doi:10.3772/j.issn.1002-0470.2010.05.005

# 基于元组空间通信的扩展呼叫处理语言协同技术①

杨 ▮② 陈俊亮

(北京邮电大学网络与交换技术国家重点实验室 北京 100876)

**摘 要** 在前期的工作中以 IETF 颁布的呼叫处理语言(CPL)为蓝本,发展出了面向综合通信业务的工作流语言——扩展呼叫处理语言(XPL)。XPL 不但能够描述呼叫类业务,还可以描述短信、彩信等数据类业务。在前期工作的基础上,在 XPL 中引入了一种基于元组空间通信的协同技术,通过这种技术,XPL 可以开发出内容更加丰富多彩的业务,不同的终端用户可以实现比较复杂的交互,从而在一定程度上满足融合网络条件下业务种类多样化的需求。

**关键词** 扩展呼叫处理语言(XPL),工作流,协同,元组空间

## 0 引 言

随着电信网络和互联网络向下一代网络的方向演进,如何快速灵活地开发种类丰富的新型增值业务成为电信领域和计算机领域所面临的一个重要问题。业务开发的一个基本问题是如何描述业务需求。可扩展标记语言(XML)因为具有易于人和机器的理解、和底层实现语言无关、易于图形化表示等优点而成为业务描述语言的重要发展方向之一。基于 XML 的语言可以分为通用型语言和面向特定领域的专业语言两种类型。通用型语言以 IBM 的业务过程执行语言(business process execution language,BPEL)[1]为代表,其主要针对通用型流程的控制,接近于高级语言的水平,已经成为工作流领域的工业标准。面向特定领域的专业语言有很多,在电信领域面向呼叫流程控制的具有代表性的语言有国际互联网工程任务组(IETF)的呼叫处理语言(calling process language,CPL)[2,3]、W3C 的呼叫控制可扩展标记语言(CCXML)[4]和 JAIN 论坛的业务生成标记语言(SCML)[5,6]等,其语言元素本身就是对呼叫处理的高度抽象,这些专业语言已经或正在成为国际标准。相比较而言,通用型语言有适用面广的优势,缺点是语言复杂,开发效率低,对开发人员的要求高。专业语言的优点是语言简单,开发效率高,对开发人员的要求低,但不能描述特定领域以外的业务。快速开发、部署业务是企业保持竞争力的关键之一,所以专业语言具有相当的研究价值。

我们认为 CPL、CCXML、SCML 等专业型语言的电信业务描述方法存在以下不足:第一,下一代网络是业务驱动的网络,网络融合条件下的业务特点应该是种类丰富、通信手段多样化,能够集成传统的呼叫类通信手段和新兴的数据类通信手段。但是 CPL 等业务描述方法依然只针对呼叫类处理,难以描述融合网络业务的需求。针对这一问题,我们对 CPL 的业务描述能力进行了较大的扩展,不仅能够使用传统呼叫通信资源,而且可以使用短信、彩信、Web、地理信息服务等其他数据类型的通信资源,关于这些工作已有另外的文章专门作了介绍[7,8]。第二,专业型语言需要专用的应用服务器的支持,因此特别重视语言安全性,像 CPL、CCXML、SCML 等专业型语言都没有并发控制的能力,每个业务脚本在业务逻辑描述上基本上是孤立的,在应用服务器上的业务实例之间一般也是彼此孤立的,不互相发生联系,不能够进行协同,其结果是业务功能简单,不能够完成复杂的操作,因而大大地限制了这些专业型语言的描述能力和所开发业务的种类和特色。本文在面向综合通信业务的工作流语言——扩展呼叫处理语言(extended-calling process language,XPL)中设计实现了一种基于元组空间(tuple space)通信的协同技术,该项目技术在保持安全性的前提下可使 XPL 具有较好的并发协同能力,适合在融合网络条件下开发较为复杂的业务。

