II CONGRESSO NACIONAL DE ENGENHARIA MECÂNICA
II NATIONAL CONGRESS OF MECHANICAL ENGINEERING
12 a 16 de Agosto de 2002 - João Pessoa – PB

SISTEMA CAD PARA WEB

Nelson Vogel
Marcelo Shimada
Marcos de S. G. Tsuzuki
Escola Politécnica da Universidade de São Paulo
Departamento de Engenharia Mecatrônica e de Sistemas Mecânicos
CEP 05508-900, São Paulo, SP, Brasil. E-mail: mtsuzuki@usp.br

Resumo

Os sistemas de computação com a arquitetura de clientes finos ("*thin client*"), também conhecidos como arquitetura em três camadas, estão se propagando cada vez mais. Isso pelas suas características de facilitar a manutenção dos sistemas, tanto em hardware quanto em software. Se houver a necessidade de aumentarmos a capacidade de processamento do sistema, será suficiente (em primeira instância) aumentar a capacidade apenas da máquina servidora. Cada vez que surgir uma nova versão melhorada, ela pode estar disponível apenas na máquina servidora. Estas características também são desejáveis para os sistemas CAD. Neste trabalho propomos uma arquitetura para um Modelador de Sólidos B-Rep de modo a permitir sua execução em um ambiente três camadas. Para isto foi utilizado o modelo ASP ("*Application Service Provider*"). Este modelo permite que um software seja alugado ao invés de ser vendido. O resultado é um programa CAD que roda em um servidor, podendo ser acessado (utilizado) a partir de um browser via uma rede local (LAN) ou a Internet. A aplicação servidora inclui um Modelador de Sólidos B-Rep desenvolvido em linguagem C++. A aplicação cliente possui uma interface com o usuário e está implementada em Java, sendo que para exibir os objetos tridimensionalmente utiliza o OpenGL. O Cliente possui um mapeamento da estrutura de dados suficiente para realizar seleções e exibir o sólido em wireframe e em shading. Além da linguagem Java, outras tecnologias abertas tais como o XML e o socket (TCP/IP) estão sendo utilizadas, garantindo a escalabilidade e portabilidade para diversas plataformas.
Palavras-chave: Sistema CAD, Cliente-Servidor, OpenGL

1. INTRODUÇÃO

No mundo digital atual, dados e informações estão disponíveis em abundância. Vários sistemas de captura de dados, de aplicações variadas, disponibilizam um grande volume de dados. A própria existência da Internet propiciou a criação e existência destes dados e informações. Acredita-se que tornar todo e qualquer serviço possível que beneficie o usuário final pela Internet é um novo paradigma. A implementação de uma aplicação em uma arquitetura cliente-servidor, significa que o sistema CAD original será dividido em duas partes. Em nosso caso, a aplicação cliente será executada por meio de um browser, e a aplicação servidora será implementada como um servidor WEB. Um aspecto principal, neste trabalho, foi a distribuição das responsabilidades entre a aplicação cliente e a aplicação servidora. O ideal é reduzir as responsabilidades da aplicação cliente para que manutenções e melhorias sejam facilmente implementáveis, bastando alterar a aplicação servidora. Um outro aspecto importante foi a escolha da tecnologia a ser utilizada, várias estão disponíveis atualmente: CORBA, DCOM, XML, dentre outras. Neste trabalho será apresentado um pequeno resumo sobre as pesquisas realizadas e o sistema desenvolvido.

2. ARQUITETURA DA SOLUÇÃO

Porque conectar computadores através de uma rede? A facilidade de comunicação e a transferência de informação era a resposta padrão até a chegada da Internet, intranets e dos "*thin clients*". Agora, a resposta inclui a transferência facilitada de recursos computacionais. Atualmente, as considerações sobre a arquitetura da aplicação cliente-servidora é o principal fator para utilizá-la efetivamente. O computador cliente executa a aplicação cliente (por exemplo, um browser). O computador servidor executa funções especializadas como publicar páginas WEB. Tanto o computador cliente como o computador servidor devem possuir um processador. Mas em uma arquitetura do tipo "thin client", o computador servidor deve ser um processador potente, e o computador cliente não necessita possuir grande potência. Entretanto, a definição de um valor absoluto para a potência desejada vai depender da aplicação específica.

