软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2017,28(5):1247−1270 [doi: 10.13328/j.cnki.jos.005106]

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

软件非功能需求权衡代价*

张 璇[1,2], 王 旭[3], 李 彤[1,2], 白 川[1], 康燕妮[1]

[1](云南大学 软件学院,云南 昆明 650091)

[2](云南省软件工程重点实验室(云南大学),云南 昆明 650091)

[3](云南大学 经济学院,云南 昆明 650091)

通讯作者: 张璇, E-mail: zhxuan@ynu.edu.cn, http://www.ynu.edu.cn

摘 要: 软件非功能需求的实现涉及软件质量这一重要问题,非功能需求的满足程度,直接影响软件质量的满足程度.针对一直以来对软件质量的一贯重视以及软件非功能需求权衡的重要性,借鉴微观经济学领域的生产理论、替代弹性原理和线性规划方法,提出了软件非功能需求权衡代价分析方法并开发了辅助工具.首先,对项目组前期完成的可信软件非功能需求可满足性分析方法进行改进,提出了利益相关者通过协商获取非功能需求评估数据的方法,建立了非功能需求本体概念并构建本体知识库;针对实现非功能需求的策略,使用前期已完成的策略推理方法,对推理产生矛盾的策略提出权衡代价分析方法;通过分析策略实施代价,为软件开发及演化提供具有实际可操作的权衡决策依据,从更加符合产业化需要的角度解决软件非功能需求权衡问题;最后,基于可信第三方认证中心软件案例的维护及演化需要,对推理出矛盾的策略进行权衡代价分析,并给予决策建议,说明所提出方法的可行性.

关键词: 软件质量;非功能需求权衡代价;生产理论;替代弹性;线性规划

中图法分类号: TP311

中文引用格式: 张璇,王旭,李彤,白川,康燕妮.软件非功能需求权衡代价.软件学报,2017,28(5):1247−1270. http://www.jos.org.cn/1000-9825/5106.htm

英文引用格式: Zhang X, Wang X, Li T, Bai C, Kang YN. Trade-Off costs of software non-functional requirements. Ruan Jian Xue Bao/Journal of Software, 2017,28(5):1247−1270 (in Chinese). http://www.jos.org.cn/1000-9825/5106.htm

Trade-Off Costs of Software Non-Functional Requirements

ZHANG Xuan[1,2], WANG Xu[3], LI Tong[1,2], BAI Chuan[1], KANG Yan-Ni[1]

[1](School of Software, Yunnan University, Kunming 650091, China)

[2](Key Laboratory of Software Engineering of Yunnan (Yunnan University), Kunming 650091, China)

[3](School of Economics, Yunnan University, Kunming 650091, China)

* 基金项目: 国家自然科学基金(61502413, 61262025, 61379032, 61662085); 云南省应用基础研究计划(C0120150180); 云南省教育厅科学研究基金(2015Z020, 2013Z056); 云南省软件工程重点实验室开放基金(2015SE202, 2012SE308); 云南大学"中青年骨干教师培养计划"专项经费; 云南大学高水平创新团队"软件工程创新团队"专项经费; 云南大学人文社科项目基金(13YNUHSS007); 云南省创新团队"数据驱动的软件工程创新团队"项目基金

Foundation item: National Natural Science Foundation of China (61502413, 61262025, 61379032, 61662085); Science Foundation of Yunnan Province (C0120150180); Science Foundation of Yunnan Educational Committee (2015Z020, 2013Z056); Science Foundation of Key Laboratory of Software Engineering of Yunnan Province (2015SE202, 2012SE308); Young Teachers Special Training Program Funding of Yunnan University; Software Engineering Innovative Research Team Funding of Yunnan University; Social Science Foundation of Yunnan University (13YNUHSS007); Data Driven Software Engineering Innovative Research Team Funding of Yunnan Province

收稿时间: 2015-12-27; 修改时间: 2016-03-17, 2016-04-21; 采用时间: 2016-05-31; jos 在线出版时间: 2016-10-11

CNKI 网络优先出版: 2016-10-12 16:26:53, http://www.cnki.net/kcms/detail/11.2560.TP.20161012.1626.021.html

Abstract: The non-functional requirements are the determinants of the software quality. The satisfaction of the non-functional requirements impacts the software quality. Considering the importance of the software quality and non-functional requirements trade-off, based on production theory, elasticity of substitution and linear programming, an approach to analyzing trade-off costs for non-functional requirements is proposed and a support tool is developed. Firstly, our previous work about satisfiability analysis on non-functional requirements is improved. A new coordination method for obtaining non-functional requirements from stakeholders is presented. Then, ontology for non-functional requirements and corresponding knowledge base are constructed. The previous reasoning for the strategies is provided for decision-making of software development and evolution. Because of the ability of addressing the conflict relationships of the non-functional requirements, our method is better for the software industry. Finally, through applying the non-functional requirements trade-off costs for maintenance and evolution of the trustworthy third-party certificate authority software case, feasibility of the proposed approach is described.

Key words: software quality; trade-off cost of non-functional requirement; production theory; elasticity of substitution; linear programming

软件非功能需求也称为质量需求或者质量属性,实际上是软件的属性或者是功能需求的约束.通常,非功能需求被认为是二级需求,没有功能需求那样重要,然而,一些软件成功与否以及用户对软件质量是否满意,则是由非功能需求决定的.因此,对于任何一个软件,非功能需求和功能需求同等重要.然而,非功能需求与功能需求不同,功能需求描述的是软件具体要做的事情,可以用功能点构成的集合来描述;而非功能需求面向的是软件的整体属性,通常描述的是软件满足某些属性的程度,难以用一种统一度量的方式来表达.因此,功能需求可以通过分而治之的方法简化,而非功能需求反映的是软件作为一个整体应具有的属性,此整体属性的满足不是通过简单累加非功能需求可以得到的,其满足不仅受到成本、时间和风险等各种因素的制约,更困难的是,软件非功能需求存在权衡问题,即当我们运用特定策略提升软件的特定非功能需求时,有可能会导致其他非功能需求受到负面影响,导致整体软件质量下降,甚至软件的失败.一个常用的典型示例是用于提升安全性的密码强度设计策略,当我们通过增强密码强度来提升安全性时,系统响应时间会受到影响,而系统响应时间在一些项目中是用户需要软件满足的重要指标.例如,在 Boehm 分析的一个早期 TRW 为政府开发信息检索分析系统的案例中提到,当最初设定的系统响应时间为 1s 时,预算项目成本为 1 亿美元,这个成本是客户无法接受的.为了让项目能够继续,经过原型分析后,系统响应时间设定为 90%的用户能够接受的 4s,项目预算成本降为 3 千万美元[1].这个案例说明,在软件项目中,不可避免地要解决非功能需求权衡的问题.另外,随着信息基础设施和现代生活对软件依赖程度的迅速增加,这个问题则更为突出,因为软件非功能需求的实现涉及软件质量这个重要的问题,非功能需求的满足程度直接影响软件质量的满足程度.用户通过软件具有一组表达其质量属性的客观能力这一事实,信任软件的行为能够实现其设定的目标[2].这也符合 TSM(trusted software methodology)有关软件可信性的定义,即“软件满足一系列需求的信任程度”[3].因此,非功能需求权衡问题是所有软件不可避免地需要解决的重点问题.

然而,非功能需求的权衡问题也一直是困扰软件工程师及学者的一个难题.经过多年的研究与探索,学术界和产业界提出了越来越多有价值的理论和方法.经过对这些相关领域研究工作的总结,我们认为,目前的方法大部分是用于获取满足非功能需求的策略[3−23],其中的一部分方法能够进一步对存在冲突的策略进行权衡分析,给予策略选择的建议[24−36],但是我们还可以再进一步,通过展示策略中需权衡方案之间的关系和权衡过程以及对权衡代价的分析,让软件项目相关决策者充分掌握非功能需求权衡时所付出的代价,有利于获取非功能需求的最优权衡.另外,面对不断变化的软件环境,非功能需求的权衡应该能够适应环境变化,例如,当技术进步、成本变动或者平台升级时,策略方案的选择应该具备对这些变化的适应性.

本文基于项目组前期提出的可信软件非功能需求可满足性分析结果[4],借鉴微观经济学中的生产理论、生产要素弹性替代原理和线性规划方法,提出软件非功能需求权衡代价分析方法.代价在本文中定义为:因策略实施,一个非功能需求提升导致另一个冲突非功能需求下降的损耗代价或减小代价.此代价本身反映的是非功能需求之间权衡的关系,但由于代价都是因策略执行而产生的,因此,本文后续内容中统一称其为策略代价.对非功能需求进行权衡代价分析的目的是:保证高优先级非功能需求在得到满足的同时,与之存在冲突的非功能需

求的损耗或者减小代价是能够接受的,从总体上保证软件的质量;同时,控制后期因质量问题而可能产生的投入成本.

1 相关工作

软件需求包含功能需求和非功能需求.目前,大部分比较成熟的需求工程方法都侧重于对功能需求进行建模,而对非功能需求的处理缺乏特别合适的解决策略.这主要是因为对非功能需求进行分析与建模非常困难,涉及面广,而且通常需要用领域特有的方式[37].另外,非功能需求难以处理还有一个重要原因是非功能需求之间存在交互关系,此交互关系导致在满足某一非功能需求时所使用的策略(在面向目标需求工程中称为软件目标)会影响其他非功能需求的可满足性.因此,对非功能需求进行权衡研究是需求工程的重要研究方向之一,相关研究工作以非功能需求的建模和推理为基础展开.

软件需求建模是提供一个指定的模型用于构造需求的抽象描述,针对非功能需求的建模.由于非功能需求之间可能存在的冲突或促进关系,建模工作的主要目的是描述这些关系,并推理在这些关系存在的情况下,对应非功能需求可行的策略,推理得到的策略和相应满足状态用于支持软件设计、过程建模和管理决策.在这一领域,Boehm 等人较早地对非功能需求间冲突的识别和诊断进行了研究,并设计实现了一个辅助非功能需求分析、识别、诊断冲突的模型及工具 QARCC(quality attribute risk and conflict consultant)[5],为非功能需求权衡问题的研究开创了先河.Mairiza 等人通过大量的文献研究,基于软件应用场景分析非功能需求之间的冲突问题,总结出 20 个非功能需求之间的冲突关系[38],表明软件非功能需求冲突的普遍存在.与此类似的相关文献不断提出来,其中很大一部分以面向目标需求工程方法为主,面向目标需求工程方法的研究包括建模及推理两个部分:建模的目的是描述实现非功能需求的策略及策略引入的非功能需求冲突及促进关系;推理包括向前推理和向后推理,向前推理是通过策略的满足状态来推理软目标(在面向目标需求工程中,软目标即非功能需求)的满足状态,向后推理是根据软目标的满足状态推理策略的满足状态.KAOS(knowledge acquisition in automated specification)[6,7],NFR(non-functional requirements)框架[8−11]和 *i**家族(包括 *i**模型、Tropos 和 GRL(goal-oriented requirement language))[12−15]是最具代表性的面向目标需求工程方法,很多相关研究都基于他们的研究成果进行扩展.Burgess 等人改进了 NFR 框架定义依赖规则集,通过自动传播状态获取满足特定非功能需求集合的最优功能,他们下一步的工作是将软目标实现所需代价,包括开发时间、成本、风险等作为评估软目标的因素,在最小成本控制下寻找最优策略[16].Liaskos 等人认为,偏好需求应区别于强制需求,因为不同利益相关者有不同的偏好需求,在制定设计决策时应考虑偏好需求的作用.因此,他们提出了基于偏好的规划方法,将偏好需求优先级用于评估实现强制需求方案的标准,从而提出了符合利益相关者需求的设计决策[17].Wei 等人提出了 Σ 形式化描述语言以描述目标树模型,实现设计方案选择的自动向前推理[18].Sebastiani 等人于 2004 年首次提出用可满足性问题求解方法 SAT(satifiability)实现目标模型的向后推理[19].2005 年,Giorgini 等人将 SAT 方法应用于 Tropos 中,实现了向前及向后推理[20,21].2010 年,Horkoff 等人使用合取范式 CNF(conjunctive normal form)定义 *i** 框架的标记传播规则,提出了主体-目标需求模型的向后推理方法[22,23].我们的前期工作通过建立可信软件非功能需求驱动的过程策略选取框架 TRPF(trustworthiness requirements-processes framework),使用可信需求元模型 TRMM(trustworthiness requirements meta-model)构建可信软件非功能需求与过程策略关系模型,通过过程策略反映非功能需求之间的冲突及促进关系,通过可满足性问题求解推理满足非功能需求的过程策略及满足状态,并根据推理结果提供过程建模建议[4].以上这些方法所产生的研究成果为非功能需求的研究奠定了重要的基础,可以为软件开发决策提供有效的建议,但仍然需要更为详细的权衡分析方法提供更为明确的证据,才能更好地辅助决策的制定.例如,实施一个存在冲突的策略,具体应该采用什么算法,能够增加多少资源,如何设计模块间关系,是否可以使用新出现的技术等等.这些具体问题的答案需要我们对非功能需求进行度量,并对存在冲突的策略进行量化权衡.

