自适应网构软件的集成开发环境 SADE

毛新军,李学斯,尹俊文,董孟高,胡翠云,吴 斌
(国防科学技术大学计算机学院,湖南长沙 410073)

摘 要: 网构软件通常驻留在动态、开放的环境中,需根据环境变化展示自适应和自演化能力,代表了一类复杂系统.如何有效支持这类软件系统的开发是软件工程面临的一项重要挑战.本文介绍了一个基于 Agent 的网构软件集成开发环境 SADE,它建立在一组网构软件关键技术基础之上,包括:基于 Agent 的网构软件抽象和构造,动态绑定的自适应和自演化机制,基于组织抽象的软件开发方法学 ODAM,自适应和自演化策略描述语言 SADL 等.论文分析了 SADE 的技术框架以及各个组成部分,包括网构软件分析和设计工具集 ODAMTools、编程工具集、运行支撑平台等;最后通过案例分析阐述了如何利用 SADE 来进行网构软件开发.

关键词: 网构软件; Agent; 自适应; SADE; ODAM
中图分类号: TP311 **文献标识码:** A **文章编号:** 0372-2112 (2010) 2A-207-06

SADE: An Integrated Development Environment for Self-Adaptive Internetwares

MAO Xin-jun, LI Xue-si, YIN Jun-wen, DONG Meng-gao, HU Cui-yun, WU Bin
(*Department of Computer Science and Technology, National University of Defense Technology, Changsha, Hunan* 410073, *China*)

Abstract: Internetware system is a kind of complex system that is typically situated in dynamic and open environments and adapts to the changes of environment. How to effectively develop such systems has become a great challenge in software engineering community. This paper introduces an agent-based integrated environment SADE for developing Internetware systems. It consists of a number of technologies for Internetware such as agent-based abstraction and construction, dynamic binding mechanism for self-adaptation and self-evolution, ODAM methodology, self-adaptive strategy description language SADL, etc. The technical framework of SADE and its components are introduced in details, including ODAMTools to analyze and design Internetware, the programming toolkits to code Internetware and run-time infrastructure. A case study is illustrated to show how to develop Internetware system with our approach.

Key words: Internetware; Agent; self-adaptive; SADE; ODAM

1 引言

随着计算机技术的不断发展以及应用面的不断扩大和深化,软件系统的规模越来越大,并呈现一些新的复杂性特点,具体表现为:(1)环境复杂性,软件系统所驻留的环境变得日趋复杂,并具有开放、动态、不可控、无法事先确定等特点,如 Internet.环境的变化将对软件系统产生实质性的影响.(2)系统复杂性,软件系统驻留在环境中,需要不断地调整自身的结构和行为以适应环境的变化、满足系统的设计目标,从而展示出自适应和自演化的复杂性特征.网构软件就是这类系统的一个集中体现,它是 Internet 时代一种新的软件形态,具有主体性、协同性、反应性、自适应性、自演化性等复杂性特征[1,2].CMU SEI 的一份研究表明:未来的软件系统将是超大规模(Ultra-Large-System)的,其中自治、自适应、自演化、自组织是这类系统的重要特征[3].

从软件工程的视点,网构软件形态的出现对支持这类软件系统开发的概念和元模型、建模和编程语言、开发和运行平台、部署和维护技术等提出了新的要求.如何有效支持网构软件系统的开发是目前软件工程面临的一项重要挑战.围绕这一问题,近年来人们开展了一系列的工作[2,4].尤其是,人们从构件、软件体系结构、模型、中间件、运行机制等方面提出了支持自适应和自演化的许多关键技术[5,6].绝大部分技术借助于以面向

收稿日期:2009-07-01;修回日期:2010-01-03
基金项目:国家 863 高技术研究发展计划(No.2007AA01Z135);国家自然科学基金(No.60773018);国家 973 重点基础研究发展规划(No.2005CB321802)