Nas aplicações cliente-servidor é preciso definir qual é o papel de cada uma das partes e, em função disso, definir a forma de comunicação. Antigamente sistemas multi-usuários eram baseados em *main-frames*, máquinas UNIX ou semelhantes, onde diversos terminais emulavam as telas do computador servidor, no qual ocorria todo o processamento. Isso era (e ainda é) muito utilizado em sistemas baseados em caracteres, nos quais não há o aproveitamento dos recursos gráficos hoje disponíveis em praticamente todos os computadores pessoais. Existe uma tecnologia recente que permite o uso de emulação de programas baseados em janelas (windows). Essa tecnologia, o MetaFrame, da empresa Citrix, (ainda) não se tornou tão popular, devido ao seu custo e ao uso de tecnologia proprietária. Por isso as aplicações cliente-servidor correspondem hoje a maior parte dos aplicativos sendo criados. Neles, parte do processamento passa a ocorrer no computador cliente, aproveitando dessa maneira o poder de processamento da mesma. Isso viabilizou a proliferação de uso dos sistemas gráficos distribuídos.

O browser, que surgiu com a expansão da Internet, é um sistema que tem ao mesmo tempo a flexibilidade de um emulador e a escalabilidade de uma aplicação cliente-sevidor. A flexibilidade do browser existe uma vez que ele surgiu para acessar diferentes sistemas remotos no mundo cibernético. Sua escalabilidade existe porque a Internet cresceu junto com o surgimento e aumento de uso dos sistemas operacionais gráficos, dentre os quais se destaca o Windows da Microsoft. Assim, os browsers tiveram que se adaptar aos recursos gráficos e figuras, que passaram a ser uma exigência dos usuários dos microcomputadores.

No início, a função do browser era apenas de exibir páginas html, uma linguagem que se tornou padrão na Internet. Com o tempo percebeu-se o poder da rede quando foram criadas páginas

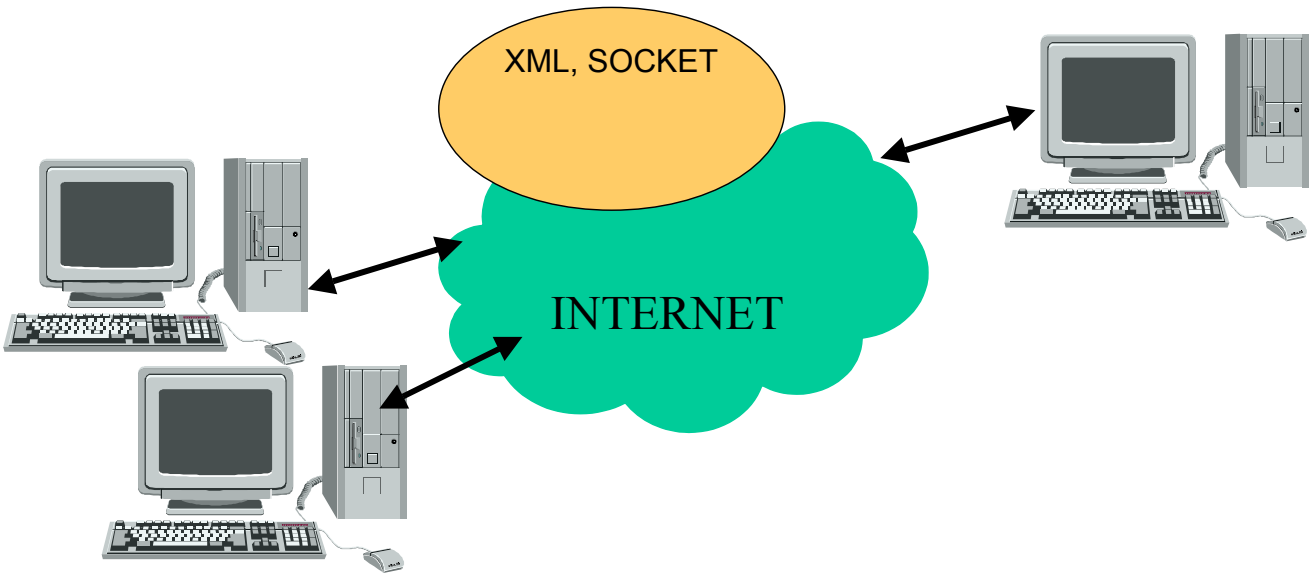

Figura 1. Arquitetura de uma aplicação cliente servidor utilizando a Internet, onde é possível que vários clientes se conectem a uma aplicação servidora.

dinâmicas, e então surgiram tecnologias e linguagens de programação para gerar páginas dinâmicas (ASP, Java Servlet, Cold Fusion, etc.) e recursos que foram agregando funcionalidades ao browser, dentre algumas podemos citar o Flash e o Java. Esses *Plug-ins* tiveram um importante papel na sobrevivência e consolidação do browser como a ferramenta padrão para a Internet.

Para a aplicação cliente havia duas possibilidades: ser criada uma aplicação cliente que será instalada no computador cliente de cada usuário, ou a aplicação cliente ser executada através de um browser. A segunda opção foi escolhida, por ser a mais prática, evitando a instalação de uma nova aplicação no computador cliente, e facilitando a manutenção de todo o sistema cliente-servidor.

