Agent 技术在 Web 服务中的应用探讨*)

许德正[1,2]　白晓颖[1]　戴桂兰[2]

(清华大学计算机科学与技术系　北京 100084)[1]　(清华大学信息研究院　北京 100084)[2]

摘　要　Web服务是一组采用面向服务的体系架构的基于标准的Web协议的软件构件，而Agent是一种在某个环境中自主行动以实现其设计目标的智能化软件实体。Web服务与多Agent系统(MAS,Multi-Agent System)在架构、范例和技术方法等方面具有很大的相似性。Web服务的可信性是未来计算机软件发展关键的问题，实现Web服务在分布式环境下的协同式测试具有一定的挑战性。本文在探讨Web服务与Agent技术的共性的基础上，分析并总结了Agent技术在Web服务中的应用。针对Web服务测试的开放性、协同性、动态性和不确定性等特点，结合Agent系统的自主性、反应性、适应性和社会性，本文提出了一种基于MAS的Web服务测试框架(MAST,Multi-Agent-based Service Testing)，并对其关键技术进行了探讨。

关键词　Web服务，多Agent系统，Agent，Web服务测试

Applying Agent to Web Services: a Survey

XU De-Zheng[1,2]　BAI Xiao-Ying[1]　DAI Gui-Lan[2]

(Department of Computer Science and Technology, Tsinghua University, Beijing 100084)[1]

(Research Institute of Information Technology, Tsinghua University, Beijing 100084)[2]

Abstract　A Web Service is a self-contained software component that reflects the Service-Oriented Architectural (SOA) approach, enabling cross-platform integration by standard Web-based protocols. While an Agent is an intelligent software entity that is capable of independent action providing controlled access to shared services and resources. Web Services share many similarities to Multi-Agent System (MAS) in architecture, paradigm and technology. The trustworthiness is a key issue in the tremendous growth of Web Services. And collaborative distributed test for Web Services becomes a great challenge. This paper is going to discuss the commonness of Web Services and Agent, and briefly summarize how Agent can be utilized to add value to Web Services, which promotes cross-fertilization of techniques between these two areas. Due to the open, dynamic and collaborative nature of Web Service test, regarding the autonomy, reactivity, pro-activity and social ability of Agent, this paper proposes a Multi-Agent based Service Testing (MAST) framework, and discuss its key technology.

Keywords　Web service, Multi-agent system, Agent, Web service testing

1　引言

Web服务是未来计算机软件的一个发展趋势，而Agent技术在Web服务中的应用是目前的研究热点之一。根据W3C的定义，Web服务是一种软件系统，其设计目的是支持机器之间基于网络的互操作，有效地解决在分布、动态、异构环境下数据、应用和系统集成等问题。针对Web服务定义了一系列的标准和规范，其中SOAP协议(Simple Object Access Protocol)实现消息传递，WSDL(Web Services Description Language)在语法上定义了服务的基本输入和输出消息以及其他调用服务相关的细节，UDDI(Universal Description, Discovery and Integration)在Web服务的发布和发现上提供了一种基于分布式的商业注册中心机制。此外，BPEL4WS(Business Process Execution Language for Web Services)用于描述服务组合和工作流，OWL-S(Ontology Web Language for Services)提供对服务在语义层次上的描述，实现基于服务领域、数据语义和功能语义等的相似性匹配，从而实现动态的服务发现、选择和组合自动化。

Agent与Web服务有着很多相似性。Agent是一种具有智能的实体，其应用主要是以一种松散耦合的Agent网络即MAS出现。多个Agent之间结构化的ACL(Agent Communication Language)进行交互、协作，完成某一任务；FIPA(Foundation for Intelligent Physical Agents) Agent在DF(Directory Facilitator)注册和发现服务。这些都与Web服务相类似。此外，Agent还具有自主性、反应性、适应性和社会性等Web服务所不具备的智能化特征。为了能让服务在分布式环境下更好地理解用户的需求并构建更适当的行为，需要利用Agent的智能化特征对其进行功能扩展。如果请求的是一种希望达到的功能或效果，调用服务的过程需要依赖于调用时的上下文(Context)，Agent能感知上下文，并且担当用户目标和服务具体行为之间的中介角色。例如“我要在圣诞节之前赶回家”这一请求，表明用户强调的是时间因素而非交通工具和价格，Agent可以帮助理解这一意图，并且考虑不同的交通工具所费时间、票务的紧张程度和路面以及天气状

