软件系统协调技术研究

王成耀 郑雪峰 涂序彦
(北京科技大学计算机科学与技术系 北京 100083)

摘 要 复杂软件系统常常由若干子系统或组件通过组装而成,子系统或组件之间的有效协调是系统开发的一个关键问题。特别是在开放、分布和移动的企业计算环境下,软件系统包含大量异构实体,对协调的可重用性、动态性和适应性提出了更高的要求,以应对业务环境和用户需求的持续变化。对软件系统协调技术的研究现状进行了综述,从协调体系结构、协调设计与实现方法学、协调活动的表示等视角对目前的协调技术进行了总结,探讨了复杂软件系统的协调策略,最后指出了目前协调技术存在的问题和面临的挑战。

关键词 协调,软件
中图法分类号 TP311.5 **文献标识码** A

Research on Coordination Technology of Software Systems

WANG Cheng-yao ZHENG Xue-feng TU Xu-yan
(Department of Computer Science and Technology, University of Science and Technology Beijing, Beijing 100083, China)

Abstract The complex software systems are often built by many subsystems or components in a compositional way. Coordination among subsystems or components is a key issue in complex software development. Especially in open, distributed and mobile enterprise computing environments, where software systems involve a large number of heterogeneous entities, coordination is demanded to be of higher reusability, dynamics and adaptability so as to cope with continuous changes of the business environments and user requirements. Recent research developments on software coordination technology were surveyed, and the current art of the state of the coordination technology was summarized in various views including coordination architectures, coordination methodologies of design and implementation, and representation of coordination activities. Then coordination strategies of complex software systems were presented. Finally current problems and challenges of coordination technology were discussed.

Keywords Coordination, Software

1 引言

协调(Coordination)是复杂大系统的核心问题,也是计算机科学、工程技术、社会经济以及生物生态等领域持续发展的普遍需求与共性问题[1]。在大系统中,由于规模庞大、结构复杂、功能综合、因素众多,直接应用最优控制理论或运筹学的方法,试图一揽子解决大系统的动态最优化与静态最优化,不仅分析工作量很大,而且技术上也很难实现。为此,在大系统理论中,发展了"分解-协调"方法[1,2]。

从软件开发的角度来看,软件系统复杂性的日益增加,迫使人们去研究有效的开发方法,通过重用现有的设计成果,以简化和加速软件开发过程。无论是设计模式、软件体系结构,还是基于组件的软件开发、基于智体(Agent)的软件开发、面向服务的软件集成,都是试图通过应用分治(Divide and Conquer)策略,一方面从整个系统中抽象出一些可重用的基本成分,以便提升软件的可重用性;另一方面,通过组装已有的软部件(进程、组件、服务、智体等)快速实现复杂系统的构建。然而,当我们面临具有较大规模和复杂性的软件系统时,如何才能使系统的各个软部件之间实现有效协调,并尽可能适应业务环境和业务需求的变化,是系统开发过程中极为重要的问题。

本文首先简要介绍协调的基本概念,然后从协调体系结构、方法学以及协调活动的表示等多种视角,总结目前软件系统协调技术的研究现状,探讨复杂软件系统的协调策略,最后指出目前协调技术研究存在的问题和面临的挑战。

2 协调的基本概念

Malone与Crowston[3]将协调定义为"对活动之间相互依赖的管理"。Coates[4]认为:协调是一种活动,用来管理协作智体之间的依赖与可能出现的冲突,以达到和谐的结果。这些定义都在一定程度上给出了协调的基本内涵。

本文将软件系统的协调定义为:协调是一种活动,用来实现部件之间的同步和系统的全局优化。其中,同步是一种严格的协调,即强协调,是必需的,否则可能导致系统的错误行

到稿日期:2008-07-18 返修日期:2008-11-10
王成耀(1966-),男,教授,CCF会员,主要研究方向为软件工程、分布式系统等,E-mail:wanghengyao@sohu.com;郑雪峰(1951-),男,教授,主要研究方向为分布式系统、网络安全等;涂序彦(1937-),男,教授,主要研究方向为人工智能、人工生命等。

