ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.17, No.3, March 2006, pp.546−558
DOI: 10.1360/jos170546

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

基于构件的分布式虚拟现实应用系统*

段作义+, 吴 威, 赵沁平

(北京航空航天大学 计算机学院,北京 100083)

Component-Based Distributed Virtual Reality Application System

DUAN Zuo-Yi+, WU Wei, ZHAO Qin-Ping

(School of Computer Science and Engineering, BeiHang University, Beijing 100083, China)

+ Corresponding author: Phn: +86-10-82317109 ext 805, Fax: +86-10-82317644, E-mail: duanzy@vrlab.buaa.edu.cn

Duan ZY, Wu W, Zhao QP. Component-Based distributed virtual reality application system. *Journal of Software*, 2006,17(3):546−558. http://www.jos.org.cn/1000-9825/17/546.htm

Abstract: This paper presents an approach of constructing the application system in Distributed Virtual Reality. Based on a component model of application system in which three types of the basic components are discussed, this paper first focuses on the "semantic" interoperability among components, and presents a Primitive-Semantic component constructing method through a semantic model to deal with the semantic interoperability while the communication protocol among components doesn't exist. It then presents a composable Distributed Virtual Reality Component-Based Architecture, including its key mechanism and algorithms. Finally the experimental result demonstrates the usability of this architecture.

Key words: distributed virtual reality; component; architecture; composability; interoperability

摘 要: 针对分布式虚拟现实应用系统的构建问题,给出一种分布式虚拟现实应用系统的构造方法.指出分布式虚拟现实系统中存在的 3 种基本构件;重点讨论构件的"语义"互操作问题,通过建立构件的语义模型,提出一种原语-语义构件的构件构造方法,该方法可以解决不存在通信协议情况下的构件之间"语义"互操作问题;给出一种支持可组装的分布式虚拟现实应用系统的基于构件的体系结构以及其中的核心机制和算法.实验结果表明了该体系结构的可用性.

关键词: 分布式虚拟现实;构件;体系结构;可组装性;互操作性
中图法分类号: TP393 文献标识码: A

分布式虚拟现实是在先进的网络平台上,将孤立的或小范围的虚拟现实系统连接起来,使处于不同地域的多个用户可以在同一个虚拟的世界中进行实时交互,协同完成各种任务[1].主要应用有分布式虚拟环境(distributed virtual environment,简称 DVE)、分布式交互仿真(distributed interactive simulation,简称 DIS)、远程沉浸(remote immersion,简称 RI)等.通常在一个分布式虚拟现实应用中,每个节点都是一个复杂的多通道人机交互系统,包括多通道人机交互输入、实时的 3D 图形绘制、3D 声音输出、力/触觉反馈输出和问题域内的计算;

* Supported by the National Grand Fundamental Research 973 Program of China under Grant No.2002CB312105 (国家重点基础研究发展规划(973))

Received 2005-06-17; Accepted 2005-12-01

各个节点则通过某个底层分布式运行时系统(runtime infrastructure)连接起来,它主要负责管理节点之间的通信,目前,HLA/RTI(high level architecture/runtime infrastructure)已成为国际上主流的分布式运行时规范.

除制定分布式运行时规范以外,构建分布式虚拟现实应用系统也是一个重要问题.国际上的趋势[2−6,8]倾向于利用构件技术建立“可组装”的软件系统.近年来,针对分布式计算的复杂软件系统的可组装(composable)问题越来越受到人们的重视[2],但可组装性(composability)也给软件系统的构造带来了新的需求.与此同时,分布式虚拟现实作为分布式计算的一个分支,其软件系统也面临着互操作性(interoperability)的问题,这意味着必须存在一个互操作标准.互操作性要求构件之间不仅能在“语法”层面上互操作,更重要的是还应该能在“语义”层面上互操作.

我们同意文献[2]给出的可组装性定义:可组装性是一种允许根据用户特定需求从已存在构件中选择并以各种不同方式装配成一个软件系统的能力.考察可组装性共同的性质[3]:(1) 可组装性意味着存在一个基于构件的结构.显然,可组装性要求可以利用已存在的构件装配成应用;(2) 可组装意味着软件系统的各个组成部分都是可互操作的.构件不能独立工作;(3) 可组装意味着构件是可重用的.这一点也将可组装和单纯模块的概念区别开来[3].有 9 个层次的可组装问题[2],对于分布式虚拟现实应用系统来说,有两个层次的可组装问题非常重要:一个是在分布式计算环境中,不同节点运行的进程需要“组装”起来共同完成一个计算任务,该层次的组装问题对应于文献[2]的“Federate”组装层次,即本文的“分布”方向的组装问题;另一个是指每个节点运行的应用是通过预先设计好的构件“组装”起来的,该层次的组装问题对应于文献[2]的“Module”层次,即本文的“进程”方向的组装问题.文献[4]给出了一个支持可组装性的分布式系统原型结构 ModISE,并讨论了相关工具,但并未对构件之间“语义”互操作问题作详细讨论;文献[5,6]用形式化的方法对可组装性理论进行了初步的研究;文献[7]给出了一个建立在 HLA 声明管理基础上的“虚通道”,通过虚通道来连接不同的 LP,并讨论了不同 LP 之间的“同步”问题,由于其与 HLA 密切相关,从而限制了该方法的应用;OneSAF[8]是目前很典型的一个可组装的系统,是美国军方下一代计算机生成兵力系统,但也是一个基于 HLA/RTI 的系统;Simulation Component Model(SCM)[9]是一个基于 OMG 的 CCM(CORBA component model)的结构模型,它主张把构件行为代码和与运行时的结合代码分隔开来.

相关工作[4−9]讨论了本文所涉及问题的不同侧面,但并未从整体上论述分布式虚拟现实应用系统的构建问题.我们认为有两个关键点在这个问题中尤其重要:(1) 如何解决构件之间的“语义”互操作;(2) 如何达到应用系统的“可组装性”并与具体运行时系统无关.因此,本文从软件体系结构的角度出发,首先给出一个分布式虚拟现实应用系统的构件模型,指出分布式虚拟现实系统中存在的 3 种基本构件;然后重点讨论了不存在通信协议情况下构件之间的“语义”互操作问题,通过建立构件的语义模型,提出了一种原语-语义构件构造方法;在此基础上,给出了一种支持可组装的与具体分布式运行时无关的分布式虚拟现实应用系统的体系结构 DVR-CA (distributed virtual reality component-based architecture).该结构支持分布式虚拟现实应用系统的“分布”和“进程”两个方向的组装,其中的相关构件利用原语-语义构件的构造方法进行构造.对在这两个方向上的互操作性所引出的相关问题进行了讨论,给出了相关算法,并进一步给出了一个分布式虚拟现实应用系统的构造方法.

