软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2015,26(4):760−777 [doi: 10.13328/j.cnki.jos.004757]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

一种基于智能体技术的软件自适应动态演化机制*

李青山[1,2,3], 王 璐[1,3], 褚 华[1,2], 张 曼[1,2]

[1](西安电子科技大学 软件工程研究所,陕西 西安 710071)

[2](西安电子科技大学 软件学院,陕西 西安 710071)

[3](西安电子科技大学 计算机学院,陕西 西安 710071)

通讯作者: 李青山, E-mail: qshli@mail.xidian.edu.cn, http://www. xidian.edu.cn

摘 要: 针对分布式软件系统在动态演化中面临的原有软件单元难以重用、忽视软件内部运行状态引发的演化需求等问题,借助智能体(agent)具有的环境适应性、变化敏感性等特征,提出了一种基于智能体技术的软件自适应动态演化机制.通过将软件单元封装为 Agent,并定义单元间的演化规则,使演化机制重用原有软件单元.通过一种基于数据推送的动态环境感知方法实现 Agent 间协作关系调整,同时满足来自内外部环境引发的动态演化需求.通过引入信息中介服务,实现了基于改进合同网的 Agent 协作策略,能够自适应地更替 Agent,满足用户意愿变更引发的动态演化需求.依据演化机制在支撑环境中的运行情况及相关能力指标的分析,说明所提出的演化机制适用于动态复杂的分布式软件系统,是一种有效的软件自适应动态演化机制.

关键词: 分布式软件系统;智能体;自适应软件;动态演化;环境感知

中图法分类号: TP311

中文引用格式: 李青山,王璐,褚华,张曼.一种基于智能体技术的软件自适应动态演化机制.软件学报,2015,26(4):760−777. http://www.jos.org.cn/1000-9825/4757.htm

英文引用格式: Li QS, Wang L, Chu H, Zhang M. Agent-Based software adaptive dynamic evolution mechanism. Ruan Jian Xue Bao/Journal of Software, 2015,26(4):760−777 (in Chinese). http://www.jos.org.cn/1000-9825/4757.htm

Agent-Based Software Adaptive Dynamic Evolution Mechanism

LI Qing-Shan[1,2,3], WANG Lu[1,3], CHU Hua[1,2], ZHANG Man[1,2]

[1](Software Engineering Institute, Xidian University, Xi'an 710071, China)

[2](School of Software, Xidian University, Xi'an 710071, China)

[3](School of Computer Science and Technology, Xidian University, Xi'an 710071, China)

Abstract: To tackle problems in the dynamic evolution of distributed software systems such as the difficulty in reusing the original software units and the neglect of evolution demand caused by internal running state, this paper proposes an agent-based software adaptive dynamic evolution mechanism, by means of the environmental adaptability, the sensitivity of changes, and other characteristics of agents. By packaging software units as agents and defining evolution rules among units, original units can be reused in such a framework. Using the dynamic environment awareness method based on data push, the collaborative relationships between agents can be adjusted, and thus the evolution requirements from both the external environment and the internal state are met. With the introduction of information intermediary services, the collaboration strategy of agents based on a modified contract net is implemented such that the agents can be changed adaptively and the evolution requirement from the users demand is satisfied. The operation performance of evolution mechanism in the environment and the analysis of the related capacity indexes demonsrate the proposed evolution mechanism is applicable to the dynamic and complex distributed software systems and is an effective software adaptive dynamic evolution mechanism.

* 基金项目: 国家自然科学基金(61173026, 61373045, 61202039); 国家高技术研究发展计划(863)(2012AA02A603); 中央高校科研业务基金(K5051223008, BDY221411); 国防"十二五"预研项目(513***103E)

收稿时间: 2014-07-30; 修改时间: 2014-10-14; 定稿时间: 2014-11-14

Key words: distributed software system; agent; adaptive software; dynamic evolution; environmental awareness

软件演化是指软件持续变化,并达到人们期望形态的过程.从 20 世纪 70 年代至今,众多学者和机构致力于软件演化的研究.早期研究多关注于静态演化,适用于静态封闭环境下的软件系统[1].然而许多重要领域的关键系统无法以停止、更新或重启等静态方式实现演化,因此需要从自适应角度进行研究,使这些关键系统能够根据变化,自适应地调节自身行为,进而实现动态演化.然而,基于传统的软件再工程或自适应中间件[2]的方法已经难以满足开放、动态环境下软件自适应动态演化的需求[3,4],因此,将能够自主学习并适应环境的智能体与软件自适应演化相结合,通过智能体所具有的环境感知性、自主调节性[5]等特点提高软件的自适应演化能力,已成为目前的研究热点.

随着网络技术的不断发展,目前,诸如全球联网的分布式金融系统、交通管制系统、指挥控制系统等资源异构且组织机构分散的分布式软件系统(distributed software system,简称 DSS)已被广泛地应用在各个领域.各节点实现方法多样、软件结构庞大复杂[6]等特点,使得分布式软件系统在面对来自环境或用户意愿变化而引发的动态演化需求时,存在着有别于一般软件系统的两个关键问题:首先,由于分布式软件系统各节点的实现方法可能不同,因此,以修改代码的方式实现演化[7]将难以完全重用原有的各种软件单元;其次,分布式软件系统本身结构的复杂性,导致了软件内部的变化也会引发软件演化.

目前,从模型驱动[8,9]、控制工程[10]、软件体系结构[11,12]等角度出发实现的分布式软件系统动态演化方法,一般以修改软件的体系结构或设计模型等软件再工程手段使系统具有自适应性,并且这些演化方法多采用对象或构件等静态非自主的实体封装分布式软件单元,不利于自主地、动态地感知软件内部运行状态的变化.

来源于分布式人工智能领域的智能体具有感知性、适应性、自治性和协作性等特征,使其非常适用于分布式软件系统,因此存在着利用智能体技术实现分布式软件系统动态演化[13,14]的相关研究成果.然而,这些方法一般将智能体作为类似对象或构件的实体,通过改变单元本身的设计和实现,将原有软件彻底改造成多 Agent 系统,从而使这些软件具有自适应演化能力,因此需要重新开发软件,而无法完全重用原有的计算单元;其次,针对环境变化而产生的演化需求,目前这些方法以智能体感知外界环境的问题作为核心的研究内容,而忽略了分布式软件系统内部状态的变化将因为其结构的复杂性,同样会对软件整体产生影响进而引发演化的这一现实问题.

本文旨在利用智能体技术,针对环境变化和用户意愿变更引发的演化需求,提出一种适用于分布式软件系统演化需求的软件自适应动态演化机制.针对目前对软件内部状态变化引发的演化需求分析不足的问题,该机制采用形式化环境建模方法和动态环境感知机制相结合的方式,有效地监控系统外部环境和软件内部状态的变化;并采用一种改进的合同网协议形成多 Agent 协作机制,实现演化过程中分布式节点的更替演化;并通过将系统中用任意语言开发出的功能单元封装为智能体,定义智能体间协作关系即演化策略,在重用原有软件单元的前提下,实现分布式软件系统在面向环境变化驱动和用户意愿变更驱动时的动态演化.

本文第 1 节介绍分布式软件演化领域以及其中通过环境感知实现演化的相关研究情况.为便于理解本文提出的基于智能体的软件自适应动态演化机制,第 2 节给出该机制的总体框架.在总体框架的指导下,第 3 节通过介绍形式化的环境建模及动态环境感知机制,给出面向环境信息变化的自适应动态演化机制.第 4 节通过介绍基于合同网的多 Agent 协作策略及其基本元素与核心算法,给出面向用户意愿变更的自适应动态演化机制.第 5 节介绍上述两种演化机制的支撑基础,即,基于智能体的动态演化体系结构,并阐述该体系结构的总体设计思路及演化机制在其中的具体映射.第 6 节介绍依据自适应演化体系建立的演化支撑环境,并选取一个分布式软件应用的典型场景,介绍如何利用演化支撑环境封装一个分布式软件,以及在应对来自环境变化和用户意愿变化的两种演化需求时,自适应演化机制的具体过程,并进行相关算法有效性的分析.第 7 节概括性地总结本文所做的工作,并对下一步需要改进的研究方向和工作内容进行展望.

1 相关工作

国内外学者和研究机构从不同的视角对分布式软件动态演化及其自适应机制进行了研究.

- 首先,从体系结构角度出发的代表性工作包括 CMU 的 Rainbow[15,16],它提供了一种在系统运行时保存一个或者多个系统体系结构模型的演化方法;UCI 的 ArchStudio[17]基于多层次总线结构的体系结构风格,利用一个外部的体系结构演化管理器实现对体系结构修改脚本的解释和执行[18];
- 从对象角度出发的研究有美国华盛顿大学研制开发的自适应通信环境 ACE(adaptive communication environment)[19],这是一个基于面向对象的工具开发包,利用运行时的动态配置功能实现对服务自适应演化的支持;
- 在利用 Agent 技术实现分布式软件系统自适应演化的研究中,IBM 公司研制的基于 Java 的 Agent 开发部署环境——ABLE(agent building and learning environment)[20]提供了由智能推理、学习构件包与构造智能 Agent 的框架构成的开发库,但该框架假定所开发的系统是静态和封闭的,无法满足开放环境下自适应系统的开发;英国的 Greer 和 Xiao 研究的 AAM(adaptive agent model)使用可配置的交互模型驱动自适应 Agent 的行为[21],该模型驱动方法类似于本文中利用演化规则驱动 Agent 间协作关系以进行自适应更改的方法,然而与本文方法相比,文献[21]定义的交互模型过于复杂,对开发人员要求较高.针对网构软件面临的环境显式化、实体主体化、运行适应性等问题.常志明提出了基于 Agent 网构软件的构件模型,将 DAgent 作为网构软件中的构件形式,利用 EBDI(environments,beliefs,desires,intensions)结构描述构件的自主行为[22],通过动态绑定关系体现构件的动态演化行为.

