基于角色的多 Agent 系统代码自动生成技术的研究

李 阳 闫 琪 齐治昌
(国防科学技术大学 长沙410073)

摘 要 多 Agent 系统代码自动生成技术是为了填补多 Agent 系统方法在实现阶段的空白。论文研究了基于角色的多 Agent 系统及其建模工具,并在此基础上提出一种基于角色的多 Agent 系统的代码自动生成技术,探讨了该技术的独到之处,同时展望该技术今后的研究方向。

关键词 Agent,MAS,RoMAS,角色,ArgoUML,代码生成,数据绑定

The Research on Automatical Code Generating Technique for RoMAS

LI Yang YAN Qi QI Zhi-Chang
(National University of Defense Technology,Changsha410073)

Abstract Automatical Code Generating Technique of MAS (Multi-Agent System) is for filling up the blank field in MAS implementation phase. This paper studies MAS and its related tools,and presents an automatical Code Generating Technique for RoMAS. We also discuss the particular aspects and the further research directions of this technique.

Keywords Agent,MAS,RoMAS,Role,ArgoUML,Code generation,Data banding

1 引言

目前,面向 Agent 的软件工程(AOSE)、多 Agent 系统(MAS)在软件工程领域的研究得到许多的关注。在面向 Agent 方法研究中,软件工程专家引入了角色概念,角色在面向 Agent 的软件工程领域是一个较新的也是一个很有研究价值的概念[1],本文正是在基于角色的多 Agent 系统开发方法(RoMAS)中提出了如何实现代码框架自动生成的一些构思。

本文旨在介绍如何基于角色的多 Agent 系统开发方法实现代码框架自动生成的思路,所以首先必须明晰一些基本的概念,以及了解相关的研究工作。在这里我们将简单介绍什么是多 Agent 系统(MAS),什么是基于角色的多 Agent 系统开发方法(RoMAS)以及基于角色的多 Agent 系统开发方法的研究现状以及代码框架的自动生成处在该方法的位置。

文章在接下来的篇幅里探讨基于角色的多 Agent 系统开发方法所使用到的开源工程设计工具 ArgoUML,代码生成的主要技术和如何扩充该工具利用其实现代码框架的自动生成。最后给出技术的独特性以及该方向的趋势。

2 RoMAS 的相关概念

2.1 Agent

Agent 的概念在很多相关研究领域中都有定义,其最基本的特点表现为智能实体。普遍认可的一个 Agent 定义是:Agent 是设计用来完成某类任务的,Agent 是在特定环境下为实现其被赋予的目标,拥有自主性、社会性、反应性和主动性等特征的有生命周期的计算实体[2]。

2.2 多 Agent 系统(MAS)

MAS 是由多个自主的、相互交互的实体构成的系统,这些实体可以通过复杂的协议进行通信,相互协作从而完成共同的目标。MAS 内的 Agent 一般具有复杂的组件,在分布式的开放终端环境下并发地活动,其主要优势在于能把一个复杂的系统分解成一些相互交互的子系统,同时重用先前定义的子系统和构件[3]。

2.3 角色

角色简单来讲就是一个个体所从事的社会行为,就好比同一个人在社会中不同环境内就具有不同角色,可以同时是父亲儿子、老师学生等等角色。在基于角色的多 Agent 系统开发方法中把角色定义为对象行为的抽象,包括该对象拥有的交互的子集以及与交互何时发生相关的约束。换句话说角色是对实体自身行为以及实体参与社会交互时采取的行为模式的规约,它的构成要素包括:属性(Attribute),目标(Goal),约束(Constraint),内部活动(Internal-Activity),服务(Service)和协议(Protocol)。

2.4 基于角色的多 Agent 系统开发方法(RoMAS)