*)国家自然科学基金项目(60403022)资助。**许德正**　硕士研究生，研究方向为Web服务测试；**白晓颖**　副教授，研究方向为软件工程和软件测试；**戴桂兰**　副研究员，研究方向为软件工程和软件测试。

况等，做出灵活的服务响应。

本文在分析 Agent 与 Web 服务的共性的基础上，总结了 Agent 技术在 Web 服务中的应用。基于 Agent 自主性、适应性等智能化特性，结合 Web 服务测试的特点，我们提出了一个基于 MAS 的 Web 服务测试框架 MAST，以支持分布式环境下服务的协同测试，并对其关键技术进行了探讨。

2 Agent 技术在 Web 服务中的应用

Web 服务由于其定义良好的标准、基础设施(Infrastructure)和互操作性而成为未来计算机软件发展的一个趋势，Agent 技术则因其智能化特点和社会化的功能(Social Capability)而优于传统应用。这两种技术的整合可以构建一个智能化分布式应用环境。语义 Web 服务的出现使 Agent 能够有机会把它的智能化特点和社会功能应用于诸如服务的组装编制(Orchestration)和流程编排(Choreography)等复杂行为上。

FIPA 规范和 Web 服务标准有很多相似的地方，其简单对应关系如表 1 所示。然而，虽然 W3C 和 FIPA 都在其标准和规范中定义了注册、发现和通信等核心功能，它们的机制是相似的，但所使用的技术实现手段却截然不同，从而使 FIPA 的规范和 Web 服务的标准与技术缺乏互操作性。

表 1 FIPA 规范和 Web 服务标准的对应关系

	FIPA	Web 服务和语义 Web
服务发现	Agent Description Ontology	WSDL,OWL-S,WSMO [17]
注册	DF	UDDI
通信协议	ACL	SOAP
语义描述	FIPA Semantic Language	OWL,WSMO
交互	FIPA Agent Interaction Protocols	BPEL4W,WS-CDL, OWL-S Process Model

提高 Agent 和 Web 服务的互操作性，填平 FIPA 规范和 Web 服务标准之间通信的鸿沟，在此基础之上通过 Agent 扩展 Web 服务，可以让 Agent 和 Web 服务之间可以进行交互，帮助 Web 服务实现高层次的应用。同时，以语义 Web 技术为桥梁，Web 服务引入通信语义等高层次概念，利用 Agent 智能性特点，让服务能更好地满足用户需求。目前，关于 Agent 技术在 Web 服务中应用的研究大致可总结为以下四大类。

2.1 基于 Agent 的服务扩展

Agent 是具有特定功能的软件实体，Web 服务可以调用这些实体来扩展服务自身，其中的研究着眼于实现二者的无缝连接和互操作自动化，比较有代表性的观点有以下两点：

第一，实现符合 FIPA 规范的 Web 服务 Agent，即将 Agent 的功能包装成 Web 服务，供非 Agent 客户端以访问 Web 服务的形式使用。二者把它们的服务同时注册在 UDDI 和 DF[6]。

第二，采用网关(Gateway)整合的方式实现 Agent 和 Web 服务的相互访问。文[1,3,4]分别提出了不同的网关模型。各种模型的特点可以总结如下：(1)网关是双向的；(2)Web 服务调用 Agent 关键在于需要一个适配器在 SOAP 和 ACL 之间进行翻译；(3)网关有自己的 DF 和 UDDI，为每个注册的 Web 服务和 Agent 功能的描述提供在另一方注册中心的映射；(4)使用 OWL-S 作为共同的服务描述语言，实现语义层次的互操作性，允许调用简单的原子操作或者二者混合的服务组合，构建复杂的过程模型。网关模型的关键问题在于减少人工干预和配置实现自动化，以及语义层次上的 Agent 和 Web 服务的发现、选择和组合等方面的扩展性。