① 国家自然科学基金(60432010)和 973 计划(2007CB307100)资助项目。
② 男,1978 年生,博士,助理研究员;研究方向:通信网,通信软件;联系人,E-mail: yangkiss@people.com.cn
(收稿日期:2007-04-11)

## 1　面向综合通信服务的语言 XPL 简介

呼叫处理语言(CPL)是一种主要用来处理呼叫转接流程的工作流语言,我们在其语法基础上引入很多新的语言特性后可以描述较为复杂的呼叫类业务如放音收号,发起第三方呼叫等,及彩信、短信、定位、Email、数据库操作等数据类业务,扩展后的语言称为扩展呼叫处理语言(XPL)。业务开发者可以通过图形化开发工具或者手工书写脚本的方式来开发业务流程。我们实现的 XPL 应用服务器曾于 2007 年在国家发改委中国下一代互联网示范工程(CNGI)通用业务平台项目支持下在运营商实际网络环境中运行。

关于 XPL 的详细定义参见文献[7],为了讨论方便,现将 XPL 中部分语法定义如下:

**定义 1**　XPL 是一种基于可扩展标记语言(XML)的流程描述语言,包含了一系列的特定标签,每个标签代表了特定的功能。XPL 通过标签之间的前后衔接来组成业务的执行流程:父标签的子标签代表了业务下一步所要执行的操作,一个父标签若有多个子标签,则选择其中满足条件的继续执行。

XPL 在 CPL 的基础上通过增加特定的标签来扩充其能力,称之为新增业务能力标签。

**定义 2**　新增业务能力标签::= <完成 CPL 中所没有的特定的业务动作>,包含的标签数量很多,这里列举一些,如 <sendSMS> 发送短信, <getLocation> 取得移动用户当前位置, <sendMMS> 发送彩信, <makeCall> 第三方发起的呼叫, <proxy> 转接当前的呼叫,等等,为简化起见,标签省去了属性和子标签说明。

业务开发者使用 XPL 来描述业务流程,开发出一个 XPL 业务脚本,这个 XPL 业务脚本部署在应用服务器上,通过 XPL 脚本翻译器转换成以 EJB 形式存在的可执行代码,之后部署在 EJB 容器中,依靠中间件的线程管理机制,形成中间件容器中的多个 XPL 业务实例。当用户使用终端设备时,各种事件通过融合网络的 Web Services 网关上报给支持 XPL 的应用服务器,通过特定的上报事件触发 XPL 业务实例开始按业务流程执行,例如用户拨打特服号码、上发短信至特服号码、通过 Web 页面发起请求等,触发执行一段 XPL 所描述的业务流程,完成一次 XPL 的会话。由于 Web Services 的无状态性,应用服务器中的消息分发器来为每次上报的 Web Services 消息找到对应的XPL业务实例,即维护特定的用户和特定的 XPL 业务实例之间的对应关系(参见图 1)。

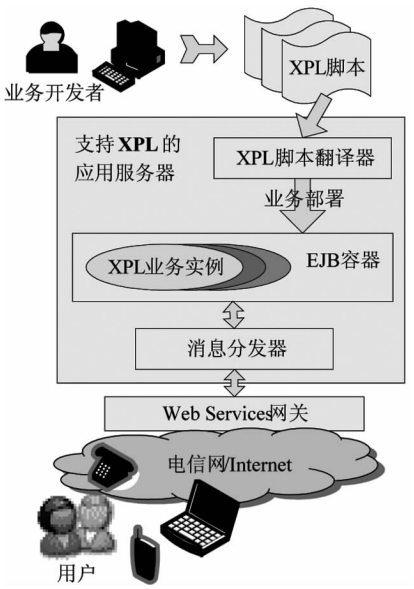


**图 1　XPL 应用服务器的体系结构**

**定义 3**　XPL 的会话 Session = (Invoke, {Action}, Return), Invoke 指用户使用终端设备发起一次 Web Services 请求触发执行一个会话(如拨打特服号码、上发短信至特服号码、点击 Web 的用户等), Return 指该次请求返回给用户终端, Action 指这段 XPL 业务流程在执行过程中应用服务器所完成的动作如数据库查询、发起第三方拨号、下发短信/彩信等。