As responsabilidades de que cada uma das aplicações envolvidas (cliente e servidor) gerou uma grande discussão. Poderíamos optar pela aplicação cliente ter um mapeamento de toda a estrutura de dados dos sólidos, ou então apenas uma interface suficiente para exibir os sólidos na tela do computador cliente. A primeira opção nos dá uma maior flexibilidade de realizar alterações no sólido sem a necessidade de utilizar a inteligência da aplicação servidora ou a força computacional do computador servidor, otimizando o fluxo de comunicação (troca de mensagens). Por outro lado, é preciso se preocupar com a escalabilidade do sistema, onde novas funções que forem desenvolvidas na aplicação servidora devem estar disponíveis na WEB, sem a necessidade de alterar a aplicação cliente.

As tecnologias disponíveis investigadas foram: CORBA, DCOM, XML e sockets. Objetos interagem um com o outro através da requisição e pelo fornecimento de informações. Durante uma interação, os objetos dinamicamente assumem os papéis de cliente ou servidores. A cola que une todos os objetos distribuidos é conhecido por ORB (Objetct Request Broker). Os ORBs fornecem meios para que um objeto localize e acione outros objetos em uma rede, sem restrição de endereço, sistema operacional ou linguagem de programação utilizada. O CORBA (*Common Object Request Broker Architecture*) define a interface que cada programa oferece, ou seja, funções que podem ser executadas remotamente a partir de outros aplicativos. Com esse protocolo teríamos dificuldade para manter a escalabilidade do sistema, uma vez que para novas funções criadas na aplicação CAD teríamos que atualizar tanto a aplicação servidora como a aplicação cliente, adaptando-os à nova interface. O DCOM (Distributed Component Object Model) é uma contribuição da Microsoft para o mercado de ORBs. Uma grande dificuldade é a sua complexa configuração. A linguagem de programação JAVA, desenvolvida pela SUN Microsystems, é uma linguagem de programação orientada a objetos que incorpora os conceitos de orientação a objetos. Ela combina o uso de compiladores e interpretadores de várias maneiras, tornando-a multi-plataforma e única (Vogel, 1998).

Assim, optou-se por uma forma mais simples – o socket – para enviar os comandos necessários para o servidor. No retorno, as informações são enviadas para os clientes através de um arquivo texto, que está em conformidade com o padrão XML. Com isso simplifica-se tanto a comunicação com vários usuários simultâneos como também a manipulação dos dados, e ainda permite a interoperabilidade com outros sistemas que trabalham com o XML (vide Figura 1).

2.1. Applet

Os applets, também conhecidos como miniaplicativos, são programas em Java que são executados a partir de um browser. Ao invés de chamar o programa a partir de uma linha de comando, é possível colocar no browser um endereço de uma página WEB, que por sua vez chama um ou mais applets, executando-os como se fossem programas normais. Esse processo elimina a necessidade de instalação do programa nos computadores clientes, deixando apenas os arquivos no computador servidor. Para esclarecer isso, o seguinte código html chama o applet "cliente.class" (arquivos com extensão "**`.class`**" são programas em Java):

```
<HTML>
<HEAD>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=windows-1252">
<TITLE>
Arquivo HTML chamando o Applet - Cliente.class
</TITLE>
```

```
</HEAD>
<BODY>
O Applet aparecera logo abaixo:<BR>
<APPLET
  CODEBASE = "."
  CODE     = "Cliente.class"
  NAME     = "Cliente"
  WIDTH    = 100%
  HEIGHT   = 100%
>
</APPLET>
</BODY>
</HTML>
```

2.2. OpenGL (J3D)

O OpenGL é uma biblioteca de funções para manipulação gráfica. Essa biblioteca oferece facilidades para a programação além de alto desempenho, que pode ainda ser aumentado quando utilizando hardware compatível com o OpenGL. Apesar de ser multi-plataforma, o OpenGL é uma linguagem compilada e, portanto, depende do sistema operacional. Isso significa que cada sistema operacional tem sua implementação de OpenGL. Em contrapartida, a linguagem JAVA se caracteriza por ser uma linguagem interpretada e multi-plataforma.

Para resolver esse paradoxo existem algumas implementações de APIs para o JAVA que aproveitam as bibliotecas de funções do OpenGL nativo do sistema operacional, seja ele qual for (por exemplo, o **`opengl32.dll`** do Windows). Assim, utilizando essas APIs, é possível criar um aplicativo em JAVA que é independente de plataforma e, ao mesmo tempo, utiliza o OpenGL que está instalado no sistema operacional (de forma transparente para o programador JAVA). Obviamente, é necessário que o OpenGL esteja instalado no computador. Assim, o aplicativo estará aproveitando inclusive recursos de aceleração por hardware, quando um dispositivo desse estiver presente no computador.

