文章编号：1007-5321(2004)03-0001-12

# 软件非功能属性研究

杨放春， 龙湘明

（北京邮电大学 交换技术与通信网国家重点实验室，北京 100876）

**摘要**：软件非功能属性在保证一个软件系统的质量中扮演关键角色. 本文分析了软件非功能属性研究的棕要意义和发展历史，从软件非功能的需求描述、设计和分析、实现机制、软件体系结构、软件开发过程以及软件质量评价 6 个方面介绍了软件非功能属性研究的主要进展，同时对已有研究中存在的非功能属性定义不一致、求精知识库缺乏、完整的开发过程欠缺等问题和不足进行了探讨，最后指出了软件非功能属性研究最有前途的发展趋势.

**关 键 词**：非功能属性；非功能需求；软件体系结构

**中图分类号**：TP311 **文献标识码**：A

# An Overview on Software Non-Functional Properties Research

YANG Fang-chun， LONG Xiang-ming

(State Key Laboratory of Switching Technology and Telecommunication Networks, Beijing University of Posts and Telecommunications, Beijing 100876, China)

**Abstract**: A key role in guaranteeing the quality of a software system is played by the software non-functional properties. As so, the research development as well as its significance on the software non-functional properties was analyzed. The advantages we mentioned above were prominent in six aspects: software non-functional requirement description; design and analysis; implementation mechanism; software architecture; software development process; and software quality evaluation. On the other hand, some disadvantage of the existed research was discussed either, such as inconsistent definition of non-functional properties; insufficient refinement repository; and deficiency of a complete developing process. In the final of the report, it is pointed out that the research on software non-functional properties is of promise.

**Key words**: non-functional properties; non functional requirement; software architecture

**收稿日期**：2004-03-01

**基金项目**：国家杰出青年基金项目(60125101)；国家自然科学基金项目(90104024)；教育部博士点基金项目

**作者简介**：杨放春(1957—)，男，教授，博士生导师. E-mail:fcyang@bupt.edu.cn

# 1 概 述

## 1.1 软件非功能属性研究的重要意义

一个软件系统的特性表现在它的功能性和非功能性(如性能、可靠性、安全等)2个方面.在许多软件系统中,尤其是大型软件系统中,非功能甚至是强制的要求,例如电信系统中的性能和可靠性要求,电子商务中的安全性要求等.

然而在许多实际系统的开发中却往往忽视了软件的非功能属性.在许多项目中,软件的非功能属性是在软件开发完成后再考虑的,但灾难由此发生:软件结构需要重新变化,大量代码需要重新编写,需要更多的时间、人力和物力.

有些系统一开始考虑到一些非功能属性,但没有进行良好的设计、分析和验证,当产品完成后才发现并不能满足当初的需求.

因忽视非功能需求而直接导致项目失败甚至造成重大经济损失的例子不胜枚举. 一个著名的案例是"伦敦救护车系统(LAS)"[1],这个系统在部署不久即陷入瘫痪,其中一个主要原因就是不能满足非功能需求(可靠性和性能极差). 文献[2]列举了更多这样的例子.

造成上述局面的原因有很多:

(1) 软件非功能属性本身的复杂性;

(2) 一些成熟的非功能分析手段,如性能分析中排队网、Petri 网等分析方法无法和软件设计有效地结合起来;

(3) 缺乏有力的工具、平台和语言的支持;

(4) 在实际设计和实现中仅仅凭借个人经验和直觉进行;

(5) 缺乏应有的重视.

因此,对软件的非功能属性进行仔细研究,找到一条能明确捕捉非功能需求,将非功能设计和功能设计有效集成以及将非功能设计、分析和验证有效结合,开发出相应的支持工具和平台,对软件系统的成功开发和应用具有重要意义.另外,软件非功能属性研究涉及的内容很广,目前尚缺乏理论体系的支持.

## 1.2 软件非功能属性研究的发展历史

对软件非功能属性局部的研究,如可靠性、安全等,是随着计算机的产生而发展起来的,甚至可以追溯到更早的时期[3],然而,将软件非功能属性当作一个整体进行系统地研究,则相对要晚得多.

