服务 Agent 的设计与实现

傅一峰,曹 健
FU Yi-feng,CAO Jian

上海交通大学 计算机系 协同计算与智能技术研究室,上海 200240
Lab for Collaborative Computing & Intelligence Tech.,Department of Computer Science & Technology,Shanghai JiaoTong University,Shanghai 200240,China
E-mail:fuyifeng1985@gmail.com

FU Yi-feng,CAO Jian.Design and implementation of service Agent.Computer Engineering and Applications,2009,45(9):80-84.

Abstract: In the field of service computing,there are always a number of different services with the same or similar function.In this paper,considering the management and selection of services with the same function,a structure definition of service Agent model is proposed,on the basis of BDI Agent.Service Agent supports the dynamic binding of services.Then the paper introduces the service Agent modeling GUI framework,which is designed and implemented according to the service Agent model.In the framework,a session-based protocol of multi-agent negotiation is implemented.At the end of the paper,an example is introduced.
Key words: service computing;service Agent;Agent negotiation

摘 要:在服务计算领域,相同功能的服务往往有多个不同的选择。针对同类服务的管理和选择问题,在 BDI Agent 的基础上,提出了一种服务 Agent 的结构定义,用以支持服务的运行时动态绑定;设计并实现了基于会话的约束求解服务 Agent 协商协议;实现了支持服务 Agent 建模的可视化工具;最后介绍一个应用实例。
关键词:服务计算;服务 Agent;Agent 协商
DOI:10.3778/j.issn.1002-8331.2009.09.023 **文章编号:**1002-8331(2009)09-0080-05 **文献标识码:**A **中图分类号:**TP393

1 引言

面向服务的体系架构 SOA[1]已经越来越得到广泛的应用。Web 服务技术也越来越成为企业间或企业内部系统间,服务发布和共享的首选方式。Web 服务技术能很方便地实现低耦合的分布式系统集成。但 Web 服务技术是一种无状态的功能响应,它存在功能单一,无法主动响应外界事件、服务之间无法相互自主协作等不足。而 Agent[2]是具有自主决策能力、相互协作能力和一定的智能性的自治实体,它为建立新一代的软件中间件平台提供了技术基础。将 Agent 引入到 Web 服务中,建立服务 Agent 的概念,可以在以下方面提高 Web 服务的性能:

(1)分类管理:服务 Agent 是一个服务综合体。它对功能类似的一组服务进行集中管理,并向外界提供高层服务。

(2)优化选择:由于同类 Web 服务的对外接口都集中到了单个服务 Agent 处,而且服务 Agent 掌握了这组 Web 服务的所有信息,因此它可以根据调用者的偏好方便地实现 Web 服务的优化选择。

(3)协作性:服务 Agent 可以通过 Agent 通讯语言(ACL[3])进行相互协作,相互调用,进而实现 Web 服务集成。

(4)智能性:服务 Agent 动态地管理一组服务,维护 Web 服务的 QoS 等信息并依据需要更新管理的服务。

目前对 Web 服务和 Agent 技术的结合研究主要集中在基于 Agent 的 Web 服务组合的研究上,即依靠 Agent 的智能性和能动性来实现 Web 服务的组合。基于 Agent 的 Web 服务动态组合[4]:每个 Agent 封装一个 Service,依靠 Agent 之间的通信以及每个 Service 之间的输入输出接口信息来动态构建一个 Agent 前后连接的流程图,进而实现了 Service 的动态组合。此外还有一些文章利用工作流和 Agent[5-6],来实现 Service 的组合。这些方法在考虑 Web 服务组合的同时都没有考虑同类 Service 的管理和选择问题。AWSS[7]:利用 Agent 技术和 Semantic Web Service 实现了按照用户需求的同类功能 Web 服务的智能搜索,并推荐搜索结果。但是 Service Agent 之间不支持协商,使得 Service 之间不能相互制约,进而使得有些公共的约束问题没法求解。本文则着重研究利用 Agent 的智能性来管理多个同类 Web 服务这一层面,并且实现平台支持约束问题的多 Agent 协商求解。事实上,随着 Web 服务的进一步普及,相同功能的 Web 服务的管理和选择也是服务 Agent 需要考虑的一个重要问题。