No início do trabalho estava sendo utilizada a biblioteca GL4Java, da Jausoft (http://www.jausoft.com/), que se mostrou muito intuitiva e apresentou um bom desempenho. No decorrer do trabalho decidiu-se por usar o J3D (http://java.sun.com/products/java-media/3D), que é uma implementação da mesma natureza do GL4Java, porém feita pela Sun. Com isso supõe-se que a implementação terá maior escalabilidade, uma vez que o produto está sendo feito por uma empresa de grande porte, criadora do Java. Além disso, o J3D oferece uma maior facilidade para programação e uma documentação mais completa.

Para utilizar bibliotecas de extensão ao JAVA, como é o caso do J3D, é preciso converter o programa listado anteriormente para utilizar o JAVA Plug-in no lugar do JVM (JAVA Virtual Machine) padrão do browser. O JAVA Plug-in permite o aproveitamento de funcionalidades que vão além da linguagem JAVA pura, como é o caso do OpenGL. O arquivo html resultante pode ser utilizado em diferentes browsers e plataformas (Bouvier, 1999; Woo, 1999).

2.3. Fluxograma da Comunicação

Mesmo definindo as tecnologias a serem utilizadas na implementação da aplicação cliente servidor, ainda é necessário definir a forma de comunicação entre a aplicação cliente e a aplicação servidora. É necessário que esta forma de comunicação seja eficiente, coerente e permita que vários usuários trabalhem simultaneamente em um mesmo projeto.

Quando o usuário aciona um comando na aplicação cliente, a aplicação servidora precisa processar o comando e avisar que houve uma mudança na estrutura de dados para todas as aplicações cliente. Em nossa implementação, a transferência da nova estrutura de dados é feita por meio de um arquivo XML. Cabe à aplicação servidora notificar a todos as aplicações clientes de que o arquivo com a nova estrutura está disponível. Enquanto processa um determinado comando, a aplicação servidora precisa também bloquear as linhas de comando para que um segundo usuário não envie um novo comando antes do primeiro ser processado.

Analisando a situação do ponto de vista da aplicação cliente é possível notar que existem duas situações onde haverá a comunicação. A primeira é quando o próprio usuário envia um comando, ou seja, a iniciativa de comunicação parte da aplicação cliente. A segunda situação surge quando um outro usuário envia um comando, e como conseqüência a aplicação servidora precisa notificar todas as aplicações clientes. Assim, a iniciativa de comunicação partiu da aplicação servidora, se analisarmos pelo ponto de vista de uma das aplicações cliente que não enviou o comando. Utilizando as conexões socket (TPC/IP) com um protocolo proprietário é relativamente

