

# 一个基于 WEB 的分布式仿真环境\*

党岗 凌云翔 胡华平 金士尧

(国防科技大学计算机系 长沙 410073)

**摘 要** 现有的许多仿真环境,由于采用某一种语言或基于库方法开发,缺乏对已有仿真模型和仿真工具复用的能力,且依赖于特定的操作系统和硬件平台,尤其缺乏在 Internet 上运行的能力。本文针对这些不足,采用软件复用和构件技术、Java 语言和 CORBA 标准相结合的方法,提出了一个基于 Web 的分布式仿真环境框架(DSEW),并进行了实例分析。

**关键词** Web, 分布式仿真, Java, CORBA, 仿真模型

**分类号** TP391.9

---

## A Distributed Simulation Environment Based on Web

Dang Gang Lin Yunxiang Hu Huaping Jin Shiyao

(Department of Computer, NUDT, Changsha, 410073)

**Abstract** Nowadays, since the a language-based or the library-based approach has been adopted in simulation environment, the reuse existing simulation models and tools, cannot be reused. Besides, they depend on the operating systems and hardware platforms. Especially they are lack of capability to operate over the Internet and the Web. Aiming these disadvantages puts forward a prototype of distributed simulation environment based on Web (DSEW) and makes a case study as well.

**Key words** Web, distributed simulation, Java, CORBA, and simulation model

---

在不断增长的网络应用中,仿真行为模型的建立、评价和改进并不是一件容易的事,因为对于一个复杂系统进行有效而简单的描述是十分困难的,从而能否充分利用已有的或可以找到的模型和仿真工具,是减少仿真建模工作量、提高仿真效率的关键因素之一。现有的很多仿真软件开发环境不仅缺乏这种模型和仿真工具复用的能力,而且缺乏互操作性,仿真应用组件不能用不同的语言编写,或不能在不同的操作系统和硬件平台上运行<sup>[1]</sup>,并且还缺乏在 Internet 和 Web 上运行和进行高效远程通讯的能力<sup>[2]</sup>。Web 技术与仿真技术的结合,为此提供了新的解决方案,产生了一种新的仿真建模方法学,并已渐渐成为仿真的一个新兴领域。

### 1 基于 Web 的仿真建模方法

基于 Web 的仿真建模方法主要体现在分布合作的模型建模、动态多媒体文件、数据和模型的感知和搜集,以及模型的分布仿真执行和分析等几个方面。这种方法主要是由 Java 在模型描述和实现中所处的地位决定的。Java 语言十分适用于实现一个高级仿真系统,它使用方便,可使模型描述与实现分离,易于实现模型复用。Java 的某些特征,如易于设计的面向对象特点和对多线程执行的支持,使 Java 具有专用仿真建模语言的重要特征,易于实现高级模型描述。特别是,它使用 JavaBeans 的组件开发模型,这种简单的编程约定允许基于 Java 的软件组件集成或应用,具有很强的可视化能力,适用于现在流行的各种图形程序设计环境。

这种基于 Web 仿真的观点是在的 SilkTM 设计后推动的。Silk 是一个商用的面向过程的离散事件仿真软件包<sup>[3]</sup>,它由一组 Java 功能类组成,融合了常用的面向进程的建模方法和面向对象方法,利用

---

\* 1999 年 1 月 20 日收稿

第一作者:党岗,男,1973 年生,博士生

Silk 建模组件开发和可视化集成, 使模型建模简化, 具有可复用性, 易于扩展模型组件。

基于 Web 的仿真环境研究现在刚刚起步, 还很不完善。因此, 开发一个基于 Java 的通用仿真底层组件类库, 并提供强大的建模和可视化机制, 显得十分有意义。

## 2 综合 Java 和 CORBA 技术的仿真

如果用户在建模时可以重用网络上已有的模型组件, 并在其它结点利用本地没有的仿真工具以执行仿真任务, 则开发一个仿真应用的工作量可以大大减小。但是这种仿真任务仅仅用 Java applet 就显得力不从心了, CORBA 可以弥补 Java 的不足。CORBA<sup>[6]</sup> 技术是一个标准的支持网络应用、分布计算、异构环境的框架, 它可实现 Internet 上客户端不同语言、不同操作系统和硬件平台的应用。CORBA 技术的一个主要方面在于客户端和服务端被一个友好的界面分开, 它允许客户端高效地服务服务端, 而不需要知道传输协议或服务端的实现细节。Java 主要依靠 HTTP/CGI 服务器驱动, 它本身不是一个 ORB (Object Request Broker), 不能提供一个分布对象的传输通路, 它需要类似 CORBA 的机制来实现与网络上存在的应用进行对话, 并为 Java 组件提供通信, 而且, Java 不能提供类似 CORBA 的分布对象服务。另外, CORBA 可以提供可伸缩的服务器到服务器的基础结构, 对象利用 CORBA ORB 进行通信, 这样可以均衡来自客户机的请求负荷, 而 CGI 不能, 会产生瓶颈。所以, 仅有 Java 还远远不够。Java 和 CORBA 的结合使其具有 Java 的简捷性, 又具有 CORBA 的通信分布和应用。

