软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2015,26(4):713−729 [doi: 10.13328/j.cnki.jos.004755]

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

基于反馈控制的软件适应性需求的识别与分析*

刘 春[1], 张 伟[2,3], 赵海燕[2,3], 金 芝[2,3]

[1](河南大学 计算机与信息工程学院,河南 开封 475001)

[2](高可信软件技术教育部重点实验室(北京大学),北京 100871)

[3](北京大学 信息科学与技术学院 软件研究所,北京 100871)

通讯作者: 刘春, E-mail: liuchun@henu.edu.cn

摘 要: 适应性需求的识别与分析是开发自适应软件系统的基础,它将明确软件系统在运行过程中所可能面临的环境变化以及应对这些变化系统应采取的措施.但是,当前的适应性需求识别与分析方法一方面缺乏对环境的分析导致无法系统性地识别环境变化,另一方面也缺乏有效的框架来同时考虑如何应对已知的环境变化和未知的环境变化.针对这些问题,提出了基于反馈控制的适应性需求识别与分析方法.该方法将自适应软件与其作用的环境看作为一个自适应控制系统,将适应性需求的识别与分析转化为对该控制系统所应具有的反馈回路的识别与分析.通过将环境看作软件的控制对象,它不仅突出了软件的环境,同时还可以通过确定环境感知反馈回路和需求感知反馈回路来分别应对已知的环境变化和未知的环境变化.最后,用一个实例说明所提出方法的可行性.

关键词: 自适应软件;适应性需求;需求分析;反馈控制回路

中图法分类号: TP311

中文引用格式: 刘春,张伟,赵海燕,金芝.基于反馈控制的软件适应性需求的识别与分析.软件学报,2015,26(4):713−729. http://www.jos.org.cn/1000-9825/4755.htm

英文引用格式: Liu C, Zhang W, Zhao HY, Jin Z. Software adaptation requirements identification and analysis based on feedback control. Ruan Jian Xue Bao/Journal of Software, 2015,26(4):713−729 (in Chinese). http://www.jos.org.cn/1000-9825/4755.htm

Software Adaptation Requirements Identification and Analysis Based on Feedback Control

LIU Chun[1], ZHANG Wei[2,3], ZHAO Hai-Yan[2,3], JIN Zhi[2,3]

[1](School of Computer and Information Engineering, He'nan Univiersity, Kaifeng 475001, China)

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

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

Abstract: The development of adaptive software systems is based on the adaptation requirements identification and analysis. It clarifies what changes the software will face during operation and how it needs to deal with them. However, the existing approaches pay little attention on the context analysis and lack of systematic process to identify the potential context changes. Moreover, they also lack of a framework to consider the adaptation requirements for both the known context changes and the unknown context changes. To address these issues, this paper provides an approach based on feedback control. Its basic idea is to model the adaptive software and its context as an adaptive control system, and to identify and analyze the adaptation requirements through identifying the feedback loops of such adaptive control system. By treating the context as the object to be controlled by software, it not only makes the context explicit, but also manages to define the context-aware feedback loops and the requirements-aware feedback loops to deal with the known context changes and the unknown context changes respectively. An example is used to illustrate the feasibility of the proposed approach.

Key words: adaptive software; adaptation requirement; requirements analysis; feedback loop

* 基金项目: 国家重点基础研究发展计划(973)(2015CB352201); 国家自然科学基金(91318301, 61300035); 中国博士后基金(2014M552000)

收稿时间: 2014-07-02; 修改时间: 2014-10-14; 定稿时间: 2014-11-14

随着软件越来越多地应用于各种复杂、开放、动态的环境(比如互联网、物联网等),软件系统在运行过程中面临着越来越多的不确定性.比如:网站系统会面临用户规模的不确定性,手机应用软件会面临用户类别以及使用环境的不确定性.在这种情况下,人们越来越希望软件系统具有自适应的能力,能够在运行时主动感知环境变化并调整自身的行为[1−3]:一方面,自适应的能力将增加系统自身的可靠性并降低维护成本;另一方面,它也将使得软件系统能够更加主动地改善服务质量,从而提升用户的体验.

要开发一个具有自适应能力的软件系统,我们需要在识别和分析软件系统的功能、性能等需求之外,还要识别和分析软件系统的适应性需求,以明确软件系统应该感知哪些环境变化、如何感知这些环境变化以及在变化发生时如何进行自我调整.实际中,感知潜在的环境变化有两种方式:一种是直接监控可能的环境变化;另一种是间接地监控环境变化所导致的结果——用户需求的不满足.比如,对网站系统来说,其所面临的一种潜在环境变化就是用户规模的增长.面对该环境变化,直接的感知方式就是直接监控用户规模的增长速度来预测用户规模的变化;而间接的方式就是感知系统的响应时间,因为用户规模的增长最终会导致系统响应时间的延迟.这两种感知方式代表了软件系统实现自适应性的两种不同方式.而采用不同的方式,识别和分析适应性需求的关注点也不同.其中,采取直接的方式要求在分析过程中明确潜在的环境变化,而采取间接的方式则要求在分析过程中明确用户需求潜在的不满足.

当前,虽然软件适应性需求的识别和分析已经吸引了许多研究人员的关注[3],但是大部分的工作采取的都是间接的方式.比如,Souza等人提出的AwReqs/EvoReqs框架[4,5]、Baresi等人提出的FLAGS框架[6,7]以及Whittle等人提出的RELAX方法[8]等采取的都是间接的方式.不同于现有工作,本文的基本想法是融合这两种环境感知方式来分析软件系统的适应性需求,以充分借鉴两种方式的优点.这是因为:1) 基于直接的方式,系统对环境变化的响应显然更加快速和有效,但采取这种方式需要知道具体的环境变化,而实际中却很难预知所有可能的环境变化;2) 相反地,虽然基于间接的方式系统的响应会滞后(因为变化所导致的影响已经发生)和不精确,但是却不需要知道导致需求不满足的环境变化.

基于该想法,本文提出一种基于反馈控制的方法来识别和分析软件系统的适应性需求.我们将软件及其作用的环境所构成的软件加强型系统(而不仅仅是软件本身)看作一个自适应的控制系统,突出软件作用的环境以系统地识别环境潜在的变化.在该系统中,软件作用的环境是被控对象,软件本身是控制器.该控制系统的目的就是通过软件对环境的控制作用来使用户的需求得到满足(基于问题框架方法[9,10],用户的需求存在于环境之中).为了应对环境的变化,该控制系统存在两种从环境到软件本身的反馈回路:环境感知反馈回路、需求感知反馈回路.前者感知环境的可能变化,并进而调整软件系统;由于不能识别所有的环境变化,后者感知潜在的用户需求的不满足,当需求不满足时对软件做出调整,以弥补前者的不足.在该自适应控制系统中,由于反馈回路的行为体现了软件系统感知、响应环境变化的适应性行为,因此本文的基本思路是:将软件适应性需求的识别和分析,转化为对该控制系统所应具有的反馈回路的识别和分析.

采用该方法,我们不仅可以分析软件应对已知和未知环境变化的适应性需求,还可以通过对应对环境变化反馈回路的识别,来明确软件系统为了应对环境变化所需要感知的现象、控制的现象、具体的控制措施以及整个适应性行为所涉及的实体,最后,可以通过对反馈回路的合并来发现适应性需求之间潜在的冲突.

本文第1节简要介绍控制系统的相关概念以及我们将自适应软件与其环境看作自适应控制系统的基本思路.第2节介绍本文所提出的适应性需求的识别和分析方法.第3节基于一个案例,详细说明本文所提出的方法并展示其可行性.第4节介绍相关研究工作.最后是结论.

1 自适应软件系统的控制模型

1.1 控制系统简介

在控制论中,控制是控制器按照给定的条件和目标对被控对象施加影响的过程和行为[11,12].因此,一个控制系统至少包含两部分:控制器和被控对象.其中,被控对象往往是给定的,而控制器的设计需要基于被控对象的模型.控制系统的目标是:通过控制器对被控对象的作用,使得被控对象的某些特征(即被控变量)达到或者接近

设定值.

在实际中,根据控制系统是否存在反馈回路,控制系统又可分为开环控制系统(如图 1 所示)和闭环控制系统(如图 2 所示).

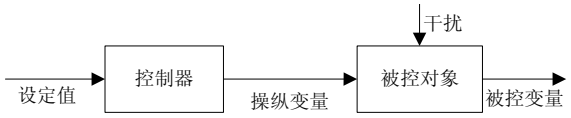


Fig.1 Model of the open loop control system

图 1 开环控制系统模型

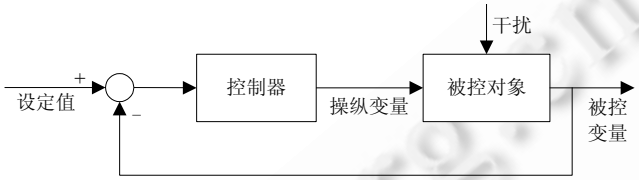


Fig.2 Model of the closed loop control system

图 2 闭环控制系统模型

如图 1 所示,开环控制系统中不存在反馈回路.在运行过程中,控制器按设定的规则对被控对象施加影响,且并不考虑控制效果.因此,该控制系统的精度往往受各种潜在的不确定性的影响,比如被控对象所受到的干扰.

