is an open standard, in GridCAD it could be placed at any host over the network and the only thing is to specify the right network address when registration or search is performed.

3.3.2 Task Checking

The Integration of Grid Computing in GridCAD enables it to carry out some operations partially and distributively. To reduce the server's responding time and improve clients' efficiency, quite a part of the server load is shifted to Grid Services on clients. But considering semantics of some CAD operations, their divisions are knotty or even impossible. Moreover, some operations are lightweight in time and space costs so that the usage of server resources is relatively insignificant and it won't affect the efficiency of the whole system. Therefore, these operations needn't to be split in Task Manager.

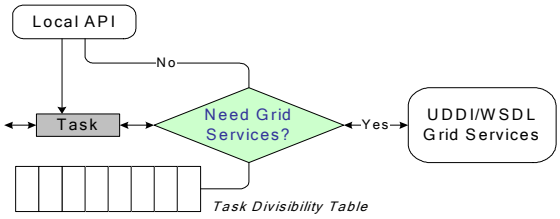


Fig. 6. Task Checking Module

The Task Checking Module is designed to solve the problem described above. It contains a Task Divisibility Table (TDT), which mainly holds segments called `tpTask` (Type of Task), `nmTask` (Name of Task), and `segTask` (Complexity Granularity of Task). Segments of `segTask` store the lower limit of complexity of this type of task. Any operation of the same type in the design process will be divided into subtasks if its `cmplTask` is larger than `segTask`. When a task arrives, Task Checking Module analyzes its `tpTask`, `nmTask` and `cmplTask`, and then seeks the record of task with type tpTask in TDT, then compares `cmplTask` and `segTask`. Finally, Task Checking Module determinates whether the operation should be split or not and how many subtasks it should be split into.

Assuming that task is an instance of specific operation submitted by an end user and `tdtItem` is a record corresponding to task in TDT, the process above could be described in the following pseudo codes:

```
if (tdtItem.segTask == 0)
    LocalApiProc(task);
else if (tdtItem.segTask > 0)
  numSplit =
       task.cmplTask/tdtItem.segTask+1;
```

TDT is presented as a configuration file of GridCAD. We assume that the tasks with value 0 of segTask would be directly carried out at the server side without being divided. numSplit indicates the largest number of subtask that could be split from original task. It is a suggestive value. The final value also depends on the number of available grid services returned by UDDI.

3.3.3 Task Splitting & Result Merging

As the key component of Task Manager, TSRM (Task Splitting & Result Merging) searches UDDI for available Grid Services about the divisible task that is verified by Task Checking Module. Then TSRM splits the task into

several subtasks in conformity with numSplit calculated by Task Checking Module, the result returned by UDDI, and task's own parameters. In the view of data structure, a sub task is complete and contains all attributes of a task object. Some parameters of a subtask may be a subset of those in its parent-task. Rules and methods of each specific task is predefined and implemented in each inherited class.

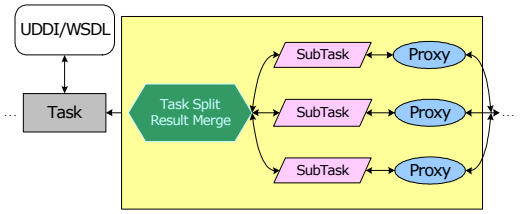


Fig. 7. Task Splitting & Merging Module

After task division, TSRM creates a proxy for each subtask to invoke Grid Service remotely, as shown in figure 7. When all calls to Grid Services return to proxies, TSRM assembles results by the rule of division and returns it to homologous client of the task.

The entire process of Task Manager will be described e.g., as follows. Model comparison is a typical operation in CAD used to find out the differences of a model from another. This task can be divided by coordinates of models. When a client submits a model comparison request, Task Manager picks it out. Task Checking Module firstly queries TDT for its divisibility property and calculates its `numSplit`, then hands over it to TSRM if it is divisible. TSRM searches UDDI for specified Grid Services related to model comparison and gets the WSDL of them. After that, the division of the model comparison task will be performed by the minimum of `numSplit` and the number of available Grid Services.

The following function depiction may help to understand the comparison operation:

```
CompareResult *ModelCompare(
    cadModel *mdl_a,
    cadModel *mdl_b,
    Point top_left,
    Point btm_right);
```

Parameters `mdl_a` and `mdl_b` are the two models to be compared. And point `top_left` and `btm_right` indicate the area to be compared and it is a rectangle represented by point pair of the diagonal corners. Function `ModelCompare` will return the result in `CompareResult` type that contains the differences of all entities in two models.

In our example we suppose that UDDI returns four Grid Services offering model comparison interface. In this case, the division of the task is to split the compare-area as shown in Figure 8.