软件开发的本质是在不同抽象层次上的系统建模过程,从需求模型,到设计模型,以及实现阶段的实现模型,在这里我们旨在阐明 RoMAS 的概念模型。基于角色的 MAS 概念模型是现有研究中一类重要模型,角色和 Agent 是两个相区别但是又相关的概念,角色更多地强调其职能的划分和角色之间的协作,Agent 则重点强调其自治性和智能行为。角色代表了一定的目标和职责,角色之间的交互刻画了社会性行为。角色的执行依赖于 Agent,Agent 是系统中进行感知和动作的实体,角色的执行必须通过它所绑定的 Agent 的基本动作来实现。所以 RoMAS 采用了对角色建模的过程,角色模型代表了一种交互与合作模式,它可以重用,角色可以一种动态的方式进行分配;同时角色模型鼓励分解,从而将复杂的 Agent 行为分解为角色,设计者可以从简单的角色入手,然后通过合成得到复杂的 Agent。由此可以看出 Agent 是角色的实例,角色存在于系统分析阶段,Agent 存在于系统设计阶段,角色通过实例化得到它的执行载体(即 Agent),实例化过程就是 Agent 绑定角色的过程。这里的绑定,即指某 Agent 获得某角

齐治昌 教授,博士导师。李 阳 硕士研究生,主要从事软件工程 AO 方法学研究。闫 琪 博士,主要从事软件工程 AO 方法学研究。

色的所有信息并将其映射为 Agent 自身的属性与方法。

2.5 什么是 RoMAS 的代码自动生成

当对 RoMAS 的概念模型有了基本了解后，研究者可能更加关注一些具体的开发工作。这里主要介绍基于 RoMAS 方法的代码自动生成。这项工作的思路来自于对面向对象的软件工程方法的借鉴，面向对象的软件工程是一种非常成熟的开发方法，从方法学，模型到具体的实现，建模语言，开发工具以及面向对象的语言等构成目前主流的软件开发元素。既然我们希望发展前途更加诱人的 RoMAS 方法，自然需要对应面向对象的方法在 RoMAS 中从理论到实践的各个相关领域作许多研究，其中许多面向对象的建模工具都具备代码自动生成的功能，本文将借助 RoMAS 使用的建摸开发工具 Argouml，探讨如何实现 RoMAS 的代码自动生成。

2.6 基于角色的多 Agent 系统开发方法的研究现状与代码生成所处地位

自从研究者对面向 Agent 的软件工程进行大量研究以来，就有很多相关工作也取得很大进展。其中多数工作围绕方法学的讨论[1,4,5]，模型的构造[6]，与面向对象方法的比较[7]等等，这些是对面向 Agent 的软件工程发展的必要工作。有了这样一个很好的基础环境，基于这些设计阶段方法学的研究，延伸出了许多实现阶段的探讨开发工作，比如针对 MAS 的建模语言[8]，Agent 工具[9]甚至设计模式。

RoMAS 是 MAS 开发方法中非常实用的一种，受到诸多大学和研究所以及企业的重视。目前 RoMAS 已经完成了概念模型的设计，这在前面的 RoMAS 的概念介绍中已经提到，同时也有对体系结构和工具的探讨，过程模型和产品模型的设计和生成原型系统的方法的设计。可以看出 RoMAS 在设计阶段已经搭建了一个比较完善的体系，为了验证其先进性并把研究成果运用于实际的工程开发，必须要有 Agent 代码的生成。Agent 代表了对系统的一种抽象，这种抽象不同于对象。正如任何模块化语言都可以用来写面向对象代码，因此也能用面向对象语言开发 Agent 代码并不奇怪，所以我们还不能完全使用一种纯 Agent 代码来做实现，尽管已经有了许多 Agent 建模语言和 Agent 代码，但是目前还不够完善，所以从语言进化角度来看由面向对象过渡到面向 Agent 还需要一个过程。