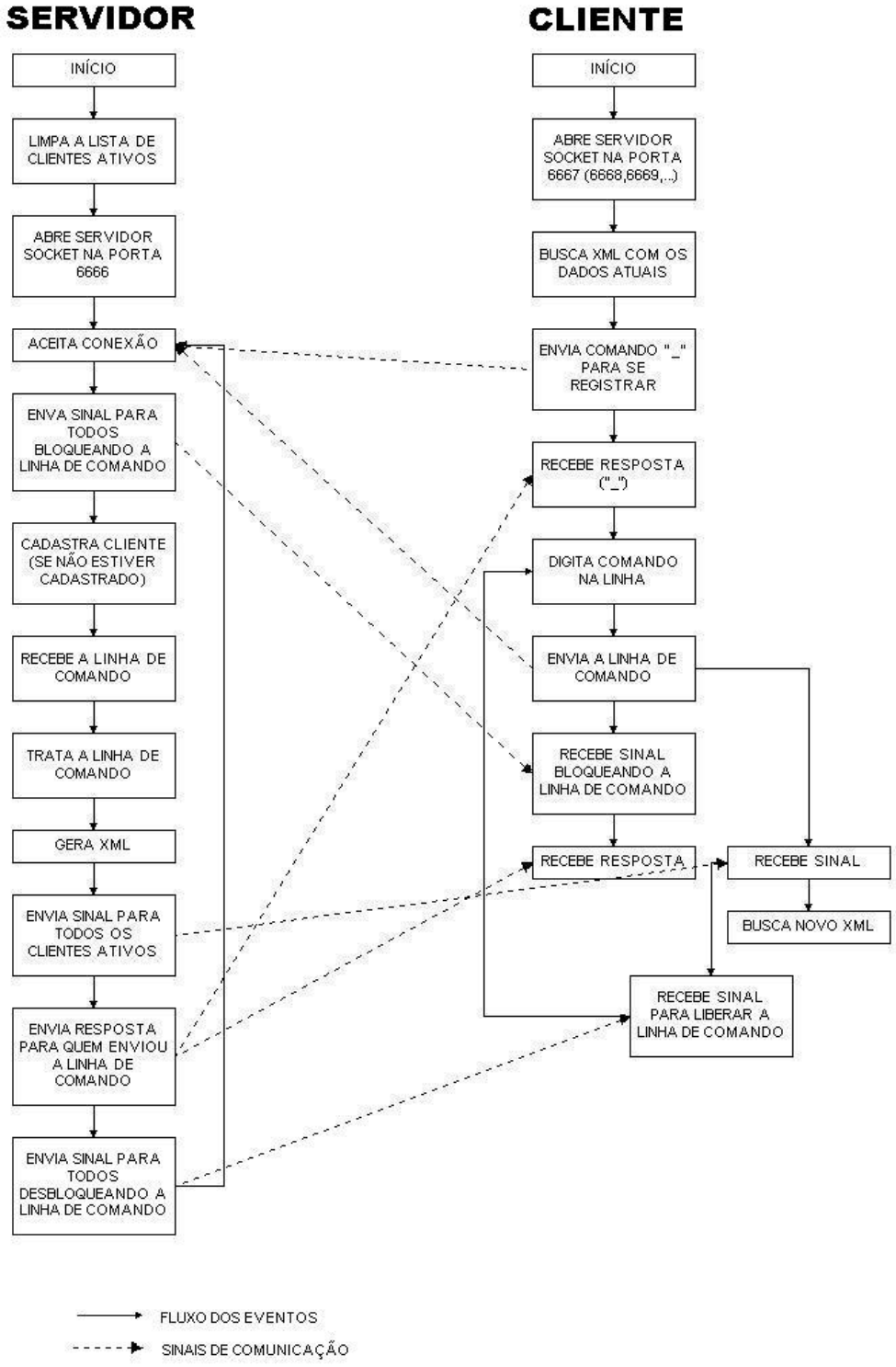

Figura 2. Fluxograma do protocolo de comunicação entre o cliente e o servidor.

simples implementar uma solução para ambas as situações. A aplicação servidora possui uma porta socket aguardando conexões de aplicações cliente, por onde serão enviados os comandos. Cada aplicação cliente cria uma conexão para enviar comandos, esta conexão recebe uma resposta sobre o sucesso no processamento do comando enviado. Além disso, uma segunda porta é criada em cada aplicação cliente, através da qual a aplicação servidora se comunica com as aplicações cliente bloqueando a linha de comando, notificando alterações na estrutura de dados, e qualquer outra notificação que seja necessária e que se enquadre na segunda situação discutida acima, onde a iniciativa de comunicação parta da aplicação servidora. Após receberem os sinais, cada uma das aplicações cliente busca o arquivo XML que foi gerado pela aplicação servidora e que contém todos os dados necessários para a visualização dos objetos na tela. A Figura 2 ilustra o fluxograma.

3. IMPLEMENTAÇÃO

Na solução aqui apresentada a aplicação servidora é o USPDesigner. O USPDesigner é um modelador de sólidos B-Rep, Mäntylä (1988), que está sendo implementado segundo a linguagem C++ (existe uma versão anterior que foi escrita em C). Esta nova versão compreende na remodelagem para permitir um melhor desempenho do sistema utilizando o OpenGL e facilitar a implementação de novas funções, dentre as quais podemos citar o suporte para WEB.

Para que o USPDesigner se conecte à WEB, as adaptações que estão sendo feitas são: geração de um arquivo XML a cada alteração na estrutura de dados, o envio de mensagens de aviso para todas as aplicações cliente ativas sempre que houver alguma mudança, e ainda a aceitação de comandos via conexões socket.

3.1. Aplicação Cliente

A aplicação cliente deve oferecer uma interface adequada para que o usuário possa utilizar o sistema com certa facilidade. Basicamente essa interface consiste em um mostrar o sólido na tela, além de ter uma linha de comando onde o usuário deverá digitar os comandos que serão processados pela aplicação servidora. Para o desenvolvimento do programa cliente está sendo utilizado o JBuilder4 Foundation, que pode ser obtido gratuitamente na Internet (http://community.borland.com/). Ainda, como principal fonte de informações sobre a linguagem Java, está sendo utilizado o tutorial fornecido pela Sun, que explica em detalhes e até exemplifica os principais componentes da linguagem (http://java.sun.com/).

Uma vez que o JAVA é uma linguagem orientada a objetos, foram criadas algumas classes representando as entidades necessárias para a execução do programa. A seguir está um breve descritivo de cada classe:

Cliente: Essa é a classe que cria a janela principal do programa, além de instanciar e chamar funções das outras classes.

ServidorSocket: Essa classe cria o servidor socket na aplicação cliente. Esse servidor é utilizado para que a aplicação cliente receba notificações sobre mudanças na estrutura de sólidos (para que busque o arquivo XML com a nova estrutura de dados) e outros comandos tais como bloqueio e desbloqueio da linha de comando.