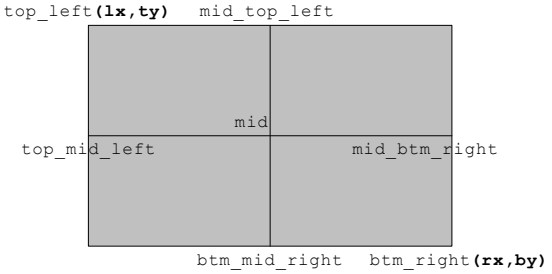


Fig. 8. Compare area splitting

After division, the original area becomes four sub-rectangles as below (also depicted in point pairs of diagonal corner):

```
(top_left, mid);
(mid_top_left, mid_btm_right);
(top_mid_left, btm_mid_right);
(mid, btm_right);
```

Points	*Coordinates*
top_left	(lx, ty)
mid_top_left	((lx+rx)/2, ty)
top_mid_left	(lx, (ty+by)/2)
mid	((lx+rx)/2, (ty+by)/2)
mid_btm_right	(rx, (ty+by)/2)
btm_mid_right	((lx+rx)/2, by)
btm_right	(rx,by)

Tab. 1. Sub area coordinates

These four parts of comparison are assigned to four Grid Services. Then the TSRM waits and gathers four results from Grid Services. The combination of the four sub-results will be returned to client.

The above discusses the execution process of a particular task in TSRM. In the implementation of Task Manager, multiple threads technology is also adopted to enable TSRM to pick several tasks from Task Queue at the same time and to transact in parallel. The more Grid Services available, the less resources of clients wasted and the more efficiency gained.

4. IMPLEMENTATION OF GRIDCAD

GridCAD is implemented on the platform of eCAD system we developed, which is a stand-alone CAD system based on ACIS [10]. eCAD supports "DWG" and "DXF" file formats, and is fully compatible with AutoCAD 2000 system of AutoDesk. eCAD is also extended with some capabilities as realistic scene rendering, models management, etc. The server of

GridCAD inherits the entities operation module and geometry engine from eCAD. The user interface and display modules of eCAD are adopted by the client of GridCAD. All the exchanges of data between the server and clients are transmitted in XML format through SOAP and HTTP, and all remote invoking is implemented by invoking the interface exposed by Web Services on the server or clients.

According to the demand of collaboration and distribution, a Session Manager Module is developed and embedded in the server side of GridCAD. The Task Manager Module is also implemented to enable GridCAD to integrate the Grid Computing ability. Several instances of particular functions of Grid Services, such as model comparing, scene rendering, etc., are implemented and spread at client sides for the Task Manager dynamically invoking.

As described in the architecture of GridCAD, Session Manager and Task Manager Modules are quite independent; therefore the Grid Services based Grid Computing functionality could be easily integrated into other distributive CAD applications. What we should do is to add a Task Manager Module for the server, and those Grid Services can be achieved with some modification.

4.1 Heterogeneous Cients

As an open software component standard, Web Services applications are based on platform independent software architecture; this breaks the barriers of operating systems and development platforms. These good characteristics of Web Services applications make the implementation of the clients of GridCAD and Grid Services very simple. We have developed different clients both on Microsoft Windows in MFC/.net and RedHat in Linux/j2se. In our experimental environment, the two kinds of clients are both running properly. To sum up, regardless of the platforms of clients and the programming languages chosen to implement clients, the server concerns only the WSDL of Web Services.

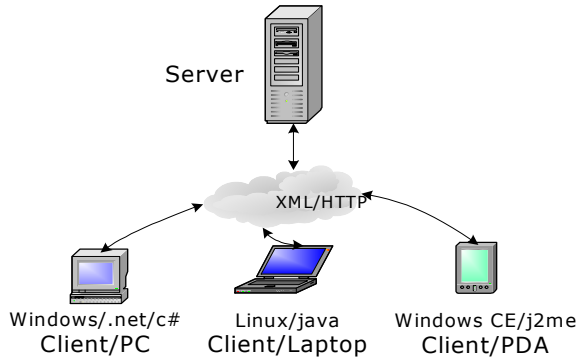


Fig. 9. Multiple platforms of clients

Figure 9 depicts the integration of heterogeneous operating systems and programming environments of GridCAD, including Windows/PC, Linux/Laptop, and Windows CE/PDA.

4.2 Model Data Representation

As an opening self-describing markup language, XML offers GridCAD a suitable data-transferring model. The usage of XML for representing model makes the structure of model more outstanding and incarnates the relations among data. Applications based on XML can also search for specific contents efficiently and accurately in XML documents, ignoring the irrelevant. For CAD model data, the XML representation makes it much easier to outline the relations of entities, view ports, and workspaces (paper space & model space in AutoCAD terms).