基金项目:国家自然科学基金(the National Natural Science Foundation of China under Grant No.60503041);国家高技术研究发展计划(863)(the National High-Tech Research and Development Plan of China under Grant No.2006AA04Z152,No.2007AA01Z137)。
作者简介:傅一峰(1985-),硕士研究生,主要研究方向:服务计算,智能技术;曹健(1972-),副教授,CCF 会员,主要研究方向:服务计算,协同信息系统和软件工程等。
收稿日期:2008-01-28 **修回日期:**2008-04-15

对于服务 Agent 相关的研究领域总体上还缺少一种可视化工具来支持服务 Agent 的创建。现有的一些创建 Agent 的工具中,TI Lab 的 JADE[8]只是提供了一套支持 JADE Agent 运行的中间件;商业软件 AgentBuilder[9]虽然提供了一套 BDI Agent 的可视化建模平台,但是通用的 BDI Agent 建模工具很难满足服务 Agent 特有的功能需求。

在 BDI Agent 模型[10]的理论基础上,研究并提出了服务 Agent 的模型结构,并实现了一套服务 Agent 的可视化建模工具用来支持此类 Agent 的模型创建。最后借助 JADE Agent 技术设计并实现了服务 Agent 的执行环境。

2　服务 Agent 平台体系结构

服务 Agent 平台是开发,分布式部署和运行服务 Agent 的一整套解决方案。其整体框架结构由图 1 所示。主要由两部分组成:其一是服务 Agent 的建模工具;其二是 Agent Container。

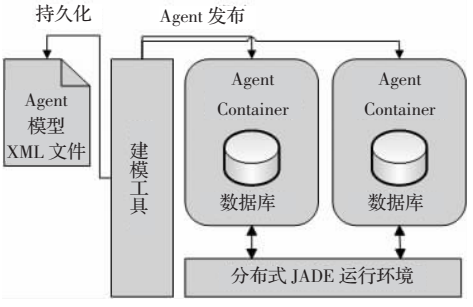


图 1　服务 Agent 平台框架总体设计

建模工具遵循 Agent 的 XML Schema,通过可视化界面可以对服务 Agent 进行完整的定义。服务 Agent 模型由建模工具产生的,并可被序列化成 XML 文件,方便移动和二次开发。定义完成的 Agent 模型需要被发布到指定的 Agent Container 才能运行。Agent Container 是一个供服务 Agent 运行的分布式环境。它建立在 JADE Agent 的环境之上,并且提供 Web 服务调用接口用以接收 Agent 的发布。

3　服务 Agent 结构定义

服务 Agent 是管理一系列 Web 服务的综合服务体,并且需要依赖 JADE 技术进行通信。在现有的 BDI Agent 概念(Belief-Desire-Intention)的基础上扩展出服务 Agent,并且给出了图 2 中的结构定义。

定义 1(服务 Agent)一个基于 BDI Agent,管理同类服务的六元组模型(World Model,Neighbor Model,Constraint Model,Service Model,Extension Behavior,Plan)。

服务 Agent 的世界模型,邻居模型,约束模型和服务模型一起组成了服务 Agent 的 Belief,是服务 Agent 对外部世界和其掌握的资源的一个统一描述。Belief 分为 GlobalBelief 和 SessionBelief,两者结构完全相同,但运行时模型的作用域不同。之所以引入 SessionBelief,是为了让服务 Agent 支持协商等事务处理。SessionBelief 下的模型生命周期都为一次协商,协商结束,模型被删除,并且 SessionBelief 下的模型在不同的协商下有不同的副本,互不干涉;而 GlobalBelief 下的模型生命周期与该 Agent 的生命周期相同,并且模型只有一个副本。BDI Agent 是一种具有极强能动性和协作能力的智能体抽象模型,因此服务 Agent 之间需要通信,存在消息的传入和传出,因而需要建立一些响应输入消息的活动部件,即 Plan 模型。Plan 响应一次消息传入,并对该类型的消息进行特定的处理。这个处理过程由一系列的行为(Behavior)组成。服务 Agent 平台将行为分为两类:系统行为和扩展行为。系统行为是由系统提供一些通用的活动,它们作为基础构架而存在在平台中,可能被所有的服务 Agent 复用,比如 Web 服务调用行为 ServiceInvoker。扩展行为用以扩展该平台的功能,由建模者自己编写一些扩展组件,嵌入到系统中。