本文第 1 节给出一个应用系统构件模型,指出应用系统中 3 种重要构件类型.第 2 节重点讨论不存在通信协议情况下的构件之间的语义互操作问题,通过建立构件语义模型和相关定理,提出一种原语-语义构件的构造方法.第 3 节对 DVR-CA 进行重点讨论,给出其核心机制和算法.第 4 节给出基于 DVR-CA 的应用系统构造方法.第 5 节对通道性能进行分析.第 6 节对比目前几种类似的软件体系结构.最后是结论.

1 系统构件模型

分布式虚拟现实应用系统各个节点上的进程共同“构建”了一个虚拟的世界,随着计算过程的推进,来自外部的输入与虚拟世界不断交互,世界的状态不断发生变化,并通过某种形式“显示”出来.分布式虚拟现实应用系统强调人与虚拟世界进行带有沉浸感的交互.

定义 **1**. 对象:某个对象 $o=\{a_1,a_2,\ldots,a_n\}$是关联的一组属性的集合.对象属性在时刻 t 的值的集合表示了对象

在时刻 t 的状态 $S_o(t)=\{v_{a1}(t),v_{a2}(t),...,v_{an}(t)\}$.

定义 **2**. 世界:一组对象的集合,$W=\{o_1,o_2,...,o_n\}$.世界表示:世界表示集$\Omega=\{a|o\in W'\wedge a\in o\}^l,W'\subseteq W,l=|\Omega|$.每一个世界表示是一些属性的排列,这些属性是世界 W 的某个子集中包含的对象拥有的所有属性.

定义 **3**. 世界的状态:某个时刻 t 世界中各个对象的状态的集合 $S_w(t)=\{S_{o1}(t),S_{o2}(t),...,S_{on}(t)\}$,不同时刻世界状态组成的集合为 Δ.

定义 **4**. 影响函数:$\varphi:I\to\Gamma$,其中 I 为输入,Γ为能对世界状态改变产生作用的因素的集合.影响函数表示了输入对世界状态改变所起的作用.

定义 **5**. 状态转移函数:世界状态上的状态转移函数$\sigma:\Delta\times\Gamma\to\Delta$.状态转移函数反映世界状态根据影响因素改变状态的过程.

定义 **6**. 表示函数:世界状态上的表示函数$\psi:\Delta\to 2^{\Omega}$.表示函数的值是多个世界表示所组成的集合.

从世界的角度来说,存在 3 种输入:用户通过硬件设备对系统的输入,来自节点计算系统内某个程序体的输入以及来自网络(某个分布式运行时)的输入.加在其上的影响函数把这些输入转换成对世界状态的影响,存在两种输出:用户通过硬件设备从各种感觉上体验到的输出以及向网络(某个分布式运行时)的输出.用户与系统的输入/输出构成了人机交互过程,同时,网络的输入/输出表征了分布式进程之间的交互,由此一个分布式虚拟现实系统运转起来.表示函数ψ使不同的世界状态可能对应不同的世界表示集,意味着系统在不同时刻可能有不同的表示组合形式.而组合分别实现世界状态转移函数,影响函数和表示函数功能的构件则能完整地对分布式虚拟现实系统进行实现.本文称它们为世界维护构件、影响构件和表示构件,不同的世界表示又可以作为表示构件的子构件,如三维图形、三维声音构件等.3 种构件之间通过一个软件体系结构以互操作的方式共同完成分布式虚拟现实应用系统的计算任务,如图 1 所示.

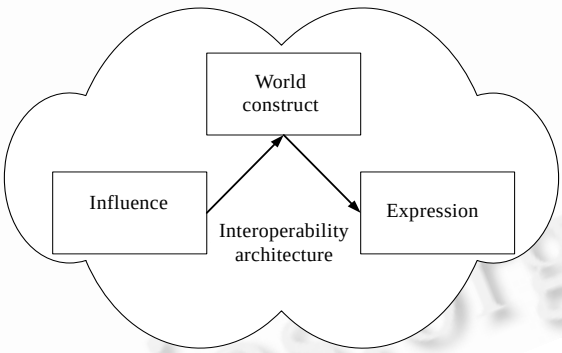

Fig.1 Component model

图 1 构件模型

2 非协议的构件语义互操作性

基于构件的软件开发方法近年来越来越受到人们的重视.文献[10]定义构件为“一组可重用服务构成的可独立交付使用的集合”,同时也存在很多其他定义[11].无论如何,多个构件之间必须能够互相协作,共同完成计算任务.因此,构件和构件之间的“语义”互操作能力就显得非常重要,这个能力要求构件必须能够理解来自其他构件的信息的含义.从抽象的意义上来说,构件需要具备针对语义不同的输入信息而进行不同处理的能力,因此,构件“语义”互操作能力可以被看成是对其内部不同的处理过程的“选择”的能力.这种能力是构件在运行时所表现出来的性质,仅把构件定义为可计算的函数[5]缺乏对构件在运行时所表现出来这种性质进行刻画的能力,因此,本文给出构件的语义模型.

2.1 语义模型

定义 **7**. 计算流程:计算过程中,每一次从程序体入口到出口的处理过程称为计算流程.

定义 **8**. 构件:构件由以下四元组组成:$c=\langle f^{select},G,I,O\rangle$,其中 f^{select} 是“选择函数”,$f^{select}:I\to G$;G 为 c 的计算流程集,对于 $g\in G$,称 g 为 c 的构件流程;I 为构件的输入集;O 为构件的输出集.这里的构件定义适用于无状态的构件,根据第 1 节的讨论,状态由世界维护,因此其他构件都可看成是无状态的,同时,如果把状态的维护看成是世界维护构件的处理逻辑,则世界维护构件也是无状态的,因此,构件针对同一个输入在不同的时刻有相同的处理,即选择函数是存在的.

定义 **9**. 语义关系:构件 $c=\langle f_c^{select},G_c,I_c,O_c\rangle$ 上语义关系 $R_s=\{\langle x,y\rangle|x\in I_c,y\in I_c,f_c^{select}(x)=f_c^{select}(y)\}$.

定理 **1**. c 上语义关系 R_s 是 I_c 上的等价关系.

证明:显然. □

考察构件对语义的处理能力,有如下定义:

定义 **10**. 语义类:构件 c 的语义类 $X\in I_c/R_s$ 或表示为 $[x]_{Rs}$,I_c/R_s 称为构件 c 语义类集.

