计算机科学2000Vol.27№.3

基于 Agent 的分布式业务过程管理系统 DBPM/AB 的设计和实现*)

The Design and Implementation of Agent Based Distributed Business Process Management System

宋德舜　麦中凡
(北京航空航天大学计算机系　北京100083)

Abstract　In the large-scale, heterogeneous, distributed and dynamic environment, it is unefficent for the traditional WFMS (workflow management system) to manage business process. Therefore, this paper provides a new approach——DBPM/AB (Agent Based Distributed Business Process Management System). The key idea of this system is that, in accordance with the structure of organization, responsibility of enacting business process is delegated to a number of autonomous Agents, and by their negotiate and coordinate with each other, this Agents together accomplish business goals. So the DBPM/AB is more agile and robust than traditional WFMS.

Keywords　Business process, BPMS, Agent, Agency, Service, Information sharing

1　引言

业务过程的执行效率决定了现代企业的成败，业务过程管理系统 BPMS 旨在提高执行业务过程的质量，降低其成本，加快其执行速度。然而，对大规模、异质、分布和动态的企业计算环境，使用传统的 BPMS（主要指目前各种工作流管理系统[1]，如：Notes、Exotica、METEOR2、WIDE 等）来管理业务过程很难实现企业的上述业务目标[2]，这主要是因为，在当今竞争激烈的商务环境中，WFMS 很难有效管理具有如下特点的业务过程[3]：1）过程的动态性和不可预见性。2）过程在物理上的分布性及其固有的高并发性。3）过程可能涉及多个企业。4）企业内部各部门相对独立，控制自己的活动、信息和资源。面对激烈的市场竞争，企业为了获得最大利润，需不断进行业务过程重组 BPR（Business Process Re-engineering）和组织关系的调整，因此要求 BPMS 也要适应这种变化。5）技术环境的异质性导致对 BPMS 设计、开发和集成的困难。

仔细分析企业业务过程的运行后，我们发现：企业范围内的业务过程实际上是为了完成特定目标而开展的一系列跨部门（或组织）活动。一方面，完成活动的责任按照组织结构分配给各部门；另一方面，完成这些活动时，由于各部门的自主性，必须靠它们相互协商和协调，共同来完成业务目标。若 BPMS 也遵照企业的这种营运模式，则必须将在应用设计时要考虑的许多运行时问题，都交给执行系统，由系统在运行时动态做出决定，如：由谁在何时执行哪个活动，每个活动能消耗多少资源等。此时，执行系统不再是执行一个预先定义了执行资源的业务过程，而是根据当前环境动态决定业务活动的执行，因此能实现对复杂业务过程的有效管理。Agent 技术的出现，和它所展现的自主性、社交能力、主动性（proactiveness）和反应性[4]，为实现这种类型的 BPMS 提供了很好的基础。在研究当前世界上几种主要的基于 Agent 的 BPMS[5~10]后，我们提出了 DBPM/AB 的概念框架，介绍了 Agent 的体系结构。

2　DBPM/AB 系统的概念框架

系统概念框架反映了企业的组织结构，描述了业务过程的执行模式。该结构中，代理处（Agency）用来描述组织机构，它可以是企业的内部部门，也可以是整个企业；代理（Agent）用来代表 Agency 的利益并负责和外界交互，是业务过程的执行主体。如图1所示。

首先，每个 Agency 能完成一些原子业务活动，如填写表单，审批文件等，我们称之为任务。将任务按一定条件组合后，得到该 Agency 能完成的服务，由代表该 Agency 利益的 Agent 负责对外提供。由于 Agency

*)本课题得到国家自然科学基金项目资助（项目编号69673003）。宋德舜　硕士生，研究方向为多 Agent 系统、软件工程。麦中凡　教授，研究方向为软件工程、过程工程、软件 Agent 等。