对象方法为基础的主流技术体系,将对象作为 Internet 环境下软件实体的概念模型和实现模型.然而对象技术本身存在固有的局限和不足:(1)非持续性运行;(2)非自主性;(3)行为的不变性和不可调整性,对象一旦创建,其结构和行为将在其生命周期中不可改变;(4)控制为基础的交互机制,语法层次上的消息传递,本质上是方法调用;(5)对环境的被动感知,不被告知,就无法感受环境的变化.因此,无论是高层的自然建模还是底层的实现支撑,对象技术都无法有效应对网构软件的环境开放性、变化敏感性、系统动态性等给软件开发方法提出的要求.与此同时,现有关于自适应和自演化网构软件系统的开发大多关注系统实现,而忽视了对复杂网构软件系统的需求分析、软件设计和建模.

2 网构软件的支撑关键技术

2.1 基于 Agent 和组织抽象的网构软件元模型

Agent 是指驻留在特定的环境下能够感知环境并能自主运行以满足设计目标的软件实体,它代表了一种新颖的概念抽象和计算抽象,更加贴近于人们对现实世界中行为实体的直观理解和认识,有助于对应用系统和软件系统中的行为实体进行自然建模.我们将网构软件中的实体抽象、封装和物化为软件 Agent,将网构软件系统视为由多个相互交互作用的 Agent 所构成的多 Agent 系统.网构软件实体的 Agent 化体现了网构软件实体主体化程度的提升以及在内容包含性、结构独立性、实体适应性、行为自主性等方面进步.尤其是软件 Agent 的行为自主性可有效支持互联网环境的动态性、开放性和不确定性,以及在此环境下网构软件的自适应性和自演化性.

社会学和组织学的思想可以为网构软件系统的开发提供高层抽象和自然建模手段.首先,越来越多的网构软件系统服务于各种各样的组织;第二,组织学和社会学可为网构软件系统的分析和建模以及自适应和自演化等特征的研究提供高层的抽象和概念,它们直观、易于理解、更加贴近现实世界,有助于简化和控制网构软件系统的复杂度,代表了软件工程在抽象层次上的进步.为了支持网构软件的分析、设计和建模,我们提出了基于组织抽象的网构软件概念模型.具体的,将网构软件系统视为具有特定组织结构和约束的社会,系统中的 Agent 扮演组织中的特定角色,从而展示相应的行为.每个角色描述了网构软件 Agent 在组织中所承担的职责、具有的权限、驻留的环境、对外提供的服务等等.

2.2 基于动态绑定思想的自适应机制

我们注意到许多社会组织具有自适应和自演化特征,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它所扮演的角色,从而展示他对组织环境的适应性.基于这一思想,我们提出了基于动态绑定思想的自适应和自演化机制[7],以支持自适应网构软件系统的分析、设计和构造.所谓动态绑定机制是指,任何网构软件 Agent 均驻留在特定的组织环境中,当组织环境发生变化时 Agent 可动态地加入(Join)一个角色或者退出(Quit)一个角色,从而获得或者失去该角色所定义的结构和行为特性,以满足系统的设计目标.角色是对 Agent 结构和行为的抽象表示.一旦 Agent 加入一个角色,我们称 Agent 绑定了该角色,该角色所定义的结构和行为信息将约束该 Agent 的运行和对外所展示的功能,在此情况下我们称 Agent 绑定了该角色.Agent 所绑定的角色具有以下二种不同的状态:活跃(Active)状态和非活跃(Inactive)状态.当 Agent 所绑定的角色处于活跃状态时,他将约束 Agent 的运行.例如,Agent 将根据角色所定义的行为规约来选择动作、实施行为.当 Agent 所绑定的角色处于非活跃状态时,他将不再约束 Agent 的运行,但是 Agent 将保留并访问该角色对应的相关信息.Agent 可以通过实施激活(Activate)和钝化(Deactivate)操作来改变 Agent 所绑定角色的状态.网构软件的自演化实际上就是网构软件在其生命周期中随着环境的变化而不断调整自身角色的过程.