以上这些方法均是通过修改软件具体实现而使系统具有演化特性的方法,并未过多地考虑已有计算单元的重用问题.

在分布式软件系统动态演化的研究中,利用环境感知方法获取演化需求,成为了一个研究热点.其中,Capar 提出了 CARISMA(context-aware reflective middleware system for mobile application)[23]系统,考虑移动计算的特性,引入环境感知的设计思想,当感知环境发生变化时,系统也能做出相应的变化与调整.然而,该系统主要考虑的是外界移动互联网环境中的改变,对各个移动计算单元内部的状态基本没有考虑.澳大利亚皇家墨尔本理工大学的 Dam 提出了一种基于 Agent 的软件演化变化传播处理方法[24],他所提出的框架的基础是一个元模型和一组对象约束语言(OCL),底层变化传播机制是基于 BDI 模型的 Agent 结构.但该框架主要关注于静态环境的建模与研究,对动态环境的复杂性考虑得较为简单.Tamura 提出了一种感知环境变化的模型 DYNAMICO,并利用它实现自适应演化[25],提出了分离对象层、感知层和自适应层的思想.然而,该方法利用静态对象作为环境监听的基本单元,难以动态地感知系统内部状态的变化.Hussein 在软件的设计阶段提出了一种环境感知模型,包括功能认证、环境定义和自适应行为定义这 3 部分,并定义了三者之间的关系[26].该模型主要用于指导软件工作者在设计阶段预先对将来来自外界环境的变化进行处理,而忽视了软件在运行过程中自身状态也可能会发生变化,同样会对整个软件造成影响.

2 自适应演化机制

本文提出的基于智能体的软件自适应动态演化机制,主要针对来自系统外部运行环境和内部状态变化以及来自用户意愿变更引发的软件演化需求.为解决目前演化方法修改软件代码而导致的软件单元重用问题,本文提出:在不改变原有软件系统的设计过程、体系结构、代码实现的基础上,通过将软件单元和实现环境感知、自主控制、知识规则推理等功能模块统一包装成为软件 Agent,使软件单元具有 Agent 独特的感知性、自主性和智能性.被包装的功能单元可以是系统内部使用任意开发语言得到的计算单元,可以是已有的计算单元,也可以是根据需求使用其他语言开发出的新功能单元,体现出本文提出的自适应演化机制与演化对象的具体实现之间的无关性.软件演化逻辑是指用户根据需求以及基本单元之间的关系,演化机制将以这些封装了软件单元后的 Agent 作为演化单元,将演化逻辑映射为对各 Agent 单元之间的协作关系的调整,有效地避免了直接接触软

件单元.上述两方面保证了本文提出的演化机制可重用以不同语言和开发方法实现的分布式软件计算单元.

在将一般的分布式软件系统的功能节点包装成为 Agent 并将 Agent 协作关系定义为演化规则后,该分布式软件系统实际上已被包装成为多 Agent 系统.因此,在考虑这种分布式软件系统如何面对来自环境和用户两种演化需求时,首先需要考虑在多 Agent 系统中,这两种变化可能会触发 Agent 间协作逻辑发生怎样的改变[27].

图 1(a)展示了参与新任务的 Agent 单元类型不发生改变而关系发生变化的情况.如图所示:在演化后,计算单元依旧是 Agent *A*,*B*,*C* 这 3 类,改变的仅为三者之间的协作关系.其触发因素与个别 Agent 的增加和删除无关,而与运行过程中的内外部环境变化有关.这种变化将直接作用于系统中所有正在运行的 Agent,导致系统重新调整这些被影响的 Agent 间的协作关系,以应对来自内外部环境的变化.图 1(b)展示了参与新任务的 Agent 单元类型发生了改变的情况.如图所示:Agent *D* 代替 Agent *A* 加入了新任务的协作关系中,其触发因素与 Agent 的删除和增加相关,来自用户对软件个别功能单元的更新.这种变化将直接作用于某个 Agent 的生命周期状态,导致系统动态更新该 Agent,以应对来自用户意愿的变化.

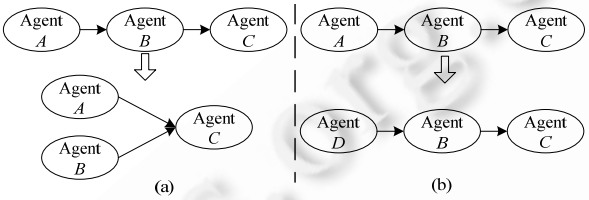


Fig.1 Cases of collaboration changes in evolution of multi-agent system

图 1 多 Agent 系统演化中协作关系的变化情况

因此,针对图 1 中所示的两类 Agent 间协作关系的变化情况,本文提出了一种基于 Agent 的软件自适应演化机制,将软件系统演化的驱动因素定义为两种:环境信息变化和用户意愿变更.该机制应对这两种不同的演化驱动因素分别提出了两种不同类型的演化策略,其中,核心模块的具体逻辑关系如图 2 所示.

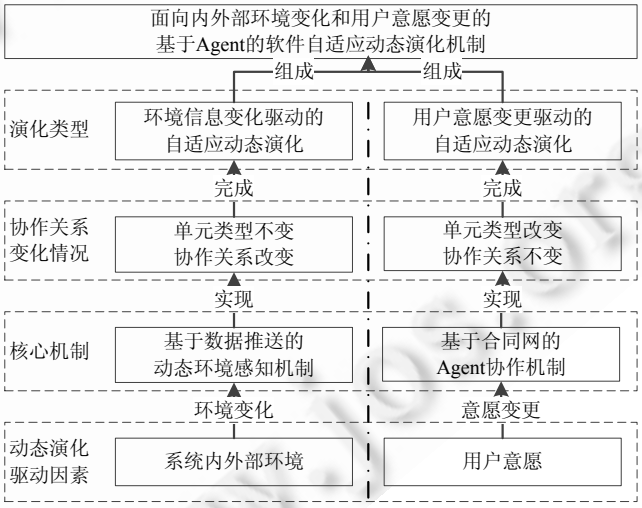


Fig.2 Logic relationship of core mechanism

图 2 核心机制逻辑关系

本文将系统内外部环境的变化映射为图 1(a)所示的 Agent 间协作关系演化,通过基于数据推送的动态环境感知机制,实现 Agent 单元类型不发生改变的条件下,动态地调整 Agent 间的协作关系,形成环境信息变化驱动的自适应动态演化机制.将用户意愿的变更映射为图 1(b)所示的 Agent 更新演化,通过基于合同网的 Agent 协作

机制,实现在不影响各 Agent 协作关系的条件下,动态地替换 Agent,形成用户意愿驱动的自适应演化机制.

3 环境变化驱动的自适应动态演化机制

环境变化是触发软件自适应演化的重要因素之一,本文主要从以下 3 个方面考虑环境驱动的演化机制的设计:

(1) 环境构成:是指影响软件系统开发、部署、运行与维护的元素的个数和类型;

(2) 感知方式:是指软件系统如何感知环境的变化和变化的内容;

(3) 演化方式:是指感知到变化后,系统应采取何种方式实现演化.

本节通过形式化建模环境,以解决环境的构成问题;通过基于数据推送的动态环境感知机制,实现对面向内外部环境信息变化的感知;通过演化规则分发算法,实现对环境变化驱动的自适应动态演化的支持.

3.1 环境的定义与分类

软件环境通常是指软件所处的物理环境,例如 CPU、内存等.本文所提出的环境不仅仅指外界的环境,还包括 Agent 状态、Agent 行为等系统内部的运行状态.为解决软件系统同时感知来自外部和内部环境变化的问题,首先需要对内部状态和外部环境分别进行形式化建模,二者以软件本身为划分依据,有着较为明显的边界,其详细定义如下所示.

定义 1(外部环境). 与软件系统运行平台相关,包括关于 Agent 的更替、演化规则的改变等软件系统全局和整体的信息.表示为四元组 $GloEVR=\langle AIDs,Capability,RuleSet,EnvironmentSet\rangle$,其中,$AIDs$ 表示能力各不相同的 Agent 的集合,$Capability$ 表示系统内存储的 Agent 当前的能力的集合,$RuleSet$ 表示系统内规则的集合,$EnvironmentSet$ 表示系统内环境信息的集合.

定义 2(内部状态). 与 Agent 个体相关,包括 Agent 之间通信内容、Agent 功能缺失情况和 Agent 损坏异常等 Agent 个体和局部的信息.表示为四元组 $PartEVR=\langle AID,State,Type,Value\rangle$,其中,$AID$ 表示 Agent 在系统中运行的唯一标识,$State$ 表示 Agent 生命状态的改变,$Type$ 表示 Agent 的类型,$Value$ 表示 Agent 能力的执行状态.

