基于 Agent 的网构软件需求监控框架*

付凌霄 彭鑫 赵文耘

（复旦大学计算机科学技术学院 上海 201203）

摘要

网构软件所面临的复杂、开放和动态变化的运行环境使其运行时行为常常会偏离需求规约。已有一些研究工作提出基于目标模型和需求推理实现软件需求的运行时监控和自修复，但还缺少实现框架，特别是缺少符合网构软件分布式和社会化特性的需求监控实现方法。针对这一问题，提出一种基于 Agent 的网构软件需求监控框架。框架中的需求监控 Agent 通过非侵入的方式实现对作为其宿主系统的网构软件实体的监控和干预，并通过 Agent 间的通信和协作实现社会化的目标委托和协作监控。为了验证框架的有效性，通过报告一个案例分析，进行了有效性评估。

关键词 网构软件，需求监控，自适应，自修复，自治愈，Agent

文章类别：软件工程

An Agent-based Requirements Monitoring Framework for Internetware

Fu Lingxiao, Peng Xin, and Zhao Wenyun

（School of Computer Science, Fudan University, Shanghai 201203)

Abstract

Running in a complicated, open and highly-dynamic environment, Internetware systems are likely to deviate from their requirements specification. In recent years, there have been a series of researches on runtime requirements monitoring and self repairing based on goal-based requirements modeling, monitoring and reasoning. However, a practical implementation framework for requirements monitoring and repairing is missing, especially for the Internetware characteristics of distribution, sociality. In this paper, we propose an agent-based requirements monitoring framework for Internetware. On the basis of our previous work, which interpreted runtime goal lifecycle in terms of goal state machine to achieve finer-grained goal monitoring and reasoning, the agents we implemented are able to monitor host systems both on internal goal satisfaction and cross-agent goal delegation at runtime, and actuate repairing policies when requirements are deviated from in a non-intrusive manner. In respect of autonomy of each software entity participant within the system, the framework organizes agents in a decentralized way. Furthermore, cross-agent monitoring is designed to adapt in different distribution environments and minimize the impact to existing communication architecture. We've conducted a case analysis to evaluate the effectiveness of our framework and demonstrate the possiblity of utilizing it on a larger-scale project.

Keywords internetware; requirements monitoring; self-adaptation; self-repairing; self-healing; agent

* 基金项目：国家自然科学基金项目（90818009）

* 通信作者：彭鑫 pengxin@fudan.edu.cn

0. 引言

网构软件是一种 Internet 环境下，由具备主体化特征的软件实体通过各种协同机制实现跨网络的互连、互通、协作和联盟的新型软件形态[1-2]。网构环境中的软件实体具备 Agent 的特性：具有某种期望和目标，具备一定的能力和行为；可以通过社会化的协作完成更加复杂的目标；协作以自主和对等的协商和所做出的承诺为基础。这种分布式、社会化和自主特性是网构化软件区别于传统软件的典型特征之一。

随着 Internet 的飞速发展，越来越多的软件应用系统已经具备了典型的网构化应用形态。这种网构化应用运行在复杂、多变、开放的网络化环境中，其运行时行为可能经常偏离预定义的系统需求规约。另一方面，网构化软件经常包含跨越组织边界的软件实体之间的委托、协同与协作，体现出很强的社会化特性。这些都使得网构化软件的可信运行成为一个突出问题，甚至会导致服务失效、违反安全性策略等严重问题。为了保证网构化软件的运行时行为和质量符合相关涉众（stakeholder）的期望，网构化软件运行平台需要提供相应的运行时需求监控和质量保障机制。

对于软件运行时行为与需求规约的一致性问题，许多学者提出通过软件需求的运行时监控和自适应调整解决，从而保证软件的可靠运行。这方面的一类典型工作是在基于目标模型（goal model）[3]的系统需求基础上，通过监控、推理、诊断和修复实现运行时系统行为的调整和正确性保障[4-6]。其中目标表示一种系统或相关涉众希望达到或实现的目标和期望。这些工作通过目标模型、目标约束和规则刻画系统需求，然后在运行时监控、发现潜在的需求违反，并通过推理和调整减少系统运行时失效和错误的发生。

网构软件所具有的社会化和自主特性使得相应的运行时需求监控和质量保障机制需要以一种自治和自管理的方式实现。自治系统能够在运行时对内部行为和外部环境进行监控和适应，从而实现自配置、自优化、自治愈和自保护等自管理目标[7]。这种自治能力一般都是通过符合 MAPE（监控 monitoring-分析 analyzing-决策 planning-执行 executing）控制循环的自适应机制实现的，即在软件运行时持续收集和分析软件实体的实时状态并根据相关知识（如需求模型、演化规则等）进行自适应调整。

面向目标的需求工程方法[3][8]中已经存在 Agent 及其类似的概念，例如 i*[9]将 Agent 和 Actor 作为具有目标、能力、信念和承诺的意识主体。Tropos[8]基于 i*进行扩展，强调了整个软件工程周期中 Agent 的主体性。这些工作在概念层次上支持基于 Agent 的社会化协作，为面向目标的需求监控建立了理论和方法基础。然而现有的相关工作主要关注于其中的需求推理和诊断技术[4, 10-11]，并未给出相应的实现框架，特别是还缺少针对网构软件分布式和社会化特性的需求监控实现方法。

针对以上问题，本文提出了一种基于 Agent 的网构软件需求监控方法。该方法以 Agent 作为网构软件需求描述、实现及监控的基本单元，在我们前期工作所研究的基于目标状态机的软件运行时需求监控方法[12]基础上实现自治计算[7]中的自治元素自管理和对等交互结构。在实现框架方面，该方法应用多 Agent 系统中间件 JADE（java agent development framework）[13]，实现面向网构软件的需求监控和自修复。对于单个网构软件实体，该框架通过需求监控 Agent 以非侵入的方式实现对宿主系统（host systems）的监控和自修复操作。需求监控 Agent 基于目标状态机对需求目标实例进行监控，当发现潜在的需求违反时根据预定义的修复规则确定修复动作；另一方面，该框架通过 Agent 间的通信和协作实现多个网构软件实体之间的目标委托和监控交互。基于所提出的实现框架，本文还通过一个案例分析对方法和实现框架的有效性进行了评估。