不同于开环控制系统,反馈回路是闭环控制系统的关键组成部分.如图 2 所示,在系统运行过程中,控制系统的输出(也即被控变量的值)通过反馈回路不断地反馈给控制器.通过反馈回路的反馈,控制器实时计算控制效果与设定值之间的差别,并根据差别的大小来调整对被控对象的控制,从而使得控制系统在各种潜在干扰的影响下,仍能保持其输出在设定值左右.

虽然图 2 所示的闭环控制系统能够应对潜在的各种不确定性,但其要求被控对象的特征在系统运行过程中不会发生变化.这是因为控制器控制逻辑的设计是基于被控对象的模型,并且一旦控制器设计完成,其控制逻辑就不会发生变化.因此,当设计时无法获得被控对象的准确模型、或者被控对象在运行过程中发生变化时,控制系统的控制精度就会下降.为此,自适应的控制系统(如图 3 所示)被引入,以使得控制系统能够在运行过程中根据被控对象的变化在线地调整控制器的控制逻辑[12].如图 3 所示:不同于一般的闭环控制系统,该系统具有额外的一种反馈回路,从而在系统运行过程中主动地辨识被控对象的变化,并进而调整控制器的控制逻辑,使得被控对象变化之后,整个系统的输出仍然能够满足用户的要求.

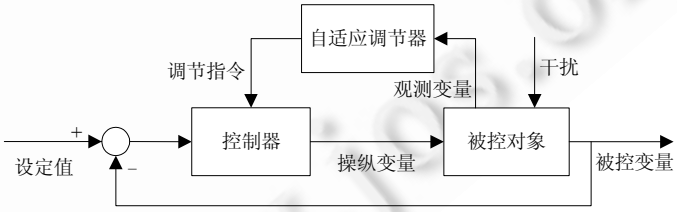


Fig.3 Model of adaptive control system

图 3 自适应控制系统模型

1.2 建模自适应软件与其环境为自适应控制系统

基于 Jackson 等人提出的问题框架方法[9,10],用户对软件的需求存在于环境(现实世界的一部分,由一组环境实体构成)之中,表达的是期望软件能够实现的某些环境现象.而构建软件的目的正是在于通过软件对环境的影响,以使用户期望的现象为真.因此,从控制论的角度出发,我们可以将软件与其所作用的环境建模为一个控制

系统.在该系统中,软件是控制器,软件所要作用的环境是被控对象,而用户所期望改变的环境现象则是控制系统的输出,也即该控制系统的被控变量.该控制系统的目标就是:通过软件对环境的控制作用,使得用户期望的那些环境现象为真.

对于传统的软件系统,我们可以将软件与其环境所构成的软件加强型系统建模为一个开环控制系统(如图4 所示).软件在根据用户需求开发完成之后,在运行过程中只是被动地响应来自环境的事件,并进而对环境施加影响,并不考虑影响的具体效果.

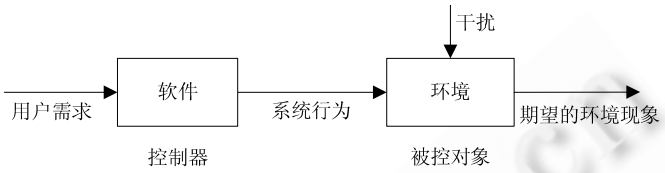


Fig.4 Sketch of the open loop control system composed by software and its context

图 4 由软件与其环境所构成的开环控制系统示意图

而对于自适应的软件系统,我们则可以将软件与其所作用的环境看作一个自适应的控制系统(如图 5 所示).不同于图 4 所示的开环控制系统,该自适应的闭环控制系统具有反馈回路来监控环境的变化,并进而调整软件以适应环境的变化.

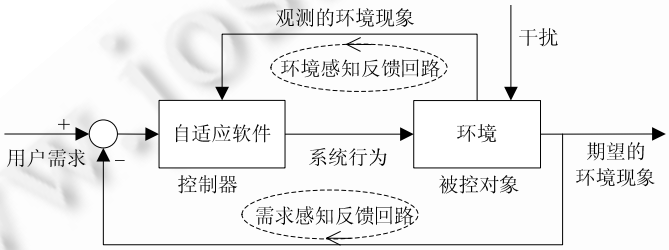


Fig.5 Sketch of the adaptive control system composed by software and its context

图 5 由软件与其环境所构成的自适应控制系统示意图

如图 5 所示,该闭环控制系统中应存在两种反馈回路:环境感知反馈回路和需求感知反馈回路.其中,环境感知反馈回路在线监测指定的环境变化,一旦某些环境变化发生时,调整软件来改变软件的行为.由于无法预知所有的环境变化,环境感知反馈回路也只能监测有限的环境变化,因此,当某些环境变化发生时,软件系统可能也无法做出响应,从而导致用户需求的不满足.在这种情况下,需求感知反馈回路可以弥补环境感知反馈回路的不足.它在软件运行过程中不断地监测表征用户需求满足与否的环境现象,在线地判断用户需求的满足情况,一旦用户需求不满足时,调整软件系统来改变软件的行为,从而通过采取一种事后的补救措施来降低或者消除未知不确定性的发生所带来的影响.

2 基于反馈控制的适应性需求的识别与分析方法

如图 5 所示,反馈回路赋予了软件应对环境变化的自适应能力,体现了自适应软件监控、分析与控制等适应性行为.因此,从控制论的角度出发,对软件适应性需求的识别与分析,可以转化为对图 5 所示的自适应控制系统应具有的反馈回路的识别与分析.为此,本文的主要策略是将自适应软件与其作用的环境建模为一个自适应控制系统,并进而通过对该控制系统所应具有的两种反馈回路的识别与分析,来实现对软件适应性需求的识别与分析.

为了表达自适应软件系统的反馈回路,本文基于问题框架方法[9,10]提出了适应性框架,为反馈回路的分析和描述提供一种模式.适应性框架将显示软件应对环境变化背后,反馈回路的具体构成和行为.基于适应性框

架,反馈回路识别和分析的主要任务就是识别环境变化,并实例化适应性框架.本节下面将首先介绍适应性框架,然后阐述适应性需求识别与建模的具体过程.

2.1 适应性框架

在问题框架方法中,Jackson 提出了问题框架这一概念以表达软件开发领域那些公认的具有明确解决方案的软件问题,比如信息显示、接受用户指令并对其他实体实施控制等问题[9].每个问题框架描述了这些软件问题中的用户需求、上下文环境一般所涉及的实体和实体的现象以及将要开发的软件与这些实体之间的关系.这些问题框架可以在软件需求的分析过程中复用,为相似软件问题的分析提供模板.

本文所提出的适应性框架如图 6 所示,该适应性框架所体现的就是软件的适应性问题.它展现了软件适应性问题所涉及的反馈回路的具体构成和行为,包括所涉及的实体以及实体之间的关系,其中,自适应调节器这一实体就是为了解决该适应性问题所要开发的解决方案.适应性框架同时提供了一种模板来描述为了应对环境变化,系统应该监控哪些现象、如何监控这些现象,以及为了应对该环境变化,应该如何控制软件系统.

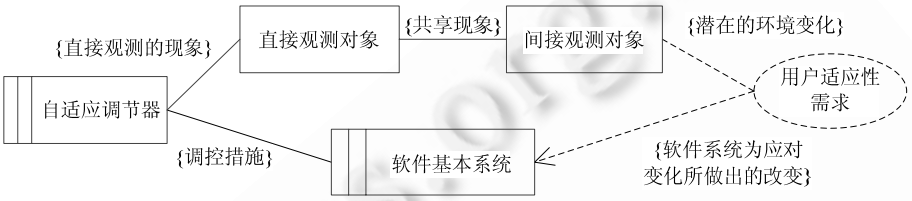


Fig.6 Diagram of the adaptive problem frame

图 6 适应性问题框架图

由于适应性需求是一种派生需求,在分析适应性需求时,软件系统的基本需求(功能与非功能性需求)已经明确,因此在该问题框架背后,我们的基本假设是:在分析适应性问题时,软件的基本系统是已知的,是适应性问题上下文环境的一部分.基于这种假设,适应性问题就是要设计自适应调节器,从而根据环境的变化调整软件的基本系统(如图 6 所示).