的自主性和对资源的独立控制权，当 Agent 要求得到其他 Agent 提供的服务时，不得无条件命令对方，而是在服务执行前，由服务的双方就完成该服务达成某种协议（我们称之为服务层合同 SLA，Service Layer Agreement）。只有具备了 SLA，Agent 才能执行与之相关的服务。其次，企业范围内大规模的业务过程总是由多个部门协作完成；体现在 DBPM/AB 中，一个业务过程由多个 Agent 提供的服务按照某种顺序组成（一个服务可能属于多个业务过程），业务过程的执行是按照定义好的服务执行步骤，在 SLA 条件下，由多个 Agent 执行各自承诺的服务，共同完成整个业务过程。可以看出，上述过程包括三个阶段：服务定义，服务提供，服务执行。我们称这三个阶段为服务的生命周期，而 DBPM/AB 就是要对服务的各个阶段提供最有效的支持。最后，由于 Agency 有各自的信息表达方式，当 Agent 通信时，为了能够互相理解所交流的信息，必须有一个双方都能理解的信息模型，我们称之为 Agent 间的信息共享模型。下面我们就这些问题分别进行讨论。

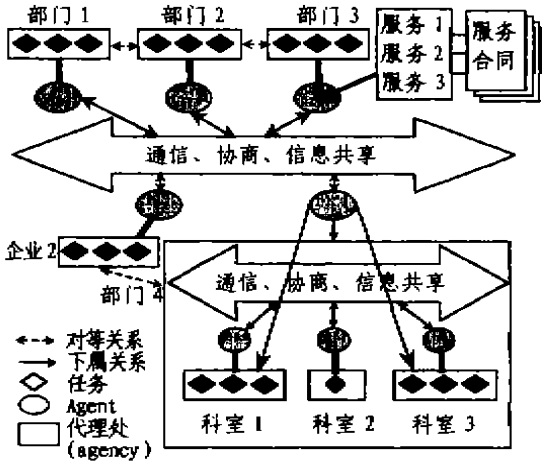

图1 DBPM/AB 系统的概念框架

2.1 代理处和代理

一个企业包括多个部门和一个企业经理。每个部门包含多个职员和一个部门负责人，它又可能负责多个子部门。显然：1）部门是递归定义的。2）企业包括两种组织关系：对等（平行部门）和分层（上下级部门）。我们用 Agency 和 Agent 为企业的这种组织结构建模，对 Agency 的定义如下：

$$Agency = agent + \{task_i\} + \{agency_j\}; i, j: integer, i \geq 0, j \geq 0$$

一个 Agency 包括：一个 Agent，一个（或空）任务集，一个（或空）子代理处集。其中：Agent 是 Agency 的管理者，当和其他 Agency 交互时代表 Agency 的利益。另外，外界和 Agency 的任何通信都必须通过 Agent。任务 Task 是 Agency 所能完成的由人或自动程序执行的原子业务活动。子 Agent 和 Agent 的关系可看作是组织结构中的上下级关系，当 Agent 要求得到子 Agency 的服务时，子 Agent 不得无理拒绝请求。但是，子 Agent 并不是盲目服从，它有一定程度的自主性，Agent 必须和它协商提供服务的条件。而对于对等 Agent 来说，由于它们是平行的关系，它完全可以拒绝从它的对等 Agent 那儿发来的服务请求；只有当接收方 Agent 认为提供服务对它有利时，双方才会就服务进行协商。

2.2 服务的生命期

服务指 Agency 对外表现的业务能力，不仅指企业提供给它的客户的服务，还指企业各部门间彼此提供的服务，如：打印表单、数据统计、信息系统设计等。它的结构类似于函数：接受外来输入，进行处理，产生输出。定义如下：

$$service = sequ\&cond[\{task_i\} + \{service_j\}]; i, j: integer, i \geq 0, j \geq 0$$

