基于 OAA 的实时多 Agent 系统研究①

俞 明 （南通明兴科技开发有限公司 南通 226000）
韩慧明 （重庆天然气净化总厂长寿分厂 重庆 401200）
董振华 （中国科学院软件研究所 北京 100080）
吴晴春 （南通明兴科技开发有限公司 南通 226000）
徐俊刚 王 强 （中国科学院软件研究所 北京 100080）

摘要：本文通过把实时调度 Agent 作为一个元 Agent 集成到 OAA（Open Agent Architecture）中，提出了一种基于 OAA 的实时多 Agent 系统结构。其中实时调度 Agent 通过用 ICL（Inter－agent Communication Language）描述的接口和过程与 OAA 中的 Facilitator 进行通信，从而能够为实时 Agent 的通信、调度和协调提供实时服务，并且支持嵌入不同的实时调度算法以满足不同的系统需求，改进系统的实时性。

关键词：实时多 Agent 系统 OAA 实时调度

1 引言

随着经济的全球化，企业的外部环境逐渐表现出虚拟性、实时性和协调性，这意味着企业要在竞争中占有优势，就应快速、准确地满足客户的要求，有效的组织新产品的生产，也就是说企业应具有实时的动态调度能力。面向这种需求，本文提出了一种面向实时多 Agent 系统的开放体系结构，可以满足各种应用对系统开放性和实时性的需求，使得系统中的各种 Agent 能够在特定的时间和质量约束下通过相互通信、协商和合作来满足应用需求和服务质量。

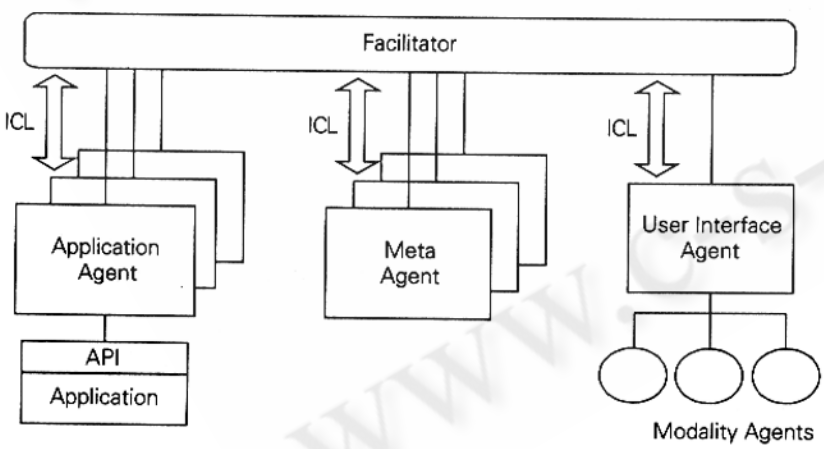

图 1 OAA 系统体系结构

2 研究背景

DECAF（Distributed, Environment－Centered Agent Framework）[1]和 Kcobalt[2]都是面向多 Agent 系统的体系架构。但是，这些多 Agent 系统结构都不提供实时特性。

OAA[3]是 SRI 公司开发的一种开放的 Agent 体系结构，它提供了基于 Agent 的分布式计算框架，利用 Facilitator 机制实现了多 Agent 间透明的通信与合作，通信语言采用 ICL（Inter－agent Communication Language）。OAA 系统的典型结构如图 1 所示，包括一个或多个用户接口 Agent，多个应用 Agent 和 Meta－Agent，它们之间地位平等，均和 Facilitator 相连。

Facilitator 是一种特殊的服务型 Agent，具体负责 Agent 间的通信和合作。应用型 Agent 能提供某类服务，这些服务可以是领域无关的，也可以是与具体用户、具体领域相关的。应用 Agent 可以通过对遗留系统进行包装而获得。Meta－Agent 的作用是协助 Facilitator 协调其他 Agent 间的合

① 国家自然科学基金项目，No. 60373055 与 60374058；江苏省南通市科学技术局工业科技创新计划项目，No. A4014。

作。用户接口 Agent 主要负责与用户的交互，可接收多种方式的输入并将其进行正确的转换。

不过，OAA 体系结构也存在以下问题：(1) 一个 Facilitator 可能会造成通信瓶颈，影响了系统的扩展性和容错性；(2) Facilitator 功能过于集中，需要管理 Agent 的能力登记、规划复杂任务、控制引导任务的执行等；(3) 与现有的实时 Agent 体系结构相比，OAA 对实时约束考虑较少，只在请求服务中提供一个 Time 参数指明完成请求的截止期，但在发布各种能力的服务中没有提供明确的实时服务信息。

