<<面向Agent的软件设计开发方法>>

图书基本信息

书名：<<面向Agent的软件设计开发方法>>

13位ISBN编号：9787121079849

10位ISBN编号：7121079844

出版时间：2009-1

出版时间：电子工业出版社

作者：薛霄

页数：279

前言

Agent和多Agent系统（MAS）现在已成为一种应对各种复杂IT情景的强大技术，如生产过程、Web服务、基于Intemet计算的市场和分布式网络管理等。
然而，新近出现的理解认为，MAS不仅是一种有效的技术，还代表了二种新型的软件开发通用范型，即基于自主软件实体（Agent）的设计和开发应用。
这种实体位于某个环境中，可以通过高层协议和语言的交互来灵活实现其目标。
这些特点非常适合于解决现时f青景下的复杂软件开发。
事实上，①自治的应用组件，反映了现代分布式系统内在的分散性，并且可视为系统被不同的利益相关者所拥有，在模块化和封装概念上进行了自然延伸；②Agent运行和交互（包括相互之间及Agent与环境之间）所采取的灵活方式，适应于现今软件在动态和不可预知的情况下运行；③Agent的概念为人工智能的成果提供了一个统一的观点，通过使用Agent和MAS作为存放智能行为的、可靠的和易管理的知识库，从而利用人工智能的成果解决现实世界中的问题。
在过去几年中，基于Agent的计算被日益接受为一种新型的软件工程范型，已经有大量的研究是关于定义合适的模型、工具和技术，以支持开发复杂的MAS软件系统。
这些研究，即面向Agent的软件工程（AOSE），不断地提出各种新的建模方法和技巧、新的设计方法和工具，尤其是新型的面向Agent的范型。
关于AOSE的科学论文在文献中出现得越来越多，遍布在不同的会议、期刊和新闻上。
因此，无论是新人还是专家，在这个领域中进行研究时，在操作所有这些材料时总会有困难。
本书试图将各种研究结果和建议有组织地综合在一起，虽然非常多样化，但都以促进。
MAS的开发为相同的总体目标。
我们的希望就是，这本书能够为研究者和学生了解AOSE的发展现状提供线索，而不用在现有的数字图书馆中搜索数以千计的文件，也不会在无尽的搜索中迷失方向。
当然，我们需要清楚地认识到，AOSE的研究仍处于初级阶段。
在AOSE被广泛接受，并且在MAS复杂软件系统研究中成为实际可用的范型之前，就必须面对所出现的挑战。
出于这些原因，本书避免支持特定的技术或方法，而只是给读者介绍不同的设计方法和实现技术，给予他们更多的选择余地。
本书的内容共分为五个部分。

内容概要

在过去几年中，Agent和多Agent系统（MAS）已经成为一种应对各种复杂IT情景的强大技术，有大量的研究是关于定义合适的模型、工具和技术以支持开发复杂的MAS软件系统。
目前关于面向Agent（AO）方法的科学文献出现得越来越多，遍布在不同的会议、期刊和时事新闻上。
因此，无论是新人还是专家，在这个领域进行研究时，都难以操纵所有这些材料。
本书试图将各种研究结果和思想有组织地综合在一起，虽然非常多样化，但都以促进复杂MAS软件系统的开发作为总体目标，希望能够为研究者和学生了解AO方法的发展现状提供线索，而不用在现有的数字图书馆中查阅数以千计的文件，也不会在无尽的搜索中迷失自己。
读者同时可以了解到软件工程新的发展趋势，以及如何将Agent思想应用于目前软件界所出现的种种新技术（比如SOA、网格服务等）中。
本书可以作为计算机软件专业硕士生和博士生的教材和参考用书，对于从事Agent理论和技术研究的人员，尤其是从事面向Agent软件工程研究的人员以及基于Agent技术的工程实践人员均具有较高的参考价值。