早期的研究是所谓的"面向产品"的研究,反映在对软件质量评价的努力上.最早的工作是 Boehm 等人在 20 世纪 70 年代末提出的对软件质量属性的分类和评价标准[4,5],随后人们提出了各种各样的软件评价标准和软件质量模型[2,6-8].

然而,软件的质量是由软件设计、开发过程予以保证的. 直到 20 世纪 90 年代初,一种"面向流程"的方法才由 L. Chung 等人[9]引入,并在后来得到发展[10]. 这是一种在软件设计、开发过程中显式的考虑非功能属性的方法.

随后的研究基本上都可以归入"面向产品"和"面向流程"这 2 种方法中,而"面向流程"近年来显得更为活跃[11-14].

## 2　软件非功能属性研究现状

### 2.1　软件非功能属性的定义

事实上，对软件的非功能属性至今没有一个一致的定义. 这里列出几个比较经典的表述：

(1) N. S. Rosa[11]

软件的功能需求定义了一个软件期望做“什么”，而非功能需求(NFRs)则指定了关于软件“如何”运行和功能“如何”展示的全局限制.

(2) X. Franch[12]

软件的属性可以用来作为描述及评价软件的一种方式.

从文献[11]中的定义可以看到，“非功能”是相对于“功能”来说的. 而文献[12]中的定义更像是软件质量评价中的表述.

另外，对非功能“属性”一词，各文献中的表述也不尽相同. 文献[11]中使用了“需求(requirements)”一词，而文献[12]中使用的是“属性(attributes)”. 其他的表述还有“约束(constraints)[15]”、“目标(goals)[16]”、“质量属性(quality attributes)[17]”等. 而与“attributes”相似的表述还有 characteristics[7,8,18]、properties[19,20]、aspects[13]等. 它们之间有细微区别，例如术语“需求”通常针对软件开发的早期阶段而言，“质量属性”通常针对最终产品的评价而言，而“属性”一词则具有最广泛的意义. 后面交替使用这些术语，其确切的含义依据上下文确定.

那么，哪些是软件的非功能属性？从最广泛接受的观点上来看，软件的非功能属性应该包括以下方面[6,9,11,12,21,22]：性能、可重用性、可维护性、安全性、可靠性和可用性. 其他的非功能属性还有可修改性、可移植性、灵活性、可扩展性和适应性等.

上述非功能属性很多都是模糊的概念，没有一个统一的定义. 而且很多非功能属性意义接近，难以区分，例如可靠性和可用性. 如何更好地定义和区分这些非功能属性是一项重要的工作.

然而，软件的非功能属性和功能属性可以严格区分吗？J. E. Burge 等人[23]曾对这个问题产生过质疑，并进行了有趣的探讨. 最后认为这两者有时并不容易区分，同时指出非功能是对功能的补充，应该充分考虑它对功能需求的影响.

### 2.2　软件非功能属性研究的主要内容和进展

软件非功能属性研究的内容很广，归纳现有的研究活动，主要包括以下几个方面：

(1) 软件非功能需求描述. 研究如何捕捉非功能需求，其中非功能需求的求精和相关性的研究是其重要内容.

(2) 软件非功能设计、分析和验证. 研究如何将非功能需求转化为非功能设计，如何将非功能设计和功能设计集成起来，并分析和验证设计是否确实满足需求.

(3) 非功能软件体系结构. 研究如何将非功能属性引入到软件体系结构中或者说如何在体系结构级别支持非功能属性.

(4) 软件非功能实现机制. 研究非功能属性在特定技术和特定平台中的实现.

(5) 支持非功能属性的软件开发过程. 如何设计支持非功能属性需求描述、设计、分析、实现的整个软件开发过程.

(6) 软件质量评价和度量. 研究构成软件产品质量的要素以及如何利用这些质量要素评价和度量一个最终的软件产品的质量. 这里主要关心质量要素中的非功能部分.

从软件非功能属性研究的2种主要的方法——“面向流程”和“面向产品”来看. 上面列举的研究方向中,(1)～(5)是“面向流程”的方法需要关注的方面,可以看到,它们涵盖了一个软件开发流程中从需求描述到设计、分析和验证以及最终实现等几个主要的活动;而(6)则主要是“面向产品”的方法涉及的内容.