为或错误结果。必须同步的根本原因是资源共享，例如不同部件(如进程)对临界资源的互斥访问；消费者必须等待生产者的数据；客户请求后必须等待服务器的应答，等等。

全局优化是一种不严格的协调，即弱协调，是为了提高系统的性能或获得更好的结果，即使不协调也不会导致系统功能的错误。弱协调只能是尽力而为的。需要优化的根本原因是活动(任务)共享，即多个部件具有相同或重叠的功能。例如，若有两个部件执行的任务相同或部分重叠，其中一个可直接利用另一个的结果以提高效率；通过合理的任务分配实现负载平衡，等等。

3 研究现状

从 Carriero 与 Gelernter 开发了第一个协调语言 Linda[5] 以来，出现了许多协调模型和语言，例如 MARS[6]，Manifold[7]，Reo[8-10] 等。特别是近 10 年来，在计算机科学的许多研究领域，如分布式计算、多智体系统、普适计算、软件开发方法、计算机支持的协同工作、Web 服务等，国内外学者在不同层面围绕协调技术开展了广泛的研究，提出了许多协调模型与框架。我们把这些模型与框架归纳为以下几类：1)在体系结构层面研究协调；2)在设计与实现方法学层面研究协调；3)在协调活动的表示层面研究协调。

3.1 协调体系结构

从体系结构的角度看，协调模型包括 3 个主要部分[11]：

· 协调实体。包括被协调的实体(对象、组件、进程、智体、服务等)与协调器。

· 协调介质。用于数据共享或交换消息的通信，例如元组空间(tuple space)、频道(channel)等。

· 协调规则。协调实体必须遵守的规则，例如由协调器定义的实现被协调实体之间通信的接口等。

概括起来，典型的体系结构有基于共享数据空间的体系结构、基于频道的体系结构、基于角色的体系结构，等等。

(1)基于共享数据空间的结构模型

基于共享数据空间的结构模型采用数据驱动方式，实体之间不直接交互，而是通过共享的数据空间实现间接通信与协调，如元组空间[5]、联邦共享逻辑数据空间[12]、数据场(data field)[13]、黑板(blackboard)等。共享的数据空间可以是集中式的，也可以是物理上分散的。

Carriero 与 Gelernter 在并行计算领域提出的 Linda 是典型的基于共享数据空间的模型。Linda 采用元组空间存储和分配数据，进程通过协调原语从元组空间中存储和检索元组。元组空间作为不同实体之间交互的中介，降低了时间和空间的耦合度。此外，还有许多 Linda 的扩展模型，如 MARS[6]，Lime[12]，TSpaces[14]，LogOp[15] 等。其中，MARS 采用可编程元组空间，元组访问的事件可以触发某个计算活动，有利于施加特定应用的协调规则。文献[16]提出了 SBC(Space Based Computing)的概念，与 Linda 的主要区别在于：Linda 采用基于服务器的体系结构，而 SBC 采用完全分散的体系结构，可很好地适用于 P2P 与移动 ad-hoc 网络。

文献[13]提出了智能自律分散系统(Intelligent Autonomous Decentralized System，IADS)的概念和体系结构，给出了 3 种典型的 IADS 体系结构(集团型 IADS、联盟型 IADS、市场型 IADS)，可用于解决大系统的分散协调问题。在 IADS 中，各个组成部分都是一个自律的智体，它能自主完成本身功能，又能主动及时地发送信息，实现相互之间的协调、协商和协作。IADS 采用数据场作为智体之间的协调介质。

(2)基于频道的结构模型

频道是实体(如组件)之间一对一连接的数据结构，包括 source 和 sink 两个端点。组件可以写 source 端或读 sink 端。频道可以是同步、异步、移动或有条件的，例如 Manifold[7]，Reo[8-10] 等。

Reo 是一个基于频道的外协调模型，把组件看作黑盒。在该模型中，协调器称为连接器。复杂的连接器可由简单的连接器组成。Reo 可以作为并发进程的协调语言，也可作为连接器组合的胶水语言(glue language)[17]。Reo 的重点仅仅在于连接器及其组合，而不涉及所连接实体的内部行为。计算实体可以是代码片段、模块、被动或主动对象、线程、进程、智体、软件组件等。

