第 40 卷 第 11 期 计 算 机 工 程 2014 年 11 月
Vol. 40 No. 11 Computer Engineering November 2014

·开发研究与工程应用· 文章编号：1000-3428(2014)11-0304-06 文献标识码：A 中图分类号：TP18

基于 Agent 的自主构件模型设计与评估

王茂光，王家瑞

（中央财经大学信息学院，北京 100081）

摘 要：为适应复杂环境和业务需求的变化，自适应软件开发要求为系统及其构成成分提供新的抽象和建模手段。自主构件能够感知并依据环境的变化自动地做出决策。采用自主构件开发复杂软件系统的主要目的是有效降低软件开发和维护的复杂性。在分析自适应软件实体功能和特征的基础上，提出一种基于智能体（Agent）的自主构件模型，该自主构件模型能动态感知环境的变化，合理封装自适应逻辑，依据策略规则进行决策，并根据目标导向，自动规划执行行为的序列。评估结果表明，该模型为建模与开发复杂自适应化软件提供了有效的底层支持。

关键词：智能体；自主构件；编程模型；自主度；评估

中文引用格式：王茂光，王家瑞. 基于 Agent 的自主构件模型设计与评估[J]. 计算机工程，2014，40(11)：304-309.

英文引用格式：Wang Maoguang，Wang Jiarui. Design and Evaluation of Autonomous Component Model Based on Agent [J]. Computer Engineering，2014，40(11)：304-309.

Design and Evaluation of Autonomous Component Model Based on Agent

WANG Maoguang，WANG Jiarui

（School of Information，Central University of Finance and Economics，Beijing 100081，China）

【Abstract】 Autonomous component is capable of sensing and making decisions automatically according to the environment changes. The purpose of developing complex software system based on autonomous component is to reduce the complexity of software development and maintenance effectively. Self-adaptive software development requires the system and constructive component should provide a new abstract and model method to adapt to complex environment and business requirements changes. After analyzing the software entity functions and characteristics，this paper proposes an autonomous component model based on intelligent agent. The autonomous component can sense the complex environment dynamic changes，encapsulates self-adaptation logic reasonably，makes decisions in accordance with policy rules，and plans the goal-directed behaviors automatically. Autonomous component provides the efficient support for modeling and developing complex self-adaptive software.

【Key words】 Agent；autonomous component；programming model；autonomy degree；evaluation

DOI：10.3969/j.issn.1000-3428.2014.11.060

1 概述

随着网络化软件系统所处的环境越来越开放、动态、难控，构成系统的软件实体也越来越异构、智能和自主。同时，系统的复杂性也不断提高，这使得系统的维护与演化的成本不断增加。现有的软件开发方法并不能很好地满足此类软件的设计需求，主要问题是网络环境的不稳定性、开放性和动态性对软件实体自适应环境的变化提出了新的技术挑战。这促使研究人员对在复杂网络环境下构造自适应软件实体及系统进行更深入的研究，有效降低系统的复杂性，增强系统自管理的能力[1-2]。

把能够适应环境变化，且自主进行决策的构件称为自主构件[3]。采用自主构件来开发软件系统的主要目的是使得软件在运行环境或业务需求发生变化时能主动适应环境的变化。一方面，从构件角度来看，自主构件与普通构件相同，它具有相对独立的功能，能够被复用和组装，成为系统的有机构成成分。另一方面，从自主性的角度来看，自主构件行为受到自身策略的驱动向外提供服务，并且可以在运

基金项目：国家自然科学基金资助项目“基于自主构件的开放分布式自适应软件系统的开发方法”（61073020）；国家留学基金资助项目（201306495003）。

作者简介：王茂光（1974－），男，副教授、博士，主研方向：分布式智能，软件工程；王家瑞，硕士研究生。

收稿日期：2013-11-15 **修回日期**：2014-04-23 **E-mail**：wangmg@cufe.edu.cn

行时刻收集系统的信息,感知变化从而做出决策。

本文设计的自主构件模型从两方面对构件的自主行为提供了支持:(1)构件模型提供了较强的自主行为描述能力,支持应用开发人员为构件设计特定于应用的自主行为;(2)对于复杂计算环境带来的具有一定普遍性的自适应需求,自主构件模型提供了自适应策略驱动,将带有普遍性的自适应需求以规则的形式存储到策略库中。

