软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2017,28(7):1676−1697 [doi: 10.13328/j.cnki.jos.005257]

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

基于验证的自适应系统决策:一种模型驱动的方法*

杨卓群[2], 金 芝[1,3]

[1](高可信软件技术教育部重点实验室(北京大学),北京 100871)

[2](中国科学院 数学与系统科学研究院 数学研究所,北京 100190)

[3](北京大学 信息科学技术学院 软件研究所,北京 100871)

通讯作者: 金芝, E-mail: zhijin@pku.edu.cn

摘 要: 自适应系统需要根据运行时上下文和自身的变化进行其行为的调节.为实现自主调节,自适应系统必须被赋予运行时监测上下文和自身变化、分析需求满足程度的变化以及推理得到自适应决策的能力.这种在线决策的行为在满足功能需求的同时,还需要保证系统满足特定的非功能需求,如可靠性和性能等.提出了一种基于验证的自适应系统优化决策方法,以保证非功能需求的满足.该方法在识别可调节目标以建模自适应机制的同时,将系统的目标模型映射为相应的行为模型,用标签转移系统表示;以可靠性需求为例,用标记目标模型规约任务的可靠性;然后将系统行为模型和可靠性规约整合为带可变状态的离散时间马尔可夫链,将候选自适应配置描述为不同可变状态间的组合;最终通过相关需求的在线验证,使系统找到关于某类上下文的最优决策配置.通过一个移动信息系统的案例展示了该方法的可行性和有效性.

关键词: 自适应系统;优化决策;需求建模;模型驱动;需求验证

中图法分类号: TP311

中文引用格式: 杨卓群,金芝.基于验证的自适应系统决策:一种模型驱动的方法.软件学报,2017,28(7):1676–1697. http://www.jos.org.cn/1000-9825/5257.htm

英文引用格式: Yang ZQ, Jin Z. Verification based decision-making for self-adaptive systems: A model-driven approach. Ruan Jian Xue Bao/Journal of Software, 2017,28(7):1676−1697 (in Chinese). http://www.jos.org.cn/1000-9825/5257.htm

Verification Based Decision-Making for Self-Adaptive Systems: A Model-Driven Approach

YANG Zhuo-Qun[2], JIN Zhi[1,3]

[1](Key Laboratory of High Confidence Software Technologies of Ministry of Education (Peking University), Beijing 100871, China)

[2](Institute of Mathematics, Academy of Mathematics and Systems Science, The Chinese Academy of Sciences, Beijing 100190, China)

[3](Insitute of Software, School of Electronics Engineering and Computer Science, Peking University, Beijing 100871, China)

Abstract: Self-Adaptive systems (SASs) are required to be capable of adjusting their behaviors in response to changes in operational contexts and themselves. To implement automatic adjustment, SASs must be endowed by abilities of monitoring changes in contexts and themselves, analyzing changes of requirement satisfaction and reasoning about adaptation decisions. The behavior of online decision-making needs to assure functional requirements as well as certain non-functional requirements such as reliability and performance. This paper proposes a verification-based optimal decision-making approach for SASs, for assuring the satisfaction of non-functional requirements. This approach models adaptation mechanisms by identifying adjustable goals and maps goal models to corresponding behavior models expressed by label transition systems. It takes reliability requirements as examples and utilizes tagged goal models to specify reliability of tasks. Then, the system behavior model and reliability specifications are integrated into discrete-time

*基金项目: 国家重点基础研究发展计划(973)(2015CB352200); 国家自然科学基金(61620106007, 91318301)

Foundation item: National Basic Research Program of China (973) (2015CB352200); National Natural Science Foundation of China (61620106007, 91318301)

收稿时间: 2016-09-27; 修改时间: 2016-11-29; 采用时间: 2017-01-20; jos 在线出版时间: 2017-02-20

CNKI 网络优先出版: 2017-02-20 14:05:52, http://www.cnki.net/kcms/detail/11.2560.TP.20170220.1405.016.html

Markov chains with variable states while adaptation candidates are characterized by combinations of different variable states. Via online verification of related requirements, the system derives the optimal decision of configurations under a certain type of contexts. The feasibility and effectiveness of the approach are illustrated through a mobile information system.

Key words: self-adaptive system; optimal decision-making; requirements modeling; model-driven; requirements verification

随着软件系统越来越深入地嵌入到人类社会和物理社会中,软件在运行时会与其他软/硬件系统、设备和用户发生密切的交互.软件运行环境变得不稳定并不断变化,自适应性成为软件系统的一个重要特性.自适应系统(self-adaptive system)能够根据环境和自身的变化对其行为进行控制和调节,从而使系统能够持续提供服务并满足用户的需求[1].自适应能力[2]在许多软件应用领域中显得越来越重要,如移动计算、环境智能和物联网等.

上下文(context)是软件运行所处环境的具体化且可操作化的描述[3].软件系统的自适应具体表现为系统随上下文的不同表现出不同的行为.为保证自适应软件系统在运行时能够根据上下文的变化进行自主调节,分析人员应在需求工程(requirements engineering)阶段识别自适应需求,规约自适应逻辑并建立有效的自适应机制[4].此外,还需进行相关的上下文分析,将系统行为关联到对应的上下文状态.这样,软件系统才能在运行时根据定义好的自适应机制完成上下文感知、需求感知、行为分析和优化决策等一系列任务.

自适应软件系统建模和分析的研究领域存在许多挑战性问题,包括建立自适应机制、实现自适应决策以满足用户需求和自适应系统在线需求验证等.在建立自适应机制方面,多数工作从目标模型出发为系统建立自适应机制,其中包括基于目标模型语义扩展的自适应机制[5,6]、基于反馈回路的自适应机制[7]和基于模糊逻辑量化非功能需求的自适应机制[8,9].在实现自适应决策方面,现有工作主要包括基于方程求解的自适应决策方法[10,11]和基于命题逻辑推理的自适应决策方法[12,13].自适应系统在线需求验证方面主要包括关于验证方法的工作[14,15]和关于验证方法应用的工作[16−19].

系统的自适应决策是一种在线行为,需要同时考虑系统业务能力的调整和系统特定的非功能需求(如可靠性、性能等可量化的非功能需求)的满足.然而,现有工作缺乏对这两个方面的综合考虑,有的仅关注决策过程中的推理计算,有的则仅考虑运行时如何验证非功能需求,两者的分离在很大程度上降低了方法的可用性.

本文以面向目标的需求建模方法为出发点,结合目标满足性驱动的在线决策,提出基于验证的自适应决策框架(verification-based adaptation decision making,简称 VADEM).该框架将业务行为决策过程与需求验证过程有机结合,通过对特定非功能需求的可满足性验证来实现自适应系统运行时的优化决策.其关键要点是:(1) 从业务需求出发构建业务目标模型,并基于 MAPE-K 回路[20]构建自适应机制的目标模型.提出了一种基于模式映射的模型转换方法,将需求目标模型转换为相应的行为模型,并表示为标签转移系统;(2) 为分析系统特定的非功能需求,如可靠性,引入系统任务的可靠性规约.依据系统任务的不确定性,将任务可靠性量化为任务的失效概率,不同的任务失效概率分布会产生不同的系统可靠性;(3) 为实现优化决策,将行为模型和任务的可靠性规约融合为可靠性分析模型,表示为离散时间马尔可夫链;可靠性需求描述为基于逻辑的性质.最终,通过概率模型检验技术求解最优的系统决策配置.

本文的主要工作包括:

- 提出了一种基于目标模型的业务需求和自适应机制统一建模框架,以及从目标模型转换为系统行为模型的行为描述生成算法,并通过分析工具进行仿真实现.
- 提出了一种任务规约描述方法,以及一个以特定非功能需求为优化决策目标,以运行时需求验证为基础的决策方法,并借助概率模型检验实现自适应优化决策过程.

本文第 1 节对 VADEM 的框架和各组成部分进行详细说明.第 2 节介绍论证案例.第 3 节建模自适应目标模型.第 4 节介绍系统行为模型的转换过程.第 5 节以可靠性为例,介绍任务可靠性的规约并构建可靠性的分析模型.第 6 节给出基于概率模型检验的自适应决策过程和分析结果.第 7 节对相关研究工作进行讨论总结和对比分析.第 8 节总结全文,并指出未来值得关注的研究方向.

1 VADEM 方法框架

VADEM 将业务逻辑与自适应逻辑相结合,将静态的需求模型与动态的行为模型相结合,通过基于系统分析模型的需求验证过程实现自适应决策.其框架如图 1 所示,包括建模和运行两大阶段.

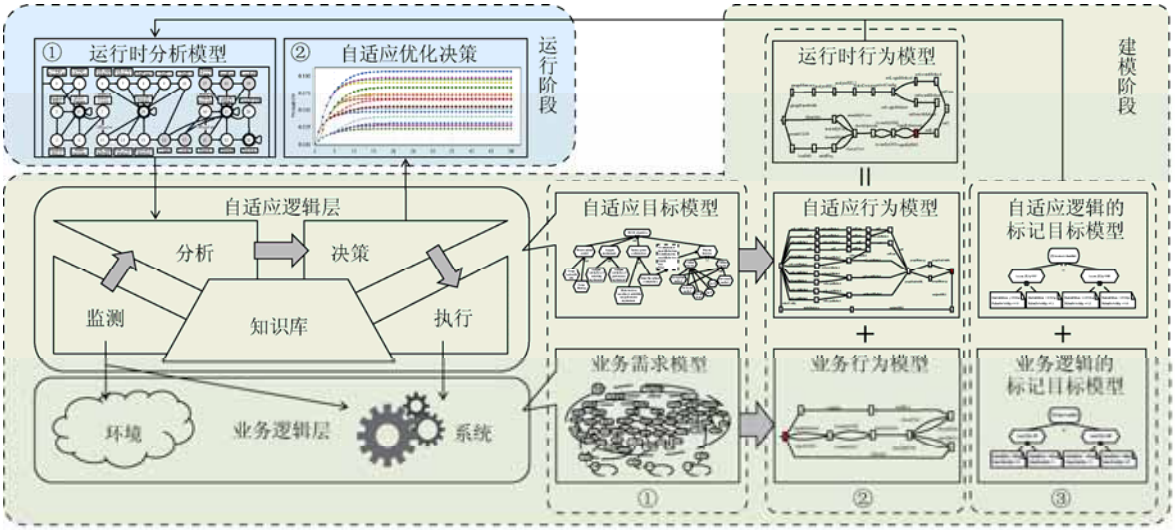


Fig.1 Framework of VADEM method

图 1 VADEM 方法框架

在建模阶段,自适应软件系统被刻画为业务逻辑层和自适应逻辑层两个层次.其中,业务逻辑层刻画了软件系统需要完成的功能目标及系统需要执行的任务;自适应逻辑层在业务逻辑的基础上刻画了系统的自适应目标及相关的自适应机制.自适应逻辑根据 MAPE-K[20]回路构建,包括监测(monitor)、分析(analyze)、决策(plan)和执行(execute)4 个任务以及 1 个知识库(knowledge base).监测组件通过系统的探针或传感器收集环境和系统中的数据;分析组件利用收集到的数据通过计算或者推理等方式,判断系统是否需要进行自适应的调整;决策组件用于获得能够满足决策目标的最优配置;执行组件将通过决策得到的最优配置部署于系统,实现对系统行为的调节.知识库包括两个部分:静态部分包括系统和环境的相关知识、决策目标和决策约束等;动态部分包括实现自适应任务所需要的知识,如系统组件的失效概率、能耗数据等,它们可以从系统运行时产生的日志中抽取.建模阶段得到的模型制品也存储在知识库中,作为运行时系统行为分析的基础.基于 MAPE-K 回路的自适应逻辑不仅用于建模阶段构建系统的自适应目标模型,还用于运行阶段支持自适应优化决策的实现[21].VADEM 依据“监测-分析-决策-执行”这 4 个自适应任务构建自适应逻辑的概念模型,VADEM 框架在建模阶段主要包括如下 3 个步骤.

(1) 分别构建业务需求模型和自适应目标模型:根据业务需求构建业务需求模型(见第 2 节),在业务需求模型的基础上识别出可调节目标(见第 3.1 节),构建自适应目标模型(见第 3.2 节).

(2) 构建系统运行时行为模型:基于建模元素关系的形式化描述(见第 4.1 节),将目标模型中的关系模式映射为行为模式(见第 4.2 节),并引入从目标模型获取行为模型的过程(见第 4.3 节).从业务需求模型和自适应目标模型中分别提取业务行为模型和自适应行为模型,并将两者结合为系统行为模型(见第 4.4 节).