文献[18] 提出了一种基于移动频道的组件协调模型，描述了用 Java 语言实现的方法。移动性允许频道连接的动态重配置。该模型支持组件可移动的动态分布式系统，提供了组件之间交互的方法，将计算部分与协调部分进行了清晰的分离，可以对协调结构进行开发和描述。

(3)基于角色的结构模型

基于角色的结构模型将软件系统看作一个组织，组织由相互关联的角色构成。特别是在包含大量同类实体的系统中，在协调模型中引入角色，一方面通过隐藏活动实体的动态性，从而简化协调，另一方面通过角色转换，以利于动态适应环境和业务的变化，例如，ARC(Actor，Role and Coordinator)模型[19,20]、powerJava[21]。

ARC 模型包含 3 种活动实体：执行者(actors)、角色与协调器，概念上是一种 3 层结构。执行者负责执行系统的功能，具有相同行为的一组执行者由角色来描述，将协调分为角色内协调和角色间协调。协调器负责角色间的协调，而角色内协调由角色负责。ARC 模型集成了面向数据和面向控制协调方法的优点，将协调控制分布到协调器和角色。角色隐藏了执行者的动态性，简化了协调，也利于重用。

文献[22]采用基于角色的方法，把协调实现为一个单独的子系统，在不同粒度级上来管理异构单元。协调子系统维护了功能系统的需求和当前状态，可以独立于功能子系统来描述和控制。采用 4 层结构：计算对象、功能角色、管理和约、组织。将角色分为功能角色与组织者角色。角色在不同时间可以由不同的参与者扮演。扮演组织者角色的参与者包括对象、组件、服务、智体以及复合体(composite)等。

在实际的复杂系统中，需要结合以上多种基本结构，如多重协调、多级协调、多步协调、多段协调、分区协调等[2]。

3.2 协调设计与实现方法学

关注点分离(Separation of concerns)一直是处理软件复杂性的重要原则，其主要思想是：一个给定问题包含若干个不同的关注点，将这些关注点分离处理，从而降低复杂性，增强适应性和重用性。根据该原则发展了不同的方法学。

协调提供了由相对独立甚至分布的实体构造软件系统的机制，可由下列等式来表示[23]：

程序 = 计算 + 协调

早期的协调模型(如 Linda)是在封闭系统的背景下提出

的，所有被协调实体在设计时已知，协调介质在概念上作为应用程序的一部分被编译。也就是采用协调作为语言的观点来建立系统。面对开放系统，要求协调介质支持运行时管理与动态适应的特性。因此，出现了协调作为服务以及协调作为方面的方法学。

(1)协调作为方面[24,25]

构成一个软件系统的各个实体不是孤立的，需要按照特定的协调协议进行交互。无论是面向对象还是基于组件的软件分解技术，协调协议不能被封装在单独的实体(对象、组件)中。因此，每个实体除了执行计算外，还需要实现与其他实体的协调，协调关注点与计算代码混杂在一起。为了将协调从计算中分离，出现了将协调作为方面的方法学。

Fuentes 与 Sanchez[24]提出了一种使用面向方面程序设计在实现级将协调与计算分离的方法，给出了从协调协议的状态机规范到面向方面制品的映射。Cuesta 等[25]展示了如何利用面向方面的 ADL(Architectural Description Languages)在体系结构级来表示协调方面。将协调关注点封装为方面，组件的重用性更好，组合更容易，有利于系统维护和进化。同时，面向方面使得组件不会意识到协调器的存在，避免了对外部协调器的引用。

(2)协调作为服务

协调作为服务可以大大降低协调与计算的耦合度，有利于实现运行时管理以及动态适应环境与用户需求的变化。LogOp[15]采用了协调作为服务的观点，扩展了 Linda 模型。文献[26]采用协调作为服务的观点，提出了一个形式框架，采用进程代数表示协调系统的语义。该框架将被协调的实体、交互空间与协调介质按角色明确分离，整个系统被划分为被协调空间(coordinated space)、交互空间(interaction space)与协调空间(coordination space)。其中，被协调空间包含被协调的实体，实体可以动态进入或离开被协调空间，发出协调请求或接收协调应答；交互空间是通信基础结构，负责转发协调请求/应答；协调空间作为协调基础结构，由若干协调介质构成，负责提供协调服务。