Belief 存储着服务 Agent 对世界与自身的认识。其中,世界模型用于描述服务 Agent 心智模型中某些状态的取值,比如服务 Agent 之间协商中产生的一些中间结果。它是一个变量的集合。服务 Agent 平台提供了四种常见的变量类型(string,int,bool,float),以及一种用来描述复杂数据类型的 XML 对象类型。

服务 Agent 之间可以相互认知,在必要的时候可以启动协商一起做出某些决定。邻居模型就是用来描述被该服务 Agent 认知的其他服务 Agent 的模型。服务 Agent 被统一命名为"agentname@container"的形式。

服务 Agent 用于管理一类相似功能的 Web 服务,因此特定的服务 Agent 与某特定领域密切相关。约束模型便是用来描述该领域中的一些常识性约束,比如服务 Agent 协商所要满足

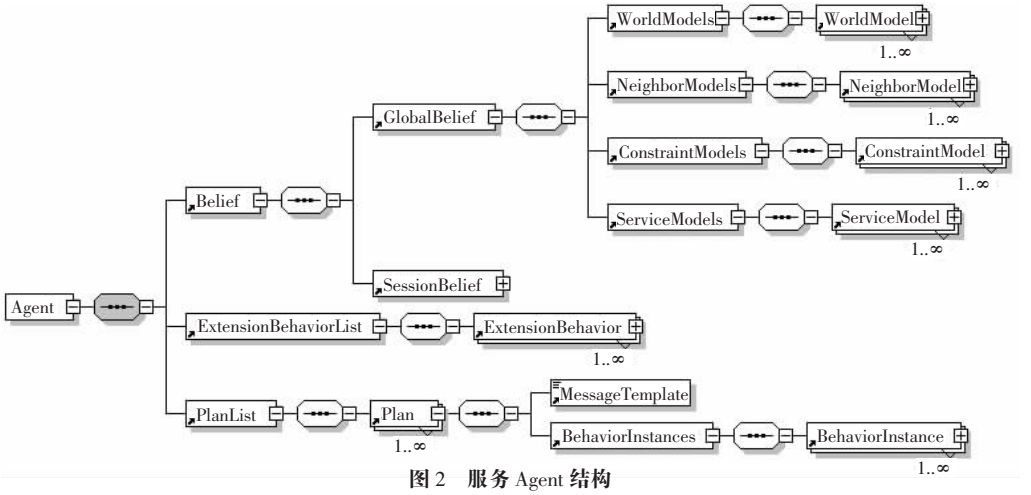


图 2　服务 Agent 结构

的约束条件。

3.1 服务模型定义

服务模型用于描述该服务 Agent 所管理的一组 Web 服务的信息。其结构定义如图 3。

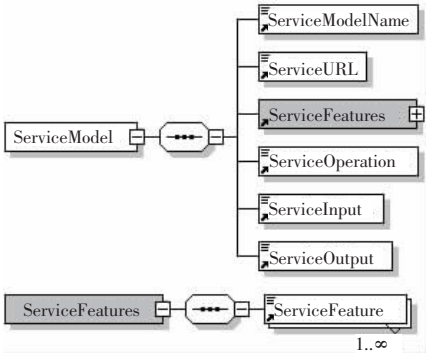


图 3 服务模型结构

定义 2(服务模型) 一个描述一个 Web 服务调用接口的调用信息和服务特性的六元组(Name,Service URL,Service Operation,Service Input,Service Output,Service Features)。