**定义 4**　业务发起者 TriggerUser::= <完成 Invoke 动作的用户(如拨打特服号码的用户、上发短信至特服号码的用户、点击 Web 的用户等)>,业务参与者 InvolvedUser::= <Action 动作所涉及到的被动参与的用户(如发起第三方拨号时建立通话的用户,下发的短信/彩信的接收方等)>。

**定义 5**　单个 XPL 业务实例的生命周期 = {(Terminal, Session i)}, i = 1,2,3…。Terminal 指 TriggerUser 的通信终端。

CPL、CCXML、SCML 中也有类似 Terminal 的概念,一般指电话终端,XPL 的 Terminal 除此之外还包括短信终端、Web 终端等。虽然种类增加了,但是由于单个 XPL 业务实例的生命周期中的每个 Session 都只能由同一个 Terminal 发起(其原因是消息分发器不可能为未知的 Terminal 维持和特定的 XPL 业务实例之间的对应关系),不同的 XPL 实例之间是孤立的,因此不同的 TriggerUser 之间不能够彼此协同,

所描述开发的业务多样性大打折扣。针对这一问题,我们采用了一种基于元组空间通信的 XPL 协同技术来解决。

## 2 元组空间通信与 XPL

元组空间通信,也称为 Generative Communication,最早是由 Gelernter 在 Linda 语言[8]中引入的作为一种基础的通信模型,目前,被引入到分布应用集成中作为移动计算的一种通信方式[9]。在分布应用集成中,独立的系统可以通过元组空间作为彼此之间进行通信的中介。发送的请求信息中的变量列表形成了一个元组,发送应用程序将元组插入元组空间,每个元组在元组空间中都是唯一的。接收者同样在它的元组获取请求中给出它自己的变量列表。在元组空间中,有与请求元组相匹配的元组时,获取发生,否则,请求者将等待直到元组提供,分布应用系统可以通过这种机制达到系统间的数据与控制的交互[10]。

分布应用协同需要在应用系统间传递异地的数据流和控制流,而数据流和控制流的传输需要有效的载体[11]。这种载体是一个相对独立的实体,需要满足系统分布特性的需求,如系统对耦合性,实时性,多对多,点对点等空间、时间、拓扑特性的需求[12]。对于分布式应用协同来说,通常采用消息中间件、事件中间件、元组空间、远程过程调用(RPC)、远程方法调用(RMI)等方法作为分布应用协同的载体。这些方法可以分为两种模式:一是共享存储模式,例如消息中间件、事件中间件、元组空间,发送者把消息放共享存储区中由接收者自己来取,如何确定该由哪一个接收实体来取和取哪一个发送实体发送的消息,由消息内容决定,发送者和接收者确定一个协议就可以;二是方法调用模式,如 RPC、RMI 等,在这种方式下发送者需指明接收者。两者都可以实现多个 XPL 业务实例之间的交互,只是一个是消息接收者主动的查询到来的消息,一个是消息接收者被动的接收到来的消息。由定义 3、4、5,我们可以看出发送者实例和接收者实例之间是完全独立的,如果采用方法调用的形式,有如下缺点:

(1) 发送者实例和接收者实例在空间上是紧耦合的。发送者实例必须要知道接收者实例究竟是谁,才能调用其上的方法,但是发送者实例和接收者实例是自治的,彼此并不知道对方的存在。