3.2 动态环境感知机制

根据定义 1 和定义 2,我们提出了一种动态环境感知框架.如图 3 所示,其整体风格类似管道结构,主要由感知模块和演化规则定制模块组成.

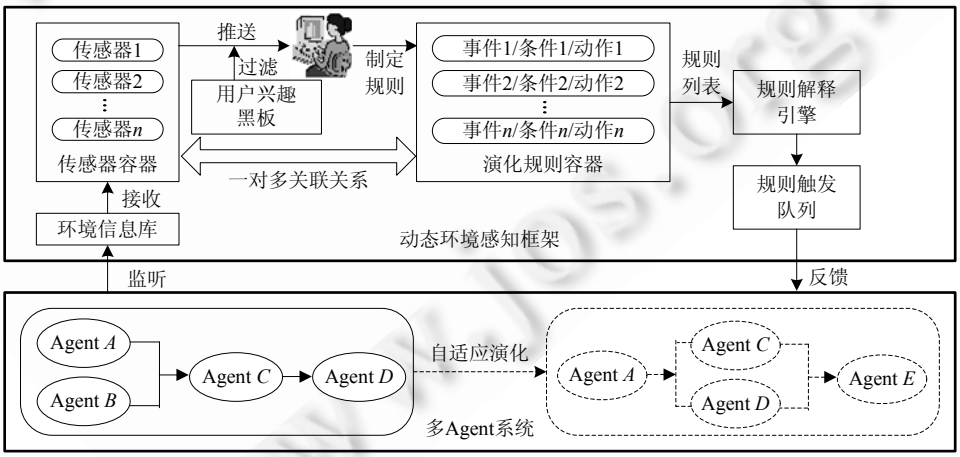


Fig.3 Dynamic environment awareness framework

图 3 动态环境感知框架

根据用户兴趣黑板记录的信息,传感器监听得到的环境信息经过过滤后可以向用户进行推送.如果用户拒

绝了某一传感器提供的信息,则该传感器将动态修改用户兴趣黑板中的相关信息以更新用户的兴趣.一条演化规则包含了一个或多个传感器的引用,即图中的一对多关联关系.在系统的运行过程中,将不断进行环境信息的监听,由环境信息库实时保存并更新最新的环境信息.传感器容器过滤出用户所关注的环境信息,并更新相应的传感器记录.同时,规则解释引擎不断地对用户定制的演化规则列表进行循环处理,根据引用的传感器数据和条件逻辑,判断某条演化规则的条件是否全部满足,若满足,则将该演化规则的引用放入规则触发队列中.

由于自适应软件与传统的控制系统理论非常相似,即所谓“收集-决策-控制”周期,所以感知技术体现在怎样完成收集的问题域.基于本文提出的动态环境感知框架,该问题域包含数据收集与数据抽象两点:数据收集即设计怎样的监视器,使之能够收集到特定的数据,从而获得对运行态软件的认识;数据抽象则是以数据收集为基础,将收集到的原始数据抽象为有用的信息,从而和用户制定的演化规则相联系起来.借鉴观察者设计模式思想,给出了基于数据推送的环境感知机制的过程模型,实现环境数据的收集过程,如图 4 所示.

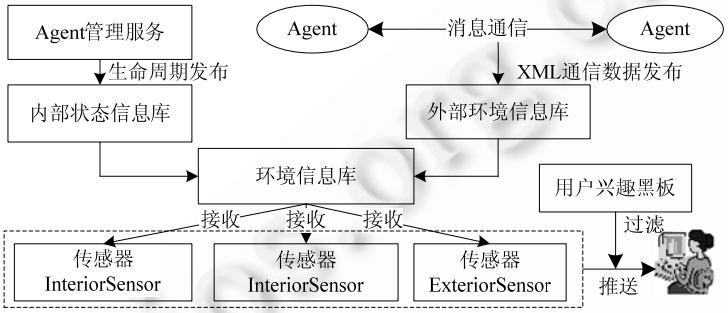


Fig.4 Process model of the dynamic environment awareness

图 4 动态环境感知过程模型

本文将自适应 Agent 对环境的感知过程分为监听、推送、接收和拒绝这 4 个阶段.由于动态环境分为内部状态和外部环境,则对动态环境的感知过程包括了对两类环境信息的感知.收集到的所有的环境信息将汇集到环境信息库中,完成对环境信息的收集过程.

(1) 环境信息的监听

首先,关于内部状态的监听由 Agent 管理服务及内部状态信息库完成,其中,Agent 管理服务用来监听 Agent 当前的生命状态信息,且只有在 Agent 的生命状态发生变化时才记录一次状态信息;对外部环境的监听由 Agent 和外部环境信息库完成,与内部环境信息不同的是,由于 Agent 之间是实时通信的,且通信的内容可能是不断变化的,所以 Agent 在通信过程中需要不断地进行环境消息的监听和记录.

(2) 环境信息的推送与拒绝

在自适应演化平台中,对于不同类型的环境变化,需要定制不同类型的传感器以完成对环境的监控功能.监听不同环境信息的传感器可以根据用户兴趣黑板中记录的用户兴趣信息,过滤得到的环境信息后再进行推送.用户也可以拒绝接受相关信息,系统会动态调整用户兴趣黑板的记录,减少相关环境信息的推送次数或取消推送.

定义 3(内部传感器). 用来订阅内部状态信息,可形式化地表示为一个三元组 *InteriorSensor*=⟨*Sname*,*AID*,*LifeMessage*⟩:*Sname* 是内部传感器的名称,也是唯一标识:*AID* 是传感器所订阅的 Agent 的唯一标识,可将传感器与内部状态信息库中的内容关联起来;*LifeMessage* 是 Agent 的生命状态值,初始化为空.平台中 Agent 的增删变化情况表示 Agent 生命状态的变化.

定义 4(外部传感器). 用来订阅外部环境信息,可表示为一个五元组 *ExteriorSensor*=⟨*Sname*,*AID*,*Attribute Name*,*AttributeType*,*AttributeValue*⟩,其中,*Sname* 是传感器的名称;*AID* 是 Agent 的唯一标识,可与外界环境信息库相关联;*AttributeName* 为 Agent 的属性名称集合,Agent 进行通信时,传递的内容为 Agent 的属性值,且可以传递多个属性;*AttributeType* 为属性的类型;*AttributeValue* 为属性的值,初始化为空.

(3) 环境信息的接收

系统根据环境信息库的内容对传感器列表不断循环,筛选出用户所关注的环境信息,将其转换为与传感器类型所匹配的环境信息,实时地对传感器内容进行更新和存储,完成对环境信息的抽象过程.

根据对上述从监听、推送、接受到拒绝这 4 个阶段的分析,可将环境感知的整体过程抽象成为环境变化感知算法.系统环境的变化由动态环境变化感知框架利用算法 1 感知,该算法通过对环境的实时监测与数据比对计算变化信息,感知外部环境变化,然后触发事件发生器,通过对环境变化信息与事件关系的分析产生相应的事件,从而触发演化规则.其中,环境感知算法的具体内容如下所示.

算法 1. 环境变化感知算法.

输入:当前外部环境信息;

输出:外部环境变化相应事件.

```
1:  Environment environment_new,environment_old;        %生成环境对象
2:  WHILE true DO                                        %循环监测当前环境
3:     environment_new=read(environment_current);        %读取当前环境信息
4:     change=environment_new-environment_old;           %计算环境变化信息
5:     IF change==0                                      %若环境未发生变化
6:        continue;                                      %继续监测
7:     END-IF
8:     ELSE                                              %若环境发生变化
9:        updateEnviromentBase(change);                  %将环境变化信息更新至环境信息库
10:       EventGenerator.Trigger(change);                %触发事件产生器
11:    END-ELSE
12:    environment_old=environment_new;
13: END-WHILE
```

3.3 规则分发机制

在感知环境信息的变化后,需要有相应的演化规则进行支持,使得系统能够根据环境信息的变化完成自适应演化.因此在动态环境感知框架的基础上,本文结合自适应演化的特点,基于“事件/条件/动作(event condition action,简称 ECA)”的演化规则,通过触发 ECA 规则,实现系统 Agent 间协作规则的动态切换,完成系统的自适应动态演化过程.ECA 规则的基本语义是:规则在事件发生时被触发,在特定的时刻由系统检查规则条件,如果条件满足,系统将执行规则的动作.多个 ECA 规则组成了一个规则库,本文将其定义为演化规则库.

定义 5(ECA 规则).

ON event of environment information changes;

[IF condition on life message or attribute is available];

DO action of switching the integration rules.

在系统正常运行的情况下,感知器对环境进行实时监控.当环境信息发生变化时,产生相应的事件,需要对传感器的值进行更新,并在演化规则库中查找匹配的条件.匹配成功,说明演化规则的条件满足.演化控制引擎对满足条件的演化逻辑文件进行解析,并得到新的演化规则.当新的规则分发成功后,各 Agent 会调整自身与其他 Agent 间的关系,形成新的协作关系,完成系统的自适应演化.其中,演化规则分发算法如下所示.

算法 2. 演化规则分发算法.