总的来说，Web 服务调用 Agent 不同于传统的调用，后者假设了被调用方必须有预定义的行为并且外界清楚了解这些行为，比如确切的操作名称和参数。而调用 Agent 的 Web 服务需要有一种类 Agent 的行为机制来匹配 Agent 具有自主性和自治性等特点的响应。因此，Web 服务在语义层次上和 Agent 进行交互，并通过 Agent 来扩展功能，这就需要一种消息翻译机制，实现二者无缝互操作。整合网关是常用的方法，它对服务调用者来说是透明的。

2.2 智能化 Web 服务

智能化的 Web 服务就是让 Web 服务具有某些智能化特点，能够理解用户的目标和意图。目前的研究主要将 Web 服务作为组件行为，自治性和意图则在 Agent 层次上体现[5]。单纯根据 OWL-S 服务描述做出选择，难以理解用户的真正目标。同时，OWL-S 无法表述依赖于上下文(Context)的知识。从高层目标到具体服务调用的映射，需要服务本身的描述和用户的上下文相结合。因此，使用 BDI(Belief,Desire,Intention) Agent 作为用户目标和 Web 服务的操作性语义的中介是个合适的选择。

Agent 和非 Agent 组件之间的重要区别在于 Agent 可以表现出自主性、自治性和反应性等特点，并理解用户或其他 Agent 的目标，具有信念、欲望和意图等属性。比如“我希望在某个时刻在北京和某人会面”这一意图并非简单的服务调用可以实现。要把这些属性直接加进 Web 服务里面，实现智能化的 Web 服务，基于传统的相关标准和规约是难以实现的，但 Web 服务可以借助 Agent 表现出这些属性。Web 服务只提供固定的操作和接口，而把诸如与用户进行交互等一些复杂问题交给 Agent 去处理。

2.3 基于 Agent 的服务发现、选择和组合

用户查找并发现自己需要的服务，选择出其中合适的服务或服务组合与之绑定。为了实现复杂的业务逻辑功能，需要解决服务的集成和协作的问题，Agent 可用作服务发现、选择和组合的执行者和协调者。

服务的发现和选择可以基于以下几种机制：(1)Brokering 机制。Broker Agent 负责解析 Web 服务提供者的能力和请求者的需求，帮助请求者找到最合适的提供者，Broker 始终作为中介和桥梁[9]。(2)基于反馈信息的服务选择机制。把 Agent 作为 Web 服务的 Proxy 帮助用户根据所要求的服务质量及标准选择服务。Proxy Agent 在每一次服务选择后收集反馈信息，以供用户或其他 Proxy Agent 作为服务选择的依据[8]。(3)把服务选择问题转化成一个 Group Decision Problems(GDP)。预定义的本体对于服务的发现和选择并不足够。基于模糊逻辑和语义网的方法通过多 Agent 间达成共识，来更好地实现服务的发现和选择。其中基于投票机制的服务选择被转化成一个 GDP[11]。

Agent 工厂[7]利用任务模型实现了服务组合的自动化，并基于语义服务描述对需求和设计进行推理。其中，任务模型包括了组件、协作模式和本体等配置项。Web 服务作为 Agent 的组件也看作是配置项，选择适合的 Web 服务插入到模板的 Slot 中。服务组合同时应该具有上下文感知(Context-aware)能力，把服务的能力、网络环境及计算资源等上下文 Context 考虑进来[10]，以达到最优化。Agent 担当 Web 服务组合执行者是需要感知当前上下文的。