定义 **11**. 语义完整性:如果对于构件 c 有 $\forall x(x\in I_c\to\exists g(g\in G_c\wedge g=f_c^{select}(x)))$,则称构件 c 是语义完整的.如果对于语义完整的构件 c,还有 $\forall g(g\in G_c\to\exists x(x\in I_c\wedge g=f_c^{select}(x)))$,则称构件 c 是最小语义完整的.

定理 **2**. 如果构件 c 是最小语义完整的,则 $|I_c/R_s|=|G_c|$.

证明:因为 $\forall x(x\in I_c\to\exists g(g\in G_c\wedge g=f_c^{select}(x)))$,所以 $\forall[x]_{Rs}([x]_{Rs}\in I_c/R_s\to\exists g(g\in G_c\wedge g=f_c^{select}(x)))$,设关系 $R=\{\langle[x]_{Rs},g\rangle|[x]_{Rs}\in I_c/R_s,g\in G_c\wedge g=f_c^{select}(x)\}$,对于 $[y]_{Rs}\neq[x]_{Rs}$,设 $[x]_{Rs}Rg$,$[y]_{Rs}Rg'$,则 $g\neq g'$.因为根据关系 R 的定义和语义类的定义,如果 $g=g'$,则 $[y]_{Rs}=[x]_{Rs}$.R 是一个从 I_c/R_s 到 G_c 的一一映射.证毕. □

定理 2 说明,从静态的观点来考察一个构件,如果构件针对并且只针对每一个语义类进行了实现,则该构件是最小语义完整的,并能满足重用性的要求.

2.2 原语-语义构件构造方法

2.2.1 原有方法的缺陷

传统的基于构件的方法利用一个语言无关的接口定义语言(IDL)给出构件接口的语法和语义,如 DCOM,CORBA.如果需要处理的输入语义本身就清晰、明确,则传统的基于构件的方法完全能够胜任.但下列情况下则不适用:构件 A 发布接口为 IA,IA 负责传递一个对象 o,但对象 o 的类名、属性组成和语义描述由外部数据源给出;存在另一个构件 B,需要通过构件 A.IA 获得对象 o,并利用对象 o 中包含的信息进行相关处理,此时问题产生了:构件 B 不能动态地理解外部数据源的语义描述(除非能自然语言理解),那么构件 B 如何来分析 o 所包含的语义信息呢?因为,表示同样的信息,在不同的计算过程中,可能由不同的外部数据描述给出,如某次计算用 Pos 来表示位置,但下次计算用 Position 来表示位置.即使我们可以修改构件 B,增加针对这一描述的处理,但构件 B 不可能通过这种办法去适应所有的可能性,同时这种修改构件的做法直接说明了 B 如果不能重用就不能称其为构件了.以 HLA 为例,同一个对象类描述在不同的 FOM[12]中可能是千差万别的.这其中的重要原因是传统的基于构件的方法只能解决存在通信协议情况下的构件之间语义互操作的问题,而对于不存在通信协议的情况就无能为力了.

针对这个问题,我们借鉴多通道人机交互中的原语系统的思想提出了原语-语义构件构造方法,可以从方法上解决上述的一类构件的构造问题.

2.2.2 本文的方法

定义 **12**. 构件原语:构件 c 的构件原语 $p=\langle action,A\rangle$ 是构件 c 行为或功能的可执行的最小单位,其中 $action$ 为动作符,具有预定义的语义,A 为参数集,$action$ 决定了 $|A|$ 和 A 中参数的语义,原语 $p_{Empty}=\langle Empty,\phi\rangle$ 为空原语,用来表征原语序列的结束.构件 c 的构件原语集表示为 P_c,$P_{Empty}\notin P_c$,$|P_c|=n$,$n\geq1$.任意一个构件流程可以被分解为一个构件原语序列,g 的组成形式为 $g=\langle p_i,p_j,\ldots,p_k\rangle$,有序偶的顺序表示构件原语的叠加顺序,又可写成 $g=p_i,p_j,\ldots,p_k$.

定义 **13**. 原语集能力:给定原语集 P,$p_{Empty}\notin P$,$|P|=n,n\geq1$,则集合 P^+ 表征原语集所有可能的计算能力.

这里我们给出实现给定原语集能力的构件的实现方法:

方法 **1**. 给定原语集 P 具备 P^+ 能力的构件 c 实现方法.

设 $g=P_iP_j\ldots P_kP_{Empty}$ 为输入.

可以分成两部分来实现构件:(1) 原语集 P 中每一个原语的实现.由于原语的语义明确,并且原语集 P 的基数为有穷自然数,因此这部分可以被实现;(2) 实现一个解析输入并调用相应原语实现的过程.这个过程可以表示为:

步骤 1. $p=g.first$; //p 为输入序列中的第 1 个原语

步骤 2. while p 不等于 p_{Empty} do

do p; //执行原语 p

$p=g.next$; //p 为输入序列中的下一个原语

end

定义 **14**. 如果对于构件 c 及其构件原语集 P_c 有 $\forall g(g\in G_c\rightarrow g\in P_c^+)$,则称构件 c 的原语集 P_c 满足构件 c,记为 P_c *set* c.

定理 **3**. 给定构件 $c=\langle f^{select},G,I,O\rangle$ 和 P_c set c,存在构件 $c'=\langle f'^{select},P'_c,I',O\rangle$,其中 $f^{select}:I'\rightarrow P'_c,P'_c\subseteq P^+c$, $I'=P'_c\{P_{Empty}\}$, f'^{select} 为去掉输入序列结尾 p_{Empty} 的函数,且 c 能由 c' 表示.

证明:存在性:利用方法 1 以原语集 P_c 构造构件 c', c' 满足定理的定义.对于构件 c 的任意一个语义类 $[x]_{Rs}\in I/R_s$, $g=f^{select}([x]_{Rs})$,设 $g=p_ip_j\ldots p_k$ (根据定义),以 $p_ip_j\ldots p_kp_{Empty}$ 作为构件 c' 的输入,则根据方法 1,构件 c' 顺序执行了 $p_ip_j\ldots p_k$, c' 模拟了 c 在语义类 $[x]_{Rs}$ 上的动作.又因为 P_c *set* c,所以 $p_m\in P_c,m=i,j,\ldots,k$.证毕. □

定义 **15**. 原语构件:定理 3 中的 $c'=\langle f^{select},P'_c,I',O\rangle$ 称为构件 c 的原语构件,表示为 c_p.容易证明,c_p 是语义完整的.方法 1 也可称为原语构件的构造方法.

定理 3 表明,可以利用满足某个构件的原语集来构造构件.原语的数量是有限的,因此,原语构件易于设计和实现;同时,原语语义固定,且原语构件只和原语相关,使它具有更强的内聚性.但只有原语构件是不够的,还应存在一个过程将输入转化成构件原语,我们定义语义构件来解决这个问题.