输入:事件触发信息;

输出:新的协作规则分发状态.

```
1:  action=msg.Action;          %获取该事件对应的分发动作
2:  rule=msg.Rule;              %获取该事件对应的分发规则
```

```
3:  IF action=="Clear_Rule"
4:      state=idle;                    %置该服务 Agent_i 状态为 idle(闲置)
5:      tcount=0;                      %当前任务数置 0,清空该服务 Agent_i 规则库中的规则
6:      Agent_i.Rules=NULL;            %记录任务执行信息,包含规则清空发生时间、任务描述等
7:      logfile<<task;
8:  END-IF
9:  IF action=="Dispatch_Rule"
10:     Search_CRC(Agenti.Sname);      %CRC 进行 Agent 定位
11: IF (Success_Search)
12:     Agenti.Rules=NULL;             %形成新的协作规则,并写入内存
13:     Construct_Rule(msg);           %形成规则查询成功的消息
14:     Construct_Message_Success(msg);
15: END-IF
16: ELSE
17:     Construct_Message_Fail();      %形成规则查询失败的消息
18:     SendMessage(msg);              %通过 MTS 向 CA 发送消息
19: END-ELSE
```

4 意愿变更驱动的自适应动态演化机制

用户意愿的变更是触发软件自适应演化的另一个重要因素.用户意愿变更是指用户对业务过程或应用目标的期许发生变化,软件需要通过动态调整或重组业务流程来满足用户对软件的新期许.在不中断系统运行的情况下,如果负责提供该服务的Agent失效或能力不足,则需要通过多Agent协作机制寻找到有能力完成任务的Agent代替原Agent,以满足用户的新需求.因此,本节提出了基于合同网的多Agent协作机制,实现对面向用户意愿变更的自适应动态演化机制的支持.

4.1 CNPIIS基本定义

合同网协议(contract net protocol,简称CNP)是在多Agent系统中应用最广泛的一种协作方法[28],是由Devis和Smith提出的一种为了实现分布式问题求解的节点间协商与协作的交互机制[29],通过引入市场中的招标、投标、中标机制[30],将任务的委派通过节点之间的招标过程实现,将协作引入到招标方和投标方的双向选择过程中[31].

借鉴传统合同网协议,本文提出了一种引入第3种角色——信息中介服务的改进合同网协议(contract net protocol based on information intermediary service,简称CNPIIS),即,利用公共消息黑板(common message blackboard,简称CMB)为Agent的协商过程提供服务和管理;并通过引入Agent执行某项任务的可信度(degree of credibility)和可用度(degree of availability)使协商过程更趋于合理有序、公平有效,更加适应于动态环境.

定义6(可信度). 可信度是指招标参与Agent执行某种类型的任务后,获得的质量评价.如公式(1)所示,其中,n为Agent完成某种类型任务的次数;v_i为完成第i次任务时,协商发起Agent对任务完成的满意度;w_i为协商发起者Agent为第i次任务执行设置的权值.

$$DoC=\sum_{i}^{n}w_i\times v_i\Big/\sum_{i}^{n}v_i \tag{1}$$

定义7(可用度). 可用度是指招标参与Agent的空闲程度.每个招标参与Agent最终只可以和一个协商发起Agent建立合约并执行任务,在未得到合同之前,它接受的投标请求越多,其与后续的协商发起Agent建立协作的可能性越低,即,其可用度越低,如公式(2)所示,其中,NRI(the number of received invitations)指的是招标参与Agent接收到和投标邀请的个数.

$$DoA=\frac{1}{NRI} \tag{2}$$

定义 8(公共消息黑板). 可表示为一个四元组 $CMB=\langle AgI,Service,History,DoC\rangle$,其中,$AgI$ 为 Agent 的能力信息和分类;$Service$ 为 CMB 可以提供的服务,$Service=\{(s_i)|s_i\in\{enroll,update,query,tenderDeal\}\}$,$enroll$ 为 Agent 能力注册服务,$update$ 为 Agent 能力更新服务,$query$ 为 Agent 能力和状态查询服务,$tenderDeal$ 为协商发起者标书发布管理服务;$History$ 为 CMB 记录的协商历史;DoC 为可信度信息.

定义 9(基于信息中介服务的合同网协议). 可表示为一个七元组 $CNPIIS=\langle Ag,IA,PA,CMB,C,Msg,Time\rangle$,其参数描述可见表 1.

Table1 Description of CNPIIS parameters

表 1 CNPIIS 参数描述

名称	描述
Ag	$Ag=\langle a_1,a_2,a,\ldots,a_n\rangle(n\geq 2)$是系统中的 Agent 集合
IA	为发起协商的 Agent,与传统的合同网协议相比职责有所不同,和 CMB 共同充当管理者角色
PA	代表招标参与 Agent 集合,Agent 具有主动性,会主动向 CMB 注册或更新能力信息
CMB	即为公共消息黑板,提供信息中介服务
C	*C* 为发起协商的 Agent,CMB 和参与投标的 Agent 之间协商命令的有效取值.$C=\{(c_i)|c_i\in\{announcing,accept,reject,refuse,propose\}$,其中,*announcing* 表示协商发起者向 CMB 发布标书,*accept* 表示协商发起者 Agent 接受标书,*reject* 表示协商发起者 Agent 拒绝标书,*refuse* 表示招标参与 Agent 拒绝投标,*propose* 表示招标参与 Agent 进行投标
Msg	代表协商的消息载体
Time	表示协商进行的系统时间

由定义 9 可知:在本文提出的合同网协议中,Agent 具有自治性和主动性,可以通过判断自身的状态主动向公共消息黑板注册或更新其能力和状态信息.公共消息黑板维护 Agent 的能力和状态信息,并以这些信息作为依据,为合同网协议协商过程的实现提供服务.因此,该合同网协议具有如下优点:首先,有效避免了不必要的协商通信,因为如果该 Agent 不具有执行某项标书的能力或处于忙碌状态,则必然无法完成协商;其次,信息中介服务起到了 Agent 间对相同或相似分布式问题求解和任务分配的协商经验共享,新加入系统中的 Agent 不必再逐步建立其协商经验知识.

4.2 CNPIIS核心算法

当用户意愿变更时,如果原有 Agent 的功能无法满足,则平台通过基于信息中介服务的合同网协作策略,寻找具有该功能的 Agent 参与竞标,以完成用户要求的新任务.其中,具体协作过程如图 5 所示.

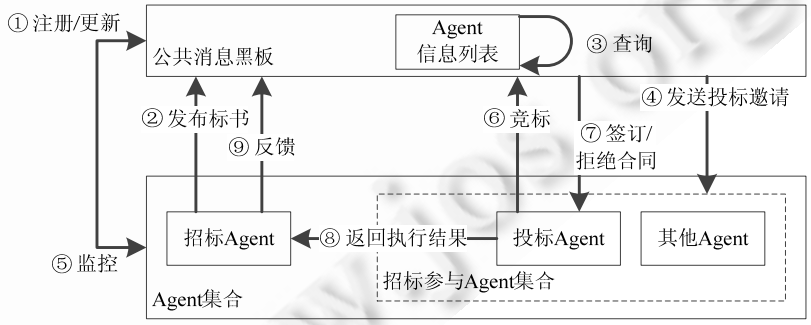


Fig.5 Collaborative process of CNPIIS

图 5 CNPIIS 协作过程

当一个 Agent 被部署到系统中或其能力发生动态改变无法提供服务时,该 Agent 会主动向 CMB 进行注册或更新其能力信息,如图 5 中标号①所示.当 Agent 需完成某项任务,而自身不具备完成能力,则向 CMB 发布标书,通过合同网的招标机制寻找合适的 Agent 以代替,如图 5 中标号②所示,其中,标书的定义如下所示.

定义 10(标书). 表示为 *Tender*=⟨*TenderID*,*Owner*,*Category*,*Specificity*,*Expires*,*Prohibition*⟩,其中,*TenderID* 表示标书的 ID,向 CM*B* 申请获得,由 CMB 统一管理;*Owner* 表示发布标书的协商发起 Agent 的标识信息,*Owner*=⟨*Name*,*IP*,*Port*⟩;*Category* 表示协商发起 Agent 招标的任务类型,任务的类型是系统任务类型分类中的一种;*Specificity*=⟨*Desc*,*Input*,*Output*⟩,包括对招标任务的描述、招标参与者需要满足的输入和输出接口;*Expires* 表示投标的截止期限;*Prohibition* 表示发布标书的 Agent 确定的禁止投标的 Agent 的集合.

服务 Agent 向公共消息黑板发布标书的算法如下所示.

算法 3. 服务 Agent 发布标书算法.

输入:需要完成的子任务;

输出:标书.

