复杂自适应多 Agent 系统的模型驱动开发方法

曹 江 毛新军 王怀民 卢锡城
(国防科技大学计算机学院 长沙 410073)

摘 要 自适应系统是一类复杂系统,如何有效地支持此类系统的工程化开发,一直是软件工程领域的关注焦点。提出了一个基于 Agent 的模型驱动软件开发方法,试图将主流软件工程中的 MDA 技术与软件 Agent 技术相结合,从而为高效、高质量地开发复杂自适应多 Agent 系统提供方法学指导。该方法将基于组织抽象和 ODAM+ 方法学所建立起来的模型视为平台无关模型,将基于 SADE 平台的实现模型视为平台相关模型,通过建立这两个不同抽象层次元模型间的映射关系,来实现从平台无关模型到平台相关模型以及最终代码框架的转换。介绍了集成 MDA 和 Agent 技术的软件方法学 ODAM+,阐述了复杂自适应系统模型驱动开发的一组关键技术,包括不同层次的元模型以及它们之间的映射关系、模型转换规则和模型转换算法,最后分析了相应的支撑软件工具和应用验证情况。

关键词 自适应系统,Agent,组织抽象,模型驱动架构

Model-driven Development Methodology for Complex Self-adaptive Multi-agent Systems

CAO Jiang MAO Xin-jun WANG Huai-min LU Xi-cheng
(School of Computer, National University of Defense Technology, Changsha 410073, China)

Abstract Self-adaptive system is a complex system. How to develop such systems in an effective engineering way has become an important topic in the software engineering community. This paper presented an agent-based model-driven methodology that attempts to integrate MDA and agent technology to support the development of complex self-adaptive systems. Our approach takes organization-based models created by ODAM+ as platform-independent models, and models based on SADE platform as platform-specific models. The transformation between these two kinds of models is performed by establishing the mapping relationships of meta-models in these two different levels. This paper introduced in details the software methodology ODAM+ and the involving technologies such as meta-models and their mapping relationships between platform-independent models and platform-specific models, transformation rules and algorithms, lastly the software toolkits supporting the methodology was introduced and a case was studied.

Keywords Self-adaptive system, Agent, Organization abstraction, Model-driven architecture

1 引言

近年来,随着计算机技术和互联网技术的不断发展及其应用的持续扩大和深入,基于网络的计算机软件及其特征正发生着深刻的变化。这种变化集中体现为以下 3 个方面:(1)环境复杂化。软件系统所驻留的环境具有开放、动态、不可控、无法事先确定等特点,如 Internet、战场环境、社会环境、物理环境等,系统与环境之间的关系变得更加密切并且环境变化将对系统产生实质性的影响。(2)系统复杂化。软件系统驻留在环境中,需要不断地调整自身,以适应外部环境和内部状态的变化,从而展示出更好的灵活性和动态性等,以实现系统的设计目标。(3)规模超大化。越来越多的系统呈现出系统或者超大规模系统的特点[1]。适应性成为系统应对环境变化以及规模持续增长,提高系统可靠性、可信性、可配置性、健壮性、友好性、多样性等的一种重要方式和手段[2,3]。所谓自适应系统是指能够根据其操作环境变化而动态调整自身行为的一类复杂系统[4]。显然,由于环境的开放性、系统的动态性等特点,此类软件系统的开发、部署和维护对软件工程方法提出了新的要求,也使得现有以对象技术为核心的主流软件工程面临一系列挑战。

越来越多的研究领域关注自适应系统的开发和应用,如自治计算、移动和 Ad-hoc 网络、传感网络、多 Agent 系统、智能机器人等。近年来,自适应系统软件工程已成为一个非常活跃的研究方向。人们通常交叉和借鉴多学科的思想和方法来开展该方向的研究,如基于控制论的自适应反馈循环、基于人工智能的自适应实现算法等。至今,有关自适应系统的软件工程研究涉及多个不同方面的内容,包括需求工程、软件体系结构、中间件、基于构件开发、程序设计语言等。现有的研究具有以下两方面的特点:(1)大多关注自适应系统开发的某些方面,如需求、建模、构造和实现,这些工作通常是相对独立

到稿日期:2011-03-16 返修日期:2011-05-25 本文受国家自然科学基金(61070034),教育部博士点基金(20094307110007)和国家 973 课题(2011CB302601)资助。

曹 江(1961—),男,硕士,主要研究方向为软件工程;毛新军(1970—),男,博士,教授,主要研究方向为软件工程;王怀民(1962—),男,博士,教授,主要研究方向为分布计算技术;卢锡城(1946—),男,教授,主要研究方向为高性能计算、分布计算技术。