2.3 网构软件开发方法 ODAM

基于网构软件概念模型和动态绑定的自适应和自演化机制,我们提出了如图 1 所示的网构软件开发方法学 ODAM(Organization-based Model Driven Agent-oriented Methodology)[8].ODAM 方法学由基于组织抽象和多个视点的网构软件模型、基于迭代的网构软件开发活动和步骤、将高层组织模型转换为底层实现模型的模型转换技术三部分组成.网构软件模型包括组织场景模型、角色行为模型、组织交互模型、角色变迁模型和组织结

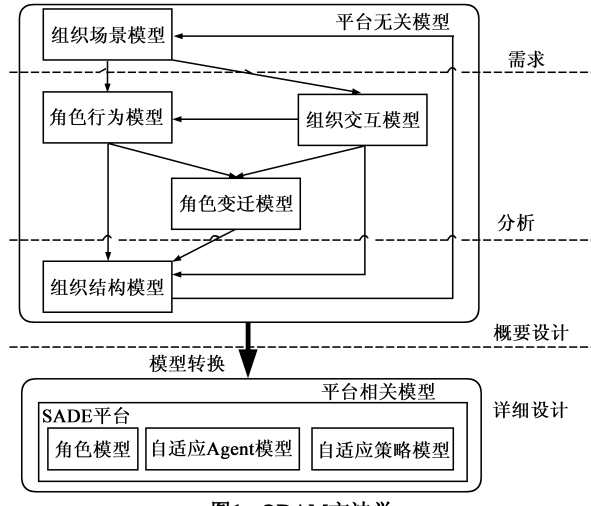

图1 ODAM方法学

构模型,它们是从不同的视点和角度对网构软件的描述和分析.整个方法学支持网构软件的需求、分析、概要设计等多个软件开发活动.此外,为了提高网构软件的开发效率和质量,ODAM 集成了主流软件工程的 MDA 思想,将高层的网构软件组织模型视为平台无关模型,将针对特定实现技术的模型视为平台相关模型,支持从平台无关模型到平台相关模型的转换.

2.4　网构软件的自适应策略描述语言 SADL

网构软件系统是一类复杂系统,当前网构软件系统的开发通常将网构软件系统的业务逻辑与自适应和自演化逻辑缠绕在一起,封装在同一软件模块中,这使得开发人员需要同时关注二个不同内容并注意二者之间的逻辑关系,导致网构软件系统的开发变得极为复杂,系统难以维护.为此,我们提出了将网构软件的业务逻辑与自适应和自演化逻辑相分离的思想,设计了相应的自适应自演化策略描述语言 SADL(Self-Adaptation strategy Description Language)[9],以对网构软件的自适应特性进行描述.

在网构软件 Agent 的生命周期中,它所处的环境可能会不断发生变化,网构软件 Agent 需要对其所处环境的变化进行感知,并根据环境的变化情况以及自身的状态来及时地调整自身的状态和行为,以适应环境的改变.一个由 SADL 编写的策略文件对应于一个网构软件 Agent 在自适应和自演化过程中所应遵循的规则、约束和限制,每个自适应自演化策略由一组自适应和自演化规则组成.自适应规则具有以下形式:

"when　(ChangeExp)　[if　(StateExp)]
{(AdaptiveAction()) * }"

其中 ChangeExp 是以环境事件的形式对网构软件 Agent 所处环境变化的描述,StateExp 描述了网构软件 Agent 的状态,AdaptiveAction 定义了在满足这两个条件时网构软件 Agent 应执行的自适应动作,如加入(Join)一个角色,退出(Quit)一个角色,激活(Activate)一个角色等.因此,一个自适应规则描述了当环境发生变化,网构软件 Agent 满足特定的状态时,它应实施的自适应行为.自演化规则刻画了网构软件 Agent 在其自适应过程中应遵循的约束和限制,这些约束和限制最终由平台中的相应模块解析,总体保障网构软件 Agent 在自适应自演化过程中按照开发者定义的约束运行.通过定义网构软件 Agent 的自适应自演化策略,软件开发人员可以清晰地表示网构软件 Agent 生命周期中的自适应行为和自演化约束,从而有效支持自适应网构软件的实现和运行.