其中 Service Input 与 Service Output 都是一个 XML Schema，分别用于描述调用 Web 服务的输入参数和输出参数格式。Service Feature 是该 Web 服务特性的描述符。用来方便对如 Web 服务的 QoS,Web 服务调用优先等级等服务特性的描述。

3.2 服务的注册和调用机制

服务 Agent 平台在建模时提供了服务注册的机制。建模者在添加服务模型的时候，服务 Agent 平台会按输入的 WSDL URL 自动解析该 Web 服务。并按照所选的 Operation,将 Web 服务以服务模型的形式注册到服务 Agent 中。为方便服务的调用，服务 Agent 平台提供了内置的 ServiceInvoker 系统行为。ServiceInvoker 以一个行为实例的形式添加到 Plan 模型中,当该 Plan 被动态调度到的时候,ServiceInvoker 实例就会调用绑定的那个服务。服务 Agent 依靠消息驱动的 Plan 的动态调度,来实现服务的选择和动态绑定。

3.3 扩展行为与规划 Plan

扩展行为用来描述用户编写的 Java Class 静态方法的调用接口,结构见图 4。其中 InParameterTypeList 元素定义了这个调用接口的调用参数类型。扩展行为的设计目的就是为了方便建模者将一些可复用的扩展接口集成到服务 Agent 中,增强服务 Agent 的能力。

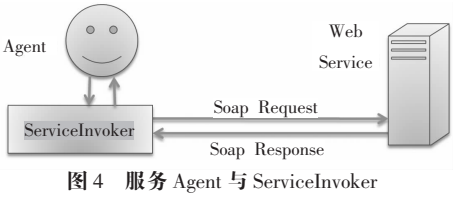


图 4 服务 Agent 与 ServiceInvoker

因为 Agent 的社会性(即相互通信的需求),所以 Plan 是基于消息的。一类消息对应着一个 Plan,一个 Agent 可以有多个 Plan 对应于多类消息。服务 Agent 平台将 Plan 分为两类:init Plan 和普通 Plan。一个普通 Plan 只有在服务 Agent 收到对应类别的消息的时候,才被执行。并且只要再次有同类消息进入,

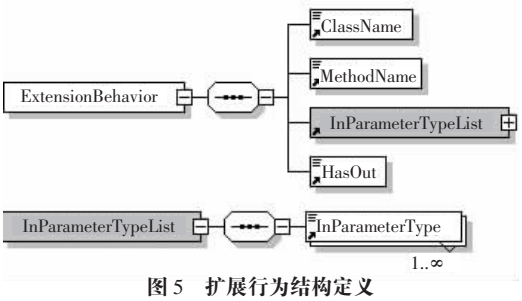


图 5 扩展行为结构定义

该 Plan 就会被再次执行。在实际应用中，发现有必要在服务 Agent 启动阶段对其进行一些初始化工作。因此，设计了 init Plan。该 Plan 用于服务 Agent 在 Container 中启动时候的初始化操作，因此 init Plan 是唯一一个非消息驱动的 Plan。Init Plan 在服务 Agent 启动的时候被自动调用,并且有且仅有一次被调用。

4 服务 Agent 运行机制

4.1 服务 Agent 的生命周期

服务启动之后立即开始接受消息,消息被暂存在服务 Agent 的个人信箱中等待被处理。当初始化命令到达之后,服务 Agent 加载并执行 init Plan。在 init Plan 执行结束之后，服务 Agent 会自动加载其余所有的普通 Plan。所有处于 Active 状态的 Plan 都会定时去个人信箱筛选未处理的消息。当发现与自己匹配的消息后，启动并执行一次 Plan 流程处理该消息。最后,当服务 Agent 接受到 kill 消息之后,终止服务,生命周期结束。