当然,针对具体的软件,非功能需求有可用的度量方法[39],如果基于这些量化的数据进行策略选择,就可以得到更优的决策建议.在此领域,也有很多学者提出了相关的方法.陈仪香教授指导的陶红伟博士在其博士学位

论文中,基于 McCall 和 Deutsch 给出的属性间关系模型,从集合论的角度给出影响软件可信性的关键属性(功能性、可维护性、可靠性和可生存性)间的量化关系[40].Boehm 等人在实现非功能需求分析、识别、冲突诊断并开发辅助工具 QARCC 后,基于 COCOMO II 成本估算模型[41],设计了 S-COST(software cost option strategy tool)模型及工具,依据非功能需求的冲突关系进行潜在成本冲突分析,并在预算控制范围内进行面向成本的策略权衡分析[24].Letier,van Lamsweerde 和 Heaven 等人量化了系统设计方案对非功能需求目标满足程度的影响,通过向前及向后状态传播规则,实现定性定量相结合的决策支持;并通过量化目标模型仿真设计方案,通过对设计方案的仿真评估设计方案对目标满足贡献的等级状态,应用多目标优化算法搜索设计空间中的优化系统设计方案[25−28].Ma 等人利用网构软件功能满足度与风险评估数据构造无差异曲线,实现对网构软件服务的评估[29].Marew 等人在面向对象分析与设计中集成非功能需求策略,使用层次分析法 AHP(analytical hierarchy process)和量化软件目标依赖图 Q-SIG(quantified softgoal interdependency graph)量化分析策略优先级,依据量化分析结果数据提供策略决策依据,辅助面向对象分析与设计[30].与 Marew 等人的工作类似,Zhu 等人使用模糊集合论和软目标依赖图 SIG(softgoal interdependency graph)构建模糊定性定量软目标依赖图 FQQSIG(fuzzy qualitative and quantitative softgoal interdependency graph)模型,进行非功能需求的交互关系分析,通过从利益相关者获得的需求数据,计算部分可量化策略的方案优先级数值;另外一部分策略则提供设计建议,为设计决策提供参考依据[31].Elahi 等人提出:在没有精确成本效益量化数据的情况下,使用相等交换多标准决策分析方法,通过重用交换,在优化决策方案时不仅可以减少向利益相关者获取交换数据的次数,而且无需引入方案的细节评估数据及标准权重就可以获取优化决策[32].Yin 等人使用 0-1 规划方法寻找满足非功能需求的功能实现最优方案,通过量化非功能需求对功能的约束,选择功能实现策略,将策略选择问题限定为 0-1 规划问题,支持策略自动优化选择,提供具有实际可操作性的策略选择方法[33].Wei 等人针对网构软件的开放、动态和易变特征,基于面向目标非功能需求推理,研究网构软件在动态构造过程中决策驱动的非功能需求定性及定量推理方法[34].Asadi 等人将非功能需求优化问题纳入特征模型自动化配置研究中,使用层次分析法 AHP 和模糊认知图 FCM(fuzzy congnitive maps)计算利益相关者对非功能需求的偏好权重,为特征配置提供非功能需求相关的决策依据[35].在 Boehm 的系统质量本体中,系统质量的交互关系用矩阵形式描述,通过使用 COCOMO II 成本估算模型[41]计算不同策略执行成本,给出基于成本使用策略的建议[36].以上这些方法根据度量非功能需求数值或者从利益相关者获取评估数据辅助决策分析,通过量化数据,建议从有限的策略方案中选择其一.本文希望将此决策分析过程进一步细化,通过反映策略方案间的关系以及非功能需求的权衡过程,即展示出策略多个选择方案之间的代价关系以及最优策略方案选择的依据,为软件设计、过程建模及管理决策提供更为直观和可操作的决策建议.另外,当软件环境发生变化时,适应性地展示变化前后以及变化之间策略代价的差异,有利于决策者全面而清晰地掌握策略决策依据,并且能够随变化而调整决策.

2 软件非功能需求权衡代价分析框架

非功能需求涉及软件质量这个重要的问题,非功能需求的满足程度直接影响软件质量的满足程度.但是,非功能需求又很难处理.首先,非功能需求是主观的,不同的人对它有不同的看法、不同的理解、不同的解释以及不同的评价方式;其次,非功能需求是相对的,对非功能需求的解释以及它的重要性按照要考虑的软件的不同而不同,非功能需求的实现也是相对的,因为可以不断改进现有的技术提高非功能需求的满足程度;最后,非功能需求之间是存在交互的,满足一个非功能需求可能会影响其他非功能需求的可满足性.总之,非功能需求很难处理,但处理好非功能需求的问题对软件开发的成败起决定作用,需要有效的方法[37].

面向非功能需求的上述 3 个典型问题,项目组前期针对可信软件,提出了一个可信软件非功能需求驱动的过程策略选取框架 TRPF[4].TRPF 由 3 个流程步骤构成.首先,基于可信软件非功能需求已有相关研究成果,提出可信软件非功能需求分解模型,在此基础上,使用模糊集合论和信息熵方法提出可信软件非功能需求获取方法,该获取方法保证非功能需求数据的有效性;然后,提出基于 TRMM 和知识库的可信软件非功能需求与过程策略关系建模方法,其中,将可信软件非功能需求分解为可信关注点和软目标,针对可信软件的可信关注点提出过程

策略,同时分析过程策略与软目标和其他可信关注点之间复杂的相关关系;最后,为权衡可信关注点以及软目标,基于可满足性问题求解方法提出选取过程策略的推理方法,并开发推理辅助工具 TACD(trustworthiness attributes collaboration diagnosis),找出满足可信软件非功能需求的过程策略.

通过 TRPF 这 3 个流程的实施,可以找出满足可信软件非功能需求的过程策略集合.然而,在这个集合中有一部分策略是存在冲突的,针对这些存在冲突的策略,在前期研究成果中给出了策略执行的建议,即通过活动间依赖关系进行限制,或者不执行无法进行限制的策略.本文继续这一工作,提出面向冲突策略的非功能需求权衡代价分析方法,并将此方法扩展至 TRPF,如图 1 所示.扩展之后的 TRPF,为了更加符合其目标,更名为 NRTF (non-functional requirements trade-off framework).在原 TRPF 的基础上,NRTF 实施了两个改进并增加一个扩展步骤,在图 1 中,加粗深色背景字体描述了改进和扩展的部分.

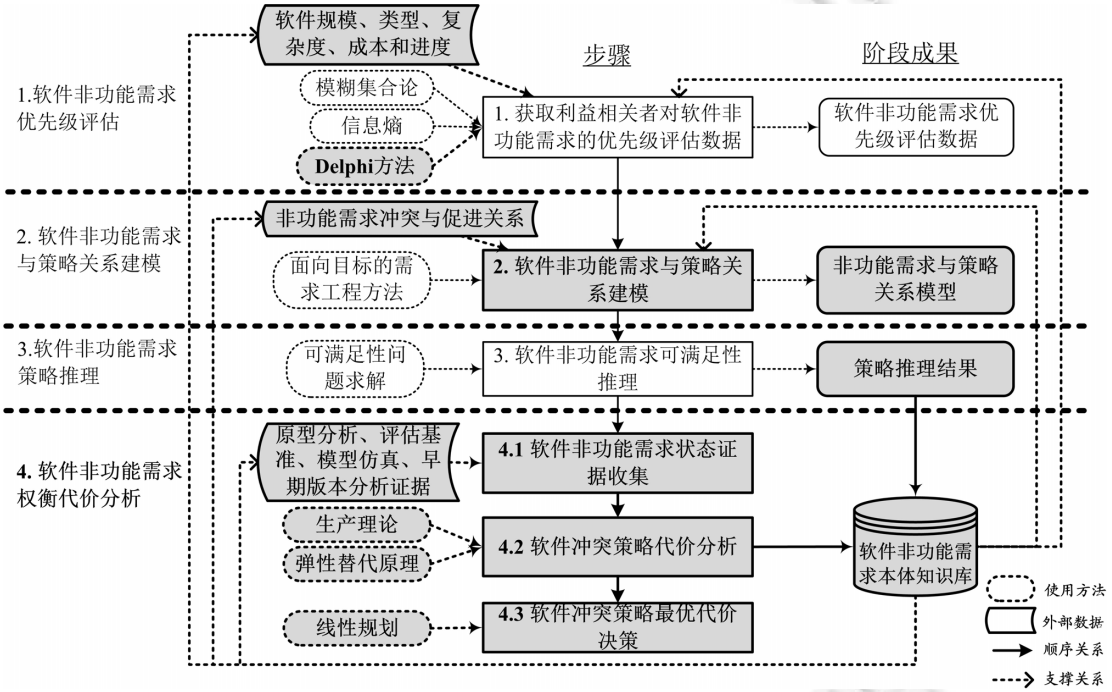


Fig.1 Non-Functional requirements trade-off framework (NRTF)

图 1 非功能需求权衡框架 NRTF

NRTF 的两个改进是:(1) 在软件非功能需求优先级评估阶段,引入软件规模、类型、复杂度、成本和进度等参照因素,辅助利益相关者更加准确地对非功能需求优先级进行评估,另外,使用 Delphi 方法[42]实现利益相关者的需求协商,以获取利益相关者一致同意的需求优先级评估结果;(2) 在软件非功能需求与策略关系建模与推理阶段,建立非功能需求本体概念和知识库,辅助建模者完成非功能需求与策略关系建模及权衡代价分析,为软件开发及演化提供全面决策依据.

在上述两个改进的基础上,NRTF 增加了一个扩展步骤:在原 TRPF 推理获取策略后,增加非功能需求权衡代价分析阶段,收集非功能需求状态证据,利用生产理论、生产要素弹性替代原理和线性规划方法,分析存在冲突的策略代价,基于证据数据和代价数据提供策略方案选取依据,支持软件设计、过程建模及管理决策的制定.

下面对 NRTF 的两个改进进行介绍,然后在第 3 节介绍扩展的非功能需求权衡代价分析方法.

1) 软件非功能需求优先级评估数据获取

软件非功能需求的优先级如果仅仅依赖软件利益相关者提供的评估数据来计算,由于评估数据的主观性,评估结果完全依赖于利益相关者对于软件和软件应用领域的熟悉程度,并且不可避免地会引入利益相关者的

偏见,导致初始获取的非功能需求不准确,甚至错误.为了尽量消除可能引入的偏见因素,在 TRPF 中,我们使用信息熵检验利益相关者提供数据的有效性,对于超出熵阈值的数据,采取重新获取或者从数据集中删除的方法消除一定的主观性.但是对于不同软件,其非功能需求实际上受到软件本身的规模、类型、复杂度、成本和进度等客观因素影响.因此,在利益相关者提供软件非功能需求评估数据时,为利益相关者增加软件规模、类型、复杂度、成本和进度等客观因素作为参考,可以进一步增强非功能需求评估的有效性.表 1 给出一个示例,这个示例参考了 Mairiza 和 Aowghi 基于大量文献分析 114 类非功能需求后,总结出的软件类型与非功能需求的匹配关系[38].

Table 1 Mapping of NFRs and types of software

表 1 软件类型与非功能需求关系

非功能需求	软件类型				
	实时系统软件	安全关键软件	过程控制软件	信息系统软件	Web 软件
空间性能	√	√	√	√	√
易用性	√	√	√	√	√
时间性能	√	√	√	√	√
精确性	√			√	
可扩展性	√				√
互操作性				√	√
可靠性	√	√	√	√	
可维护性	√		√		
可用性	√		√	√	
可生存性	√	√	√		
机密性	√	√		√	√
完整性	√	√	√	√	√
防危性(safety)	√	√	√		
可信赖性	√	√			
可修改性(repairability)				√	
可改变性(modifiability)				√	
隐私性				√	√
可移植性	√				
可重用性				√	
兼容性	√				

和软件类型一样,软件规模、复杂度、成本和进度等都可以为不同软件非功能需求提供不同的客观参考价值,当面对一个具体软件项目时,这些参考因素可以辅助项目组获得更为客观的初始非功能需求数据.当然,这些参考数据应随着技术的进步和环境的变化,采用迭代方式不断获取、细化并调整完善.当面对一个软件兼具多个软件类型时,需要综合考虑.另外,还需要根据软件所处的生命周期阶段,分别按照新开发、维护和演化进行调整.