Furthermore, incrementally transferring of data is also feasible for updating the changed data so that re-sending the entire model won't be necessary. The sample code below shows how to describe the modification of a model in XML format. In this example, the entity with "entity no 0000" is modified by resetting start and end point.

```
<?xml version="1.0" ?>
<!DOCTYPE GridCAD (View Source for full
doctype...)>
<dwgupdate>
     <blocks>
          <block blkname="*Model_Space">
               <entity no="0000">
                    <line>
                         <Lpt0
                     data1="400.054434"
                     data2="120.145565"
                     data3="0.000000"
                     type="double" size="3" />
                         <Lpt1
                     data1="500.477618"
                     data2="214.478589"
                     data3="0.000000"
                     type="double" size="3" />
                    </line>
               </entity>
     </blocks>
</dwgupdate>
```

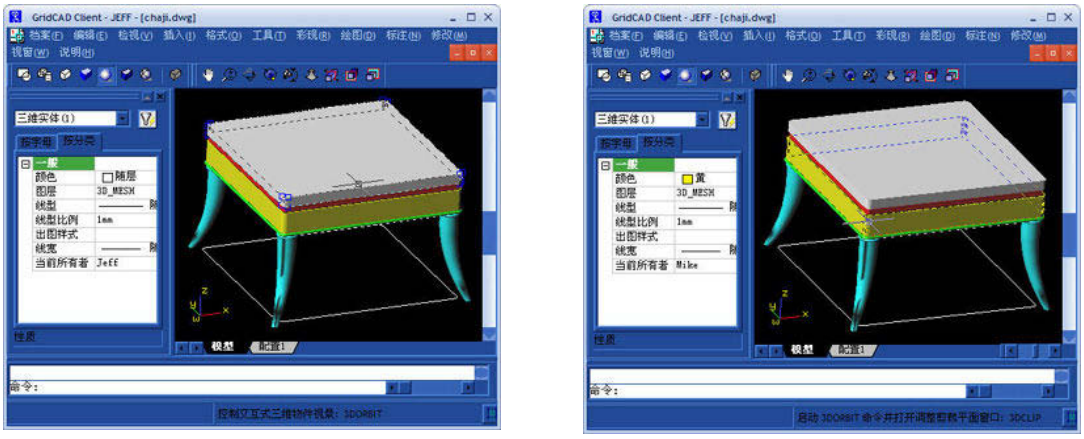
In GridCAD we also retain an appropriative Web Service - Updates Receiving Service on each client to get the modifications of model from the server side. The server stores all of models within the entire design environment and builds indexes by owner clients for these models; any changes of models can be sent to the right clients by the indexes. An Updates Informing Service is set on the server side; its activation depends on the process of a task. When a task that makes changes to model is finished, the server starts its

Updates Receiving Service and sends the updated data to all clients related.

4.3 Examples and Experimental Results

4.3.1 Collaborative Design

By queries Entity Owner Table, Task Manager Module determines a request should be processed or blocked. Figure 10 describes 2 scenarios that user "Jeff" check properties and owner info of different entities in the same model. Jeff is the owner of the surface of tea table and can modify it, but can't modify the parameters of the soleplate; the soleplate was checked out by "Mike" at the same time.



a b

Fig. 10. Screen shots of collaborative design from GridCAD (for the enlarged figures, please visit our website)

4.3.2 Grid Computing and Experimental Results

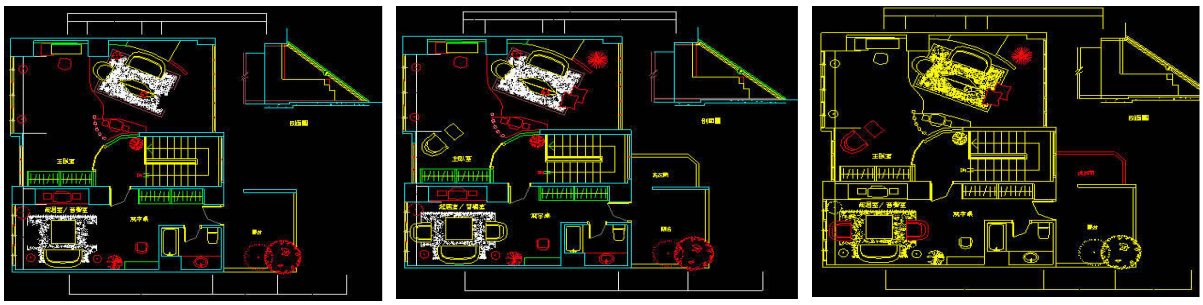
The model comparison operation is a time consuming process in CAD applications. The experimental data of model comparison operation in GridCAD are illustrated by Table 2 and Figure 11. We have tested three cases for different numbers of Grid Services involved. The first case is without Grid Service. The second case is with two Grid Services. The third case is with 4 Grid Services. As listed in Table 2, in the first case the server carries out the entire operation without the Grid Services support and costs 45.532 seconds. The second one costs 26.438 seconds. The third case costs 16.443 seconds. Because a model comparison is made up of a large amount of entity matches, multiple Grid Services not only share the load of comparing task but also decrease the mismatching ratio by splitting the model space for the entities in different region will not be taken into comparison.