定义 **16**. 语义构件:构件 c 的语义类 $[x]_{Rs}$ 的语义构件 $c'=\langle f^{select},G,I,O\rangle$,其中 $|I/Rs|=1$, $|G|=1$, $O\subseteq P_c^+\{p_{Empty}\}$, $|O|=1$, P_c *set* c, $f^{select}([x]_{Rs})=g$, $g\in G$.根据定义,语义构件是最小语义完整的.构件 c 的语义构件集合记为 C_s.

我们给出实现语义构件的方法:

方法 **2**. 构件 c 的语义类 $[x]_{Rs}$ 的语义构件 c_s 的实现方法.

由于对 c_s 来说,$[x]_{Rs}$ 的语义是明确的,因此,c_s 的内部实现是把输入 $[x]_{Rs}$ 转换成构件 c 相应的原语序列并在末尾加上空原语的过程.

图 2 表示了构件 c 的原语构件和语义构件的关系.

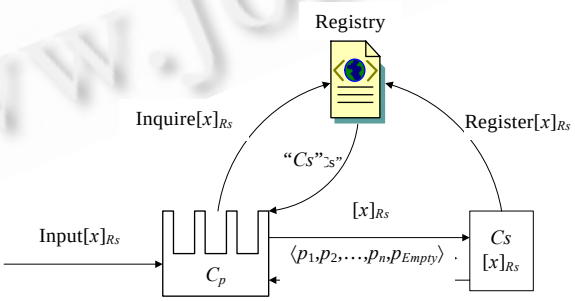

Fig.2 Relationship between primitive component and semantic component

图 2 原语构件和语义构件的关系

构件 c 每个语义构件 $c_s\in C_s$ 都被注册其能理解的输入类,当输入到来,原语构件 c_p 查询哪个语义构件能处理该输入类,然后将输入传递给该语义构件,语义构件处理完输入后将输入转换成 c_p 的原语序列,c_p 执行该原语序列.如果对所有语义类都实现了语义构件,则构件的语义完整性得到保证.原语-语义构件的方法支持嵌套,即某

个原语构件可以是另一原语构件的语义构件.

在实际应用中,构件原语可以凭经验抽取,并可在开发过程中不断演化,最后找到一个满足构件的原语集.例如,对于 3 维图形构件,原语集 $P=\{\langle Draw,\{m,px,py,pz,rx,ry,rz\}\rangle\}$在很大程度上能满足需求,包含一个原语:在位置$((px,py,pz))$以方向$((rx,ry,rz))$绘制$(Draw)$三维模型$(m)$.

3 基于构件的分布式虚拟现实应用系统体系结构

DVR-CA 是一个分布式的软件体系结构.该体系结构允许通过“组装”构件的方法来形成一个分布式虚拟现实应用系统,各构件通过该体系结构进行互操作,协同完成任务.该结构还是可扩展的,允许不断地加入新的构件.这里,我们首先给出该体系结构,然后针对该结构进行讨论.

3.1 元数据(meta-data)与元信息(meta-info)

元数据管理构件负责管理 DVR-CA 中的所有元数据.元数据定义了 DVR-CA 中的基本语义,存在 3 类元数据:控制类、信息类和兴趣类元数据.元数据的实例组成的信息称为元信息,图 3 中黑色实线箭头表示了元信息的流动.传递元信息时需要进行“列集”和“散集”,DVR-CA 定义元信息用 XML 表示,并以字符流的方式传递该 XML 文档.

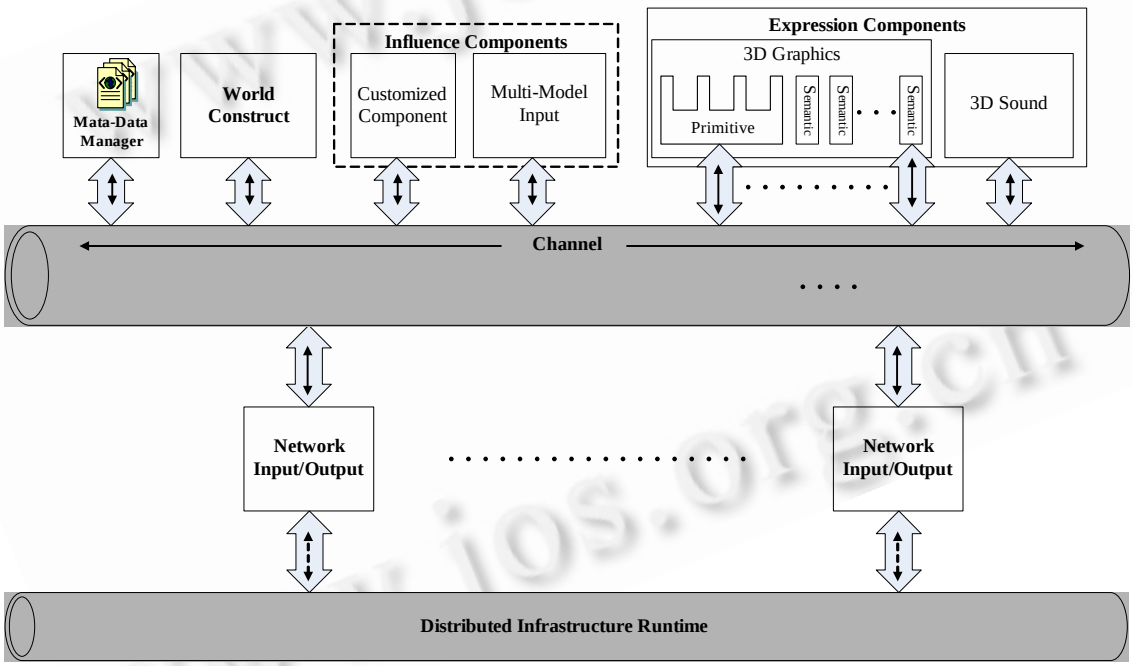

Fig.3 The overview of DVR-CA

图 3 DVR-CA 总体结构图

控制类元数据定义了 DVR-CA 可能发生的事件.有些事件是“异步”事件,有些则是“同步”事件.主要有两类,分别表示控制事件的发生和信息传递事件的发生.

信息类元数据定义如何在 DVR-CA 中表示信息.由于 DVR-CA 的核心是世界状态,因此 DVR-CA 的信息类元数据针对对象进行了元数据定义.还包括一些构件需要的配置元数据.