当从利益相关者获取的软件非功能需求优先级评估数据不一致时,简单地要求利益相关者重新提交评估数据或者直接删除其评估数据,会导致部分利益相关者的需求不能得到满足,而且在这个过程中,有可能正确的需求会被忽略或删除.有效增强软件利益相关者之间的沟通协作,可以更好地保证获取的最终非功能需求状态是全体利益相关者一致需要软件具备的质量状态或条件.在交互式决策制定方法中,Delphi 方法[42]是最为广泛接受并使用的方法.Delphi 方法是 20 世纪 50 年代前后由 Dalkey 等人在 Rand 公司开发的.Delphi 方法的特征是通过多次协商过程的迭代,最终获得多人统一认同的结论.在获取软件非功能需求优先级评估数据时,使用 Delphi 方法,遵循如下步骤.

步骤 1. 从不同利益相关者处获取各自的评估数据,对评估数据进行熵运算,如果不存在分歧意见,则输出评估数据;如果存在分歧意见,则将相关分歧意见和已经统一的意见进行整理,并将所有意见以列表的形式详细地表示出来.对已经统一的和存在分歧的意见给出相应比例后,反馈给所有利益相关者,并且进入步骤 2.

步骤 2. 利益相关者在获取上述反馈数据后,需要决定修改原评估数据或给出维持原评估数据的理由,

再次提交新评估数据后,如果已经不存在分歧意见,则输出评估数据;如果仍然存在分歧意见,则再一次对已经一致的和仍然存在分歧的数据进行总结,并反馈给所有利益相关者,进入步骤 3.

步骤 3. 利益相关者根据总结的反馈意见和其他利益相关者提出的理由,修改评估数据,形成最终评估数据.如果对最终评估数据进行熵运算,其结果值在阈值范围内,则完成数据采集;如果结果值仍然表明存在分歧意见,则继续上述步骤,直到达成一致.

通常,采用 Delphi 方法能够在 3~5 轮迭代后达成一致意见[42].Delphi 方法引入的重要性在于:当少数利益相关者持有正确评估意见时,通过 Delphi 方法中提交理由阶段可以有效实现利益相关者之间的沟通,避免少数利益相关者的正确意见被忽略或删除而影响最终评估数据的正确性.更重要的是,利益相关者之间的协商,可以避免因利益相关者需求失衡而导致项目失败的问题.

2) 软件非功能需求本体知识库

本体(ontology)概念来源于哲学,在哲学领域是关于存在本质的研究.运用到计算机与信息科学领域,本体用于领域知识建模,描述一个概念体系.在需求工程领域.针对非功能需求,由于其特定的主观性,已经积累了很多丰富但并不完全一致的定义,这导致在软件开发及演化过程中,来自不同领域的利益相关者难以协作,而这样大量不一致的定义问题还在加剧中,毕竟软件工程现在以及将来都面临着快速变化带来的强有力挑战.但是,统一对非功能需求进行定义是不可行的,不同的学科领域对非功能需求的定义本质上应该存在着差异,而且统一的非功能需求定义也是不必要的,因为软件开发及演化处于快速变化中,非功能需求的定义应该能够适应变化.既不能统一定义,又需要跨领域协作,为了解决这个问题,Boehm 在 INCOSE 2015 上提出了系统质量(system quality,简称 SQ)本体概念[36],按照软件工程需要,改进 Gruber 本体定义[43],提出系统质量本体结构 SQDS(SQ definition structure),期望通过质量本体定义,能够在不同学科领域利益相关者进行软件系统定义、开发及演化时提高协同工作效率.

Boehm 的质量本体定义结构 SQDS 包含类(class)、个体(individual)、参照(referent)、状态(state)、过程(process)和关系(relation)这 6 个元素,其中,类是非功能需求,类的层次结构是非功能需求的分解结构,不同应用领域、不同软件、不同学科都可以定义各自的分解结构;个体是描述具有不同参照、状态、过程和关系的子类;参照是度量非功能需求的指标数据;状态分为内部状态和外部状态,内部状态使用具体的参照指标数据表达质量满足等级,系统外部因素定义为外部状态;过程是实现不同非功能需求时使用的不同过程和活动;关系描述的是非功能需求之间的冲突及促进关系.

建立本体的终极目标是要建立知识库,通过借鉴 Boehm 的质量本体概念,我们改进原有知识库[4],建立非功能需求本体知识库,非功能需求本体知识库改进原有知识库中的非功能需求库和过程策略库.对于非功能需求库,改进后仍然提供利益相关者及对应的非功能需求以及这些非功能需求的分解结构,但同时增加非功能需求的参照指标、状态和过程知识.增加新知识后的非功能需求结构仍然使用扩展的巴克斯范式 EBNF(extended Backus-Naur form)定义,定义中的“[ ]”表示可选定义的结构成分,可选意味着可以出现 0 次到 1 次,“{ }”表示重复定义的结构成分,表示可以重复 0 次到无数次.为使描述简洁,下面仅给出增加的结构定义部分,原结构定义参见文献[4].

- 非功能需求=‘NFR’, 非功能需求名, 非功能需求描述, {利益相关者}, {子需求}, {参照指标}, {状态}, {过程}.
- 参照指标=‘REFERENT’, 参照指标名, 参照指标描述, [参照指标公式].
- 状态=‘STATE’, 内部状态, {外部状态}.
- 内部状态=内部状态名, 内部状态描述, {参照指标}.
- 外部状态=外部状态名, 外部状态描述, 外部状态影响.
- 过程=‘PROCESS’, 过程名, 过程描述, 活动, “;”, {活动}.

在这里,我们将活动定义为一个抽象数据类型,定义数据结构及在数据结构上的操作:

- 活动=‘ACTIVITY’, 活动名, [‘IMPORTS:’, 变量声明, “;”, {变量声明}], [‘EXPORTS:’, 变量声明,

“;”, {变量声明}], [‘LOCALS:’, 变量声明, “;”, {变量声明}], ‘BODY’, 活动体.

定义中的 IMPORTS 子句、EXPORTS 子句和 LOCALS 子句分别定义了输入、输出和本地数据结构,描述了活动执行的输入条件、输出结果和活动内部处理.BODY 定义了活动体部分,活动体可以是一个细化活动的子过程,也可以由细化的任务列表构成.

- 变量声明=变量名, ‘:’, 变量类型.
- 变量类型=“STRING” | “INTEGER” | “ROLE” | “MESSAGE” | “Report_Type” | “Notification_Type” | “Procedure_Type” | “System_Type” | “Requirements_Type” | ”Plan_Type”.
- 活动体=任务, “;”, {任务} | 过程名.
- 任务=‘TASK’, 任务名, 任务描述.

对于过程策略库,采用迭代增加的方式不断扩充完善策略,建立策略库,策略库仍然存储策略和策略对非功能需求的影响关系.表 2 给出了部分策略对非功能需求影响的示例.

Table 2 Examples of strategies’ effects on the non-functional requirements

表 2 策略对非功能需求影响的示例

非功能需求	策略	促进的非功能需求	冲突的非功能需求
安全性(security)	单代理密钥分发	–	可靠性,易用性
	单点数据备份	–	可靠性
	完整性检验,分析攻击面	可靠性	–
	DOS 攻击检测与防御	–	时间性能
	增加认证强度,实现最小密码设计,建立多层防御体系	–	易用性,时间性能
	数据加密,定义最小安全标准	–	时间性能
可信赖性	增加硬件冗余	防危性(safety),可靠性	安全性,成本
	形式化验证	–	时间性能
	自动输入输出确认	资源效用	–
	模块化	灵活性	–
	数据加密,可信监控	安全性	时间性能
	领域内架构重用	灵活性	–
兼容性	建立多领域架构	可移植性,互操作性	可信赖性,可维护性
	多接口设计	可维护性	安全性
可维护性	增加访问接口	–	安全性
	智能监控	安全性	–
	软件诊断	安全性	–
可修改性	松耦合	–	时间性能
	增强设计和验证	–	成本,进度
	增加多入口	–	安全性
	采用敏捷方法	–	安全性,防危性
	增加备用空间	可靠性	安全,成本
可靠性	增加边界盾牌	–	可维护性
	可靠性优化设计	–	灵活性
	冗余设计	防危性	可维护性,安全性
	错误树分析	安全性	–
	失效模式与影响分析	安全性	–
	容错设计	易用性	安全性,精确性
	防差错设计	易用性,安全性	–
灵活性	用户编程	互操作性	可信赖性
	模块化	可信赖性	–
易用性	提供多项选择	–	可维护性

这些策略对非功能需求的影响体现为非功能需求之间的冲突或促进关系,对于其中产生冲突关系的策略,为了分析其影响相关非功能需求的权衡代价,需要在原策略库中增加策略方案知识.为使描述简洁,下面仅给出增加的策略方案结构,原结构定义参见文献[4].

- 策略库=非功能需求名, ‘:’, {策略}.

- 策略='STRATEGY', 策略名, 策略描述, [贡献, [冲突控制], {策略方案}].

在文献[4]中,策略贡献即为策略对非功能需求的影响,反映了非功能需求之间的关系.当策略贡献是导致非功能需求产生冲突时,文献[4]给出了冲突控制建议.本文增加策略方案知识,支持非功能需求权衡代价分析.

- 策略方案='SOLUTION', 策略方案名, 策略方案描述, ['(', 非功能需求 *i*, '.', 参照指标名, ',', 非功能需求 *j*, '.', 参照指标名, ')'].

其中,非功能需求 $i(1\leqslant i\leqslant m)$和非功能需求 $j(1\leqslant j\leqslant m)$是策略引发冲突的 m 个非功能需求中的两个,参照指标定义在非功能需求库中.

本体知识库的开发采用迭代方式增加、修改或细化其中的知识,逐步构建非功能需求的本体知识概念体系,并通过知识库的使用与共享,不仅在不同项目中提供可重用知识,加快项目进度、保证项目成功率,而且,通过不同项目的使用,迭代扩充知识库,适应动态变化的软件环境.

基于知识库,在获取利益相关者非功能需求优先级评估数据后,使用 TRMM[4]建模软件非功能需求与策略之间的关系,不仅描述实现软件非功能需求的策略,更重要的是,展示出非功能需求之间因策略可能产生的冲突和促进关系.在此基础上,使用可满足性问题求解方法进行向后推理[4],能够让项目组在接受一定程度的矛盾(此推理矛盾本质上是由非功能需求之间的冲突关系导致的)的情况下找到满足非功能需求的策略集合.对于此集合中存在矛盾的策略,如果能够进一步解决矛盾,找到最优非功能需求权衡,则会为决策者提供更有价值的决策建议,有利于策略的有效实施.例如,冗余设计有利于提升软件可靠性,但冗余的物理或数字空间会成为软件可能被入侵的新资源,那么,应该增加多少冗余空间是最有利于权衡可靠性和安全性?诸如此类的权衡问题,是本文下面要解决的问题,即对因冲突而导致推理出矛盾的策略进行对应非功能需求权衡代价分析,找出实施策略提升一个非功能需求而让另一个非功能需求降低或者损耗的最优代价.

下面,我们借鉴微观经济学中的生产理论、替代弹性原理和线性规划方法进行非功能需求的权衡代价分析,为软件设计、过程建模和管理决策提供最优的、实际可操作的权衡建议.

3 软件非功能需求权衡代价分析

非功能需求的冲突本质上是因策略执行而不可避免引入的,不执行引入冲突的策略则不会导致冲突.但很多情况下,为了提升非功能需求,必须执行策略,只是必须权衡好如何最优执行策略,或者说,找到存在冲突的非功能需求之间的最佳权衡关系.借鉴微观经济学的生产理论,可以将策略引发的非功能需求的提升与降低理解为策略的代价.例如,在使用认证策略时,安全性和系统处理时间可以视为认证策略的代价要素,当使用较高安全性的签名认证策略时,系统处理时间在奔腾 3 处理平台上是 38ms;如果降低安全标准,采用口令或者基于哈希算法的认证策略,系统处理时间相应降低为 28ms 和 30ms.也就是说,认证策略提升安全性的代价是牺牲系统响应时间,或者提高系统响应时间的代价是降低安全性.高安全性必然导致处理时间增长、系统响应速度减慢,而降低安全性则可以提高系统处理时间,提升系统响应速度.当然,不是所有策略都可以通过对非功能需求的度量提供量化数据,因此,我们将策略分为两类.