(2) 发送者实例和接收者实例在时间上是紧耦合的。接收者实例需要阻塞地等待发送者实例的调用,即需要保持和发送者的同步。当有多个消息发送者时,因为发送者实例调用的到达时间是随机的,接收者实例必须要处理所有可能的到达序列组合。假设接收者实例面对 $N$ 个独立的调用,那么接收者实例的处理分支就会有 $N!$ 个,反映在 XPL 脚本中体现为业务开发者需要考虑的流程分支成级数增长,这样就背离了 XPL 这种面向特定领域的语言灵活好用的设计初衷。

(3) 有可能多个实例彼此之间相互等待调用,陷入死锁状态。这也不是一种面向特定领域的语言所希望出现的。

所以,只有采用共享存储模式才有可能解决这一问题。相对于消息中间件、事件中间件方法,Linda 模型非常简单,它仅支持基本的数据类型和有限的 5 种操作原语,而且操作的匹配机制固定且单一,但它的简单性恰恰适合应用在 XPL 这种面向特定领域的工作流语言中,因为这种面向特定领域的工作流语言往往需求比较的简单和单一,如果引入过于复杂的操作反而会破坏语言易懂、易用的特点。Linda 从元组空间里检索元组采用结合模式匹配(associative pattern matching)机制,和检索有关的原语是一个可以含形参的元组模式(tuple schemata),也称为模板,一个元组 T 和一个模板 M 匹配的条件是:

$$(\#(T)=\#(M))\wedge(\forall j\in\{1,\cdots,\#(T)\}:$$

$$(t_j(M)=t_j(T))\wedge((V_j(M)=V_j(T))\vee(V_j(M)=\text{null})))$$

我们在 Linda 模型的基础上来定义 XPL 中负责控制协同的语言元素。

**定义 6** 对元组空间的写操作::= "< Notify >",该标签有 4 个属性(Q _ Name, MsgContent, Order, Clear)。Q _ Name 是目的元组空间名,若不存在则新建一个采用该名称的元组空间。MsgContent = {S}是消息内容,S 为一字符串,Order 是对元组空间中消息的操作顺序,Clear 表示是否先清空该元组空间。标签含有 4 个属性。属性 Name 指定写入的元组空间的名称。属性 Content 指定写入的元组空间的内容,为一字符串,各个域值用";"隔开。属性 Ording 用来选择从元组空间的头部还是尾部写入。属性 Clear 用来表示在写入之前是否需要将该元组空间清空。

**定义 7** 对元组空间的读操作::= "< Pick >",该标签含有 5 个属性(Q _ Name, Block, TimeOut,

Order,Condition,Clear)从指定的元组空间接收消息。Block 表示若无满足条件 Condition 的消息,是否需要阻塞等待该消息,TimeOut 为等待时长。Condition = (MatchField, MatchContent, MatchType)是三个等长数组构成的三元组,数组 MatchFeild 的每个元素为待比较的 S 在 MsgContent 中的位置,数组 MatchContent 的每个元素是和指定的 S 作比较的字符串,数组 MatchType 的每个元素是字符串匹配策略。

通过在 XPL 中引入定义 6 和定义 7 的语言元素,我们将其发展成为了具有并发协同能力的面向电信领域的业务描述语言。相应地,支持 XPL 的应用服务器需要在原有的基础上增加一个元组空间处理器模块。

由于开发 XPL 应用服务器时 EJB 规范尚不支持支持有状态会话 Bean 的定时器机制,并且不支持线程阻塞控制,所以定义 6 中的 TimeOut 和 Block 功能无法在 EJB 容器中实现。故对应用服务器的整体架构扩充为如图 2 所示。当 XPL 业务实例按照标签组成的业务流程运行至 < Notify > 、< Pick > 时,XPL 业务实例通过消息分发器调用元组空间处理器的 Web Services 方法,由元组空间处理器来完成定义 6 至定义 7 的功能。由于元组服务器也采用 Web Services 技术,所以它和 XPL 应用服务器的交互模式可以参 ParlayX 网关,消息分发器的工作模式不用修改,保证了 XPL 应用服务器的通用性。