1: $agent_a$ needs to perform the task T.
2: $planning(T)\Rightarrow T=\{t_1\cup t_2\cup t_3\cup t_4\cup\ldots\cup t_n\}$. %task T is planned to subtasks in sequences of $\{t_1,t_2,t_3,...,t_n\}$
3: FOR EACH subtask t_i in T, $0<i<n+1$
4: IF $agent_a$ is able to complete subtask t_i
5: execute the subtask t_i %执行任务 t_i
6: END-IF
7: ELSE
8: make announcement and find the contractor for subtask t_i through contract net.
9: send $tender_i$ to CMB. %向公共消息黑板 CMB 发布标书
10: END-ELSE
11: END-FOR

CMB 接收到招标发起者发布的标书后,根据标书发布的信息和所维护的 Agent 信息列表,查找可参与投标的 Agent,如图 5 中标号③所示.例如,招标发起 Agent 需要招标一项任务 t,若 $Agent_a$ 具有完成此项任务的能力,则 $Agent_a$ 是此项招标的招标参与者.CMB 根据查询的结果向招标参与 Agent 集合发送投标邀请,如图 5 中标号④所示.CMB 还对新加入系统的 Agent 进行监控,若新加入的 Agent 具备提供某项标书所描述的招标的能力,则 CMB 也会向此 Agent 发送投标邀请,如图 5 中标号⑤所示,其相应算法如下所示.

算法 4. CMB 发送投标邀请算法.

输入:投标候选 Agent;

输出:投标邀请.

1: $participant(tender_i)=\varnothing$.
2: FOR EACH $agent_j\in CMB$ && $agent_j$ is capable to compelte task t_i %获得候选 Agent
3: $participatant=Participator\cup agent_j$.
4: send call for proposal to each participant. %发送投标邀请
5: END-FOR
6: put the $tender_i$ in the tender queue.
7: FOR EACH tender in tender queue
8: WHILE $time<message\text{-}deadline\ of\ tender_i$ %监控更新的 Agent 信息
9: IF $agent_m$ is new or update && $agent_m.supports(tender_i)$
10: send call for proposal to $agent_m$. %发送投标邀请
11: END-IF
12: END-WHILE
13: END-FOR

获得投标邀请的 Agent 读取投标要求内容,根据自身能力和状态确定是否参与投标.参与者可拒绝投标,也

可发送投标请求给 CMB 参与竞标,如图 5 中标号⑥所示.招标发起 Agent 按照一定的评估标准依次向投标的 Agent 发出请求,直到找到同意签订合同的投标 Agent.签订合同后,招标发起 Agent 会对其他的投标 Agent 发送拒绝签订合同通知,如图 5 中标号⑦所示,算法如下所示.

算法 5. 服务 Agent 处理投标算法.

输入:参与投标的 Agent 集合;

输出:签订或拒绝签订信息.

```
1:  read all propose from participant and call this set A.
2:  FOR EACH propose p∈A.
3:      compute the utility of by its cost, DoC and DoA.
4:  END-FOR
5:  sort the participant in descending order in list qList.
6:  DO                                                    %发送投标邀请
7:      send request to the qList[i].
8:      IF qList[i] Agrees
9:          Send accept-propose to qList[i].
10:         Send reject-propose to the rest participant in qList.
11:     END-IF
12:     ELSE i++
13:     END-ELSE
14: WHILE i<length of qList
15:     Send call for proposal to agent_m.                %发送投标邀请
16: END-DO-WHILE
```

投标参与 Agent 接受招标发起 Agent 合同之后,执行相应的任务,并将执行结果通知给招标发起 Agent,如图 5 中标号⑧所示.招标发起 Agent 对执行结果进行评价,并将评价和结果反馈给 CMB,如图 5 中标号⑨所示.经过以上流程,投标的 Agent 将代替招标的 Agent 完成相关任务,满足用户变更后的新需求,最终实现软件动态自适应演化.

5 自适应动态演化平台

为上述两种演化机制的实际应用,设计了一种演化体系结构,该体系结构是软件自适应演化机制的基础和静态表示.依据该体系结构,可以实现对应的软件,即演化支撑环境,包括演化支撑平台及相关工作集.

本文参考 FIPA 平台规范所描述的 MAS 抽象体系结构,结合 Agent 技术和两种自适应演化机制的理论和方法以及对分布式软件单元的重用要求,定义了基于 Agent 的软件动态演化体系结构,其主要由 3 部分组成,如图 6 所示,包括演化基本单元定义、软件演化逻辑描述、演化机制过程控制与管理.在该体系结构中,认为 Agent 是提供某种计算或服务的工作单元[32].

当外部环境变化或内部运行状态发生改变时,将触发面向环境变化的自适应演化.如图 6 中标号①所示,演化感知中心(evolution perception center,简称 EPC)通过算法 1 监听外界环境和系统内部 Agent 间通信的变化,根据用户兴趣黑板,过滤出用户关注的环境信息,并触发事先定义的公共事件或实例化事件.如图 6 中标号②所示,事件触发的相应 ECA 演化规则将被加载到演化控制引擎(evolution control engine,简称 ECE),并通过算法 2 将新的任务分发给相应的 Agent.如图 6 中标号③所示,完成演化逻辑的切换,动态调整各 Agent 间的协作关系,实现对面向环境变化驱动的自适应演化机制的支持.

当用户意愿发生改变时,将引发面向意愿变更的自适应演化,即,新的协作单元需要参与到新的协作关系中.如图 6 中标号④所示,根据用户新意愿,动态查找能力注册中心(capability registry center,简称 CRC).如果存在

具有这项能力的 Agent,则直接利用算法 3~算法 5,在公共消息黑板 CMB 的辅助下进行基于信息中介服务合同网的多 Agent 协作,将这个 Agent 替换到新的协作关系中,如图 6 中标号⑧所示;如果并不存在这样的 Agent,则需要利用 Agent 包装工具将新增单元包装为 Agent 单元,并利用知识规则设计工具定义其知识规则,如图 6 中标号⑤所示.当完成包装和规则定义后,平台将加载这个新的功能 Agent 并向 Agent 管理服务和能力注册中心进行注册,如图 6 中标号⑥和标号⑦所示.此时,由于算法 4 可以动态感知到新加入的 Agent,并向它们发起投标邀请,因此同样可以借助信息中介合同网协议,实现在不中断系统运行的情况下完成 Agent 更替,满足用户意愿变更的自适应演化需求.

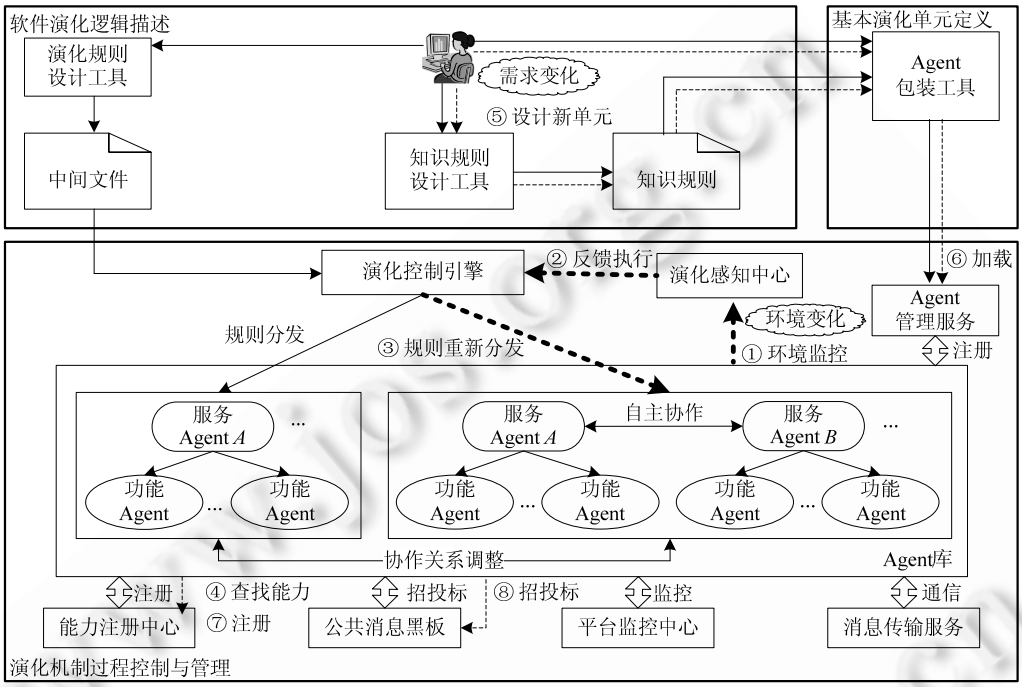


Fig.6 Architecture of the agent-based dynamic evolution

图 6 基于 Agent 的软件动态演化体系结构

如图 6 所示,利用基本演化单元定义中的 Agent 包装工具,可完成对分布式软件单元的封装;利用软件演化逻辑描述中的演化规则设计工具和知识规则设计工具,可完成对 Agent 间协作关系的定义和 Agent 内部知识规则的定义.这两个模块将共同实现对原有软件单元重用的支持.演化机制过程控制与管理则通过内部关键模块和相关支撑模块,例如 Agent 管理服务(agent management service,简称 AMS)、平台监控中心(platform monitor center,简称 PMC)、消息传输服务(message transition service,简称 MTS)等,具体实现对上述两种演化机制的支持,控制和过程管理等功能.

依据上述体系结构,研制了基于 Agent 的软件系统自适应动态演化支撑环境,包括实现演化体系结构中演化机制过程控制与管理的软件演化支持平台、实现基本演化单元定义的 Agent 包装工具,以及实现软件演化逻辑描述的演化规则设计工具等相关工具集.其中,演化支撑平台如图 7 所示.