1. 背景知识

1.1 目标模型

面向目标的需求工程（goal-oriented requirements engineering, GORE[3]）方法是需求工程研究中的活跃领域。GORE 使用目标模型描述通过 Agent 合作而期望满足的系统状态，其中 Agent 是拥有目标并可进行社会化交互、协作的主体。目标分为硬目标（hard goal）和软目标（soft goal）。前者的满足状况是绝对的二值选择（满足或不满足），而后者的满足状况则由相对程度表示。目标可以通过 And/OR 分解进行精化。硬目标通常会最终精化为一组可以直接实现或完成的任务（task）。因此整个目标模型完整描述了相关涉众的期望、能力、目标实现方式和依赖关系等信息，系统地刻画了需求的 Who/What/Why/How 关系。

本文的运行时需求监控和修复主要关注于功能性的硬目标，同时以 Agent 概念反映网构软件的主体化和社会化特性。图 1 是一个基于我们前期工作[12]中的案例修改的一个目标模型实例，其中描述了涉及三方的网构软件交互与协作场景，每个网构软件实体都作为一个独立的 Agent。该场景中，商品订购系统（COS，commodity ordering system）为用户提供商品订购功能，顶层目标“订购商品”被 And-分解为“选择商品”、“订单支付”以及“配送商品”3 个子目标。商品订购系统本身并不具备配送能力，因此“配送商品”子目标被委托给另外两个提供配送服务的区域配送中心（regional distribution center，RDC1 和 RDC2）。支持 Agent 概念的需求模型能够自然地描述网构软件的需求实现方式：COS 具有自己的业务目标（实现“订购商品”），自身具有一部分实现目标的能力（如“选择商品”），同时通过交互和协作达成一部分无法独立完成的子目标（如“配送商品”）。Agent 之间的协作体现出社会化特性： RDC1 或 RDC2 与 COS 之间存在约定和承诺，如收到请求后 24h 内完成商品配送；当 COS 获悉目标失效或与目标委托相关的承诺被违反时，可以通过 Agent 替换的方式选择其它 RDC 完成配送，例如将 RDC1 替换为 RDC2。

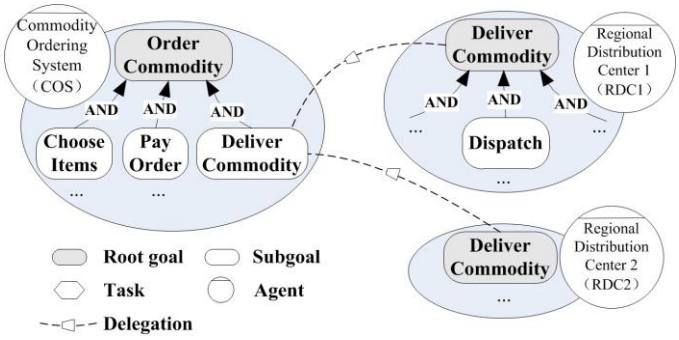

Fig. 1 Goal model for a commodity ordering system.

图 1 商品订购系统目标模型

1.2 需求监控与修复

目标模型是抽象层次较高的需求描述，其对应的需求规约可能会在软件实现及运行时与实际情况出现不一致，由此衍生出对软件系统需求监控的研究[4-6, 10-11]，以需求模型为基础对软件进行监控、分析和自适应决策。

在之前的工作中，我们提出并实现了一种基于状态化目标模型（stateful goal model）的监控方法[12]。软件系统中的每一个运行和会话实例都对应一次根目标的实现过程，具体表现为一个目标模型实例及其中各个目标对象在生存周期中的状态转换过程。我们使用目标状态机描述目标的运行时生存周期，其基本状态包括“未激活”、“已激活”、“执行中”、“挂起”、“已满足”以及“已失败”。目标在生存周期中存在诸多制约条件，分布在不同状态和状态迁移事件上，包括被激活时的上下文条件、执行前后的前/后置条件、激活后的承诺条件、以及多个事件间时序关系等，条件不满足意味需求被违反。插入宿主系统的探针（probes）实时监控系统行为的变化，并引发目标状态机演化。当需求规约被违反时，我们通过分析各目标状态和可用修复策略，选择适当的修复动作并执行。

修复策略的拟定总是优先考虑被违反的目标结点处的本地策略，包括阻止、重试和补偿。当本地修复不可用或无效时，违反消息沿目标模型结构和委托关系自底向上传播，直到找到可用的修复策略为止，以最小化修复开销。不同目标的约束条件违反，对应不同的修复策略。例如图 1 系统中，COS“支付订单”目标执行的后置条件“订单金额被完全支付并且得到系统确认”被违反时，本地修复策略是重试支付操作； RDC1 和 RDC2 接受委托并激活“配送商品”目标的上下文条件和商品种类和配送距离有关，当条件不满足时不予响应服务请求。当配送中心在配送服务过程中有目标被违反且本地修复策略无效时，委托方 COS 的“配送商品”目标也无法得到满足。通过全局修复策略，COS 可以转交服务请求给其它提供同样服务的有效 Agent。通过非侵入的方式和较低复杂度完成这类 Agent 协作场景的自适应，是本文实现框架的重要目的之一。

1.3 多 Agent 系统与网构化软件

在面向目标的需求工程中，Agent 是指具有目标、能力、信念和承诺的意识主体，其具体体现既可以是人或组织，也可以是软件组件或硬件设备。网构软件系统中涉及的软件实体可以看作是具备社会化属性的 Agent。因此，网构软件的特性使得面向 Agent 的需求建模方法及语言（如 i*）成为刻画网构软件实体需求、实现运行时需求监控和质量保障的自然选择。

为了使需求概念层面上的 Agent 能够更好地对应于实现框架中的基本单元，本文所提出的网构化软件需求监控框架使用多 Agent 系统开发框架 JADE[13]实现。多 Agent 系统是由多个软件 Agent 组成、通过相互间的通信和协作共同完成复杂任务的系统。JADE 是基于 Java、遵循 FIPA（the foundation for intelligent physical agents）规范的开发框架和运行平台，为多 Agent 系统开发提供了程序语言级的中间件支持。