ClienteSocket: Essa classe é instanciada cada vez que o usuário digita um comando. A classe cria uma conexão com a aplicação servidora e envia a linha de comando, recebendo um retorno sobre o sucesso/insucesso da operação.

ClienteXML: Responsável por buscar o arquivo XML e gerar, a partir desse, a nova estrutura que será mostrada e manipulada na tela da aplicação cliente.

Propriedades: Abre uma caixa de diálogo com as opções de visualização do sólido.

Ajuda: Abre uma caixa de diálogo com textos dando um auxilio ao usuário como, por exemplo, o uso do mouse para mover e rotacionar objetos.

3.2. Seleção de Entidades

Na implementação de aplicações em Fortran, o uso de variáveis de tipos inteiro e de ponto flutuante, era um fato muito comum. No desenvolvimento de aplicações voltadas para a Internet, o tipo de variável que deve ser o mais utilizado é o tipo String. Este fato também ocorre em nossa implementação, permitindo uma facilidade na transferência da informação em ambientes multi-plataforma e multi-linguagem. Assim, todo comando executado pelo usuário na aplicação cliente deve ser transformado em uma String que será enviada para a aplicação servidora pelo socket.

A aplicação servidora receberá a String, interpretará a cadeia de caracteres recebida e executará o comando. Entretanto, vários comandos necessitam que parâmetros sejam fornecidos, para a correta execução dos comandos. Desta maneira, a aplicação cliente não necessita possuir uma estrutura de dados completa, pois a consistência dos dados será feita pela aplicação servidora.

É necessário que a aplicação cliente possua informações suficientes para exibir os sólidos que estão sendo manipulados. O que corresponde a agrupamentos de faces, cada face corresponde a um agrupamento de laços e cada laço corresponde a um agrupamento de vértices. Devido à forma como os elementos são armazenados no OpenGL, é conveniente fornecer também a lista de arestas e a lista de vértices de cada sólido. Cada elemento possui o seu identificador que deve ser fornecido como parâmetro para a aplicação servidora, se for o caso.

3.3. Arquivo XML

O XML (eXtensible Markup Language) é uma meta-linguagem, Autodesk (2001). O seu grande sucesso se deve a sua simplicidade e escalabilidade. Aqui explicaremos sobre como o foi utilizado neste projeto. Para isso está colocado a seguir o cabeçalho do arquivo (DTD – Data Type Definition), que define o tipo de dados que o arquivo deve conter, com uma breve explicação (vide Tabela 1). Neste exemplo foram listados apenas alguns dos itens disponíveis.

Tabela 1 – DTD utilizado no projeto

`<?xml version="1.0"?>`	Define a versão da meta-linguagem (atualmente 1.0)
`<!DOCTYPE TUDO[`	Inicia definindo que o conteúdo do documento é um objeto do tipo “TUDO”
`<!ELEMENT TUDO (ARESTA)*>`	Define o objeto do tipo “TUDO” é formado por várias “ARESTA”s (o asterisco significa vários)
`<!ELEMENT ARESTA (PONTO,PONTO)>`	Cada aresta é formada por dois “PONTO”s
`<!ELEMENT PONTO (X,Y,Z)>`	Cada ponto é formado por três elementos: “X”, “Y” e “Z”
`<!ELEMENT X (#PCDATA)>`	Elemento “X” contém dados
`<!ELEMENT Y (#PCDATA)>`	Elemento “Y” contém dados
`<!ELEMENT Z (#PCDATA)>`	Elemento “Z” contém dados
`]>`	Fim do DTD (Data Type Definition)

Após esse cabeçalho são colocados os dados, que em nosso caso são as informações sobre as coordenadas das arestas. O exemplo da Figura 3 mostra um arquivo contendo uma aresta entre os pontos `(0,0,0)` e `(1,1,1)` e outra aresta entre os pontos `(0,-2,3)` e `(2,4,2)`. Comentários não foram colocados por ser de fácil entendimento:

```
<?xml version="1.0"?>
<!DOCTYPE TUDO[
<!ELEMENT TUDO (ARESTA)*>
<!ELEMENT ARESTA (PONTO,PONTO)>
<!ELEMENT PONTO (X,Y,Z)>
<!ELEMENT X (#PCDATA)>
<!ELEMENT Y (#PCDATA)>
<!ELEMENT Z (#PCDATA)>
]>

<TUDO>
```

```
<ARESTA>
 <PONTO>
  <X>0</X>
  <Y>0</Y>
  <Z>0</Z>
 </PONTO>
 <PONTO>
  <X>1</X>
  <Y>1</Y>
  <Z>1</Z>
 </PONTO>
</ARESTA>
<ARESTA>
 <PONTO>
  <Z>3</Z>
  <X>0</X>
  <Y>-2</Y>
 </PONTO>
 <PONTO>
  <Z>2</Z>
  <X>2</X>
  <Y>4</Y>
 </PONTO>
</ARESTA>
</TUDO>
```

Figura 3. Exemplo de arquivo XML.