2　相关工作

自适应系统的目标体现为系统元素的自我管理以及它们之间的交互,因此,相关领域的研究人员十分关注构件的适应性问题[3]。

典型的 Fractal 是一个层次化、动态的构件框架[4]。Fractal 支持复合构件的概念,一个复合构件由多个原子构件或复合构件组成,并且可以在运行时动态修改 Fractal 应用系统的软件体系结构。文献[5]介绍了一种对 Fractal 构件模型进行扩展以支持自适应行为的方法,其所做的扩展主要包括:提供一个收集环境上下文信息的服务以获取运行时刻信息,基于"事件-条件-动作"的模式定义自适应策略。

而 Accord[6]是一种基于高层规则进行组织和协调的构件模型,主要由规约计算逻辑的功能端口、协调逻辑的控制端口和对自适应规则进行管理的操作端口 3 个部分构成。

K-Component[7-8]提供了一个自适应的构件编程模型,并基于合作强化学习技术提供了一个协调模型,支持在无中心环境下构建自适应系统。在 K-Component 框架中,封装自适应逻辑的适应合约使用特定的描述语言来定义,包含一组声明、表达式,以及监控操作和适应动作。

这些构件模型在具体表示上还是有区别的。在表示构件知识的状态模型方面,Accord 采用的是比较简单的"属性-值"对的方式,Fractal 则是通过资源本体对原始数据进行组织。K-Component 除了"属性-值"的方式之外,支持使用马尔可夫决策过程来组织自身的状态,这是为了更好支持其自适应策略的需要。

在自适应策略方面,Fractal 和 Accord 均是采用规则的方式来描述,这也是绝大多数自适应构件模型采取的方式,均支持对规则的在线修改。K-Component 除了规则外,还可以通过指定参数的方式支持构件执行强化学习策略,这样构件可以动态调整自己的行为。就评估时机而言,三者对于信息的收集和判断与计算逻辑都是正交的,即策略的评估与动作执行与构件提供服务是异步的,此外也有不少自适应构件模型的自适应策略评估与功能逻辑同步执行,如一些基于 AOP 技术、在功能请求到来前后收集信息并进行自适应评估。

现有研究工作中,对于自适应行为的支持往往限定于某种特定的决策机制。例如,Fractal 和 Accord 模型中,自适应主要通过一组预先定义好的策略规则来描述,这就意味着管理员需要对这个环境中可能出现的变化,其对应的运行时刻系统事件,以及相对应的解决方案有一个预先的了解,这一假设在复杂网络环境下有时无法成立[9];K-Component 模型通过强化学习技术可以在运行时刻调整自身的策略,然而由于缺少对网络环境特征的系统分析与支持,模型往往无法直接应用于自适应网络化软件的构造。

3　自主构件模型

针对新网络计算环境的特点,一种应对上述技术挑战的可行途径是不再将运行系统的构成单元视为被动的受管对象,而是将其建模为具有主动能力的软件实体。这些实体能够通过观察运行时刻信息,主动地对自身实现进行调整,或是改变与其他实体的交互;继而,系统通过这些构成单元局部间的相互作用表现出整体性的自适应行为。这类具有主动能力的计算实体,称为自主构件(Autonomous Component,AC)。

自主构件是具有自主性的构件,能够感知环境的变化,并根据环境的变化自动地做出决策响应环境的变化[10]。自主构件同时具备了构件以及 Agent 的关键性质。自主构件从能力要求来看,除了对外提供服务,自主构件区别于传统构件的能力主要有:

(1)功能性:能对外提供服务,可以通过调用外部构件的服务来实现自身的功能性需求;并具有质量属性评估的功能。

(2)自决策性:对功能性和非功能性目标的实现和改进具有决策能力。

(3)自适应性:自主构件对于不同应用环境具有适应能力,在不同场景下,对外所展现的行为会有所不同。能够决定如何选择和使用外部资源(依赖服务),并规范依赖服务的行为。