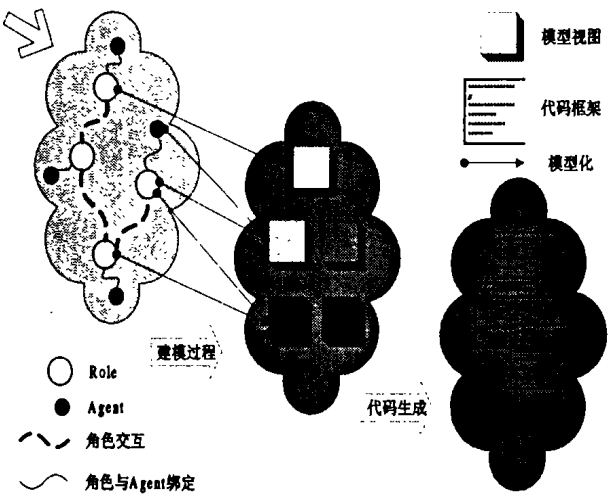

图1

RoMAS 方法也是遵循软件工程开发方式，因此其开发过程与 OO 方法是有对应的。首先对现实世界的问题模型进行描述，从中确定对问题解决所需要达到的目标（在 RoMAS 反映成 Goal）和需要获得的关键元素（比如问题中的角色，环境等）；然后对问题模型用 RoMAS 工具 ArgoUML 进行建模，刻画角色间的关系，并具体用 Agent 绑定角色，赋予 Agent 智能特性（包括 Agent 属性，约束，服务，协议和转换等等）；最后对 Agent 静态动态图（关系图，时序图，协议图）实现高级转换生成基本的 Agent 的代码框架。而图1正是对上述过程的描述，也清楚显示了代码生成功能所处在 RoMAS 开发阶段的位置。

3 RoMAS 建模工具

任何一套开发方法运用到实践中离不开具体工具的支持，通常需要设计或选取适当的开发环境与建模工具用于支持所提出的建模方法，支持 MAS 的工具目前有很多种，比如 uAgent Building Shell，Agent Tcl，uJava Agent Template 等等不一而足，但是因为面向 Agent 的 MAS 仍主要处在研究阶段同时这些工具很多是针对专门解决某些特定领域的问题而设计的，并没有一个基于角色 MAS 的通用工具。RoMAS 目前选用对开源软件 ArgoUML[10]进行扩展，设计了一套支持 RoMAS 方法的建模工具 RoArgoUML 实现代码的自动生成。

无论是 AO 还是 OO 方法开发软件，重要的是在思维认识上进行创新的工作。虽然设计者必须动手进行设计，但是，实际上最重要的工作是进行构思创新而不是数据录入。如果设计者提高了他们的思维创造能力，自然就能够设计出更好的软件。代码的自动生成正是着眼于这个思路，希望能提高利用 AO 方法开发软件效率。当前的面向对象的 CASE 工具几乎都提供自动化的图形用户界面（GUI）来减少设计者的手工输入量，设计成果以一些图形表示，然后再将这些图型转换成代码，形成最终的成果。ArgoUML 被认为是一种面向对象的分析设计工具，和当前用于软件建模的 CASE 工具具有许多相似之处。比如，ArgoUML 与其它商用 CASE 工具一样提供自动化的图形设计界面。ArgoUML 虽然还不能提供流行的 CASE 工具的许多功能，但它仍具备一些优势（这里仅介绍与本文主旨相关的优势）：

（1）ArgoUML 是一个开放源代码的软件。这对开发一个支持新的方法学的工具来说极具诱惑，开发者可以方便交流而不用考虑版权问题。

（2）ArgoUML 是纯 Java 程序的产品。ArgoUML 可以在各种平台上运行。

（3）ArgoUML 广泛地支持开放的标准——UML，XMI，OCL 等。从这方面来看，ArgoUML 仍然领先于许多商业工具。实际上，ArgoUML 很大一部分都是从 UML 规范自动生成的。另外，ArgoUML 希望支持对象约束语言（OCL）和 XML 模型交换格式（XMI），而其他一些工具并不支持。采用开放的 XMI 文件格式这个优势也正是为我们对在其上面扩充代码自动生成充满信心，这在后面会介绍到。

既然谈到 CASE 工具，就很自然拿 ArgoUML 和其他面向对象的 CASE 工具进行类比，这里我们仅从源码是否公开，功能特点和所支持方法学等与本文相关的这些方面简单比较了 ArgoUML 和 Rational Rose，表1显示了两者比较的结果。

表1

	ArgoUML	Rational Rose
源码是否公开	是	否
是否具备代码生成	否	是
功能	简单	复杂
对方法学的支持	可扩充支持 RoMAS	仅支持面向对象方法

4 在RoArgoUML上的扩充步骤和主要代码生成技术

4.1 什么是XMI

在介绍ArgoUML第三点性质时曾提到ArgoUML广泛地支持开放的标准—UML，XMI。这里需要对XMI有个了解，简单地来说，XMI(XML Metadata Interchange)是XML的一种应用，它是为了在程序员和其它程序员之间交换元数据。XMI也可以帮助程序员应用UML语言解决不同语言编程的问题，XMI比较有效地定义了不同应用，不同平台上元数据交换的问题。