近年来，随着面向服务计算的兴起，面向服务的协调得到了研究者的广泛关注[27-30]。其中，WS-Coordination(Web Services Coordination)[27]与 WS-CF(Web Services Coordination Framework)[28]是用于 Web 服务环境下分布式活动协调的一般框架，协调实体包括客户、协调器与参与者。通过上下文来关联被协调实体的交互。客户是协调的发起者，由协调器根据上下文中定义的协调类型确定协调规则，负责客户和参与者在特定点上的协调。WS-Coordination 的协调服务包含 3 种子服务：激活服务(由服务提供者使用，创建服务的协调上下文)、注册服务(由服务请求者使用，通知协调服务有关的需求)与协议服务(执行实际的协调)。本质上，协议服务描述了协调服务的形式以及交互方式，特别是描述了用于交互的消息，但是没有指出协议服务如何执行协调活动。文献[29]提出了一种基于 ECA(Event-Condition-Action)规则的服务协调框架，采用基于 XML 的语言 WS-ECA (Web Services-ECA)描述 ECA 规则，ECA 规则定义了服务的反应性行为和服务交互，可以动态增加和移去。

3.3 协调活动的表示

解决复杂软件系统协调的关键问题是：如何把各个自主子系统之间的相互依赖和相关的协调过程进行抽象和表示。典型的模型有基于状态机的协调、适应性协调、基于公共知识库的协调等。

(1)基于状态机的协调

协调活动依赖于被协调实体以及目标任务的当前状态。状态机可以较好地描述协调行为与状态之间的关系[31,32]。

文献[31]在计算机支持的协同工作的背景下，提出了一个扩展的协同描述语言 X-CODL(Extended Collaboration Description Language)，采用基于状态的协调模型与多点协调机制，将协调活动表示为一组相互关联的讨论线程(discussion thread)。讨论线程作为协调活动的执行容器，在控制块中记录特定活动的相关信息，通过规则控制对这些信息的访问。讨论线程包含 6 个状态：new，ready，running，pending，completed 与 aborted，类似于操作系统的线程或进程状态。在讨论线程中定义协调点，引入了 3 个协调操作符：communication，decision 与 prerequisite。但仅仅这 3 个操作符是不能充分表达协调活动的。文献[32] 提出了一个基于多智体规划的协调模型，采用功能依赖图和混合自动机表示多智体规划。

(2)适应性协调

无论是自主计算、多智体系统，还是动态体系结构，建立运行时适应系统的基本方法是建立由松耦合单元构成的系统。通过这些单元的动态协调和重新配置来满足环境和业务变化的需求。在这种松耦合系统中，各部分的关系变得不确定。协调的目的之一就是使这种松耦合系统更具适应性[22,33]。

为了适应业务环境和业务需求的不断变化，文献[34]提出了一个适应性 Agent 模型 AAM(Adaptive Agent Model)。AAM 使用可配置的交互模型来驱动智体的协调行为，交互模型由人类专家使用类 UML 的格式来指定或重新配置。AAM 同时是一种方法学，用来指导业务知识模型的组织层次，以驱动适应性的智体行为，实现持续维护。但 AAM 模型还没有得到具体应用的验证。文献[35] 提出了一个体现情景觉察(situation-awareness)和适应性协调的抽象模型，定义了情景和进程之间的 5 种基本关系：precondition，do，trigger，tell 和 prefer。该模型具有强大的表达能力，可以捕获上下文实例之间的时序关系；服务提供者或开发者可以定义触发、允许或禁止进程执行的情景；用户可以表达服务使用的偏好；通过使用 5 个基本关系，可以对控制结构进行建模。

(3)基于公共知识库的协调

软件实体之间有效协调的关键是参与的实体能达到共享的知识状态，以便能在正确的时间由正确的实体执行正确的操作。同时，要达到共享的知识状态也需要协调。这就需要有一个共享的知识库(如共同词汇表)[2,30,36]。

公共知识库(Common Knowledge Base，CKB)可以采用基于本体的广义知识表达方法[2]。文献[30]开发了一个基于协调本体的原型 Web 服务，采用规则来管理实际的协调过程。OpenKnowledge(OK)项目[37]提出了一种基于声明性交互的知识共享形式，交互规范负责组件之间的消息传递，交互协调提供了组件之间的协调形式。知识工程师通过使用 LCC(Lightweight Coordination Calculus)语言来建立智体交互的知识模型。