的，未能从工程化的角度为自适应系统的开发提供系统的支持[3]。(2)更多关注如何将这类系统开发出来，而很少考虑此类系统开发的效率和质量。

由于自适应系统自身的复杂性，如何为此类系统的开发提供系统的方法学指导，并采用有效的技术手段来提高系统开发的效率和质量，成为当前自适应系统软件工程面临的一个重要挑战。为了迎接这一挑战，需要寻求新的、统一的抽象来规约、分析、设计和实现复杂自适应系统，同时借鉴软件工程领域成功的技术和经验来有效、高质量地推动自适应系统的开发。

近年出现的面向 Agent 软件工程(AOSE, Agent-Oriented Software Engineering)以自主 Agent 作为基本抽象和模块单元来描述、分析、设计和构造软件系统，代表了一种新的软件开发范型。由于 Agent 的环境驻留性和行为自主性能够有效应对环境和系统复杂化所带来的诸多挑战，因此这一技术近年来引起了人们的关注和重视[5]。由于适应性的基础和前提是环境敏感性和行为自主性，因此借助于 Agent 抽象和技术，可以自然对自适应系统中的实体及其自适应特征进行抽象、建模和分析，对自适应系统的构造和实现提供新颖的技术手段。越来越多的人认识到 Agent 技术在支持复杂系统(如自适应系统等)开发方面的潜力，但是至今这一潜力未能得到充分的展示。导致这种状况的原因是多方面的[6]，其中之一是当前面向 Agent 软件工程未能充分借鉴主流软件工程中的成功方法、经验和实践，并针对复杂系统的特点来开展针对性的研究与应用。

模型驱动开发区分平台无关模型(PIM, Platform Independent Model)和平台相关模型(PSM, Platform Specific Model)，通过建立 PIM 与 PSM 间的映射和转换来实现软件开发，提高软件开发的效率和质量，并在面向对象软件工程中得到成功的实践和应用。将 MDA 思想引入到 AOSE 领域并与 Agent 技术相结合，以支持复杂自适应系统的开发，无疑具有非常重要的意义。首先，由于 Agent 技术提供了统一的高层抽象和有效的技术手段来描述、分析、涉及和实现自适应系统，因此通过 MDA 的思想可以实现高层的抽象模型到底层实现模型的自动或者半自动的映射和转换，以提高复杂自适应系统的开发效率和质量。其次，不同于成熟和标准化的对象技术，当前面向 Agent 软件开发技术具有多样化的特点，如多样化的软件 Agent 体系结构、建模语言和开发方法学。因此，将这二者技术相结合，有助于解决多样化平台实现问题。第三，自适应系统有其特有的复杂性，如环境动态性、系统持续调整性等，对这类系统的开发往往需要从多个不同的视点(如环境视点、自适应视点等)对系统进行建模和分析，同时需要综合不同视点的模型来支持系统的设计和实现。显然，如果所有这些工作都由开发人员来完成，将是非常繁杂和低效的，MDA 的引入将有助于充分挖掘不同模型之间的相关性，借助于元层上的大量可重用模式以及成功的实践和经验，可减轻软件开发人员的逻辑思维活动，减少可能潜在引入的故障。

本文针对如何高效、高质量地支持复杂自适应系统的开发这一问题，将软件 Agent 技术和 MDA 技术相结合，提出了一种基于 Agent 的模型驱动开放方法。该方法将基于组织抽象所建立起来的模型视为 PIM，将基于实现平台所建立起来的模型视为 PSM，通过建立这二个不同抽象层次元模型间的映射关系，实现模型驱动的软件开发。本文第 2 节介绍了集成 Agent 技术和 MDA 技术的软件开发方法学 ODAM+；第 3 节描述了 ODAM+所涉及的 PIM 和 PSM 以及在元层上它们之间的映射关系；第 4 节定义了模型转换规则和算法；第 5 节介绍了相应的支撑软件工具和案例分析情况；最后总结全文，展望进一步工作。

2 集成 MDA 和 Agent 技术的软件方法学 ODAM+

ODAM 是一个基于组织抽象和思想的面向 Agent 软件方法学[7]。它借助于社会组织学思想和抽象来理解和认识复杂自适应系统，提供了结构化的步骤和可视化语言 AAML 来对系统进行建模、分析、设计和实现。不同于已有的面向 Agent 开发方法学，ODAM 提供了对系统的外部环境以及自适应性进行显式描述和分析的表示机制和模型要素，可以有效支持复杂自适应性系统的开发。

需求阶段，开发人员通过抽取和分析组织场景来获取和描述需求。组织场景模型((OCM, Organization Scenario Model))描述了从外部环境所观察到的系统行为以及环境与系统二者之间的交互。分析阶段，开发人员根据组织场景模型建立系统的角色行为模型(RBM, Role Behavior Model)、Agent 交互模型(AIM, Agent Interaction Model)和角色变迁模型(RTM, Role Transition Model)。Agent 交互模型描述了系统功能是如何通过系统中不同 Agent 间的交互和协同来完成的。角色行为模型描述了系统中每个角色的职责、资源、环境、活动和服务以及这些要素间的相互关系。角色变迁模型描述系统中 Agent 如何根据外部环境和内部状态变化动态地调整它在系统中所扮演的角色。ODAM 将 Agent 对角色的动态绑定视为 Agent 适应环境的一种方式和手段。