- 一类是可度量策略.这类策略如我们上面列举的两个示例:冗余设计策略和认证策略.这类策略的代价可以通过策略影响的非功能需求的度量数据,使用弹性替代原理分析策略代价,通过构造策略代价线,结合利益相关者对非功能需求的优先级和非功能需求的状态约束选取最优策略代价.在可度量策略中,如果策略需要考虑成本代价,例如增加备用空间策略可以提高可修改性和可靠性,但增加的备用空间显然需要投入成本,对于这类策略的成本代价分析,可以使用 Boehm 的 COCOMO II 成本估算模型[41,44]对此类策略的投入成本进行代价估算.
- 另一类是不可度量策略.这类策略不能通过度量其所影响的非功能需求提供量化数据,例如,使用敏捷方法策略可以提高软件开发效率并解决不断变化的需求,但无法保证所开发出软件的安全性和防危性.由于敏捷方法是一种软件工程方法,实施整套方法会导致上述非功能需求之间产生交互关系,但无法基于敏捷方法对其所影响的非功能需求进行度量,对于这类不可度量的策略,只能通过经验评估给

予决策参考.

在上述两类策略中,不可度量策略通过经验评估给予决策参考建议,可度量策略中与成本代价相关的策略使用 COCOMO II 成本估算模型[41,44]计算其成本代价.在此我们不做详细分析,下面对可度量策略中其他非成本代价策略进行分析.

3.1 可度量策略代价分析

微观经济学的生产理论中,在劳动力和资本两个生产要素均可变的情况下,等产量线本质上表示了劳动力和资本之间的弹性替代关系,通过成本线约束,可以确定特定成本下使用的劳动力和资本以及能够获得的产量.如果生产过程固定,即所能投入的劳动力和资本比例是固定的,那么通过线性规划,将成本和生产过程的约束施加到生产函数上,则可以获得约束范围内最优的生产要素比例和生产过程.在线性规划中,成本可以是由劳动力成本和资本成本构成的等成本线,也可以是固定劳动力成本线和资本成本线.总之,通过在生产函数上施加生产过程和成本约束,可以找到最优生产决策点,最优决策点是在生产成本限定的情况下,生产要素的最优替代关系和最优生产过程[45,46].

在本文中,策略代价相当于生产产量,代价要素由冲突的非功能需求构成,也就相当于劳动力和资本生产要素,而利益相关者对这些冲突非功能需求优先级的评估数据则构成代价成本约束,相当于生产理论中的成本约束,因此,借鉴生产理论概念,本文的策略代价是冲突非功能需求之间提升和降低的代价关系,当增加利益相关者的非功能需求优先级约束后,可以找到利益相关者满意的权衡代价关系.当然,如果再增加对非功能需求的最低或最高代价约束,可以进一步找到最优权衡代价.下面,我们对此方法进行详细描述.

首先,经过 NRTF 的前 3 个流程步骤,我们获取了满足软件非功能需求的策略集合.此策略集合中的所有策略都是有助于提升软件质量的,但是,其中部分策略推理出矛盾存在,这个矛盾其实反映了因策略实施产生冲突的非功能需求需要权衡,即策略在保证一个非功能需求得到满足的同时,使另一个非功能需求不能满足或者部分不能满足.因此,接下来需要对这些存在矛盾的策略进行进一步的代价权衡,以确定如何具体使用该策略.

3.1.1 策略代价

对于存在矛盾的策略,其矛盾本质上反映了因策略实施将产生非功能需求之间的冲突,需要对策略进行代价分析以获得非功能需求之间的权衡.因此,下面首先定义策略代价:

定义 1(策略代价). 一个策略 S 的代价 C_s 是策略 S 引发冲突的非功能需求集合 N 上的一个二元偏序关系集合 $C_S \subseteq N \times N$,其中,

(1) N 是策略 S 引发冲突的非功能需求集合,$\forall n \in N$ 是一个非功能需求;

(2) C_s 是 N 中冲突非功能需求之间的二元偏序关系的集合,$C_S=\{(f(n_i),f(n_j))|n_i,n_j \in N \wedge n_i$ 与 n_j 冲突$\}$,其中,$f(n)$ 是基于策略 S 对非功能需求 n 的度量,$\forall c_s \in C_S$,是 C_s 的一个代价点.

在用微观经济学的生产理论来描述这个代价时,因策略而产生冲突的非功能需求是策略代价的要素,通过代价要素的度量值关系构造代价线.以图 2 所示为例,n_1 和 n_2 分别代表两个因策略 S 产生冲突而需要权衡的非功能需求,当策略 S 实施,在 n_1 取度量值为 $a_1(f(n_1)=a_1)$时,在 n_2 对应度量值为 $b_1(f(n_2)=b_1)$;在 n_1 取 $a_2(f(n_1)=a_2)$时,在 n_2 为 $b_2(f(n_2)=b_2)$.依此类推,产生了策略 S 的代价线,这个代价线上的各个代价点即是策略 S 的代价 Cs 集合中的各个二元偏序关系元素.代价线上由 $a_i(1 \leqslant i \leqslant m)$和 $b_j(1 \leqslant j \leqslant m)$决定的代价点描述的是因策略 S 实施而在非功能需求 n_1 和 n_2 之间反映出的权衡关系,此权衡关系可以视为 n_1 和 n_2 之间的弹性替代关系.由图 2 可见,如果 n_1 要取最高的 a_4,则 n_2 只能取最低的 b_4.与此类似,当对 n_1 降低要求,取 a_3 时,n_2 就可以达到比 b_4 高的 b_3,即 n_1 和 n_2 之间可以视为此消彼长的替代关系,这个关系描述了非功能需求间相互替代时的代价.

为了构造策略 S 的代价线,需要对策略 S 影响的非功能需求进行度量.由于对非功能需求的度量方法本身存在差异,基于策略度量非功能需求必然会导致得到的度量数据在量纲上存在差异,为了能够将不同量纲的度量数据用于权衡计算,需要对度量数据进行规范化处理.由于我们重点分析度量数据的递增和递减关系,因此采用最小-最大规范化(min-max scaling)方法进行数据规范化处理.规范化函数 $g(x)$如公式(1)所示.

$$x'=g(x)=1+(x-x_{\min})(m-1)/(x_{\max}-x_{\min}) \tag{1}$$

其中,x 是策略影响的非功能需求 n 在不同策略方案上的度量数据($x=f(n)$),x'是规范化后的数据,m 是策略方案数量,$x_{\min}$ 是策略影响的非功能需求在不同策略方案上度量数据的最小值,$x_{\max}$ 是最大值.

以认证策略为例,认证策略有利于增强安全性,但导致时间性能下降.其代价线由口令、摘要和签名 3 种不同认证策略方案所对应的系统速度与安全级别度量值构成,如图 3 所示,横坐标表示的是不同认证方案的安全级别,纵坐标的系统速度使用 3 种认证方案每秒的处理数据量来反映.这两类数据显然具有不同的量纲,因此必须进行数据规范化处理.将数据规范化处理后,基于认证方案安全级别与系统速度之间的代价关系可以确定代价点 $c_{si}(1\leqslant i\leqslant m,m$ 是策略方案数量),由所有代价点 $c_{si}(1\leqslant i\leqslant m)$连线之后,就构成了认证策略的代价线.

当然,认证策略中的认证方案不仅有口令、摘要和签名 3 种方案,还可以有生物特征认证方案,而生物特征认证方案又可以分为指纹认证、脸型认证、虹膜认证等.摘要认证方案也可以按照不同算法分类,而认证方案的要素可以是单要素,也可以是多要素.总之,不同项目需要根据具体项目可选认证方式构建策略代价线.

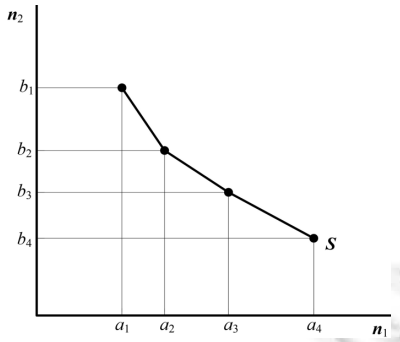


Fig.2 Trade-Off cost of strategies

图 2 策略的权衡代价

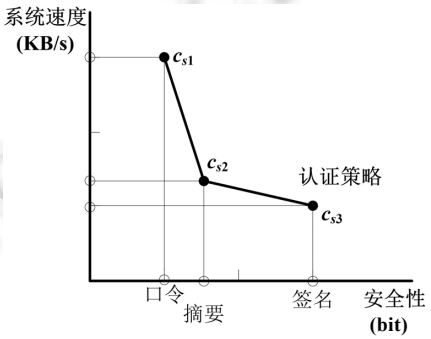


Fig.3 An trade-off cost example of authentication strategy

图 3 认证策略的权衡代价示例

3.1.2 最优权衡代价决策

由图 3 的代价线我们可以看出,如果我们需要较高的安全性,代价就是牺牲系统速度;如果需要保证系统具有较快的响应时间,则代价是不能选择较高安全性的认证方案.要选择利益相关者需要的最优策略代价,就需要对策略代价施加约束.在最优化领域中,生产理论中的线性规划通过对极大化或极小化生产函数施加约束的形式寻找最优生产决策.基于这个思想,我们在寻找最优权衡代价决策时,通过增加利益相关者对非功能需求的优先级约束和基线约束,就可以找到利益相关者需要的最优权衡代价.

如图 4 所示,对策略代价,我们增加两个约束条件.

- 一个是利益相关者对非功能需求的优先级约束 $IN_i/IN_j(1\leqslant i\leqslant m,1\leqslant j\leqslant m)$.此约束是利益相关者对非功能需求优先级的评估数据比值,本质上反映了利益相关者对冲突的非功能需求之间的权衡倾向.如图 4 所示,斜率小的评估比值 IN_1/IN_2 表示相对来说,利益相关者需要较高的非功能需求 n_1 和相对较低的非功能需求 n_2;而斜率大的评估比值 IN_1/IN_2 表示利益相关者需要较高的非功能需求 n_2 和相对较低的非功能需求 n_1.当然,如果斜率为 1,则表明利益相关者认为两个非功能需求同等重要.
- 另一个约束条件是利益相关者对非功能需求设定的最低基线约束 $N_{i\text{-}\min}(1\leqslant i\leqslant m)$和最高基线约束 $N_{i\text{-}\max}(1\leqslant i\leqslant m)$.基线约束是利益相关者设定对非功能需求的最低和最高要求,这个要求通过 NRTF 中软件非功能需求状态证据收集阶段收集的证据生成,证据来源于利益相关者要求、原型分析、软件评估基准(benchmark)、模型仿真、软件早期版本或者其他形式的分析[47].正如文章开篇介绍的 TRW 为政府开发信息检索分析系统的案例[1],4s 就是 90%客户满意的系统响应时间,这是对时间性能的最低基线约束,这个约束就是通过原型分析获得证据后给出的.

借鉴线性规划方法的思想,对于策略 S 所影响的非功能需求 n_i 和 n_j,策略 S 的代价是集合 C_S,规范化后代价为 $C'_S, C'_S=\{(g(f(n_i)),g(f(n_j)))\}$.对于 $\forall(a,b)\in C'_S$,当策略 S 在非功能需求 n_i 取度量值并规范化为 a 时,非功能需

求 n_j 度量规范化值为 $b.IN_i$ 和 IN_j 分别是利益相关者对非功能需求 n_i 和 n_j 的优先级约束,$N_{i\text{-min}}$ 和 $N_{i\text{-max}}$ 分别是 n_i 的最小和最大基线约束.同样,$N_{j\text{-min}}$ 和 $N_{j\text{-max}}$ 分别是 n_j 的最小和最大基线约束.最优权衡决策目标函数和约束如下所示.

- 目标函数

$$OP=\sum_{i,j=1}^{m}|IN_i\cdot a-IN_j\cdot b|(\forall(a,b)\in C'_S)\text{取极小化值} \tag{2}$$

- n_i 基线约束

$$a\leqslant N_{i\text{-max}}\text{ 且 }a\geqslant N_{i\text{-min}} \tag{3}$$

- n_j 基线约束

$$b\leqslant N_{j\text{-max}}\text{ 且 }b\geqslant N_{j\text{-min}} \tag{4}$$

- 非负约束

$$a,b\geqslant 0(\forall(a,b)\in C'_S) \tag{5}$$

基于 n_i 和 n_j 的基线约束以及优先级约束,目标函数取极小值的(a,b)即是优化后的权衡决策.