服务由任务和子服务按照某种顺序和条件组成，最简单的服务是任务，最复杂的服务是整个业务过程，可以看出，服务最终由任务实现。服务的生命期包括三个阶段，如图2所示。

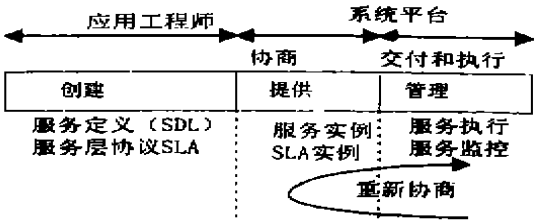

图2 服务的生命期

服务生命期的第一阶段是应用工程师定义服务和与之相应的 SLA。我们采用服务描述语言 SDL 来描述服务（类似过程定义语言）、规格说明服务的实现。其语法如下：

service :: = ⟨service_name⟩⟨input⟩⟨output⟩⟨guard⟩⟨body⟩

⟨service_name⟩：指服务名字，对整个企业的所有业务过程是唯一的。

⟨input⟩：输入集；服务执行前用到的强制或可选信息，由服务的接收方（Client）或服务提供方（Server）提供。

⟨output⟩：输出集；服务完成后返回给 Client 的结果信息。

⟨guard⟩：保护条件；服务执行前的条件检查，只有它为真才开始执行服务。

⟨body⟩ ::= ⟨block_type⟩ ':' ⟨block_identifier⟩ '{' ⟨block_list⟩ '}→' ⟨completion_expression⟩。

⟨block_type⟩ ::= 'sequence' | 'can_para' | 'must para' | 'loop'。

⟨block_list⟩ ::= ⟨block_type⟩ ':' ⟨block_identifier⟩ '{' ⟨execution_list⟩ '}→' ⟨completion_expression⟩。

其中：sequence：指顺序服务；can_para：指能够并行执行的服务；must_para：指一定要并行执行的服务，loop：指在条件没满足前服务一直循环。⟨block_identifier⟩：指出服务（子服务）的完成状态。有三种：true、false、unknown。⟨execution_list⟩：由逗号分开的条件（测试某项信息）和子服务构成。⟨completion_expression⟩：指出服务（子服务）正确完成的条件。⟨body⟩：说明服务的实现，服务体由一个或多个子服务组成，子服务块可以是任务，也可以是它的子 Agent 提供的服务或它的对等 Agent 能提供的服务。

SDL 说明了 Server 如何实现服务后，Client 通过引用服务原型调用 Server 提供的服务。这样便实现了对服务的封装，只要原型不变，服务实现的变动（也就是 agency 内业务过程的变动）不影响整个业务过程的执行，系统具有更好的灵活性和可连续演进性。在服务体中，对任务的调用也是通过原型实现的。它的语法格式和服务原型相同。语法如下：

service_prototype ::= ⟨service_name⟩ ⟨input⟩ ⟨output⟩。

task_prototype ::= ⟨tast_name⟩ ⟨input⟩ ⟨output⟩。

用 SDL 确定服务的功能后，接下来就是确定该服务的 SLA 模版，它描述了协商的具体内容。

服务生命期的第二阶段是通过面向服务的协商确定提供服务的条件，也就是将 SLA 模版实例化的过程。然后，根据该 SLA 实例，对服务进行相应的实例化。

服务生命期的第三阶段是服务的调度和执行阶段。当 Agent 接到服务执行请求时，按照 SLA 为服务分配执行资源，执行服务，并监控和纠正服务执行过程中发生的异常，当服务完成后将结果返回给请求服务的 Agent。该阶段由 Agent 自动完成（我们将在第3节中详细讨论）。

2.3 信息共享

上面提到，各自主 Agent 都有自己的信息表达方式，为了保证交流信息的一致性，我们使用共享信息模型（SIM），共享的信息模型由许多基本的信息对象类组成，每个类的前缀是信息模型 SIM 的名字，各 Agent 的信息模型都有到 SIM 的映射。

Agent 服务在服务描述中，服务的输入和输出使用负责该服务的 Agent 的信息模型。而服务原型的输入和输出则使用 Client 的信息模型来定义，Client 的服务原型和 Server 的服务定义有以下三个特点：1）服务名必须一致；服务名对所有 Agent 是共同的，不论其信息模型如何，2）每个信息对象有一个前缀，即定义该对象的信息模型的名字。3）Client 和 Server 使用的信息模型到 SIM 的映射。

3 Agent 体系结构

Agent 是本系统中执行业务过程的实体，负责管理 agency 的各种资源，并根据业务目标和 agency 当前的资源使用情况，完成：Agent 间面向服务的协商、自动调度并执行服务、服务的监控。体系结构如下图3：