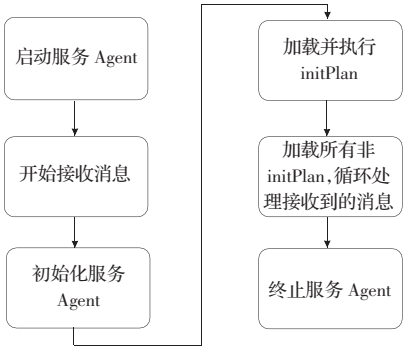


图 6 服务 Agent 生命周期

4.2 服务 Agent 的消息机制

服务 Agent 依靠 JADE 平台实现 ACL 消息传递。消息格式符合 FIPA[11]规范。服务 Agent 每次 Plan 的执行都有可能需要发送出一系列的消息。在 Plan 执行的结束处,服务 Agent 平台负责将这些产生的消息发送出去。依靠 JADE 平台可以很好地实现消息的路由。使用统一的规范,为异构平台间 Agent 的通信奠定了基础。

4.3 协商协议和 SessionBelief

服务 Agent 可能碰到一些公共约束问题求解的问题。当服务 Agent 的某个约束模型不能靠自己的知识求解时,就需要启动和该约束相关 Agent 之间的协商。

服务 Agent 平台为支持协商提供了三方面的特性:

(1)SessionBelief

一次协商启动之后,服务 Agent 运行时平台就会对所有的

SessionBelief 克隆出一个副本,作为本次协商的“局部模型”,以区分不同协商下的这些模型副本的取值。

(2)协商协议和保留的消息模板类型

协商一般由一个 Controller Agent 发起。Controller Agent 会发送 initSession 消息给所有被期望参与协商的 Agent,包括自己在内。其他 Agent 接收到 initSession 消息之后,就执行一次服务 Agent 平台内置的 initSession Plan。该 Plan 中具体做 SessionBelief 克隆等初始化工作。完成 Plan 执行后,发回 initSessionResponse 消息。Controller Agent 接收到所有返回的 initSessionResponse 消息后。发送 startSession 消息,协商开始。协商结束时,也由 Controller Agent 发送 endSession 消息,通知所有的参与者结束本次协商。其他 Agent 接收到 endSession 消息后就会销毁本次协商中的 SessionBelief 副本。

(3)内置 Plan

服务 Agent 平台对于 startSession 消息和 endSession 消息提供了内置的对应处理 Plan。在 startSession 消息中,捎带了一个全局唯一的 Session ID 用于标示此次 Session。startSession Plan 会依据这个 ID 创建所有 Session Belief 模型的副本,作为本次(协商)会话中的“局部变量”。endSession Plan 则相反,在收到 endSession 消息之后,就将本次会话的所有 Session Belief 模型的副本销毁。

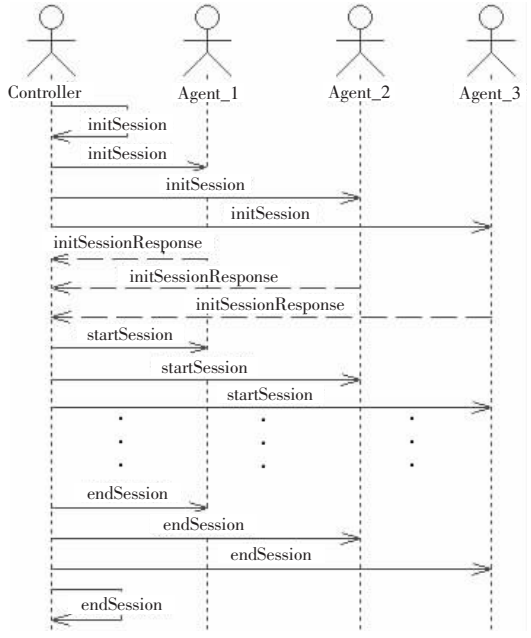


图 7　协商流程

5　案例分析

下面用一个例子阐述服务 Agent 利用该平台进行服务协商的过程。

5.1　问题描述

有个人想要用身边的 20 000 元积蓄购买保险。眼下有意外伤害保险、健康保险、人寿保险、分红保险 4 个险种可供购买。这里需要 4 个服务 Agent,分别用 accident,health,endowment,bonus 4 个服务 Agent 来负责购买意外伤害保险、健康保险、人寿保险、分红保险。4 个服务 Agent 都有各自相关的多个 Web 服务可供选择(如表 1)。该购买者希望每年在保险上的支出不超过 3 500 元;60 岁后每年获利不小于 25 000 元;遇到重大伤害后总赔付不小于 35 000 元。在加上一次性付费不能超过 20 000 元,一共 4 个约束需要协商。