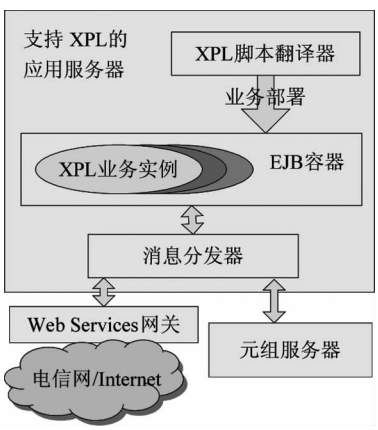


**图 2　应用服务器与元组服务器**

## 3　引入元组空间通信后 XPL 新增的典型功能

如前所述,在引入元组空间通信以前,TriggerUser 彼此之间不能够发生联系(如图 3(a)所示)。当引入元组空间通信之后,TriggerUser 彼此之间可以通过并发协同共同地来完成比较复杂的业务,业务模式从用户单纯使用网络资源发展为用户之间通过网络资源来进行互动(如图 3(b)所示)。在这里我们用 XPL 描述一个较为复杂的业务,说明引入元组空间通信后 XPL 新增的典型功能。

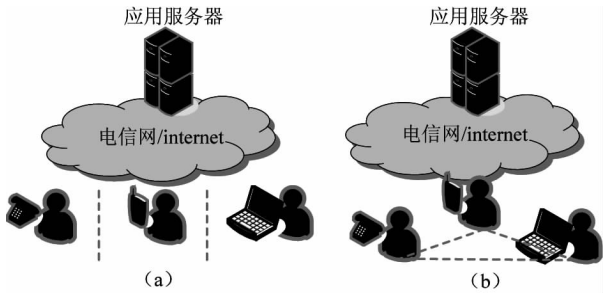


**图 3　引入元组空间通信后终端用户之间的关系变化**

业务场景是:有陪聊代理多人,代理使用手机以短信的方式通知业务其开始工作,开始接受客户的电话咨询;客户拨打特服号码,业务分配一个代理为他服务。

该业务由两个脚本构成,脚本 login.xml 用来处理代理通过短信登陆/注销系统,脚本 service.xml 用来处理用户的呼叫。代理首先通过短信发送验证码到短信特服号码,login.xml 使用数据库操作进行验证码和短信源号码(即手机号)的匹配验证,若不匹配则向代理发送短信提示错误,若匹配则将该手机号放入共享消息元组空间。当客户拨打呼叫特服号码时,service.xml 首先将用户排队。若该用户排第一位,service.xml 查看共享消息元组空间中有无空闲代理,如果有则将当前的呼叫对所有的空闲坐席进行顺呼,若成功则进入通话状态,通话完代理挂机后业务向客户发送短信回执。由于空闲代理可以自由使用手机和客户以外的人通话,故上述转接有可能不成功,此时业务选择一个空闲最久的代理用短信提醒他主动给该客户回电。如果该客户不是排第一位或者没有发现空闲的代理则向客户循环放音等待。service.xml 通过元组空间操作完成代理和客户的排队。

图 4、图 5 中列出用 XPL 描述的业务流程中的主要部分。为了节省篇幅,没有写成 XML 形式,其中数据库操作 < SqlSelect > 、放音操作 < runUI > 、发送彩信操作 < sendMMS > 、发送短信操作 < sendSMS > 是在 XPL 中已有的语言元素,为简化起见,没有列出属性。