许多社会组织具有自适应的特征，组织中的个体扮演组织中特定的角色并能随着环境的变化不断调整他所扮演的角色，从而展示他对组织环境的适应性。基于组织学思想，将自适应系统视为一个多 Agent 系统组织，将组织中能适应环境变化的 Agent 称为自适应 Agent(Self-adaptive Agent，简述为 SA)。SA 驻留在环境中能够感知环境变化，并根据外部环境或者其内部状态变化动态地调整它在组织中所扮演的角色，从而展示它对环境变化的适应性。SA 通过加入(Join)/退出(Quit)角色获得或失去该角色所定义的结构和行为特征。当 SA 加入一个角色(称之为绑定了该角色)，它就拥有该角色的属性、行为、环境和服务。SA 也可通过执行激活(Resume)和钝化(Suspend)动作将其绑定的角色的状态在活跃和非活跃之间转换。只有当绑定的角色处于活跃状态时，SA 才能执行角色中的行为。否则，角色中定义的行为不能对 SA 产生影响，但是此时 SA 仍能访问已绑定的角色中的信息。我们将这一自适应机制称为动态绑定机制。

设计阶段，开发人员根据需求和分析模型设计出系统的组织结构模型(OSM, Organization Structure Model)。组织结构模型描述了系统的全局约束、系统中的角色以及它们之间的组织关系。我们已经开发了支持 ODAM 的软件工具 ODAMTools，它提供了基于 AAML 的模型编辑、模型一致性检查等功能。

SADE(Self-Adaptation Development Environment)是一

个支持自适应 MAS 开发和运行的 CASE 平台[8]，它采用反应式的软件 Agent 体系结构，支持基于动态绑定的自适应机制来构造和运行自适应多 Agent 系统。SADE 支持将自适应 MAS 的业务逻辑与自适应逻辑相分离，以简化复杂自适应系统的开发和维护，为此提供了自适应策略描述语言 SADL，用于对自适应逻辑进行描述；以及一组可重用软件开发包，用于实现系统的业务逻辑。SADE 的运行引擎和环境负责将自适应 MAS 的业务逻辑和自适应逻辑相融合，进而实现自适应 MAS 的运行。

ODAM+是对 ODAM 的扩展，它将 MDA 思想融入到面向 Agent 软件开发方法中，并为此提供开发过程、模型转换、支撑工具等方面的支持。ODAM+将基于组织抽象所建立起来的高层分析和设计模型视为平台无关模型(PIM, Platform-Independent Model)，将基于 SADE 这一特定平台和技术所建立起来的实现模型视为平台相关模型(PSM, Platform-Specific Model)。通过建立这两个不同抽象层次模型间的映射关系，实现从 PIM 到 PSM 以及最终代码框架的转换。整个 ODAM+的组成结构如图 1 所示。

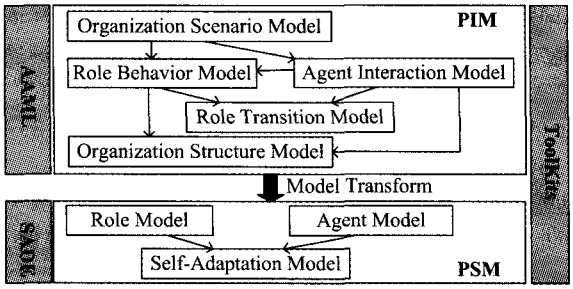

图 1 ODAM+的组成结构

3 模型及映射

3.1 平台无关模型

ODAM+提供了一组基于组织抽象的元概念(见图 2)以及可视建模语言 AAML，以对 MAS 进行建模和分析。一个 MAS 被视为一个组织(Organization)，每个组织都有其特定的组织结构(Organization Structure)。组织结构描述了构成组织的角色(Role)、角色间的关系以及系统的全局性约束(Organization Rule)。组织中的角色定义了其在组织中所处的环境(Environment)、所拥有的资源(Resource)、职责(Responsibility)、活动(Activity)以及对外提供的服务(Service)。Agent 处于组织中，它扮演着组织中特定的角色并与其他 Agent进行交互。一个 Agent 可以扮演多个角色，一个角色也可以为多个 Agent 所扮演。

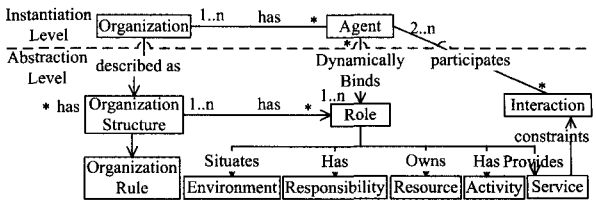

图 2 ODAM+的组织元模型

基于上述组织元模型，ODAM+提供了可视化的建模语言 AAML，从结构和行为两个视点描述 MAS 的不同抽象层次的模型，包括组织场景模型、角色行为模型、Agent 交互模型、组织结构模型、角色变迁模型。基于组织抽象的元概念以及模型都和自适应多 Agent 系统的实现技术和平台无关，它也不限定于特定的 Agent 体系结构和交互技术，在 ODAM+方法中将它们视为平台无关模型。