4.2 基本思路

在ArgoUML，XMI和XML之间的联系的基础上，可以形成如下基本思路：若要实现代码自动生成，首先可以在ArgoUML上添加功能菜单(利用其开源特性)，对使用ArgoUML所画的工程图进行转换，因为工程图的保存形式正是以Argo，XMI和Pgml等文件格式组成，而XMI文件又很好地刻画出RoMAS的Agent和其目标、约束、内部活动以及Agent之间的协议。最后利用代码生成技术对XMI文件进行扫描，并分析之，最终转换成基本代码框架。

4.3 UML模型到XML的转换

我们使用的ArgoUML绘制的工程图仍是标准UML，UML的目的在于成为人类工程学，但是UML本身不适宜于完成计算机化这一过程。

通过以文本方式提供"序列化"UML数据的构件，XML消除了二者之间的部分差距。"XML元数据交换(XMI)"将XML应用到诸如UML的抽象系统。XMI方法捕捉和表达UML表达的关系，而抛弃特定UML图的大多数可视细节。这种将事物划分成必不可少的内容与可有可无的形式的做法增强了UML的可管理性。要实现思路所呈现方式，可见从UML模型自动派生XML模式是很重要的。XMI是一种标准格式，可用于把UML模型派生成XML模式[11]。现通过一个RoMAS简单的Agent例子(这也是引用了RoboCup案例)来说明如何由一个Agent实例的UML图进行模式改变的。图2是现实世界一个FootballPlayer角色在RoMAS中绑定的一个具体Agent和相对应的Xml文档描述，在文章接下来部分都将使用这个例子。

4.4 数据绑定和代码生成

最后在技术上需要考虑的是完成对XML模式文档的解析识别，生成基本代码框架，这里需要使用到的是一种数据绑定技术。数据绑定提供了一种简单而直接的方法，以在Java平台应用程序中使用XML，应用程序可以在很大程度上忽略XML文档的实际结构，而直接使用那些文档的数据内容。虽然这种方法不能适合于所有应用程序，但ArgoUML正好是XML用于数据交换的一类应用程序，所以是比较理想的。

为了简化编写处理XML的Java程序，已经建立了多种编程接口，可以方便帮助我们对ArgoUML的XML文档进行解析介入。这里介绍三种在ArgoUML库里面可非常容易找到的几个简单接口：(1)Document Object Model (DOM，文档对象模型)，(2)Simple API for XML (SAX)，(3)Java API for XML Processing (JAXP)。

(1)用于读取和操作XML文件的标准是文档对象模型(Document Object Model，DOM)，最普通的XML处理工作是解析XML文档。解析包括读取XML文档并确定其结构和内容。当使用DOM解析器解析一个XML文档时，得到一个层次化的数据结构(DOM树)，它表示解析器在XML文档中发现的所有内容。

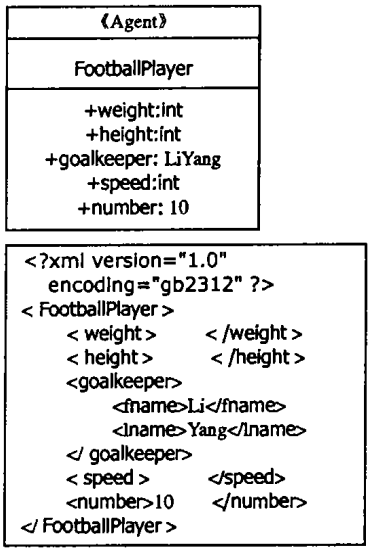

图2

(2)一种替代DOM技术就是Simple API for XML，或称为SAX。SAX允许在读取文档时处理它，从而不必等待整个文档被存储之后才采取操作。

SAX和DOM不是相互排斥的，可以结合使用，其次从应用程序目的来看，Argouml生成的XML文档多数情况下无须修改数据，数据容量较大，所以较多选择SAX。

(3)JAXP专用于处理XML的Java API (Java API for XML Processing)，专门提供XML文档解析的Java接口，但是将其称为抽象层更准确。它不提供处理XML的新方式，不补充SAX或DOM，也不向Java和XML处理提供新功能。它只是使通过DOM和SAX处理一些困难任务更容易。