基于 JADE 实现，将 Agent 概念在软件实体的底层实现和运行时抽象行为的表现进行统一，使软件实体成为具备社会化协作能力的自治单元，在运行时为各个宿主系统提供基于目标模型的需求监控和自修复支持。使用 JADE 实现的自治管理支持无缝的分布式跨平台通信，用于需求监控和自修复控制的 Agent 间通信基于 JADE 框架实现，完全独立于宿主系统的既有业务逻辑和通信机制。

2. 方法框架

2.1 框架概览

本文提出的面向网构软件的运行时需求监控框架 ARSGM（agent via runtime stateful goal model）如图 2 所示。图中每一个分布式结点上的宿主系统都是一个网构软件实体，在其上部署并运行一组用于需求监控和修复的 JADE Agent，使之成为具有运行时自适应能力的 ARSGM Agent（途中 Agent 图元标记的圆角矩形）。ARSGM Agent 一方面通过探针和效用器（actuators）对宿主系统的本地运行时行为进行监控和修复，另一方面通过 JADE 封装的通信机制实现多个 ARSGM Agent 之间的监控逻辑交互（agent Channel），不影响软件实体原有的业务逻辑交互（native channel），从而保证了对宿主系统的非侵入性。

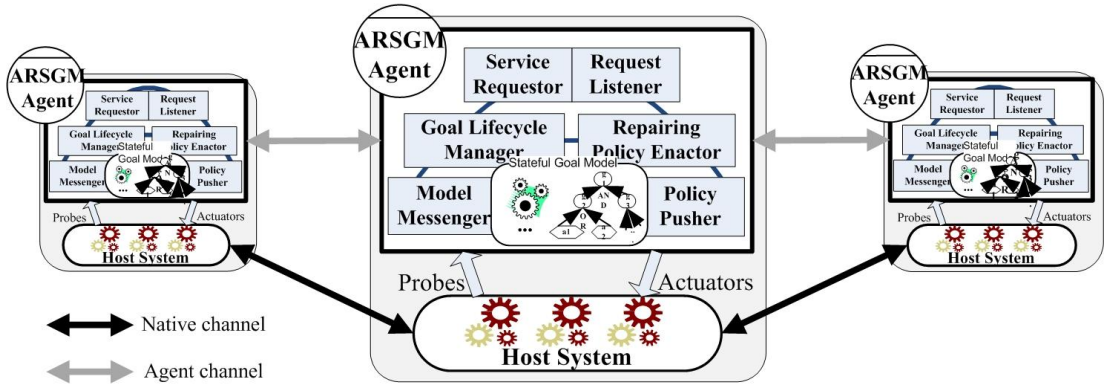

Fig. 2 Framework of agent-based requirements monitoring and self-adaptation.

图 2 基于 Agent 的需求监控及自适应框架

ARSGM Agent 的自治管理模块（圆角矩形内的粗边框矩形部分）实现包括基于状态化目标模型的自适应模块以及用于软件实体间服务通信的管理模块。两部分均由 JADE Agent 对象组成和实现。其中，自适应模块监控宿主系统目标的实现过程，为每一次目标实现过程创建包含一组目标的目标模型实例。该模块通过宿主系统提供的监控接口插入探针（probes）获取系统的运行时事件，系统事件按照映射规则转换为对应目标的状态机事件，触发目标状态推理，由此实现对目标生存周期的管理。在此过程中，目标状态推理根据目标规约分析可能的需求违反，并依据预定义的自修复策略规划必要的干预动作，由效用器通过宿主系统支持的干预接口执行修复动作。

ARSGM 通过接口扩展的方式实现非侵入式的自适应控制，宿主系统通过一些必要的扩展和配置后即可与所部署的需求监控 Agent 实现集成。该框架假设宿主系统能够提供所需要的事件捕获机制，并具备一些相关的调整和干预手段，例如可重配置的体系结构或参数等。

该框架符合文献[7]中所提出的自治软件系统基本结构。每个网构软件实体和自治管理模块一起构成了 ARSGM Agent 自治元素，其中目标模型、相关需求约束和自修复规则等构成了自适应所需的知识基础（knowledge base），而监控和修复过程则符合 MAPE 控制循环。另一方面，基于 Agent 的通信与协作实现了自治元素间的自主交互。

2.2 内部需求监控及自适应

ARSGM Agent 的自适应管理遵循经典的 MAPE 循环，基于状态化的目标模型，通过探针收集的宿主系统运行事件和状态，演化及推理系统运行时的需求满足情况，在系统行为与需求发生偏差时针对发生需求违反的原因选择对应的修复策略，并通过具体的效用器实施。用于宿主系统内部行为需求推理的自适应模块包括模型消息器（model messenger，MM）、目标生存周期管理器（goal lifecycle manager，GLM）、修复策略拟定器（repairing policy enactor，RPE）和决策结果推送器（policy pusher，PP）。作为需求监控的基础，状态化的运行时目标模型在第 2 部分已经介绍，并在文献[12]中有详细描述。

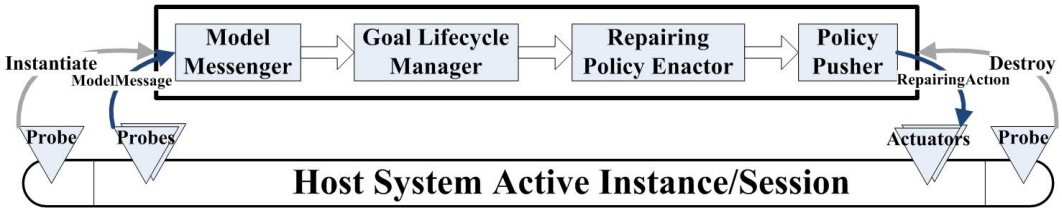

Fig. 3 Local monitoring structure of ARSGM.