3.2 平台相关模型

在 Agent 开发平台 JADE(Java Agent DEvelopment Framework)基础之上，设计和实现了自适应多 Agent 系统开发和运行支撑环境 SADE[8]。JADE 是一个基于 Java 的 Multi-Agent 开发环境，它遵循 FIPA 提供的 Agent 技术标准，以中间件方式提供多 Agent 系统的开发和运行支持。JADE 提供了以下一组部件来支持多 Agent 系统的开发和运行：实现自主 Agent 的软件开发包、事件机制、目录服务、FIPA ACL 交互等。

一个由 SADE 开发的软件系统基于如图 3 所示的实现元模型，并包含以下一组基本模型。

• 角色类模型(RCM, Role Class Model)，定义和封装构成系统的基本模块单元角色类，每个角色类进一步封装了其属性、行为、服务、事件和环境。环境是自适应 MAS 系统需要关注的一个非常重要的因素。不同的环境需要不同的传感器去感知环境的变化。SADE 系统提供了一些常见环境的基本传感器，如基于网络的事件传感器、网络带宽传感器等。

• Agent 模型(AM, Agent Model)，定义和封装了 Agent 的一些基本信息和行为，如名字、对应的自适应策略等。SADE 允许一个 Agent 可以有 0..n 个不同的自适应策略。Agent 既可以在 Agent 模型中定义它所特有的行为，也可以通过绑定角色获得相应的行为。

• 自适应策略模型(SSM, Self-adaptive Strategy Model)，定义了基于 SADL 描述的 Agent 自适应策略。SADE 要求每一个自适应策略用于刻画某个特定 Agent 的自适应特征。

这些实现模型依赖于 SADE 及其所提供的一组关键技术，包括自适应策略描述语言、可重用软件包、运行支撑环境等，因而是一组平台相关模型。在开发层面，SADE 主要通过以下两个方面来支持复杂自适应 MAS 的开发。

• 可重用软件包，它封装了自适应 MAS 的一些基本功能，如自适应机制、生命周期管理、事件机制、软传感器等。软件开发人员可通过重用这些类来实现系统的业务功能，从而简化复杂自适应 MAS 的开发。

• 自适应特征描述语言 SADL，它是一个描述性的语言，用于定义 Agent 的自适应策略。自适应策略描述了 Agent 的自适应逻辑，它包含初始策略和一组一般策略。初始策略定义了自适应 Agent 在加载策略的时刻就需要执行的行为。一般策略的定义包括策略的声明和自适应规则的定义，其中策略的声明用来定义策略的名字。每个自适应策略均包含一个或者多个自适应，它定义了自适应 Agent 在不同的场景下需执行的自适应行为。一般地，自适应规则具有如下形式："when(ChangeExp) [if(StateExp)] {(AdaptiveAction())*}"。其中，ChangeExp 是对自适应规则所处环境变化的描述，StateExp 描述了自适应 Agent 的状态，AdaptiveAction 定义了在满足这两个条件时 SA 执行的自适应动作序列，如 Join 一个角色，或者 Quit 一个角色。该形式描述了当外部环境和内部状态发生变化以及 Agent 内部状态满足特定条件时，Agent 该执行什么样的自适应动作以调整角色从而适应变化。

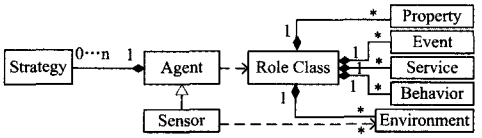

图 3　基于 SADE 的软件实现元模型

3.3　元模型映射

实现 PIM 到 PSM 转换的前提是必须确立两个不同抽象层次的元模型间的映射关系，从而为模型转换提供元层映射基础。表 1 描述了 PIM 和 PSM 元概念之间的对应关系。

表 1　元模型映射表

PSM 的元概念	PIM 的元概念	对应的 PIM 模型
Agent	Agent	RTM, AIM
Role Class	Role	OSM, AIM, RBM
Property	Resource	RBM
Behavior	Responsibility, Activity, Interaction	RBM, AIM
Environment	Environment	OSM, RBM
Strategy	Dynamic Binding	RTM
Event		RBM, AIM, OCM
Service	Service	RBM
Sensor	Environment	OCM