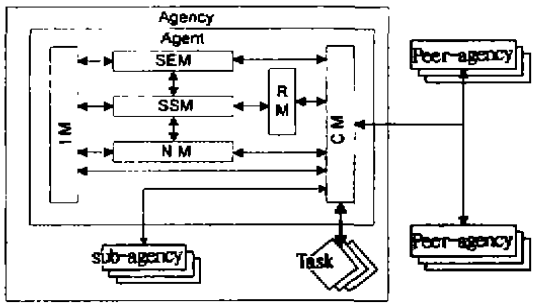

图3 Agent 体系结构

3.1 信息管理 IM

IM 是 Agent 的知识库，用来存放 Agent 对自己和环境的认识信息。管理的信息包括两部分：1）SI（Self Information）：它所有的子 Agent 和子 agency；它能提供的服务的 SDL 描述及该服务允许的最多并发调用；每项服务的 SLA 模版；它所承诺的服务的 SLA 实例；它管理的资源和当前可获得的资源；服务的当前调度计划；运行时信息（如：当前活动的服务，当前对活动服务的调用次数等，服务的执行情况）。2）AI（Aquaintance Information）：和它相关的对等 Agent；和它们协商的优先顺序；这些 Agent 所能提供的服务；这些 Agent 向自己承诺的服务的 SLA 实例；和它们交互的历史；它们的信息模型等。

3.2 协商管理 NM

当 SSM 调度某项服务时，若它没有相应的有效 SLA，则 SSM 通知该模块达成该服务的 SLA。NM 通过 IM 决定哪些 Agent 能够提供所需服务，根据自己的服务请求条件产生初始提议；当受到对方返回的提议后，再适当修改条件后产生自己的反提议，直到最后达到一个双方都能接受的协议或协商失败。这两种协商过程对所有 Agent 都相同，我们用协商协议来描

述，如图4。其过程如下：

1)当客户端说 cando 时协商开始。状态1→状态2。

2)服务器要么指示它能够(状态2→状态3)或它不能(状态2→失败)。

3)如果服务器承认它能够，或者客户端通过检查它的 AM 知道服务器能够，客户端发出提议(propose)，状态3→状态4。

4)服务器或者拒绝该提议(状态4→失败)，或者接受该提议(状态4→状态5)，或者提出反提议(状态4→状态6)。

5)如果服务器接受，客户端或者否决该协议(状态5→失败)，或者确认该协议(状态5→成功)。

6)若服务器反提议，客户端或者接受新协议(状态6→状态7)；或者拒绝(状态6→失败)，或者反提议一个新的协议(状态6→状态4)，在该阶段可能反复几次。

7)若客户端最终接受协议(状态6→状态7)，服务器或者确认(状态7→成功)，或者否决(状态7→失败)。

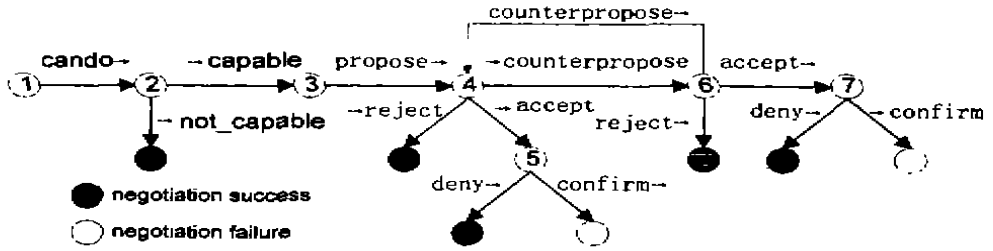

图4 DBPM/AB 的协商协议

3.3 资源管理 RM

该模块负责对 Agent 所属资源进行注册和注销管理。资源包括：执行任务的资源(人或程序)、执行服务的资源(能为他提供服务的子 Agent 或对等 Agent)。

资源的注册分三种情况。1)分布在各节点上的人或程序的注册。若资源表中无该资源实例，则必须指明它能完成的任务，然后再重新注册，RM 为该资源增加一条记录；2)有 SLA 的 Agent 的注册。当它加入系统时向接受服务的 Agent 发注册消息。3)子 Agent 的注册。当它加入系统时向父 Agent 发注册消息。

资源的注销有两种情况：1)资源正常注销前发送注销消息。2)资源所在节点因异常注销而没有发注销消息时，由 RM 强制注销。因此，SSM 调度服务前，RM 必须检查资源情况，以发现这种异常注销。