支撑平台作为演化体系结构中的关键部分,是动态演化机制的承载平台,提供动态演化过程中所需的各种服务.例如:平台管理工具集中包括的 Agent 库管理、Agent 管理中心、Agent 能力中心等,实现对平台内部 Agent 的信息管理和运行控制;公共消息黑板和熟人管理中心实现对基于合同网的协作机制的支持.用户可以使用系统运行控制功能、手动切换演化脚本,通过平台配置工具完成平台的功能设定.针对外部环境变化引发的演化需求,通过全局演化控制功能,完成对感知外界的外界传感器和外界事件进行定义.针对内部状态变化引发的演

化需求,通过局部演化控制功能,完成对感知内部状态的内部传感器和内部事件进行定义.同时,针对用户意愿变更引发的演化需求,平台监控工具也将实时监控系统内部各 Agent 之间的通信情况和各 Agent 的运行情况,动态完成 Agent 的查找和替换.通过上述功能,特定应用场景下的分布式软件系统中的软件单元在被封装为 Agent 并部署到平台后,平台将能够实现对来自两种演化需求的自适应动态演化的支持.

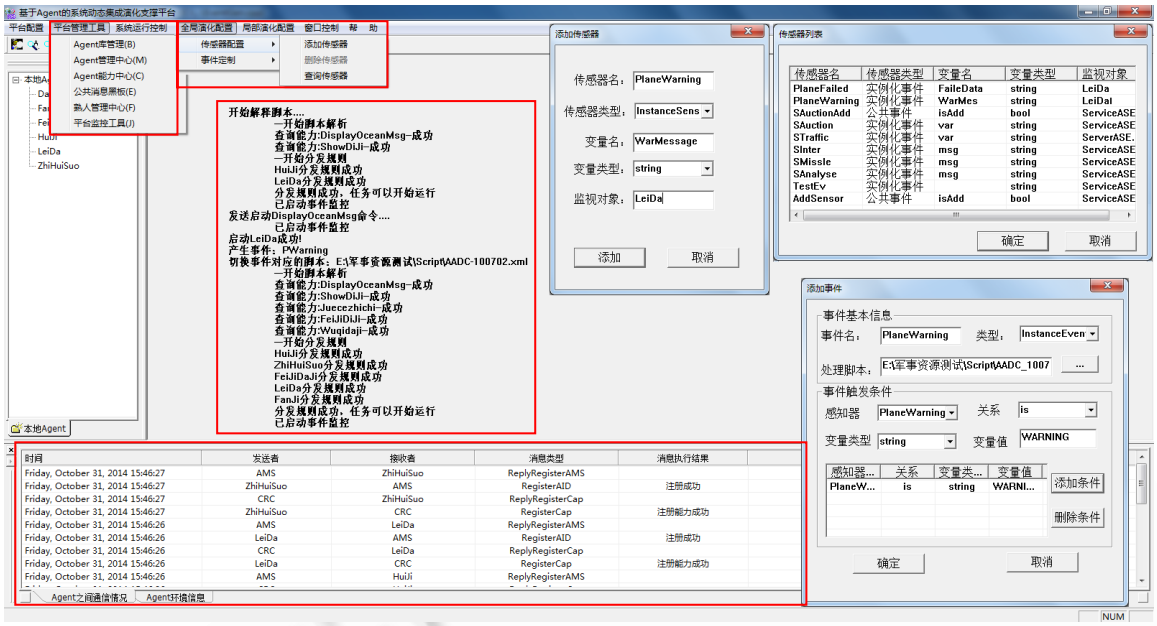


Fig.7 Evolution support platform

图 7 演化支撑平台

6 案例分析与实验

为验证演化支撑平台对演化机制的支持能力,特别选取防空指挥控制领域作为典型应用场景.该领域中的分布式软件系统需要实时运行且不容许停机维护,并且处在动态变化的环境中.因此,将利用演化支撑平台支持该领域中软件系统的演化需求.在该应用场景中的软件系统存在着指挥所、雷达、情报中心、拦截机、反击中心等软件单元.情报中心汇总雷达收集到的敌情信息,指挥所负责根据情报中心的信息拟定作战策略,反击中心将管理拦击机等打击武器.当敌机未接近警戒区时,属于正常防御状态;如果敌机进入警戒区时,情报中心将发出紧急警报,指挥所将指挥打击中心进行打击,反击中心将派出拦截机打击敌机.若指挥所被敌机毁坏后,则需要尽快更替新的指挥所继续进行决策工作.反击中心可以动态添加新的打击武器,例如导弹等,扩展武器种类.

演化平台在该应用场景下运行的硬件环境为 Intel(R) Core(TM)2 Duo E7500@ 2.93GHZ,1.87GB;软件环境为 Windows7 SP1;网络环境为百兆以太网(局域网).

6.1 演化过程分析

首先,软件单元将通过 Agent 包装工具封装为指挥所 Agent、雷达 Agent、汇集 Agent、飞机打击 Agent 和导弹打击 Agent、反击 Agent 等.敌机入侵位置的变化将被映射为外部环境变化,指挥所毁伤被映射为内部状态变化,对敌打击手段的更新被映射为用户意愿的变更.通过演化规则设计工具,分别设计正常情况时、敌机入侵至警戒区域时、指挥所毁伤时等不同演化规则.演化支撑平台启动后,将自动加载所有 Agent,并加载和分发特定的演化规则,在运行过程中不断监测各个 Agent 的状态信息及环境信息,以便于随时应对不同的变化,做出演化动作.

当敌机已进入我方飞机警戒区时,指挥所 Agent 将根据汇集 Agent 和雷达 Agent 的信息,指挥反击 Agent

派出飞机打击 Agent 进行拦截.该情景下的实际运行情况如图 8 所示,在通过环境感知机制动态感知到“敌机进入警戒区”这个外部环境变化信息后,预先定义的飞机拦截事件被触发.平台从正常状态下的演化规则自动切换到了敌机入侵时的演化规则,指挥所 Agent 启动了飞机打击 Agent 对敌机目标进行打击.

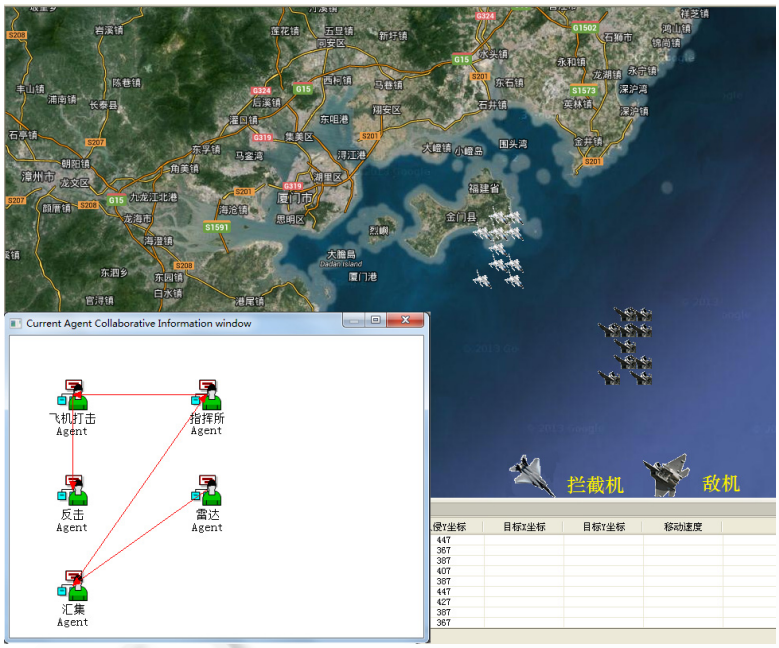


Fig.8 Relationship between agents in the aircraft interception

图 8 飞机拦截时 Agent 间协作关系

当指挥所 Agent 被敌机轰炸而发生毁伤时,应当动态完成新的指挥所 Agent 替换.在此情景下的平台运行情况如图 9 所示,在通过环境感知机制感知到“指挥所毁伤”这个内部状态变化后,毁伤事件被触发.平台重新启动了备份的指挥所 Agent,继续进行指挥工作.

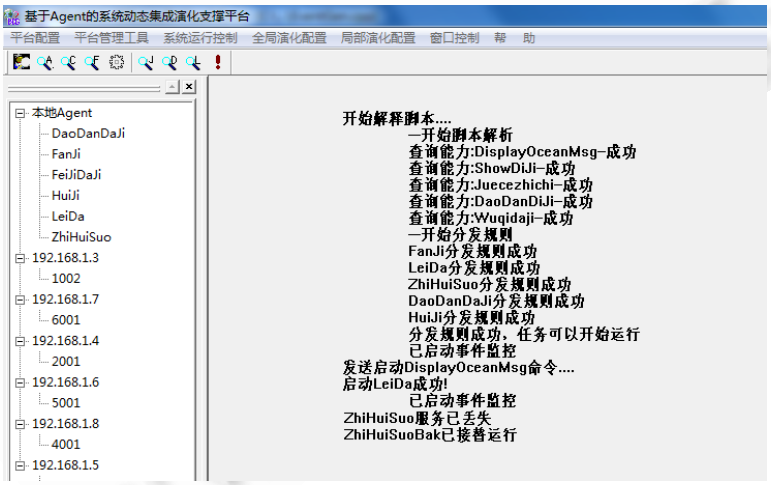


Fig.9 Dynamic replacement of the damage node

图 9 动态替换毁伤节点

当用户需要增强打击威力时,需要替换另一种打击方式——导弹打击,因此,导弹打击 Agent 应当替换飞机打击 Agent 加入到协作关系中.在此情景下的平台运行情况如图 10 所示,通过 Agent 能力中心查找到了具有这一功能的导弹打击 Agent 并加载到平台中.导弹打击 Agent 在通过基于合同网的协作机制竞标成功后,替换了原有的飞机打击 Agent,参加到了拦截任务中,并立刻发射了导弹对敌机进行了拦截.