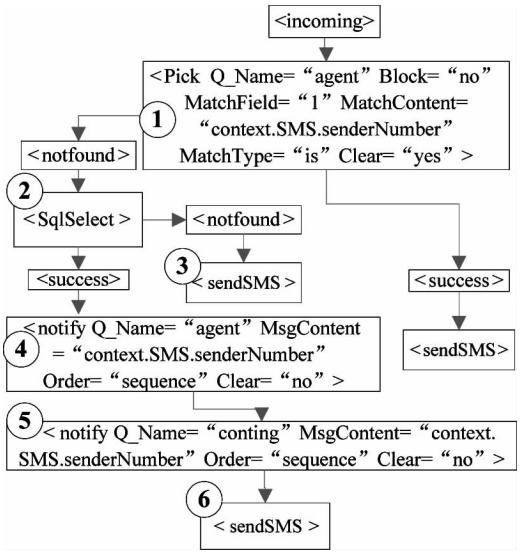


**图 4　代理登陆脚本 login.xml**

元组空间数据结构说明：Q _ Name = "agent"，S 个数为 1，值为代理手机号；Q _ Name = "counting"，S 个数为 1，值为固定字符串"counting"，该元组空间作为计数器表示当前可服务的代理人数；Q _ Name = "user"，S 个数为 1，值为客户手机号。

脚本 login.xml 流程说明：代理发送验证码短信到特服号，若元组空间 agent 中已有该代理手机号①，完成注销流程，否则在数据库中查找对应的验证码进行验证②（简单起见省略 CMP 和 EJB _ QL 书写的脚本），不成功给代理发送短信提示③，若成功将代理手机号放入元组空间 agent④，当前可服务的代理人数加 1⑤，短信提示登陆成功⑥。

脚本 service.xml 流程说明：客户拨打特服号码接入系统，进入元组空间 user①，客户是否排第一位②，不是的话给客户放音③让客户等待，< run _ UI >