4. RESULTADOS

A seguir são exibidas duas telas da aplicação cliente. Na Figura 4, o AntiAliasing do OpenGL está ligado. Na Figura 5, o Antialiasing não está selecionado.

5. CONCLUSÕES

Nesse trabalho foi desenvolvida uma interface para o USPDesigner, que é um modelador de sólidos B-Rep, programa CAD que pode ser utilizado via a Web, no modelo ASP. Foram criadas as

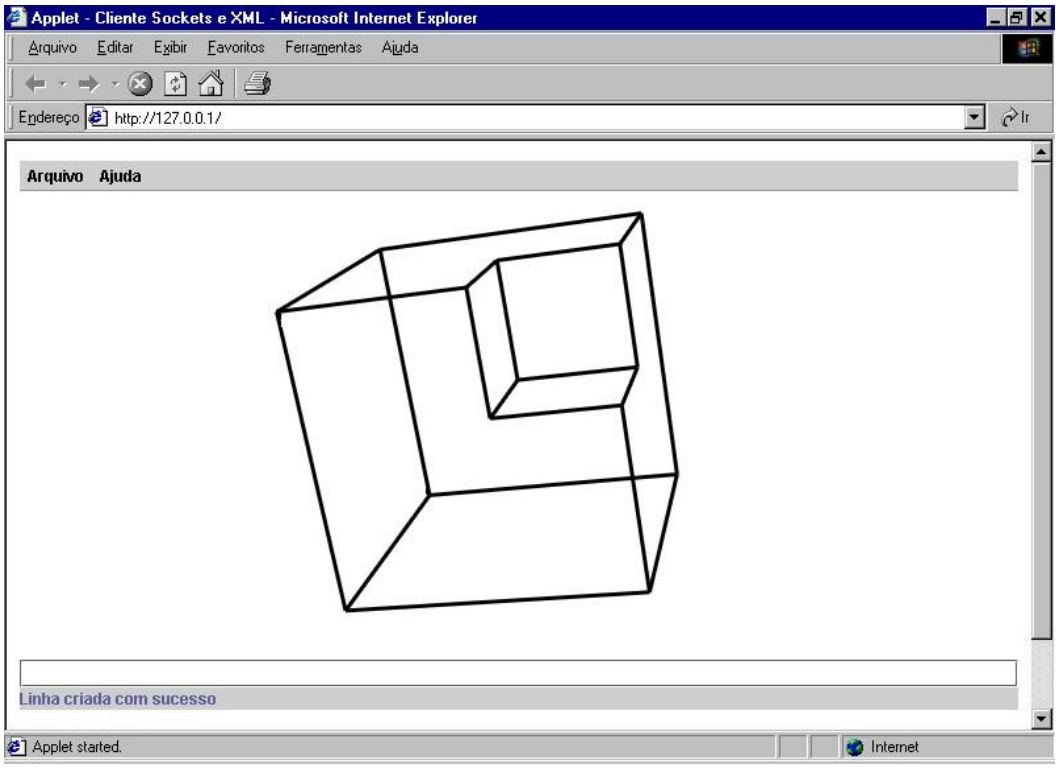

Figura 4. Sólido sendo exibido na aplicação cliente com Antialiasing.