Java 和 CORBA 的结合十分适用于分布式仿真应用。在客户端, 利用 JavaBeans, 将仿真应用嵌入到可移动容器 (HTML 或 Jars) 中, 用户用拖放操作来和这些对象进行交互。CORBA IDL 描述了仿真对象、对象间的通讯和事件机制等操作, IDL 界面定义可以用 Java、C++ 或其它语言, 并通过使用不同的仿真算法, 在不同的操作系统和硬件平台上实现。Java 开发的仿真应用, 在 Web 浏览器上运行, 并由 IIOP 访问设备, 以获得模型执行的运行时间支持。

采用这种方法, 本文针对分布式仿真应用, 提出了一个基于 Web 的分布式仿真环境 (DSEW) 框架, 它具有可重用、可扩展、互操作性好、易于远程通讯、可视化能力强等特点。

## 3 DSEW: 一个基于 Web 的仿真环境

DSEW 是一个基于 Java/CORBA 的分布式仿真环境 (如图 1 所示), 它包括以下几部分:

(1) Web 客户端浏览器 (WCB): 用 Java applet 实现, 从 Web 仿真服务器上下载, 并在客户端结点的 Web 浏览器上应用;

(2) Web 仿真服务器 (WSS): WSS 为客户端应用提供基本的仿真服务, WSS 不同于传统的仿真服务器, 它是一个由一组基于 JavaBeans 的功能组件构成的仿真服务支撑环境;

(3) 仿真工具服务器 (STS): 每个 STS 可运行一个特定仿真工具;

(4) 仿真模型服务器 (SMS): 每个 SMS 有一个或多个仿真模型或子模型组件;

(5) 基于 CORBA 的底层框架 (CORBA IIOP): 用于实现数据在 Internet 上各种 ORB 的通讯。

### 3.1 客户端应用

Web 客户应用通过 Web 浏览器访问 Web 仿真服务器上的组件, 下载客户应用程序, 并在 Web 客户浏览器上运行。它根据不同的内容和要求调用相应的 HTML 页面, 请求得到相应的服务和信息, 包括模型构造表的创建、仿真执行过程的可视化和动画表现、以及仿真结果的可视化表现和分析等。

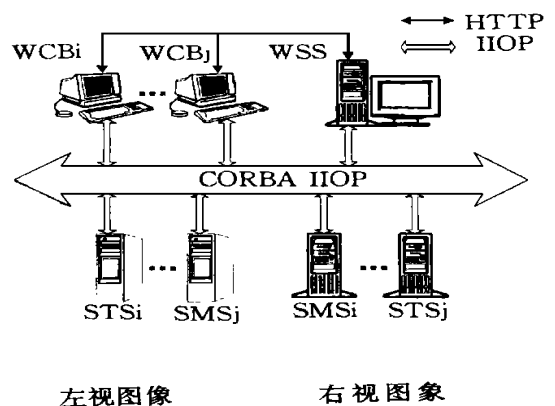


图 1 DSEW 基本结构框架

Fig. 1 Structure of DSEW

### 3.2 Web 仿真服务器(WSS)

Web 仿真服务器(WSS)提供一个 URL 下载 Java 脚本以实现客户应用。DSEW 用户连接 Web 仿真服务器,请求仿真服务,下载客户应用,并在自己的结点通过 Web 浏览器运行。这里的 Web 仿真服务器不同于传统意义上的仿真服务器,事实上 WSS 即可以作为服务器端,又可以作为客户端,它是一个基于组件的 Web 仿真支撑环境,组件用 JavaBeans 构造。根据用户需求将 WSS 上的 Web 仿真支撑环境(W SSE)的组件划分为:

(1) 基本组件(BasicComp):基本组件包括一些通用的机制,如初始化,时间控制,资源状态等。

(2) 模型开发组件(ModelDevelop):在模型开发组件中,模型开发者可创建新的模型或选择一个已存在的仿真模型。在这一组件中,开发者定义模型的输入输出接口,如输入参数或数据、输入文件、输出文件、动画跟踪文件等。在模型定义后,模型开发者还需定义仿真系统,模型开发组件将提供一组仿真系统的选择。