基于基线约束,我们可以得到最优策略方案的可选区域,如图 4 的阴影区域,再加上基于利益相关者对非功能需求的优先级约束,策略的最优权衡代价是满足基线约束的可选区域内代价线与优先级约束线的交点或者是距离优先级约束线最近的代价点.以图 4 所示为例,$N_{1\text{-max}}$ 是对非功能需求 n_1 的最高基线约束,即策略的代价决策要求 n_1 必须小于 $N_{1\text{-max}}$;而 $N_{2\text{-min}}$ 是对非功能需求 n_2 的最低基线约束,同样,策略的代价决策要求 n_2 必须大于 $N_{2\text{-min}}$.那么,满足此约束的策略可选代价在阴影区域内,在这个区域内,只有 c_{s1},c_{s2} 和 c_{s3} 满足基线约束.此时,如果利益相关者对非功能需求的评估数据比值是$(IN_1/IN_2)_1$,那么 c_{s1} 是最接近优先级约束的代价点,因此,c_{s1} 是在约束$(IN_1/IN_2)_1$ 下的最优权衡代价.与此类似,当利益相关者的优先级约束是$(IN_1/IN_2)_2$ 时,c_{s3} 是最优权衡代价.但是,当利益相关者的评估数据比值是$(IN_1/IN_2)_3$时,c_{s4}是最接近优先级约束的代价点,但是这个代价让非功能需求 n_1 和 n_2 的基线约束均不能得到满足,此时需要确定是否可以调整 n_1 的最高约束基线约束或者 n_2 的最低约束基线约束,如果不能调整,则取 c_{s3} 为次优权衡代价.

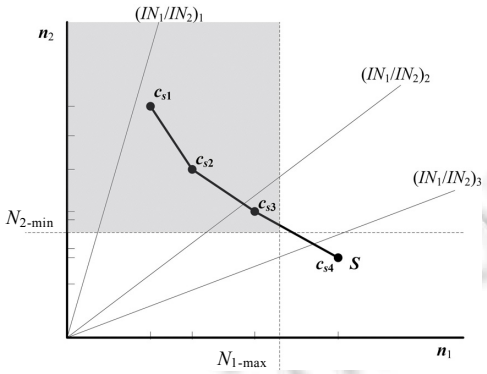


Fig.4 Optimal trade-off cost

图 4 最优权衡代价

在实际软件项目中,部分策略需要在多个冲突的非功能需求之间进行权衡,此时,构造多条策略代价线,将利益相关者的优先级约束和非功能需求的基线约束施加到多条策略代价线上,在一个最优策略权衡区间中选择最优权衡代价.以图 5 所示为例,图 5 的上图是策略 S 在冲突的非功能需求 n_1 和 n_2 之间权衡的代价线,下图是策略 S 在冲突的非功能需求 n_1 和 n_3 之间权衡的代价线,当使用利益相关者优先级约束和非功能需求基线约束后,代价线与优先级约束线交点之间形成的深色区域表示的是在 3 个非功能需求之间的最优权衡区间,代价点 c_{s2} 和 c_{s3} 在此区间中,通过目标函数极小化取值,可以求得它们中的最优代价点.

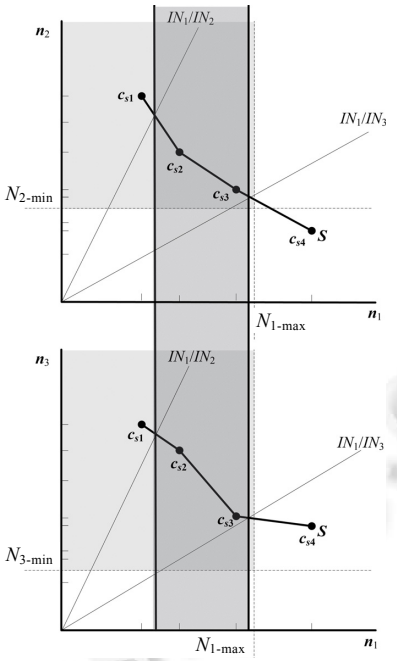


Fig.5 Trade-Off cost of strategies with multi-factors

图 5 多要素策略权衡代价

3.1.3 软件环境变化影响分析

软件开发、维护与演化处于一个不断变化的环境之中,在软件整个生命周期过程中,使用最先进并经过验证的新技术能够对软件质量有促进作用.因此,在进行策略的权衡代价分析时,有必要考虑技术进步带来的代价变化,这个代价变化可以用新的代价线表示,如图 6 所示.

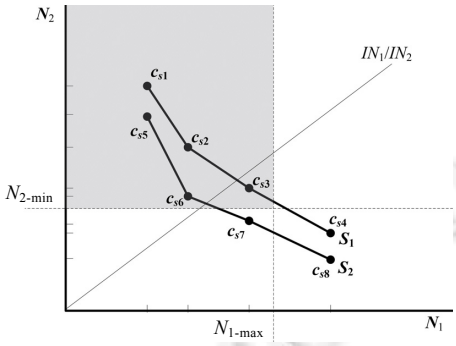


Fig.6 Impact of technological progress

图 6 技术进步的影响

在图 6 中,S_1 和 S_2 分别是策略的原有代价线和考虑技术进步后的新代价线,技术进步意味着同样的策略在技术进步时,非功能需求会受到相应的影响,受到影响后代价线有可能是从 S_1 进步为 S_2,也可能是从 S_2 进步为 S_1.对于其他类似因素的环境变化影响,例如平台变换、项目成员技能水平变化、硬件升级或者引入新的管理体制等,都可以用第 3.1.1 节定义的方法构造变化后的策略代价线以反映动态多变的软件环境对策略代价的影响.

另外,如果非功能需求基线发生变化,例如项目成本缩减,非功能需求 n_1 最大投入减少,此时调整基线约束,策略的可选代价区域随之改变,如图 7 阴影部分所示,原可选策略区域因为基线的调整而缩小为深色区域部分,

浅色区域部分是在原基线范围内的可选区域,变化后的策略可选代价区域已经不包含此部分.同样,非功能需求优先级评估数据也可以随软件环境的变化或者利益相关者对软件需求的变化而变化,此时,增加新的非功能需求优先级约束线.在所有新的约束条件下,使用第 3.1.2 节定义的方法可以获得最优权衡代价决策.

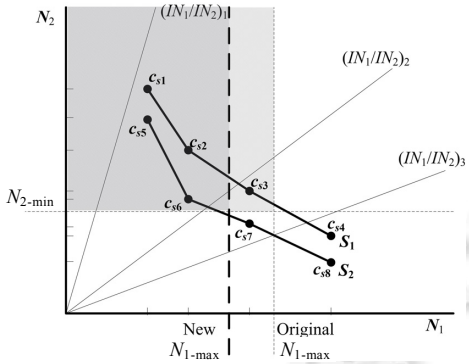


Fig.7 Impact of changed baseline

图 7 基线改变的影响

3.2 非功能需求权衡代价分析工具

有效的支持工具有利于决策者快速获得决策建议.项目组前期使用 Matlab,基于可满足性问题求解开源工具 zChaff[48]开发了 TACD 工具,辅助获取利益相关者的非功能需求评估数据,建模非功能需求与策略关系并完成推理.本文工作作为延续,使用 Matlab 实现非功能需求权衡代价分析.

首先,针对本文提出改进的非功能需求评估数据获取方法对 TACD 工具进行改进,主要改进是引入软件类型参照因素,辅助利益相关者对非功能需求优先级进行评估;并使用 Delphi 方法[42]实现利益相关者的需求协商,以获取利益相关者一致同意的需求评估结果.这两项改进的实现均在 Delphi 方法的协商过程中,通过对利益相关者的输入数据进行检查,并对偏离数据给予提示的方式实现.对于第 1 项改进,在确定软件类型后对输入的评估数据进行检查,根据软件类型与软件非功能需求的对应关系,对偏离数据予以提示;与此类似,第 2 项改进是检查所有利益相关者评估数据之间的偏离程度,如果偏离程度超过预定阈值,则同样予以提示.在获取完所有利益相关者评估数据后,如果有提示数据,按照第 2.1 节 Delphi 方法的实施步骤开展协商过程,在协商达成一致意见后,将评估数据输入非功能需求权衡代价分析工具.非功能需求权衡代价分析工具由策略方案管理组件、代价计算组件和代价分析可视化组件这 3 个核心功能组件构成.图 8 展示了非功能需求权衡代价分析的主要功能.

1) 策略方案管理组件

策略方案是实施策略以满足非功能需求的具体算法、方法或技术等.针对引起非功能需求冲突的策略,每一个方案都有一个权衡代价,我们仍然以第 3.1.1 节图 3 的认证策略为例,在权衡安全性与时间性能时,选择的认证策略方案是口令、摘要和签名,在这 3 个方案中,如果选择安全级别高的方案,则相应要牺牲更多的时间代价.由于不同策略的方案不同,同一个策略的方案又可以有多个,因此,策略方案管理组件通过迭代增加的方式逐渐扩充策略方案并保存到知识库中.首先,我们从知识库中提取已知方案供使用,在使用过程中,如果自定义了新的方案,则增加新方案入库,通过不断使用的过程逐步积累丰富的策略方案,扩充策略库,并根据需要对策略方案进行修改和检验,或者删除不再需要的方案.

2) 代价计算组件

代价计算过程由下述 4 个步骤构成.

步骤 1. 用户选择需权衡策略,输入策略方案数据和基线约束数据,根据方案数据计算代价点位置,并将邻近代价点连线后得到代价线.

步骤 2. 计算代价线与非功能需求优先级约束线(此优先级直接通过已完成的 TACD 工具获取)的交点,

求出每个代价点与优先级约束线的距离,如果是多个非功能需求权衡,则计算多个代价点与优先级约束线的距离累加值,按照距离值由小至大的顺序排列.此时,如果没有设置非功能需求基线约束,则输出距离最近的代价点为最优权衡代价;如果设置了非功能需求的基线约束,则进入步骤 3.

步骤 3. 根据设置的非功能需求基线约束计算可选代价区域,按照距离值由小至大的顺序判断各个代价点是否在可选代价区域内:如果距离最近的代价点在可选代价区域内,则输出此代价点为最优权衡代价;如果距离最近的代价点不在可选代价区域内,则判断距离次近的代价点.如此循环,直至找到既在约束区域内,又与优先级约束线距离近的代价点,输出为最优权衡代价.

步骤 4. 根据各个代价点是否在可选代价区域内以及与优先级约束线的排序距离,输出各个代价点的权衡代价分析数据.

3) 代价分析可视化组件

为了清晰地展示非功能需求权衡代价分析结果,在最终得到的分析图中,如图 8 中右下方图所示:当点击每一个代价点时,将在右侧显示代价分析数据,并且允许拖动非功能需求的基线约束线(图中的虚线),动态地、交互式地反映最优权衡代价选择的变化.

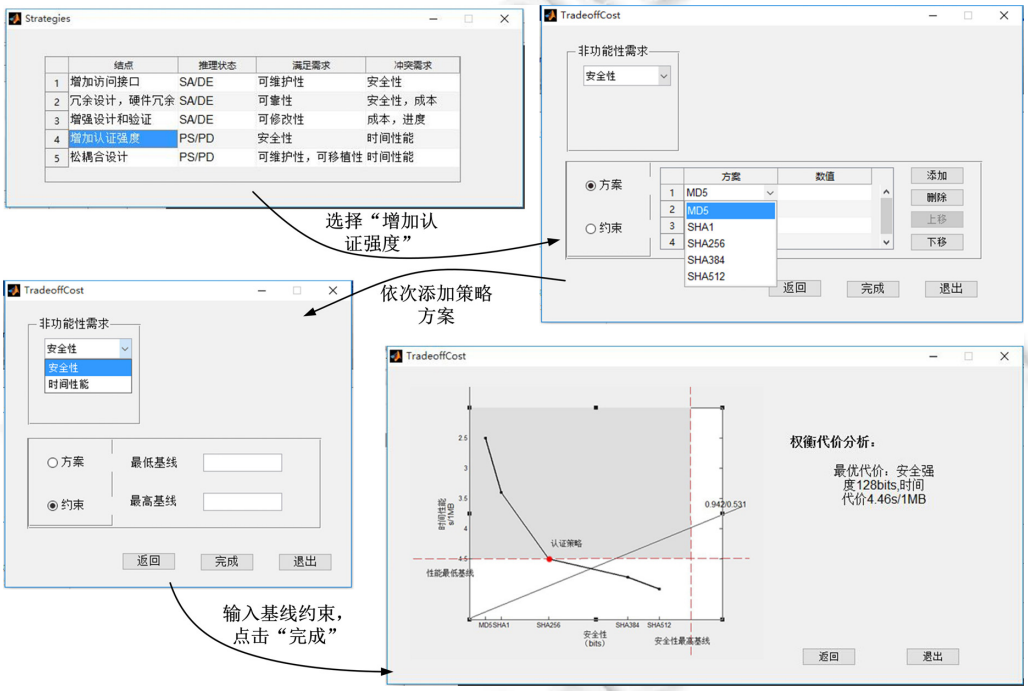


Fig.8 Tool for trade-off cost analysis of non-functional requirements

图 8 非功能需求权衡代价分析工具

图 8 展示了非功能需求权衡代价分析的主要功能:首先,选择前期推理出存在矛盾的策略,输入策略权衡方案数据和对应非功能需求基线约束后,生成代价分析结果.点击其中的每一个代价点,就可以在右侧看到权衡代价分析的结果.