(3) 规约系统行为:针对特定的非功能需求,描述系统单个任务的规约,比如可靠性等(见第 5.1 节).

在运行阶段,系统根据 MAPE-K 回路实现自适应优化决策,决策目标为待满足的特定非功能需求,决策过程与需求验证过程有机结合.因此,系统在运行时需要针对不同类型的非功能需求构建不同的分析模型,通过验证的方法来分析这些需求的满足程度.VADEM 框架在运行阶段主要包括如下两个步骤.

(1) 构建系统运行时分析模型:将系统行为模型和任务规约结合生成运行时的系统分析模型(见第 5.2 节),并将特定的非功能需求描述为基于逻辑的性质(见第 5.3 节).

(2) 获得自适应优化决策:监测相关上下文,利用模型检验技术,对特定非功能需求进行验证并求解最优任务配置(见第 6 节).

2 研究案例

商业区中商铺的传统销售方式表现为,通过销售人员向消费者详细地介绍促销信息,吸引消费者并使其购买商品.为了能使更广泛的目标受众便捷地获取商品信息,并能在浏览信息后及时购买所需商品,销售者希望构建一个商业区移动信息系统(mobile information system for business district),以下简称为 MobIS.MobIS 是一个基于位置信息的移动端应用系统,其业务需求(business requirement)为:用户希望能够随时随地获取周边多样的商品促销信息(BR1)和用户可以根据自己的需要,在线购买商品(BR2).

系统在实现业务需求的同时,还需尽可能地满足用户提供的一些使用偏好(preference),如:用户希望能够获得更新颖的促销信息(P1),用户希望接收信息的体验更加丰富(P2),用户希望更迅速地获取促销信息(P3),用户希望系统接收信息的响应速度更快(P4),用户希望操作更加简便(P5).

由于用户所处环境网络是多变的,系统需要提供不同的获取信息业务逻辑:连网的用户可以选择在线方式获取信息;无网络的用户可以选择离线方式获取信息.为保证用户能随时获取信息,系统应支持按用户给定的频率推送信息;为保证用户能随地获取周边的商品信息,系统需要提供定位用户的功能;为保证信息多样化,系统需要支持获取不同格式的信息,如媒体格式或者文本格式;为使用户的操作更简便,系统需要提供一些智能化的操作,比如语音搜索等.

需求分析人员需要针对业务需求构建业务逻辑.VADEM 采用 i*框架[22]的建模元素,通过面向目标的方法对软件系统的业务逻辑进行需求建模.建模元素包括以下几类:主体(actor)用于建模的系统和环境,系统主体内部描述了系统的业务逻辑,环境主体描述系统运行所依赖的计算环境;目标(goal)描述了系统的功能需求;任务(task)用于描述实现目标所要执行的动作;分解关系(decomposition link)用于刻画目标之间或任务之间的精化关系,“与”分解用 and 表示,“或”分解用 or 表示;方法-目标关系(means-ends link)针对实现同一目标提供了不同的任务选项,也表达了一种“或”关系;任务依赖(task dependency)是指一个主体所执行的任务依赖于其他主体.此外,添加有序关系(ordered link)用于描述通过“与”分解得到的同一层次上不同目标(或任务)间实现(或执行)顺序上的先后关系.通过对业务目标的分解,可得到图 2 所示的 MobIS 业务需求模型.

由于网络的带宽会影响用户的定位任务,设备的内存会影响系统的下载任务,银行会影响用户的在线支付,因此这三者是系统运行所依赖的外部主体.MobIS 主体内部描述了系统业务逻辑的分解与实现,本文主要考察“在线信息获取”和“离线信息获取”两条业务逻辑.根据图 2 所示需求模型中分解得到的任务,实现“在线信息获取”的任务集合为 $\{t1,t2,t9,t3,t12,t13,t14\}$;实现“离线信息获取”的任务集合为 $\{t1,t2,t9,t4,t17\}$.在这两个任务集合中,均包括有多个执行选项的任务,比如 $t1$ 可以通过 $t5$ 或 $t6$ 实现,$t2$ 可以通过 $t7$ 或 $t8$ 实现,$t12$ 需要通过设定 T_0 的数值实现,$t13$ 需要通过设定 B 的数值实现.也就是说,每条业务逻辑具有多种实现方式.具体采用哪种方式,需要在运行时根据情况进行决策.由于不同配置的任务在运行时上下文中具有不同的特点,比如失效概率不同、能耗不同等,这会影响系统某些特定非功能需求的满足,比如可靠性需求、性能需求等.因此,MobIS 需要具有自适应能力,调节任务在不同上下文中的配置,以确保这些特定的非功能需求得到最大程度的满足.本文将如下两条重要性递减的可靠性需求作为系统的自适应决策目标.

- REL1(启动后接收成功的可靠性). 用户从启动系统到成功接收信息的概率应尽可能地不小于 0.90.
- REL2(登录后接收失败的可靠性). 用户登录系统后接收信息失败的概率应尽可能地不大于 0.05.

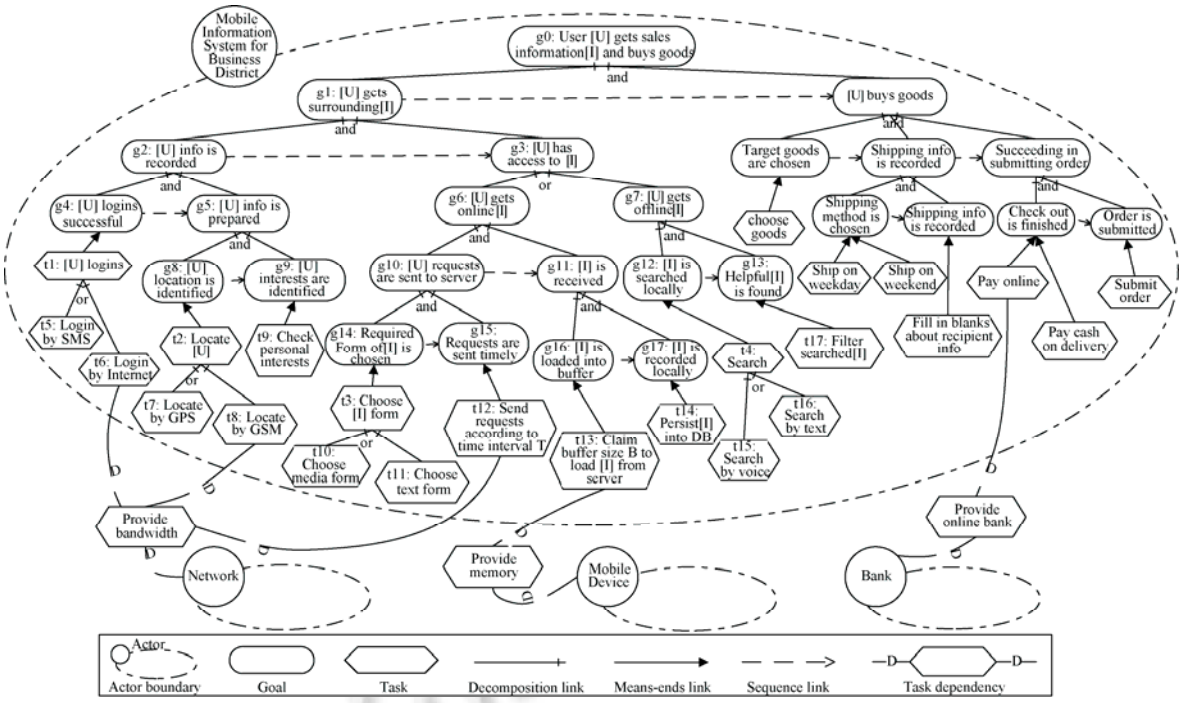


Fig.2 Requirements model of MobIS

图 2 商业区移动信息系统的需求模型

3 从业务目标到自适应机制

本节首先介绍可调节业务目标的类型和识别规则,然后根据识别出的可调节目标对自适应机制任务进行实例化,以构建自适应目标模型,刻画自适应逻辑.

3.1 可调节目标的识别

根据系统自身调节方式的不同,自适应可分为结构自适应(structural adaptation)和参数自适应(parametric adaptation)两类[23].结构自适应是指系统通过配置不同的组件来实现调节,参数自适应是指系统通过配置组件中相关参数来实现调节.

对应到面向目标的需求模型中,通过带结构选项的配置任务直接完成的目标应被识别为底层结构可调节目标;通过带参数选项的配置任务直接完成的目标应被识别为底层参数可调节目标,它们都属于模型中叶子节点目标,直接关联到任务.可调节目标具有传递性,如果子目标是可调节目标,那么其父目标也是可调节目标.并且,如果父目标只包含一种类型的可调节目标,那么它也是这种类型;如果父目标包含两种类型的可调节目标,则称为混合可调节目标.

图 2 中,底层结构可调节目标为{$g4$,$g8$,$g12$,$g14$},底层参数可调节目标为{$g15$,$g16$}.通过传递性识别出的结构可调节目标为{$g2$,$g5$,$g7$},参数可调节目标为{$g11$},混合可调节目标为{$g1$,$g3$,$g6$,$g10$}.

3.2 实例化自适应目标模型的构造

可调节目标的实现方式有多种选择,具体采用哪种方式则需要通过自适应决策过程来确定.比如,图 2 中的底层可调节目标 $g8$ 最终分解为两个可选的系统任务,即 $t7$ 和 $t8$.运行时,系统需要选择其中一个来实现定位功能.为此,系统需要执行具体的自适应任务,包括监测带宽大小,验证系统当前配置是否满足非功能需求;若发现非功能需求的满足程度没有达到预期值,则需要进行自适应决策并选择最优的定位方式进行部署.

根据每个底层可调节目标,确定图 1 所示的 4 个自适应任务的实例层任务,就是对自适应机制概念模型的实例化.实例化过程需要回答的问题包括:系统需要监测哪些环境参数?分析哪些需求的满足程度?决策并配置哪些任务结构或者任务参数?这些问题可以通过以下 4 个步骤来回答.

(1) 监测任务实例化:业务逻辑中任务所依赖的环境都应通过监测任务预以监测.

(2) 分析任务实例化:用户提出的特定非功能需求在当前配置下的满足程度都应进行分析.

(3) 决策任务实例化:决策任务由两个实例构成,首先要根据决策目标求得可选配置,然后按照某种筛选原则从中选取最优配置.

(4) 执行任务实例化:经决策确定的系统配置应于当前业务逻辑结束后、下个业务逻辑开始前进行部署.业务逻辑结束的标志为:顶层业务目标得以实现,或者业务流程中,由于某任务失败而引发业务逻辑终止.

按照以上步骤,信息获取逻辑对应的自适应机制通过实例化,可以得到如图 3 所示的自适应目标模型.监测任务实例化为测量带宽和内存,监测实例的执行顺序可以通过不同的排列组合得到.分析任务实例化为按重要性顺序依次分析 REL1 和 REL2 的满足情况.决策任务包括“根据可靠性需求进行决策”和“筛选最优配置”两个子任务,决策过程需要基于系统运行时的分析模型得出.灰色任务的具体内容分别在第 5 节和第 6 节详细介绍.决策任务完成之后,系统得到了一组最优配置,每一个配置都需要通过相应的执行任务部署到系统中.与监测任务实例的顺序类似,执行任务实例的完成顺序也有多种选择.

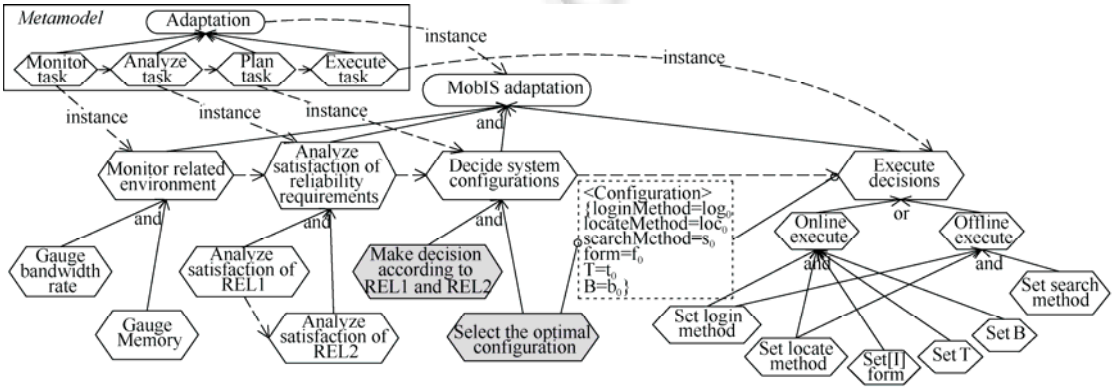


Fig.3 Adaptation goal model

图 3 自适应目标模型

4 从目标模型到系统行为模型

为了能够得到运行时的系统行为模型,VADEM 将业务需求模型和自适应目标模型中的元素关系模式映射到行为模式,提出从目标模型生成行为描述的算法.由算法生成的行为描述可以通过分析工具仿真生成相应的行为模型.