4 协调策略

根据协调的特性，我们将软件系统的协调策略从不同角度分为内协调与外协调、运行时协调与设计时协调，等等。

4.1 内协调与外协调

内协调(Endogenous Coordination)指的是协调机制包含在被协调实体内部的代码中，例如 Linda[5]。外协调(Exogenous Coordination)将被协调实体看作一个黑盒，由独立的协调器从外部对实体的行为进行协调，而不需要知道实体的内部结构与行为[9]，例如 Manifold[7]，Reo[8,9,10]，powerJava[21]等。外协调提升了协调模式和实体的重用性，便于维护和进化，但由于只能通过实体的外部可观察行为实施协调，因而不利于细粒度的协调；内协调则相反。

从实现的角度看，鉴于目前的程序设计语言不提供将组件与组件之间的相互连接进行明确分离的机制，因此采用内协调策略，把组件间的协调硬编码到组件内部，实现相对容易；外协调需要把组件间的相互连接和协调作为一个单独的设计问题来处理，需要开发程序设计语言和工具来支持组件以及组件间协调的分离[38]。

4.2 设计时协调与运行时协调

对于设计时协调，被协调的实体及其行为在软件的设计阶段必须是已知的、明确的；而对于运行时协调来说，被协调的实体及其行为在软件的设计阶段可以是未知的。

设计时协调可以采用内协调方式，但不能适应开放系统环境；运行时协调需要动态适应环境与用户需求的变化，灵活性强，但实现难度大。

4.3 直接协调与间接协调

对于直接协调，实体之间采取直接的交互方式；而间接协调则是通过数据中介(如共享数据空间)或控制中介(如协调器)实现实体之间的交互协调。

此外，还有其他的协调策略，如定性协调与定量协调、隐式协调与显式协调、数据驱动的协调与控制驱动的协调等。

面对多样化和动态的企业计算环境，软件系统越来越变得开放、分布、移动和异构。业务环境与业务需求不断变化，老的服务不断淘汰，新的服务不断出现。因此，我们认为，建立复杂软件系统需要采取下列协调策略：

(1)协调与计算分离

在开放系统中，由于系统所包含的组件和资源在设计时是未知的，将协调代码硬编码到系统中的做法是不可行的，需要将协调与计算分离，以支持模块化和标准化的开发方法，改善软件的重用性。

(2)动态协调

面对开放、分布、异构、移动的普适计算环境，要求软件系统具有更高的随需应变能力。例如，由于系统故障或过载，某个服务可能不可用或者不能提供期望的服务质量，此时应在多种服务中作出选择；服务可能动态加入或离开系统；当业务需求改变时，需要在运行时建立新的业务流程来满足用户新的需求。因此，需要在运行时通过若干松耦合单元的动态协调和重新配置来适应业务环境和业务需求的变化。

(3)自主协调

对于大规模、动态、开放的软件系统，进程和资源在系统设计时是不可能完全已知的，而且可能不断变化。因此，不能期望每个运行单元获得全局知识，需要通过动态推理自主地解决协调问题[36]。

结束语 面对大型软件系统的复杂性及其应用环境的多样性，要求软件系统能动态适应业务环境和需求的持续改变。因此，系统各部分之间的有效协调成为系统开发过程中极为重要的问题。目前关于协调技术的研究仍存在下列问题：

(1)缺乏普适性的协调语言和工具来表示和处理协调问题。目前为止，有关协调技术的研究主要集中在专用协调语言和环境的开发，每种新的语言提出一套支持特定范例的协调抽象集。然而，协调问题并不总是符合特定范例的。需要开发支持协调抽象的程序设计语言和工具，提供强有力的手段来明确表达实体之间的相互依赖和相关的协调过程[39]。

(2)还没有有效的实现机制来应对自主系统运行时的动态协调。对于完全已知的静态系统，协调问题还比较容易解决。主要困难在于软件实体可能是不可预知或动态的。尽管有很多协调模型是针对自主系统运行时的动态适应性协调而提出的，但由于自主系统的黑盒特性，而且在设计时对系统的未来行为难以预测，导致很多模型只是理想化的理论模型，难以实现。