SADE 中的 Agent 可以是自适应 Agent，也可以是普通 Agent。系统中各个 Agent 是自适应 Agent 还是普通 Agent 取决于应用需求，并通过继承不同 Agent 类加以实现。PSM 中的 Role Class 与 PIM 中的 Role 相对应，PIM 中角色的资源被封装为 Role 类中的属性，角色的职责、活动和交互被封装为 Role Class 的行为，Role 的环境对应于 Role Class 的环境。角色变迁模型 PIM 所描述的自适应特征信息将被映射为由 SADL 描述的自适应策略。一般地，不同的环境需要不同的传感设施。

4　模型转换算法

在 ODAM+中，从 PIM 到 PSM 的转换通过以下两个阶段来完成：从 ODAM+的 PIM 模型到 SADE 的 PSM 模型，从 SADE 的 PSM 模型到目标代码框架。

4.1　从 ODAM+模型到 SADE 模型

从 ODAM+的 PIM 模型到 SADE 的 PSM 模型的转换分为以下 3 个步骤：首先，根据组织结构模型生成目标软件系统的 Role Class 模型；其次，根据角色行为模型、角色交互模型对角色类模型进行精化；第三，根据角色交互模型、角色变迁模型生成 Agent 模型和自适应策略模型。

• 生成角色类模型

```
GenerateRCModel (Input: OSM, Output: RCM) {
for (each role r in organization structure model M ) // generate role class models
    if(role model M_r generated from r doesn't exist)
        generate a role model M_r;
        M_r.name = r.name;
    endif
    if(role r inherits from role r_0)
        if(role model M_r0 generated from r_0 doesn't exist)
            generate a role model M_r0;
            M_r0.name = r_0.name;
        endif
        establish the inheritance relationship between M_r and M_r0 ;
    endif
end for
}
```

图 4　生成角色类模型算法

ODAM+组织结构模型中的每一个角色被转换为 SADE 模型中的对应角色类（算法见图 4），它们具有相同的名字。如果 ODAM+组织结构模型中的角色之间存在某些关系（如继承等），那么这一关系将同样被转换为 SADE 模型中对应角色类之间的关系（如继承）。

• 精化角色类模型

该步骤根据 ODAM+的 RBM，将对上一步所生成的角色类模型做进一步的精化。首先，角色行为模型中各个角色所具有的职责将被转换为角色类模型中相应的行为，并将职责间的分解关系转换为行为间的分解关系。其次，将角色行为模型中各个角色的环境转换为角色类中的环境，并确定环境的类型，为其生成相应的传感器。第三，角色行为模型中的服务转换为角色类中的服务。第四，角色模型中的资源转换为角色类中的属性。图 5 详细描述了此转换算法。

```
RefineRCModel(Input RBM, Output RCM) {
for(each role behavior model RBM and the role r that it describes)
    find the role class model M_r generated from r;
    for(each responsibility rp in RBM) //generate behaviors
        generate a behavior beh of M_r ;
        beh.name = rp.name;
        if(rp is decomposed as a set of sub-responsibilities)
            for(each sub-responsibility srp of rp)
                generate a sub-behavior sbeh of beh;
                sbeh.name = srp.name;
            endfor
        endif
    endfor
    for(each service s in RBM) //generate services
        generate a service ser of M_r ;
        ser.name = s.name;
    end for
    for(each environment env in RBM)// generate environments
        if(there is service-dependence d between r and env )
            generate an environment en_r of M_r ;
            en_r.type = "Service";
            en_r.name = d.name;
            create the sensor for the environment
        end if
        if(there is active-dependence d between r and env)
            generate an environment en_r of M_r ;
            en_r.type = "Role";
            en_r.name = d.name;
        end if
    end for
    for(each resource rs in M)//generate attributes
        generate a Class c;
        c.name = rs.name;
        generate an attribute att of M_r ;
        att.name = c.name.toLowerCase();
        att.type = c.name;
    end for
end for
}
```

图 5　精化角色类模型算法

• 生成 Agent 模型和自适应策略模型

该步骤将根据 ODAM+的 RTM 和 AIM 生成 SADE 中的 Agent 模型和自适应策略模型。首先，根据 RTM 和 AIM 生成 SADE 中的 Agent 模型，并保留名字的一致性。其次，根据 RTM 生成自适应策略模型，包括生成构成策略的自适应规则、触发事件、自适应动作等。具体算法见图 6。

```
GenAgt&StrModels(Input: RTM, AIM, Output: AM, SSM) {
    for(each agent a in AIM)
        generate an Agent model M_a;// generate SAgent models
        M_a.name = a.name;
    end for
    for(each role transition model RTM and the agent a it describes)
        if (no corresponding agent model)
            generate an Agent model M_a;
            M_a.name = a.name;
        endif
        generate a Strategy model M_s for M_a; //generate strategy models
        M_s.name = a.name +"Strategy";
        // generate adaptive rules of the strategy model
        for(each transition t in RTM)
            generate an adaptive rule rl of M_s;
            set the change event of rl with the event of t;
            set the adaptive action set of rl with the action set of t;
            for(each role r as active role in the source configuration of t)
                generate a state st of rl;
                st.name = r.name + "@Active";
            end for
            for(each role r as inactive role in the source configuration of t)
                generate a state st of rl;
                st.name = r.name + "@Inactive";
            end for
        end for
    end for
}
```