数据绑定和这些接口的不同在于(用DOM举例)，都在内存中建立文档的表示，都需要在内部表示和标准文本XML之间双向转换。两者的区别在于文档模型尽可能保持XML结构，而数据绑定只关心应用程序所使用的文档数据。

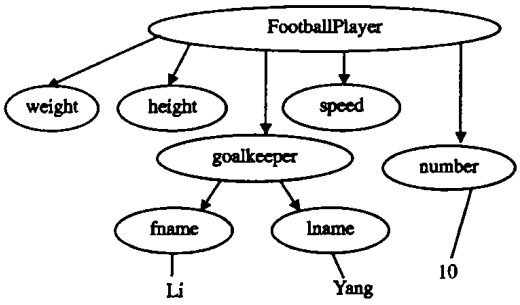

图3 文档模型视图

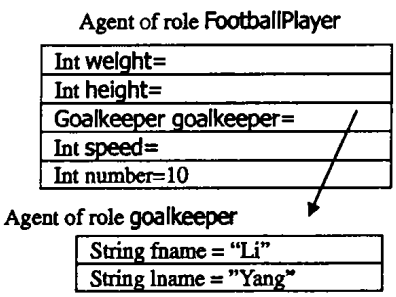

图4 文档的数据绑定视图

从图3可以看出文档成分(图2的DOM视图)——在这个

例子中只有元素和文本节点，通过反映原始 XML 文档的结构连接在一起。形成的节点树很容易和原始文档联系，但要解释树中表示的实际数据就不那么容易了。如果应用程序使用 XML 文档模型方法，就需要处理这种类型的树。

图4表示同一文档的数据绑定视图。在这里，转换过程几乎隐藏了原始 XML 文档的所有结构，与文档模型方法相比，抽掉了许多文档细节，数据绑定通常需要的内存更少。因为只通过两个 Agent（文档模型方法使用了11个单独的元素），更容易看清楚真正的数据，也更快更容易访问这些数据。

4.5 数据绑定框架

用 XML 文档文法生成 Java 语言代码有多种 XML 数据绑定框架：比如 JAXB[12]、Castor、JBind、Quick 和 Zeus 等等。JAXB、Castor 和 JBind 都提供了根据 XML 文档的 Schema 描述生成代码，而 Quick 和 Zeus 根据 DTD 描述生成代码。尽管 RoMAS 是面向 Agent 的开发方法，但我们在前面已经说明需要继承大量面向对象已有的技术，包括代码生成技术。这里主要介绍 RoMAS 实际使用的 JAXB。

用于 XML 绑定的 JavaAPI（Java API for XML Binding，JAXB）是一个处于不断发展中的 Java 平台数据绑定标准，其目的是定义一种方法，生成与 XML 结构相链接的 Java 语言代码。

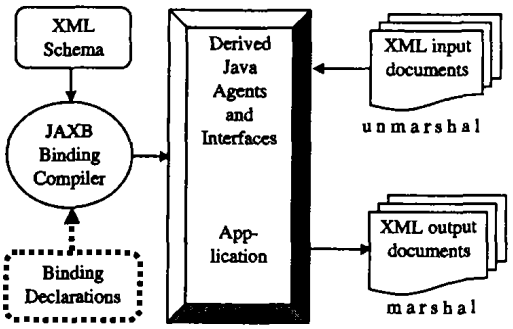

图5 JAXB 体系结构

其处理过程如下：首先根据应用程序所要操作的 XML 数据格式，撰写相应的 XML DTD 或 Schema，使用与 JAXB 一起提供的模式编译器从文档类型定义 DTD 或描述信息结构的模型 Schema 产生一系列相关的 Java Agent 和 Interface 源文件。我们这里借助 Schema 描述来完成 JAXB 的剩余工作。JAXB 直接根据 Schema 文档描述生成为文档所定义的元素类型和用法相匹配的角色目标的层次结构。JAXB 所提供的绑定编译器 xjc 是一个命令行工具，它将 Schema 文档作为输入，将文件生成到指定的输出包和目标目录。最后，再使用 JAXB API 得到存储在有效 XML 文件中的数据，并将它们转换为 Java 表示的实例 Agent。

