第33卷 第19期 计 算 机 工 程 2007年10月
Vol.33 No.19 Computer Engineering October 2007
·软件技术与数据库· 文章编号：1000—3428(2007)19—0060—03 文献标识码：A 中图分类号：TP391

开放分布式计算时限模型的设计与实现

申利民，李 峰
（燕山大学信息科学与工程学院，秦皇岛 066004）

摘 要：将开放实时分布式系统的需求分离为功能需求、监视需求和操控需求，并将其映射为具有协同层和功能层的软件模型，利用Java消息服务和EJB技术，提出了一个实现构架，以车辆导航系统为例说明模型和构架的交互性和实时性。
关键词：开放分布式计算；时限模型；关注点分离；时控协同器；交互协同器

Design and Implementation of Time-constrained Model for Open Distributed Computation

SHEN Li-min, LI Feng
(College of Information Science and Engineering, Yanshan University, Qinhuangdao 066004)

【Abstract】 This paper separates requirements of open distributed system into functional requirement, monitoring requirement and manipulating requirement, mapping the system as a coordination layer and a function layer. Based on Java information service and EJB technology, it proposes an infrastructure implementation. Experimental results of vehicle navigation system show the module's interactivity and real-time interaction.
【Key words】 open distributed computation; time-constrained model; separation of concerns; timing coordinator; interaction coordinator

在开放分布式环境下，实时信息系统要面对静态和动态环境的变化以及用户需求的变化[1]，需要解决的问题如下：(1)应用系统可能要求添加或减少实体，实体也可能离开或进入系统，新功能的实体也可能出现。(2)在传统的实时系统中，连接交互逻辑、实时约束逻辑和功能逻辑相互交织在一起，并以程序代码的形式硬化在构件之中，修改和扩充系统困难，缺乏适应性和柔性[2]。(3)通过已存在的标准商品化构件或Web服务，可以构建系统[3]，这些构件是些“黑盒子”，既不允许代码检查也不允许代码变化，往往服务提供者的QoS是不可控的。(4)用户的功能要求和实时策略也会发生变化。

以上这些因素不仅使开放分布式环境下的实时性难以得到保证，而且使得用户的功能需求和实时策略变得更加复杂。为此，利用关注点分离的原理[4]，构建一个双层软件模型，将系统分为计算实体、时控协同器和交互协同器，采用不侵入代码的方式将独立的时控协同器、交互协同器和计算实体动态地松散耦合起来，进而降低了开发的复杂性，适应环境和用户需求的变化。

1 模型的设计

1.1 分离关注点

软件的需求可划分为功能需求和非功能需求。功能需求是系统要完成的功能和任务，描述系统要展示的行为，即软件在某种输入条件下要给出确定的输出必须做出处理或转换。非功能需求不描述软件做什么，而是描述软件如何做，做得怎么样，定义功能需求满足的约束、条件或指标。

设想一个具有GPS和交通控制中心辅助的汽车导航系统。对一个指定的目的地，GPS系统能够帮助汽车到达目的地，但是GPS系统选择的路径不一定是最优路径，需要交通控制中心提供当前的路况和交通信息。不同的交通控制中心控制不同的区域，导航系统为了及时、准确地获取更新的信息，需要随着位置的变化与不同的控制中心建立通信连接。如果请求的信息在规定的时间内不能从交通控制中心获得，导航系统就必须根据GPS选择路径。

从这个实例可以看出，开放实时分布式系统，有3种主要的要求[5]：(1)交互动态性，系统中自治实体的交互布局是频繁改变的；(2)实时性，需要对系统的行为和自治实体，施加实时的约束；(3)功能性，系统需要完成的功能任务。

交互动态性和实时性是非功能需求，如果这些需求相互交织和散落，非功能需求和功能需求都很难变化，对非功能策略的变化就会引起对功能代码的修改，反之亦然。为了实现满足多方面要求的系统，分别进行功能性需求分析和非功能性需求分析，把隐含在应用需求中的关注点分离出来，独立地对它们进行分析和建模。