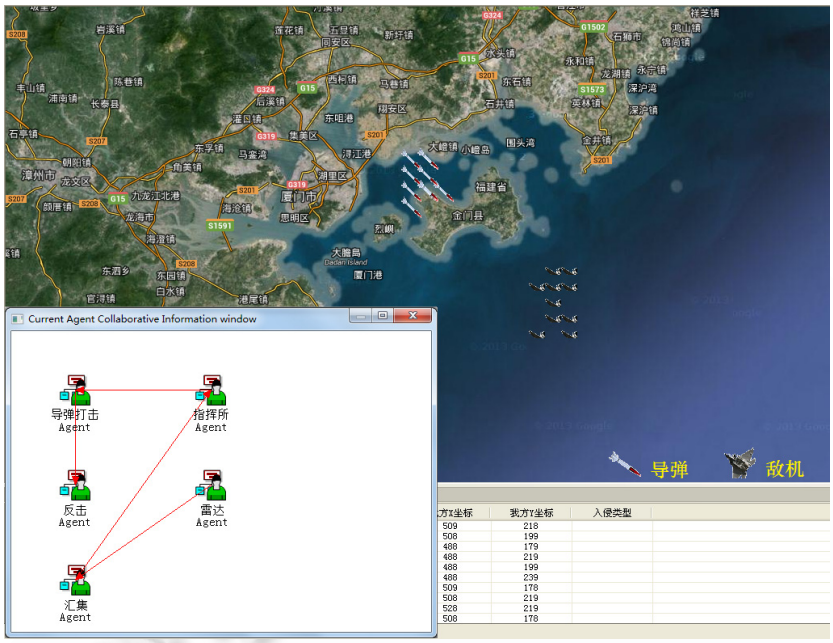


Fig.10 Relationship between agents in the missile interception

图 10 导弹拦截时 Agent 间的协作关系

6.2 算法有效性分析

为衡量本文提出的自适应演化机制的能力,我们从协作关系重组时间、节点毁伤切换时间、通信率这 3 个方面对本文提出的动态演化机制中的各种算法的有效性进行评估.

协作关系重组时间是指从演化平台加载新的演化规则到新规则完全分发到各 Agent 并更新协作关系的时间,体现了本文提出的面向环境变化驱动的自适应动态演化机制应对外界环境变化的能力.本文通过在不同节点数下改变外界环境,对该时间进行了实验,如图 11 所示.由于协作关系更新以上文所述的规则分发机制为基础,依赖于数据的总传输时间,因此,重组时间与协作节点数成正比关系,说明本文提出的面向环境变化驱动的演化机制在应对外界环境变化时,算法 1、算法 2 在节点数量上升时的时间效率处在可接受的范围内,不会产生激增现象.

节点毁伤切换时间是指从某个节点毁伤到被平台感知并找到备份节点接替运行所经历的时间,体现了本文提出的面向环境变化的自适应动态演化机制中感知内部状态变化的能力.本文通过在 40 个服务 Agent、102 个功能 Agent 的条件下多次进行模拟节点毁伤,对该时间进行了实验,如图 12 所示.毁伤节点的判断依赖于支撑平台中平台监控中心的超时值设定,默认值为 5s.

由图 12 可以看出:节点毁伤切换时间在多次实验中稳定在 6s 左右,说明本文提出的面向环境变化的演化机制能够在合理的时间内感知并应对来自系统内部运行状态的变化.

通信率是指单位时间内 Agent 间的通信次数,体现了本文提出的面向用户意愿变更的自适应动态演化机制中,多 Agent 协作机制,即算法 3~算法 5 的有效性.本文在平台中分别实现了基于经典合同网协议(CNP)的协作机制和基于信息中介服务的合同网协议(CNPIIS)的协作机制.这两种机制在 10 个服务 Agent、26 个功能 Agent

的相同条件下,完成相同任务时的通信率变化情况,如图 13 所示.由图可得:CNP 的通信率高于 CNPIIS;并且随着任务数的增多,CNPIIS 的通信率逐步下降并趋于稳定,CNP 的通信率基本不变.这是由于,在任务数较少的初始运行阶段,Agent 需要向信息中介服务注册或更新自身信息,CNPIIS 处于信息积累阶段,因此通信率与 CNP 相近;随着系统的运行,任务数量增多,信息中介服务中的 Agent 信息列表也逐渐充实并开始发挥作用.当需要将具有某项功能的 Agent 替换到协作关系中时,只需在信息中介服务中查找到相关 Agent 并发送投标邀请,避免了不必要的协商通信,因此通信率相较于 CNP 会有所下降.

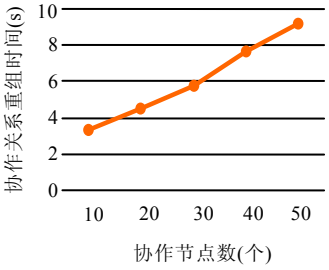


Fig.11 Variation of collaboration restructuring time

图 11 协作关系重组时间变化图

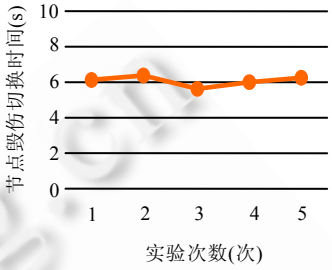


Fig.12 Variation of switch time of node destruction

图 12 节点毁伤切换时间变化图

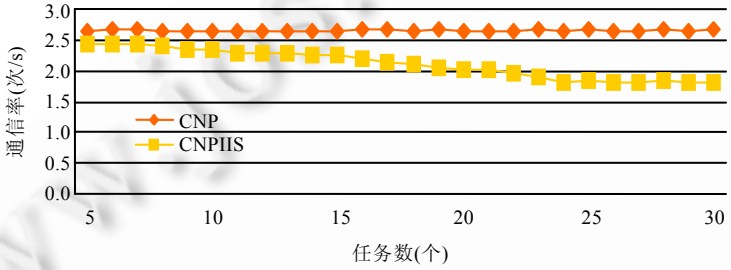


Fig.13 Variation of communication rate

图 13 通信率变化图

7 总结与展望

本文针对分布式软件系统在自适应动态演化中存在的演化方法难以重用、软件内部运行状态影响软件演化等关键问题,借助智能体技术,通过封装软件单元为 Agent 并定义演化逻辑,提出了可重用原有软件单元的自适应动态演化机制.通过动态环境感知机制,满足来自内外部环境变化的演化需求.通过基于合同网的 Agent 协作机制,满足来自用户意愿变更的演化需求.

然而,如何应对来自分布式软件系统更多类型的演化需求、如何对演化支撑平台本身的性能进行验证、如何完善演化机制中各种算法并进一步验证其性能,是本文下一步的研究内容.相信随着这些关键问题的攻破,基于 Agent 的软件自适应动态演化技术将为用户带来应变能力更好的分布式软件系统.

References:

[1] Amogh D, Constantine D. An agent-based model for the evolution of the Internet ecosystem. In: Mallik R, ed. Proc. of the 2009 1st Int'l Conf. on COMmunication Systems and NETworks (COMSNETS 2009). Bangalore: IEEE Press, 2009. 1−10. [doi: 10.1109/COMSNETS.2009.4808892]

[2] Tao XP, Feng XY, Li X, Zhang GQ, Lü J. Communication mechanism for Mogent system. Ruan Jian Xue Bao/Journal of Software, 2000,11(8):1060−1065 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/11/1060.htm

[3] Wang QX, Shen JR, Mei H. Study about adaptive software. Computer Science, 2004,31(10):168−171 (in Chinese with English abstract).