兴趣类元数据是 DVR-CA 元数据的重要组成部分.存在两种兴趣类元数据:兴趣标识和兴趣表示元数据.兴趣标识元数据定义了如何表示某个元信息的特征,元信息携带该“特征”;兴趣表示元数据定义了如何表示某个构件对具备什么样特征的元信息感兴趣.DVR-CA 规定每个表征“类型”的控制类和信息类元数据都被定义为一个非负整数的 IV(interest value)值,用来表示该类元数据的“特征”.

DVR-CA 定义了通过填表的方法来扩充非兴趣类元数据的机制.分别为“类表”(类名,标识符,IV 值,语义)和“参数表”(类标识符,参数名,标识符,取值类型/范围,语义).前者用来定义控制类和信息类元数据,后者用来定义元数据类的参数.

3.2 通信机制

DVR-CA 的通信机制定义了构件之间是如何进行通信的.由以下规则和算法组成:

(1) 构件从元数据管理构件获得所有“类型”元数据标识符和 IV 值的对应关系表 IVTable.IVTable 上的查询函数 f^{IV}: $Set^{Identifier} \rightarrow Set^{IV}$, $Set^{Identifier}$ 是类标识符集合,Set^{IV} 是 IV 值集合,且 f^{IV} 是一一映射的;

(2) 通道构件负责维护构件兴趣表 $ITable=\{\langle id,I\rangle | id$ 属于 id 集合,I 为兴趣表示$\}$.构件利用元信息 DVR_COM_REGISTER/DVR_COM_REGRES 向通道注册兴趣;利用元信息 DVR_COM_UNREG/DVR_COM_UNREGRES 撤消注册;

(3) 构件发送的元信息由兴趣标识和信息体构成.元信息 $i=\langle I\_descriptor,Info\rangle$,$I\_descriptor=\langle I\_SCOPE, I\_IV\rangle$;

(4) 构件发送元信息时负责为每个待发送的元信息生成兴趣标识.

对于非网络输入构件 c 发送的元信息 i 的兴趣标识 $I\_descriptor$ 生成算法见算法 1.

算法 **1**.

对于非网络输入构件 c

步骤 1. $i.I\_descriptor.I\_SCOPE=c.I\_SCOPE$;

步骤 2. $i.I\_descriptor.I\_IV=f^{IV}(i.Info.Identifier)$;

对于网络输入构件 c,设从网络接收到信息的兴趣标识为 $Input.I\_descriptor$

步骤 1. if $Input.I\_descriptor.I\_SCOPE=c.I\_SCOPE$

then $i.I\_descriptor=Input.I\_descriptor$;

else

$i.I\_descriptor.I\_SCOPE=c.I\_SCOPE$;

$i.I\_descriptor.I\_IV=f^{IV}$(网络输入元数据类型);

end.

(5) 兴趣匹配算法见算法 2.

算法 **2**.

$i.I\_SCOPE = I.I\_SCOPE \wedge i.I\_IV \in I.I\_IV\_SET \Rightarrow i\ M\ I$,$i$ 为兴趣标识,I 为兴趣表示,$i\ M\ I$ 表示 i 与 I 匹配.

(6) 通道负责转发从某个构件发送过来的元信息,我们定义该过程为通道路由过程.对于元信息 i,通道路由算法见算法 3.

算法 **3**.

步骤 1. for t 为所有的 $ITable$ 表项 do

if $i.I\_descriptor\ M\ t.I$ then add $t.id$ into $DesSet$;

end

步骤 2. if $DesSet$ is empty,把 i 发送给网络输出构件,返回;

步骤 3. for d 为所有的 $DesSet$ 中的 id do

将 i 发送给 d;

end.

3.3 主要组成构件及其构造

3.3.1 通道构件

通道是 DVR-CA 的重要组成部分,可以把它看成是一个多播的拥有“足够多”的全双工通路的信息传递系统,它利用第 3.2 节中的通信机制决定信息的流向.在分布式计算过程中,每个节点都运行一个通道构件实例,分布在不同结点上的通道构件构成了逻辑上的“通道”,其内部流动的是元信息.如果在相同的节点,元信息的流动是在通道构件内部完成的,对于不同节点之间,元信息的流动通过挂接在通道构件上的分布式运行时支持构件(网络输入/输出构件)来实现.由于只需传递由元信息组成的信息流,通道构件需要且只需要实现一个语义为(Set,Get)的接口对,其中 Set 接口用来实现通道从外界接收信息的过程,而 Get 则用来实现向外界发送信息的过程,为了避免其他构件向通道进行轮询,Get 采用“回调”机制实现.由于通道构件需处理输入的语义明确,内部逻辑单一,因此,通道构件可采用传统方法来实现.

3.3.2 世界维护构件

类似数据库的管理,世界维护构件是由“增,删,改,查”4 个原语构成的原语构件,其语义构件为用户构件.

3.3.3 用户构件

用户特定领域的计算代码在用户构件中实现,如仿真某辆坦克的行为.用户构件是影响构件的原语构件,接收来自影响构件的输出原语序列,并输出世界维护构件的原语序列.因此,用户构件同时是世界维护构件的语义构件.

3.3.4 多通道人机交互输入构件

该构件是影响构件的语义构件,其输出是用户构件的原语序列.本文的原语-语义构件方法在此和多通道人机交互原语系统统一起来.

3.3.5 表示构件

表示构件包括三维图形构件、三维声音构件和二维图形构件等.表示构件是处理向用户输出的构件,其输入是世界的状态.每个构件都采用原语-语义构件的方法构造.

3.3.6 网络输入输出构件

网络输入输出构件一起将运行在不同节点上的通道实例连接成一个逻辑“通道”,构件通过该通道进行通信.网络输入输出构件负责与通道构件和具体的分布式运行时进行通信.更换不同网络输入输出构件,可使得 DVR-CA 平滑地适应不同的分布式运行时;同时,如果输入和输出分别使用不同的运行时,且用户构件实现的是收集/转换/播放(Record/Transform/Replay)的功能,则 DVR-CA 可以用来构造不同分布式运行时之间的“网关”.而已完成的该用户构件又可加入到原有的构件库中来,如图 4 所示.

网络输入/输出构件执行逻辑和具体的运行时相关,可以采用传统方法实现,也可为网络输入/输出构件开发针对某个具体运行时的应用程序“框架”[13],从而加速网络输入输出构件的开发过程.同时,网络输入构件的输出有可能是用户构件的输入,此时,网络输入构件又可看成是用户构件的另一个语义构件.同时,网络输出构件有可能是对世界维护构件的输出,此时,网络输出构件又可看成是表示构件.网络输入/输出构件逻辑上是分开的,实现中可能由于具体限制的原因而需要合二为一.