图 3 ARSGM Agent 本地监控结构

Agent 内部的监控结构如图 3 所示。Agent 自治管理模块的生存周期与宿主系统保持一致，贯穿实例或会话的运行时阶段。附着于宿主系统的探针在会话实例开始和结束的同时创建和解构对应的自适应模块。在系统运行时，当自适应模块关注的事件发生时（如变量赋值、函数调用、函数返回、抛出异常等），探针将监控到的事件及相关系统状态（如函数参数，返回值，或可见的系统变量等）编码为状态化目标模型可以解析的模型消息，通过 JADE O2A（object to agent）通道传送给自适应模块中的 MM。MM 将解析后的监控数据传送给 GLM。基于状态化目标模型的规则，GLM 动态更新当前宿主系统运行实例对应的目标模型中各目标结点的状态机状态。若需求规约对应的目标约束条件被违反，则目标状态机将触发进入修复决策状态的事件，RPE 在此刻被调用，并按照文献[12]中的修复策略选择算法为被违反的目标选择修复策略。修复策略的拟定结果传递至 PP，PP 调用特定应用相关的效用器执行修复。

ARSGM 所采用的主要修复策略如表 1 所示。每种修复策略作用于不同宿主系统的不同目标时，其具体实现与特定应用场景高度相关，如编程语言、可访问资源以及所暴露应用程序接口等，因此作用于宿主系统的效用器的实现方式因场景而异，不在本文的重点讨论范围内。关于这些修复策略的详细描述参见文献[12]。

Table 1 Repairing Policies of ARSGM

表 1 ARSGM 修复策略

Repairing Policy	Policy Type	Description
Prevention	Local	Prevent system behavior corresponding to the ongoing state transition
Compensation	Local	Compensate the unsatisfied constraint by executing extra actions
Retry	Local	Retry system behavior corresponding to current state or transition
Goal Substitution	Global	Try to activate and execute an alternative or-branch who is available
Agent Substitution	Global	Delegate current goal to another agent who provides the same service

2.3 多 Agent 交互的自适应

宿主系统除了具备复杂的内部行为逻辑，在很多情况下还需要与其他软件实体通信，对于具备社会化特性的网构软件实体尤其如此。软件实体间关系建立在相关协议（agreement）的协商和维护上，通过遵循这些协议获取其它实体提供的服务或协作完成目标，ARSGM 管理基于这些协议的软件实体交互，但并不关心协议的商讨过程。为了以轻量级的方式对宿主系统进行自适应控制，特别是支持组件服务的框架和网构软件，ARSGM 并不破坏宿主系统原有的通信协议与交互方式。因此，宿主系统在进行服务调用或其他通信动作的同时，ARSGM 自治管理模块之间会同步相对独立的 Agent 交互通道（图 2 中灰色和黑色箭头）。宿主系统间的业务通信平行于自治模块间的通信，但是受服务请求器（service requestor，SR）和请求监听器（request listener，RL）监控和管理。

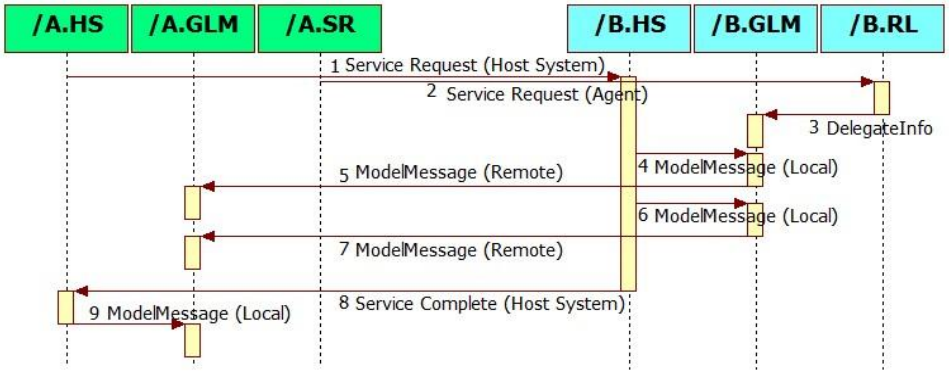

Fig. 4 Communication between ordinary ARSGM Agents.

图 4 ARSGM Agent 通信模式

如图 4 所示的两个一般 ASRGM Agent 的通信模式，当它们的宿主系统通过 IIOP 或 SOAP 消息建立服务组件或 Web 服务连接时，这些软件实体也建立了抽象的目标委托关系。这种抽象关系被 ARSGM 自治管理模块具体化。目标委托关系涉及双方 Agent 的需求满足，服务请求方的服务请求器（A.SR）与服务提供方的请求监听器（B.RL）随宿主系统服务请求关系的建立而进行通信（1，2）。A.SR 与 B.RL 建立连接的结果是 A 和 B 自适应模块中的 GLM 相应获得对方 Agent 的信息（3）。B 的目标状态变化随 B 运行时行为改变，并影响 A 所委托目标的状态（4、5、6、7）。宿主系统间的服务调用完成也对应其目标委托的完成（8、9）。若 B 在被调用过程中目标被违反且本地修复失败，则 B.GLM 通知 A.GLM 目标委托失败（可能发生为 5、7 处），A.RPE 随即被触发并尝试 Agent 替换。

ARSGM Agent 间通信模式的设计和实现主要基于网构化环境中分布式和社会化特性监控的考虑：1）分布式软件实体存在多种交互模式和方式，如层次结构型、点对点、同步/异步等。监控通信平行于业务通信，可以局部化监控信息以减少各实体相应的目标监控和修复开销。ARSGM Agent 间的通信服务于目标状态机监控，仍然基于宿主系统的运行时事件，例如图 4 中宿主的服务请求与完成可以映射为一个同步远程函数的调用和返回，也可以是一组操作的始末，这种灵活性和松耦合性消除了宿主系统间通信对自适应开发框架通信能力的依赖[14]；2）软件实体间通过社会化协商达成的协议往往体现为宿主系统目标需求规约的一部分，如图 1 中 RDC1 向 COS 做出承诺，收到请求后 24h 内完成商品配送，则“24h 内完成”是 COS“配送商品”目标的承诺条件之一，自治管理模块在收到服务请求的监控信息后会实时检查这一约束的满足或违反；3）当目标被违反需要寻找新 Agent 进行替换时，同样涉及实体间的社会化协商和交互。具体的寻找有效 Agent 和 Agent 替换动作与特定应用相关，修复策略应用后导致的运行时行为及监控模式仍符合图 4 所示，保持了监控逻辑的一致性。