(4)交互性和协同性:能参与交互和合作,多个自主构件可以协作完成复杂的任务。

自主构件要能够感知环境,根据需要动态监控环境的变化,分析环境的动态变化,通过自适应策略驱动自主构件作出决策,并执行自身的行为,必要时作用于环境[11]。

从功能上分析,自主构件通过感知器感知环境在运行时刻获取所需要的信息,基于给定的自适应策略作出决策,确定需执行的调整行为,最后通过效

应器对系统中的受控资源进行管理,其核心功能是自适应决策。

自主构件的功能模块构成一个执行循环:监控→分析→决策→执行。监控部分负责监控环境的变化;分析部分对当前环境进行分析,建立决策模型,并使得自主构件可以对环境进行学习,帮助自主构件预测未来;决策部分通过决策规划实现目标的行为;执行部分执行规划得到的行为序列,并对执行过程控制管理。以上 4 个部分使用的数据作为共享的知识存储在知识库中,共享的知识包括网络拓扑信息、系统日志、策略库等。自主构件通过监控→分析→决策→执行来动态的监控环境变化并做出实时响应。

由于自主构件类似于 Agent 具有自主、目标驱动等特点,因此可以把自主构件看作具有特殊功能具有自管理、自决策特性的 Agent[11]。但面临的问题是,一方面,当前对 Agent 开发和运行平台的规范进行支持的实现提供商还是较少;另一方面,网络上的软件实体并非都是 Agent,且对 Agent 互操作协议等的支持有限,从而单纯基于 Agent 技术进行开发将在一定程度上限制其适用范围。因此,借鉴了传统构件的通用特征和 BDI Agent 特有的特点来设计开发自主构件。将策略/决策规则、质量属性评估引入构件模型,提出一种以决策为中心的自主构件模型。

定义 自主构件的形式化模型是一个六元组 $<K,A,G,P,R,Q>$。其中,K 为自主构件的知识库;A 为关于自主构件的行为集合;G 为自主构件的目标集合;P 为自主构件规划的集合;R 为自主构件策略集;Q 为自主构件质量属性评估函数。K 为自主构件的知识库,它是自主构件对当前状态下环境感知的一种反映和描述,包括环境状态、领域知识等。

自主构件的行为集 A 是一组行为描述的集合,它表明自主构件能够执行的行为,行为封装了自主构件执行的任务和自身的能力。自主构件的一个行为可看作是一个动作或多个动作的组合,动作执行的前提条件就是行为的前件,而动作执行后的整体效果就是行为的后件。定义具体的行为 $Beh \in A$, $Beh = (BehaviorID, Constraints, Params, Effects)$。*BehaviorID* 标识了执行的行为;*Constraints* 声明了执行行为的具体条件,一般为规则表达式;*Params* 是执行行为的输入参数;*Effects* 为指定行为执行后得到的结果。

自主构件的目标 G 就是自主构件准备努力去实施的动作或者达到的某个状态。目标有主次或轻重缓急之分,凡要求第一位达到的目标赋予优先因子 f_1,次位的赋予优先因子 $f_2,\cdots$,规定 $f_k \geqslant f_{k+1}$。若要区别具有相同优先因子目标的差别,可分别赋予不同的权重,目标 G 通常可以以分层的树状结构表示。$G=(Goals,Rels)$,分别表示原子目标的集合和目标关系的集合,$Goals=\{g_1,g_2,\cdots,g_m\}$,$Rels=\{\cap,\cup\}$ 表示原子目标复合的与、或关系。因为 2 个不同的目标往往有逻辑上的关系,所以进一步分析可得目标的互斥、蕴含、前件、等价、依赖关系[12]。

自主构件通过规划 P 生成具体的规划方案,每个自主构件都有一个规划库,用以存储实现某个目标的规划方案。规划 P 是自主构件执行的行动序列,它指明了自主构件为实现一定的需求和目标的主要方法和途径。自主构件间的交互在一定的时刻和状态下发生,设 $r_i=(c_i,b_i,t_i,g)$,$r_j=(c_j,b_j,t_j,v)\in I$,分别表示构件 c_i 在 t_i 时刻执行行为 b_i,实现目标 g;c_j 在 t_j 时刻执行行为 b_j,实现目标 v。自主构件间的多次交互就形成了一个行为序列。

R 为自主构件策略,指导自主构件的行为,策略驱动了构件在给定的环境状态下执行动作的决策过程,这一决策过程是构件自适应决策的核心[3]。自主构件可以封装多种决策过程,当决策过程为基于规则的前向推理时,具体策略为一个规则集;当决策过程为学习算法时,具体策略可以为一个马尔可夫决策过程等。