在该框架中,虚线椭圆代表的是从用户角度出发所描述的用户适应性需求.该用户适应性需求表达的是在软件的基本系统已经存在的情况下,当一个环境变化发生时,用户期望软件基本系统所做出的改变.该改变并不是系统应对环境变化的具体控制措施,而是系统采取具体控制措施之后的效果.因此一般来说,该用户适应性需求引用如下两方面的内容:1) 潜在的环境变化;2) 软件基本系统为应对变化所做出的改变(如图 6 中适应性需求与实体之间的虚线所示).这里,潜在的环境变化是一种具有不确定性的环境现象,是一种发生与否不受系统和用户控制的环境现象,并且其发生会对用户需求的满足产生影响.在该框架中,该潜在的环境变化又可以分为如下两种:

- 反映环境真实变迁的环境变化;
- 反映用户需求不满足的环境变化.

第 1 种环境变化就是系统作用环境中的真实环境变迁,比如用户规模的增长;而第 2 种环境变化则反映的是用户需求的不满足.将用户需求的不满足也作为一种环境变化的合理性来源于问题框架方法,基于问题框架方法,用户对软件系统的需求存在于环境之中,表达的是期望发生的环境现象.当真实发生的环境现象与用户的期望存在偏差时,用户的需求就会不满足,因此我们可以认为:用户需求不满足时的环境现象也是一种环境变化,只是该环境变化与用户期望的变化之间存在偏差.对于网站系统来说,一种用户需求是请求页面到达用户桌面的时间,当用户的需求是响应必须小于 5s,而真实的响应时间是大于 5s 时,那么这个“响应时间大于 5s”就是反映用户需求不满足的环境变化.当适应性框架中的潜在环境变化为第 1 种环境变化时,该适应性框架将描述环境感知反馈回路;而当潜在的环境变化为第 2 种环境变化时,该适应性框架则描述的是需求感知反馈回路.因此,我们可以利用该适应性框架来描述两种不同的反馈回路.

为了满足用户适应性需求,我们需要建立自适应调节器来感知相应的环境变化,并调节基本系统以使其做出期望的改变.由于环境变化一般体现为某个实体的变化,而实际中直接监测某些实体的变化可能比较困难,或者这些实体的变化可能是不可观测的,这时往往会以间接的方式通过监测一些中间实体,并利用这些中间实体与被观测实体之间的共享现象来实现对需要观测的环境变化的观测(如图6所示).因此,从用户适应性需求引用的环境变化到适应性调节器的感知路径上往往会涉及多个环境实体.这些实体与自适应调节器以及软件的基本系统一起构成了满足用户适应性需求的反馈回路.

在适应性框架中,调控措施与用户期望系统做出的改变之间的区别在于:用户期望系统做出的改变表达的是一种效果,而调控措施则是达到该效果的具体策略.一般来说,自适应调节器为调节软件基本系统所采取的具体调控措施有如下3种:

1) 改变基本系统的某个参数值;
2) 在基本系统完成某个目标的多个可选任务之间选择另外一个任务;
3) 松弛某个目标的要求来满足优先级更高的目标.

2.2 适应性需求识别与分析的具体过程

通过将软件适应性需求的识别与分析转化为对反馈回路的识别与分析,本文所提出的适应性需求识别与分析方法的具体过程如图7所示.

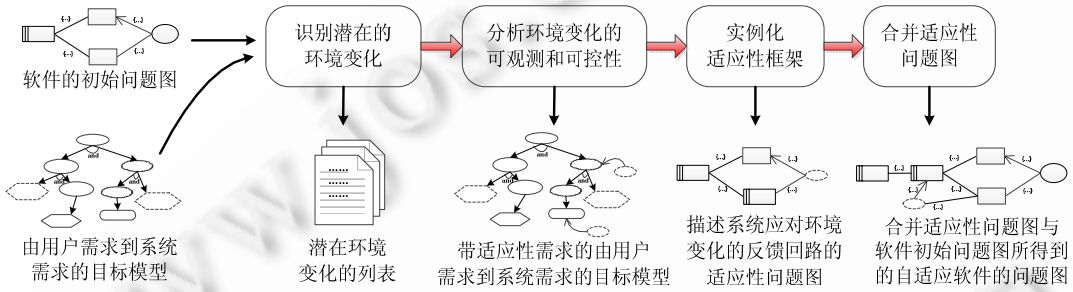


Fig.7 Proposed process to identify and analyze the adaptation requirements

图7 本文所提适应性需求识别与分析方法的具体过程

该过程包含了4个活动,主要应用3种分析工具:问题图、目标模型图和环境变化列表.该过程的基本假设是系统的基本需求是已知的,并且要求输入两种软件制品:软件的初始问题图和由用户需求到系统需求的目标模型图.其中,初始问题图的意义是帮助分析人员明确将要开发的软件的作用环境,包括环境中各个实体及其关系;而目标模型图的意义是明确用户的需求以及系统的需求(系统应该完成的任务和遵循的约束).基于这两种输入,该过程首先识别潜在的环境变化,包括反映环境变迁的环境变化和用户需求的不满足;然后分析环境变化的可观测性和可控性;其次,针对每个环境变化,实例化适应性框架,得到应对每个环境变化的适应性问题图;最后,合并各个适应性问题图与软件的初始问题图,得到自适应软件系统的问题图.下面详细介绍4个活动的具体任务.

活动1:识别潜在的环境变化.

该活动的主要任务是识别两种环境变化:一种是反映环境真实变迁的环境变化,另一种是反映用户需求不满足的环境变化.从软件的初始问题图所界定的环境出发,识别第1环境变化的具体方式是:首先,头脑风暴环境中每个实体的可能会发生变化的属性及其具体变化,该属性的变化即是潜在的环境变化;然后,分析每个环境变化对系统需求和用户需求的影响.而识别第2种环境变化的方式是:直接从目标模型图的各个用户需求出发,分析潜在的可能会不满足的需求(体现为目标模型中目标与软目标的不满足),该需求的不满足即是第2种环境变化,然后分析其具体影响.

活动2:分析环境变化的可观测性和可控性.

- 可观测性反映的是所识别的环境变化在实际中是否可以通过技术手段监测到,比如:网站用户规模增加这一环境变化是可观测的;而系统突然断电和由于地震等所导致的物理破坏则是不可观测的;
- 可控性则反映的是软件系统是否可以采取一些措施来降低或者消除环境变化所带来的影响,比如:网站用户规模增加这一环境变化是可控的;而地震等所导致的物理破坏则是不可控的.

在实际中,软件系统只能应对那些既可观测又可控的环境变化.基于对环境变化的这两种属性的区分,该活动的具体任务包括:1) 分析环境变化的可观测性和可控性;2) 结合环境变化的影响和其可观测性和可控性,确定需要考虑的环境变化;3) 对于需要考虑的环境变化,确定相应的用户适应性需求,即,描述当其发生时用户期望系统所做出的改变;4) 将确定的用户适应性需求标注在由用户需求到系统需求的目标模型中,以建立用户适应性需求与系统基本需求之间的联系.

活动 3:实例化适应性框架.

该活动的任务是针对每一个适应性需求,通过实例化适应性框架来描述每个适应性需求背后反馈回路的结构和行为,最终得到反映反馈回路的适应性问题图以及相应的文字描述.在该活动之后,活动 2 所建立的目标模型会进一步更新,因为我们在分析具体的调控措施过程中可能会发现一些新的系统需求(系统的任务或者约束),比如增加实现某个目标的可选任务来实现自适应.

活动 4:合并适应性问题图.

该活动的任务包括:1) 合并活动 3 所得到的各个适应性问题图以及软件的初始问题图,最终得到自适应软件系统的问题图;2) 在合并适应性问题图的过程中,分析和解决各个适应性需求之间潜在的冲突,适应性需求之间潜在的冲突往往体现在对相同实体的控制.所得到的自适应软件系统的问题图将反映自适应软件系统所具有的各个反馈回路.

本文下面将以一个具体案例来详细说明上述过程中每个活动的目的、任务以及每个活动所建立的制品.

3 案例研究

本节将以一个提供新闻服务的网站系统为例,来说明本文所提出方法的具体过程.

3.1 案例介绍

在适应性与自管理系统的软件工程研究领域,研究人员在相关网站上发布了一些例子来供大家研究(http://seams.self-adapt.org/wiki/Examplars).本文将选择 ZNN.com 作为案例来说明本文所提出的方法.ZNN.com 是一个简化的、向用户提供多媒体新闻服务的网站系统,它曾被 Cheng 等人用来说明其基于架构的自适应软件系统开发框架 Rainbow[13],也曾被用来比较两种实现自适应软件系统的方法[14]:基于架构的方法 Rainbow 和基于需求的方法 AwReqs/EvoReqs.

通常,网站系统可为用户提供多种服务,比如新闻、邮件、即时通信等.在本案例中,我们仅考虑该网站的新闻服务.为了满足用户的新闻服务需求,网站在实际中可以多种方式(比如文本、图片、视频等)来展现新闻内容.同时,在提供新闻内容的过程中,整个网站系统还需要满足用户对系统的性能、系统的成本等方面的需求.

输入 1:软件的初始问题图.