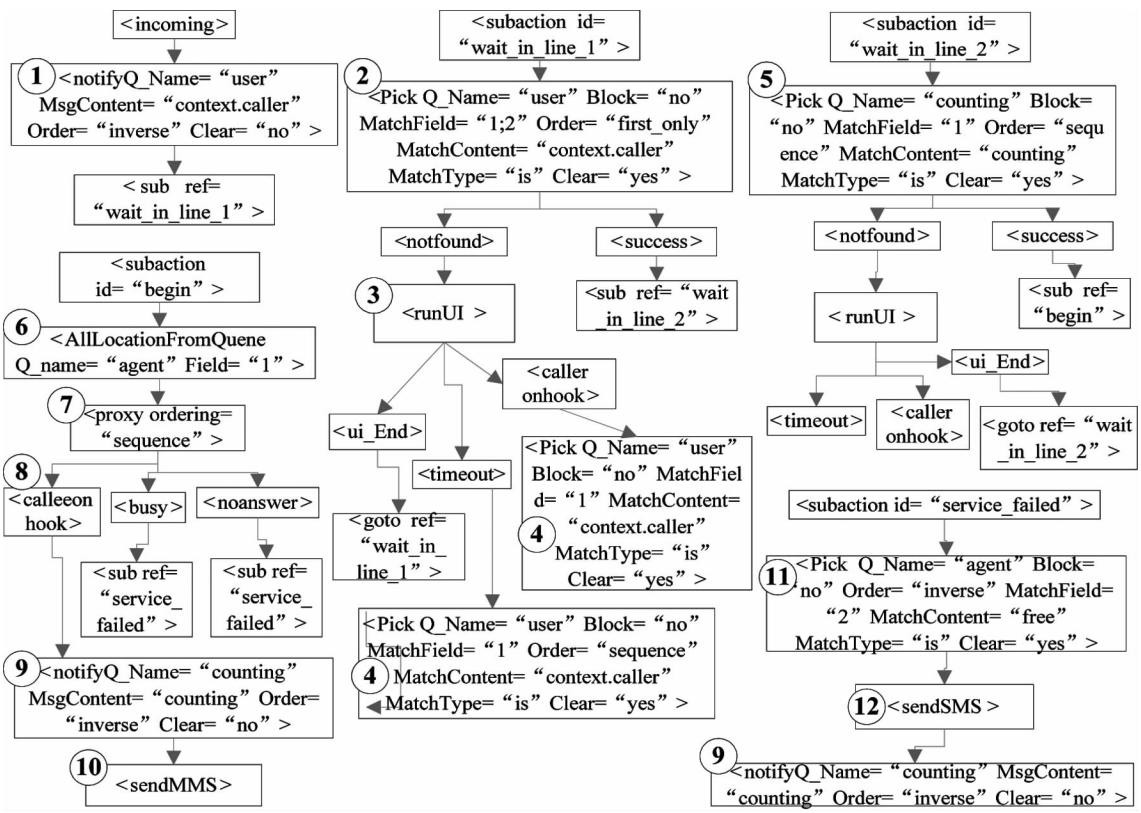


**图 5　用户服务脚本 service.xml**

是分段标签，网络上报放音结束事件 < ui _ End > 驱动 < goto > 进入循环体 wait _ in _ line _ 1，发生超时 < timeout > 或用户挂机事件 < calleronhook > 则将该用户从元组空间 user 中清除④并结束业务。若客户排第一位则查看当前是否有可服务的代理⑤，若没有放音等待，若有将 agent 中所有代理手机号作为呼转的目标地址⑥并开始顺呼⑦，和最先摘机的代理建立连接，代理挂机⑧，将计数器 counting 加 1⑨，给用户发送代理的彩信名片（简单起见通过 URL 访问彩信图片）⑩。代理在工作期间自由使用手机，不可避免的⑦有可能不成功，业务会向代理发一条短信指示 agent 中排第一的代理稍后给用户回电。

## 4 相关工作比较

下面主要通过 XPL 和 CPL、CCXML、SCML 4 种同样面向通信领域的流程描述语言在多用户协同能力方面的对比,来进一步说明 XPL 的特点。

**表 1 面向电信业务领域语言的对比**

| | 底层协议 | 多用户协同 | 协同的控制方式 | 协同的用户之间的通信途径 | 协同的通用性 | 适用的典型业务 | 使用难易程度 |
|---|---|---|---|---|---|---|---|
| CPL | SIP/H.323 | 无 | 无 | 无 | 无 | 无 | 无 |
| XPL | ParlayX 及其他 Web Services | 支持 | 不同的业务实例之间通过元组空间进行通信 | Web/SMS/MMS/Calling | 强 | 多用户/多客服系统 | 较难 |
| SCML | JCC | 很弱 | 同一个业务实例进行第三方发起的呼叫 | Web/Calling | 弱 | 点击拨号 | 容易 |
| CCXML | JTAPI | 较弱 | 在呼叫会话中控制多个呼叫方的加入和离开 | Calling | 弱 | 电话会议 | 较容易 |

CPL 主要面向 SIP 和 H.323,不是和底层协议完全无关的语言。CCXML、SCML、XPL 是和底层协议无关的语言。CCXML 基于 JTAPI (java telephone API)、SCML 基于 JCC (java calling control),XPL 主要基于 parlay Web Services 及扩展的接口,所以 XPL 描述生成的业务更适合在融合网络的条件下运行,XPL 更适合第三方增值业务提供商用来开发业务。XPL 可以认为是一种具有并发协调能力的 Web Services 组合语言。

CPL 主要完成呼叫转移类的简单业务,基本上可以认为其没有多用户协同的能力。SCML 可以通过网页由第三方建立起主叫和被叫之间的通话,可以认为有微弱的多用户协同能力(实际上 XPL 也可以通过 < makeCall > 来完成类似的功能,只不过在这里着重讨论 XPL 在引入元组空间通信之后具有的多用户协同能力)。CCXML 用来控制同一个呼叫会话中的多个呼叫方的动作。相比较而言,XPL 的协同控制模型是一种一般性的并发控制方法,通用性最好,通过对元组空间的恰当控制,可以开发出像多用户/多客服这样的复杂业务,其他的面向通信领域的语言不具备这样的能力。当然在开发这种复杂业务时对用户的能力要求相比其他几种语言要高。另外,由于当前 PalayX 标准尚不具有对呼叫腿(calling leg)的控制能力,XPL 还不能够描述电话会议类业务。