3.4 结构特性讨论

3.4.1 可扩展性

DVR-CA 的可扩展性主要体现在以下 4 个方面:

(1) 任何非通道构件只与通道构件相关,与任何其他构件都不存在依赖关系;它们和通道构件通信的方式是统一和唯一的,即通过通道提供的(Set,Get)向通道发送或从通道接收元信息流.从这个角度来说,处在图 3 底层的网络输入/输出构件和处在上层的所有构件的地位是平等的.这种“平等”的地位使得在 DVR-CA 上可以增加构件而对原有构件不产生影响;

(2) 原语构件与语义构件的构造方法使得可以通过增加语义构件的方法来增强原语构件的表示能力;

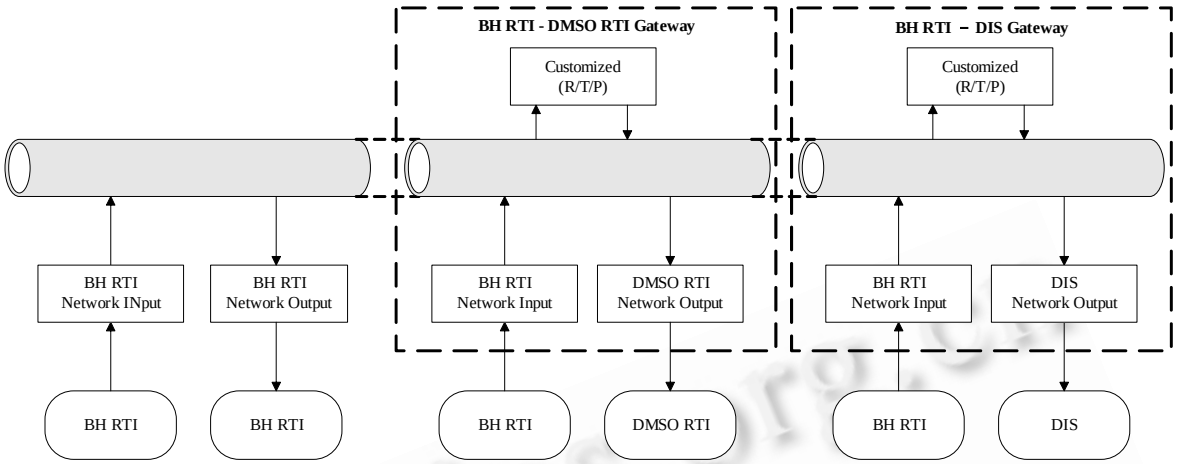

Fig.4 Channel and gateway

图 4 通道的形成及网关

(3) 用户构件允许用户开发自己领域内的应用,将用户定制的功能加入到 DVR-CA 的运行环境中;

(4) 元数据的可扩充机制与 DVR-CA 的通信机制,保证了 DVR-CA 能胜任用户定制的计算任务.

3.4.2 互操作性与可组装性

DVR-CA 结构下,构件之间的“语法”互操作性由构件接口定义及元信息的 XML 表示保证.同时,原语-语义构件的构造方法和传统的基于构件的方法一起实现了构件之间的“语义”互操作性.不同构件之间信息的交换通过通道来实现,通过挂接不同的分布式运行时,来实现分布式节点之间的信息交换.

在 DVR-CA 结构下,可通过对初始元数据 DVR_SYS_COMPOSE 来配置系统的组成.而在运行过程中,又可根据 DVR_SYS_COMPOSE 的改变来动态改变系统的组成,同时,DVR-CA 的元数据,通道构件的存在和兴趣机制等一起促成了“进程”和“分布”两个方向上可组装性的实现.

4 应用系统构造方法

目前我们已经实现了运行管理构件,元数据管理构件,世界维护构件,多通道人机交互输入构件,三维图形原语构件,三维声音原语构件,基于 BH RTI 2.0 和 DMSO RTI 1.3v6 的网络输入/输出构件.其中,世界维护构件的原语有 4 个,语义分别是对世界状态的增、删、改和查并附带相关参数;三维图形原语构件的原语为 1 个,语义为绘制某个三维几何模型,相关参数为绘制位置和方向;三维声音原语构件的原语为 1 个,语义为播放某个声音文件,相关参数为发声位置.

在 DVR-CA 结构下,存在 3 种范围:全局范围,即整个分布式计算范围;系统范围,即同一个应用系统范围,I_SCOPE 值相同;进程范围,即构件运行在同一进程内范围.基于 DVR-CA 的系统构造流程如下:

步骤 1. 在全局范围内填写自定义的元数据,提交给元数据管理构件;

步骤 2. 在全局范围内编写不存在的构件,如用户构件,表示原语和语义构件,并加入到构件库;

步骤 3. 在系统范围内配置系统的组成,提交给元数据管理构件;

步骤 4. 在系统范围内配置构件的 I_SCOPE 值,逻辑上在同一系统的构件 I_SCOPE 值相同;

步骤 5. 在系统范围内配置驱动方式,提交给元数据管理构件,该驱动方式为网络输入/输出构件使用;

步骤 6. 在全局范围内配置具体运行时需要的元数据,其由网络输入/输出构件通过具体运行时接口规范加以使用,如 HLA FOM;

步骤 7. 在系统范围内分析系统组成,从构件库得到构件;

步骤 8. 在进程范围内启动各节点的运行管理.

图 5 给出了 DVR-CA 的典型信息流,以基于 HLA/RTI 的例子加以说明.