总的来说，服务发现和选择机制应满足：(1)一种语言让服务提供者和请求者分别描述服务的能力和对服务的需求，支持服务的自动发现和调用；(2)一种有效的匹配技术在模糊匹配的情况下能在语义上理解请求者的需求和提供者的描述。通过 Middle agent 实现的 Matchmaking 和 Brokering 是 MAS 中的协调机制在 Web 服务发现和选择中的应用。动态的服务组合自动化不能仅仅基于传统工作流技术，利用智能化 Agent 把服务组件整合成组合服务、触发服务组件的执行和监测服务组合的情况。

2.4 会话模型与服务交互

基于 Web 服务的高层次的应用要求松耦合、点对点、动态和主动的交互，并不仅仅是单纯的调用者和被调用者这样的角色，而是需要一种会话模型的交互模式。一种可选的方法是把 Web 服务所提供的具体操作泛化为 Speech Act 的操作。比如一个 Web 服务拥有一个叫做 Inform 的操作，其参数相当于在 FIPA ACL 消息里的 Content。这样就使 Web 服务之间或者 Web 服务和 Agent 之间的交互更像是 Agent 之间的会话(Conversations)[2]。

Web 服务的描述也需要包含会话模型。通信协议的描述可以嵌于 DAML-S 的 Process Model 中[12]，Agent 对该通信协议进行推理从而得出 Web 服务的交互、会话功能及其行为序列，以达到 Web 服务的个性化。调用和被调用是双向的。把 Agent 的会话模型用于上述的情景，能帮助 Web 服务实现高层次的应用，尤其是对基于 B2B 的整合。

3 Web 服务测试

Web 服务测试包括了 Web 服务基础设施的验证与确认、独立的 Web 服务测试和集成的 Web 服务测试。分布性特征使系统变得更加复杂，这使 Web 服务测试不同于传统的测试。Web 服务测试需要能够适应面向服务的新的分布式计算体系架构。为保证服务的质量，Web 服务需要从多个层次进行验证与确认，包括基础设施、单元服务、集成服务等；测试需涵盖服务的功能、性能、可靠性、安全等各个方面，其体系结构和应用的复杂性，以及技术和规范不断的发展变化，对测试研究提出了新的挑战[13~16]。

开放性：在与面向服务框架相符合的情况下，任何人在任何时间、地点都可以注册、查找和访问服务节点。服务的开发、发布、使用各环节相对独立；服务管理相对分散。目前尚未建立有效的服务监控机制。

协同性：一方面，服务的开发者、维护者和使用者相互独立，服务的发布、发现和调用的过程相互分离；另一方面，各服务单元分布自制，服务之间动态绑定，通过标准协议实现匹配和交互，协同工作。由于服务的分布特征会出现大量用户通过不同的环境访问一个服务的情形，服务的开发环境与其应用环境有很大的不同。这些差异和应用的不确定性都增加了 Web 服务测试的困难。因此，需在协同开发和运行环境下保证整个业务流程的正确性、完整性和一致性。

动态性：Web 服务及服务集成的发布、发现和绑定都是动态完成的。动态性使得系统构建更加实时、高效，可充分利用各种网络资源，但也带来系统不稳定性及行为难以预测和控制等问题。

不确定性：Web 服务的开发环境与其应用环境有很大的不同。在发布之前，很难对其实际的运行场景进行预测，如访问的用户类型、并发用户数量、Web 服务调用的装载模式和访问方式等，这些差异和应用的不确定性都增加了 Web 服务测试的困难。服务的发现、选择和组合过程的不确定性也增加了测试难度。

4 基于 MAS 的 Web 服务测试框架

Web 服务的可信性是未来计算机软件发展关键的问题，将 Agent 技术用于 Web 服务测试，尤其是动态的 Web 服务组合以及服务的正确性和完备性验证，可以有效解决 Web 服务测试的开放性、协同性、动态性和不确定性等问题。