## 5 结 论

在前期的工作中,我们在 IETF 颁布的呼叫处理语言 CPL 的基础上发展出了一种新的工作流语言 XPL,本文对 XPL 中引入的一种基于元组空间通信的并发协同技术进行了详细介绍,通过该项技术使得 XPL 的业务描述能力有了很大的增强,适合在融合网络的条件下开发较为复杂的业务。今后的工作需要在实际使用中进一步完善 XPL 的这种协同控制技术,推动 XPL 的实用化。

**参考文献**


[1] Andrews T, Curbera F, Dholakia H. Business process execution language for web services. version 1.1. http://www-128. ibm. com/developerworks/library/specification/ws-bpel/: IBM, 2007

[2] Lennox J, Wu X, Schulzrinne H. RFC 3880, call processing language (CPL): a language for user control of internet telephony services. http://www.ietf.org/rfc/rfc3880.txt: IETF, 2004

[3] Lennox J, Schulzrinne H. RFC 2824, call processing language framework and requirements. http://www.ietf.org/rfc/rfc2824.txt: IETF, 2000

[4] Auburn R J, Voxeo. Voice browser call control: CCXML. version 1.0. http://www.w3.org/TR/2005/WD-ccxml-20050629/: W3C, 2005

[5] Bakker J L, Jain R. Next generation service creation using XML scripting language. In: Proceedings of the 2002 IEEE International Conference on Communications (ICC 2002), New York, USA, 2002.4.2001-2007

[6] Bakker J L, Jain R. A service creation markup language for scripting next generation network services. http://ietfreport.isoc.org/all-ids/draft-bakker-jain-scml-00.txt: IETF, 2002

[7] 杨▮,温嘉佳,陈俊亮.扩展呼叫处理语言:一种面向综

合通信服务的语言.软件学报, 2008,19(5):1224-1233
[8] 杨▮,陈俊亮,孟祥武.一种面向 LBS 的电信增值业务生成方法及实现.软件学报,2009,20(4): 965-974
[9] Carriero N, Gelernter D. Application experience with Linda. In: Proceedings of the ACM/SIGPLAN Symposium on Parallel Programming. New York: ACM Press, 1988. 23. 173-187
[10] Matsuoka S, Kawai S. Using tuple space communication in distributed object oriented languages. In: Proceedings of the Object Oriented Programming Systems, Languages and Applications, San Diego, California, USA, 1988. 276-284
[11] Omicini A, Zambonelli F. Tuple centres for the coordination of internet agents. In: Proceedings of the 1999 ACM Symposium on Applied Computing, San Antonio, Texas, USA, 1999. 183-190
[12] 徐罡,黄涛,刘绍华等. 分布应用集成核心技术研究综述. 计算机学报, 2005,28(4):433-444
[13] Andrews G R. Paradigm for process interaction in distributed programs. *ACM Computing Surveys*, 1991, 23(1):49-90

# Extended-calling process language coordination technique based on tuple space communication

Yang Qin, Chen Junliang
(State Key Laboratory of Net Working and Switching Technology, Beijing University of Post and Telecommunication, Beijing 100876)

**Abstract**

In this study, a coordination technology based on tuple space communication was added to the extended-calling process language (XPL), an integrated-communication-services-oriented workflow language developed in the previous work of the study from the calling process language (CPL) published by the internet engineering task force (IETF), which has the ability to describe both calling and digital services such as SMS, SMM and so on. Through the technology, the XPL can have the ability to describe more rich and colorful services, and different terminal end users can have complicate interaction, so this can meet the requirement of the variety of the services to some extend under the condition of net merging.

**Key word**: extended-calling process language (XPL), workflow, coordination, tuple space