质量属性评估函数 $Q=\{f_1,f_2,\cdots,f_k\}$,是质量属性评估函数的集合,对于与领域相关的自主构件的执行结果进行质量属性评估,如自主度评估。

上述形式化模型将自主构件的功能和非功能属性(质量属性评估)规约在统一的模型内;模型支持策略驱动和学习算法,使自主构件感知自身和外部环境的变化,并根据目标规划和决策进行自我管理。

自主构件决策的关键过程是,给定目标 G:在当前状态 cs,设结果状态 rs,规划行为序列 $\alpha_1,\alpha_2,\cdots,\alpha_k$ 构成复合行为 $\alpha'=\alpha_1\oplus\alpha_2\oplus\cdots\oplus\alpha_k$,$\oplus$ 表示行为的复合算子(顺序、迭代、选择等),满足 $cs\xrightarrow{\alpha_1}s_1\xrightarrow{\alpha_2}s_2\xrightarrow{\alpha_3}\cdots\xrightarrow{\alpha_k}rs$,$s_i$ 为中间转移状态。具体的行为序列 $\alpha'=\alpha_1\oplus\alpha_2\oplus\cdots\oplus\alpha_k$ 构成具体的决策结果。

4 自主构件设计

在开放网络环境中,构件所依赖的服务质量和数量、链接的状态等都是动态变化的。理想状态下,具有自适应能力的构件能够在网络环境不断变化的情况下,无需外界进行显式控制,就能调整其自身内部行为或结构保证正常对外提供服务。