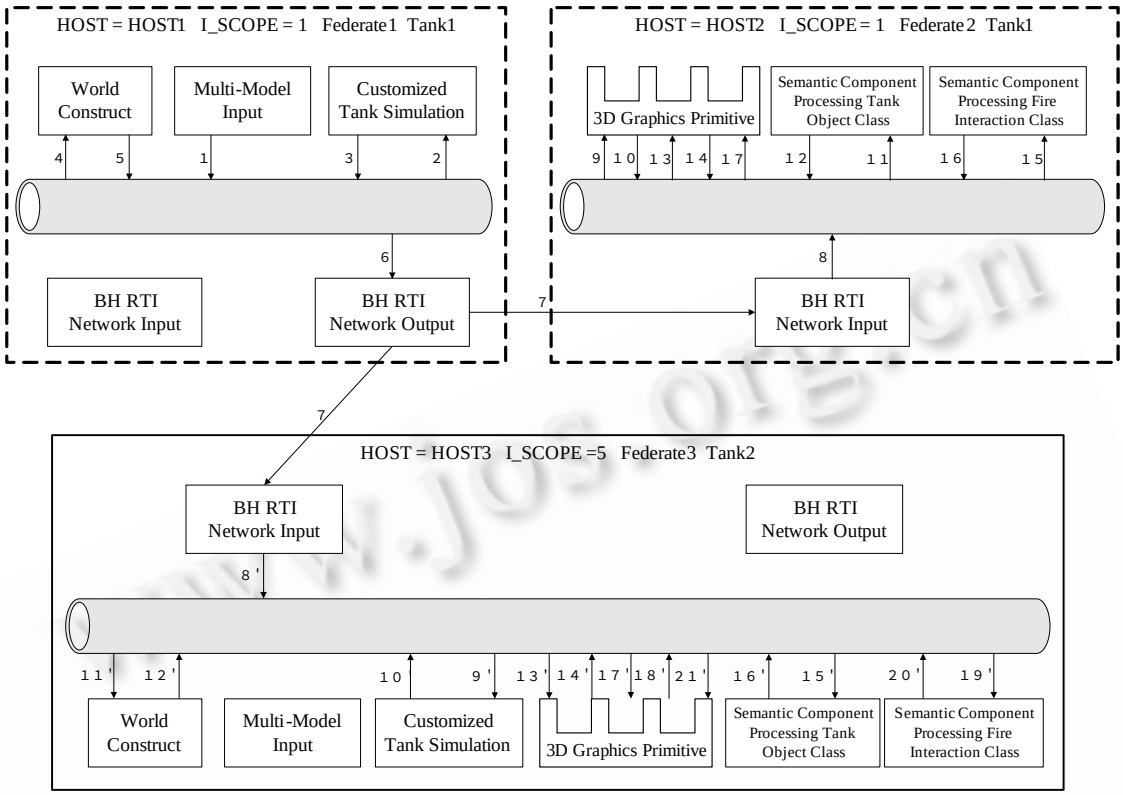

Fig.5 Typical information stream in DVR-CA

图 5 DVR-CA 典型信息流

该例的想定是:在某区域进行两辆坦克的攻防演练.每辆坦克仿真系统人机界面都包括键盘、三维场景输出.其中一辆坦克系统的三维图形原语构件及其相应的语义构件与该系统的其他构件分别配置在两个节点上.该联盟演练 FOM 中定义了坦克对象类 TANK 和开火交互类 FIRE,另外还定义了信息传递交互类 DVR_INFO,其参数 SCOPE 表示 I_SCOPE 值,参数 MESSAGE 为一个字符串,用来传递通过 XML 序列化后的元信息.该图表示的坦克 1 向坦克 2 开火的信息流过程:

(1) 人机交互输入构件接收到用户按下“F”键,将该动作转化成输入原语“Fire”,以元信息方式发送给通道;

(2~3) 用户构件接收到该输入,形成一个开火交互“fire”,最终形成一个世界维护构件的原语“ADD fire”,以元信息方式发送给通道;

(4~5) 世界维护构件接收到该原语,执行相应动作;当被允许向外界输出状态时,世界维护构件向通道输出世界状态;

(6) 分两个子过程:一个是向表示构件输出世界状态,由于在 Federate1,没有注册对该元信息感兴趣的构件,通道将世界状态信息发送给网络输出构件,它把该元信息通过 DVR_INFO 交互发送给 BH RTI,利用分布式运行时将该信息传递到其他对该元信息感兴趣的进程;另一个是向网络输出构件输出具有本地所有权的部分世界状态,通道将该元信息通过 TANK 对象更新和 FIRE 交互发送给 BH RTI;

(7) BH RTI 发送 DVR_INFO 交互到 Federate2,因为 Federate2 声明订购 DVR_INFO,并且 DDM 声明要求其中的 SCOPE=1;BH RTI 发送 TANK 对象更新和 FIRE 交互到 Federate3;

(8) Federate2 的网络输入构件将 DVR_INFO 交互中的元信息(世界状态)放到通道上;

(9) 通道将世界状态发送给三维图形原语构件;

(10~17) 三维原语构件和它的语义构件交互的过程,最终形成两个三维图形原语:在(x,y,z)处绘制坦克和在(x',y',z')处绘制开炮烟尘;

(8′~21′) 过程与(10~17)类似,发生在 Federate3.

5 性能分析

我们通过实验对通道传输元信息的性能进行了分析:该实验随机选择构件发送 128~4 096 字节数的元信息,在每个构件数情况下做 100 次.每个加入构件通过第 3.2 节的注册机制向通道构件注册兴趣.实验数据在一个节点上获得以屏蔽某个具体的分布式运行时对通信的影响,利用 QueryPerformanceCounter()实现微秒级时间的测量.

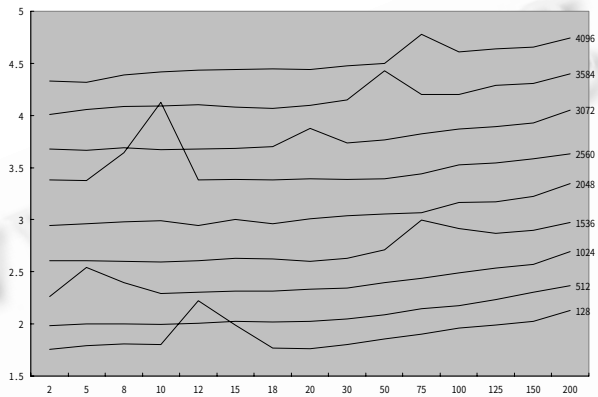

Fig. 6 Communication experimental data in channel

图 6 通道通信实验数据

图 6 中横轴表示通道上的连接的构件数,纵轴表示通道从得到元信息到完成传递给另一构件的平均时间,单位为μs,每条曲线表示不同的传输字节数,数据在 Windows XP,Pentium IV 1.7GHz,512M 的机器上获得.数据显示,传输字节数越大,总体耗费时间越长,同时,随着构件数目的增多,通信耗时有增加的趋势.

另外,我们还测得了在 DVR-CA 结构下,加载/卸载一个非通道构件的平均时间,分别为 0.241ms 和 0.247ms.这样,对于在通道构件挂接了 10 个左右构件的节点来说,平均组装花费为 2.4ms.

6 相关工作比较

我们从分布式系统构造和主流的分布式计算体系等角度对 DVR-CA 和类似的基于构件的体系结构进行了比较,见表 1.