在业务过程的运行中，资源是动态确定的，因此 RM 必须检查资源的可得情况：1)如果执行某服务或任务的资源注册，RM 将该事件通知 SSM，SSM 将该资源分配给等待它的任务或服务。2)如果 RM 检查到执行某服务或任务的资源注销后，通知 SM 为这些任务或服务重新配置资源。

3.4 服务调度管理 SSM

当要执行 Agent 提供的服务时，该模块负责如下三部分功能：1)服务调度。逐步解析出用 SDL 描述的服务体的组成部分，为服务执行路径的每一个节点(任务、子 Agent 提供的服务、对等 Agent 提供的服务)分配执行资源，并提供执行该服务所必需的输入信息。由于资源的动态性，用五个队列(等待、就绪、运行、悬挂、完成)实现服务的调度和执行。其中，等待队列中的服务还没有分配到资源，等待分配；就绪队列中的服务已获得资源，等待执行；运行队列中的服务正在运行；完成队列中的服务已经结束，它占用的资源被释放，等待 SSM 分析其下一个子服务。悬挂队列中的服务得到资源，但推迟服务的执行，2)一方面，处理 RM 发来的资源事件：a)空闲资源注册事件：将该资源分配给等待队列中的合适任务。b)资源注销(正常或异常)事件：为执行队列中的任务重新分配资源。若分配不成功，将该服务放入等待队列中；否则，放入就绪队列中。另一方面，处理 SEM 发来的服务完成消息，从完成队列中取一个子服务，为它解析和调度下一条子服务，直到服务完成。3)最后，处理服务执行中 SEM 不能解决的高层异常，通过 NM 重新协商该服务的 SLA，或通过 SEM 终止该服务。

3.5 服务执行管理 SEM

该模块负责在服务管理(服务生命期第三阶段)中对服务的执行、监控。包括有以下三个功能：1)服务执行管理。SSM 为服务分配好资源后，SEM 从就绪队列中选取一个发送到相应的资源执行。若发送成功，将该服务移到执行队列中，否则重发。执行完成后，将该服务移到完成队列中，从执行队列中将它删除，同时将结果通知 SSM。对每一个资源实体，我们用一个表管理该资源当前负责执行的所有服务。另外，在服务执行期间，我们用不同的线程处理不同的任务或服务实例，因此，Agent 可在任何时候同时执行多个服务。2)信息管理。SEM 通过 CM 在任务、服务以及其它 Agent 间进行信息路由。3)服务监控：查询服务的执行状态、接受

服务提供者对服务的汇报、异常处理。其中，异常处理包括：a)功能性异常。指某个资源正在执行的服务已经进入错误状态，SEM 通知 SSM 要么重新调度相同的服务实例，要么对该服务重新调度(配置相同服务类型的另一个实列)。例如：Task1产生一个错误，Agent 选择另一个自由的和功能上相等的任务，并用调用 Task1的原始信息调用它，b)资源性异常。指完成某服务的资源实例发生错误(如：该资源已经注销)，SEM 通知 SSM 重新调度所有已经分配给该资源完成的服务。例如：Agent 对外提供的服务 S 由 A、B、C 三人完成，若 A 缺席，Agent 试图重新调度那些分配给 A 的服务。c)服务异常：Agent 不能及时完成它所作的服务承诺，SEM 通知 NMM 重新协商 SLA，当重新协商失败时，终止该服务。

3.6 通信管理 CM

该模块负责：1)将发送给其它 Agent 的信息以 Agent 通信语言和共享信息模型打包成消息；2)接受和解释从其它 Agent 发来的消息；3)接受它自己的任务发来的消息。因此，基于消息的 Agent 通信必须共享一种公用消息语法，从而能用一种统一的方式来解释不同的消息类型，消息的语法如下：

```
Message::=(Action)(Sender)(Recipient)(Conversation)(Service)(Content)(Info_model)
        //因为和同一服务的多个实例的多个通信并发存在，CM 必须保证会话互不干扰，因此，Agent 在消息中指明发送者 Sender、接受者 Recipient、会话 Conversation 和服务名 Service，来唯一标识 agent 间的通信。
(Sender)::=(Agent_id)
(Recipient)::=(Agent_id)
(Conversation)::=(conversation_id)
(Service)::=(service_name)
(Info_model)::=(model_id)
(Action)::='cando'|'capable'|'not_capable'|'propose'|'counter_propose'|'accept'|'deny'|'confirm'|'renegotiate'|'request'|'report'|'inform'。
```