需要指出的是,第(3)方面“非功能软件体系结构”事实上是上面第(2)方面“软件非功能设计”中的一部分,因为它格外重要,已经形成一个重要的研究领域,所以这里对它单独进行讨论.

2.2.1 软件非功能需求描述

软件开发的第1个步骤是进行软件需求描述. 该需求包括功能需求和非功能需求2个方面. 功能需求通常比较明确和具体,而且大多数具有局部特点,较容易捕捉和描述,常用的描述手段是采用“用例[24]”或“场景[25]”的方式.

然而,相比于功能需求而言,非功能需求有很多不同的特点. 首先,非功能需求通常比较“抽象”,而且主观成分较多. 例如“性能”,就是一个很抽象的概念,不同的人会有不同的理解. 其次,非功能需求通常具有全局意义,例如“性能”通常就是针对整个系统而言. 另外,一个系统通常要考虑多个非功能需求,例如“性能”、“安全”、“可靠性”等,它们之间存在某种制约和依赖关系,这些关系也需要明确地区分和确定. 因为这些原因,要准确而直观地描述非功能需求并非易事. 在很长时间内,非功能需求都是使用自然语言的方式描述的,具有很大的随意性,缺乏精确性和完整性,给软件的设计和开发带来很大困难. L. Chung 等人[9,10,26]第一次就非功能需求的表示和描述进行了较为深入而全面的探讨,提出了著名的NFR计划. 在NFR计划中,使用术语“软目标”来表达“非功能需求”. 例如“好的性能”就是一个软目标,在每个软目标上有一个“可满足”断言,以表达此非功能需求是否会被满足.

一个非功能需求可以对应于一个或多个实现方法. 有意思的是,在NFR计划中,这些实现方法也被称为软目标,名为“操作化软目标”,为便于区别,将非功能需求称为“NFR软目标”.

为了能对抽象的非功能需求进行较好地描述,L. Chung 等人提出了软目标“求精”的概念,实际上就是将一个较抽象的软目标分解为多个较具体的子目标. 例如“好的性能”可以分解为“好的时间性能”和“好的空间性能”,而“好的时间性能”又可以再分解为“好的响应时间”和“好的吞吐量”. 当目标足够具体时,便可以指导设计. 因此,“操作化软目标”总是紧跟在“最”具体的“NFR软目标”后面. 同样,操作化软目标本身也可以求精,以便得到多种具体的实现方案.

同时,L. Chung 等人还研究了软目标之间的“依赖关系”. 同种类型之间软目标的依赖称为“显式依赖”,例如父目标和子目标的依赖,可以有2种依赖关系:AND和OR. AND表示当所有子目标满足时父目标才满足;OR表示只要有一个子目标满足,父目标便满足. 不同类型的软目标之间的依赖称为“隐式依赖”,又称为“相关性”. L. Chung 等人将相关性分为“正相关”和“负相关”2种. “正相关”意味着一个软目标将“有利于”另外一个软目标,“负相关”则是一个软目标对另一个软目标起到反作用. 例如,可靠性和可用性之间就是正相关的,而性能和安全之间往往是负相关的.

NFR计划使用了一套自定义的完整的图形元素进行描述:“NFR软目标”用“云图”表示,“操作化软目标”用加黑的“云图”表示;软目标之间不同的依赖关系用不同的连线表示. 所有软

目标和它们之间的关系组成了 SIG(softgoals independencies graph)图. 在 NFR 计划中,进行非功能描述和设计的过程就是绘制 SIG 图的过程.

NFR 计划在几个信息系统的开发中得到了重要应用[26], 对系统的精确性、性能和安全进行了较好地描述和设计. 然而 NFR 计划提供的是"定性"的描述方法,因此无法进行"定量"的分析和验证. 同时,由于采用一套封闭的图形表示法,在与功能部分集成时,不便于与 UML 等主流建模工具结合使用.

除了 L. Chung 等人的图形化方法,还有许多纯形式化的描述非功能属性的语言. 这类语言有 X. Franch[12,18]提出的 NoFun 以及 N. S. Rosa[11]提出的 Process-NFL 等. NoFun 和 Process-NFL 的贡献是使得非功能需求的描述可以与体系结构描述语言(功能部分)结合起来. 无论是 NoFun 还是 Process-NFL,采用的都是与图形化方法中相似的概念,而表达能力甚至更弱,例如没有对非功能属性的求精和相关性进行很好地表达,因此是否实用还值得怀疑.