The process of each Grid Service is neutral and parallel so that only the one with maximal time-cost should be taken into consideration; it is 18.176 seconds in second case and 7.681 seconds in the third one. In Grid Services supporting cases, because Grid Services are interoperated remotely, the time spent on task splitting/merging, message passing, and data transferring also should be added into final data statistics.

The statistics in Table 2 show that the Task Manager with Grid Computing works appropriately and effectively on large-scale computations and operations to greatly reduce the time costs and to cut down the load of server. The original models and result of comparison are illustrated in Figure 11 as screen shots.

Executed by	Server	2 Grid Services		4 Grid Services			
Compare Region	Total	First half	Second half	Top left	Top right	Bottom left	Bottom right
Entities in models	6489/6502	4691/4697	1798/1805	2585/2590	2106/2107	1684/1686	114/119
Execution time	42.532	18.176	8.262	2.634	7.681	3.675	2.453
Sum time	42.532	26.438			16.443		
Maximal time-cost	42.532	18.176			7.681		
Task split time	0.0	7.504			7.801		
Result merge time	0.0	2.337			2.614		
Net transfer time	0.0	2.010			1.062		
Totally costs	42.532	30.027			19.158		

Tab. 2. Timings of model comparison with various numbers of Grid Services
(Taken in Environment: Intel Pentium III 667MHZ, 394M SDRAM, Windows 2000 pro Service Pack 3)



a b c

Fig. 11. Screenshots of original models and comparing result (Original models are shown as picture a and b, the differences are marked with red color in c, for the enlarged figures, please visit our website)

5. CONCLUSION AND FUTURE WORK

This paper presents the architecture of a Web Services based CAD system, GridCAD, which provides a vehicle for Grid Computing. The advantage of Web Services is to enable a distributed environment in which any client can interoperate with server clearly and seamlessly in a platform-independent, language-independent fashion. The components based architecture also enhances expansibility and maintainability of GridCAD.

As the most outstanding approach in GridCAD, its Grid Computing mechanism makes good use of system resources to obtain the power as a super computational environment. The experimental test proves that our system could achieve real-time interoperation with complex CAD computations and operations under collaborative design.

6. ACKNOWLEDGEMENTS

This work was supported in part by the NSF Grant CCR-0201253 and Chinese National 973 project, project number: 2002CB312106.

7. REFERENCES

[1] Nam, T. J. and Wright, D. K. CoIIIDE, “A shared 3D workspace for CAD”, In Proceedings of the 4th International Conference on Network Entities, Leeds, UK, 1998, 103~105.

[2] Lee J.Y., Kim, H., Han, S.B. and Park, S.B. “Network-centric feature-based modeling”, Proceedings of Pacific Graphics '99, Seoul Korea, 1999.280~289.

[3] Stork A, Von Lukas U, Schultz R, “Enhancing a commercial 3D CAD system by CSCW functionality or enabling co-operative modeling via WAN”, Proceedings of the ASME Design Engineering Technical Conferences, September 1998, Atlanta, CIE-5711.

[4] Stock, A. and Jasonch, U., “A collaborative engineering environment”, Proceedings of TeamCAD’97 Workshop on Collaborative Design, Altana, GA, 1997, 25~33.

[5] Dietrich, U., von Lukas, U. and Morche, "Cooperative modeling with TOBACO", Proceedings TeamCAD 97, GVU/NIST Workshop on Collaborative Design, Atlanta, May 12-13, 1997. 115~122.

[6] Bidarra, Berg et al., “Collaborative Modeling with Features”, Proceeding of DETC’ 01, 2001 ASME Design Engineering Technical Conferences, 2001, 1~11.

[7] Liangjie Zhang, Jenyao Chung and Qun Zhou, “Introduction of a Grid architecture and toolkit for building Grid solutions”, http://www-106.ibm.com/developerworks/webservices/library/ws-grid1/, IBM developerWorks, Web Services, 2002.

[8] Sandeep Chatterjee, “Developing Real World Web Services-based Applications”, http://javaboutique.internet.com/articles/WSApplications/, 2002.

[9] S. Tuecke, K. Czajkowski, J. Frey and etc. “Open Grid Services Infrastructure (OGSI), (draft)”, February, 2003.

[10] www.spatial.com.