(3) 可视化表现组件(VisualPresent):可分为文本图形表现和动画表现两方面。文本图形表现是采用文本和静态图形的形式表现仿真应用,多用于仿真结果的可视化,使用 CGI script 技术通过 HTML 页面表现。动画表现是基于 Java 的二维或三维动画系统实现,表现仿真应用的动态情景,多用于仿真执行过程的可视化,例如在仿真过程中各状态变量的变化等等。

(4) 交互控制组件(InteractControl):交互控制组件提供了一系列为完成交互任务产生的对话框,例如模型开发者能很容易地通过简单的鼠标操作创建新的描述输入表,同时交互控制组件还提供了错误处理、执照申请、结果输出等信息。

(5) 数据库访问组件(DBAccess):数据库访问组件主要用于输入和输出结果数据的记录和管理。同时,数据库访问组件还提供了输入信息和输出数据的比较功能,根据输入信息调用相应的组件,以及相同模型的不同仿真结果的比较,并将结果数据送给用户或仿真专家系统。

(6) 结果分析组件(ResultAnaly):结果分析组件包括一系列统计程序,计算并统计仿真结果数据。此外,还有支持仿真专家系统的组件和协作支持组件等。

### 3.3 仿真工具服务器(STS)和仿真模型服务器(SMS)

SMS 和 STS 可被看成已经存在的组件封装进 DSEW, CORBA 提供了一个标准的面向对象方法用于封装继承资源。当根据 CORBA 标准实现时,容易封装新的 SMS 和 STS,并可重用,因而 DSEW 提供了即插即用的功能。

仿真模型(SM)和仿真工具(ST)接口是用 CORBA IDL(Interface Definition Language)描述的,它的描述与实现的语言、位置、硬件、软件结构,以及网络技术无关。IDL 提供了用来定义 SM 和 ST 封装要求服务的基本的结构化的数据类型和操作,IDL 编译器可将 IDL 映射到最常用的语言(如 C, C++, Java),IDL 定义的封装被编译成 Java 代码,然后分派到服务器组件和客户组件。SMS 和 STS 接口包含一些描述本机所有仿真模型消息的代码和一些描述本机所有仿真工具的代码,封装成一个对象,当发现所需要仿真模型或子模型和仿真工具,则通过网络获得,并运行相应的仿真模型。SMS 和 STS 与客户应用间透明的通讯,是通过图示的 CORBA IIOP 实现的。IIOP(Internet Inter-ORB Protocol)定义了 GIOP(General Inter-ORB Protocol)消息怎样在 TCP/IP 上传输,从而使数据可以在 Internet 上实现各种 ORB 的通讯。

## 4 实例分析

例如一个在结点机 A 的仿真应用,希望开发一个大型仿真模型 SM,但结点机 A 没有所需要的子模型 SBM1、SBM2 和适合 SM 运行的仿真工具 ST,则该用户通过 Web 发现所需要的子模型 SBM1 在结点机 B,子模型 SBM2 在结点机 D,仿真工具 ST 在结点机 C。要完成这样的仿真任务,首先用户在结点机 A 感知 SBM1、SBM2 和 ST 在网络的什么地方,并分别从结点机 B、D 下载子模型到结点机 A,组合 SBM1、SBM2 到 SM 中,然后访问结点机 C 的 ST 执行模型 SM 的仿真,最后将仿真结果和分析数据送到结点机 A。

#### 4.1 IDL 描述

图 2 例示了仿真模型服务器(SMS)和仿真工具服务器(STS)通过 CORBA IIOP 与 Web 仿真服务器之间进行通讯的 IDL 接口文件。

SM 的接口提供两个例外: No\_Models, 报告该仿真模型服务器中没有模型索引; Model\_NotFound, 报告该仿真模型服务器中找不到所需要的模型组件。同时, SM 的接口提供两个操作: GetModelIndex, 获得该仿真模型服务器中的模型索引; GetModel, 获得所需要的模型组件。

ST 的接口提供一个例外: Fault\_information, 报告仿真出错信息。同时, ST 的接口提供一个操作: Execution\_sim, 执行仿真任务, 并报告仿真结果。