表 1　服务 Agent 的注册服务　　元

		一次性付费	遇交通事故赔付
accident	serviceAccident1	10 000	50 000
	serviceAccident2	5 000	20 000
	serviceAccident3	2 000	10 000
		每年付	遇重大住院赔付
health	serviceHealth1	500	5 000
	serviceHealth2	1 000	6 000
	serviceHealth3	2 000	8 000
		每年付	60 岁后每年得到
endowment	serviceEndowment1	1 000	10 000
	serviceEndowment2	2 000	15 000
	serviceEndowment3	3 000	20 000
		一次性付费	50 岁后每年得到
bonus	serviceBonus1	10 000	5 000
	serviceBonus2	20 000	7 000
	serviceBonus3	40 000	10 000

5.2　建立模型[1]

(1)accident 服务 Agent

①World Model={choice(cost,interest)}[2]

②Neighbor Model={health,endowment,bonus}

③Constraint Model={accident.choice.cost+bonus.choice.cost<=20000;accident.choice.interest+health.choice.interest>=35000}[3]

④Service Model={serviceAccident1,serviceAccident2,serviceAccident3}

⑤Extension Behaviors 涉及到分布式约束满足算法 DCSP[12] 的一些相关功能调用。

⑥Plans 包含了 init Plan 和一些基于消息的 Plan 的建模。本例中涉及到 DCSP 的一些消息协议的实现。

(2)health 服务 Agent

①World Model={choice(yearlyCost,interest)}

②Neighbor Model={accident,endowment,bonus}

③Constraint Model={health.choice.yearlyCost+endowment.choice.yearlyCost<=3500;accident.choice.interest+health.choice.interest>=35000}

④Service Model={serviceHealth1,serviceHealth2,serviceHealth3}

(3)endowment 服务 Agent

①World Model={choice(yearlyCost,yearlyInterest)}

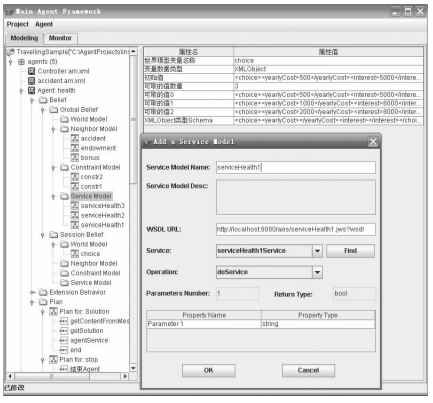
②Neighbor Model={accident,health,bonus}

③Constraint Model={health.choice.yearlyCost+endowment.choice.yearlyCost<=3500;endowment.choice.yearlyInterest+bonus.

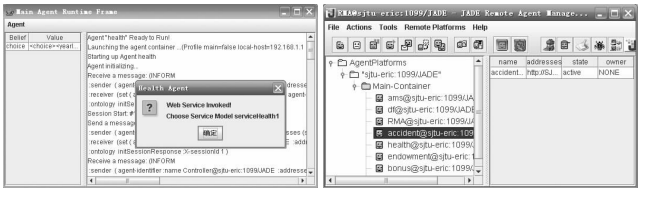
1 本例中,World Model 都为 SessionBelief,其他 Model 都为 GlobalBelief

2 choice(cost,interest):表示 XML 对象类型<choice><cost><interest></interest></cost></choice>

3 accident.choice.cost:表示 accident 服务 Agent 的当前选择的一次性支付费用,其中 choice 为 World Model