非功能的实时需求分离为2个方面：

(1)监视需求。系统需要监视功能行为，验证它们是否满足所希望的质量要求，定义要监视和验证时限属性。如性能、响应时间、网速等检测。

(2)操控需求。通过施加在功能部件上具体约束和操作行为来满足非功能性的要求。如口令验证、改变处理方法、交互拓扑变动等。

1.2 双层模型

为了能够在软件结构上找到对关注点的支持，并将它们映射到软件结构的部件上，在模型中设计了3类结构部件：

(1)功能部件。实现系统的功能要求；

(2)操控部件。完成非功能的适应操作；

(3)监视部件。监视功能部件和操控部件的行为是否满足系统的非功能性要求，一个监视部件可能监视多个功能部件和操控部件。

根据以上分析，把隐含在对象内部的交互行为和实时约

基金项目：国家留学基金资助项目(2003813003)
作者简介：申利民(1962－)，男，教授、博士生导师，主研方向：柔性软件工程，软件协同技术；李 峰，硕士
收稿日期：2007-03-19 **E-mail：**shenllmm@sina.com

束行为从传统自治实体中分离出来，把系统分为 2 层：(1)功能层完成纯的功能任务，由计算实体构成；(2)协同层协调功能层的计算实体，满足系统非功能要求，定义了对功能层的约束条件，由时控协同器和交互协同器构成。

计算实体只完成纯功能的计算任务，不再关心与之通信的其他计算实体以及需施加在它上面的实时约束，交互协同器只专注调整计算实体的交互关系，时控协同器只负责监视时限行为。运行时，协同器把这些分离的、独立的和直交实体组合在一起。各部分可以独立地进行设计和建模，提高了各类实体的模块性和复用性，而且满足了动态适应性的要求。模型的基本框架如图 1 所示。

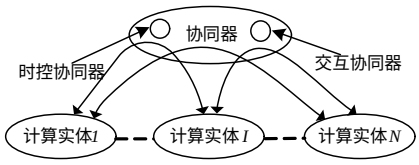

图 1　模型基本框架

1.3　部件间的通信

在该模型中，确立 2 种类型的交互和通信媒体：消息和事件。消息只由计算实体产生，传送计算信息，通过计算实体间的连接通道进行点到点的异步传输，消息连接通道由交互协同器建立；事件可由协同器或计算实体产生，传送的是控制信息，事件告诉了事件源的特定行为和状态。

大多数实时约束，如最终时限、最早时限、频率等都可以映射到时间轴上的事件或消息。而可靠性、安全性、容错性可以映射为空间轴上的消息和事件的操控。事件和消息既是各实体的媒介，又是非功能要求实现的媒介，同时适合于分布式系统。

该模型利用事件机制控制信息的交换。当实体在当前状态观察到特定事件发生时，根据事件类型改变当前的状态，执行相应的动作。事件源触发一个事件后，仍然能够继续执行，无需等待事件的回应，同时在环境中传播的事件能够被对该事件感兴趣的实体观察到。一般而言，实时约束可以分为最终时限约束和延时约束，可以用 2 个事件的时间戳t_1,t_2和时间段d来表达。时限约束表示为 $t_1-t_2\le d$，延时约束表示为 $t_1-t_2\ge d,d\ge 0$。

实时行为在限定的时间内或在延迟的时间外完成对某事件响应和处理活动。在模型中，时控协同器通过产生计时事件，监视实体行为间的时限关系，同时交互协同器通过动态地改变计算实体间的消息连接回应时控协同器的时控事件，满足时限要求。

1.4　模型的构成

1.4.1　计算实体

计算实体是功能构件，是一个自治的和活动的独立对象，是只具有 send 和 receive 接口的“ 黑箱 ”，通过 send 和 receive 原语发送或接收消息，实现与外界的通信。它只把消息发送到自己的发送端口，而不需要知道消息的接收者和协同者；只根据收到的消息内容改变自身的状态和行为。只要能够得到合适的消息，计算实体就能输出所要求的消息。因此，模型实现了匿名通信，通信协同器对计算实体是透明的，计算实体很容易移动到其他环境中。

计算实体封装一个执行线程，具有状态和行为，有 2 个概念位置：消息接收队列和消息发送队列。状态规定了在特定条件下要完成的操作，行为是一系列针对操作消息的处理动作，实体之间通过消息进行异步通信，未处理的消息存储在队列中。

常用的操作原语有：

(1)send(MessageType, message)：发送一个消息；

(2)receive(MessageType, message)：接收一个消息；

(3)raise(event-name,V_1,⋯,V_k)：产生一个带有参数V_1,⋯,V_k的特定事件，表明计算实体的状态和发生的动作。

1.4.2　交互协同器

交互协同器有自身状态和基本操作，存储已存在的计算实体的标识名和接口信息，其主要功能是依据事件和协同规则建立计算实体之间的连接，根据建立良好的连接关系，并转发消息，动态管理和重新配置系统的交互布局。交互协同器在当前状态等待一些特定事件的发生，当观察到某一特定事件后，根据事件类型改变其状态，然后执行相应的行为。在交互协同器中最基本的行为是建立和断开计算实体间的消息连接。通常，当前状态的所有动作执行完毕后，交互协同器仍然处在当前的状态下，直到观察到有其他事件发生，再改变当前状态，执行另一状态上的操作。