4.1 符号描述

首先引入一组术语.

- 原子任务是指需求模型和自适应目标模型中的叶节点任务,它们无法继续分解.
- 抽象任务是指有限多个任务的封装,但这些任务具体如何执行并不作为关注对象,抽象任务用 τ 表示.
- 任务序列 TS(task sequence)是指具有先后顺序的一组任务,用 $t_1 \rightarrow t_2 \rightarrow \ldots \rightarrow t_n$ 表示,$t_i \in T$ 可以是原子任务或抽象任务,T 是系统的任务集合.比如,序列 $t_1 \rightarrow \tau \rightarrow t_2$ 描述了系统最开始执行的任务是 t_1,最后执行的任务是 t_2,中间也执行了若干任务,而且它们被封装在其中.
- Op 是序列算子的集合,描述了任务序列之间的关系,包括前缀($\rightarrow$)、顺序(;)、并发(||)和选择($\square$).

♦$\rightarrow$:一个首先执行任务 t,随后表现任务序列 TS 的行为描述为 $t \rightarrow TS$.

♦;:任务序列 *TS*1 和 *TS*2 的顺序合成记作 *TS*1;*TS*2,其行为表现为 *TS*1,在 *TS*1 终止后表现为 *TS*2.

♦||:任务序列 *TS*1 和 *TS*2 的并发合成记作 *TS*1||*TS*2,其行为表现为 *TS*1;*TS*2 或 *TS*2;*TS*1.

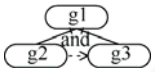
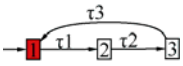
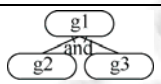
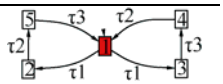
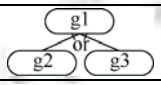
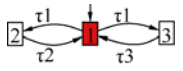
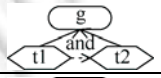
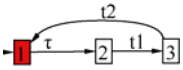
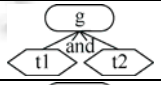
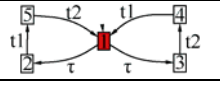
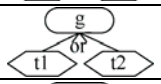
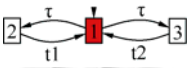
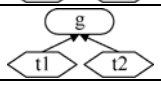
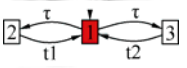
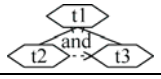
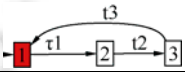
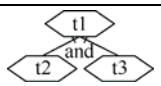
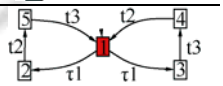
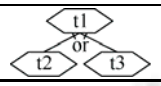
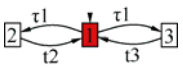
♦□:选择算子允许系统根据执行的当前任务选择后续的行为.(t_1→*TS*1)□(t_2→*TS*2)的行为描述为如果 t_1 被执行,则后续表现为任务序列 *TS*1;如果 t_2 被执行,则后续表现为任务序列 *TS*2.

4.2 从关系模式到行为模式的映射

为实现目标模型到行为模型的转换,本节首先考察目标模型中不同关系模式表现出的行为模式,并以此作为目标模型转化为行为模型的基本原理.表 1 按照分解关系的类别展示了 10 条映射规则,这些映射规则将不同的关系模式映射为相应的行为模式.表 1 仅用两个子节点的分解关系作说明,但映射原理对 *n* 个子节点的分解关系同样适用.

Table 1 Mapping from relation patterns to behavior patterns

表 1 关系模式到行为模式的映射

关系类别	关系 ID	关系名称	关系模式	算子	映射描述	行为模式
目标间的分解关系	1	目标间的有序与分解		;	TS1=τ1→(TS2;TS3) TS2=τ2→TS3 TS3=τ3→TS1	
	2	目标间的无序与分解		‖	TS1=(τ1→TS2)[](τ1→TS3) TS2=τ2→τ3→TS1 TS3=τ3→τ2→TS1	
	3	目标间的或分解		□	TS1=τ1→(TS2[]TS3) TS2=τ2→TS1 TS3=τ3→TS1	
目标到任务的分解关系	4	目标到任务的有序与分解		;	TS=τ→(TS1;TS2) TS1=t1→TS2 TS2=t2→TS	
	5	目标到任务的无序与分解		‖	TS=(τ→TS1)[](τ→TS2) TS1=t1→t2→TS TS2=t2→t1→TS	
	6	目标到任务的或分解		□	TS=τ→(TS1[]TS2) TS1=t1→TS TS2=t2→TS	
	7	方法-目标关系		□	TS=τ→(TS1[]TS2) TS1=t1→TS TS2=t2→TS	
任务间的分解关系	8	任务间的有序与分解		;	TS1=τ1→(TS2;TS3) TS2=t2→TS3 TS3=t3→TS1	
	9	任务间的无序与分解		‖	TS1=(τ1→TS2)[](τ1→TS3) TS2=t2→t3→TS1 TS3=t3→t2→TS1	
	10	任务间的或分解		□	TS1=τ1→(TS2[]TS3) TS2=t2→TS1 TS3=t3→TS1	

关系模式表现出不同的业务逻辑:关系 1、关系 4 和关系 8 描述了分解得到有序目标或任务的逻辑;关系 2、关系 5 和关系 9 描述了分解得到无序目标或任务的逻辑,它们均存在两条可执行的任务序列;关系 3、关系 6、关系 7 和关系 10 描述了有选择性完成目标或执行任务的逻辑,它们也存在两条可执行的任务序列.

不同的关系模式对应不同的序列算子,并可以通过第 4.1 节引入的符号刻画该关系模式表现的行为.由关系模式得出映射描述的过程需要遵循以下规则.

- 目标映射为实现该目标所需要执行的任务序列.
- 任务映射为完成该任务所执行的子任务序列.
- 描述中等式左侧为任务序列,右侧为该任务序列的行为表现.
- 任务序列的执行顺序由关系模式的逻辑顺序对应的序列算子指定.

- 分解关系按相应逻辑可以映射为相关任务序列的顺序、并发或选择运算.
- 方法-目标关系映射为相关任务序列的选择运算.
- 只有叶节点映射的任务序列中包含具体执行的任务.
- 抽象任务用于行为的显示化表示.

通过这些规则得出的映射描述能够符合关系模式表达出的业务逻辑.比如,在关系 1 的映射描述中,*TS*1,*TS*2 和 *TS*3 分别表示完成 *g*1,*g*2 和 *g*3 所需要执行的任务序列.*TS*1=τ1→(*TS*2;*TS*3)是指 *TS*1 通过τ1 进入 *TS*2 和 *TS*3 的顺序合成.它表达的逻辑为 *g*1 需要通过 *g*2 和 *g*3 来实现.*TS*2=τ2→*TS*3 描述了 *TS*2 表现为先执行τ2,然后进入 *TS*3.它表达的逻辑为 *g*2 需要在 *g*3 前实现.*TS*3=τ3→*TS*1 描述了 *TS*3 表现为先执行τ3,然后重新回到 *TS*1.它表达了实现 *g*3 后,*g*1 也得以实现,并且此时系统开始新一轮的任务流程.

映射描述可以作为(process analysis toolkit,简称 PAT)[24]执行语言 CSP#(communicating sequential programs)的子集进行编译和仿真,从而得到相应的行为模式.容易看出,每个行为模式都表现为一个标签转移系统(labeled transition system,简称 LTS).一个标签转移系统定义为 $LTS=(S,s_0,\Sigma,\rightarrow_s)$,其中,$S$ 是有限状态集,$s_0\in S$ 是初始状态,Σ是有限动作集,$\rightarrow_s\subseteq S\times\Sigma\times S$ 是状态间的转移关系.通过行为模式和关系模式的对照可以看出,LTS 中的转移关系描述了系统任务的执行过程,而状态刻画了任务完成后的系统状态.

4.3 行为描述的生成

模式映射的原理表明,为得到系统行为模型,需要先得到系统行为的形式化描述.然而,随着系统的业务需求模型和自适应目标模型分解关系层次的增多,人工描述的工作量大且容易出错.本节引入一种行为描述的生成算法,通过遍历目标模型中的节点,生成 PAT 可执行的代码描述.

首先,将目标模型中非叶节点的无序分解关系转化为有序分解关系.这一步骤可以通过节点复制和重命名的方式实现,具体过程如下:

(1) 从无序分解的父节点开始,复制和重命名子树.复制得到子树的个数为 $n!$,n 为该无序分解得到的子节点的个数.

(2) 将无序分解的子节点进行全排列,建立相应的序关系.

(3) 将原父节点通过“或”分解关联到每一棵子树的父节点.

(4) 保留所有与叶节点相关的无序分解关系.

图 4 展示了这一转化过程的应用实例.左侧的目标模型中,*g*1 通过无序分解得到 *g*2 和 *g*3.模型表明系统可以先实现 *g*2 再实现 *g*3,或者先实现 *g*3 再实现 *g*2.右侧的模型展示了转化后的结果:*g*1 可以通过 *g*1a 或者 *g*1b 实现,如果选择 *g*1a,则 *g*2 先于 *g*3 实现;如果选择 *g*1b,则 *g*3 先于 *g*2 实现.

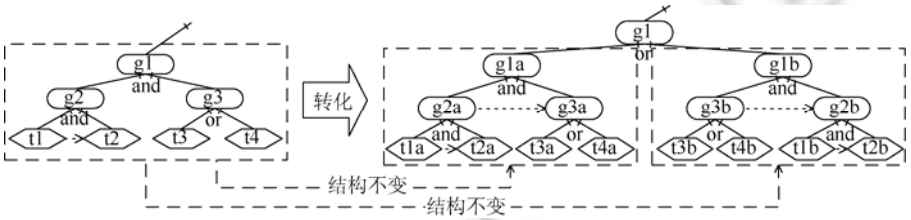


Fig.4 Transformation of relation patterns in goal models

图 4 目标模型中关系模式的转化

算法 1 展示了映射描述的生成算法,其输入为转化后的目标模型,输出为与目标模型表达逻辑一致的映射描述.其主要思想为:遍历目标模型中的节点,识别节点所属的关系模式类型,并根据不同类型输出相应的映射描述.算法中的函数 toupper( )将小写字符转化为大写字符,用于生成任务序列的名称.函数 findNext( )通过递归方式自底向上获取非空关联节点,即如果在本层找不到有序分解指定的关联节点,则需寻找父节点的关联节点,以此类推.另外,算法的第 10 行实际上输出了 $n!$条描述,每一条描述都是叶节点的一种排列组合.在用邻接表存储目标树结构时,算法在最坏情况下会遍历所有节点,其时间复杂度为 $O(n)$.读者可参考并运行算法 1 的 R 代码

分析实例(github.com/ZhuoqunYang/TreeTransToCSP),以了解算法的效果.

算法 1. Generate formal description.

Input: GoalModel;

Output: DescriptionFile.