Table 1 Comparison among architectures

表 **1** 结构比较表

	DVR-CA	ModISE	OneSAF	SCM
Composability in “process”	YES	YES(IE)	YES	NO
Composability in “distribution”	YES	YES (IE)	YES	NO
Extensibility	YES	YES	YES	YES
HLA/RTI	YES	YES	YES	YES
Other runtime	YES	YES	NO	YES
Mixed runtime	YES	YES	NO	YES

从目前研究热点之一“可组装性”的比较来看,DVR-CA 支持在“进程”和“分布”两个方向(层次)上的可组装性,增强了分布虚拟现实应用系统的重构能力,ModISE 的可组装性支持只存在与其互操作引擎(IE)内;从支持的主流的分布式计算体系方面来看,DVR-CA 能够灵活地适应多种运行时支持环境并可混合使用,从而支持了运行时之间的网关的构建,OneSAF 的体系结构则仅仅适用于 HLA/RTI;从可扩展性的比较上看,OneSAF 的体系结构仅考虑了自身的应用;SCM 作为一种构件模型,未对可组装性的支持做深入的探讨.

7 结束语

本文给出了一种分布式虚拟现实应用系统的构造方法.通过建立构件的语义模型,提出了一种原语-语义构件构造方法,该方法可以解决不存在通信协议情况下的构件之间“语义”互操作问题;给出了一种支持可组装的并与具体分布式运行时无关的分布式虚拟现实应用系统基于构件的体系结构,并利用原语-语义构件构造方法对相关构件进行了实现.通过实验分析,DVR-CA 通道传输在可预期的构件数和传输元信息量范围内的传输时间只是 10^0 微秒级,同时节点的平均组装时间为 10^0 毫秒级,可以满足系统应用的需要.

目前我们正在研究如何把 DVR-CA 和 Grid 体系结构相融合,以构建一个支持远程沉浸的分布式虚拟现实网格服务系统.我们的想法是在 DVR-CA 的基础上,利用网格的资源/注册机制来对构件进行管理,利用它的 FTP 服务进行构件的动态部署,利用虚组织技术对系统的动态负载平衡进行调节并进行容错处理,对于远程三维图形的输出采用 VNC[14]协议以支持一个非常“瘦”的客户端或利用动态部署把相应的输出构件部署到客户端.

致谢 在此,我们向对本文的工作给予支持和建议的老师表示感谢.

References:

[1] Zhao QP, Shen XK, Xia CH, Wang ZQ. Dvenet: A distributed virtual environment. Journal of Computer Research & Development, 1998,35(12):1064−1068 (in Chinese with English abstract).

[2] Petty MD, Weisel EW. A composability lexicon. In: Proc. of the Spring 2003 Simulation Interoperability Workshop. 2003. 03S-SIW-023.

[3] Pollack RH, Baldwin R. Requirements for composing simulations: A use-case approach. In: Proc. of the Spring 2003 Simulation Interoperability Workshop. 2003. 03S-SIW-013.

[4] Biddle M, Perry C. An architecture for composable interoperability. In: Proc. of the Fall 2000 Simulation Interoperability Workshop. 2000. 00F-SIW-073.

[5] Petty MD, Weisel EW. A formal basis for a theory of semantic composability. In: Proc. of the Spring 2003 Simulation Interoperability Workshop. 2003. 03S-SIW-054.

[6] Weisel EW, Petty MD, Mielke RR. Validity of models and classes of models in semantic composability. In: Proc. of the Fall 2003 Simulation Interoperability Workshop. 2003. 03F-SIW-073.

[7] Huang JY. Design of plug and play simulator over distributed environment. In: Proc. of the Fall 1997 Simulation Interoperability Workshop. 1997. 97F-SIW-014.

[8] Franceschini DJ, Hawkes KR, Graffuis S. System composition in OneSAF. In: Proc. of the Spring 2003 Simulation Interoperability Workshop. 2003. 03S-SIW-052.

[9] Parr S, Keith-Magee R. The next step—Applying the model driven architecture to HLA. In: Proc. of the Spring 2003 Simulation Interoperability Workshop. 2003. 03S-SIW-123.

[10] Brown AW, Short K. On components and objects: The foundations of component based development. In: Proc. of the 5th Int’l Symp. Assessment of Software Tools and Technologies. Los Alamitos: IEEE CS Press, 1997. 112−121.

[11] Friedrich LF, Stankovic J, Humphrey M, Marley M, Jr Haskins J. A survey of configurable, component-based operating systems for embedded applications. Micro: IEEE, 2001,21(3):54−68.

[12] IEEE std 1516.2-2000, IEEE standard for modeling and simulation (M&S) high level architecture (HLA). Object Model Template (OMT) Specification. 2000.

[13] Duan ZY, Wu W. Research on distributed-interactive-simulation-oriented application framework. Journal of System Simulation, 2003,15(Supp.):234−237 (in Chinese with English abstract).

[14] Richardson T, Stafford-Fraser Q, Wood KR, Hopper A. Virtual network computing. IEEE Internet Computing, 1998,2(1):33−38.

附中文参考文献:

[1] 赵沁平,沈旭昆,夏春和,王兆其.DVENET:一个分布式虚拟环境.计算机研究与发展,1998,35(12):1064−1068.

[13] 段作义,吴威.面向分布式交互仿真领域的应用程序框架的研究.系统仿真学报,2003,15(增刊):234−237.

段作义(1974－),男,天津人,博士生,主要研究领域为分布式虚拟现实.

吴威(1961－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为虚拟现实,分布式系统.

赵沁平(1948－),博士,教授,博士生导师,CCF 高级会员,主要研究领域为虚拟现实,可视化,人工智能.

第 10 届中国机器学习会议

征 文 通 知

2006 年 10 月 13-15 日, 海口

The 10th China Conference on Machine Learning

October 13-15, 2006, Haikou

第 10 届中国机器学习会议（CCML2006）由中国人工智能学会机器学习专业委员会和中国计算机学会模式识别与人工智能专业委员会联合主办，海南大学承办，海南软件学院协办。该系列会议每两年举行一次，现已成为国内机器学习界最主要的学术活动。此次会议将为机器学习及相关研究领域的学者交流最新研究成果、进行广泛的学术讨论提供便利，并且将邀请国内机器学习领域的著名学者做精彩报告。

征稿范围（征求但不限于如下主题）

机器学习的新理论、新技术与新应用
人类学习的计算模型
计算学习理论
监督学习
非监督学习
强化学习
多示例学习
半监督学习
集成学习
多策略学习
基于案例的推理
增量学习与在线学习
对复杂结构数据的学习
增强学习系统可理解性
数据挖掘与知识发现
神经网络
神经网络集成
进化计算
人工生命
模糊集与粗糙集
多 Agent 系统中的学习
模式识别
信息检索
生物信息学
语音、图像处理与理解
自然语言理解

重要日期

全文投稿：2006 年 3 月 15 日
录用通知：2006 年 5 月 15 日
修改定稿：2006 年 6 月 15 日

会议咨询

电话: 0898-66272862（曾水香），66288382（雷景生）