(3)协调的智能水平不高。目前的协调模型，即使是针对运行时动态协调的适应性模型，大多是从系统设计和实现的方法学角度来讨论的，主要致力于计算和协调方面的分离。然而，如何真正实现协调行为的动态适应性，实现智能协调，却是其中的关键。

Internet 应用的开发对传统分布式软件开发方法提出了新的问题和需求。一方面，Internet 环境分散、开放、异构、松耦合，只能得到部分知识，各部分的关系不确定；另一方面，期望应用系统能适应环境与业务需求的变化，能处理非预期的情况。面对分布、自主的复杂应用，有许多需要进一步研究的问题。例如，如何表示和管理自主软件实体之间的协调依赖；如何保证协调能产生预期的行为；当改变或增加新的协调约束时，如何避免与当前行为不发生冲突，等等。此外，人类个体以及人类社会具有强有力的机制来协调复杂行为，借鉴人类的协调机制来研究拟人协调技术，必然会丰富现有的协调模型。

参 考 文 献

[1] 涂序彦. 论协调[J]. 科学学与科学技术管理，1981，5：17-20

[2] 涂序彦，王枞，郭燕慧. 大系统控制论(修订版)[M]. 北京：北京邮电大学出版社，2005

[3] Malone T，Crowston K. The interdisciplinary study of coordination[J]. ACM Computing Surveys，1994，26(1)：87-119

[4] Coates G . Agent co - ordination aided distributed computational engineering design[J]. Expert Systems with Applications，2006，31：776-786

[5] Carriero N，Gelernter D. LINDA in context[J]. Communications of the ACM，1989，32：444-458

[6] Cabri G，Leonardi L，Zambonelli F . MARS：A Programmable Coordination Architecture for Mobile Agents[J]. IEEE Internet Computing，2000，4：26-35

[7] Arbab F，Herman I，Spilling P. An Overview of Manifold and Its Implementation[J]. Concurrency-Practice and Experience，1993，5(1)：23-70

[8] Arbab F. Abstract Behavior Types: A foundation model for components and their composition [J]. Science of Computer Programming, 2005, 55: 3-52

[9] Arbab F. Coordination for Component Composition [J]. Electronic Notes in Theoretical Computer Science, 2006, 160: 15-40

[10] Arbab F, Chothia T, Meng S, et al. Component Connectors with QoS Guarantees [C] // Proceedings of 9th International Conference on Coordination Models and Languages. COORDINATION 2007. June 2007 LNCS 4467: 286-304

[11] Leymann F, Pottinger S. Rethinking the Coordination Models of WS- Coordination and WS-CF [C] // Proceedings of the 3rd European Conference on Web Services, ECOWS'05. Nov. 2005: 160-169

[12] Picco G, Murphy A, Roman G. Lime: Linda Meets Mobility [C] // Proceedings of 21th IEEE International Conference on Software Engineering, ICSE'99. May 1999: 368-377

[13] Tu Xuyan , Tang Tao . Intelligent Autonomous Decentralized System (IADS) [C] // Proceedings of 2nd International Workshop on Autonomous Decentralized System, IWADS'02. Nov. 2002: 10-15

[14] Wyckoff P, McLaughry S W, Lehman T J, et al. TSpaces [J]. IBM Systems Journal, 1998, 37(3)

[15] Menezes R, Omicin A, Viroli M. On the Semantics of Coordination Models for Distributed Systems: The LogOp Case Study [J]. Electronic Notes in Theoretical Computer Science, 2004, 97: 97-124

[16] Kuhn E. Space Based Computing [EB/OL]. http: // www. complang. tuwien. ac. at /eva/

[17] Proenca J, Clarke D. Coordination Models Orc and Reo Compared [J]. Electronic Notes in Theoretical Computer Science, 2008, 194: 57-76

[18] Juan G S, Arbab F, et al. A Channel-based Coordination Model for Components [J]. Electronic Notes in Theoretical Computer Science 68, 2003, 3. URL: http: // www. elsevier. nl/locate/entcs/volume68. html