1for $w_i \in GM$ //GM 是 goal model 中节点的集合
2 **if** relation of w_i is ordered-AND //w_i 是有序分解的父节点
3 write toupper(w_i=w_k) to DescriptionFile //w_k 是第 1 个需要实现的子节点,输出为大写的节点名称
4 record w_i //w_i 记录为已经访问
5 **else if** relation of w_i is OR //w_i 是或分解的父节点
6 write toupper(w_i=w_k[]…[]w_{k+n}) to DescriptionFile //w_k…w_{k+1} 是 w_i 的子节点
7 record w_i //w_i 记录为已经访问
8 **else if** relation of w_i is unordered-AND //w_i 是无序分解的父节点(子节点均为叶节点)
9 *tail*←*findNext*() //*tail* 为从 w_i 开始通过自底向上找到的第 1 个关联节点,若找不到则为根节点
10 *Multi* write *toupper*(w_i)=w_k→…→w_{k+n}→*toupper*(*tail*) to DescriptionFile //w_k…w_{k+1} 服从全排列顺序
11 record w_i,w_k,…,w_{k+n} //w_i 及所有子节点记录为已经访问
12 **else if** w_i is leaf & w_i is not recorded //w_i 是没有访问过的叶节点
13 *tail*←*findBrother*()
14 write *toupper*(w_i)=w_i→*toupper*(*tail*) to DescriptionFile
15 record w_i
16 **end if**
17 **end for**
18 **return** DescriptionFile //输出行为描述的文件

Lapouchnian 等人[25]也采用目标模型对自主计算系统行为的设计提供支持,目标模型中的任务选项可以映射到系统的行为选项.这一工作阐述了目标模型在系统设计阶段的重要作用,但并没有关注目标模型如何作为运行时的资产,以支持系统运行时的行为分析.对于自适应软件系统,运行时的模型管理和行为分析非常重要.本文主要关注根据非功能需求的满足程度进行自适应决策,并采用目标模型建模需求,这就决定需要建立从目标模型到行为模型的完整映射.同时,本文侧重点为目标模型对运行时行为分析的支持.从这一角度出发,目标模型需要作为运行时模型进行管理和维护,同时也应用于支持系统运行时的行为分析.

4.4 业务行为和自适应行为的融合

根据行为模型的生成过程,系统分析员需要分别对业务需求模型和自适应目标模型消除非叶节点的无序分解关系,并将模型结果作为算法 1 的输入,最终可以得到关于业务逻辑和自适应逻辑的描述.为使业务行为和自适应行为能够融合,需要在两种逻辑的描述上作一些调整.

- 将自适应逻辑描述中最后的任务序列替换为业务逻辑描述中最初的任务序列.
- 将业务逻辑描述中最后的任务序列替换为自适应逻辑描述中最初的任务序列.

图 5 展示了 MobIS 的业务需求模型中的信息获取逻辑和自适应逻辑的描述,其中,任务序列根据目标或者任务具体的内容来命名,序列编号与目标模型中的任务编号一致.通过仿真,可以得到如图 6 所示的业务行为模型和图 7 所示的自适应行为模型.读者可运行描述程序(github.com/ZhuoqunYang/PATForSAS)进行深入了解.

图 7 仅举例表达了若干条可选的行为路径,实际上存在 126(=3!+5!)条路径.为了说明优化决策过程,下文选取两条特定的自适应任务执行路径,分别对应于“在线信息获取”和“离线信息获取”逻辑.

- “在线信息获取”的自适应逻辑

GaugeBandwidth→GaugeBuffer→AnalyzeREL1→AnalyzeREL2→MakeDecision→SelectConfig→SetRegist

→SetLocate→SetForm→SetT→SetB

• “离线信息获取”的自适应逻辑

GaugeBandwidth→GaugeBuffer→AnalyzeREL1→AnalyzeREL2→MakeDecision→SelectConfig→SetRegist →SetLocate→SetSearch

这两条路径中配置任务的执行顺序与业务逻辑中相关任务的执行顺序是一致的.比如,若业务逻辑中登录任务在定位任务前执行,则在自适应逻辑中登录方式的配置需在定位方式配置前完成.

使用上述两条自适应任务路径,系统行为模型可以通过业务行为模型和自适应行为模型的叠加来得到.比如,图 5 中业务行为描述的终点需要改为自适应行为描述的起点,即→BTS1GetInfo 改为→ATSAdapt.同时,自适应行为描述的终点需要改为业务行为描述的起点,即→ATSAdapt 改为→BTS1GetInfo.通过对修改后行为描述的仿真,可将业务行为和自适应行为有机融合,得到如图 8 所示的系统行为模型.

```
//将可执行形式化描述用PAT编译并仿真
//可以生成对应的基于LTS的行为模型
//业务需求模型的可执行形式化描述
BTS1GetInfo=BTS2RecordU;
BTS2RecordU=BTS4SuccessLogin;
BTS3UAccessI=BTS6Online[]BTS7Offline;
BTS4SuccessLogin=BTT1ULogin;
BTS5PrepareInfo=BTS8LocateU;
BTS6Online=BTS10SendReq;
BTS7Offline=BTS12SearchDB;
BTS8IdentifyLocate=BTT2Locate;
BTS9IdentifyInteresBTS=BTT9CheckInteresBTS;
BTS10SendReq=BTS14ChooseForm;
BTS11Load=BTS16LoadToBuffer;
BTS12SearchDB=BTT4Search;
BTS13UFindI=BTT17Filter;
BTS14ChooseForm=BTT3ChooseForm;
BTS15SendTimely=BTT12Send;
BTS16LoadToBuffer=BTT13Load;
BTS17RecordI=BTT14Persist;
BTT1ULogin=BTT5LoginSMS[]BTT6LoginINT;
BTT2Locate=BTT7LocGPS[]BTT8LocGSM;
BTT3ChooseForm=BTT10ChooseMedia[]BTT11ChooseText;
BTT4Search=BTT15SearchVoice[]BTT16SearchText;
BTT5LoginSMS=loginBySMS->BTS5PrepareInfo;
BTT6LoginINT=loginByInternet->BTS5PrepareInfo;
BTT7LocGPS=locateByGPS->BTS9IdentifyInteresBTS;
BTT8LocGSM=locateByGSM->BTS9IdentifyInteresBTS;
BTT9CheckInteresBTS=checkInteresBTS->BTS3UAccessI;
BTT10ChooseMedia=chooseMedia->BTS15SendTimely;
BTT11ChooseText=chooseText->BTS15SendTimely;
BTT12Send=sendReq->BTS11Load;
BTT13Load=loadInfo->BTS17RecordI;
BTT14Persist=persisBTToDB->BTS1GetInfo;
BTT15SearchVoice=searchByVoice->BTS13UFindI;
BTT16SearchText=searchByText->BTS13UFindI;
BTT17Filter=filterInfo->BTS1GetInfo;
//自适应目标模型的可执行形式化描述
ATSAdapt=ATT1Monitor;
ATT1Monitor=ATT1Monitor1[]ATT1Monitor2;
ATT1Monitor1=gaugeBandwidth->gaugeMemory->ATT2Analyze;
ATT1Monitor2=gaugeMemory->gaugeBandwidth->ATT2Analyze;
ATT2Analyze=ATT21AnalyzeRel1;
ATT3Plan=ATT31MakeDecision;
ATT4Execute=ATT5Online[]ATT6Offline;
//在线信息获取业务逻辑配置共有5x4x3x2x1=120种行为路径,此处选择四种展示
ATT5Online=ATT5Online1[]ATT5Online2[]ATT5Online3[]ATT5Online120;
ATT5Online1=setLoginMethod->setlocateMethod->setForm->setT->setB->ATSAdapt;
ATT5Online2=setlocateMethod->setlocateMethod->setB->setT->setForm->ATSAdapt;
ATT5Online3=setT->setlocateMethod->setForm->setLoginMethod->setB->ATSAdapt;
ATT5Online120=setB->setT->setForm->setlocateMethod->setLoginMethod-
>ATSAdapt;
//离线信息获取业务逻辑配置共有3x2x1=6种行为路径
ATT6Offline=ATT6Offline1[]ATT6Offline2[]ATT6Offline3[]ATT6Offline4[]ATT6Offline5[]
            ATT6Offline6;
ATT6Offline1=setLoginMethod->setLocateMethod->setSearchMethod->ATSAdapt;
ATT6Offline2=setLoginMethod->setSearchMethod->setLocateMethod->ATSAdapt;
ATT6Offline3=setLocateMethod->setLoginMethod->setSearchMethod->ATSAdapt;
ATT6Offline4=setLocateMethod->setSearchMethod->setLoginMethod->ATSAdapt;
ATT6Offline5=setSearchMethod->setLoginMethod->setLocateMethod->ATSAdapt;
ATT6Offline6=setSearchMethod->setLocateMethod->setLoginMethod->ATSAdapt;
ATT21AnalyzeRel1=analyzeREL1->ATT21AnalyzeRel2;
ATT21AnalyzeRel2=analyzeREL2->ATT3Plan;
ATT31MakeDecision=makeDecision->ATT32SelectConfig;
ATT32SelectConfig=selectConfig->ATT4Execute;
```

Fig.5 Formal descriptions of system behaviors

图 5 系统行为的形式化描述

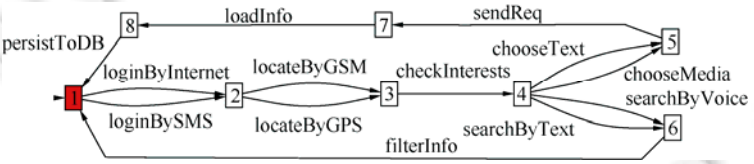


Fig.6 Business behavior model

图 6 业务行为模型

Fig.7 Adaptation behavior model

图 7 自适应行为模型

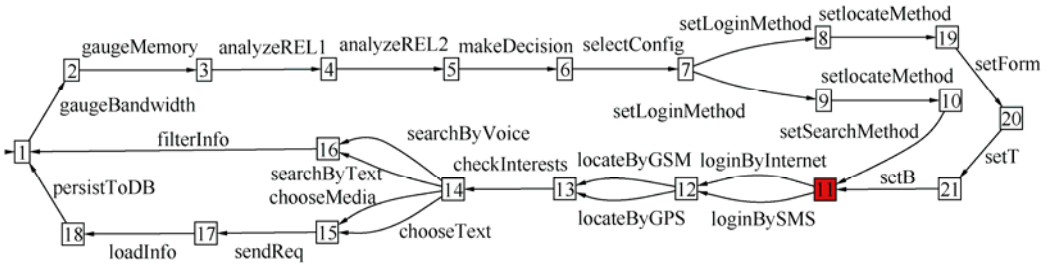


Fig.8 System behavior model

图 8 系统行为模型

5 基于行为模型的动态分析

VADEM 是一种基于量化决策的方法,它适用于针对可量化的非功能需求的优化决策,比如可靠性需求、性能需求等.可靠性需求可以通过系统任务的失效或成功概率进行衡量,而性能需求可以通过任务的时间消耗和能量消耗进行衡量.对于不同非功能需求进行优化决策,需要构建不同的决策分析模型.比如,对于可靠性的分析,需要构建概率分析模型;对于性能的分析,则需要构建带时间和资源约束的分析模型.因此,VADEM 是一个可扩展的框架,对于不同的非功能需求,只需要根据系统行为模型构建相应的量化分析模型即可.

本节以可靠性需求为例来说明 VADEM 在针对特定非功能需求进行自适应优化决策中的应用.可靠性需求是指系统成功或失败的概率有多大[26].系统执行任务具有不确定性,可将其量化为任务的失效概率.不同的失效概率分布会产生不同的系统可靠性.因此,在自适应决策问题中,系统可靠性可以通过行为路径中某一状态的可达概率来衡量.基于可靠性需求的优化决策是指通过决策分析方法在所有可选行为路径中找到最优的行为路径,使其上的概率结果最大程度地满足可靠性需求.

为实现这一优化决策过程,需要构建系统的概率分析模型.离散时间马尔可夫链(discrete time Markov chains,简称 DTMCs)是一种被广泛应用于建模基于组件构成的系统的形式化方法,在对系统可靠性的评估和预测方面有良好的应用价值[27].用 DTMCs 建模的系统满足或近似满足马尔可夫性,这一问题在关于可靠性建模的工作[28]中已给出证明和讨论.在本方法中,组件用其所执行的任务来替代.因此,构建系统的运行时分析模型就是要构建系统的离散时间马尔可夫链.

由于图 8 所示的系统行为模型缺乏关于可靠性的描述,不能直接用作运行时的可靠性分析模型.因此,构建系统 DTMC 的过程需要首先对系统行为的可靠性进行规约,然后再将可靠性规约与已得到的行为模型有机融合.

5.1 标记目标模型

由于系统任务的可靠性随上下文发生改变,系统的可靠性也随着上下文的不同而不同.系统的整体可靠性可看作是单个系统任务可靠性的综合结果,因此,需要对单个任务进行规约.引入标记目标模型(tagged goal model).

$$TGM=(GM,TAG(T_{leaf})),$$

其中,GM 是目标模型,$TAG(T_{leaf})$是叶节点任务的可靠性的标记集.对于某一任务 t,有:

$$TAG(t)=\{(c_i,FP(c_i,p))|c_i\in C\},$$

其中,c_i 是相关上下文分类集合 C 中的第 i 类,$FP(c_i,p)$是带有参数 p 的任务 t 在该类上下文中的失效概率(failure probability).对于没有参数的任务,p 为空值.任务的失效概率可以通过对系统运行时产生的日志数据统计分析得出.本方法基于上下文分类来规约可靠性,是考虑到虽然通常难以建立上下文与自适应系统之间的精确解析关系[29],但对相关上下文建模和分类是可以实现的[30].