自主构件将数据、操作、质量属性封装在一起。自主构件是部署在网络上、能被第三方复用的软件实体,如图 1 所示。

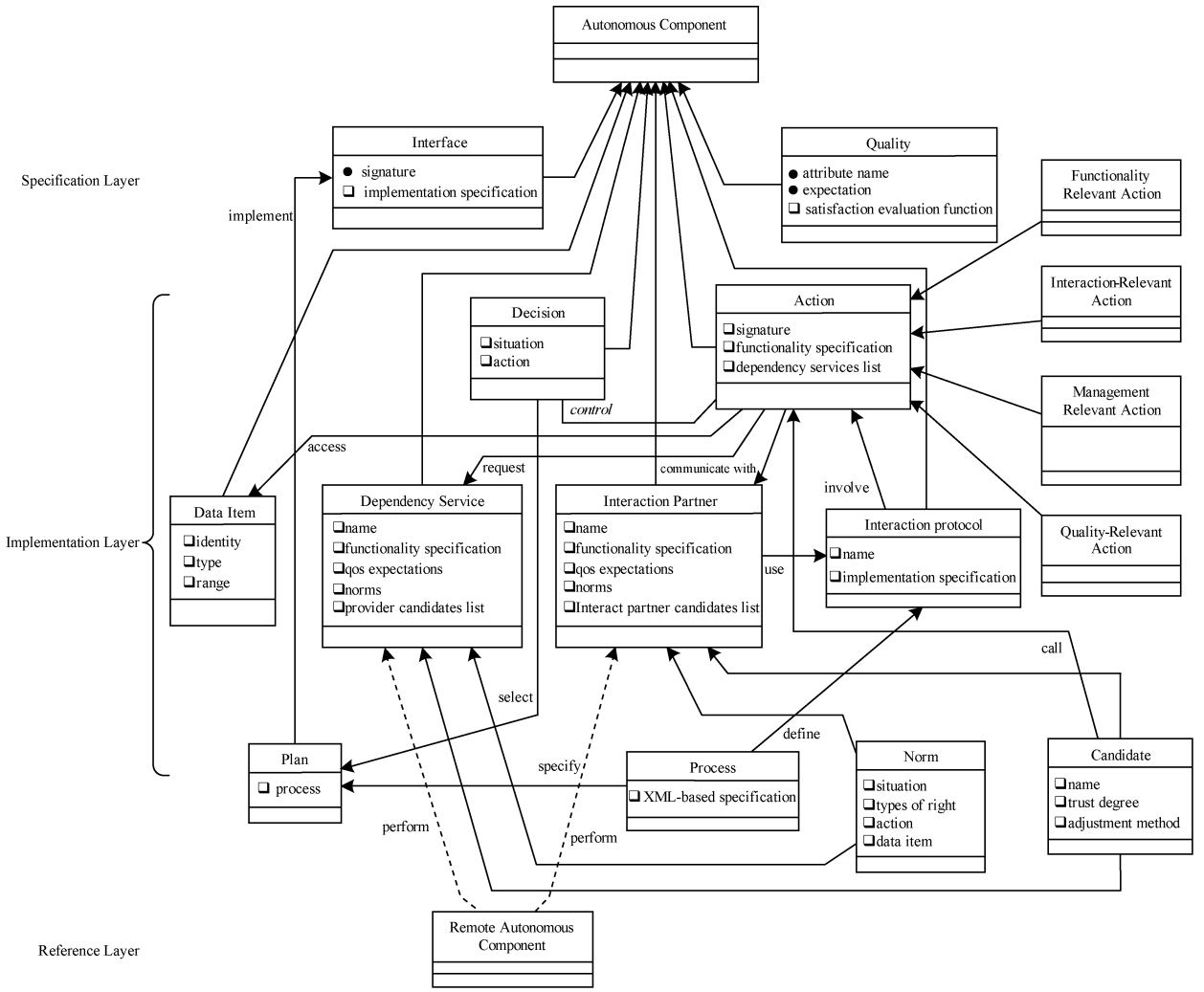

图 1　自主构件设计模型

自主构件主要由以下 8 个部分组成:

(1)接口:即自主构件对外部提供的服务接口,由一组操作来实现。主要由感知器和效应器 2 个接口组成,分别用于感知环境和作用于环境。

(2)质量属性:是影响质量的各种因素的评估函数,这些因素包括内部数据、内部操作、外部依赖服务以及交互对象等。质量属性与依赖因素之间的关系可以是相互促进的(正面影响),也可能是相互妨碍的(负面影响)。这是自主构件选择依赖服务和交互对象的主要根据。

(3)内部数据:由数据名、类型,以及允许的取值范围定义。往往用于记录自主构件内部和外部环境的动态变化。

(4)外部依赖服务:定义了自主构件对外部计算资源的依赖,以及对计算资源的计算能力和服务质量的期待。每个服务可能有多个服务提供者来提供,自主构件将根据提供者的服务状态来选择合适的提供者,以实现自身的目标、提供自身的服务质量。自主构件将会维护一个候选者的列表,并实时根据提供者的服务状态、以及对服务质量的期望,评估提供者的可信度,以选择合适的服务提供者。

(5)交互对象:定义了自主构件在合作过程中,对交互对象的期望列表。其对交互对象的规约与依赖服务类似,不同的是自主构件通过服务请求与依赖服务进行交互,而通过信息传递与交互对象进行交互。

(6)交互协议:自主构件在选择不同的交互对象后,需要根据交互对象确定具体的交互协议。交互协议定义了自主构件能够支持的交互方式。

(7)动作:定义了自主构件的具体行为,自主构件有能力执行多个不同并发的任务。自主构件中一个具体的任务被封装为一个行为,自主构件可以动态添加或删除封装的行为,行为成为自主构件执行的任务列表。

(8)决策:决定是否采取行动,以及如何采取行动,以便更好地实现自身的目标。有 4 种类型的决

策:1)功能性相关的决策,决定是否以及何时执行动作或提供服务,用何种方式(选择哪个规划)实现服务;2)质量相关的决策,决定如何选择服务提供者以及交互对象,如何保证服务提供者或交互对象遵循行为规范;3)交互相关的决策,负责选择交互对象以及交互协议,确定是否继续交互;4)自适应相关的决策,当内部状态和外部状态(主要是指外部服务提供者和交互对象的服务状态)发生变化时,如何应对。

在自主构件抽象规约的基础上,自主构件的基本构成及其运行机理如图 2 所示。

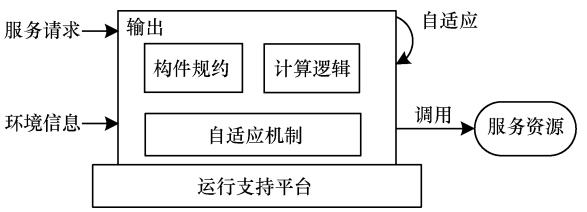

图 2 自主构件运行示意图

部署运行自主构件时,主要实现如下:

(1)每个自主构件实现由 3 个部分构成,包括构件的抽象规约、封装的计算逻辑,以及自适应支持机制。其中,构件的抽象规约包括对外发布的服务接口描述,以及对自身自适应行为的描述;计算逻辑描述了构件执行的业务逻辑(行为),以及与自适应相关的功能实现(如对内部参数的配置);自适应机制主要由资源管理模块、信息监控模块、决策模块,以及执行模块组成,它们分别为自主构件服务资源的管理,自适应行为的监控、分析、决策、执行提供支持。

(2)在运行时刻,自主构件处理外界的服务请求,同时监测与其自适应行为相关的环境信息,在此基础上决定应执行的决策方法,并在执行的过程中选择合适的服务资源来实现服务。

(3)底层支撑平台为自主构件提供了运行环境、互操作机制等公共服务。

当开发一个自主构件时,开发人员需要根据自主构件模型,物理实现相关的关键模块;这些信息被自主构件的部署工具所解析并映射到图 2 所示的运行时刻结构中。在运行时刻,自主构件运行于一个容器中,其计算行为建立在一组自主性相关的基础设施服务之上。

自主构件经过设计、实施、测试、验证,经过安装和配置,自主构件需要向系统注册安装和配置信息,以便在进行后续自决策时使用。安装配置完成后,自主构件进入自己的生命周期:首先进入就绪状态,随时等待运行。自主构件由就绪状态启动后进行初始化工作,完成以后进入活跃状态;当系统需要将其暂停时,自主构件进入挂起状态,直到有恢复命令自主构件才能回到活跃状态;当运行条件不满足时,自主构件自动进入等待状态,直到该自主构件被唤醒;当自主构件需要移动到其他节点时,则进入迁移状态,直到在目标节点开始执行,它才进入活跃状态。活跃状态包含监控、分析、决策、执行等功能状态。若自主构件退出,则进入新的就绪状态。在此状态下,如果被卸载,自主构件就进入了消亡状态,从而完成了其完整的生命周期。在活跃状态,自主构件通过决策引擎会循环检测其运行状态:正常态:系统各项性能指标处于正常;异常态:系统运行过程中,数据处理发生错误或者运行时发现故障;恢复态:更新、添加和删除无法正常运行的服务,调整自身状态;重构态:自主构件恢复失败时,系统进入重构状态,即对某些服务重新进行配置。自主构件生命周期如图 3 所示。

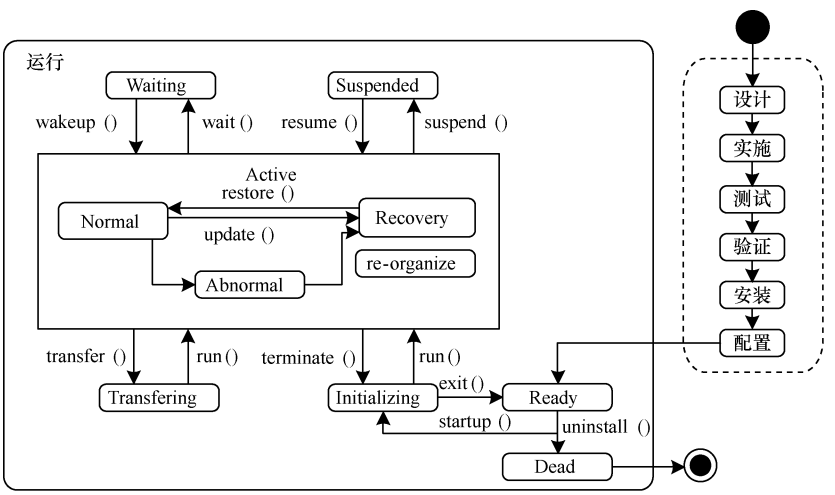

图 3 自主构件生命周期

5　自主度评估

自主构件的优势在于具有一定的自主性,能够主动适应环境的变化并做出决策。在具体应用领域,自主度要求映射到系统具体的质量属性中。以交通系统中的汽车为例,设计的目的是赋予汽车一定的自主度,如根据道路状况自主决定交叉路口行车路径的能力,测试自主度对其行为产生的影响。显然对于给定交通网络,车辆的速度、路面的拥堵率与路面上行驶的汽车有直接的关系。在初始自主度为 0.2(限制抽取的规则数/当前场景的总策略规则数),限速为 45 km/h 的情况下,系统通过设计道路、车辆、交通信号等模拟交通畅通、拥堵、事故等不同场景,自动决策行车路线[3]。其中,自主度简单量化为一种决策的能力,则模拟得交通拥堵场景下拥堵率如图 4 所示。