基于问题框架方法,该新闻网站的问题图如图 8 所示(具体建立问题图的过程可参考文献[9]).其中,用户需求(由椭圆表示)是在客户端上点击操作之后能够浏览到相关的新闻内容.系统作用的环境包括客户端、互联网以及用户(矩形代表各种实体).而 ZNN.com 是将要开发的系统(由带双竖杠的矩形表示).图中各个实体之间的连接线代表共享现象,而用户需求到实体之间的虚连接线代表需求对实体现象的引用.

输入 2:由用户需求到系统需求的目标模型图.

如图 8 所示,问题图仅能展示整个软件问题的框架,不能描述软件系统本身应该完成的任务和应遵循的约束.因此,本文提出建立由用户需求到系统需求的目标模型.其目的在于:借鉴目标模型这一结构化的方式来系统地由用户需求推理软件系统功能与非功能的基本需求.同时,在推理的过程中明确我们对环境所做出的假设,以进一步帮助分析人员充分认识环境的特征.

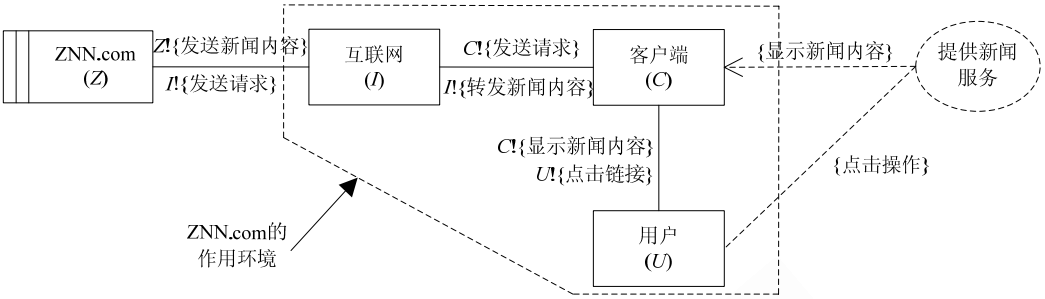


Fig.8 Initial problem diagram of the ZNN.com

图 8 ZNN.com 的初始问题图

为了建立由用户需求到系统需求的目标模型,本文应用了面向目标的需求工程(GORE)的“目标”、“软目标”、“任务”、“质量约束”、“领域假设”这一组概念[15],各个概念的解释如下:

目标 (⬭)	目标通常用来表达用户对系统的功能需求,也即期望软件系统做什么.在由用户需求到系统需求的目标模型中,目标更强调的是表达用户期望实现的现象.这一现象可以是环境某些实体的现象,也可以是将要建立的软件系统的现象
软目标 (⬭)	软目标通常用来表达用户对系统的非功能性的需求,它们往往都是主观的、没有明确的满足标准.在由用户需求到系统需求的目标模型中,软目标强调的是表达对期望实现的某些现象的主观性约束,比如服务的响应时间要快、成本要低等
任务 (⬡)	任务是软件系统将要完成的具体活动,这些活动体现了软件系统实现其目标的具体措施
质量约束 (▭)	质量约束往往表达的也是对系统质量特性的关注.但是,不同于软目标,质量约束一般会明确地描述其关注的质量特性的值或者范围
领域假设 (⬭)	领域假设表达的是人们在建立软件系统过程中对环境所作的假设.它可以描述环境中某个实体可以完成的功能,也可以描述实体的非功能性的特征

基于上述概念,针对 ZNN.com 所建立的目标模型如图 9 所示.

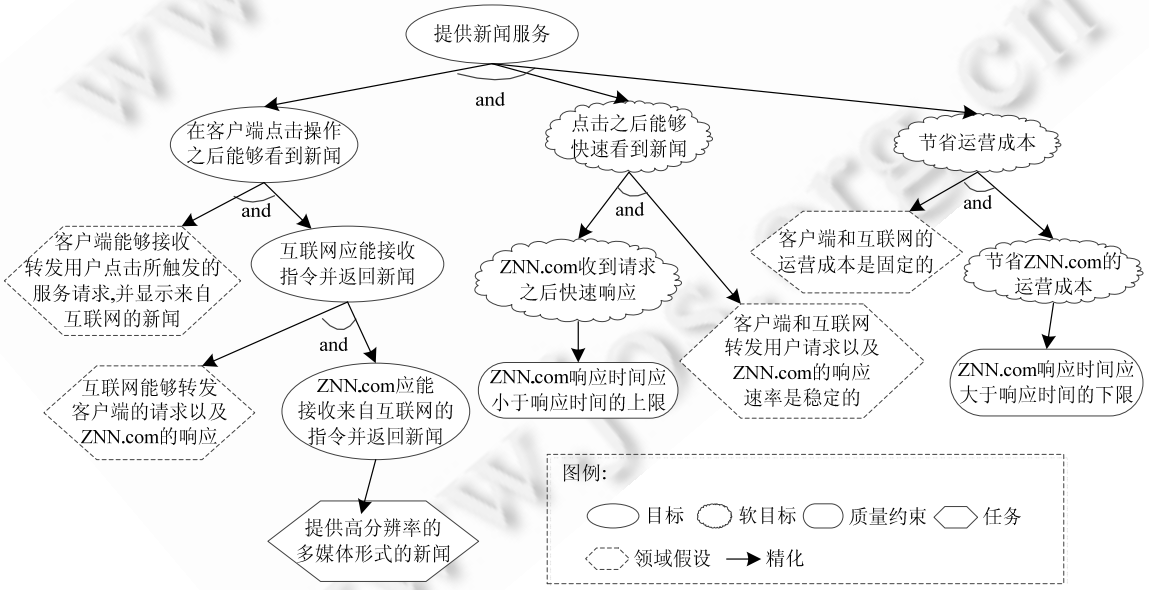


Fig.9 ZNN.com’s goal model reasoning from users’ requirements to system requirements

图 9 ZNN.com 的由用户需求到系统需求的目标模型图

对于 ZNN.com 这一新闻服务网站,用户的需求可以具体化为一个目标和两个软目标:“在客户端点击操作

之后能看到新闻”、“点击之后能够快速看到新闻”、“节省运营成本”.基于对环境中每个实体性质的分析,由这些目标所推理出的系统需求包括一个任务和两个质量约束:“提供高分辨率的多媒体形式的新闻”、“ZNN.com 响应时间应小于响应时间的上限”、“ZNN.com 响应时间应大于响应时间的下限”.

3.2 案例分析

在实际中,我们按照图 8 所示的软件问题结构、图 9 所示的软件系统需求开发出相应的软件,以及在假设一定用户规模的基础上选定相应数量的服务器来提供高分辨率多媒体新闻内容,ZNN.com 就能够为用户提供符合要求的新闻服务.但在 ZNN.com 的运行过程中,其作用环境由于其开放性不可避免地会面临许多干扰,比如用户规模会超过设计预期、客户端和互联网会过载以及极端情况的互联网中断网络服务、系统断电等.因此,我们需要识别 ZNN.com 的适应性需求,以在开发时赋予 ZNN.com 一定的适应性能力,从而保证用户需求持续得到满足.以该新闻网站系统为案例,本文的适应性需求分析过程如下.

1) 识别潜在的环境变化

分别从图 8 所示的软件的初始问题图和图 9 所示的目标模型图出发,ZNN.com 潜在的两种环境变化的识别过程分别见表 1 和表 2.

Table 1 The first kind of context changes of ZNN.com

表 1 ZNN.com 的第 1 种环境变化的分析

环境实体	可能会变化的实体属性	环境的可能变化(第 1 种环境变化)	影响分析
用户	用户规模	*CC*1:单位时间内平均访问量的增加	当用户规模超过设计预期时,系统的质量约束“ZNN.com 响应时间不超过响应时间上限”就会不成立,从而影响用户目标“点击之后快速看到新闻”的满足.而当用户规模降低时,可能会使得系统质量约束“ZNN.com 响应时间应大于响应时间的下限”不再成立,造成系统运营成本的浪费
	访问意图	*CC*2:单位时间内平均访问量的减少	
互联网	网络状况	*CC*3:网络拥堵	网络拥堵会延迟了用户请求到达 ZNN.com 的时间,同时也延迟了 ZNN.com 的响应到达客户端的时间,从而影响了用户目标“点击之后快速看到新闻”的满足
		*CC*4:网络中断	网络中断直接导致用户请求无法到达 ZNN.com,也使得 ZNN.com 的响应无法到达客户端,从而影响了用户目标“点击之后快速看到新闻”的满足
客户端	负载	*CC*5:客户端过载	当客户端负载过载时,可能会导致客户端性能下载甚至宕机,影响客户端接收并展示来自 ZNN.com 的新闻内容的效率,影响用户目标“点击之后快速看到新闻”的满足

Table 2 The second kind of context changes of ZNN.com

表 2 ZNN.com 的第 2 种环境变化的分析

用户需求	用户需求潜在的不满足(第 2 种环境变化)	影响分析
点击之后能够快速看到新闻	*CC*6:点击之后很长时间系统没有响应,看不到新闻	影响用户的心情,以及对网站的感受,降低用户对网站的接受度,从而降低网站的声誉

如图 8 中的软件的初始问题图所示,ZNN.com 的作用环境包含了 3 种环境实体:用户、互联网、客户端.经过对这 3 个环境实体的分析,我们识别了 5 种潜在的第 1 种环境变化.