标记目标模型定义了任务可靠性的规约方式.在 MobIS 中,对于图 2 所示的定位任务和载入任务,其可靠性

规约可以分别描述为图 9(a)和图 9(b).其中,定位为结构配置类型的任务,而载入属于参数配置类型的任务,两者规约的区别在于对概率的描述是否含有参数.为便于说明,本文将上下文 Network 分为两类:C_1(*BindwidthRate*≤100kbps)和 C_2(*BindwidthRate*>100kbps).

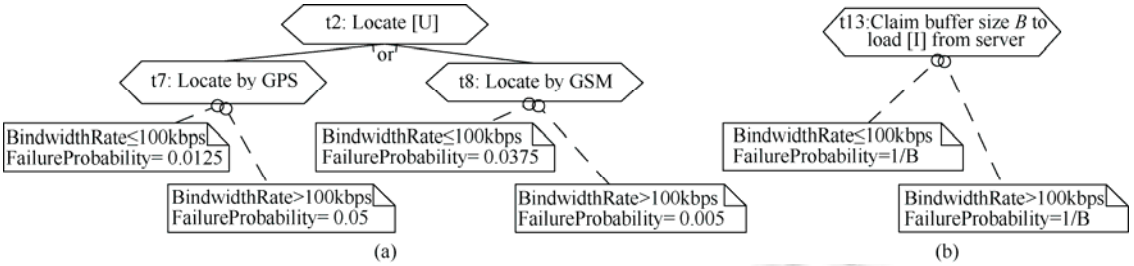


Fig.9 Reliability specifications of locating task (a) and loading task (b)

图 9 定位任务(a)和载入任务(b)的可靠性规约

5.2 可靠性分析模型

DTMCs 是具有马尔可夫性的离散随机过程,未来状态的概率分布只依赖于当前状态.它被定义为状态间具有概率性迁移的克里普克结构,状态表示系统的配置,状态间的迁移发生于离散时间并且具有一定的概率.经典的 DTMCs 具有固定的结构,包括固定的状态以及迁移概率.然而,这对于自适应系统运行时的分析是不合适的,因为系统在某一时刻会存在多种配置方式.所以,对于自适应系统而言,DTMCs 中的状态应该是可调整的,相应的迁移概率也需要随着状态结构的变化而变化.为此,VADEM 采用“可变状态”扩展 DTMCs,并将其用作自适应系统运行时的可靠性分析模型.具体而言,一个可变离散时间马尔可夫链(variable DTMC)定义为

$$DTMC^V=(S_I,S_V,s_0,P,L),$$

其中,

- S_I 是不变状态的有限集合;
- S_V 是可变状态的有限集合;
- $s_0\in S_I\cup S_V$ 是初始状态;
- $P:(S_I\cup S_V)\times(S_I\cup S_V)\to[0,1]$是迁移概率矩阵的集合.$P(s_i,s_j)$表示状态 s_i 到 s_j 的迁移概率.对于一个给定的结构,$\sum_{s'\in S}P(s,s')=1,\forall s\in S_I\cup S_V$;
- $L:(S_I\cup S_V)\to 2^{AP}$ 是标签函数.*AP* 是原子命题的集合.标签函数将每一个状态与在该状态下为真的原子命题关联在一起.

对于状态 $s\in S$,如果 $P(s,s)=1$,则称其为吸收状态(absorbing state),它只有一条到自身的环.如果一个 DTMC 含有至少一个吸收状态,则称其为吸收的 DTMC(absorbing DTMC).在最简单的可靠性分析中,一个 DTMC 需要包含两个吸收状态,分别描述系统的失效状态和系统完成所有任务后的成功状态.吸收状态可以扩展到多个,用于表示不同情况下的系统失效.由此,不同类型的任务失效可以迁移到不同类的失效状态.

算法 2. Generate variable DTMC.

Input: LTS, TGM;

Output: $DTMC^V$.

```
1    for  w_i∈S_LTS find next state w_{i+1}
2      if  |w_{i+1}|==1   //后继节点只有一个,表示没有分支行为
3        if  |Σ(w_i,w_{i+1})|==1   //迁移只有一条
4          add s_i to S_I, label Σ(w_i,w_{i+1}) to s_i, add f_{s_i} to S_I   //将 LTS 中的动作Σ赋值给 s_i 的标签, f_{s_i} 是 s_i 的失效状态
5        Else
6          add s_i to S_V, label {Σ(w_i,w_{i+1})} to s_i, add f_{s_i} to S_I   //添加可变状态,标签是动作Σ的集合
7        end if
```

8 **if** w_i is the last state of LTS
9 add s_{i+1} to S_I labeled with "success" //添加成功状态
10 **end if**
11 **else**
12 for each branch performs 算法 2 //若有分支路径,则对每一个路径分别采用算法 2
13 **end if**
14 **end for**
15 **for** $s_i \in S_I \cup S_V$
16 **if** s_i is not the last state
17 find task t performing $L(s_i)$ in TGM //在目标模型中找到相应
18 add $T(s_i, f_{s_i})$ with $P(s_i, f_{s_i})=TAG(t).FP$ //添加失效迁移概率,如果 $s_i \in S_V$,则 $P(s_i, f_{s_i})$为概率向量
19 add $T(s_i,s_{i+1})$ with $P(s_i,s_{i+1})=1-TAG(t).FP$ //添加到下一个状态的成功迁移概率
20 add $T(f_{s_i}, f_{s_i})$ with $P(f_{s_i}, f_{s_i})=1$ //对失效状态添加概率为 1 的环
21 **else**
22 add $T(s_i,s_i)$ with $P(s_i,s_i)=1$ //对成功状态添加概率为 1 的环
23 **end if**
24 **end for**
25 classify failure states with same labels //对失效状态进行分类并添加标签,可以有效减少状态数量
26 **return** S_I, S_V, s_0, P, L //返回 DTMC^V 的构成元素

构建 DTMC^V 需要将系统行为模型与任务可靠性规约相融合,具体步骤如下.

(1) 将系统行为模型中的迁移对应到 DTMC^V 中的状态,这种状态描述的是系统任务的执行状态.

(2) 将系统行为模型中的状态对应到 DTMC^V 中的迁移,这种迁移描述的是系统任务的执行结果.

(3) 加入若干失效状态和一个成功状态,分别表示系统任务的失败和成功.

(4) 为状态发出的迁移添加相应任务可靠性规约中的概率作为迁移概率(包括任务失效/成功的概率).

以上步骤可以通过算法 2 实现,其输入为系统行为的 LTS 和标记目标模型,输出为 DTMC^V 的构成元素.算法 2 中第 1 行~第 14 行用于构造状态和原子命题,第 15 行~第 24 行用于构造状态间的迁移和迁移概率.在用邻接表存储 LTS 对应的有向图时,算法的时间复杂度为 $O(n+e)$,其中,n 为 LTS 的状态数,e 为 LTS 的有向边数.由算法可知,DTMC^V 的状态由 LTS 的迁移生成,其上的命题为 LTS 的动作.此外还添加了若干吸收状态描述行为的失效.DTMC^V 状态间的迁移概率由 TGM 中系统行为的可靠性规约生成,包括失效概率 FP 以及成功概率 $1-FP$.

将上述算法应用于 MobIS 行为模型和标记目标模型,可得出如图 10 所示的 DTMC^V.

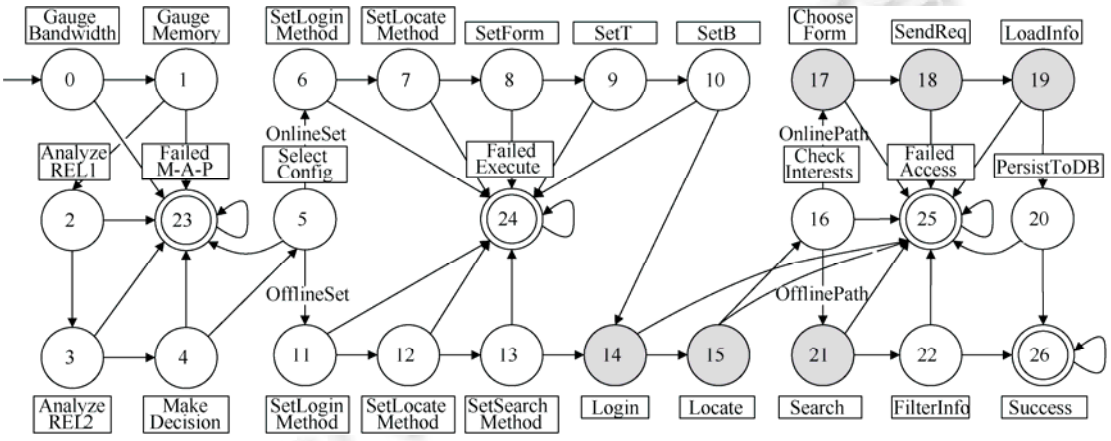


Fig.10 Reliability analysis model of MobIS

图 10 MobIS 的可靠性分析模型

图 10 中包含了 4 个吸收状态:s_{23},s_{24} 和 s_{25} 分别表示不同类别的系统失效状态;s_{26} 表示系统的成功状态.灰色的状态表示可变状态,系统在状态 s_{14},s_{15},s_{17} 和 s_{21} 中有不同的结构选项;在 s_{18} 和 s_{19} 中有不同的参数选项.因此,从这些状态出发的迁移概率均为概率向量.另外,s_5 和 s_{16} 是分支状态,从该状态出发,系统会表现出不同的行为,但在运行时,系统只能选择其中一个路径分支.可变状态和路径分支为系统自适应提供了决策选项,自适应优化决策就是要在这些可选路径中,根据可靠性需求的满足情况,确定出最优行为路径,包括路径中状态的结构和参数.

表 2 给出了图 10 中不变状态对应任务的失效概率.如果状态 s_i 的失效概率是 FP,则其成功概率是 1–FP.例如,$FP_{s_0}=0.0005$ 是指 $P(s_0,s_{23})$=0.0005,从而有 $P(s_0,s_1)$=0.9995.可变状态对应的任务受上下文的影响,因此其相应的失效概率需根据上下文的分类给出,见表 3.其中,T_0 和 B 表示系统与外部交互的配置参数.T_0 是时间间隔配置的参数,交互时间间隔越短,交互失败的概率越大,反之则越小.B 是缓存配置的参数,缓存越小,载入信息的失效概率就越大,反之则越小.

Table 2 Failure probabilities (FP) of invariable states

表 2 不变状态的失效概率

S	FP	S	FP	S	FP	S	FP	S	FP	S	FP
S_0	0.000 5	S_1	0.000 5	S_2	0.000 2	S_3	0.000 2	S_4	0.000 3	S_5	0.000 3
S_6	0.000 3	S_7	0.000 3	S_8	0.000 3	S_9	0.000 3	S_{10}	0.000 3	S_{11}	0.000 3
S_{12}	0.000 3	S_{13}	0.000 3	S_{16}	0.000 5	S_{20}	0.000 5	S_{22}	0.000 2	–	–

Table 3 Failure probabilities (FP) of variable states

表 3 可变状态的失效概率

Context=C_1 (*Bandwidth*≤100kbps)										
S	S_{14}		S_{15}		S_{17}		S_{18}	S_{19}	S_{21}	
Option	SMS	Internet	GPS	GSM	Media	Text	T_0	B	Voice	Text
FP	0.005	0.025	0.012 5	0.037 5	0.02	0.012 5	0.05/T_0	1/B	0.01	0.005
Context=C_2 (*Bandwidth*>100kbps)										
S	S_{14}		S_{15}		S_{17}		S_{18}	S_{19}	S_{21}	
Option	SMS	Internet	GPS	GSM	Media	Text	T_0	B	Voice	Text
FP	0.037 5	0.012 5	0.05	0.005	0.005	0.002 5	0.05/T_0	1/B	0.01	0.005

5.3 可靠性需求

基于可靠性需求的自适应优化决策过程需要通过验证可靠性需求的满足程度来实现.现有关于需求验证的研究工作[17,18]表明,对于用 DTMCs 建模的系统,往往用形式化语言来描述系统的可靠性需求并通过模型检验的方法来判断描述后的性质是否满足.本文采用概率计算树逻辑(probabilistic computation tree logic,简称 PCTL)[31]来描述一组可靠性需求.PCTL 由如下语法定义:

$$\Phi ::= \text{true} \mid a \mid \Phi \wedge \Phi \mid \neg\Phi \mid P_{\triangleright\lhd p}(\Psi)$$

$$\Psi ::= X\Phi \mid \Phi\, U^{\leq t}\Phi$$

其中,$p\in[0,1]$,$\bowtie\in\{<,\leq,>,\geq\}$,$t\in R_{\geq 0}$,$a$ 是指原子命题.时态运算符 X 和 U 分别称为“下一个(next)”和“直到(until)”,这与时态逻辑是相同的.由Φ产生的公式称为状态公式,它们的真值可以通过所包含的每一个状态的真值进行判断.由Ψ产生的公式称为路径公式,它们的真值需要通过对每一个执行路径进行计算来评估.根据运算符 X 和 U,还可以导出其他运算符.比如“最终(eventually)”运算符$\Diamond^{\leq t}$ 可以定义为 $\text{true}U^{\leq t}\Phi$,表示最终到公式Φ的时间不超过 t.通常,$U^{\leq\infty}$简写为 U.