3　网构软件的集成开发环境 SADE

网构软件集成开发环境 SADE 如图 2 所示,它由支持利用 ODAM 来对网构软件进行分析和设计的工具集、支持利用 Java 和 SADL 来对网构软件进行编程的工具集、网构软件运行支撑平台等部分组成.模型编辑器、检查器和转换器支持开发人员利用 ODAM 方法学对网构软件进行分析和设计,开发包、SADL 编辑器和编译器等支持网构软件的开发和构造,运行引擎、冲突检查和消解模块支持网构软件的运行.

SADE 建立在 JADE[10] 基础设施之上.JADE 是一个基于 Java 的 Multi-Agent 开发环境,它借助于 Java 虚拟机,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 本身提供了一部分组件来支持多 Agent 系统的开发和运行,包括实现自主 Agent 的软件开发包,事件机制,目录服务,FIPA ACL 交互等等.SADE 在此基础上提供:

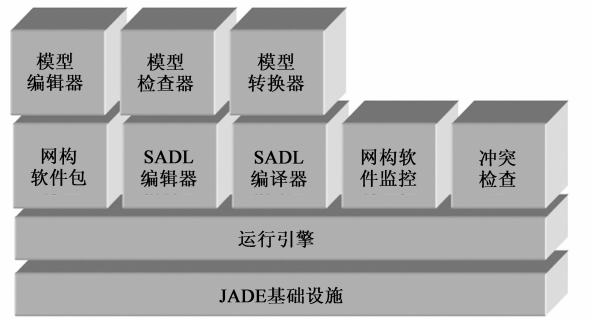

图2　SADE的技术框架

①运行引擎　运行引擎为网构软件的运行提供支撑,它主要提供了以下的功能:(1)支持动态绑定机制,允许网构软件 Agent 通过"Join"、"Quit"等自适应操作来动态绑定不同的角色,适应环境的变化;(2)负责加载网构软件 Agent 的自适应策略;(3)负责网构软件 Agent 生命周期的管理.

②网构软件开发包　为了支持网构软件基于动态绑定思想的自适应和自演化的核心机制,SADE 提供了一组可重用的软件开发包.该开发包提供了网构软件的一组公共功能,软件人员可以通过继承这些可重用类来进行系统的开发和维护.具体的,网构软件开发包主要包含了以下一组构件:(1)Role 类,封装了业务逻辑以及该类所对应的环境;(2) SAgent 类,是对具有自适应和自演化能力 Agent 的抽象和封装;(3)Environment 类,封装了环境的基本属性和功能,SADE 支持对网构软件 Agent 的环境进行显式的表示和编码;(4)Publisher 和 Subscriber 类,用于对环境事件的订阅和发布.

③SADL 编辑器　SADL 编辑器支持对 SADL 策略描述语言进行编辑,它支持在线的语法检查以及错误提示,提供了语法高亮显示、内容辅助提示、语法错误提示、编译错误位置定位等功能.目前 SADL 编辑器已经作为插件集成到 Eclipse 开发环境中(如图 3 所示).

④SADL 编译器 SADL 编译器负责将由 SADL 语言描述的自适应逻辑翻译为 Java 程序,这些 Java 程序经过编译后生成 Java 中间码,然后在 SADE 环境下来运行.SADL 编译器除了提供编译功能之外,还提供了 SADL描述的语法错误分析以及修正建议功能.目前 SADL 编译器已经作为插件集成到 Eclipse 开发环境中(见图 3),并与 SADL 编辑器相集成,支持对编辑的策略文件进行编译和分析.