图 6　生成 Agent 模型和自适应策略模型算法

4.2　从 SADE 模型到代码框架

采用基于 JET（Java Emitter Templates）的模板方式实现从 SADE 模型到代码框架的转换。作为 EMF（Eclipse Modeling Framework）的一个组成部分，JET 提供了类似 JSP 的方式来书写模板。

根据 SADE 的角色类模型、Agent 模型和自适应策略模型，所生成的目标代码框架包含以下 3 类程序代码：角色类、Agent 类和自适应策略。对于 SADE 模型中的每个角色类，将生成一个角色类程序代码，该类继承可重用软件包中的 Role 类，并设置其缺省的两个方法“init()”和“exit()”，并在“Init()”方法中生成代码“addBehvior(BehvaiorName);”和

"addEnvrionment(new Envriornment(EnvrionmentType, EnvrionmentName);",以加入该角色类的行为和环境。对于角色类中的每一个行为,将生成一个行为类,该类继承可重用软件包中的 Behavior 类,并设置其初始方法"action()"和"done()"。角色类的每个服务需要在该角色类的"init()"和"exit()"方法中进行注册和注销。对于每一个 Agent 类模型,如果该 Agent 对应有自适应策略,则其为自适应 Agent,为此需要生成一个继承"SAgent"的 Agent 类,否则生成一个继承 Agent 的 Agent 类。对于自适应 Agent,需要设置其缺省的方法"setup()"和"takeDown()"。该 Agent 的自适应策略信息需要在"setup()"方法中增加代码"setStrategy(this,StrategyName);"以初始化。对于每个自适应策略模型,将生成一个用 SADL 描述的自适应代码(见图 7)。

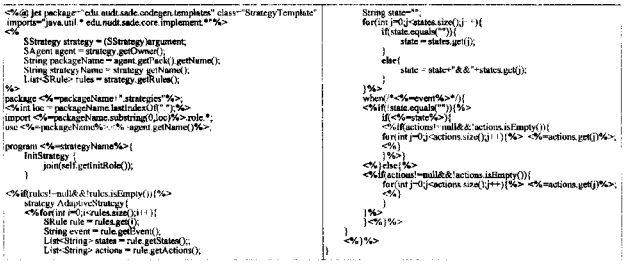

图 7 生成基于 SAD 的自适应策略描述代码模板

5 支撑工具和应用验证

为了支持自适应系统的模型驱动开发,开发了软件工具 ODAMTransformer,并将其集成到支撑 ODAM+的软件工具集 ODAMTools 中(见图 8)。ODAMTransformer 以 ODAMTools 所建立起来的可视化平台无关模型为输入,支持不同抽象层次的模型转换:从 ODAM+的平台无关模型到 SADE 的平台相关模型及至最终的程序代码框架。基于 ODAMTransformer 所生成的程序代码框架包含以下两方面形式:基于 Java 的自适应系统业务逻辑代码框架和基于 SADL 的自适应逻辑代码框架。这些代码框架定义了构成目标软件系统的基本构件成分以及它们之间的交互关系,描述了系统中自适应 Agent 的自适应策略。

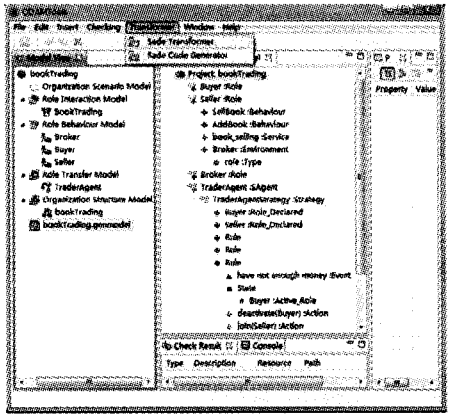

图 8 ODAMTransformer 软件工具

基于自适应多 Agent 系统的模型驱动开发方法 ODAM+以及一组支撑关键技术和软件工具,我们成功开展了若干个应用案例的分析和开发,包括开放环境下自适应机器人 GarbageCleaner、基于互联网的自适应交易系统等[8]。由于篇幅限制,这些案例分析的细节不再详述。

实践表明,本文所提出的方法和工具可以生成结构良好的自适应系统目标代码框架,尤其是它们将自适应逻辑和业务逻辑相分离,在代码生成过程中重用一些好的设计和实现模式,使生成的代码结构具有较好的可理解性、可维护性和可追踪性,从而提高了复杂自适应系统的软件开发效率和质量。