[4] Chang ZM, Mao XJ, Qi ZC. The network structure of software component model based on agent and its implementation. Ruan Jian Xue Bao/Journal of Software, 2008,19(5):1113−1124 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/19/1113.htm

[5] Wang J, Chen WR. A reliability-oriented evolution method of software architecture based on contribution degree of component. Journal of Software, 2012,7(8):1744−1750. [doi:10.4304/jsw.7.8.1744-1750]

[6] Pan JL, Han MH. The evolutionary optimization of system of systems based on Agent model. In: Güllüoğlu SS, Ata A, eds. Proc. of the 2014 IEEE Int'l Conf. on System Science and Engineering (ICSSE 2014). Shanghai: IEEE Press, 2014. 231−235. [doi: 10.1109/ICSSE.2014.6887940]

[7] Perrouin G, Morin B, Chauvel F, Fleurey F. Towards flexible evolution of dynamically adaptive systems. In: Tamura G, Villegas N, eds. Proc. of the 2012 ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems (SEAMS 2012). Zurich: IEEE Press, 2012. 1353−1356. [doi: 10.1109/ICSE.2012.6227081]

[8] Vogel T, Giese H. Model-Driven engineering of self-adaptive software with EUREMA. ACM Trans. on Autonomous and Adaptive Systems (TAAS), 2014,8(4):18. [doi: 10.1145/2555612]

[9] Breivold HP, Crnkovic I, Land R, Larsson S. Using dependency model to support software architecture evolution. In: Caporuscio M, ed. Proc. of the 2008 23rd IEEE/ACM Int'l Conf. on Automated Software Engineering Workshops. Láquila: IEEE Press, 2008. 82−91. [doi: 10.1109/ASEW.2008.4686324]

[10] Patikirikorala T, Colman A, Han J, Wang LP. A systematic survey on the design of self-adaptive software systems using control engineering approaches. In: Tamura G, Villegas N, eds. Proc. of the 2012 ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems (SEAMS 2012). Zurich: IEEE Press, 2012. 33−42. [doi: 10.1109/SEAMS.2012.6224389]

[11] Breivold HP, Crnkovic I, Larsson M. A systematic review of software architecture evolution research. Information and Software Technology, 2012,54(1):16−40. [doi: 10.1016/j.infsof.2011.06.002]

[12] Breivold HP, Crnkovic I, Larsson M. Software architecture evolution through evolvability analysis. Journal of Systems and Software, 2012,85(11):2574−2592. [doi: 10.1016/j.jss.2012.05.085]

[13] Sato GY, Azevedo HJSD, Barthès JPA. Agent and multi-agent applications to support distributed communities of practice: A short review. Autonomous Agents and Multi-Agent Systems, 2012,25(1):87−129. [doi: 10.1007/s10458-011-9170-9]

[14] Ranjbar-Sahraei B, Ammar HB, Bloembergen D, Tuyls K, Weiss G. Evolution of cooperation in arbitrary complex networks. In: Mailler R, ed. Proc. of the 2014 Int'l Conf. on Autonomous Agents and Multi-Agent Systems (AAMAS 2014). Istanbul: ACM Press, 2014. 677−684.

[15] Cheng SW. Rainbow: Cost-Effective software architecture-based self-adaptation [Ph.D. Thesis]. Pittsburgh: Carnegie Mellon University, 2008.

[16] Garlan D, Cheng SW, Huang AC, Schmerl B, Steenkiste P. Rainbow: Architecture-Based self-adaptation with reusable infrastructure. IEEE Computer, 2004,37(10):46−54. [doi: 10.1109/MC.2004.175]

[17] Gao L, Hailu A. Ranking management strategies with complex outcomes: An AHP-fuzzy evaluation of recreational fishing using an integrated agent-based model of a coral reef ecosystem. Environmental Modelling & Software, 2012,31:3−18. [doi: 10.1016/j.envsoft.2011.12.002]

[18] Viroli M, Holvoet T, Ricci A, Schelfthout K, Zambonelli F. Infrastructures for the environment of multiAgent systems. Autonomous Agents and Multi-Agent Systems, 2007,14(1):49−60. [doi: 10.1007/s10458-006-9001-6]

[19] Blair GS, Coulson G, Blair L, Duran-Limon H, Grance P, Moreira R, Parlavantzas N. Reflection, self-awareness and self-healing in OpenORB. In: Garlan D, Kramer J, Wolf A, eds. Proc. of the Workshop on Self-Healing Systems of 2002 10th Int'l Symp. on the Foundations of Software Engineering (FSE-10). New York: ACM Press, 2002. 9−14. [doi: 10.1145/582128.582131]

[20] Diao Y, Hellerstein JL, Parekh S, Bigus JP. Managing Web server performance with autotune agents. IBM Systems Journal, 2003, 42(1):136−149. [doi: 10.1147/SJ.2003.5386833]

[21] Liang X, Rabertson D, Croitoru M, Dashmapatra S, Dupplaw D, Hu B. Adaptive agent model: An agent interaction and computation model. In: Chang CK, ed. Proc. of the 31st Annual Int'l Computer Software and Applications Conf. (COMPSAC 2007). Beijing: IEEE Press, 2007. 153−158. [doi: 10.1109/COMPSAC.2007.51]

[22] Lü J, Ma XX, Tao XP, Xu F, Hu H. Research and development of network configuration software. Science in China (Series E: Information Sciences), 2006,36(10):1037−1080 (in Chinese with English abstract).

[23] Capar L, Emmerich W, Mascolo C. CARISMA: Context-Aware reflective middleware system for mobile applications. IEEE Trans. on Software Engineering, 2003,29(10):929−945. [doi: 10.1109/TSE.2003.1237173]

[24] Dam KH, Winikoff M, Padgham L. An agent-oriented approach to change propagation in software evolution. In: Gorton I, ed. Proc. of the Australian Software Engineering Conf. Los Alamitos: IEEE Press, 2006. 309−318. [doi: 10.1109/ASWEC.2006.10]

[25] Hussein M, Han J, Yu J, Colman A. Enabling runtime evolution of context-aware adaptive services. In: Ghose AK, Tekli J, eds. Proc. of the 2013 IEEE Int'l Conf. on Services Computing (SCC 2013). Santa Clara: IEEE Press, 2013. 248−255. [doi: 10.1109/SCC.2013.77]

[26] Tamura G, Villegas NM, Müller HA, Duchien L, Seinturier L. Improving context-awareness in self-adaptation using the DYNAMICO reference model. In: Sm M, ed. Proc. of the 2013 8th Int'l Symp. on Software Engineering for Adaptive and Self-Managing Systems (SEAMS 2013). San Francisco: IEEE Press, 2013. 153−162. [doi: 10.1109/SEAMS.2013.6595502]

[27] Mokni A, Huehard M, Urtado C, Vauttier S. Software architecture: Modeling and evolution management. In: Dalhoumi S, ed. Proc. of the Advances on Cognitive Automation at Laboratoire de Génie Informatique et d'Ingénierie de Production (LGI2P). Ecole des Mines d'Alès: IEEE Press, 2013. 26−29.

[28] Boukredera D, Maamri R, Aknine S. Modeling and analysis of reliable contract net protocol using timed colored Petri nets. In: Xu Y, ed. Proc. of the 2013 IEEE/WIC/ACM Int'l Joint Conf. on Web Intelligence (WI 2013) and Intelligent Agent Technologies (IAT 2013). Atlanta: IEEE Press, 2013. 17−24. [doi: 10.1109/WI-IAT.2013.85]

[29] Mukherjee S, Chaudhury S. Towards adaptive multi-agent planning in cyber physical space. In: Xu Y, ed. Proc. of the 2013 IEEE/WIC/ACM Int'l Joint Conf. on Web Intelligence (WI 2013) and Intelligent Agent Technologies (IAT 2013). Atlanta: IEEE Press, 2013. 445−450. [doi: 10.1109/WI-IAT.2013.63]

[30] Chen L, Qin XS, Yang Y, Gao ZP, Qu Z. The contract net based task allocation algorithm for wireless sensor network. In: Elmaghraby A, Tantug C, eds. Proc. of the 2012 IEEE Symp. on Computers and Communications (ISCC 2012). Cappadocia: IEEE Press, 2012. 000600−000604. [doi: 10.1109/ISCC.2012.6249362]

[31] Hsieh FS, Lin JB. A problem solver for scheduling workflows in multi-agents systems based on Petri nets. In: Seceleanu C, Yoshida K, eds. Proc. of the 2013 IEEE 37th Annual Computer Software and Applications Conf. on Workshops (COMPSACW 2013). Kyoto: IEEE Press, 2013. 316−321. [doi: 10.1109/COMPSACW.2013.47]

[32] Li QS, Wu G, Xue B, Chu H. A flexible multi-agent system model in system simulation. Information Technology Journal, 2011, 10(12):2371−2377. [doi: 10.3923/itj.2011.2371.2377]

附中文参考文献:

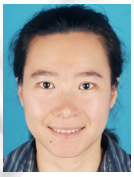
[2] 陶先平,冯新宇,李新,张冠群,吕建.Mogent 系统的通信机制.软件学报,2000,11(8):1060−1065. http://www.jos.org.cn/1000-9825/11/1060.htm

[3] 王千祥,申峻嵘,梅宏.自适应软件初探.计算机科学,2004,31(10):168−171.

[4] 常志明,毛新军,齐治昌.基于 Agent 的网构软件构件模型及其实现.软件学报,2008,19(5):1113−1124. http://www.jos.org.cn/1000-9825/19/1113.htm

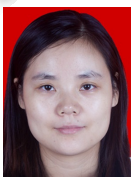
[22] 吕建,马晓星,陶先平,徐锋,胡昊.网构软件的研究与进展.中国科学(E 辑:信息科学),2006,36(10):1037−1080.

李青山(1973－),男,甘肃庆阳人,博士,教授,CCF 会员,主要研究领域为软件演化,Agent 技术,软件集成,软件体系结构,元搜索技术.



褚华(1977－),女,博士,副教授,主要研究领域为软件体系结构,逆向工程,程序分析.

王璐(1991－),女,博士生,主要研究领域为 Agent 技术,软件演化,软件体系结构.



张曼(1984－),女,博士,讲师,主要研究领域为软件工程,业务流程建模,工作流网.