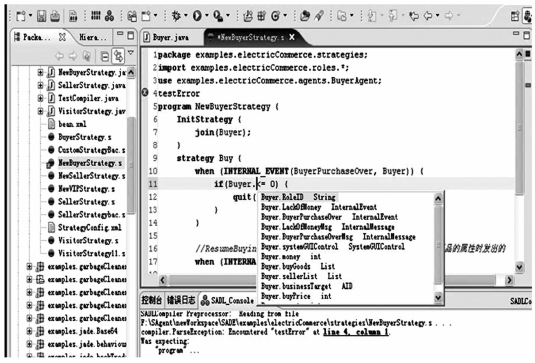

图3 集成到Eclipse中的SADL编辑器和编译器

⑤网构软件监控器 网构软件监控器负责监控网构软件中的各个 Agent,获取和监视其状态和自适应、自演化的行为和特征.用户也可以通过该软件工具来对网构软件 Agent 进行控制,包括生成新的网构软件 Agent、删除已有的网构软件 Agent 等等.

⑥冲突检查器 由于个体 Agent 的自主性和多 Agent系统中对 Agent 之间相互协调要求的固有矛盾,以及动态绑定过程本身要遵循的一些规则与约束,网构软件 Agent 在随环境变化的自适应和自演化过程中,通常会引发一些冲突问题.冲突检查器负责对这些冲突进行检查和消解,保障网构软件系统正确运行.

⑦模型编辑器 为了支持开发人员利用 ODAM 方法学来对网构软件进行分析和设计,SADE 提供了 ODAM编辑器.该编辑器采用图形化的方式对组织场景

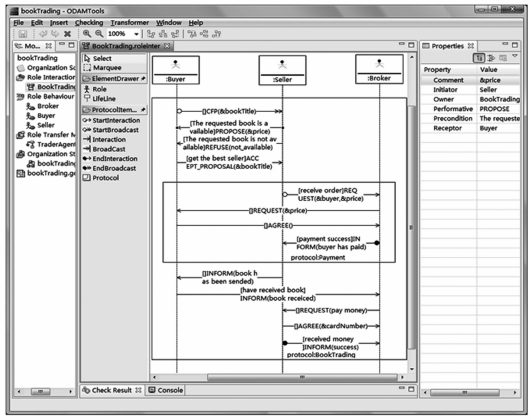

图4 支持模型编辑检查和转换的ODAMTools

模型、角色行为模型、组织交互模型、角色变迁模型等进行可视化的建模(如图 4 所示).

⑧模型检查器 为提高 ODAM 模型的质量,及时发现其中存在的问题,SADE 给出了 ODAM 模型的一致性、完整性和正确性的描述,并提供了模型检查器以对开发人员编辑的 ODAM 模型进行检查和分析,并将发现的问题及时反馈给开发人员.模型检查器与模型编辑器集成在一起,支持在线和离线模型检查.

⑨模型转换器 SADE 还提供了一个模型转换器,将用 ODAM 描述的平台无关模型转换为由 SADL 和软件开发包所描述的平台相关模型,从而简化网构软件系统的开发.目前,模型转换器能够生成业务逻辑的代码框架以及基于 SADL 的自适应自演化策略描述.

4 网构软件的开发过程和案例分析

基于网构软件的上述关键技术和支撑环境 SADE,网构软件系统的开发步骤描述如下.首先,利用 ODAM 方法学和 SADE 环境中的模型编辑器和模型检查器,对网构软件进行迭代的分析和设计,得到基于组织抽象的网构软件分析和设计模型.然后,利用 SADE 环境中的模型转换器,将网构软件的高层分析和设计模型转换为针对 SADE 平台的实现代码框架.第三,借助于 SADL、软件开发包以及 SADE 中的 SADL 语言编辑器和编译器,对网构软件代码进行精化、优化和完善,产生可运行网构软件程序代码.第四,将程序代码部署在 SADE 环境下运行.

由于篇幅限制,下面通过案例分析来讲解如何进行第三步骤的工作.我们开发了一个简单的网上交易系统来支持用户通过互联网进行交易.系统中有四类角色的网构软件 Agent:浏览者(Visitor)、买者(Buyer)、卖者(Seller)和管理者(Manager).浏览者可浏览和查看系统中的商品信息,但是他不能进行商品交易;买者拥有资金,他可以购买所需的商品;卖者拥有商品,他可以出售自己的商品;管理者对加入系统的交易者进行管理.下面两个典型场景用来展示处于动态、开放环境下网构软件 Agent 的自适应行为.