为了弥补 OAA 在实时调度方面的不足和分散 Facilitator 的功能，我们在 OAA 体系结构的基础上，增加了实时 Agent 和实时调度 Agent 两种类型的 Agent，提高了 OAA 系统的实时性和调度能力，从而为实现实时多 Agent 系统提供开放的体系结构。

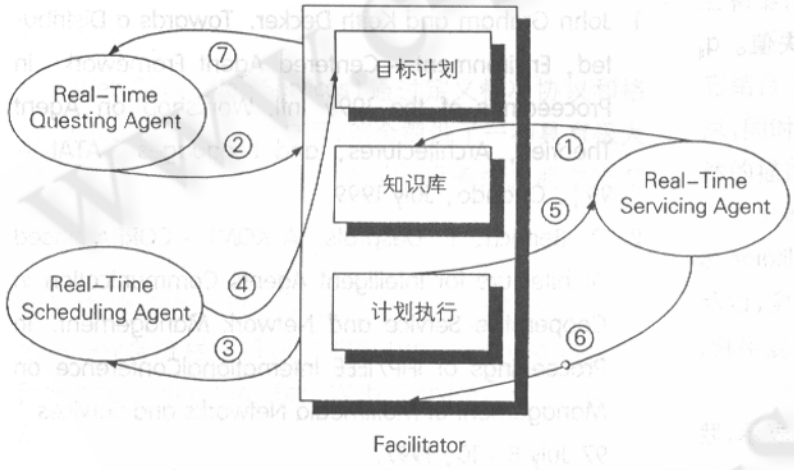

图 2　基于 OAA 的实时多 Agent 系统体系结构

3　基于 OAA 的实时多 Agent 系统模型

在 OAA 的基础上，通过定义两种实时 Agent 和一种实时调度 Agent，提出了一种实时多 Agent 系统的体系结构，如图 2 所示。实时 Agent 包括实时请求 Agent 和实时服务 Agent，前一种指的是用户接口 Agent 和部分应用型 Agent，后一种主要指的是应用型 Agent。详细的调度和运行机制是这样的：(1) 实时服务 Agent 向 Facilitator 注册服务能力的各种参数，并且存放在知识库中；(2) 实时请求 Agent 向 Facilitator 发送一个请求；(3) Facilitator 把请求传递给实时调度 Agent 进行调度；(4) 实时调度 Agent 将调度结果（主要是指请求的优先级）提交给 Facilitator，Facilitator 中的目标计划模块根据请求的参数和优先级进行规划；(5) Facilitator 中的计划执行模块安排这一请求到相应的实时服务 Agent 执行；(6) 实时服务 Agent 将执行结果提交给 Facilitator；(7) Facilitator 将执行结果返回给实时请求 Agent。

我们提出的实时多 Agent 系统架构体现了 Agent 间交互请求的时间和价值（重要性）约束两个特征。和文献[4,5]提到的假设一样：每一个 Agent 请求（任务）可以由一个或多个 Agent 以多种方式来完成，每一种方式有不同的执行时间，并且返回结果的质量等级也不同；不失一般性，假设执行时间越长，执行结果的质量就越高。

3.1　实时 Agent 模型

实时 Agent 包括实时请求 Agent 和实时服务 Agent。实时请求 Agent 主要指的是用户接口 Agent，也可以是其它应用型 Agent，它向其它 Agent 提出各种实时请求服务。实时服务 Agent 能够对特定的实时请求 Agent 提出的请求任务进行求解。在某些情况下，一个应用型 Agent 可以既是实时请求 Agent，又是实时服务 Agent。下面我们对这两种实时 Agent 进行形式化描述。

3.1.1　实时请求 Agent

一个实时请求 Agent 我们用 RTRAgent 表示，它可以这样定义：

$$RTRAgent = \{R_1, R_2, \cdots, R_n\}$$

每一个 RTRAgent 由一组请求（Request）$R_1, R_2, \cdots, R_n$ 组成，每一个请求就是一个要求解的任务，又可以定义为：

$$Ri = <C_i, Ns_i, D_i, V_i, H_i>$$

其中，C_i 表示请求的最坏情形执行时间，即请求在最坏情况下无中断执行所需的时间；Ns_i 表示能够完成请求的 Solvabe 的名称；D_i 表示请求的截止期，就是请求最好在某个时刻之前完成，如果系统不能保证请求在截止期之前完成，将进行降级处理；V_i 指的是请求的重要性水平，我们也可以说是请求的价值；H_i 指的是请求的质量阀值，就是特定请求返回执行结果的质量等级的最小值，请求必须由能够满足求解结果质量阀值的实时服务 Agent 来执行。