PCTL 可以方便地表达用 DTMC 建模的系统的相关可靠性需求,比如关于从某一状态出发最终达到失效状态或者成功状态概率的约束,这些性质都属于系统可达性(reachability).可达性可以表示为 $P_{\bowtie\lhd p}(\Diamond\Phi)$,其语义为最终到达任何满足公式Φ的状态的概率需要满足 $\bowtie\lhd p$ 的约束.多数情况下,Φ只描述吸收状态所对应的原子命题.当其描述失效状态时,概率约束表示为≤x,其中 x 是失效概率的上界;当其描述成功状态时,概率约束表示为≥x,其中,x 是成功概率的下界.

为形式化描述需求中“尽可能被满足”的语义,PCTL 原有运算符 $\bowtie$ 需要被扩展为~$\bowtie$,表示尽可能的关系,比

如~≥表示尽可能不小于,~<表示尽可能小于.通过这种扩展,第 2 节中的“启动后接收成功的可靠性”和“登录后接收失败的可靠性”可以形式化为:

• REL1. $P_{\sim\geq 0.90}$(“GaugeBandwidth” U “Success”)描述了“从 s_0 到达 s_{26} 的概率应尽可能地不小于 0.90”.

• REL2. $P_{\sim\leq 0.05}$(“Login” U “FailedAccess”)描述了“从 s_{14} 到 s_{25} 的概率应尽可能地不大于 0.05”.

运行时的自适应决策配置需要满足可靠性需求,故 REL1 和 REL2 应作为自适应决策问题的决策目标.

6 基于动态分析的决策

VADEM 方法使用概率模型检验(probabilistic model checking)[32]的技术来实现关于可靠性需求的优化决策过程.概率模型检验是一种用于验证概率系统量化性质的技术,并可以借助于模型检验工具完成具体验证过程.优化决策的问题归结为在系统不同结构和参数组合下的需求验证问题.

自适应软件系统的在线优化决策过程为:

(1) 监测:监测已识别出的上下文变量.

(2) 分析:调用模型检验模块,验证当前系统配置是否在该上下文中满足给定的可靠性需求.如果需求得以满足,则系统保持当前配置;否则,系统需要对配置进行优化决策.

(3) 验证:调用模型检验模块,分别对行为模型中的可选路径(包括不同结构配置和不同参数配置)进行需求验证,以得到不同行为路径下相应的验证结果.

(4) 筛选:如果多个行为路径中的验证结果均满足给定阈值,则需根据用户的使用偏好,筛选出最优的行为路径,并将该路径中状态的结构和参数的配置作为系统任务的最优配置;如果所有行为路径对应的概率结果均未达到给定阈值,则选择概率最接近阈值的行为路径作为决策结果,并将该路径中状态结构和参数的配置作为系统任务的配置.

(5) 执行:将筛选得到的新结构和参数配置在系统中部署.

上述过程可以通过算法 3 描述,算法第 3 行和第 9 行中模型检验的时间复杂度与具体的模型和待验证性质的描述方式是相关的.比如在 $DTMC=(S,s_0,P,L)$中验证 PCTL 公式Φ时,模型检验的整体时间复杂度与规模$|\Phi|$之间呈线性关系,而与规模$|S|$之间是多项式关系[33].第 7 行要求配置之前再寻找行为模型中的可选配置集合,这一过程可以通过遍历 $DTMC^V$ 中的状态和有向边实现,因此时间复杂度与算法 2 相同,为 $O(n+e)$,其中,n 和 e 分别是 $DTMC^V$ 的状态规模和迁移规模.第 8 行的 for 循环表明需要对所有可配置路径分别进行模型检验,因此调用验证模块的时间复杂度为 $O(n)$,n 为可选配置路径集合的规模,其时间复杂度可通过并行求解来加以降低.第 12 行表明决策过程还需要对不同路径下的验证结果进行筛选,这等价于在规模为 n 的无序数组中寻找最大值,因此时间复杂度是 $O(n)$.

算法 3. Runtime adaptation decision-making.

Input: $DTMC^V$, $config_{old}$, req, THRESHOLD;

Output: $config_{new}$.

```
1   context←monitorContext( )   //监测函数无输入,返回上下文参量的数值
2   invoke model checking module
3   result←verifyReq(DTMC^V, config_old, req, context)   //当前配置下的验证结果
4   if (result≥THRESHOLD)
5      config_new←config_old
6   else
7      find all alternative configuration config_i in DTMC^V
8      for each config_i in DTMC^V
9         invoke model checking module
10        r_i←verifyReq(DTMC^V, config_i, req, context)   //每一个可选配置的验证结果
```

11 **end for**

12 select r_{opt} according to preferences

13 $config_{new} \leftarrow config_{opt}$ which produces r_{opt}

14 **end if**

15 **return** $config_{new}$

下面以第 5.2 节中得到的离散时间马尔可夫链和上下文 C_2(*Bandwidth*>100kbps)为例,展示如何通过验证求解和筛选来获得 MobIS 的最优配置.为实现这一目标,本节使用随机模型检验工具 PRISM[34]来验证可靠性需求.待验证的可靠性需求的形式化描述在第 5.3 节中给出.离散时间马尔可夫链的参数见表 2 和表 3.其中,参数 T_0 以 5 为步长从区间[5,25]上取值,单位为 min;参数 B 以 50 为步长从区间[50,250]上取值,单位为 MB.对于每一类上下文 Network,MobIS 共有 208(=2×2×(2×5×5+2))种候选配置.读者可参考系统的 PRISM 模块代码(github.com/ZhuoqunYang/PRISMForSAS)进行具体分析.

最优配置的选择分为两步:首先是结构配置的验证和筛选,其次是在筛选得到的特定结构配置基础上进行参数配置的验证和筛选.

(1) 最优结构配置

图 11 和图 12 分别展示了参数对(T_0,B)取中间值(15,150)时,不同的结构配置下,系统关于 REL1 和 REL2 的概率结果.系统需要根据这些概率结果筛选最优的结构配置.由于 REL1 的阈值是 0.90,在图 11 中最终概率在 0.90 之上的 13 种配置均可作为候选配置.而 REL2 的阈值是 0.05,因此,在图 12 中最终概率在 0.050 之下的 5 种配置均可作为候选配置.这两组候选配置的交集为{SMS,GSM,Offline(Text)},{Internet,GSM,Online(Media)},{Internet,GSM,Online(Text)},{Internet,GSM,Offline(Voice)}和{Internet,GSM,Offline(Text)}.根据第 2 节中的用户使用偏好 $P1$,为保证用户能够获得更新颖的信息,系统需要选择 Online 方式获取信息.根据用户偏好 $P2$,为保证用户的交互体验更加丰富,系统需要选择获取媒体信息.至此,系统通过筛选得到了一种结构配置,即{Internet,GSM,Online(Media)}.这就是自适应优化决策得到的最优结构配置.

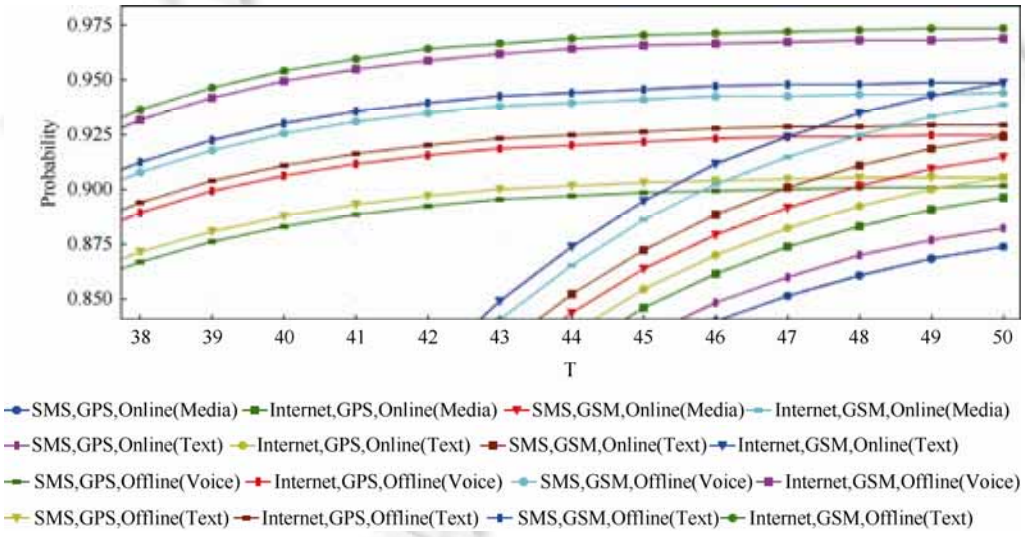


Fig.11 Probabilities about REL1 under different structural configurations

图 11 不同结构配置下关于 REL1 的概率

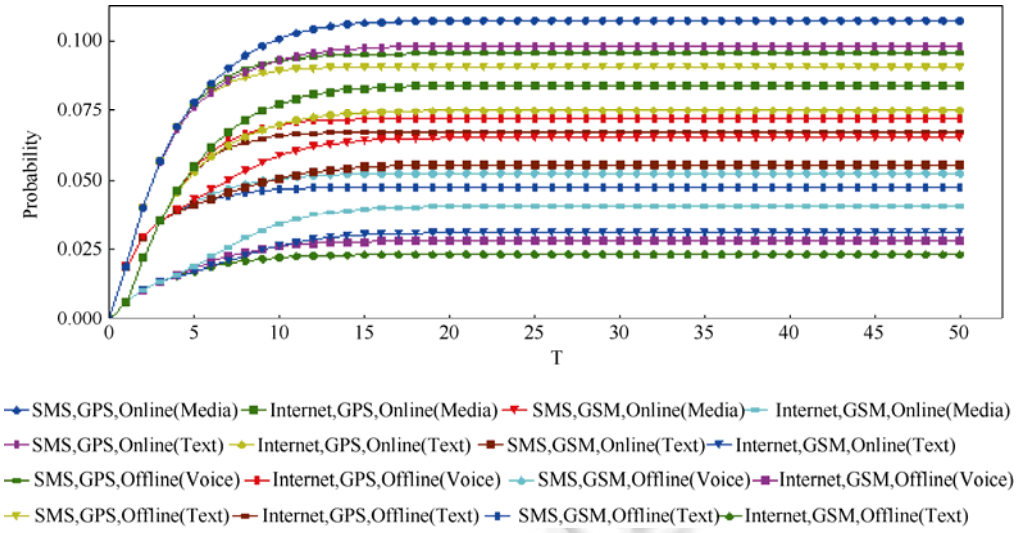


Fig.12 Probabilities about REL2 under different structural configurations

图 12 不同结构配置下关于 REL2 的概率

(2) 最优参数配置

图 13 和图 14 分别展示了在结构{Internet,GSM,Online(Media)}中不同的参数配置下,系统关于 REL1 和 REL2 的概率.由图 13 可知,这 25 种参数配置均满足 REL1.对于相同的参数 T_0,参数 B 越大,最终概率就越大;对于相同的参数 B,T_0 越大,最终概率也越大.由图 14 可知,最上方的 5 种参数配置不满足 REL2,因为它们的最终概率大于 0.050.因此,系统需要从剩余的 20 种配置中选取最优的参数配置.根据用户偏好 $P3$,为使用户尽快获得信息,系统需要尽量缩短交互时间.因此,系统首先应选择那些时间间隔较小的可选配置.根据用户偏好 $P4$,为保证系统响应速度更快,系统需要增大缓存.所以,系统还应选择那些缓存较大的可选配置.最终,经过筛选可确定参数对(5,250)作为最优参数配置.