<<面向Agent的软件设计开发方法>>

书籍目录

第1章 面向Agent的软件开发抽象 1.1 引言 1.2 Agent的开发抽象 1.3 Agent的体系架构 1.4 Agent的组织类型 1.5 Agent与组件的对比 1.6 语义重用的Agent和组件 1.7 小结第2章 面向Agen七的软件工程 2.1 引言 2.2 关键主题 2.2.1 需求工程 2.2.2 语言 2.2.3 建模语言 2.2.4 平台 2.2.5 方法学 2.3 方法过程 2.3.1 分析 2.3.2 设计 2.3.3 实现 2.3.4 测试 2.4 更多的信息 2.5 小结第3章 面向Agent的经典开发方法 3.1 引言 3.2 Gaia方法 3.2.1 Gaia初始版本 3.2.2 Gaia的第2个版本 3.2.3 RoadMap方法 3.2.4 采用AUML对Gaia的扩展 3.2.5 结论 3.3 Tropos方法 3.3.1 概况 3.3.2 形式化Tropos方法 3.3.3 基于社会性的MAS架构 3.3.4 目标模型 3.3.5 结论 3.4 MaSE方法 3.4.1 概况 3.4.2 分析阶段 3.4.3 设计阶段 3.4.4 AgentTool 3.4.5 结论 3.5 小结第4章 面向Agent的特殊开发方法 4.1 引言 4.2 ADELFE方法 4.2.1 背景介绍 4.2.2 初始需求 4.2.3 最终需求 4.2.4 分析阶段 4.2.5 设计阶段 4.2.6 ADELFE 工具 4.2.7 结论 4.3 MESSAGE方法 4.3.1 背景介绍 4.3.2 疗法综述 4.3.3 旅行Agent案例分析/设计 4.3.4 对于底层设计的考虑 4.3.5 MESSAGE的评价 4.3.6 结论 4.4 Prometheus方法 4.4.1 方法综述 4.4.2 系统规范 4.4.3 框架设计 4.4.4 详细设计 4.4.5 工具支持 4.4.6结论 4.5小结第5章 面向Agent方法的比较和评估 5.1 引言 5.2 评估框架 5.2.1 概念和属性 5.2.2 符号和建模技巧 5.2.3 开发过程 5.2.4 语用 5.2.5 衡量标准 5.3 对Gaia的评估 5.3.1 概念和属性 5.3.2 符号和建模技巧 5.3.3 开发过程 5.3.4 语用 5.4 对Tropos的评估 5.4.1 概念和属性 5.4.2 符号和建模技巧 5.4.3 开发过程 5.4.4 语用 5.5 评估MaSE 5.5.1 概念和属性 5.5.2 符号和建模技巧 5.5.3 开发过程 5.5.4 语用 5.6 评估总结 5.6.1 支持阶段 5.6.2 Agent架构 5.6.3 开放系统中的交互 5.6.4 迭代开发 5.6.5 辅助要素 5.7 小结第6章 按需定制的开发框架HDA 6.1 引言 6.2 HDA的定义 6.2.1 方法工程学 6.2.2 HAD的定义 6.2.3 HDA的使用规则 6.2.4 元模型 6.2.5 潜在的问题 6.2.6 应用的关键 6.3 基于HDA的设计模式划分 6.3.1 Agent组织层次模式 6.3.2 Agent交互层次模式 6.3.3 Agent协调层次模式 6.3.4 Agent架构层次模式 6.3.5 移动Agent层次模式 6.4 设计模式在AgentBuilder中的应用 6.5 小结第7章 HDA在C41系统项目中的应用 7.1 引言 7.2 方法选取阶段 7.2.1 RoadMap建模方法 7.2.2 人工鱼建模方法 7.3 需求分析阶段 7.3.1 选取元模型 7.3.2 C4I系统中的应用 7.4 MAS框架设计阶段 7.4.1 选取元模型 7.4.2 C4I系统应用 7.5 Agent建模阶段 7.5.1 选取元模型 7.5.2 C4I系统应用 7.6 软件实现阶段 7.6.1 元模型抽取 7.6.2 C4I系统的应用 7.7 小结第8章 AUML方法 8.1 引言 8.2 AUML的目的 8.3 目前AUML的相关工作 8.3.1 时序图 8.3.2 Agent类图 8.4 AUML的发展方向 8.4.1 模型 8.4.2 工具 8.4.3 算法 8.4.4 语义学 8.4.5 应用 8.5 小结第9章 多Agent系统的基础设施 9.1 引言 9.2 MAS的基础设施 9.2.1 MAS基础设施的概念 9.2.2 MAS基础设施的作用 9.2.3 基础设施的授权VS控制 9.2.4 与FIPA兼容的基础设施 9.3 授权型基础设施JADE 9.3.1 运行时系统 9.3.2 Agent模型 9.3.3 测试和管理工具 9.3.4 应用 9.4 MAS中的协调基础设施 9.4.1 AS中的协调模式 9.4.2 协调对MAS工程化的影响 9.4.3 MAS协调的行为理论框架 9.4.4 制品（Artifact）与协调基础设施 9.4.5 MAS工程中的协调平衡 9.5 小结第10章 面向Agent软件工程的路线图 10.1 引言 10.2 Agent作为新的建模范型 10.3 构建多Agent系统的方法 10.3.1 FIPA对于未来MAS设计的建议 10.3.2 MAS的验证和测试 10.4 实现、部署和运行的工具 10.4.1 Agent的设计工具 10.4.2 Agent的实现工具 10.4.3 Agent的部署工具 10.5 应用机遇 10.5.1 信息服务中的Agent 10.5.2 普适计算中的Agent 10.6 面向Agent的软件工程路线图 10.7 小结参考文献附录A 中英文缩略词对照表