自治管理模块间的通信运行在 JADE Runtime 之上，自动选择 HTTP、IIOP 和 SMTP 共 3 种传输协议中的最优协议。这种与宿主系统原有业务逻辑剥离的通信模式能够以宿主系统不知觉的方式，在必要时对目标委托关系实现服务提供方替换。因此，使用 ARSGM 框架，在软件架构、抽象模型和自适应逻辑中以自然的 Agent 语义实现了 Agent 替换修复策略。

2.4 容器属性 Agent 间的自适应控制

ARSGM 实现的需求监控框架强调对等的分布式自适应控制。对等概念体现在软件实体的生存周期和系统架构上。ARSGM Agent 自治管理模块的生存周期与宿主系统一致，在运行时只能与同一时刻存在的其它活跃 ARSGM Agent 交互。

对于特定的系统框架，例如基于 IIOP 的组件服务，提供服务的远程对象在实例化后需

要在 ORBD 上进行注册，从而使自己对服务请求方可见。注册后的远程对象依次响应组件服务请求并返回执行结果。对这类系统应用 ARSGM 框架，每个远程对象都对应一个 ARSGM Agent 对象；远程对象进行服务注册时，其对应的自治管理模块也会在 JADE runtime 上注册服务，使宿主系统在 Agent 层上也可见；远程对象的每一次服务响应都是一次新的会话（session），对应独立的 ARSGM Agent 实体。从这一层面看，这类软件实体具备**容器**的属性，其通信模式如图 5 所示。当 Agent 试图与容器属性软件实体交互时，Agent 容器会创建新会话并实例化 Agent 实体，新实例化的 Agent（对应新会话）实际与发起服务请求的 Agent 之间建立对等的交互连接。这一特性在 Web 服务密集的 SOA 及网构软件架构中体现得更为明显：Web 服务的调用方发送服务调用请求后，承载 Web 服务的服务引擎接收并解析请求消息，然后根据请求对象将已部署的服务对象实例化，服务对象调用所请求的服务接口后将返回值返回。因此对 Web 服务场景应用 ARSGM 框架时，将 Web 服务引擎如 AXIS 和 JBOSS 等作为具备容器属性的 Agent，从而实现对 Web 服务实体及服务请求方实体的自适应控制。

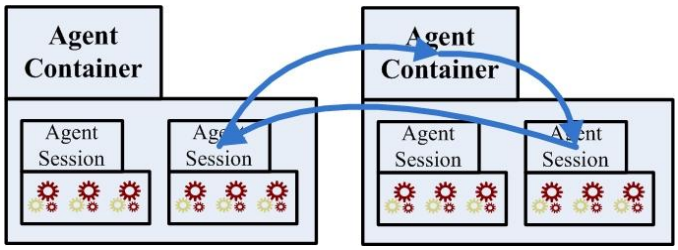

Fig. 5 Communication between container ARSGM Agents.

图 5 容器属性 ARSGM Agent 通信模式

3. 案例分析

为了验证 ARSGM 框架的可行性和有效性，我们将其应用于一个作为网构化软件实例的在线商品订购系统。这一部分将介绍案例应用场景，并对框架的有效性进行评估。

3.1 案例系统实现

我们将第 2 节中的商品订购系统目标模型扩展后并使用 Java 实现，软件实体间的通信基于 RMI-IIOP（remote method invocation over internet inter-orb protocol，提供较 RMI 更灵活、较整 CORBA 更轻量级的分布式支持），以模拟网构化环境中的业务场景。扩展后的目标模型如图 6 所示。该场景涉及 3 类系统参与软件实体，分别是商品订购系统（COS），区域配送中心（RDC），以及配送单元（delivery unit，DU）。

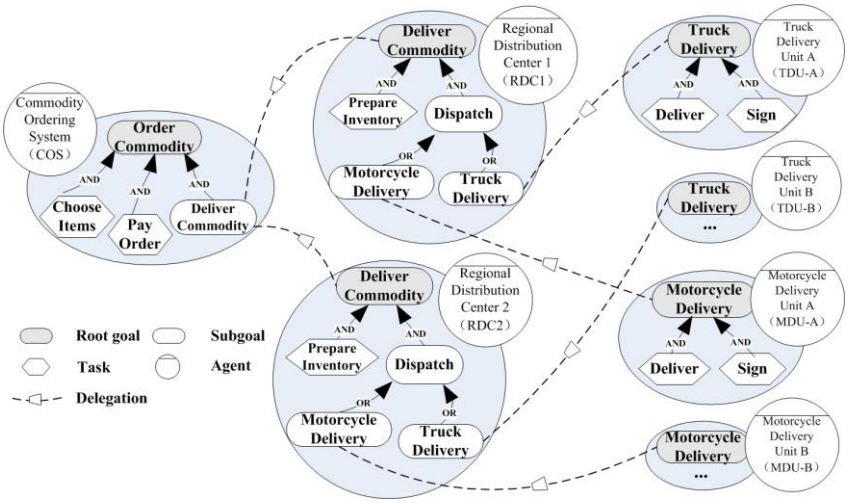

Fig. 6 Goal model for a commodity ordering scenario.