通信主要发生在协商和服务执行时信息的传递。在任务管理中，消息在 SEM 和它所管理的任务间路由；在服务管理中，消息在它的 SEM 和服务提供者/消费者间路由；在协商中，消息在其 NM 和被协商的 Agent 间路由。另外，CM 检查进来消息的合法性，确保这些消息在当前背景下的正确性(例如：如果 Agent 收到的消息不是来自于和它协商的 Agent，将产生一个错误消息信息)。

通信过程：Client 提供的信息由 Client 翻译(查映射表的过程)为共享信息模型，发送给 Server，由 Server 翻译为它本地的信息模型，在 Server 完成服务后，输出给 Client 的信息由 Server 翻译为共享信息模型，发送给 Client，由 Client 翻译为它本地的信息模型。若在该过程中，产生通信失败(Agent 间发送的信息不符合语法)，由 CM 检测并负责重发消息。

结束语 首先，本系统通过 SDL 定义服务和 SEM 执行服务，完成了 WFMS 的核心功能：通过自动执行预先定义的任务序列实现业务过程的自动化。其次，DBPM/AB 中，Agent 通过控制和分配企业包括的各种信息系统、数据库、设备和人，从而完成资源管理。传统的 WFMS 系统并不支持如此嵌入资源管理功能；相反，过程在实施前必须确定其规模(如：包括的活动数，执行路径等)，并配备好资源，因此很难有效地在资源动态变化的环境中运行。第三，WFMS 通过预先定义业务过程中可替换的路径来管理异常。而在本系统中，Agent 动态地对环境变化作出反应，根据资源的可获得性和任务的关键性，试图重新协商服务并为它们安排资源。最后，WFMS 由一个集中的工作流引擎来监控系统中所有事件。当业务过程跨一个大型企业时，该体系结构有其明显的局限。本系统采用分布式体系结构，按照组织结构，将企业范围内的业务过程分解到代表各职能部门的 Agent。由 Agent 完成各自较为独立的子过程，即服务；通过 Agent 间相互协调，共同完成业务过程的自动化。这样，各部门业务的变动不会影响 WMFS 执行其他业务过程，系统具有较好的演进性。

总之，传统的分布式计算技术和 WFMS 不仅不能较好支持过程的自动化，而且灵活性很差。而我们研究的基于 Agent 的业务过程管理系统不仅能克服了它们的不足，而且还具备如下的优点：1)更高的灵活性。Agent 决策的基础是当前环境状态，而不是预先的定义。2)更好的伸缩性和扩展性。当 Agent 管理的服务变动时，对其它 Agent 的影响很小。3)更强的适应性。通过业务过程执行的历史信息来指导 Agent 对当前活动的决择。

参考文献

1 Mohan G., Jose S. Recent Trends in Workflow Management Products, Standards and Research. Available at. http://WWW.almaden.ibm.com/u/mohan/
2 Alonso G, et al. Functionality and Limitations of Current Workflow Manangement Systems
3 Jennings N R. et al. Agent-Based Business Process Man Agent. Int Journal of Cooperative Information System, 1996, 5(2~3): 105~130
4 Jennings N R. et al. Using Intelligent Agents to Manage Business Process. 1997
5 Norman T J, et al. Designing and Implementing a Multi-Agent Architecture for Business Process Management. 1998
6 Wooldridge M J, Jennings N R. Intelligent Agents. Theory and Practice. The Knowledge Engineering Review, 1995, 10(2): 115~152
7 Jennings N R, et al. Autonomous Agents for Business Process Management. 1999
8 Cockburn D, Jennings N R ARCHON: A Distributed Artificial Intelligence System For Industrial Applications
9 Brazier M T, et al. DESIRE: Modelling Multi-Agent Systems In a Compositional Formal Framework
10 Vulkan N, Jennings N R. Efficient Mechanisms for the Supply of Services in Multi-Agent Environments. 11/98/