尽管已经被引入 Web 服务测试自动化领域，但 Agent 更多地仅仅是在分布式环境下充当服务的代理(Proxy)这样一个角色，并未充分发挥其智能化特点。结合 Web 服务的各种标准和规约，利用 Agent 自主性、反应性、适应性和社会性等特点，我们提出一种基于 MAS 的 Web 服务测试框架(MAST)[18]，以实现分布式环境下 Web 服务协同测试自动化。该框架的主要功能包括：(1)将测试流程分解成不同的子任务，包括基于 Web 服务规约的测试用例生成、集中式的测试计划及分布式的测试执行、测试流程的监控和测试结果的综合与分析。不同的 Agent 负责完成不同的子任务；(2)负责将分布式测试执行和测试流程监控的 Agent 组织成不同的分组，一个分组负责一个测试计划的实施；(3)测试用例的生成、测试的组织、协调、调度和监控都是自动化的、动态的。同时，针对 Web 服务的重配置和重组合，提供动态的分布式测试机制；(4)引入基于规则的机制，通过规则引擎和推理，实现测试计划和 Agent 之间的协作；(5)提供同步与异步的信息交互环境，支持多协同用户在分布式协同环境下进行交互式自动测试执行和结果分析。MAST 的基本架构如图 1 所示。

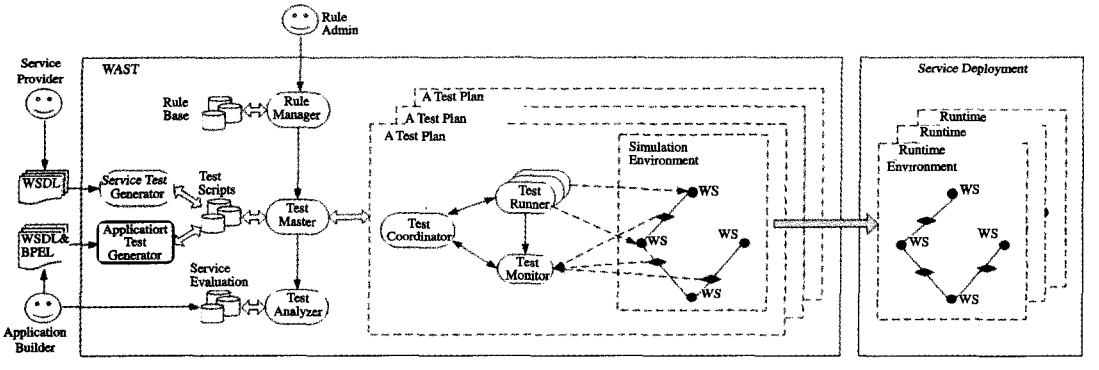

图 1 MAST 基本架构

根据测试流程的子任务分解，该框架定义了以下几个基本的 Agent 类：TestGenerator 负责基于服务规约语义分析的测试用例的生成；TestMaster 根据测试用例制定测试计划和定义 Agent 分组；TestCoordinator 负责计划内各 Agent 间的行为调度并实施测试计划，把执行测试的单元任务分配给 TestRunner，测试结果由 TestAnalyzer 收集和分析。TestMonitor 监控测试执行过程和测试对象服务的状态。基于规则引擎的 RuleManager 支持实时的规则定义和更新。

基于 MAS 的 Web 服务测试的关键技术难点包括：(1)建立 Agent 的协调(Coordination)机制；(2)Agent 的远程动态监控(Monitoring)；(3)定义服务测试 Agent 之间的协议(Agent Interaction Protocols，AIP)，以便有效地在各 Agent 之间交换命令、请求、各种状态和数据信息。

4.1 协调机制

Agent 之间通过交互协作共同构建 MAS，这不仅仅是一个通信问题，除了定义交互协议和交互规则(Rule)，更重要的是一个协调的问题。协调的目的就是要管理 Agent 之间的交互行为以及依赖关系。在 Web 服务测试的场景下，如何去定义测试流程中 Agent 之间的协调关系，如何去处理任务分解和协调行为，是拟解决的关键问题的重点。