交互协同器使用以下基本操作原语：

(1)event(event-name)：观察事件，触发相应操作；

(2)connect(ce_1, ce_2)：在计算实体ce_1和ce_2之间建立消息连接；

(3)disconnect(ce_1, ce_2)：断开计算实体之间的连接；

(4)raise(event-name, V_1, ⋯ ,V_k)：产生一个带有参数V_1,⋯,V_k特定事件，表明交互协同器的状态和发生的动作；

(5)wait-event()：等待事件的发生。

1.4.3　时控协同器

时控协同器有自身状态和基本操作，检查实时约束行为。主要功能是根据自身的状态和时限约束策略对特定的事件模式进行监视，根据监视结果发布相应事件，使用以下基本操作原语：event,raise,impose,wait-event,dismiss，其中，event, raise, wait-even 操作与交互协同器相同。

(1)impose(E_1,E_2,bounded-time,timeout-event)：对事件模式(E_1,E_2)施加时限监查。在E_1事件发生后的bounded-time时间间隔内，如果E_2没有发生，将产生一个timeout-event事件，在timeout-event事件产生后，解除已施加的时限监查；如果E_2在bounded-time时间间隔内发生，表明系统满足该模式的时间约束。

(2)dismiss (E_1,E_2,Bounded-time)：解除已施加在事件模式(E_1,E_2)上的时限监查。

(3)impose-delay(E_1,E_2,bounded-time,delaybreak-event)：对事件模式(E_1,E_2)施加延时监查，即在E_1事件发生后的bounded-time时间间隔内，如果E_2发生，将产生一个delaybreak-event事件，如果E_2在bounded-time时间间隔外发生，表明系统满足该模式的时限约束。

(4)dismiss-delay(E_1,E_2,bounded-time)：解除已施加在事件模式(E_1,E_2)上的延时监查。

2　模型的实现构架

2.1　实现构架

模型采用 JMS(Java message service) 技术和 EJB (enterprise Java bean)技术[5]来实现。实现构架由 3 个部分组成，如图 2 所示。

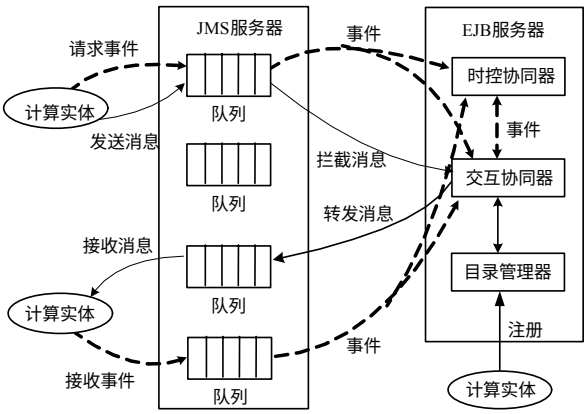

图 2　模型的实现架构

(1)JMS 客户端。运行计算实体，消息的发送者和接收者，创建与 JMS 服务器的连接和会话，创建消息发送者和接收者对象。

(2)JMS 服务器。系统中各节点通信的平台，提供对计算实体消息队列的管理，能够动态创建和撤消一个消息队列，不同应用之间可以依靠 JMS 的点对点(PTP)和发布/订阅(Pub/Sub)方式，实现同步或异步的消息通信和事件触发。

(3)EJB 服务器。EJB 组件赖以生存的运行和管理环境，部署基于消息驱动 Bean 开发的时控协同器和交互协同器。

2.2　实现技术

2.2.1　计算实体

计算实体接收和发送消息的操作是由计算实体中的消息发送对象 sender(sendQueue)和消息接收对象 receiver(receiveQueue)实现，这 2 个对象是计算实体初始化时，利用 JMS 提供的 API 服务创建的。在计算实体内部只需利用 sender 对象中的 send(message)方法就可以直接向消息发送接口 sendQueue 发送消息，利用 receiver 对象的 receive(message)方法就可以从消息接收接口 receiveQueue 接收消息。