(1)用户绑定的角色随着环境的改变而变化:Tom 进入系统后,扮演 Visitor 角色浏览商品.当他感知到系统中出售他需要商品的服务出现,就扮演 Buyer 角色以购买商品.

(2)用户暂时离开某个角色,然后又恢复该角色:在 Tom 扮演 Buyer 角色准备购买需要的商品时,发现商品的售价大于它拥有的资金,如果此时 Tom 有要出售的商品,则加入 Seller 角色,通过销售商品获得更多的资金,而后再次扮演 Buyer 角色来购买商品.

首先,识别和抽象出构成系统的各个角色,并通过

继承角色抽象类对角色加以实现.在本案例中,我们设计了四类角色,包括:“Visitor”、“Buyer”、“Seller”和“Manager”.

其次,识别和分析系统中所需要的网构软件 Agent,定义网构软件 Agent 的属性,并进行初始化设置(指定网构软件 Agent 要绑定的角色的路径,策略配置文件和初始策略名),本案例中的 Agent 类图如图 5 所示.

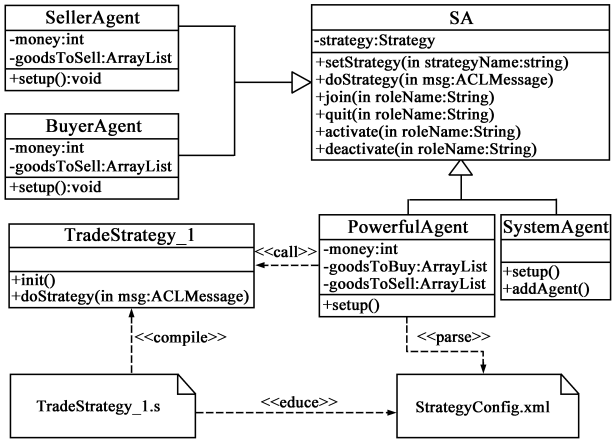

图5　网上交易系统的Agent类图

最后,使用 SADL 定义网构软件 Agent 的自适应自演化策略.在本案例中,我们将具有自适应能力的 Agent 抽象为 PowerfulAgent,其自适应自演化策略描述如图 6 所示.

```
1package examples.electricCommerce.strategies;
2import examples.electricCommerce.roles.*;
3//PowerfulAgent uses the strategy
4use examples.electricCommerce.agents.PowerfulAgent;
5program TradeStrategy_1 {
6    //Agent joins Visitor initially
7    InitStrategy { join(Visitor); }
8    strategy Wait {//Agent waits for service occurrence
9        when (SERVICE_EVENT(Sell, Book)) {
10           deactivate(Visitor); join(Buyer); } }
11   strategy Buy {
12       //Agent binds to various role for buying goods
13       when (INTERNAL_EVENT(LackOfMoney, Buyer)) {
14          if (self.goodsToBuy.length>0
15              && Seller@UNBOUND) {
16              deactivate(Buyer);join(Seller);
17          }
18       }
19       when (INTERNAL_EVENT(PurchaseOver, Buyer)) {
20          if (self.goodsToBuy.length > 0) {
21              deactivate(Buyer); activate(Visitor);
22          } else{ quit(Buyer); quit(Visitor); } }
23   }
24   strategy Sell {
25       //Agent changes role to prepare to buy again
26       when (INTERNAL_EVENT(SaleOver, Seller)) {
27          if (Buyer@INACTIVELY-BOUND) {
28              quit(Seller); activate(Buyer); } }
29   }
30}  //of program TradeStrategy_1
```

图6　PowerfulAgent的自适应策略

系统运行界面如图 7 所示,其中左上角的 Agents List 展示了当前系统中的所有应用相关 Agent,用户双击右上角的带有角色名的图片可以添加初始绑定该角色的 Agent,我们规定初始绑定 Visitor 角色的 Agent 为 PowerfulAgent.下面的每一栏从左至右分别展示了 Agent 的名字,当前加入的角色以及 Agent 参与交易和自适应动作执行的信息,角色图片背景为灰色说明该角色当前处于非活跃状态.