图中所示的 IDL 描述通过 IDL 到 Java 的编译器, 生成客户端(Sub)和服务端(Skeleton)的接口文件。客户端的 Sub 负责收集实现客户应用的 Java Applet 的请求, 这些请求通过 CORBA IIOP 发送到服务器端。服务器端的 Skeleton 负责接收这些请求, 激活 SM 和 ST 服务器, 并将结果返回客户端。

```

module DSEW {
    const long MAX_LENGTH_NAME= 20;
    struct infos {
        string< MAX_LENGTH_NAME> Modelname;
        string description;
    };
    interface SimModels {
        exception No_Models{};
        exception Model_NotFound {};
        string GetModelIndex (in string host_name,
            in string path) raises (No_Models);
        infos GetModel(in string host_name, in string path,
            in string model_name) raises (Model_NotFound);
    }; end of interface SimModels
    interface Simtools {
        const long RESULT_SIZE= 100000;
        typedef string< RESULT_SIZE> results_report;
        exception Fault_information{};
        results_report Execution_sim(in infos modelinfo)
            raises (Fault_information);
    }; end of interface Simtools
end of DSEW module
  
```

图 2 SM 和 ST 的 IDL 描述  
Fig. 2 IDL description of SM and ST

#### 4.2 仿真实现

针对一个具体的离散事件仿真实例(如售票处问题), 实现过程如下:

(1) 提交仿真任务: 客户端浏览器访问 Web 仿真服务器主页, 通过该主页的超链接, 产生仿真事件输入表(如图 3 所示)。客户输入仿真的各种参数, 以交互方式提交仿真任务。图中所示的是针对一般的单/单或多服务模型离散事件输入表。

(2) 处理提交的仿真任务: Web 仿真服务器接收到所提交的仿真任务后, Web 仿真服务器首先在本地查找相应的仿真模型和仿真工具, 若有则直接在本地进行仿真任务, 否则在网络上寻找仿真模型和工具, 并进行仿真, 报告结果。在仿真执行中, Web 仿真服务器将通过主页以动画形式告诉客户仿真正在进行。

(3) 仿真结果表现: 当仿真任务完成后, Web 仿真服务器将仿真结果以文本和可视化形式送到客户浏览器上。使结果尽可能用图形或动画的形式表现, 便于客户进行评价。

图 3 仿真事件输入表  
Fig. 3 Input table of simulation event

#### 5 结束语

本文实现的 DSEW 仿真环境是一个原型框架, 还有许多方面需要改进和完善, 如模型组件的组织、动画机制的扩充等。特别是, 用 VRML 语言对该环境的扩充, 可以使 DSEW 具有更广阔的应用前景。此外, 由于 CORBA 实时性的缺乏, 导致 DSEW 的实时性不能得到充分保证, 这是下一步要解决的问题。

#### 参考文献

and Simulation, January 11-14, 1998, San Diego, CA, USA

2 Giuseppe Lazeolla and Andrea D'Ambrgio. A Web-Based Environment for the Reuse of Simulation Models. 1998 International Conference on Web-based Modeling and Simulation, January 11-14, 1998, San Diego, CA, USA

3 The Silk Homepage. Introduction to Silk. 1997, <http://www.threadtec.com>

4 Object Management Group. The Common Object Request Broker: Architecture and Specification (Revision 2.0). July 1995

(上接第79页)

$$E^2(z) = \prod_{i=1}^M \prod_{j=1}^M \alpha^2 t_{L+i} t_{L+j} / M^2$$

$$D(z) = E(z^2) - E^2(z) = (1 + 2\alpha^2) \prod_{i=1}^M t_{L+i}^2 / M^2$$

$$\text{故 } z \sim N\left(\prod_{i=1}^M \alpha t_{L+i} / M, (1 + 2\alpha^2) \prod_{i=1}^M t_{L+i}^2 / M^2\right).$$

3. 待检验图像利用 Piva 算法嵌入了序列  $X$  ( $X$  与检测时用的  $X$  不同)。

此时  $z = \prod_{i=1}^M x_i t_{L+i} / M = \prod_{i=1}^M x_i (t_{L+i} + \alpha t_{L+i} x_i) / M$ , 其中  $\{x\}$  与  $\{x\}$  不相关。

$$E(z) = 0$$

$$E(z^2) = E\left(\prod_{i=1}^M \prod_{j=1}^M x_i t_{L+i} x_j t_{L+j} / M^2\right) + E\left(\prod_{i=1}^M \prod_{j=1}^M \alpha x_i t_{L+i} t_{L+j} x_j\right) / M^2$$

$$+ E\left(\prod_{i=1}^M \prod_{j=1}^M \alpha x_j t_{L+j} t_{L+i} x_i\right) / M^2$$

$$+ E\left(\prod_{i=1}^M \prod_{j=1}^M \alpha^2 t_{L+i} t_{L+j} x_i x_j\right) / M^2 = (1 + \alpha^2) \prod_{i=1}^M t_{L+i}^2 / M^2$$

$$\text{故 } z \sim N\left(0, (1 + \alpha^2) \prod_{i=1}^M t_{L+i}^2 / M^2\right)$$