(a)建模结果和服务模型注册



(b)Health 服务 Agent 运行结果 (c)JADE 运行结果

图 8 案例建模与运行结果

choice.yearlyInterest>=25000}

④Service Model={serviceEndowment1,serviceEndowment2,serviceEndowment3}

(4)bonus 服务 Agent

①World Model={choice(cost,yearlyInterest)}

②Neighbor Model={accident,health,endowment}

③Constraint Model={accident.choice.cost+bonus.choice.cost<=20000;endowment.choice.yearlyInterest+bonus.choice.yearlyInterest>=25000}

④Service Model={serviceBonus1,serviceBonus2,serviceBonus3}

5.3 运行结果

最后运行结果如下:

accident:{cost=10000;interest=50000}

health:{yearlyCost=500;interest=5000}

endowment:{yearlyCost=3000;yearlyInterest=20000}

bonus:{cost=10000;yearlyInterest=5000}

即,accident 选择了 Web 服务 serviceAccident1;health 选择了 serviceHealth1;endowment 选择 serviceEndowment3;bonus 选择 serviceBonus1。

6 总结及展望

首先提出了服务 Agent 平台体系结构,在此基础上给出了一整套的服务 Agent 结构定义标准,提出了服务的注册和调用机制、消息驱动的 Plan 模型和基于会话的服务 Agent 协商协议,最后实现了支持服务 Agent 建模的可视化工具。服务 Agent 平台能表达实际业务中的多种情况,并在很大程度上方便了服务 Agent 的模型创建。

下一步将继续在以下方面完善本模型和平台:

(1)现阶段的模型中,消息和 Plan 的匹配是一一对应的硬绑定,并没有考虑到当前的 belief 状态。下一步将考虑引入规则引擎,实现消息与 Plan 之间的松耦合关系。借助规则引擎,Agent 在接收到消息之后可以按照当前的 belief 状态来判断启动某条 Plan 或者选择忽略该消息。

(2)现阶段模型中,Service 的注册还是以 Service Model 的形式由建模者手动注册到 Agent 中。下一步将考虑在 Agent 的执行过程中,动态地发现匹配的 Service,实现 Service Model 的动态创建和动态剔除。

参考文献:

[1] Jerstad I,Dustdar S,Thanh D V.A service oriented architecture framework for collaborative services[C]//Proceedings of the 14th IEEE International Workshops on Enabling Technologies:Infrastructure for Collaborative Enterprise(WETICE'05),2005:121-125.

[2] Cabri G,Ferrari L,Leonardi L.Agent role-based collaboration and coordination:a survey about existing approaches [C]//IEEE International Conference on Systems,Man and Cybernetics,2004,6:5473-5478.

[3] ACL[EB/OL].http://www.fipa.org/repository/aclspecs.html.

[4] Liu S,Küngas P,Matskin M.Agent-based Web service composition with JADE and JXTA[C]//SWWS'06,2006:110-116.

[5] Buhler P,Vidal J M.Enacting BPEL4WS specified workflows with multiagent systems[C]//Proceedings of the Workshop on Web Services and Agent-Based Engineering,2004.

[6] Sun Zhi-Zhong,Li Bin,Li Liang.An adaptive agent coordination framework for Web services composition[C]//2007 International Conference on Machine Learning and Cybernetics,2007,7:3870-3875.

[7] Zhang Tiantian,Luo Junzhou,Kong Weining.Carrying on automatic service recommendation by Agents[M]//Lecture Notes in Computer Science,2006,3865:254-263.

[8] JADE[EB/OL].http://jade.tilab.com/.

[9] AgentBuilder[EB/OL].[2008-04].http://www.agentbuilder.com/.

[10] 曹健,李明禄,张申生.基于多 Agent 协商的服务流程定制[J].计算机学报,2006,29(7).

[11] FIPA[EB/OL].http://www.fipa.org/.

[12] Yokoo M.Weak-commitment search for solving constraint satisfaction problems[C]//Proceedings of the Twelfth National Conference on Artificial Intelligence,1994:313-318.