3.1.2　实时服务 Agent

一个实时服务 Agent 我们用 RTSAgent 表示，参考

文献[7]中的定义，我们这样定义它：

$RTSAgent = \{S_1, S_2, \cdots, S_n\}$

每个实时服务 Agent 可以提供一组服务(Service) $S_1, S_2, \cdots, S_k$，每个服务就是实时服务 Agent 针对实时请求 Agent 提出的请求而设计的解决方案，定义为：

$S_i = <O_i, ES_i>$

其中，O_i 代表 S_i 的最佳执行结果，表示为：

$O_i = optimum(o_{i1}, o_{i2}, \cdots, o_{im})$

ES_i 是 S_i 的各种执行方案(Execution Strategy)的集合，定义为：

$ES_i = \{es_{i1}, es_{i2}, \cdots, es_{im}\}$

其中 es_{ij} 表示 S_i 的一个具体执行方案，定义为：

$es_{ij} = <o_{ij}, ex_{ij}, q_{ij}, tv_{ij}>$

其中，o_{ij} 表示这个执行方案的结果，ex_{ij} 表示该执行方案的运行时间，q_{ij} 表示该执行方案所属的质量等级，tv_{ij} 是一个折衷值，表示从一种较优的执行方案降至次之的执行方案时单位时间内的质量等级损失值。q_{ij} 和 tv_{ij} 的取值可按下列公式计算：

$q_{ij} = (o_{ij}/O_i)$

$tv_{ij} = ((q_{ij} - q_{ij+1})/q_{ij})/(ex_{ij} - ex_{ij+1})$

3.2 实时调度 Agent 模型

实时调度 Agent 的目标就是对经过 Facilitator 传来的实时请求 Agent 提出的各种请求进行调度，以尽可能地满足各个请求的截止期，使系统尽快恢复平稳，获得最大系统效益。

一个实时调度 Agent 我们用 RTSSAgent 来表示，我们这样定义它：

$RTSSAgent = \{SS_1, SS_2, \cdots, SS_n\}$

每个调度 Agent 可以提供一组调度方案(Scheduling Scheme) $SS_1, SS_2, \cdots, SS_n$，每个调度方案是针对实时请求 Agent 提出的实时请求而设计的实时调度方案，其中嵌入一种实时调度算法，它又可以这样定义：

$SS_i = \{A_i, O_i, I_i\}$

其中 A_i 是嵌入调度方案中的调度算法名称；O_i 指的是调度方案的输出，一般指任务的优先级，也可以是其他输出；I_i 指的是调度方案的输入，可以是调度的截止期、价值等参数，它是 R_i 的子集：

$I_i \subseteq R_i$

就是说调度方案可以是 design - to - time 的，也可以是 design - to - criteria 的，这根据实际情况来确定选用何种调度方案。对于加工新订单来说，如果时间是最重要的，可以使用嵌入 EDF(Earliest Deadline First：最早截止期优先)算法的调度方案；如果价值(关键性)是最重要的，可以考虑使用嵌入 HVF(Highest Value First：最大价值优先)算法的调度方案，等等。

4 结论及进一步工作

本文提出了一种基于 OAA 的实时多 Agent 系统体系结构，并且对各种 Agent 进行了形式化描述。在这个结构中，我们把调度 Agent 作为一个元 Agent 集成到 OAA 中，调度 Agent 通过用 ICL 描述的接口和过程与 Facilitator 进行通信。

参考文献

1 John Graham and Keith Decker. Towards a Distributed, Environment - Centered Agent Framework. In Proceedings of the 1999 Intl. Workshop on Agent Theories, Architectures, and Languages [ATAL - 99], Orlando, July 1999.

2 D. Benech, T. Desprats. A KQML - CORBA based Architecture for Intelligent Agents Communication in Cooperative Service and Network Management. In Proceedings of IFIP/IEEE InternationalConference on Management of Multimedia Networks and Services '97 July 8 - 10, 1997.

3 David L. Martin, Adam J. Cheyer, Douglas B. Moran. The Open Agent Architecture: A Framework for Building Distributed Software System. Applied Artificial Intelligence. vol. 13, pp. 91 - 128. Jan - Mar 1999.

4 Alan Garveyand Victor Lesser. Design - to - time Real - Time Scheduling. IEEE Transactions on Systems, Man and Cybernetics - Special Issue on Planning, Scheduling and Control. vol. 23, no. 6, 1993.

5 Thomas Wagner and Victor Lessor. Design - to - Criteria Scheduling: Real - Time Agent Control. Proceedings of the 2000 AAAI Spring Symposium on Real - Time Systems.