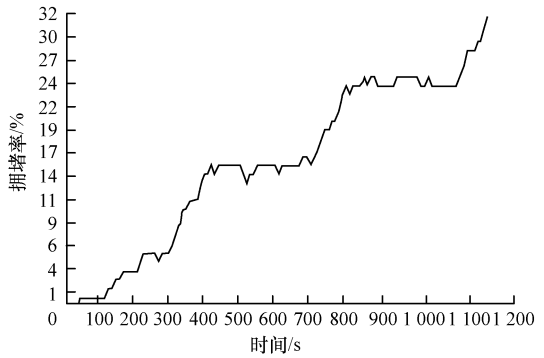

图 4　系统质量属性评估

经过分析验证,多数场景随着构件自主度的提高,可以提高系统的质量属性,如有效降低道路的拥堵率、提高平均行驶速度等。但在有的场景下,如路况复杂、车辆接近饱和,即使提高自主度值往往质量属性也会下降,因为此时个别自主构件决策复杂度的提高有时会与整个系统相冲突。因为当自主度过大时,在某些复杂度很高的环境下,由于自主构件的感知范围或推理能力的局限性,过分依赖自身的决策会导致一系列问题,具体体现为,车辆的速度明显下降,车辆在低速时的油耗率会大大增加等实际属性值。

6　结束语

本文根据自适应软件的开发要求,提出了一种基于 Agent 的自主构件模型,分析并实现了自主构件的规约和物理结构。该自主构件模型能动态感知环境的变化,通过自适应策略支持自主决策,并有一定的自主度能执行自身行为对环境变化做出响应。下一步将研究自主构件的决策能力对整个系统的影响,并对其质量属性进行评估和验证。

参考文献

[1] 张海俊,史忠植. 自主计算软件工程方法[J]. 小型微型计算机系统,2006,27(6):1077-1082.

[2] 张海俊. 基于主体的自主计算研究[D]. 北京:中国科学院研究生院,2005.

[3] 王茂光. 基于自主构件的自适应网构软件开发方法[D]. 北京:北京大学,2011.

[4] Bruneton E, Coupaye T, Leclerc M, et al. An Open Component Model and Its Support in Java [C]// Proceedings of International Symposium on Component-based Software Engineering. Edinburgh, UK: [s. n.], 2004:7-22.

[5] David P C, Ledoux T. Towards a Framework for Self-adaptive Component-based Applications[C]//Proceedings of International Conference on Distributed Applications and Interoperable Systems. Paris, France: [s. n.], 2003:1-14.

[6] Liu Hua, Parashar M. Accord: A Programming Framework for Autonomic Applications [J]. IEEE Transactions on Systems, Man, and Cybernetics, 2006, 36(3):341-352.

[7] Dowling J, Cahill V. The K-component Architecture Meta-model for Self-adaptive Software[C]//Proceedings of the 3rd International Conference on Metalevel Architectures and Separation of Crosscutting Concerns. [S. l.]: Springer, 2001:81-88.

[8] Dowling J. The Decentralised Coordination of Self-adaptive Components for Autonomic Distributed Systems [D]. Dublin, Ireland: University of Dublin, 2004.

[9] Cheng B H C, Lemos R, Giese H, et al. Software Engineering for Self-adaptive Systems: A Research Roadmap[EB/OL]. (2013-04-17). http://dx. doi. org/10. 1007/978-3-642-02161-9_1 2009.

[10] Jiao Wenpin. Using Autonomous Components to Improve Runtime Qualities of Software [J]. IET Software, 2011, 5(1):1-20.

[11] Wang Maoguang, Jie Junjing, She Tingxun, et al. An Agent-based Autonomous Component Model for Internetware [C]//Proceedings of International Conference on Web Information Systems and Mining. [S. l.]: IEEE Computer Society, 2010:348-352.

[12] DeLoach S A, Miller M. A Goal Model for Adaptive Complex Systems [J]. International Journal of Computational Intelligence: Theory and Practice, 2010, 5(2):83-92.

编辑　顾逸斐