图 6 商品订购场景目标模型

每类参与实体的实例在运行时相对独立并且可交互。COS 是具备图形界面的 Java 程序，运行终端用户通过交互完成“选择商品”、“支付订单”等操作。用户完成支付操作后，COS 立即激活并开始执行“配送商品”目标。提供配送服务的 RDC 实现为 IIOP 远程对象，通过在 ORBD 上注册对象，为调用者提供组件服务。RDC 通过两种方式满足“物品派送”目标，分别是“摩托车送货”和“货车送货”。送货目标进一步委托给相应的 DU，DU 负责将商品送至顾客指定的地址并确认顾客签收商品。本例中 DU 的实现也为 IIOP 远程对象，模拟现实业务流程中配送单元的社会化特性，即配送中心选择可用的配送单元完成配送服务，配送单元受配送中心委托后依次进行商品配送。我们实现了两个区域配送中心（RDC 1 和 RDC 2）软件服务，每个 RDC 又关联两个配送单元分别承担摩托车运输和货车运输。

3.2 应用 ARSGM 框架

通过上述商品订购系统与本文的 ARSGM 框架整合，图 6 中 Agent 对应的软件实例将在运行时具备图 2 所示的自治结构和特性。使用 ARSGM 需要下列工作：

1）对网构软件系统进行需求建模。遵循文献[12]中状态化目标模型描述，网构系统的需求描述包括目标模型的结构信息、目标状态机的外部事件（对应探针监控下的宿主程序行为）和内部事件（对应其他目标结点的状态演化）映射关系、目标约束条件以及条件违反时对应的修复策略和效用器操作等。

2）描述软件实体间服务请求和调用的相关信息。这部分信息是 ARSGM 服务请求器、请求监听器对目标委托关系进行自适应控制的知识基础。1）和 2）的信息编码为 xml 文件，作为 ARSGM Agent 生存周期伊始的输入。本例中，图 6 的每个 Agent 都有各自的目标模型和服务描述文件，描述文件通过硬编码的方式创建。

3）获取与宿主系统实现相关的探针和效用器设施。本例中，为了最小化整合 ARSGM 框架而对宿主系统的改动，我们使用 AspectJ[15]对图 6 实现进行代码插装（instrumentation）。AspectJ 是 Java 平台的面向方面实现，提供了非侵入式的关注点分离解决方案。我们实现了一个小工具，可以根据 1）和 2）所得的目标模型和服务描述文件，半自动化生成适用于宿主系统的 AspectJ 文件。该文件编织进宿主系统代码，通过通知（advice）代码实现探针功能，并在外部事件映射的连接点（joinpoint）被触发，将现场信息通过消息方式传送给 ARSGM 模型消息器。效用器由显式的宿主系统 API 简化实现。

4）将 ARSGM 框架实现的 jar 包、目标模型和服务描述文件以及工具生成的 AspectJ 文件包含进宿主系统的构建路径中并重新构建项目，完成 ARSGM 框架的整合。

3.3 案例场景

ARSGM Agent 在运行时实时监控宿主系统的内部行为和外部交互。监控下的商品订购系统在其中一次运行时记录的部分自治管理模块事件及对应的系统行为如表 2 所示。

Table 2 Monitored Behaviors of Commodity Ordering System at Runtime (partial)

表 2 商品订购系统行为监控数据（部分）

Timestep	RDC 1		COS	
	System behavior	ARSGM event	System behavior	ARSGM event
1	RDC-1.init	launchPlatform	/	/
2	Context.rebind	serviceRegister	/	/
12	/	/	main entry	launchPlatform
13	/	/	COS.init	modelInstantiate
14	/	/	Choose.onClick	ChooseItems.start
21	/	/	[rmCall] deliver	serviceRequest
22	[rmCall] deliver	modelInstantiate	/	/

表 2 中第 1 列 TS 表示发生监控事件的相对时间步（timestep）。RDC-1 实例初始化时，ARSGM Agent 在 JADE Runtime 上创建 JADE 平台，表示该远程对象实例的自治实体（1）。远程对象在 ORBD 上注册服务的同时，Agent 层面也相应注册（2）。当 COS 系统进入主函数入口时，对应的 JADE 平台也随之建立（12）；绑定的 ARSGM Agent 设施在 COS 类初始化时创建对应的目标模型实例，表示该实体成为活跃 Agent（13）。COS 目标模型创建后默认激活“商品订购”和“选择商品”目标。用户点击按钮进入商品选择视图，对应“选择商品”目标开始执行（14）。用户完成商品选择和支付后，COS 默认调用 RDC-1 服务实现“商品配送”目标（21），Agent 层也对应发起服务请求。RDC-1 组件服务被调用，新的活跃会话（Session）被建立（22），同时，请求监听器在收到 COS Agent 的服务请求后，将其 Agent 信息告知新会话对应的实体，该次组件服务的调用状态随目标模型的状态演化实时反馈至 COS Agent。

3.4 需求监控与修复

我们手动为商品订购系统注入运行时错误，以检验 ARSGM 框架在处理运行时需求违反情况、尤其是涉及多 Agent 交互时的有效性，结果如表 3 所示。

Table 3 Event Sequence Corresponding to Goal Constraints Violation (partial)

表 3 需求违反情况下目标模型的事件序列（部分）

Timestep	Goal	Event	Repairing Policy
1	TDU-A.Deliver	CCViolated	n/a
2	TDU-A.TruckDelivery	SubFail	n/a
3	RDC-1.TruckDelivery	DelegateFail	n/a
4	RDC-1.Dispatch	SubFail	goalSubstitution
5	RDC-1.DeliverCommodity	SubFail	n/a
6	COS.DeliverCommodity	DelegateFail	agentSubstitution
7	RDC-2.DeliverCommodity	modelInstantiate	

文献[6]对 Agent 内部需求违反实时的自适应有详细讨论，因此本文主要考虑涉及多 Agent 委托关系的修复场景。表 3 记录了由配送单元引发的目标委托失败时自适应模块处理的状态化目标模型事件以及相应采取的修复策略。相对时间步从需求违反触发的事件开始计算。本例中，COS 默认调用 RDC-1 提供配送服务，RDC-1 默认选择货车送货方式。我们将货车运送单元 TDU-A 的配送任务承诺条件设定为“30 秒内完成”，并且在其对应的函数体内加入额外代码使其响应时间超过 30s。当 TDU-A 执行“送货”任务时间到达 30s 但尚未返回，自适应模块侦测到需求被违反，触发“送货”任务的 CCViolated 事件（1）。RPE 发