6 相关工作对比分析

在运行时适应环境变化被视为是驻留在开放环境(如 Internet、物理系统等)下软件系统的一项重要特性,也是超大规模软件系统的一个基本特征[1]。由于应用的广泛性,自适应系统的研究受到了传感网络、自治计算、人工智能、分布计算等领域的关注和重视。在软件工程领域,有关自适应系统研究近年来非常活跃,现有的大部分工作借助于控制反馈理论和反射技术,从软件体系结构、软构件、中间件、运行支撑环境等方面,来研究支撑自适应系统构造和运行的关键技术[3]。无疑,这些工作为自适应系统的开发提供了扎实的实现技术,但是从软件工程的视点,现有的工作并不系统。由于自适应系统自身的复杂性,如何为此类系统的开发提供方法学指导,提高开发的效率和质量,仍然是自适应系统软件工程面临的一项重要挑战。

近年来出现的 Agent 技术被认为是当前解决诸如自主和自适应软件系统开发的一种重要方法[3]。本世纪初,人们提出了几十种面向 Agent 建模语言和开发方法学(如 Gaia、Tropos、AML 等),试图为复杂系统的工程化开发提供技术支持。但是,这些方法和语言或者施加过多的约束和限制(如要求环境是静态的、系统是封闭或者半封闭的),难以充分发挥 Agent 技术的潜力,真正用于解决复杂系统(如自适应系统)的开发,或者缺乏有效的实现技术支撑,造成分析设计与实现相脱节,因而难以为人们所接受[9]。文献[10]认识到在方法学层面上支撑自适应系统开发的重要性,讨论了如何对面向 Agent 开发方法学 Gaia 进行可能的扩充来支持自适应系统的开发,但是该工作并没有提供系统方法来支持自适应系统的开发。文献[12]引入了 Context 模型来显示描述开放环境,设计了多种角色空间模型来支持对系统多样化角色及其变化的建模和分析。在这方面,最近一项重要的工作是 O-MaSE 方法学,它基于适应计算系统的元模型,允许 Agent 通过重新扮演角色来实现运行时的重组织和自适应。该方法并没有特别关注对适应性的描述,也没有提供相应模型来支持对适应性进行分析[11]。

近年来,将模型驱动开发思想引入到面向 Agent 软件工程领域来支持复杂多 Agent 系统的开发受到了人们的关注[16]。可以注意到现有的工作大多针对不同的抽象模型和实现技术提供不同的模型转换方法,进而服务于不同的软件开发目的。文献[13,18]介绍了基于 MAS-ML 建模语言所描述的高层模型转换为基于 Jade 平台实现模型的方法。文献[14]提出了针对面向 Agent 软件开发方法 INGENIAS 的模型转换方法及其支撑工具。文献[15]基于 SMART 概念框架,提出了一种基于构件的面向 Agent 模型转换方法,它可以将高层的分析和设计模型转换为针对 JACK 的实现模型;针对现有面向 Agent 软件开发方法学只关注软件的分析和设计,忽视从设计到平台代码的生成这一问题,文献[17]分析了现有面向 Agent 开发方法学存在的问题,即关注分析和设计,

忽视到实现的转换，为此提出了将基于 Tropos 方法学的软件设计模型转换为一类基于 Malaca 元模型的平台代码，包括 JADE 平台、FIPA-OS 平台等。文献[19]介绍了基于自动工具支持的模型转换技术，亦即将基于 Tropos 的规划分解转换为基于 UML 表示的活动。

结束语 面向 Agent 软件工程的研究近年来取得了长足进步，但其实践与应用却不尽如人意，未能像人们所预期的那样，在解决复杂系统的开发以及在大范围的工业化应用中发挥作用。其原因之一在于，现有面向 Agent 软件工程的研究未能充分借鉴软件工程(尤其是 OO 软件工程)的成功技术和实践。MDA 可提高软件开发的效率和质量，将这一技术引入到面向 Agent 软件工程，可以充分发挥 Agent 技术潜力，更好地促进复杂系统的开发。

本文提出了一个集成 Agent 技术和 MDA 的模型驱动软件开发方法 ODAM+，以更好地支持复杂自适应系统的开发。该方法建立在我们已有的两项研究工作基础之上：基于组织抽象的面向 Agent 开发方法学 ODAM、自适应系统的开发和运行支撑环境 SADE。不同于已有的工作，上述两项研究均提供了核心机制、元模型和模型、语言设施对复杂自适应系统的环境以及自适应特征进行显式的描述和分析。ODAM+将基于组织抽象和 ODAM+方法学所建立起来的模型视为平台无关模型，将基于 SADE 平台的实现模型视为平台相关模型，通过建立这两个不同抽象层次元模型间的映射关系，实现从平台无关模型到平台相关模型以及最终代码框架的转换。

我们已经基于此方法成功进行了多个自适应系统应用案例的开发，充分展示了技术的可行性和有效性。进一步，通过重用一些有效的模式和经验，提高通过模型转换所生成代码的质量。此外，正在研究将基于 ODAM+的模型转换为其他平台的方法和技术，如 JADE、JACK 等一类应用较为广泛的平台，以扩大 MDA 技术在面向 Agent 软件工程的应用范围。