[19] Kwiat K , Ren S P . A Coordination Model for Improving Software System Attack-tolerance and Survivability in Open Hostile Environments [C] // Proceedings of the IEEE International Conference on Sensor Networks. Ubiquitous and Trustworthy Computing, SUTC'06. June 2006: 394-406

[20] Ren S P, Yu Y, Chen N, et al. Actors, Roles and Coordinators - A Coordination Model for Open Distributed and Embedded Systems [C] // Proceedings of 8th International Conference on Coordination Models and Languages, COORDINATION 2006. June 2006, LNCS 4038: 247-265

[21] Baldoni M, Boella G. Roles as a Coordination Construct: Introducing powerJava [J]. Electronic Notes in Theoretical Computer Science, 2006, 150: 9-29

[22] Colman A, Han J. Using role-based coordination to achieve software adaptability [J]. Science of Computer Programming, 2007, 64: 223-245

[23] Busi N, Zavattaro G. A Process Algebraic View of Coordination [J]. Electronic Notes in Theoretical Computer Science, 2006, 162: 141-145

[24] Fuentes L, Sanchez P. Aspect-oriented Coordination [J]. Electronic Notes in Theoretical Computer Science, 2007, 189: 87-103

[25] Cuesta C, Romay M, et al. Coordination as an architectural aspect [J]. Electronic Notes in Theoretical Computer Science, 2006, 154: 25-41

[26] Viroli M, Omicini A. Coordination as a Service: Ontological and Formal Foundation [J]. Electronic Notes in Theoretical Computer Science, 2003, 68: 457-482

[27] Microsoft, IBM, Hitachi, et al. The Specification of WS-Coordination Version 1. 0 [EB/OL]. http: // www-128. ibm. com/developerworks/ library/specification/ws-tx/, 2005

[28] Arjuna, Fujitsu, IONA, et al. Web Services Coordination Framework Version 1. 0 [EB/OL]. http: // developers. sun. com/techtopics/ webservices/wscaf/wscf. pdf, 2003

[29] Jung J Y, Park J, Han S K, et al. An ECA-based framework for decentralized coordination of ubiquitous web services [J]. Information and Software Technology, 2007, 49: 1141-1161

[30] Moyaux T , Smith B , Paurobally S . Towards Service - oriented Ontology-based Coordination [C] // Proceedings of IEEE International Conference on Web Services, ICWS'06. Sept. 2006: 265-274

[31] Zhang J. Extended Collaboration Description Language (X-CODL) [C] // Proceedings of the 10th IEEE International Enterprise Distributed Object Computing Conference, EDOC'06. Oct. 2006: 56-66

[32] Marc F, Cartault I. Multi-Agent Planning as a Coordination Model for Self-organized Systems [C] // Proceedings of the IEEE/WIC International Conference on Intelligent Agent Technology, IAT'03. Oct. 2003: 218-224

[33] Cubo J, Salaün G, Cámara J, et al. Context-based Adaptation of Component Behavioural Interfaces [C] // Proceedings of 9th International Conference on Coordination Models and Languages, COORDINATION 2007. June 2007, LNCS 4467: 305-323

[34] Xiao L, Robertson D, et al. Adaptive Agent Model: an Agent Interaction and Computation Model [C] // Proceedings of the 31st Annual International Computer Software and Applications Conference, COMPSAC'07. July 2007: 153-158

[35] Yau S S, Huang D, Gong H, et al. Situation-awareness for Adaptive Coordination in Service-based Systems [C] // Proceedings of the 29th Annual International Computer Software and Applications Conference, COMPSAC'05. July 2005: 107-112

[36] Vecht B, Dignum F, et al. A Dynamic Coordination Mechanism Using Adjustable Autonomy [C] // COIN 2007 Workshops. LNAI 4870. 2008: 83-96

[37] Robertson D. Open Knowledge: Semantic Webs through Peer-to-Peer Interaction [EB/OL]. http: // www. openk. org/

[38] Bortenschlager M, Reich S. A Generic Coordination Architecture as an Enabler for Mobile Collaborative Applications [C] // Proceedings of the 15th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, WETICE'06. June 2006: 125-130

[39] Guo J. A Coordination Framework for Software Component Based Development [C] // Proceedings of the 30th Annual International Computer Software and Applications Conference, COMPSAC'06. Sept. 2006: 291-298