现没有可用的策略对应“送货”任务的承诺条件违反，任务状态机迁移至失败状态，并触发 TDU-A“货车运输”目标的 SubFail 事件（2）。TDU-A 的“货车运输”目标是 And 分解顶层目标，没有可用修复策略，迁移至失败状态后触发 RDC-1“货车运输”目标的 DelegateFail 事件（3）。RDC-1“货车配送”目标没有可用的配送单元以委托服务，于是将失败事件上传至 RDC-1“物品派送”目标（4）。“物品派送”目标被 OR 分解，可以选择目标替换修复策略，但是由于货物属性不满足“摩托车运输”目标激活的上下文条件（货物尺寸超过阈值），因此策略不可用，目标违反继续向上传播至委托服务的 COS“配送商品”目标（5，6）。COS 中的 RPE 通过服务请求器发现 RDC-2 的配送服务可用，因此选定 Agent 替换作为修复策略。COS 的 PP 通过效用器获取 RDC-2 的远程对象并进行服务调用，RDC-2 远程对象收到服务请求并响应，会话对应的 Agent 实体被创建（7）。

RDC-2 远程对象调用正常返回后，其返回值由 COS Agent 的效用器通过系统接口反馈至宿主系统。之前被诊断为目标失败的 RDC-1 和 TDU-A 则自动结束当前会话并丢弃返回值。对于 COS 而言，其配送服务在首选委托对象无法满足目标时，由 ARSGM 框架完成了不知觉的自适应控制，保证了服务的可靠性。

4. 相关工作与讨论

在本文实现的基于 Agent 的软件需求监控及修复框架中，自适应分析和决策知识基于我们之前所研究的状态化目标模型。与文献[12]中实现的集中式自适应框架不同，所有被监控系统的运行时目标模型信息都保存在专用的监控服务器端，基于目标模型的分析和推理工作也发生在服务器端，本文的实现框架将每个分布式系统中的软件实体都 Agent 化，更接近文献[7]描述的自治计算系统中的自治元素，也更能体现软件实体间交互的社会化。

使用需求模型为基础对运行时系统进行自适应是活跃研究领域。Feather 等人[5-6]在面向目标的需求分析方法发展的早期，便提出了基于目标驱动的需求规约和形式化监控逻辑描述相结合的方法，为系统运行时行为和需求的动态调和提供支持。但是他们的研究主要关注由系统运行时环境改变造成的需求满足偏差。近年来，目标模型推理被应用在需求驱动的自适应方法中。Wang 等人[4, 10]提出了一种粒度可调的软件自适应框架，当监控到目标违反时，使用 SAT Solver 诊断可能导致错误的目标，然后根据诊断结果选择对软目标贡献度最大的目标配置，应用在系统的下一次会话中。Dalpiaz 等人[11]提出的软件自配置方法基于 Tropos 目标模型，具有较强的社会性，并且其目标推理算法蕴含了状态化目标的概念，但是该方法抽象层次较高，缺少完整的程序实现。

随着 Web Serive 环境的兴起，专门针对 Web Serive 的软件自适应也呈现出很多研究问题和成果。Liu 等人[16]通过扩展 BPEL 语言及解释引擎，通过声明式的方式为 BPEL 流程中出现的异常情况提供 SKIP，RETRY，IGNORE 和 ALTERNATE 预定义修复动作，提高了 BPEL 流程的可靠性。Chen 等人[17]的研究结合 Web Service 和目标模型，考虑系统中软目标描述的各类 QoS 属性，通过权衡动态选择合适的 Web 服务。本文实现的自适应框架支持 Java 实现的本地系统和部分分布式系统，并且通过案例分析的模拟，初步验证了对 Web Service 系统支持的可行性。

多 Agent 系统是用以实现（网构环境下）自治计算的理想途径之一。IBM 提出了一种基于多 Agent 系统实现的自治计算实现架构[18]。该方法从数据中心服务器的原型实例出发，更加关注与服务质量相关的资源分配，以进行启动时的自配置和运行时的自优化，而非本文更加通用的功能性目标监控与自修复；此外，独立资源仲裁元素（resource arbiter）的存在使得该架构并未实现真正的分布式。与之类似，文献[19]提出的网构软件协作框架中，存在专用协调构件（facilitator）进行环境信息的中转和推理，以引导自主构件间的协作关系；文献[20]通过 Agent 学习的方式提出了网构软件构建中提高可用性的优化方案。文献[14]通过

阐述自治计算、多 Agent 系统及面向服务计算三者间的类比关系，讨论了在 SOA 环境下使用 Agent 技术开发自治计算系统的意义及问题。ARSGM 中的软件实体自适应反馈控制和 JADE 平台整合，是对这一概念的初步实践，在保证开发框架非侵入性和可用性的基础上，为更复杂的计算提供了可能性。

5. 结束语

本文提出并实现了一种可以应用于网构软件系统的自适应框架原型。网构化软件实体具备 Agent 特征，我们的框架为这些实体的运行时自治管理提供了支持。使用状态化的目标模型，软件的运行时行为对系统需求的影响被监控、分析。JADE 实现的自适应控制模块将软件实体 Agent 化，在分布式、社会化场景中对软件实体的委托和协作进行对等的监控及修复。通过自适应框架对分布式实例的应用，我们讨论了非侵入式的框架整合方式以及运行时需求监控和 Agent 交互管理的有效性。在未来的工作中，我们将对更大规模的真实网构软件系统进行案例研究，并在需求监控方法中加入对系统软目标的支持。

参考文献

[1] Mei Hong, Huanggang, Zhao Haiyan, et al. An architecture based software development approach for internetware [J]. Science in China: Series E Information Sciences, 2006, 36(10): 1100-1126 (in Chinese)

(梅宏, 黄罡, 赵海燕, 等. 一种以软件体系结构为中心的网构软件开发方法[J]. 中国科学: E 辑信息科学, 2006, 36(10): 1100-1126）

[2] Lü Jian, Ma Xiaoxing, Tao Xianping, et al. The research and development on the internetware [J]. Science in China: Series E Information Sciences, 2006, 36(10): 1037-1080 (in Chinese)