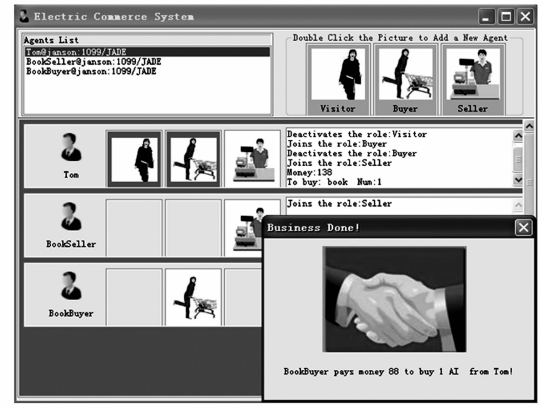

图7　网上交易系统运行结果

5　结论和进一步工作

近年来,如何支持复杂、自适应网构系统的开发成为人们关注的一项重要研究课题.现有研究方向之一是基于对象技术,通过反射机制实现自适应网构软件系统的开发.然而,由于对象固有的静态特性,这种技术并不能很好的适合动态和开放环境下自适应系统的实现.另一个研究方向是基于软件体系结构技术,这种技术着重关注系统整体的自适应,通过对软件体系结构中的部件和连结子的增加、删除、替换实现系统的自适应.研究方向之三是借助软件 Agent 技术,通过 Agent 在运行期间转换角色或类型实现系统的自适应.软件 Agent 技术被视为开发动态、复杂软件系统的有效泛型[11],Agent 的环境驻留性和行为自主性能够有效支持自适应性和自演化性.

当前有关自适应网构软件系统的研究往往将高层的分析和设计与底层的实现独立开发,缺乏统一的机制、方法学和语言设施来系统地支持自适应网构软件系统的开发.在方法学层面上,人们做了很多的努力,取得了许多成果[12,13].但是,高层的自适应描述和分析往往缺乏底层的实现支持.针对这一问题,我们开展了网构软件自适应和自演化的关键技术研究,提出了支持网构软件系统开发的方法学 ODAM 和支撑环境 SADE.不同于已有的方法,我们的研究成果具有以下一些特点:(1)基于 Agent 和组织抽象来对自适应和自演化网构软件系统进行分析和设计;(2)提出了支持对自适应和自演化进行描述、分析和构造的动态绑定机制;(3)主张将自适应逻辑和业务逻辑相分离,提出了自适应逻辑描述语言 SADL;(4)开发了一组集成的软件工具集合,来支持自适应和自演化网构软件的分析、设计和实现.

基于文中提出的网构软件系统开发的方法学 ODAM 和支撑环境 SADE,我们实现了网上交易系统、垃圾清理系统等自适应软件系统.这些开发实践和应用案例的分析说明了上述开发方法和支撑环境的可行性和有效性.

未来将在以下方面进一步展开研究:在机制层,对自适应机制进一步扩充,支持更多类型的自适应软件的开发;并对系统演化过程中出现的冲突问题进行深入研究,提出更为优化的解决方案;在应用层,我们希望通过更多的、系统所处环境更为复杂的案例的分析和研究来验证我们工作的有效性.

参考文献:

[1] 杨芙清,梅宏,吕建等.浅谈软件技术发展[J].电子学报,2002,30(12A):1901 – 1909.
Yang Fu-qing, Mei Hong, Lü Jian, et al. Some discussion on the development of software technology[J]. Acta Electronica Sinica, 2002, 30(12A): 1901 – 1909. (in Chinese)

[2] 吕建,马晓星,陶先平,徐锋,胡昊.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.
Lü jian, Ma xiaoxing, Tao xianping, Xu feng, Hu hao. Internetware: research and progress[J]. Science in China (Series E), 2006, 36(10): 1037 – 1080. (in Chinese)

[3] Ultra-Large-Scale Systems: The Software Challenge of the Future[R]. Software Engineering Institute, Carnegie Mellon University, 2006.