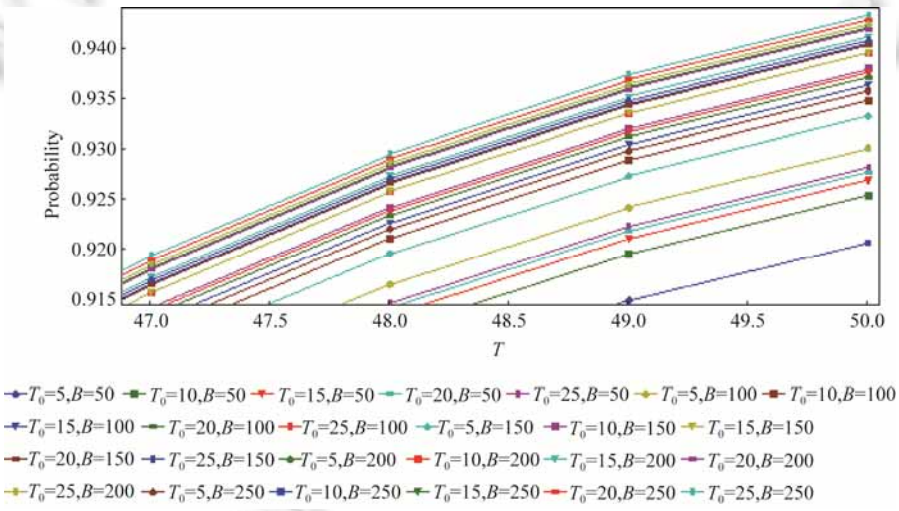


Fig.13 Probabilities about REL1 under different parametric configurations

图 13 不同参数配置下关于 REL1 的概率

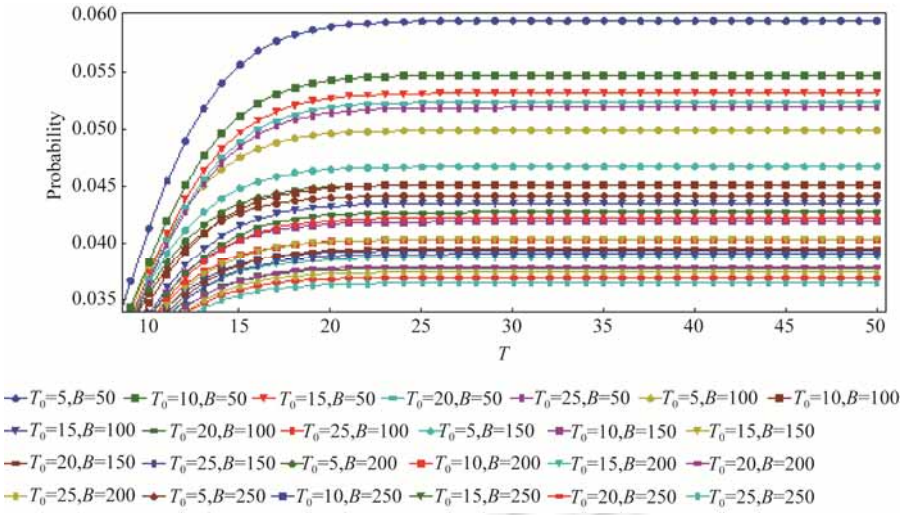


Fig.14 Probabilities about REL2 under different parametric configurations

图 14 不同参数配置下关于 REL2 的概率

综合上述最优结构配置和最优参数配置,对照图 2 中的业务需求模型,系统在上下文 C_2 中的最优决策配置为{登录任务配置=*LoginByInternet*,定位任务配置=*LocateByGSM*,选择格式任务配置=*ChooseMediaForm*,时间间隔参数=5min,缓存参数=250MB}.需要注意,当系统执行“选择格式”任务($t3$)的配置时,隐含地表示出已经选择了“在线信息获取”的业务逻辑.这一最优决策配置关于 REL1 和 REL2 的概率分别为 0.935 8 和 0.044 2.概率结果说明了 VADEM 在自适应软件系统运行时优化决策过程中的有效性.

为说明决策方法的效率,本文通过对 MobIS 的离散时间马尔可夫链模块的复制来增加系统状态数量,并对扩展后的系统行为模型进行模型检验的测试.本部分实验内容仍可以参考前述 GitHub 链接中提供的相关程序.图 15 和图 16 展示了效率的相关实验结果.随着系统复制个数的增加,系统的状态和迁移数量都急剧增加.结果表明,VADEM 在模型构建方面的时间消耗呈线性增长.在模型检验方面,对状态数量在万级以上的系统来说,通过需求验证进行优化决策的过程受到了时间制约.但是,对于状态数量在万级以下的系统来说,本方法能够在秒级的时间内求得系统可靠性的概率结果.

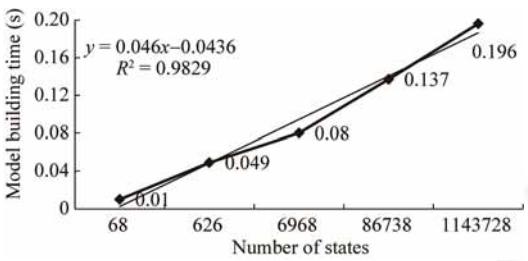


Fig.15 Time cost for model building

图 15 模型构建的时间消耗

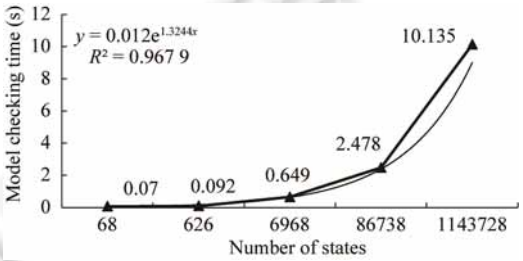


Fig.16 Time cost for model checking

图 16 模型检验的时间消耗

7 相关工作

相关工作包括 3 个方面:自适应机制建模、自适应决策实现和需求可满足性验证.

在自适应机制建模方面,Qureshi 等人[5]为提出一种需求建模语言 Adaptive RML,意在为非工程人员提供一种容易阅读和理解的需求表示方式.该语言具有与目标建模方法一致的图形化元素,并且可以形式化地映射到

Techne[35]语言.除了包括目标和任务等常规模型元素外,Qureshi 等人还考虑了上下文和资源约束.虽然建模语言的语义丰富,但是操作相对复杂.Cheng 等人[6]提出一种目标建模方法来构建动态适应系统的需求,同时考虑与需求相关的不确定性.它引入一种威胁模型来描述不确定性的原因及其对需求的影响.然而该方法局限于对需求不满足情况下解决方案的描述,未能用于运行时的自适应配置.Arcaini 等人[7]提出通过 MAPE-K 回路构建自适应机制,并对自适应行为进行形式化规约,以验证行为的有效性和正确性.但是该自适应机制采用线性时序逻辑,不能适用于求解非功能需求满足性的问题.Baresi 等人[8]提出 FLAGS 模型,它在 KAOS 模型的基础上扩展了自适应目标和自适应对策,通过将需求看作具有活性的运行时实体来触发自适应过程.当目标不被满足时,会触发自适应对策,目标模型会随即做出相应修正,以保证系统目标能得以实现.基于模糊目标的自适应机制能够准确地刻画自适应需求和在需求不满足时的系统对策,同时它还提供了良定的形式化描述.但其形式化只用于对目标和对策进行规约描述,而没有实现运行时的自适应过程.Mendonça 等人[9]基于模糊逻辑提出根据上下文和可靠性变化求解配置选项的框架,其中,可靠性需求通过模糊逻辑进行量化,可选配置通过目标模型进行识别.但是该框架会受到推理规则精度和规模的限制.

在自适应决策实现方面,现有的方法主要有两种思路,即方程计算和命题推理.基于计算的自适应决策主要是利用方程计算得出优化的配置结果,以满足系统的目标.Angelopoulos 等人[10]提出基于模型预测控制的决策方法,建立了需求和可选自适应决策之间的动态联系.该方法通过控制器对可选配置进行优化求解,以保证需求得以满足.然而该方法依赖于系统的动态数学模型,需要精确的领域知识作为基础,对于模型参数比较敏感,所以其应用条件相对严格.Esfahani 等人[11]提出 POISED 框架,该方法在不确定上下文中,通过系统的全局优化配置来提高系统的质量属性.该方法利用概率评估不确定性带来的影响,利用对不同配置在目标函数下的表现来确定最优化的解决方案.虽然 POISED 的普适性更强,但是它是一种基于架构的方法,当有多组质量属性需要被提高时,该方法的复杂性也会随之提高.基于推理的自适应决策主要是利用逻辑推理得出优化的配置结果,以满足系统需求,典型的工作为基于目标模型的推理.Wang 等人[12]提出了一种监测和诊断系统功能需求的框架,监测组件通过对需求的前置条件和后置条件的真值判断生成日志数据.诊断组件利用这些数据分析得出失效的需求,这一分析过程转化为命题满足性问题,并通过 SAT 求解器实现.为实现自适应决策,Wang 和 Mylopoulos 在后续工作[13]中给出了解决方案,其主要思想仍然是基于目标贡献关系的推理,对所有目标配置进行评估,并确定决策配置.基于目标推理的方法与系统需求是紧密相关的,因此它可以高效地解决多目标决策问题,并且考虑到目标之间的权衡.但是,由于推理需要依托于目标模型,导致其可解决的问题种类是有限的.

在自适应系统需求在线验证方面,现有的工作比较多元化,相关工作主要包括两个方面:其一是系统的形式化建模,即找到一种方法刻画系统行为;其二是性质描述,即找到一种方法描述待验证需求.Goldsby 等人[14]提出 AMOEBA-RT 方法,通过运行时监测和验证技术确保动态适应系统的需求得以满足.AMOEBA-RT 采用线性时序逻辑对程序层的自适应需求进行描述,并给出相应的模型检验类的实现代码.它能够根据人们对于程序的认识,合理地描述自适应需求,但其模型检验仅处于代码设计阶段.关于基于线性时序逻辑进行需求验证的问题,Zhao 等人[36]给出了一种解决方案,他们使用线性时态逻辑描述自适应软件系统以及验证性质,并通过标签转移系统的分析工具 LTSA 得出验证结果.Filieri 和 Tamburrelli[15]提出了一种通过模型检验来验证自适应系统可靠性需求的方法,该方法利用离散时间马尔可夫链作为系统模型并将需求表达为逻辑公式,这对本文决策方法的提出有很大的启发.Epifani 等人[16]基于需求验证的思想在迭代模型驱动开发方向上进行了基础性研究,该方法通过对开发后系统的需求验证来判断系统的行为是否满足了设计决策,并根据结果来指导后续开发过程.这是需求验证在软件开发过程中应用的范例.Calinescu 等人[17]介绍了自适应系统需要具有运行时量化验证的能力,这也成为后续自适应系统运行时需求验证方向的思想基础,该工作较为系统地介绍了验证的关注点以及验证过程中的形式化和量化思路.Ghezzi 和 Sharifloo[18]提出一种基于模型的方法来验证软件产品线中的量化非功能需求,该方法基于特征模型,通过需求验证,对软件产品线的设计方案进行评估分析.这一方法也是需求验证的典型应用之一.此外,Iftikhar 和 Weyns[19]提出 ActivFORMS 方法,以确保自适应目标的在线验证并且可以根据目标的变化对系统进行调整.其与本文的不同之处在于,该工作是将功能需求的验证与自适应机制相结合,而本文是将非功能需求的验证引入到自适应

机制中.