4 案例分析

本文仍然以我们参与的可信第三方认证中心软件项目为例,在项目组前期研究[4]中,已经完成的工作包括:构建可信软件非功能需求与过程策略框架 TRPF,从利益相关者获取软件非功能需求优先级评估数据并进行数

据有效性检验,对非功能需求与过程策略的关系进行建模,通过利益相关者的评估数据,基于关系模型,推理找出满足非功能需求的过程策略,并对其中存在矛盾的策略提供决策建议.

在本文中,我们对原有框架进行改进和扩展,继续在可信第三方认证中心项目中进行实践与分析,为可信第三方认证中心软件的维护与演化需求提供决策依据,具体完成工作包括:引入软件所属类型参考因素,使用 Delphi 方法[42],基于协商的方式向利益相关者迭代获取非功能需求优先级评估数据;对推理存在矛盾的策略进行权衡代价分析,为维护及演化决策提供依据.

4.1 软件非功能需求获取

首先确定利益相关者,由于前期研究选取的利益相关者较好地代表了项目的涉众,而且为了与前期研究结果形成对比关系,我们仍然选取认证中心相关 5 名利益相关者(S_1:认证中心监管部门;S_2:工程方;S_3:认证中心专家;S_4:软件操作部门;S_5:证书持有者)对认证中心软件进行非功能需求优先级的评估.在评估过程中,我们增加了软件所属类型的参照数据,在第 1 轮评估数据收集后,通过熵运算,仍然是证书持有者 S_5 的熵数据相对偏差较大,因此使用 Delphi 方法,将此不一致结果和详细数据分析情况反馈给所有利益相关者,请利益相关者再次提交评估数据.在再次提交的评估数据中,证书持有者 S_5 反馈可移植性需求需要提升,对于不直接与安全服务器连接的客户端软件,例如提交申请数据的客户端,应提供多平台支持,通过将此反馈意见再次提交给所有利益相关者后,最终获得了统一的非功能需求优先级评估数据.

由于此次收集数据时利益相关者已经使用了按照前期研究成果开发的认证中心软件,经过使用,他们对软件的维护及演化需求与原需求相比,发生了如表 3 所示的变化.

Table 3 Evaluation data changes for non-functional requirements

表 3 非功能需求评估数据变化

非功能需求	原模糊排序值	新模糊排序值
n_{32}:容错性	0.815	↓0.085
n_6:可维护性	0.834	0.915↑
n_{61}:可测试性	0.408	0.898↑
n_{62}:可重用性	0.469	↓0.173
n_{63}:可修改性	0.408	0.915↑
n_{65}:可扩展性	0.746	↓0.200
n_{72}:空间性能	0.531	0.746↑
n_{91}:可生存性	0.684	0.898↑
n_{10}:可移植性	0.200	0.684↑

其中,容错性、可重用性和可扩展性需求降低,原因是容错带来较大的风险.另外,认证中心软件没有明确的可重用性需求,也无扩展需求.与这 3 个需求相反,对可维护性、可测试性、可修改性、空间性能、可生存性和可移植性的需求都有或多或少的提升.主要原因有两个.

- 一个原因是,软件投入使用后对其可维护性有了更大的需求,当然,原可维护性需求也相对较高,而本次新获取的数据又进一步反映了可维护性需求的提升,相应地,也提升了与维护相关的可测试性、可修改性、空间性能和可生存性需求.
- 另一个原因是,客户端软件的多平台需求提升了对软件的可移植性需求,由于认证中心软件的安全原因,已投入使用软件所基于的平台选择了 Linux 系统,但经过使用,对与安全服务器没有直接连接的客户端,例如提交申请数据的客户端,用户提出了多平台使用的需求,相应地提升了软件可移植性的需求.

4.2 非功能需求权衡代价分析

确定认证中心软件的非功能需求后,扩展策略选择范围,重新基于 TRMM 建模非功能需求与策略关系,并进行推理.由于建模及推理过程与原方法[4]相同,下面我们仅给出推理存在矛盾的策略结果,见表 4.对于这些存在矛盾的策略,我们在下一步对其进行权衡代价分析.

Table 4 Statuses of nodes

表 4 节点状态

节点类型	节点	推理状态	满足需求	冲突需求
策略	增加访问接口	SA/DE	可维护性	安全性
	冗余设计,硬件冗余	SA/DE	可靠性	安全性,成本
	增强设计和验证	SA/DE	可修改性	成本,进度
	增加认证强度	PS/PD	安全性	时间性能
	松耦合设计	PS/PD	可维护性,可移植性	时间性能

在表 4 所示的存在矛盾的策略中,冗余设计和硬件冗余导致的成本代价可以用 COCOMO II 模型[41,44]计算.同样,增强设计和验证导致的成本和进度代价也是用 COCOMO II 模型计算.因此,下面我们仅对其余需权衡代价的策略进行分析.

1) 增加访问接口

增加访问接口策略的使用需要在可维护性和安全性之间进行权衡,因为每增加一个访问接口就可以方便维护工作的介入,提升一定的可维护性,但同时增大了软件的攻击面[49],导致软件安全性降低.因此,增加访问接口策略的代价为 $C_S=\{(f(n_6),f(n_4))\}$,其中,S 是增加访问接口策略,n_6 是可维护性需求,$f(n_6)$是基于策略 S 的可维护性度量数据,n_4 是安全性需求,$f(n_4)$是基于策略 S 的安全性度量数据.

对于增加访问接口影响的安全性,我们使用 Manadhata 和 Wing 的攻击面[49]度量.攻击面定义为攻击者能够进入软件系统并且造成破坏的资源的集合,由于攻击面通过对软件本身的研究能够描述软件被破坏的潜在可能性和实施攻击所要付诸努力的程度,因此,通过攻击面的度量可以确定软件被攻击者利用其资源的可能性,若攻击面大,则表明软件被攻击者利用其资源的可能性大,从而表明软件不安全的可能性大.在使用增加访问接口策略时,增加访问接口就增加了软件的攻击面,使软件安全性降低,因此,分析增加访问接口策略代价时,安全性用攻击面度量定义为

$$f(n_4)=\sum_{m\in M} der_m(m)+\sum_{c\in C} der_c(c)+\sum_{d\in D} der_d(d) \tag{6}$$

其中,$der_i(i)=count(i)\times q_i/e_i(i=m,c,d)$分别是攻击面度量资源——函数 *method*、通道 *channel* 和数据项 *data item*;$count()$是资源数量;q_i 是资源的潜在破坏能力指标,资源特权越高,其破坏能力就越大;e_i 是利用攻击面资源 i 实施攻击需要付出的努力程度指标,这个努力程度依赖于资源的访问权限,如果攻击者需要使用资源实施威胁更大的攻击,就需要获得这些资源更高的访问权限,当然就需要付出更大的努力,因此,q_i/e_i 体现了资源的破坏力及努力比例关系.基于攻击面的软件安全度量适用于同一个软件的不同版本或者有相似功能的不同软件的安全比较,非常适合本项目用于对增加访问接口策略的分析.

另外,在使用增加访问接口策略时,为了提升可维护性,希望增加的接口主要是提供远程访问文件和设备的接口以及对文件和字符处理的监控接口.按照攻击面计算时使用的函数、通道和数据资源进行分类,需要增加的接口可以归类为网络函数、文件处理函数和设备管理函数、网络协议通道、文件、字符及 URL 数据.表 5 给出了增加访问接口策略的细节数据.

Table 5 Adding access interface strategy

表 5 增加访问接口策略

资源类型	资源	数量	破坏及努力率
函数	权限为 root,访问类型为 authenticated	入口 1~4,出口 1~2	5/3
	权限和访问类型为 authenticated	入口 2~8,出口 1~4	3/3
通道	协议类型为 SSL,访问类型为 remote unauthenticated	1	1/1
	协议类型为 socket,访问类型为 local authenticated	1	1/4
数据	类型为文件,访问类型为 root	3~12	1/5
	类型为文件,访问类型为 authenticated	1	1/3

根据对资源的合理安排,增加访问接口策略可以设计 4 个可维护性逐渐递增的方案,见表 6.

Table 6 Four selections of adding access interface strategy

表 6 增加访问接口策略的 4 个方案

	增加接口方案	维护性	攻击面
①	增加 root 函数入口 1 个,出口 1 个;增加 authenticated 函数入口 2 个,出口 1 个;增加 SSl 协议和 socket 协议,以 root 权限增加文件访问数据项 3 项,以 authenticated 权限增加文件访问数据项 1 项.	1	8.52
②	增加 root 函数入口 2 个,出口 1 个;增加 authenticated 函数入口 4 个,出口 2 个;增加 SSl 协议和 socket 协议,以 root 权限增加文件访问数据项 6 项,以 authenticated 权限增加文件访问数据项 1 项.	2	13.78
③	增加 root 函数入口 3 个,出口 2 个;增加 authenticated 函数入口 6 个,出口 2 个;增加 SSl 协议和 socket 协议,以 root 权限增加文件访问数据项 9 项,以 authenticated 权限增加文件访问数据项 1 项.	3	19.71
④	增加 root 函数入口 4 个,出口 2 个;增加 authenticated 函数入口 8 个,出口 2 个;增加 SSl 协议和 socket 协议,以 root 权限增加文件访问数据项 12 项,以 authenticated 权限增加文件访问数据项 1 项.	4	23.98

通过对这 4 个方案进行相关可维护性和安全性度量,我们得到如图 9 所示的权衡代价分析.

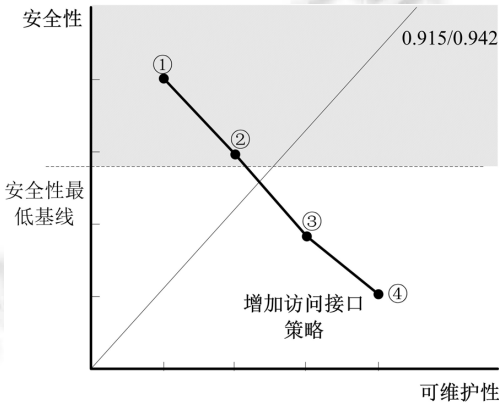


Fig.9 Trade-Off cost of adding access interface strategy

图 9 增加访问接口策略权衡代价

在此我们假设:如果增加资源多于第 4 个方案,则安全性的降低无法估计,也不可接受.因此,策略代价的计算根据攻击面增大的数值计算安全性下降的比例.使用公式(6)计算攻击面后,使用公式(1)进行规范化处理,得到方案 1 的安全性度量规范化值:x'=1+(8.52−23.98)3/(8.52−23.98)=4,方案 2 的安全性度量规范化值为 x'=1+(13.78−23.98)3/(8.52−23.98)=2.98,其他方案的规范化值计算类似.另外,策略的每一个方案提升可维护性的度量值从 1 到 4,以等量递增方式计算.最终,规范化后的策略代价 C'_S={(1,4),(2,2.98),(3,1.83),(4,1)},对应于此代价数值,分别得到①~④代价点.另外,可维护性与安全性的优先级约束 0.915/0.942 来源于从利益相关者获取的优先级评估数据,安全性最低基线不允许安全性下降超过 30%(此比例仅考虑增加访问接口策略 4 个方案的相对比例,不是整体软件安全性的比例),可维护性最高基线允许尽量增加可访问的接口,即没有可维护性的基线约束.通过可维护性与安全性的优先级约束和安全性最低基线约束后,阴影部分是可选代价区域,优先级约束线与代价线的交点是最优权衡代价位置,通过将 C'_S 各个代价值输入最优权衡决策目标函数(2)和约束公式(3)~约束公式(5),计算得到距离优先级约束线最近的代价点②是最优权衡代价方案.方案②保证从一定程度上支持可信认证中心软件的维护工作,提升了一定的可维护性,同时也保证了软件较高的安全性需求.