2.2.2　时控协同器

通过 EJB 的 MessageDrivenBean, EJBTimer 服务的 TimerService,TimedObject 接口以及 JMS 的 MessageListener 接口，为时控协同器提供计时操作和事件处理。TimerService 接口使时控协同器能够访问到 EJB 容器提供的 Timer 服务，利用该接口提供的 createTimer()方法可以创建一个定时器。MessageListener 接口主要提供了一个监听事件的 onMessage 方法，当一个事件到达时，EJB 服务器自动调用该方法处理事件。

在时控协同器中，对事件模式(E_1,E_2)施加实时监视约束的impose和解除实时约束的dismiss定义在onMessage方法中。在impose的具体操作如下：

(1)根据观察到E_1事件时，利用createTimer (bounded-time, null)方法为该事件创建一个定时器timer并施加了一个时间约束bounded-time。

(2)利用ejbTimeout(timer)方法中实现时间超时发生的动作，主要是发布一个timeout-event事件，如果E_2事件在超时之前发生，则撤消这个定时器。当时控协同器不处理事件或消息时，被放回到EJB容器的缓冲池中，等待下一个事件的发生。

2.2.3　交互协同器

实现 EJB 规范中的 MessageDrivenBean 接口和 JMS 的 MessageListener 接口，这 2 个接口的作用与时控协同器的接口作用相似。在 onMessage()方法中定义了 3 种处理事件和消息的操作：connect,disconnect,route。当一个消息或事件到达与它建立联系的队列时，EJB 服务器把该事件或消息传递给 onMessage 方法，然后根据该事件的类型和交互协同器的状态执行 connect,disconnect,route。

connect 方法是建立连接的基本原语，它的 2 个参数分别为被连接的 2 个计算实体标识名。当执行 connect 时，用实体标识名到目录管理器中实体的相关信息，将 2 个计算实体的标识名、发送接口和接收接口信息存储到一个计算实体连接关系表中，该表的一条记录用来表示实体之间的连接关系。route 方法主要根据建立的连接关系转发消息。当交互协同器监听到计算实体发送一个消息后，根据消息发送者标识名在连接表中查找对应消息接收者的接收接口 queueName，然后由自定义的消息发送方法 messageSender(Messagetype, message, type, queNam)把消息发送到该接口，实现消息的转发功能。方法 messageSender 的主要功能是发送消息到相应队列，在这个方法中首先根据参数 QueNam 的值，建立与该队列的连接，然后把消息发送到该队列中。当交互协同器不处理事件或消息时，被放回到 EJB 容器的缓冲池中，等待下一个事件的发生。

3　实例研究

在 1.1 节描述的实例中，假设汽车在旅行中经过几个区域，在每个决策地点需要从交通控制中心接收路况信息，如果 20s 之后没有收到路况信息，就采用 GPS 系统选择路径。此外，交通控制中心需要每 5min 被检测一次，根据当前位置和目的地决定是否连接到另一个交通控制中心。

要实现上面描述的系统要求，利用本模型把系统分为以下 5 个不同类型的独立的实体：

(1)导航器只是一个决策者，根据状态信息和接收到的消息决定车辆的路径；

(2)GPS 系统；

(3)交通控制中心提供及时的路况信息服务；

(4)时控协同器通过特定事件模式施加时限监察；

(5)交互协同器在导航器、交通控制中心和 GPS 系统之间按时限要求建立连接和转发消息。

其中，导航器、GPS 系统和交通控制中心是模型中的功能计算实体。导航器、协同器安装在汽车上。交互协同规则和实时协同规则可存放在各自的规则库中，这样便于协同规则的修改和更新。

下面描述的是基本的处理流程。当汽车到达一个决策点时，用户触发导航器产生一个 Information-requested 事件，同时发出 Information-requested 消息，请求提供路况信息。时控协同器观察到 Information-requested 事件后，执行 impose (“Information-requested”,“Information received”,20,“received-timeout”)为该事件模式施加 20s 的时限约束。交互协同器根据测量的位置和目的地选择交通控制中心，如果与交通控制中心连接成功，交互协同器将发出 traffic-center 事件，表明了交互协同器已经在交通控制中心和导航器之间建立了一个连接，时控协同器执行方法 impose(“traffic-center”,“traffic-center”,5*60,“traffic-center-timeout”)为事件模式施加 5min 的时限约束。如果时控协同器在 20s 内没有观察到 information-received 事件，将会产生 Received-Timeout 事件。交互协同器观察到 received-timeout 事件，将导航器与 GPS 连接。交互协同器当观察到 traffic-ceter-timout 事件后，根据汽车当前的位置和目的地，利用 find- traffic-center 服务查找

（下转第 65 页）