Agent 之间的交互和协调是通过 ACL 消息传递来完成的。为了设计和管理交互行为和规则，需要定义 Agent 的可观察行为和设计其交互协议，定义交互规则，处理 Agent 的创建和销毁、通信行为和同步等问题。本框架采用的协调机制模型既要关注协调对象(Coordinable)即 Agent 所发生的事件和状态的变化，也要关心 Agent 之间的具体数据交换。一方面通过共享数据空间进行数据交换和通信，另一方面共享数据空间作为主动的协调媒体同时担负着监控和检测通信事件的任务。作为协调媒体(Coordination Media)的 TestCoordinator 通过对规则引擎的查询和更新自动完成协作和调度的过程，基于测试计划进行任务分解，并把子任务绑定到分布式计算资源。计算资源划分为不同的协调分组，协同环境下协作感知，要求 Agent 对信息的访问和操作能够被同一分组内其他的协作 Agent 所感知。

4.2 监控机制

监控机制是系统鲁棒性的关键。在动态的、复杂的多 Agent 分布式环境下，需要以一种总揽全局的方式了解整个系统和各个 Agent 的状态。由于 Agent 之间的交互和 Agent 的状态不一定是可预知的，可能会有通信失败、执行任务失败的情况。因此，需要监控 Agent 的运行行为，以监测各种失效和错误并且引导系统或用户去诊断和修复。

TestMonitor 的关注点是在分布式环境下感知环境和被测试对象 Web 服务的状态、检测任务、计划进度、个体状态、协调分组状态、失败和错误，保证监控结果的精确性和正确性。这就需要检测对系统预定义行为和 Agent 之间的交互关系等的违反情况，比较预测值、关系和状态与实际监控值、关系和状态，诊断这种违反情况并且引导系统修复。本系统的监控，一方面要针对预定义测试流程和行为去监控 Agent 的动作，另一方面系统要以一种可视化的方式提供给用户整个系统的运作情况、测试计划的执行进度、单独 Agent 状态和 Agent 协调分组状态等等。Agent 之间协调行为的修正是由监控机制触发的，协调媒体是有效的监控器。

Agent 的内部结构可能并非事先预知，监控机制要以 Agent 之间的通信消息作为切入点。基于预定义的面向 Web 服务测试的 Agent 交互协议，TestMonitor 判断 Agent 之间是否正以一种正确的方式或序列进行协作。

4.3 交互协议

为了让 Agent 协同工作，需要相应的约束和规则，而 Agent 交互协议就是用于定义这些约束和规则，以管理 MAS 中 Agent 的交互和通信。Agent 交互协议可以使 Agent 以一种正确的可预知的方式运作，是 MAS 中的决定因素。Agent 交互协议也可以看作是一种可重用的软件组件，它代表着 Agent 的交互模式。

定义交互协议，实际上是定义了 Agent 之间相互传送的消息本身和 Agent 的协作行为。我们采用 AUML(Agent Unified Modeling Language)定义系统行为，包括测试 Agent 之间的协作、交互的内容和测试领域相关的本体。从 Web 服务协同测试的角度出发，综合考虑协议的基本元素，包括目的、消息、本体、规则、输入和输出、状态和转移、参与者，重点针对 Web 服务测试的动态性和不确定性，设计可扩展和可重用的 Agent 交互协议。同时，基于 Colored Petri Nets(CPN)，我们将使用形式化方法描述 Agent 交互协议。

结束语 Web 服务是未来分布式系统中应用集成的主流，Agent 的 FIPA 规范和 Web 服务标准有很多相似的地方，利用 Agent 的智能化特征从不同的层次上扩展 Web 服务功能目前已有大量的研究成果。Web 服务的可信性是未来计算机软件发展关键的问题，而 Agent 在 Web 服务测试中的应用是一个具有挑战性的领域。为了实现 Web 服务在分布式环境下的协同式测试，结合 Agent 的自主性、反应性、适应性、社会性等特点，我们对基于 MAS 的 Web 服务测试(MAST)进行了探讨，以满足 Web 服务测试的开放性、协同性、动态性和不确定性等要求。目前在 Agent 和 Web 服务不同的范例和标准的约束下，二者结合是基于相互适配，只有吸取相互的优点，改进相关的标准和范例，以达到兼容，让 Agent 可以作为一个内部元素参与到 Web 服务应用中，才能达到真正的融合，反之亦然。随着相关研究的深入，二者的结合必将在不同的应用领域发挥越来越大的作用，创造巨大的价值。