2) 冗余设计和硬件冗余

冗余设计和硬件冗余是提升软件可靠性的策略,与增加访问接口类似,增加的冗余空间和硬件都相应增大了软件的攻击面.同样,用上述增加访问接口策略的权衡代价分析方法可以进行类似的分析.在此,为使描述简

洁,我们不对此策略的权衡代价分析过程进行详细描述.

3) 增加认证强度

增加认证强度策略是提高软件安全性的策略,但是这个策略可能需要牺牲系统的时间性能作为代价.表 7 是不同哈希算法的运行时间和安全强度,测试这些算法运行时间的平台采用了与目前项目相同的奔腾 4 平台,算法的安全强度参照哈希算法碰撞强度确定.

Table 7 Time performance of Hash algorithms

表 7 哈希算法的时间性能

哈希算法	时间(KB/s)	规范化	安全(bit)	规范化
MD5	412	5	<64	1
SHA1	293	2.63	<80	1.4
SHA256	224	1.26	128	2.6
SHA384	219	1.16	192	4.2
SHA512	211	1	224	5

图 10 给出了认证策略的权衡代价分析.同样,安全性与时间性能的优先级约束 0.942/0.531 来源于利益相关者的优先级评估数据,性能最低基线约束设定为每秒处理数据不能少于 1KB,而安全性不设定基线约束,在可选代价阴影区域内,离优先级约束线距离最近的是 SHA256,因此,SHA256 是在目前约束条件下的最优策略.

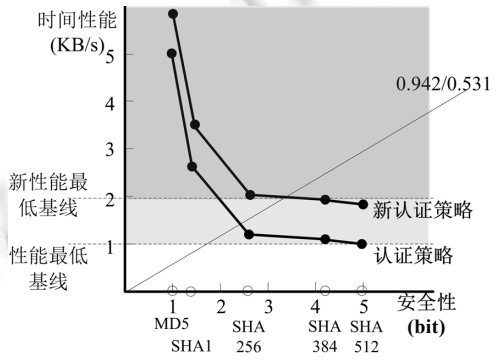


Fig.10 Trade-Off cost of authentication strategy

图 10 认证策略权衡代价

另外,如果对认证策略进一步分析其代价,可以体现本文方法的两个重要贡献.

(1) 提供详细权衡过程,有利于获得最优决策.

例如,通过图 10 我们可以看到,SHA256,SHA384 和 SHA512 的时间性能代价分别是 1.26,1.16 和 1,而其安全强度有明显的增强.此时,如果时间性能下降是可接受的,则选择 SHA384 或者 SHA512,可以获得更高的安全性.

(2) 权衡过程能够适应软件环境的变化,并且能随变化调整最优决策.

例如,如果可信认证中心的软件环境发生变化,假设硬件平台升级或者算法优化,认证策略算法的运行效率会提升,那么代价线也会相应移动,按照前面第 3.1.3 节的分析,参考第 3.1.3 节的图 6,目前的代价线将会上升,从认证策略代价线上升为新认证策略代价线.此时,更高安全性的 SHA384 将成为最优权衡代价的策略方案,使用 SHA384 后,可以更大程度地提升安全性,同时保证时间性能代价是可接受的.而如果性能最低基线也发生变化,从性能最低基线上升为新性能最低基线,那么最优策略方案可选区域将缩小为深色区域部分.此时,对于认证策略的代价线,在可选区域内的 SHA1 方案是最优权衡代价方案;而对于新认证策略的代价线,在可选区域内 SHA384 方案是最优权衡代价方案.

4) 松耦合设计

为了实现客户端可移植性,增强软件可维护性,项目组拟对移植客户端进行松耦合设计,但是松耦合设计在

提升可移植性及可维护性时需要牺牲的代价是时间性能.对于耦合度的计算,我们采用描述软件代码线性依赖关系的循环复杂度(cyclomatic complexity number)[50]度量耦合度,通过与上述认证策略类似的方式进行策略权衡代价分析.为使描述简洁,在此不详细展开此策略的代价分析.

根据可信第三方认证中心软件的非功能需求推理,我们获得存在矛盾的上述策略.通过权衡分析,对于与成本相关的策略,通过 Boehm 的 COCOMO II 成本估算模型[41,44]可估算成本代价和进度代价.对于其他可度量策略,使用本文提出的非功能需求权衡代价分析方法可以找出最优权衡代价,可以有效地支持最优决策的制定.

4.3 非功能需求权衡分析方法有效性

通过将本文方法在实际软件项目案例中的使用,通过与项目组在使用过程中以及使用完成后的沟通,我们从优势与存在问题两个方面对本文方法的有效性进行如下分析和总结.

1) 优势

首先,在软件非功能需求获取阶段引入软件类型参照因素,辅助利益相关者对非功能需求重要程度进行评估取得了一定的效果,通过利益相关者反馈,基于软件客观所属类型与软件非功能需求之间存在的内在关系,可以辅助利益相关者在项目早期需求工程阶段提高对部分非功能需求的重视,这部分非功能需求有可能是项目组在早期无法预见其重要性的.另外,通过引入 Delphi 方法,可以有效地促进利益相关者之间的协商沟通,不仅可以让利益相关者保持各自对非功能需求优先级的评估意见,而且可以使不同利益相关者在非功能需求重要程度的评估上取得一定程度的相互认同,有利于在非功能需求权衡决策时达成一致.

其次,在非功能需求权衡分析中,与其他相关研究工作相比,基于微观经济学的生产理论和替代弹性原理描述非功能需求间的权衡关系并使用线性规划方法实现决策优化,经项目组反馈,认为该方法具有实际可操作性,具体有如下的优势.

- 通过对策略所影响的非功能需求进行度量,构造策略代价线,能够清晰地展示策略多方案间的代价关系,同时支持策略方案的调整,包括增加或减少策略方案以及展示不同软件环境下策略方案的变化.
- 通过将利益相关者对非功能需求的优先级约束施加到策略代价线上,有效地引入了利益相关者对软件质量的期望需求;同时,通过采集证据,将非功能需求的基线约束施加到策略决策分析中,可以保证非功能需求权衡决策的有效性.
- 在软件全生命周期过程中,变化是不可避免的.本文提出的方法能够适应变化,展示不同软件环境下的策略代价和非功能需求约束条件,不仅支持权衡决策能适应环境变化,而且可以支持不同变化阶段权衡分析结果的比较,使利益相关者了解变化前的权衡结果,也可以依据可预计变化做出最优权衡决策.

2) 存在问题

在使用本文提出的软件非功能需求权衡分析方法时,需要为权衡分析收集一定的项目数据后才能开展分析工作,这些数据包括利益相关者对软件非功能需求优先级的评估数据、非功能需求状态证据和非功能需求度量数据.因此,权衡分析工作无法在项目初期快速提供分析结果,分析结果需要在项目数据的逐步收集过程中产生.然而,在项目早期无项目数据支持而获得的分析结果有可能在后期实施中出现与项目矛盾的情况,从而带来负面影响.由于本文方法具有一定的变化适应性,权衡分析可以逐步逼近最优决策,以获取符合项目实际需求的决策结果.

另外,构建本体知识库的最终目标是重用领域知识,实现对软件非功能需求建模、推理及权衡分析的辅助,并且期望更进一步地实现自动化,提高工作效率.但如何高效重用领域知识,如何快速辅助建模、推理和权衡分析等等,仍然是具有挑战性的研究工作.

5 结论与展望

面向软件非功能需求,我们深入分析非功能需求之间的冲突关系,找到冲突的根源本质是来源于实现非功能需求的策略.基于冲突根源-策略,本文解决存在冲突的非功能需求的两个重要问题:一是如何选择策略的最优权衡方案,实现冲突非功能需求之间的最佳权衡;二是如何保证最优策略决策能够适应不断改变的软件环境,

不仅支持软件开发,而且支持软件维护和演化.通过借鉴微观经济学中的生产理论、生产要素弹性替代原理和线性规划方法,我们将非功能需求视为策略的代价要素,非功能需求之间的权衡代价关系变化就形成了策略的代价.通过利益相关者提供对非功能需求优先级的评估数据约束,采集非功能需求的最低和最高需求证据约束,借鉴线性规划方法,找出策略的最优权衡代价.当软件环境发生变化时,策略代价可随之变化调整,利益相关者对非功能需求的变化也可以随之变化调整,并最终能够适应性地提供最优权衡代价.

本文改进并扩展了项目组前期的工作成果,具体包括:

1) 在获取非功能需求最优权衡代价时,利益相关者对非功能需求优先级的评估数据是策略代价最优化的重要约束,为了能够获取全体利益相关者一致同意的、正确的非功能需求,我们对前期工作的方法进行改进,引入软件类型参考因素,辅助利益相关者提供评估数据.另外,使用 Delphi 方法[42],迭代交互式地向利益相关者收集评估数据.通过这些改进,避免了少数利益相关者的正确评估数据被忽略或删除,这在本文案例中得到了充分的验证.
2) 跨学科、跨领域的软件定义、开发、维护与演化是当今软件工程师和学者面临的普遍现象,在这个普遍现象之中,不同利益相关者之间常常因为大家的专业背景和所处的环境差异而难以协作沟通.为了适应并解决这个问题,我们借鉴 Boehm 的质量本体概念[36]改进原知识库,构建非功能需求本体知识库,迭代存储与非功能需求密切相关的类、个体、参照、状态、过程和关系知识,促进不同领域、学科利益相关者之间的高效协作.
3) 重点解决冲突策略的权衡代价问题,通过提供最优权衡代价分析方法并开发分析支持工具,为冲突策略的决策提供重要依据.

与其他同类相关研究工作相比,本文的贡献如下:

1) 细化策略代价分析过程.其他同类研究工作通过非功能需求建模与推理,大部分是找出满足非功能需求的策略集合,或者进一步通过度量非功能需求,按照利益相关者的需要,给予多个策略选一的决策建议,由于没有权衡分析细节,决策结果不可追溯,也无法让决策者掌握权衡代价的细节.本文通过更进一步分析权衡过程,详细展示策略影响的非功能需求权衡代价分析细节,不仅可以让决策者追溯决策结果,并且在提供详细权衡代价分析过程中让决策者掌握权衡代价细节,有利于决策者根据具体软件情况选择或调整最优权衡代价,而且可以支持决策者在进一步提高软件质量过程中掌握权衡代价的依据.
2) 适应软件环境的变化.软件开发及演化处于一个动态多变的环境之中,硬件不断升级、软件技术不断优化、项目成员经常变动等等,都对软件开发、维护及演化工作带来了极大的挑战.本文对非功能需求权衡代价的分析能够应对这一挑战,可以适应性地根据环境变化来调整策略的代价和约束,反映不同软件环境下策略的代价变化.

通过对可信第三方认证中心软件项目的案例分析,本文提出的非功能需求权衡代价分析方法可以有效地支持最优决策过程.今后,针对本文的方法和开发的支持工具,还需进一步开展 3 个方面的研究工作.

1) 通过案例分析,目前开发的非功能需求权衡代价分析工具需要扩展和集成.首先,目前的工具仅能基于软件类型辅助利益相关者对非功能需求进行优先级的评估,下一步将基于已有研究成果和实证研究成果增加软件规模、复杂度、成本和进度因素的参照影响分析;其次,为了支持不断扩充的策略方案,进一步扩大应用范围,需要移植开发平台;最后,需要将不同策略方案的支持工具集成到 TACD 工具中,并进一步增强可视化的支持能力.
2) 软件环境的变化为非功能需求带来了一些新的特征.例如,随着软件逐步向移动平台移植,移动平台对能源消耗提出的节能新需求是原来非移动平台所不考虑的.在此领域,Qian 等人开展了相关研究工作[51−54].我们下一步也将对这一类特殊的新兴发展领域进行研究,以全面适应软件环境的不断变化.
3) 重用知识可以辅助软件工程师完成非功能需求建模、推理和权衡分析工作,并提高工作效率,甚至实现部分自动化.我们下一步将针对领域知识重用进行研究,增加本文方法权衡分析的效率.

References:

[1] Boehm B. Unifying software and systems engineering. IEEE Computer, 2000,33(3):114−116. [doi: 10.1109/2.825714]

[2] Wang HM, Liu XD, Lang B, Xie B, Mao XG. Software trustworthiness classification specification (TRUSTIE-STC v2.0). Technical Specification, School of Computer Science and Engineering, BeiHang University, *et al*., 2009 (in Chinese).