参 考 文 献

[1] Northrop L, et al. Ultra-large-scale Systems: The Software Challenge of the Future[M]. Software Engineering Institute, Carnegie Mellon University, USA, 2006

[2] 吕建，马晓星，陶先平，等. 网构软件的研究与进展[J]. 中国科学：E 辑信息科学，2006，36(10)：1037-1080

[3] Cheng B H, de Lemos R, Giese H, et al. Software engineering for self-adaptive systems: A research roadmap[C]// Proc. of Software Engineering for Self-Adaptive Systems, LNCS 5525. Springer, 2009: 1-13

[4] Salehie M, Tahvildari L. Self-adaptive Software: Landscape and Research Challenges[J]. ACM Transactions on Autonomous and Adaptive Systems, 2009, 4(2): 1-40

[5] Luck M, Mcburney P, Preist C. Agent Technology: Enabling Next Generation Computing[EB/OL]. http://www.agentlink.org, 2005

[6] Winikoff M. Future Directions for Agent-based Software Engineering[J]. International Journal of Agent-Oriented Software Engineering, 2009, 3(4): 402-410

[7] 毛新军，王戟. 自适应多 Agent 系统的面向 Agent 软件开发方法学 ODAM[J]. 计算机研究与发展，2008，45(11)：1892-1901

[8] 毛新军，等. 自适应网构软件的集成开发环境 SADE[J]. 电子学报，2010，38(2A)：207-212

[9] Weyns D, Helleboogh A, Holvoet T. How to Get Multi-agent Systems Accepted in Industry? [J]. International Journal of Agent-Oriented Software Engineering, 2009, 3(4): 383-390

[10] Cernuzzi L, Zambonelli F. Dealing with Adaptive Multi-Agent Organizations in the Gaia Methodology [C] // Proc. of 5th AOSE. Springer, 2005: 217-228

[11] DeLoach S A, García-Ojeda J C. O-MaSE: A Customizable Approach to Designing and Building Complex, Adaptive Multiagent Systems[C]// LNCS 4957. Springer, 2008: 1-15

[12] Wang Li-ming, Li Ya-chong. A Systematic Methodology for Adaptive Systems in Open Environments [C] // PRIMA 2006, LNAI 4088. Springer, 2006: 117-128

[13] Anarosa A, Brandão F, et al. A Model Driven Approach to Develop Multi-agent Systems [J]. Monografias em Ciência da Computação, 2005, 9(5): 1-15

[14] Pavón J, Gómez-Sanz J J, Fuentes R. Model Driven Development of Multi-Agent Systems [C] // Proc. of ECMDA-FA, LNCS 4066. Springer, 2006: 284-298

[15] Jayatilleke G B, et al. A model driven component-based development framework for agents[J]. International Journal of Computer Systems Science & Engineering, 2005, 4(1): 273-282

[16] France R, Rumpe B. Model-driven Development of Complex Software: A Research Roadmap[C] // Proc. of Future of Software Engineering. 2007: 37-54

[17] Amor M, Fuentes L, Vallecillo A. Bridging the Gap Between Agent-Oriented Design and Implementation Using MDA[C] // Proc. of Agent-oriented Software Engineering, LNCS3382. Springer, 2005: 93-108

[18] De Maria B A, et al. Developing Multi-Agent Systems Based on MDA[C]// Proc. of the 17th Conference on Advanced Information Systems Engineering. 2005

[19] Perini, Susi A. Automating Model Transformations in Agent-Oriented Modeling[C]// Proc. of Agent Oriented Software Engineering. Springer, 2006: 167-178

(上接第 104 页)

[8] Koyama A, Arai J, Barolli L, et al. EZRP: an Enhanced Zone-based Routing Protocol for Ad-hoc Networks: Research Articles [J]. Concurrency and Computation: Practice & Experience, 2007, 19(8): 1157-1170

[9] 党琦. Ad Hoc 网络区域路由协议自适应算法的研究[D]. 南京：南京理工大学，2007

[10] 赵辉. 基于功耗考虑的有效性区域路由协议研究[D]. 西安：西安电子科技大学，2008

[11] 王朝瑞. 图论[M]. 北京：国防工业出版社，1985：146-148

[12] 严蔚敏，吴伟民. 数据结构(C 语言版)[M]. 北京：清华大学出版社，1997：156-193

[13] Zygmun J H, Peatlman M R, Samar P. The Interzone Routing Protocol(IERP) for Ad Hoc Networks[EB/OL]. IETF Internet Draft, draft-ietf-manet-zone-ierp-02. txt, 2002

[14] Zygmunt J H, Peatlman M R, Samar P. The Bordercasting Resolution Protocol(BRP) for Ad Hoc Networks[EB/OL]. IETF Internet Draft, draft-ietf-manet-zone-brp-02. txt, 2002

[15] 戴树森，费鹤良，等. 可靠性试验及其统计分析[M]. 北京：国防工业出版社，1983：168-170