参 考 文 献

1 Integrating Web Services into Agentcities Recommendation. http://www.agentcities.org

2 Gibbins N, Harris S, Shadbolt N. Agent-based Semantic Web Services [C]. In: Proceedings of the 12th International Conference on World Wide Web. 710～717

3 Greenwood D, Nagy J, Calisti M. Semantic Enhancement of a Web Service Integration Gateway [C]. The Workshop on Service-oriented Computing and Agent-based Engineering, 2005. http://www.whitestein.com/resources/papers/SOCABE05.pdf

4 Greenwood D, Calisti M. An Automatic, Bi-directional Service Integration Gateway [C]. In: Proceedings of the 2nd International Workshop on Web Services and Agent Based Engineering, 2004. http://www.agentus.com/WSABE2004/program/7.pdf

5 Dickinson I, Wooldridge M. Agents are not (just) Web Services: Considering BDI Agents and Web Services [C]. In: The Workshop on Service-oriented Computing and Agent-based Engineering, 2005. http://www.hpl.hp.com/techreports/2005/HPL-2005-123.pdf

6 Lyell M, Rosen L, Casagni-Simkins M, et al. On Software Agents and Web Services: Usage and Design Concepts and Issues [C]. In: Workshop on Web Services and Agent-based Engineering. http://www.agentus.com/WSABE2003/program/lyell.pdf

(下转第 150 页)

②“与分解”操作

如果子兄弟目标是逻辑与关系，而且孙子兄弟目标是逻辑或关系，这种情况下，要进行“与分解”操作，要将与关系分解，如图 3(c)所示。

③“或分解”操作

如果子兄弟目标是逻辑或关系，而且孙子兄弟目标是逻辑与关系，这种情况下，要进行“或合并”操作，要将与关系分解，如图 3(d)所示。

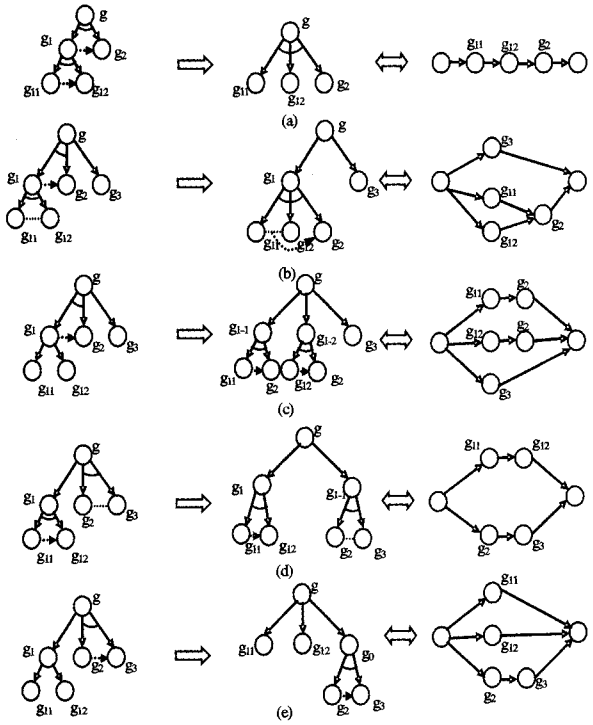

图 3 目标任务网络的规范

④“或合并”操作

如果子兄弟目标是逻辑与关系，而且孙子兄弟目标是逻辑或关系，这种情况下，要进行“或合并”操作，要将或关系合并，如图 3(e)所示。

此外，在形成规范的目标分解后，需要对目标分解形成的各种方案进行评估，有关分解方案评估的内容将在以后介绍。

结束语 本文针对迁移工作流在发布工作流服务需求之前目标任务的分解，提出了一种目标分解的结构模型，将目标分解形成与/或任务网络，并对目标结果之间的时序关系进行了详细分析。进一步的工作是研究合理有效的知识表达，以及对目标分解结果的评估机制。