对于用户,我们考虑了其两个可能会发生变化的属性:用户规模和访问意图.经过分析,这两个属性的变化会带来如下两个环境变化:“*CC*1:单位时间内平均访问量的增加”和“*CC*2:单位时间内平均访问量的减少”(CC 是 context change 的缩写).当用户规模增加时,会导致单位时间内系统平均访问量的增加,并且当规模超过设计预期时,系统的质量约束“ZNN.com 的响应时间应小于响应时间的上限”就会不成立,从而影响用户目标“点击之后能够快速看到新闻”的满足.而当用户规模降低时,可能会使得系统质量约束“ZNN.com 的响应时间应大于响应时间的下限”不再成立,造成系统运营成本的浪费.同时,当有恶意的用户访问时,他可能会在短时间内发送大量服务请求,也会造成 ZNN.com 单位时间内访问量的增加,从而影响系统的响应时间.

对于环境实体“互联网”,我们考虑其可能变化的属性“网络状况”.实际中,该属性的变化可能导致两个环境

变化:“*CC*3:网络拥堵”和“*CC*4:网络中断”.网络拥堵会延迟用户请求到达 ZNN.com 的时间,同时也延迟 ZNN.com 的响应到达客户端的时间,从而影响了用户软件目标“点击之后快速看到新闻”的满足.而网络中断直接导致用户请求无法到达 ZNN.com,也使得 ZNN.com 的响应无法到达客户端,从而也影响了用户软目标“点击之后快速看到新闻”的满足.

而对于环境实体“客户端”,我们考虑了其可能变化的属性“负载”.实际中,该属性的变化可能会导致一个环境变化:客户端过载.因为在实际中,客户端负责接收用户的点击操作,并展示 ZNN.com 所返回的新闻内容,当客户端负载过载时,可能会导致客户端性能下载甚至宕机,影响客户端接收用户的点击操作以及展示来自 ZNN.com 的新闻的效率,从而影响用户目标“点击之后快速看到新闻”的满足.

基于图 9 所示的目标模型,我们在考虑应对用户需求不满足的适应性需求时,选择了软目标“点击之后能够快速看到新闻”(如表 2 所示),这是因为,该软目标在两个软目标中具有较高的优先级.并且,在实际中除了网络和客户端的性能、用户规模外,还有很多其他不确定性因素会影响该目标的满足.往往即使在网络和客户端状况良好、用户规模稳定的情况下,一些其他的因素(比如运行在服务器上的其他程序的竞争)也会影响 ZNN.com 的性能,使其响应时间大于用户容忍的响应时间的上限,从而使用户在操作之后不能很快看到新闻.

2) 分析环境变化的可观测性和可控性

基于上一步对潜在环境变化的识别,对这些环境变化可观测性和可控性的分析见表 3.具体的分析过程解释如下.

环境变化“*CC*1:单位时间内平均访问量的增加”和“*CC*2:单位时间内平均访问量的减少”是可以通过统计单位时间内 ZNN.com 所接收到的访问请求个数来监测的,因此这两个环境变化是可观测的.对于这两个环境变化的影响(即,造成系统响应时间的增加或者降低),一种常见的应对措施是:在设计时保留一定的性能余量,使得系统可以在运行时增减运行服务器来适应单位时间内平均访问量的变化.但是,实际的平均访问量的范围在设计时是未知的,设计时如果设定过多的性能余量也会增加成本,因此,通过增减服务器来适应平均访问量的变化,其效果是有限的.为此,在该措施之外,我们可以补充一种措施:更改新闻内容的分辨率(比如:降低高分辨率多媒体新闻的分辨率为低分辨率的多媒体新闻、或者直接提供文本形式的新闻).其中,前一种措施属于对系统参数(运行服务器的个数)的修改,而后一种措施是在满足目标的多个可选任务之间选择另外一个任务来满足目标(为了该措施,我们增加了两个新任务“提供低分辨率的多媒体形式的新闻”和“提供文本形式的新闻”来满足目标“ZNN.com 应能接收互联网的指令并返回新闻”,如图 10 所示).而这两种措施的存在,也说明了环境变化“单位时间内平均访问量的增加”和“单位时间内平均访问量的减少”是可控的.综合这两个环境变化所造成的影响及其可观测性和可控性,我们认为,这两个环境变化是需要考虑的.当面对这两个变化时,我们期望系统通过这两项措施相应改变自身处理请求的能力,因此,我们确定了两个用户适应需求“*AR*1:当单位时间内平均访问量增加时,增强系统对请求的处理能力”和“*AR*2:当单位时间内平均访问量减少时,降低系统处理请求的能力”.

对于“*CC*3:网络拥堵”和“*CC*4:网络中断”这两个环境变化,虽然我们可以通过 ping 等方式来对其进行监测,但是系统却无法采取有效措施来消除或者降低它们的影响.因此,这两个变化是可观测但不可控的,因此,这两个环境变化在适应性需求分析过程中是不需要考虑的.对于环境变化“*CC*5:客户端过载”,由于我们往往无法采取措施来监测其发生,也无法采取有效的措施来消除或者降低其影响,因此该环境变化也是不可观测、不可控的.所以,该环境变化在适应性需求分析过程中也是不需要考虑的.

而对于环境变化“*CC*6:点击之后很长时间系统没有响应,看不到新闻”,虽然该环境变化实际中可能会由很多因素所导致,比如网络的状况、客户端的负载等,但是其中主要的因素还是软件系统本身的响应时间,因此,对该环境变化的观测可以转化为对系统本身响应时间的观测.而为了应对该变化,系统可以采取增加服务器等措施,因此可以认为该环境变化是可观测和可控的.又由于该环境变化的影响是不可忽略的,因此该环境变化是需要考虑的.为此,我们确定了应对该环境变化的用户适应性需求“*AR*3:当系统响应时间超过上限时,增强系统处理请求的能力”.

Table 3 The observability and controllability analysis on context changes

表 3 环境变化的可观测性和可控性分析

潜在的环境变化	影响分析	可观测性、可控性分析	是否需要考虑	适应性需求
*CC*1:单位时间内平均访问量的增加	当用户规模超过设计预期时,系统的质量约束“响应时间不超过响应时间上限”就会不成立,从而影响用户目标“点击之后快速看到新闻”的满足	可观测、可控	是	*AR*1:当单位时间内平均访问量增加时,增强系统对请求的处理能力
*CC*2:单位时间内平均访问量的减少	而当用户规模降低时,可能会使得系统质量约束“系统响应时间应大于响应时间的下限”不再成立,造成系统运营成本的浪费	可观测、可控	是	*AR*2:当单位时间内平均访问量减少时,降低系统对请求的处理能力
*CC*3:网络拥堵	网络拥堵会延迟用户请求到达 ZNN.com 的时间,同时也延迟了 ZNN.com 的响应到达客户端的时间,从而影响了用户目标“点击之后快速看到新闻”的满足	可观测、不可控	否	
*CC*4:网络中断	网络中断直接导致用户请求无法到达 ZNN.com,也使得 ZNN.com 的响应无法到达客户端,从而影响了用户目标“点击之后快速看到新闻”的满足	可观测、但不可控	否	
*CC*5:客户端过载	当客户端负载过载时,可能会导致客户端性能下载甚至宕机,影响客户端接收并展示来自 ZNN.com 的新闻内容的效率,影响用户目标“点击之后快速看到新闻”的满足	不可观测、也不可控	否	
*CC*6:点击之后很长时间系统没有响应,看不到新闻	影响用户的心情,以及对网站的感受,降低用户对网站的接受度,从而降低网站的声誉	可观测、可控	是	*AR*3:当系统响应时间超过上限时,增强系统处理请求的能力

图 10 展示了环境变化的可观测和可控性分析后,更新目标模型所得到的标注用户适应性需求的目标模型.