[3] Amoroso E, Taylor C, Watson J, Weiss J. A process-oriented methodology for assessing and improving software trustworthiness. In: Proc. of the ACM Conf. on Computer and Communications Security (CCS). 1994. 39−50. [doi: 10.1145/191177.191188]

[4] Zhang X, Li T, Wang X, Yu Q, Yu Y, Zhu R. Formal analysis to non-functional requirement of trustworthy software. Ruan Jian Xue Bao/Journal of Software, 2015,26(10):2545−2566 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4813.htm [doi: 10.13328/j.cnki.jos.004813]

[5] Boehm B, In H. Identifying quality-requirement conflicts. Software, 1996,13(2):25−35. [doi: 10.1109/52.506460]

[6] Dardenne A, van Lamsweerde A, Fickas S. Goal-Directed requirements acquisition. Science of Computer Programming, 1993, 20(1,2):3−50. [doi: 10.1016/0167-6423(93)90021-G]

[7] van Lamsweerde A, Darimont R, Letier E. Managing conflicts in goal-driven requirements engineering. IEEE Trans. on Software Engingeering, 1998,24(1):908−926. [doi: 10.1109/32.730542]

[8] Mylopoulos J, Chung L, Nixon B. Representing and using nonfunctional requirements: A process-oriented approach. IEEE Trans. on Software Engineering, 1992,18(6):483−497. [doi: 10.1109/32.142871]

[9] Chung L, Nixon BA. Dealing with non-functional requirements: Three experimental studies of a process-oriented approach. In: Proc. of the 17th Int'l Conf. on Software Engineering (ICSE). New York: ACM Press, 1995. 25−25. [doi: 10.1145/225014.225017]

[10] Chung L, Nixon BA, Yu E, Mylopoulos J. Non-Functional Requirements in Software Engineering. New York: Springer Science+ Business Media, 2000. [doi: 10.1007/978-1-4615-5269-7]

[11] Chung L, do Prado Leite JCS. On non-functional requirements in software engineering. In: Proc. of the Conceptual Modeling: Foundations and Applications. Berlin, Heidelberg: Springer-Verlag, 2009. 363−379. [doi: 10.1007/978-3-642-02463-4_19]

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

[13] Castro J, Kolp M, Mylopoulos J. Towards requirements-driven information systems engineering: The Tropos project. Information Systems, 2002,27(6):365−389. [doi: 10.1016/S0306-4379(02)00012-1]

[14] Amyot D, Mussbacher G. URN: Towards a new standard for the visual description of requirements. In: Proc. of the Telecommunications and Beyond: The Broader Applicability of SDL and MSC. Berlin, Heidelberg: Springer-Verlag, 2003. 21−37. [doi: 10.1007/3-540-36573-7_2]

[15] Amyot D, Ghanavati S, Horkoff J, Mussbacher G, Peyton L, Yu E. Evaluating goal models within the goal-oriented requirement language. Int'l Journal of Intelligent Systems, 2010,25:841−877. [doi: 10.1002/int.20433]

[16] Burgess C, Krishna A, Jiang L. Towards optimising non-functionalrequirements. In: Proc. of the Int'l Conf. on Quality Software. 2009. 269−277. [doi: 10.1109/QSIC.2009.42]

[17] Liaskos S, McIlraith SA, Sohrabi S, Mylopoulos J. Representing and reasoning about preferences in requirements engineering. In: Best Papers of RE'10: Requirements Engineering in a Multi-Faceted World. 2011. 227−249. [doi: 10.1007/s00766-011-0129-9]

[18] Wei B, Jin Z, Zowghi D, Yin B. Automated reasoning with goal tree models for software quality requirements. In: Proc. of 2012 IEEE the 36th Int'l Conf. on Computer Software and Applications Workshops (COMPSACW). 2012. 373−378. [doi: 10.1109/COMPSACW.2012.73]

[19] Sebastiani R, Giorgini P, Mylopoulos J. Simple and minimum-cost satisfiability for goal models. In: Proc. of the Advanced Information Systems Engineering. Berlin, Heidelberg: Springer-Verlag, 2004. 20−35. [doi: 10.1007/978-3-540-25975-6_4]

[20] Giorgini P, Mylopoulos J, Nicchiarelli E, Sebastiani R. Formal reasoning techniques for goal models. LNCS Journal on Data Semantics, 2003,2800:1−20. [doi: 10.1007/978-3-540-39733-5_1]

[21] Giorgini P, Mylopoulos J, Sebastiani R. Goal-Oriented requirements analysis and reasoning in the tropos mehtodology. Engineering Applications of Artificial Intelligence, 2005,18(2):159−171. [doi: 10.1016/j.engappai.2004.11.017]

[22] Horkoff J, Yu E. Finding solutions in goal models: an interactive backward reasoning approach. In: Proc. of the Conceptual Modeling (ER 2010). Berlin, Heidelberg: Springer-Verlag, 2010. 59−75. [doi: 10.1007/978-3-642-16373-9_5]

[23] Horkoff J. Iterative, interactive analysis of agent-goal models for early requirements engineering [Ph.D. Thesis]. University of Toronto, 2012.

[24] Boehm B, In H. Software cost option strategy tool (S-COST). In: Proc. of the 12th Annual Int'l Computer Software and Application Conf. (COMPSAC'96). Seoul: IEEE Computer Society Press, 1996. 15−20. [doi: 10.1109/CMPSAC.1996.542289]

[25] Letier E, van Lamsweerde A. Reasoning about partial goal satisfaction for requirements and design engineering. In: Proc. of the ACM SIGSOFT Software Engineering Notes. New York, 2004. 53−62. [doi: 10.1145/1041685.1029905]

[26] Van Lamsweerde A. Reasoning about alternative requirements options. In: Proc. of the Conceptual Modeling: Foundations and Applications. Berlin: Springer-Verlag, 2009. 380−397. [doi: 10.1007/978-3-642-02463-4_20]

[27] Van Lamsweerde A. Requirements Engineering: From System Goals to UML Models to Software Specifications. Hoboken: John Wiley & Sons, 2009.

[28] Heaven W, Letier E. Simulating and optimizing design decisions in quantitative goal models. In: Proc. of the 19th IEEE Int'l Requirements Engineering Conf. Trento, 2011. 79−88. [doi: 10.1109/RE.2011.6051653]

[29] Ma WT, Liu L, Ye XJ, Wang JM, Mylopoulos J. Requirements-Driven internetware services evaluation. In: Proc. of the 1st Asia-Pacific Symp. on Internetware. 2009. 1−7. [doi: 10.1145/1640206.1640212]

[30] Marew T, Lee JS, Bae DH. Tactics based approach for integrating non-functional requirements in object-oriented analysis and design. Journal of Systems and Software, 2009,82(10):1642−1656. [doi: 10.1016/j.jss.2009.03.032]

[31] Zhu MX, Luo XX, Chen XH, Wu DD. A non-functional requirements tradeoff model in trustworthy software. Information Science, 2012,191(5):61−75. [doi: 10.1016/j.ins.2011.07.046]

[32] Elahi G, Yu E. Comparing alternatives for analyzing requirements trade-offs—In the absence of numerical data. Information and Software Technology, 2012,54(6):517−530. [doi: 10.1016/j.infsof.2011.10.007]

[33] Yin B, Jin Z, Zhang W, Zhao HY, Wei B. Finding optimal solution for satisficing non-functional requirements via 0-1 programming. In: Proc. of the COMPSAC 2013. 2013. 415−424. [doi: 10.1109/COMPSAC.2013.69]

[34] Wei B, Jin Z, Zowghi D, Yin B. Implementation decision making for internetware driven by quality requirements. Science China Information Sciences, 2014,57(7):1−19. [doi: 10.1007/s11432-014-5117-5]

[35] Asadi M, Soltani S, Gasevic D, Hatala M, Bagheri E. Toward automated feature model configuration with optimizing non-functional requirements. Information and Software Technology, 2014,56(9):1144−1165. [doi: 10.1016/j.infsof.2014.03.005]

[36] Boehm B, Kukreja N. An initial ontology for system quality attributes. In: Proc. of the Int'l Council on System Engineering (INCOSE). Seattle, 2015. 13−16. [doi: 10.1002/j.2334-5837.2015.00067.x]

[37] Jin Z, Liu L, Jin Y. Software Requirements Engineering: Principles and Methods. Beijing: Science Press, 2008 (in Chinese).

[38] Mairiza D, Zowghi D. Constructing a catalogue of conflicts among non-functional requirements. In: Proc. of the Evaluation of Novel Approaches to Software Engineering, Communications in Computer and Information Science, Vol.230. Springer-Verlag, 2011. 31−44. [doi: 10.1007/978-3-642-23391-3_3]

[39] Denne M, Cleland-Huang J, Software by Numbers. Prentice Hall, Inc., 2003.

[40] Tao HW. Research on the measurement models of software trustworthiness based on attributes [Ph.D. Thesis]. Shanghai: East China Normal University, 2011 (in Chinese with English abstract).

[41] Boehm B, Abts C, Brown A, Chulani S, Clark B, Horowitz E, Madachy R, Reifer D, Steece B. Software Cost Estimation with COCOMO II. Prentice Hall PTR, 2000.

[42] Dalkey N, Helmer O. An experimental application of the Delphi method to the use of experts. Management Science, 1963,9(3): 458−467. [doi: 10.1287/mnsc.9.3.458]

[43] Gruber T. A translation approach to portable ontology specifications. Knowledge Acquisition, 1993,5(2):199−220. [doi: 10.1006/knac.1993.1008]

[44] Boehm B. Software cost estimation tools—COCOMO II. 2000. http://greenbay.usc.edu/csci577/fall2015/tools

[45] Salvatore D. Microeconomics: Theory and Applications. 5th ed., New York: Oxford University Press, 2008.

[46] Sher W, Pinola R. Microeconomic Theory: A Synthesis of Classical Theory and the Modern Approach. New York: Elsevier North Holland, 1981.

[47] Boehm, B. The Incremental Commitment Spiral Model. Addison-Wesley Publishing Company, 2014.

[48] Princeton Univerisity. zChaff 2007.3.12. 2007. http://www.princeton.edu/~chaff/zchaff.html

[49] Manadhata PK, Wing JM. An attack surface metric. IEEE Trans. on Software Engineering, 2011,37(3):371−386. [doi: 10.1109/TSE.2010.60]

[50] McCabe TJ. A complexity measure. IEEE Trans. on Software Engineering, 1976,2(4):308−320. [doi: 10.1109/TSE.1976.233837]

[51] Qian H, Andresen D. An energy-saving task scheduler for mobile devices. In: Proc. of the 14th IEEE/ACIS Int'l Conf. on Computer and Information Science (ICIS 2015). Las Vegas, 2015. [doi: 10.1109/ICIS.2015.7166631]

[52] Qian H, Andresen D. Jade: Reducing energy consumption of android app. Int'l Journal of Networked and Distributed Computing (IJNDC), 2015,3(3):150−158. [doi: 10.2991/ijndc.2015.3.3.2]

[53] Qian H, Andresen D. Reducing mobile device energy consumption with computation offloading. In: Proc. of the 16th IEEE/ACIS Int'l Conf. on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD 2015). Takamatsu, 2015. [doi: 10.1109/SNPD.2015.7176219]

[54] Qian H, Andresen D. Extending mobile device's battery life by offloading computation to cloud. In: Proc. of the 2nd ACM Int'l Conf. on Mobile Software Engineering and Systems (MOBILESoft 2015). 2015. [doi: 10.1109/MobileSoft.2015.39]

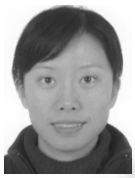
附中文参考文献:

[2] 王怀民,刘旭东,郎波,谢冰,毛晓光.软件可信分级规范 v2.0.技术规范,北京航空航天大学计算机学院,等,2009.

[4] 张璇,李彤,王旭,于倩,郁湧,朱锐.可信软件非功能需求形式化表示与可满足分析.软件学报,2015,26(10):2545−2566. http://www.jos.org.cn/1000-9825/4813. htm [doi: 10.13328/j.cnki.jos.004813]

[37] 金芝,刘璘,金英.软件需求工程:原理和方法.北京:科学出版社,2008.

[40] 陶红伟.基于属性的软件可信性度量模型研究[博士学位论文].上海:华东师范大学,2011.



张璇(1978－),女,江苏南京人,博士,副教授,CCF 会员,主要研究领域为需求工程,软件过程,可信软件.

王旭(1976－),男,博士,副教授,主要研究领域为金融安全,区域经济,计量经济学.

李彤(1963－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为软件过程,软件工程.

白川(1992－),男,硕士生,CCF 学生会员,主要研究领域为软件工程,需求工程.

康燕妮(1992－),女,硕士生,CCF 学生会员,主要研究领域为软件工程,需求工程.