(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] Lamsweerde A V. Goal-oriented requirements engineering: a guided tour [C] //5th IEEE Int Symp on Requirements Engineerin (RE'01). Toronto, ON: IEEE, 2001: 249-262

[4] Wang Yiqiao, Mylopoulos J. Self-repair through reconfiguration: a requirements engineering approach [C] //24th IEEE/ACM Int Conf on Automated Software Engineering (ASE 2009). Auckland: IEEE, 2009: 257-268

[5] Fickas S, Feather M S. Requirements monitroring in dynamic environments [C] //2nd IEEE Int Symp on Requirements Engineerin (RE'95). York: IEEE, 1995: 140-147

[6] Feather M S, Fickas S, Lamsweerde A V, et al. Reconciling system requirements and runtime behavior [C] //9th Int Workshop on Software Specification and Design (IWSSD 1998). Ise-shima: IEEE, 1998: 50-59

[7] Kephart J O, Chess D M. The vision of autonomic computing [J]. IEEE Computer, 2003, 36(1): 41-50

[8] Bresciani P, Perini A, Giorgini P, et al. Tropos: an agent-oriented software development methodology [C] //3rd Int Joint Conf on Autonomous Agents and Multiagent Systems (AAMAS 2004). New York, NY: Speinger, 2004: 203-236

[9] University of Toronto, KM lab. i*: Intentional Strategic Actor Relationships Modeling [EB/OL]. www.cs.toronto.edu/km/istar

[10] Wang Yiqiap, Mcllraith S A, Yu Yijun, et al. An automated approach to monitoring and diagnosing requirements [C] //24th IEEE/ACM Int Conf on Automated Software Engineering (ASE 2007). Atlanta, GA: IEEE, 2007: 293-302.

[11] Dalpiaz F, Giorgini P, Mylopoulos J. An architecture for requirements-driven self-reconfiguration [G] // LNCS 5565: Advanced Information Systems Engineering, 21st Int Conf (CAiSE 2009). Berlin: Springer, 2009: 246-260

[12] Fu Lingxiao, Peng Xin, Yu Yijun, et al. Stateful requirements monitoring for self-repairing of software systems: Technical Report FDSE-TR201101 [R/OL]. Software Engineering Lab, Fudan University, 2011, http://www.se.fudan.sh.cn/paper/techreport/1.pdf

[13] Bellifemine F, Caire G. Greenwood D. Developing multi-agent systems with JADE [M]. Hoboken: John Wiley & Sons, 2007

[14] Brazier F M T, Kephart J O, Parunak H V D, et al. Agents and service-oriented computing for autonomic computing: a research agenda [J]. IEEE Internet Computing, 2009, 13(3): 82-87

[15] The Eclipse Foundation. The AspectJ Project [CP/OL]. www.eclipse.org/aspectj

[16] Liu An, Li Qing, Huang Liusheng, et al. A declarative approach to enhancing the reliability of BPEL processes [C] //2007 IEEE Int Conf on Web Services (ICWS 2007). Salt Lake City, UT: IEEE, 2007: 272-279

[17] Peng Xin, Chen Bihuan, Yu Yijun, et al. Self-tuning of software systems through goal-based feedback loop control [C] //18th Int Requirements Engineering Conf (RE 2010). Sydney, NSW: IEEE, 2010: 104-107

[18] Tesauro G, Chess D M, Walsh W E, et al. A multi-agent systems approach to autonomic computing [C] //3rd Int Joint Conf on Autonomous Agents and Multiagent Systems (AAMAS 2004). New York, NY: IEEE, 2004: 464-471

[19] Liu Wen, Sun Xi, Jiao Wenpin, et al. An autonomous-component-based cooperation framework for internetware [J]. Journal of Computer Research and Development, 2006, 43(Z1): 217-221 (in Chinese)

(刘文, 孙熙, 焦文品, 等. 一种基于自主构件的网构软件协作框架[J]. 计算机研究与发展, 2006, 43(Z1): 217-221)

[20] Zhuang Lei, Sun Xi, Zhou Li, et al. An agent-based approach to automating the composing of internet-based systems with high availability [J]. Journal of Computer Research and Development, 2006, 43(Z1): 195-199 (in Chinese)

(庄磊, 孙熙, 周立, 等. 一种基于 Agent 的高可用性网构软件自动构建方法[J]. 计算机研究与发展, 2006, 43(Z1): 195-199)

FU Lingxiao was born in 1987. He is a M.S. candidate at Fudan university. His research interests include Self-Adaptive Systems , Autonomous Computing, etc.

付凌霄(1987-), 男, 四川自贡人, 复旦大学硕士研究生, 主要研究领域为自适应系统, 自治计算等。

PENG Xin was born in 1979. He received the Ph.D. degree from Fudan University in 2006. He is a associate professor at Fudan University. His research interests include software maintenance, adaptable software, software reuse, etc.

彭鑫(1979-), 男, 湖北黄冈人, 2006 年在复旦大学获得博士学位, 现为复旦大学计算机学院副教授、硕士生导师, 主要研究领域为软件维护, 自适应软件, 软件复用与产品线。在 RE、ICSM、ICSR、WCRE、IST Journal、JCST、计算机学报、软件学报、电子学报、计算机研究与发展等国内外会议及期刊上发表论文 30 余篇, 主持自然科学基金项目 1 项, 参加 863 项目多项。CCF 高级会员,会员号:E200010133S。

ZHAO Wenyun was born in 1964. He received the M.S. degree from Fudan University in 1989. He is a professor and doctoral supervisor at Fudan University. His research interests include software engineering, electronic commerce, etc.

赵文耘(1964-), 男, 江苏常熟人, 1989 年在复旦大学获得硕士学位, 现为复旦大学计算机学院教授、博士生导师, 主要研究领域为软件工程、电子商务。在 RE、ICSM、ICSR、WCRE、IST Journal、JCST、计算机学报、软件学报、电子学报、计算机研究与发展等国内外会议及期刊上发表论文 50 余篇, 主持自然科学基金项目 2 项、863 项目多项。CCF 高级会员。