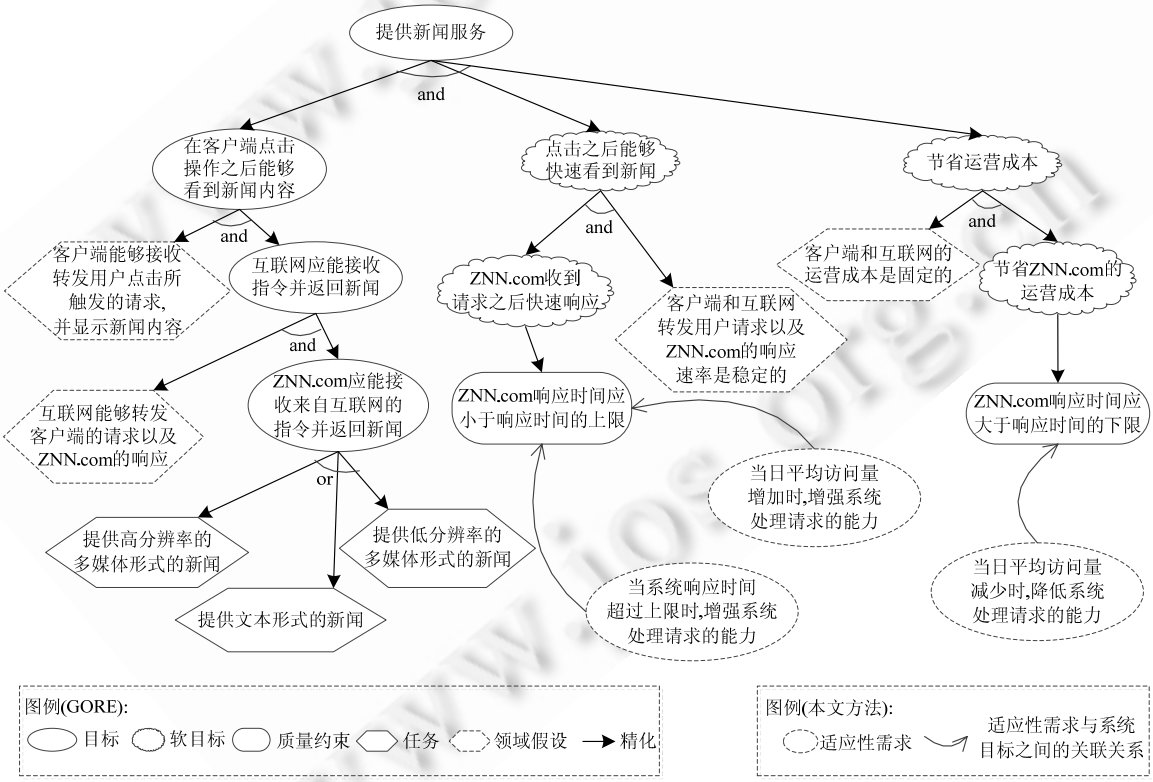


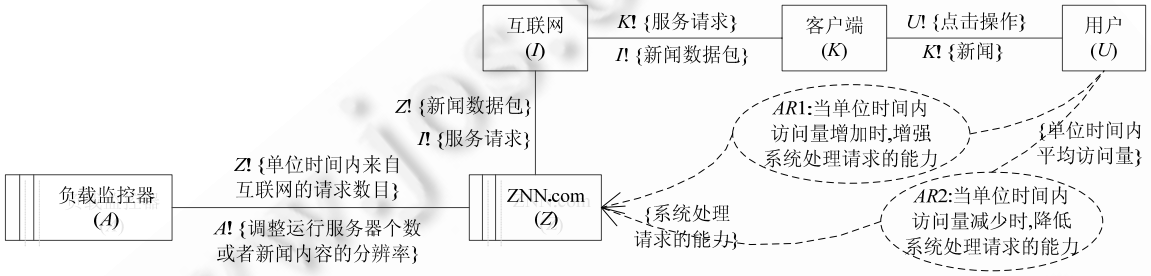
Fig.10 ZNN.com’s updated goal model marked by users’ adaptation requirements

图 10 标注用户适应性需求的 ZNN.com 由用户需求到系统需求的目标模型

它显示了所确定的用户适应性需求以及它们所关联的系统目标.这种关联关系是一种“应对”关系,它反映的是当某些需求因为环境的变化将不满足或者已经不满足时,期望系统所做出的改变.而用户适应性需求表达的正是这些期望的改变.

3) 实例化适应性框架

图 11 显示了适应性需求 *AR*1 和适应性需求 *AR*2 背后的适应性问题图,该适应性问题涉及用户、客户端、互联网、ZNN.com 以及负载监控器.而该适应性问题就是要设计开发其中的负载监控器,来在线地监测用户规模变化所导致的系统单位时间内平均访问量的变化,并根据访问量的变化来动态地调节 ZNN.com,从而实现增强或者降低 ZNN.com 对请求的处理能力.由于实际中用户规模的增加或者减少,一个直接体现就是 ZNN.com 所接收到的请求数量的变化,因此对于单位时间内平均访问量的监测,就可以转化为对单位时间内 ZNN.com 所接收到服务请求数目的监测.所以在该适应性问题中,直接监控的对象就是 ZNN.com,直接监控的现象(或者变量)就是单位时间内 ZNN.com 接收来自互联网的请求数目.而反馈回路的真正构成也因此仅涉及负载监控器和 ZNN.com.由于我们在前面分析“单位时间内平均访问量的增加或者减少”这一环境变化的可控性时已明确,可以通过增减服务器和更改新闻内容的分辨率来应对该环境变化,因此,为了实现在运行时增强或者降低系统处理请求的能力,我们采取的调控措施是调整运行服务器的个数和新闻内容的分辨率.实施该调控措施的具体调控规则如图 11 所示.



*AR*1:当单位时间内平均访问量增加时,增强系统处理请求的能力
直接监控对象:ZNN.com
直接监控现象:单位时间内来自互联网的请求数目
调控措施:调整运行服务器个数、新闻内容的分辨率
启动条件:单位时间内平均访问量>*N*(单服务器情况下提供高分辨率多媒体新闻内容时,系统响应时间小于响应时间上限的最大访问量)
调控规则:1) 当单位时间内平均访问量>*N*时,如果有备用的服务器,则启动一台服务器;否则,转步骤4)
2) 当单位时间内平均访问量>2*N*时,如果有备用的服务器,则启动一台服务器;否则,转步骤4)
3) 依步骤1)和步骤2)类推,每当单位时间内的平均访问量增加*N*,如果有备用的服务器,则启动一台服务器;否则,转步骤4)
4) 如果当前的新闻是高分辨率多媒体形式的,则设定新闻形式为低分辨率的多媒体形式
5) 如果当前的新闻是低分辨率的多媒体形式,则设定新闻形式为文本
6) 如果当前的新闻已经是文本形式的,则给出报警提示当前可用的资源已采用完毕

*AR*2:当单位时间内平均访问量减少时,降低系统处理请求的能力
直接监控对象:ZNN.com
直接监控现象:单位时间内来自互联网的请求数目
调控措施:调整运行服务器个数、新闻内容的分辨率
启动条件:单位时间内平均访问量减少幅度>*N*
调控规则:1) 如果当前的新闻已经是文本形式的,则设定新闻形式为低分辨率的多媒体形式
2) 如果当前的新闻是低分辨率的多媒体形式,则设定新闻形式为高分辨率多媒体形式
3) 如果当前的新闻已是高分辨率多媒体形式,且有多台服务器运行,则关闭一台服务器
4) 如果当前的新闻已经是高分辨率多媒体形式,且只有一台服务器运行,则什么也不做

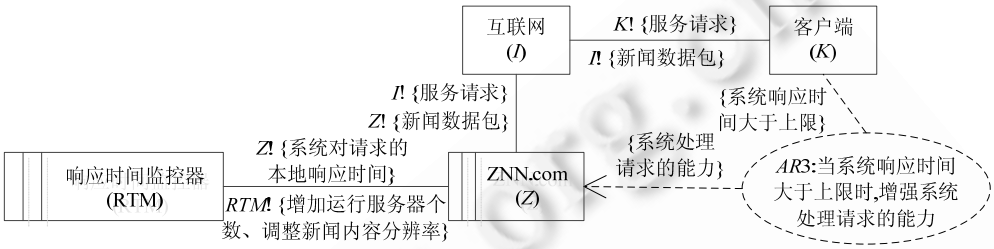
Fig.11 Diagram of the problem of adapting to the change of visiting number

图 11 应对用户规模变化的适应性问题图

假设初始情况下,系统只开启一台服务器、提供高分辨率的多媒体新闻服务,且此时系统能够响应的最大请求数目是 *N*,那么当用户规模增加导致单位时间内访问量大于 *N* 时,用户适应性需求 *AR*1 的调控规则就被触发,以增加 ZNN.com 的运行服务器个数或者降低新闻内容的分辨率,具体的调整规则如图 11 所示.而当单位时间内访问量减少的幅度>*N* 时,用户适应性需求 *AR*2 的调控规则就会被触发,以提升新闻内容的分辨率或者关闭

运行服务器,具体的调控规则如图 11 所示.这里的调控规则都是对调控措施的细化.