[4] 吕建,陶先平,马晓星,胡昊,徐锋,曹春.基于 Agent 的网构软件模型研究[J].中国科学 E 辑信息科学,2005,35(12):1233 – 1253.
Lü jian, Tao xianping, Ma xiaoxing, Hu hao, Xu feng, Cao chun. On agent-based software model for internetware[J]. Science in China (Series E), 2005, 35(12): 1233 – 1253. (in Chinese)

[5] Garlan D, Cheng S W, Huang A C. Rainbow: Architecture-based self-adaptation with reusable infrastructure[J]. Computer, 2004, 37(10): 46 – 54.

[6] 王千祥,申峻嵘,梅宏.自适应软件初探[J].计算机科学,2004,31(10):168 – 171.
Wang Qian-xiang, Shen Jun-rong, Mei Hong. An introduction to self-adaptive software[J]. Computer Science, 2004, 31(10): 168 – 171. (in Chinese)

[7] Xinjun Mao, Lijun Shang, Hong Zhu, Ji Wang. The adaptive casteship mechanism for developing multi-agent systems[J]. International Journal of Computer Applications in Technology, 2008, 31(1/2): 17 – 34.

[8] Xinjun Mao, Ji Wang. Engineering Adaptive Multi-agent Systems with ODAM Methodology[R]. Proceedings of Prima 2007, LNAI, Springer-Verlag Berlin Heidelberg, 2009. 380 – 385.

[9] 郝小雷,董孟高,毛新军,齐治昌.自适应 Agent 策略描述语言的设计及编译器实现[J].电子学报,2009,37(4A):65 – 69.
Hao Xiaolei, Dong Menggao, Mao Xinjun, Qi Zhichang. The design of self-adaptive agent strategy description language and the implementation of SADL compiler[J]. Acta Electronica Sinica, 2009, 37(4A): 65 – 69. (in Chinese)

[10] F. Bellifemine, G. Caire, A. Poggi, and G. Rimassa. : JADE: A white paper[J]. Telecom Italia Lab Journal EXP, 2003, 3(3): 6 – 19.

[11] N. R. Jennings. An agent-based approach for building complex software systems[A]. Communication of ACM[C]. New York: ACM, 2001. 35 – 41.

[12] Thomas Juan, Leon Sterling. A Meta-model for Intelligent Adaptive Multi-Agent Systems in Open Environments[A]. Proc. Of AAMAS[C]. New York: ACM, 2003. 1024 – 1025.

[13] Luca Cernuzzi , Franco Zambonelli. Dealing with Adaptive Multi-Agent Organizations in the Gaia Methodology[A]. Agent-Oriented Software Engineering VI[C]. Heidelberg, Springer Berlin 2005. 109 – 123.

作者简介:

毛新军 男,1970 年生于浙江江山.博士,教授,博士生导师,中国计算机学会会员.感兴趣的方向包括:面向 Agent 的软件工程、新颖软件开发学、分布计算技术、软件体系结构和模式等.
E-mail: xjmao@nudt.edu.cn

李学斯 男,1984 年 3 月出生于吉林梅河口.国防科学技术大学计算机学院硕士生,研究方向为软件工程、多 Agent 系统.
E-mail: lxsamao@163.com

尹俊文 男,1969 年生.国防科学技术大学计算机学院副教授,研究方向为软件工程.
E-mail: jwyin@nudt.edu.cn

董孟高 男,1979 年生于陕西高陵.国防科学技术大学计算机学院博士生,研究方向为软件工程,多 Agent 系统.
E-mail: mgdong@nudt.edu.cn

胡翠云 女,1985 年生于河南辉县.国防科学技术大学计算机学院博士生,研究方向为软件工程、多 Agent 系统.
E-mail: hcy56316@163.com

吴 斌 男,1982 年生于云南曲靖.国防科学技术大学计算机学院硕士生,研究方向为软件工程、多 Agent 系统.
E-mail: wubin_nudt@163.com