通过使用绑定声明，JAXB 规范定义了一些方法来定制生成数据绑定的一些方面。包括：

(1)用于控制所生成 Agent 的名称和目标、约束的选项；

(2)指定由绑定所使用的现有实现 Agent 的内部活动、服务以及角色所遵循的协议，行为激发规则和 Agent 角色转换；

(3)允许(有限地)控制验证处理和用于编组和数据分解的序列化器/反序列化器(serializer/deserializer)的选项。

尽管 JAXB 可以提供 Agent 的代码框架，但是目前整个面向 Agent 方法仍处于研究阶段，所以很多 Agent 的优势在生成的代码体系中很难体现出来，比如对环境的感知，角色的演变等等，而生成的代码仅是对 Agent 关系图、接口图的体现，所以本文主旨是提供一种途径解决目前我们研究 RoMAS 的代码生成。

结论与展望 本文首先主要介绍了 RoMAS 方法学的基本概念模型和开发工具 ArgoUML 的性质，然后指出代码生成在整个 RoMAS 开发方法中所处地位，根据开发工具 ArgoUML 的性质提出一种适合 ArgoUML 的扩充思路：对 ArgoUML 工程图的 XMI 文档进行模式转换获得 XML 模式，并对 XML 文档进行解析，获取基本 Agent 以及相关名称和目标、约束等信息，最后利用数据绑定技术生成基本 Agent 代码框架和 Agent 接口框架。

在该思路指导下，可对 ArgoUML 进行初步改进，加入代码生成的菜单功能。该功能可将所建的基于 Agent 方法的工程模型用 Agent 关系图所描述，目前的 ArgoUML 在其 lib 内提供了 jaxp 文件包，内含有 DOM 和 SAX 这两个前面介绍的解析器，根据 Agent 关系图对应的 XMI 文档可选择生成单个 Agent 框架代码或是一份 XMI 所对应的全部的 Agent 框架代码，但是由于目前 MAS 整体研究环境的限制，该代码框架仍仅以 Java 方式呈现。

这种局限性导致的另一主要问题在于目前对 Agent 的一些优势特点，比如自主性，这包括对环境的感知，角色的演变等等很难从代码自动生成中体现出来，所以目前 RoMAS 代码框架的自动生成区别于基于 OO 方法的代码生成在以下几个方面：

第一，RoMAS 引入角色就在于角色能很好地自主刻画现实世界问题，在这里 RoMAS 的角色概念可从生成的代码框架所体现出来。

第二，Agent 之间的关系，及其自身的属性，协议等静态概念也可以从中体现。

本方法不一定会成为将来唯一的面向 Agent 开发方法代码自动生成方式，但是由于从 OO 方法到 AO 方法需要一个过渡，在这个阶段本方法既借鉴了 OO 的一些技术但同时又发展了 RoMAS 的空白领域，这也是以前所不曾有过的研究，个人认为对未来此类开发的重点研究方向将主要集中在继续开展本研究尚不能做到的一些方面，比如所生成代码的类别范围将更为宽广，更进一步的研究包括诸如角色间的一些动态行为怎么体现在 Agent 代码框架之间，这包括角色间协议的交互，Agent 转换，对环境的感知等等，以及对新型的基于 Agent 的代码结构的研究，可以预见这些都将会是非常大的课题，毕竟在目前这样的阶段暂还不能触及如此深远。

参 考 文 献

1 Iglesias C A, Garijo M, Gonz'alez J C. A Survey of Agent-Oriented Methodologies
2 Wooldridge M, Jennings N R, Kinny D. The Gaia Methodology for Agent-Oriented Analysis and Design
3 Wang A I, Conradi R, Liu Chunnian. A Multi-Agent Architecture for Cooperative Software Engineering
4 Wooldridge M, Jennings N R, Kinny D. A Methodology for Agent-Oriented Analysis and Design
5 Wood M F, DeLoach S A. An Overview of the Multiagent Systems Engineering Methodology
6 Yan Qi, Mao XinJun, Zhu Hong, Qi ZhiChang. Modelling Multi-Agent Systems with Soft Genes, Roles and Agents
7 Odell J. Objects and Agents, How do they differ?
8 da Silva Carlos V T, de Lucena J P. MAS-ML, A Multi-Agent System Modeling Language
9 DeLoach S A. Analysis and Design using MaSE and agentTool
10 argouml. tigris. org
11 Benoit Marchal Working XML: UML, XMI, and code generation
12 Sosnoski D M. XML in Java: Data binding, Code generation approaches - JAXB and more