图 12 显示了用户适应性需求 *AR*3 背后的适应性问题图.由于在实际中监测包含网络传输和客户端显示时间在内的响应时间比较困难,一种更可行的方式是:首先,通过测试等方式估计数据在客户端和系统之间的传输时间;然后,将用户期望的响应时间减去该传输时间,得到用户期望的系统本地响应时间;进而监测 ZNN.com 的本地响应时间,并根据 ZNN.com 真实的本地响应时间与期望的本地响应时间的差别来调整 ZNN.com.因此,该问题的直接监控对象也是ZNN.com,直接监控现象就是单位时间内ZNN.com对请求的平均响应时间,而具体的调控措施仍然是通过调整运行服务器的个数和新闻内容的分辨率来实现对系统处理请求能力的调节.在实际运行过程中,当单位时间内 ZNN.com 的平均响应时间大于系统本地响应时间的上限时,相应的调控规则(如图 12 所示)就会被触发,以增加运行 ZNN.com 的服务器个数或者降低新闻内容的分辨率.



*AR*3:当单位时间内系统响应时间大于上限时,增强系统处理请求的能力
直接监控对象:ZNN.com
直接监控现象:系统对请求的本地响应时间
调控措施:增加运行服务器个数、调整新闻内容的分辨率
启动条件:单位时间内平均响应时间>MAX_{RS}(系统响应时间的上限)
调控规则:1) 如果有未启用的服务器,则启动一台服务器
2) 如果当前的新闻是高分辨率多媒体形式的,则设定新闻形式为低分辨率多媒体形式
3) 如果当前的新闻是低分辨率多媒体形式的,则设定新闻形式为文本形式
4) 如果当前的新闻是文本形式的,则报警提示当前单位时间内系统平均响应时间超过上限

Fig.12 Diagram of the problem of adapting to response time's excess over the maximum

图 12 应对系统响应时间大于上限的适应性问题图

4) 合并适应性问题图

合并图 8 所示的 ZNN.com 的初始问题图以及图 11 和图 12 所示的适应性问题图,自适应的 ZNN.com 的问题图如图 13 所示.如图 13 所示:我们要建立自适应的 ZNN.com,除了设计开发能够提供新闻服务的 ZNN.com 之外,还要设计开发能够根据用户规模变化来调节 ZNN.com 的负载监控器、能够根据系统响应时间变化来调节 ZNN.com 的响应时间监控器以及协调负载监控和响应时间监控器的自适应调节器.

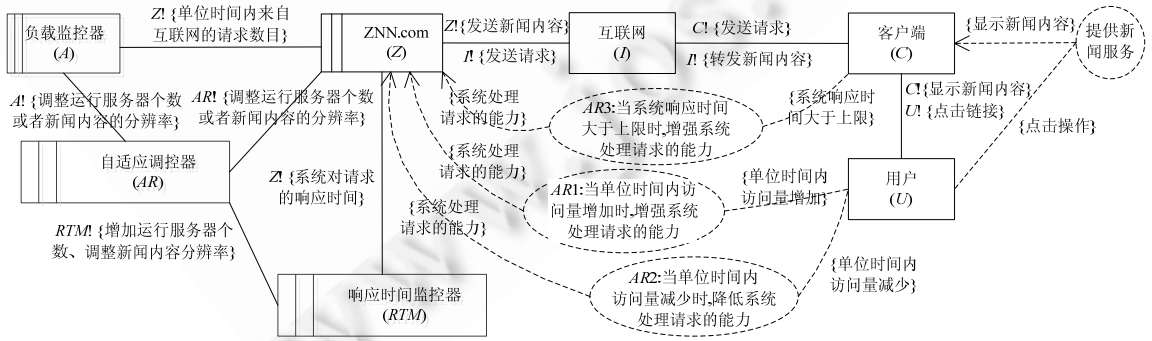


Fig.13 Problem diagram of the adaptive ZNN.com

图 13 自适应 ZNN.com 的问题图

自适应调节器的引入,是为了解决负载监控器和响应时间监控器之间潜在的冲突:当用户规模增加时,系统的响应时间会延迟,此时,负载监控器和响应时间监控器可能会同时给 ZNN.com 发出指令.在这种情况下,我们就需要设计一个新的软件单元来协调负载监控器和响应时间监控器的行为.显然,在该冲突中,负载监控器应该具有比响应时间监控器更高的优先级,因此,自适应调节器应该执行来自负载监控器的指令.

4 相关工作

对软件系统适应性需求的研究,最早可追溯到 Fickas 等人对环境监控需求(monitoring requirement)的研究[16].在这项研究中,Fickas 等人提出,环境在软件的运行过程中可能会发生变化并影响软件系统的正常运行,因此我们需要分析软件系统的监控需求,并提出了相应的分析方法.在此之后,随着自适应软件系统获得越来越多的关注以及适应性需求对于开发自适应软件系统所具有的重要性,越来越多的研究人员展开了适应性需求的研究[3].

当前,大部分的研究工作主要集中在如何利用目标模型来有效地分析和建模软件适应性需求.这是因为:一方面,面向目标的需求工程方法是当前需求工程领域的主流方法;另一方面,目标模型的"或"结构可以表达满足一个目标的不同方式,提供了一种简洁的方式来表示软件的适应性需求.在这方面,两个知名的研究工作是 Souza 等人提出的 AwReqs/EvoReqs 框架[4,5]和 Baresi 等人提出的 FLAG 语言[6,7].在 AwReqs/EvoReqs 框架中,感知需求(awareness requirement)表达需要在运行时通过获取后验信息来判断是否满足用户需求[4];进化需求(evolution requirement)[5]则表达感知需求一旦不满足时系统的适应性行为.在面向目标的需求工程方法的基础上,Souza 等人提出这两种新类型的需求来表达软件的适应性.而在 FLAG 语言背后,Baresi 等人的思路是基于模糊逻辑来增加目标模型中目标满足程度的灵活性,同时增加适应性目标来表达当其他目标不满足时系统需要采取的措施.除此之外,Cheng 等人在 KAOS 目标模型中,利用"妨碍性目标"来表达影响某个目标满足的环境不确定性,并通过添加一些子目标、或者更高层的目标、或者放松被影响的目标来表达应对不确定性的策略[17].Ali 等人扩展了面向目标的 Tropos 建模方法,将所依赖的不同环境明确标注在目标模型"或"结构的不同分支上,以表明"或"结构的不同分支表示的是不同环境下的软件需求[18].Fu 等人在目标模型之上为每个目标绑定一个由自动机所表达的行为模型,以支持运行时的监控[19].

虽然上述研究工作提出了具体方法来分析和建模软件适应性需求,但是对于如何系统地识别环境变化以及相应的适应性需求、如何区分和处理已知的和未知的环境变化,这些方法并没有提供有效的措施.同时,虽然目标模型的结构使其便于表达和推理分析软件适应性需求,并且利用目标模型来表达适应性需求也便于将适应性需求与软件的基本需求统一起来,但是目标模型的"与/或"结构也使其不能展现软件系统从监控环境变化、分析决策到调整自身行为的整个适应性过程.而这却对软件设计开发人员充分理解软件系统的适应性需求具有重要意义.

不同于这些工作,本文将问题框架方法与面向目标的方法结合起来,并基于反馈控制,将软件适应性需求的识别与分析转化为对由软件与其环境所构成的自适应控制系统的反馈回路的识别与分析.在这个过程中,我们不仅强调对环境的分析,使得分析人员可以在确定环境边界和特征的基础上采取系统性的方法来有效识别环境变化,同时通过分析该自适应控制系统的两种不同回路(环境感知反馈回路和需求感知反馈回路),可以有效地处理已知的和未知的环境变化.基于此,问题框架方法与面向目标方法的结合,也使得分析人员可以利用问题框架方法的结构化问题分析方法来具体地分析每个适应性需求背后的适应性问题,在分析适应性问题具体结构的过程中,充分展现系统需要监控的变量、控制的变量以及具体的监控和调控措施.在这方面,与本文工作类似的是 Salifu 等人对软件监控和调整问题的研究工作.在文献[20]中,Salifu 等人提出了应用问题框架方法来分析软件监控问题(monitoring problem)和调整问题(switching problem).这里的监控问题和调整问题,分析的是软件应该如何监控环境变化以及如何进行自我调整.虽然他们也给出了识别环境变化的建议性措施,但是对于如何系统地识别环境变化以及如何处理无法预知的环境变化,并没有给出相应的方法.

除了上述面向目标的分析方法之外,其他的一些经典工作包括 Whittle 所提出的 RELAX 语言[8]以及 Sawyer

等人所提出的需求感知(requirements awareness)框架[21,22].RELAX 语言是 Whittle 等人在考虑不确定性存在的情况下所提出来的,旨在通过引入一些松弛操作符(比如 as soon as possible,as early as possible)来表达那些具有弹性的软件需求的描述性语言.而在需求感知框架背后,Sawyer 所考虑的是我们无法在设计时预知所有的环境不确定性,因此,一种实现自适应性的有效方式是软件在运行过程中保存需求模型(称为 live requirements),并在线地根据环境变化调整需求模型,然后,通过需求模型与系统结构之间的关联性达到对系统的调整.虽然 RELAX 提供了一种表达适应性需求的有效方式,但是由于它是描述性语言,也无法清晰地展现适应性问题的具体结构,包括适应性问题所涉及的环境实体、从监控到调整的具体路径等信息.而需求感知框架虽然提供了构建自适应性软件系统的一种途径,但是目前并没有可供遵循的有效方法.