章节摘录

1.自治性（Autonomy）自治性也就是指一方具有独立性，能够根据自己的意愿来行动。
从广义上讲，商务交易中参与者的自主决策就反映了自治性。
没有人能够强迫你买卖任何东西；也没有人能够强迫你听命于另一方，或者与之妥协；更没有人能够强迫你使用某种特定的推理策略。
特别地，一个自治个体甚至不需要任何外部意义下的“理智”，因为这样的要求会限制它的自治性。
如果某种方法能够使交互各方的自治性呈现出良好的效果，那么这种方法应该很容易适用于几乎所有的场合。
值得注意的是，有些场合所呈现出的自治现象，真实情况或许并不是由于自治性，而是由其他一些原因引起的。
例如，参与者没有对消息做出反应，可能并不是它做出的自治性反应，而是因为基础机制失效。
当然，在相同的框架下，实现自治的参与者也可以自如地处理基础机制的变化。
因此，判断某个行为是自治性的体现，还是由于基础机制的失效而引起的，这点可能很重要。
一般来说，不受约束的计算自治性就如同现实世界中不受约束的自治性一样，将会产生不可预期的结果。
并且，为了给出完整的计算模型，或者做出可靠的预测，我们都必须假设参与者的自治I生在某种程度上受到约束。
典型地，根据协议进行交互就可以实现对自治性的限制。
2.异构性（Heterogeneity）异构性源于设计者构建组件的方式相互独立，导致组件的信息模型或过程模型各不相同。
一般来说，在功能系统中，异构性的产生是由于历史原因造成的。
没有人开始就想设计一个异构系统，但是构建大型系统的最终结果往往是异构的。
在设定一个开放系统的参数时，不假定内部结构的同质性是非常重要的。
最终，为了能够让这些组件一起运作，就要对它们之间的异构性施加一定的限制。
也就是说，一定要有对共同性的说明。
就信息模型而言，通过一个共享本体来捕捉共同性（Gruber，1991）；这是同Agent相关的，但是并不特指Agent，因为它的产生也是源于将异构信息源组合起来。
就过程模型而言，可以通过确定典型外部事件的方法来捕捉共同性（Singh，2003）。
这个思路是，Agent的行为标记是指对外界的响应结果，而不需要暴露内部的构造细节。
这些标记具有被标准化的潜力，事实上是分布式数据库事务处理中两阶段提交协议所采用的方法（Gray和Renter，1993）。
任何按照特定标准实现的Agent，都被要求公布合适的特定事件，但并不需要暴露内部的实现细节。
3.动态性（Dynamism）动态性是指管理者能够独立对系统进行灵活配置，并且可以根据需要改变其配置，而不需要明确通知相关各方（系统中的其他成员）。
开放系统具有最大程度的动态性，因为原则上它们根本不需要管理者。
事实上，它们出于特定目的可能会需要一些管理功能，如监测和安全、根据应用限制潜在的成员参与等。
原则上，这些功能可以分布于成员之间，不需要管理者，但是出于社会政治原因，系统中经常需要有一方来负责。
Agent方法所倡导的动态配置技术有一些变种，已经被广泛使用，例如在UDDI中。
然而，更深层次的挑战并不在于服务发现，.也就是UDDI所强调的，而是在于如何从几个可能发现的服务中选择一个最合适的服务实现。
一些与之最相关的Agent方法。

编辑推荐

《面向Agent的软件设计开发方法》可以作为计算机软件专业硕士生和博士生的教材和参考用书，对于从事Agent理论和技术研究的人员，尤其是从事面向Agent软件工程研究的人员以及基于Agent技术的工程实践人员均具有较高的参考价值。

版权说明