上述工作对本文思想的形成都有启示意义.与这些工作相比,在建模方面,VADEM 方法从目标建模的角度出发建立自适应机制,并在目标模型中融入 MAPE-K 控制机制,有简便、易行的建模原则和模型语义,且便于生成行为模型,进而用于支持运行时的非功能需求验证和系统行为分析.虽然系统行为模型可以从体系结构的基础上进行构建,但是目标模型将用户需求和系统任务紧密联系在一起.同时,以目标模型为出发点,能够使需求工程阶段的模型对设计决策产生指导价值,这也是模型驱动的意义所在.在自适应决策实现方面,VADEM 方法从目标模型出发,可以对多种需求进行描述.决策结果通过贝叶斯推理得出,是一种基于不确定性量化推理的过程.在需求的在线验证方面,VADEM 的主要特点在于构造具有可变结构的行为分析模型,并用这些可变结构表示系统的可选配置.VADEM 方法展示了需求验证过程在优化决策中的作用.

8 结束语

VADEM 方法基于需求验证实现自适应软件系统运行时的优化决策,以保证系统的决策配置能够满足特定的非功能需求.优化决策的过程是通过在系统分析模型中的不同结构和参数配置下,对相关需求进行验证来完成的.该方法也是一种模型驱动的方法,涉及到的若干模型相互关联.

本文工作的主要意义在于:

(1) 自适应目标模型提供了一种构建自适应机制的思路,有别于现有的其他基于目标模型建立自适应机制的方法,它用简单的建模元素和语义显示化描述了自适应机制,并用于构造自适应行为模型.

(2) 通过目标的形式化和目标满足过程的仿真,将目标模型转换为行为模型,使得需求模型能够对系统运行时的行为分析提供指导.这使需求模型不再单纯地成为用于捕获和描述用户期望的概念模型,而可以进一步用于系统运行时的行为描述.这也符合自适应系统需要运行时模型(models@runtime)的思想.

(3) 标记目标模型描述了同一个任务在不同上下文和不同配置中的可靠性规约,这是基于验证的自适应决策方法的关键前提.这一想法在自适应决策与重配置的研究方面还缺乏关注和讨论.

(4) 虽然自适应系统的需求工程领域中有大量关于需求验证的工作,但是并没有将需求验证作为自适应优化决策的方法,这也是本文的核心思想.需求验证过程和自适应决策过程是紧密相关的,需求验证即是为了说明系统的配置能否满足需求,而自适应决策过程就是为了确保需求的满足.

未来的研究工作中,我们主要关注两个方向.其一是在不确定性情况下对系统进行需求验证,比如非确定性存在的情况下,自适应系统应如何决策并确保决策配置满足某些特定非功能需求.另一个关注方向是需求模糊化给自适应系统带来的灵活性和弹性,尤其是能否通过量化验证的方法来衡量模糊需求对系统弹性到底带来多大益处.当然,自适应系统领域还有很多其他研究问题有待深入探索和解决.

References:

[1] Cheng BHC, Lemos R, Giese H, Inverardi P, Magee J. Software engineering for self-adaptive systems: A research roadmap. In: Software Engineering for Self-Adaptive Systems I. LNCS 5525, Springer-Verlag, 2009. 1–26. [doi: 10.1007/978-3-642-02161-9_1]

[2] Salehie M, Tahvildari L. Self-Adaptive software: Landscape and research challenge. ACM Trans. on Autonomous and Adaptive Systems, 2009,4:1–42. [doi: 10.1145/1516533.1516538]

[3] Siadat SH, Song M. Understanding requirement engineering for context-aware service-based applications. Journal of Software Engineering & Applications, 2012,5(8):536–544. [doi: 10.4236/jsea.2012.58062]

[4] Yang ZQ, Li Z, Jin Z, Chen YC. A systematic literature review of requirements modeling and analysis for self-adaptive systems. In: Proc. of the 20th Int'l Working Conf. on Requirements Engineering: Foundation for Software Quality (REFSQ 2014). 2014. 55–71. [doi: 10.1007/978-3-319-05843-6_5]

[5] Qureshi NA, Jureta IJ, Perini A. Towards a requirements modeling language for self-adaptive systems. In: Proc. of the 18th Int'l Working Conf. on Requirements Engineering: Foundation for Software Quality (REFSQ 2012). 2012. 263–279. [doi: 10.1007/978-3-642-28714-5_24]

[6] Cheng BHC, Sawyer P, Benecomo N, Whittle J. A goal-based modeling approach to develop requirements of an adaptive system with environmental uncertainty. In: Proc. of the 12th Int'l Conf. on Model Driven Engineering Languages and Systems (MODELS 2009). 2009. 468–483. [doi: 10.1007/978-3-642-04425-0_36]

[7] Arcaini P, Riccobene E, Scandurra P. Modeling and analyzing MAPE-K feedback loops for self-adaptation. In: Proc. of the 10th Int'l Symp. on Software Engineering for Adaptive and Self-Managing Systems (SEAMS 2015). 2015. 13–23. [doi: 10.1109/SEAMS.2015.10]

[8] Baresi L, Pasquale L, Spoletini P. Fuzzy goals for requirement-driven adaptation. In: Proc. of the 18th Int'l Conf. on Requirements Engineering (RE 2010). 2010. 125−134. [doi: 10.1109/RE.2010.25]

[9] Mendonça DF, Ali R, Rodrigues GN. Modelling and analysing contextual failures for dependability requirements. In: Proc. of the 9th Int'l Symp. on Software Engineering for Adaptive and Self-Managing Systems (SEAMS 2014). 2014. 55–64. [doi: 10.1145/2593929.2593947]

[10] Angelopoulos K, Papadopoulos AV, Souza, VES, Mylopoulos J. Model predictive control for software systems with CobRA. In: Proc. of the 11th Int'l Symp. on Software Engineering for Adaptive and Self-Managing Systems (SEAMS 2016). 2016. 35–46. [doi: 10.1145/2897053.2897054]

[11] Esfahani N, Kouroshfar E, Malek S. Taming uncertainty in self-adaptive software. In: Proc. of the 19th ACM SIGSOFT Symp. and the 13th European Conf. on Foundations of Software Engineering (ESEC/FSE 2011). 2011. 234–244. [doi: 10.1145/2025113.2025147]

[12] Wang Y, Mcilraith SA, Yu Y, Mylopoulos J. Monitoring and diagnosing software requirements. Automated Software Engineering, 2009,16(1):3–35. [doi: 10.1007/s10515-008-0042-8]

[13] Wang Y, Mylopoulos J. Self-Repair through reconfiguration: A requirements engineering approach. In: Proc. of the 26th IEEE/ACM Int'l Conf. on Automated Software Engineering (ASE 2009). 2009. 257–268. [doi: 10.1109/ASE.2009.66]

[14] Goldsby HJ, Cheng BHC, Zhang J. AMOEBA-RT: Run-Time verification of adaptive software. In: Models in Software Engineering. LNCS 5002, Springer-Verlag, 2008. 212–224. [doi: 10.1007/978-3-540-69073-3_23]

[15] Filieri A, Tamburrelli G. Probabilistic verification at runtime for self-adaptive systems. In: Assurances for Self-Adaptive Systems. LNCS 7740, Springer-Verlag, 2013. 30–59. [doi: 10.1007/978-3-642-36249-1_2]

[16] Epifani I, Ghezzi C, Mirandola R, Tamburrelli G. Model evolution by run-time parameter adaptation. In: Proc. of the 31st Int'l Conf. on Software Engineering (ICSE 2009). 2009. 111–121. [doi: 10.1109/ICSE.2009.5070513]

[17] Calinescu R, Ghezzi C, Kwiatkowska M, Mirandola R. Self-Adaptive software needs quantitative verification at runtime. Communications of the ACM, 2012,55(9):69–77. [doi: 10.1145/2330667.2330686]

[18] Ghezzi C, Sharifloo AM. Model-Based verification of quantitative non-functional properties for software product lines. Information & Software Technology, 2013,55(3):508–524. [doi: 10.1016/j.infsof.2012.07.017]

[19] Iftikhar MU, Weyns D. ActivFORMS: Active formal models for self-adaptation. In: Proc. of the 9th Int'l Symp. on Software Engineering for Adaptive and Self-Managing Systems (SEAMS 2014). 2014. 125–134. [doi: 10.1145/2593929.2593944]

[20] Brun Y, Serugendo GDM, Gacek C, Giese H, Kienle H, Litoiu M, Muller H, Pezze M, Shaw M. Engineering self-adaptive systems through feedback loops. In: Software Engineering for Self-Adaptive Systems I. LNCS 5525, Springer-Verlag, 2009. 48–70. [doi: 10.1007/978-3-642-02161-9_3]

[21] Yang Z, Jin Z. Modeling and specifying parametric adaptation mechanism for self-adaptive systems. In: Proc. of the 1st Asia Pacific Requirements Engineering Symp. (APRES 2014). 2014. 105–119. [doi: 10.1007/978-3-662-43610-3_9]

[22] Yu E. Towards modelling and reasoning support for early-phase requirements engineering. In: Proc. of the 3rd IEEE Int'l Symp. on Requirements Engineering (ISRE 1997). 1997. 226–235. [doi: 10.1109/ISRE.1997.566873]

[23] Mckinley PK, Sadjadi SM, Kasten EP, Cheng BHC. Composing adaptive software. Computer, 2004,37(7):56–64. [doi: 10.1109/MC.2004.48]

[24] Sun J, Liu Y, Dong JS, Pang J. PAT: Towards flexible verification under fairness. In: Proc. of the 21st Int'l Conf. on Computer Aided Verification (CAV 2009). 2009. 709–714. [doi: 10.1007/978-3-642-02658-4_59]

[25] Lapouchnian A, Liaskos S, Mylopoulos J, Yu Y. Towards requirements-driven autonomic systems design. ACM SIGSOFT Software Engineering Notes, 2005,30(4):1–7. [doi: 10.1145/1083063.1083075]

[26] Jung HW, Kim SG, Chung CS. Measuring software product quality: A survey of ISO/IEC 9126. IEEE Software, 2004,21(5):88–92. [doi: 10.1109/MS.2004.1331309]

[27] Immonen A, Niemela E. Survey of reliability and availability prediction methods from the viewpoint of software architecture. Software and Systems Modeling, 2008,7(1):49–65. [doi: 10.1007/s10270-006-0040-x]

[28] Cheung RC. A user-oriented software reliability model. IEEE Trans. on Software Engineering, 1980,6(2):118–125. [doi: 10.1109/TSE.1980.234477]

[29] Souza VES, Lapouchnian A, Mylopoulos J. System identification for adaptive software systems: A requirements engineering Perspective. In: Proc. of the Int'l Conf. on Conceptual Modeling (ER 2011). 2011. 346–361. [doi: 10.1007/978-3-642-24606-7_26]

[30] Ali R, Dalpiaz F, Giorgini P. A goal-based framework for contextual requirements modeling and analysis. Requirements Engineering, 2010,15(4):439–458. [doi: 10.1007/s00766-010-0110-z]

[31] Baier C, Katoen JP. Principles of Model Checking. Cambridge: MIT Press, 2008. 780–787.

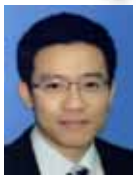
[32] Forejt V, Kwiatkowska M, Norman G, Parker D. Automated verification techniques for probabilistic systems. In: Formal Methods for Eternal Networked Software Systems. LNCS 6659, Springer-Verlag, 2010. 53–113. [doi: 10.1007/978-3-642-21455-4_3]

[33] Kwiatkowska M, Norman G, Parker D. Stochastic model checking. In: Proc. of the Int'l Conf. on Formal Methods for Performance Evaluation. 2007. 220–270. [doi: 10.1007/978-3-540-72522-0_6]

[34] Hinton A, Kwiatkowska M, Norman G, Parker D. PRISM: A tool for automatic verification of probabilistic systems. In: Proc. of the 12th Int'l Conf. on Tools and Algorithms for the Construction and Analysis of Systems. 2006. 441–444. [doi: 10.1007/11691372_29]

[35] Jureta IJ, Borgida A, Ernst NA, Mylopoulos J. Techne: Towards a new generation of requirements modeling languages with goals, preferences, and inconsistency handling. In: Proc. of the 18th IEEE Int'l Conf. on Requirements Engineering (RE 2010). 2010. 115–124. [doi: 10.1109/RE.2010.24]

[36] Zhao Y, Li Z, Shen H, Ma D. Development of global specification for dynamically adaptive software. Computing, 2013,95(9):785–816. [doi: 10.1007/s00607-013-0295-3]



杨卓群(1988－),男,山东济南人,学士,主要研究领域为需求工程,自适应软件系统,自适应算法.

金芝(1962－),女,博士,教授,博士生导师,CCF 会士,主要研究领域为需求工程,知识工程,人工智能.