近年来,在将控制论的思想、方法和技术引入到软件工程的研究方面,也出现了许多研究工作.比如,蔡开元教授已明确提出了软件控制论这一新的研究方向[11].特别是,由于控制论在处理系统所面临的不确定性方面具有成熟的方法和技术,基于控制论来展开自适应软件系统的相关研究得到了许多关注,目前已有一些研究人员展开了初步的研究.比如:Cheng 等人声称,自适应软件系统的建立需要基于反馈控制原理[2];Brun 等人提出,通过建立反馈回路来实现自适应性[23];Muller 等人提出,需要在分析阶段明确自适应软件系统所应具有的反馈回路[24];Vogel 等人设计了一种建模自适应软件系统中反馈回路的类 UML 语言[25];Chen 等人设计了一个 PID 控制器,在运行时根据目标模型推理对 Web 系统的调控,以保证系统的可靠性[26];Souza 等人阐述了如何从需求工程的视角,将自适应软件系统建模为一个控制系统[27].但是不同于现有工作,本文将具有自适应能力的软件系统与其运行环境所构成的软件加强型系统,而不是软件本身建模为一个控制系统.这样就可以在软件适应性需求的分析过程中将环境明确地纳入到分析过程中来,而且可以将软件适应性需求的识别与分析转化为对该控制系统所应具有的反馈回路的识别与分析,以充分利用控制论的相关概念和方法来分析软件适应性需求.

5 结束语

基于控制论,本文提出了一种软件系统适应性需求的识别和分析方法.通过将自适应软件系统与其环境看作是一个自适应控制系统,该方法将软件适应性需求的识别与分析转化为对该控制系统反馈回路的识别与分析.基于该方法,我们不仅可以分析软件将要作用的环境,以系统性地识别环境的可能变化,还可以通过识别和分析环境感知反馈回路和需求感知反馈回路,来分别应对已知的和未知的环境变化;并且通过反馈回路,我们可以明确为应对环境变化,软件系统需要监控的现象、控制的现象以及具体的调控措施;最后,可以通过对反馈回路的合并来识别适应性需求之间潜在的冲突.

虽然本文以一个网站系统来说明所提方法的可行性,但是我们还需要更多的案例研究来验证其有效性.特别是在对具体调控措施的制定以及不同调控措施之间可能产生冲突的分析方面,我们还需要作进一步的研究.同时,对于一些安全至关重要的系统来说,如何利用形式化的方法来建模适应性需求、如何验证适应性需求的正确性以及适应性需求软件基本需求的一致性,也需要进一步的研究.

References:

[1] 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]

[2] Cheng BHC, de Lemos R, Giese H, Inverardi P, Magee J. Software engineering for self-adaptive systems: A research roadmap. LNCS 5525, 2009. 1−26. [doi: 10.1007/978-3-642-02161-9_1]

[3] 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]

[4] Souza VES, Lapouchnian A, Robinson WN, Mylopoulos J. Awareness requirements for adaptive systems. In: Proc. of the 6th Int'l Symp. on Software Engineering for Adaptive and Self-Managing Systems (SEAMS 2011). 2011. 60−69. [doi: 10.1145/1988008.1988018]

[5] Souza VES, Lapouchnian A, Mylopoulos J. (Requirement) Evolution requirements for adaptive systems. In: Proc. of the 7th Int'l Symp. on Software Engineering for Adaptive and Self-Managing Systems (SEAMS 2012). 2012. 155−164. [doi: 10.1109/SEAMS.2012.6224402]

[6] Baresi L, Pasquale L. Live goals for adaptive service compositions. In: Proc. of the ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems (SEAMS 2010). 2010. 114−123. [doi: 10.1145/1808984.1808997]

[7] 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]

[8] Whittle J, Sawyer P, Bencomo N, Cheng BHC, Bruel J-M. RELAX: Incorporating uncertainty into the specification of self-adaptive systems. In: Proc. of the 17th Int'l Conf. on Requirements Engineering (RE 2009). 2009. 79−88. [doi: 10.1109/RE.2009.36]

[9] Jackson M. Problem Frames: Analyzing and Structuring Software Development Problems. ACM Press, 2001.

[10] Hall JG, Rapanotti L, Jackson M. Problem priented software engineering: Solving the package router control problem. IEEE Trans. on Software Engineering, 2008,34:226−241. [doi: 10.1109/TSE.2007.70769]

[11] Cai KY, Cangussu JW, DeCarlo RA, Mathur AP. An overview of software cybernetics. In: Proc. of the 11th Annual Int'l Workshop on Software Technology and Engineering Practices (STEP 2003). 2003. 77−86. [doi: 10.1109/STEP.2003.4]

[12] Dumont GA, Huzmezan M. Concepts, methods and techniques in adaptive control. In: Proc. of the American Control Conf. (ACC 2002). 2002. 1137−1150. [doi: 10.1109/ACC.2002.1023173]

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

[14] Angelopoulos K, Souza VES, Pimentel J. Requirements and architectural approaches to adaptive software systems: A comparative study. In: Proc. of the 8th Int'l Symp. on Software Engineering for Adaptive and Self-Managing Systems (SEAMS 2013). 2013. 23−32. [doi: 10.1109/SEAMS.2013.6595489]

[15] Jureta I, Mylopoulos J, Faulkner S. Revisiting the core ontology and problem in requirements engineering. In: Proc. of the 16th IEEE Int'l Conf. on Requirements Engineering (RE 2008). 2008. 71−80. [doi: 10.1109/RE.2008.13]

[16] Fickas S, Feather MS. Requirements monitoring in dynamic environment. In: Proc. of the 2nd Int'l Symp. on Requirements Engineering (RE'95). 1995. 140−147. [doi: 10.1109/ISRE.1995.512555]

[17] 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]

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

[19] Fu LX, Peng X, Yu YJ, Mylopoulos J, Zhao WY. Stateful requirements monitoring for self-repairing socio-technical systems. In: Proc. of the 20th IEEE Int'l Conf. on Requirements Engineering Conf. (RE 2012). 2012. 121−130. [doi: 10.1109/RE.2012.6345796]

[20] Salifu M, Yu YJ, Nuseibeh B. Specifying monitoring and switching problems in context. In: Proc. of the 15th IEEE Int'l Conf. on Requirements Engineering (RE 2007). 2007. 211−220. [doi: 10.1109/RE.2007.21]

[21] Sawyer P, Bencomo N, Whitle J. Requirements-Aware systems: A research agenda for RE for self-adaptive systems. In: Proc. of the 18th IEEE Int'l Conf. on Requirements Engineering (RE 2010). 2010. 95−103. [doi: 10.1109/RE.2010.21]

[22] Welsh K, Sawyer P, Bencomo N. Towards requirements aware systems: Runtime resolution of design time assumptions. In: Proc. of the 26th IEEE/ACM Int'l Conf. on Automated Software Engineering (ASE 2011). 2011. 560−563. [doi: 10.1109/ASE.2011.6100125]

[23] Brun Y, Serugendo GDM, Gacek C, Giese H, Kienle H, Litoiu M, Müller H, Pezzè M, Shaw M. Engineering self-adaptive systems through feedback loops. LNCS 5525, 2009. 48−70. [doi: 10.1007/978-3-642-02161-9_3]

[24] Muller H, Pezze M, Shaw M. Visibility of control in adaptive systems. In: Proc. of the 2nd Int'l Workshop on Ultra-Large-Scale Software-Intensive Systems (ULSSIS 2008). 2008. 23−26. [doi: 10.1145/1370700.1370707]

[25] Vogel T, Giese H. A language for feedback loops in self-adaptive systems: Executable runtime megamodels. In: Proc. of the 7th Int'l Symp. on Software Engineering for Adaptive and Self-Managing Systems (SEAMS 2012). 2012. 129−138. [doi: 10.1109/SEAMS.2012.6224399]

[26] Souza VES, Mylopoulos J. From awareness requirements to adaptive systems: A control-theoretic approach. In: Proc. of the 2nd Int'l Workshop on Requirements@Run.Time (RE@RunTime 2011). 2011. 9−15. [doi: 10.1109/ReRunTime.2011.6046242]

[27] Chen BH, Peng X, Yu YJ, Zhao WY. Are your sites down? Requirements driven self-turing for the surviavability of Web system. In: Proc. of the 19th IEEE Int'l Conf. on Requirements Engineering (RE 2011). 2011. 219−228. [doi: 10.1109/RE.2011.6051650]

刘春(1982－),男,河南信阳人,博士,讲师,主要研究领域为需求工程.

赵海燕(1966－),女,博士,副教授,CCF 高级会员,主要研究领域为软件工程,领域工程.

张伟(1978－),男,博士,副教授,主要研究领域为需求工程,软件复用.

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,基于知识的软件工程.