参 考 文 献

1 曾广周，等. 基于移动计算范型的迁移工作流研究. 计算机学报，2003，26(10)：1343～1349

2 Rolland C，Souveyet C，Ben A C. Guiding goal modeling using scenarios. IEEE Transactions on Software Engineering，1998，24(12)：1055～1071

3 李淑霞，等. 敏捷制造车间任务规划系统的设计与开发. 中国机械工程，2003，8：1299～1302

4 Srivastava B，Koehler J. Web service composition-current solutions and open problems. In：Procedings of ICAPS'03 Workshop on Planning for Web Serviecs，Trento，Italy，2003. 28～35

5 曹健，等. 基于目标驱动和过程重用的 Web 服务客户化定制模型. 计算机学报，2005，28(4)：721～730

6 Albert D B. A Survey of Factory Control Algorithms that can be Implemented in a Multi-agent Heterarchy：Dispatching，Scheduling，and Pull. J. of Manufacturing Systems，1988，17(4)：297～320

7 Chung P W，Cheunga L，Stader J. Knowledge-based process management-An approach to handling adaptive workflow. Knowledge-based Systems，2003，16(3)：149～160

8 Cohen P R，Levesque H J. Intention is choice with commitment. Artificial Intelligence，1990，42(2-3)：213～261

9 Jennings N R，et al. Using Archon to Develop Real-world DAI Applications. IEEE Expert Intelligent Systems & their Applications，1996，11(6)

(上接第 143 页)

7 Richards D，van Splunter S，Brazier E，et al. Composing Web Services Using an Agent Factory [C]. In：the 1st International Workshop on Web Services and Agent-based Engineering，2003. http://www.agentus.com/WSABE2003/program/richards.pdf

8 Maximilien E M，Singh M P. Agent-based Architecture for Autonomic Web Service Selection [C]. In：the 1st International Workshop on Web Services and Agent-based Engineering，2003. http://www.agentus.com/WSABE2003/program/maximilien.pdf

9 Sycara K，Paolucci M，Soudry J. Dynamic Discovery and Coordination of Agent-based Semantic Web Services [J]. IEEE Internet Computing，2001，8(3)：66～73

10 Maamar Z，Most'efaoui S K，Yahyaoui H. Towards an Agent-based and Context-oriented Approach for Web Services Composition [J]. IEEE Transactions on Knowledge and Data Engineering，2005，17(5)：686～697

11 Huang C L，Lo C C，Li Y，et al. Service Discovery Through Multi-agent Consensus [C]. In：Proceedings of the Workshop on Service Oriented System Engineering (SOSE 2005)，2005. 37～44

12 Baldoni M，Baroglio C，Martelli A，et al. Reasoning About Interaction for Personalizing Web Service Fruition [C]. In：Proceedings of WOA 2003：dagli Oggetti agli Agenti，Cagliari，Italy，September 2003

13 白晓颖，赵冲冲，戴桂兰. Web 服务测试研究[J]. 计算机科学，2005，33(2)：252～256

14 罗玲，白晓颖. Web Services 技术分析[J]. 计算机科学，2003，31(4)：19～23

15 De B. Web Services - Challenges and Solutions：[WIPRO white paper]. 2003. http://www.wipro.com

16 De B. Testing Web Services - An Overview：[WIPRO white paper]. 2003. http://www.wipro.com

17 http://www.wsmo.org/

18 Bai X，Dai G，Xu D，et al. A Multi-agent Based Framework for Collaborative Testing on Web Services [C]. In：Proceedings of the 2nd International Workshop on Collaborative Computing，Integration，and Assurance，WCCIA，2006. 205 ～ 210