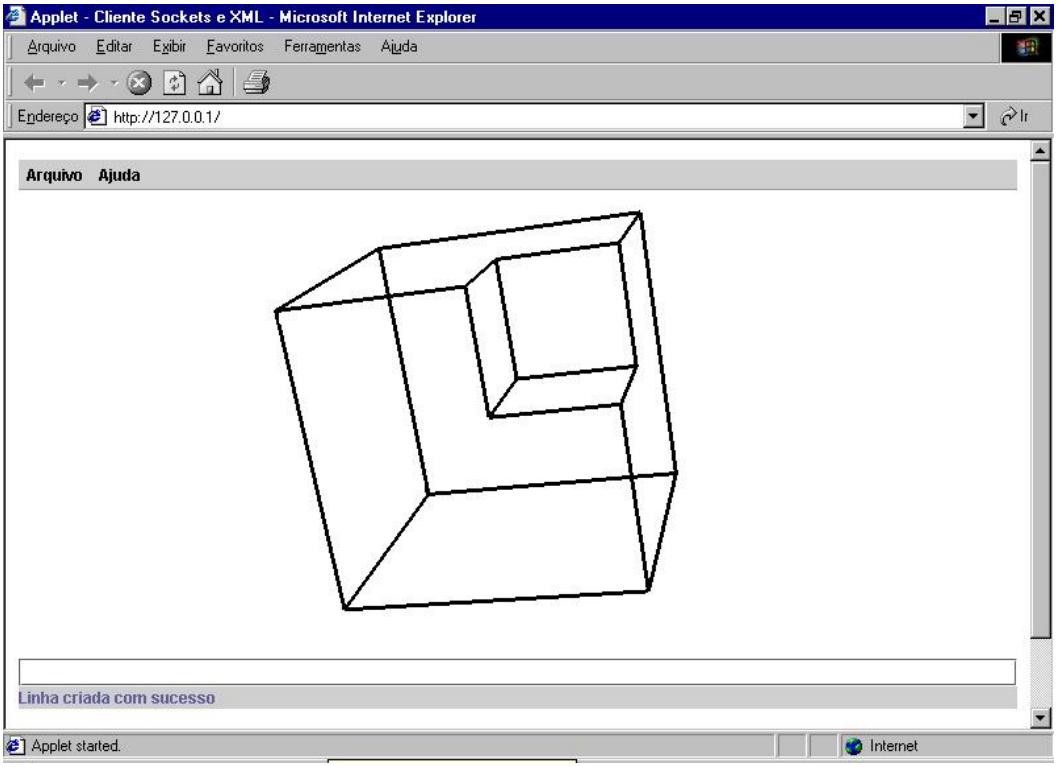

Figura 5. Sólido sendo exibido na aplicação cliente sem Antialiasing.

funcionalidades básicas para uso do modelador, sendo que a arquitetura da solução é bastante flexível, permitindo a ampliação de tais funcionalidades no futuro.

Espera-se poder utilizar esta nova versão do USPDesigner nas atividades didáticas da disciplina de CAD/CAM. Uma das atividades é a definição de um algoritmo para cálculo de volume, centro de gravidade e matriz de inércia de sólidos. Sabe-se que para este cálculo é suficiente ter-se uma representação por superfícies desde que a consistência dos dados seja mantida. Que é o nosso caso, pois a consistência dos dados será mantida pela aplicação servidora. Assim, a aplicação cliente está funcionando como um simplificador de estrutura, permitindo que o trabalho seja mais focado.

6. AGRADECIMENTOS

O autor foi parcialmente suportado pelo CNPq – proc. 300.224/96-6. Este projeto foi suportado pela FAPESP – proc. 99/12276-1.

7. REFERÊNCIAS

Autodesk, 2001, White Paper: The XML Revolution.

Bouvier, Dennis J., 1999, (K Computing) - Getting Started with the Java 3D™ API, © Sun Microsystems, Inc.

Hall, Brian Beej, 2001, Beej's Guide to Network Programming (Using Internet Sockets), Revision Version 2.3.0 July 25.

Mäntylä, M., 1988, An Introduction to Solid Modeling, Computer Science Press.

Woo, Neider, Davis & Shreiner, 1999, OpenGL Programming Guide, Addison-Wesley Publishing Company, Massachusetts.

Vogel, Andreas; Duddy, Keith, 1998, Java programming with CORBA, 2nd ed. New York : Wiley computer publishing.

8. DIREITOS AUTORAIS

Os autores são os únicos responsáveis pelo conteúdo do material impresso incluído neste trabalho.

TITLE: CAD SYSTEM FOR THE WEB

Nelson Vogel
Marcelo Shimada
Marcos de S. G. Tsuzuki
Escola Politécnica da Universidade de São Paulo
Departamento de Engenharia Mecatrônica e de Sistemas Mecânicos
CEP 05508-900, São Paulo, SP, Brasil. E-mail: mtsuzuki@usp.br

Abstract. The thin client architecture systems, also known as three-tier architecture, are being used more and more over the years. This is happening because of its features that facilitate the system's maintenance, in both hardware and software. If there is a need to increase the system processing capacity, it is enough (in first instance) to increase the capacity of the server machine. Each new improved version should be available only in the server machine. These features are also desirable in CAD systems. In this work we have modified a B-Rep Solid Modeler architecture to allow its execution in a three-tier environment. The ASP (Application Service Provider) model, which is being highly spread out, was used to achieve that. This model allows a software to be rented instead of being sold. The result is a CAD program that wheel in a server, and which can be accessed (used) from a browser in a local area network (LAN) or the Internet. The server application includes a B-Rep Solid Modeler, developed in C++ programming language. The client side application will have a user interface and is implemented in JAVA. To show objects three-dimensionally it uses the OpenGL. The client will have a data structure mapping, enough to carry through selections and to show the solid in wireframe. Beyond the open JAVA language, other technologies such as the XML and socket (TCP/IP) are being used, guaranteeing the scalability and portability to many platforms. The use of an open architecture is a world-wide trend in the Information Technology business, which leads us to believe that this will be also a trend for CAD applications in the near future.

Keywords: CAD System, OpenGL, Client-Server