另外,还有使用基于逻辑系统的概念来表达非功能属性的,如 first-order logic[27], temporal logic[28]和 predicate logic[29]. 它们在某种程度上增加了对非功能属性的精确描述和判断的能力,但是对于没有逻辑背景知识的系统设计人员来讲,这是一种很难掌握的方式,而且难以与软件开发过程结合起来.

需要指出的是,非功能属性的求精和非功能属性之间的相关性是非功能需求描述中最重要的 2 个方面,这些工作完全由设计者完成是困难的. 实际上应该存在这样一套知识库供设计者选择,目前这样的知识库还非常少. 因此,丰富和完善非功能属性求精知识库和相关性知识库是今后研究中的一个重要方面.

2.2.2　软件非功能设计、分析与验证

选择和确定一组满足非功能需求的"决定"的过程,就是软件非功能设计. 实际上,在前面提到的非功能需求描述框架或描述语言中,很多已经包含了"设计"的内容. 例如 NoFun 语言中的 NF-behavior、Process-NFL 中的 NF-Realizations 以及 NFR 计划中的"操作化软目标",都是在描述一个"设计决定".

需要指出的是,非功能设计不是独立于功能设计而存在的,而是反过来影响功能设计,最终和功能设计集成起来的. 因此,非功能设计的过程是一个非常复杂的过程.

关于非功能和功能的集成,L. Chung 的框架没有很好地解决. 在这方面作出努力的是 L. M. Cysneiros 等人,他们通过引入 LEL(language extended lexicon)[14],试图将 L. Chung 的非功能描述手段和 UML 的功能描述手段(用框图、类图、顺序图、协作图等)结合起来,从而将非功能设计和功能设计有效地集成.

所谓"非功能分析和验证",就是确认所作出的非功能设计是否真的满足非功能需求,特别是在与功能设计集成之后,这一工作显得尤其重要. 然而,无论是 L. Chung 的 NFR 计划[10],还是 N. S. Rosa 等人的框架[11]以及 L. M. Cysneiros 等人的集成策略,都回避了这个问题,他们的框架中采用的都是定性的方法,因而无法进行定量分析.

造成分析和验证困难的一个重要原因是许多成型的分析方法,如性能分析中的排队网络、Petri 网,可靠性分析中的各种可靠性模型和可靠性预测方法等,都需要复杂的数学手段和仿真工具,这通常不是一个软件设计师所能胜任的. "设计"和"分析"之间确实存在着一道鸿沟.

致力于这一工作并企图填补这一鸿沟的是 Pooley、J. Skene 和 W. Emmerich 等人以及 OMG 组织. 他们的思想是:如果设计方法本身就隐含分析领域中的概念,例如在 UML 的设计

元素中引入性能分析中的“资源”等概念,那么就可以通过一套映射手段将设计本身自动转化为分析方法,例如将“资源”和排队网络中的“队列”对应起来,就可以自动进行排队网络分析,这样就避免了设计人员需要熟悉复杂的分析方法. Pooley 等人最早使用 UML 来表达性能分析中的概念[30],而 OMG 新制定的草案——real time profile[31]则使用 UML 的扩展机制——UML profile 对性能进行了建模. 此后,J. Skene 和 W. Emmerich 做了性能设计到性能分析(排队网络)的映射工作[32].

文献[33]则使用 UML profile 对可靠性进行了建模. 文献[34]提出了 SecureUML,对安全性进行建模.

需要指出的是,并不是所有的非功能设计都是可以采用数学手段或者仿真工具予以分析和验证的,通常只限于能够定量描述的非功能属性. 有些非功能属性,如灵活性、可移植性、易用性等,并不是具备明显的数学模型和数学特征,对它们的分析和验证,更多的借助于约定的习惯、成熟的模式或已有的经验,有时可能需要编写试验程序予以验证.

2.2.3 非功能软件体系结构

非功能软件体系结构是指满足非功能属性的软件体系结构. 非功能属性大都具有全局性,因此非常适合在软件体系结构级别支持非功能属性.

一种方式是在软件体系结构描述语言(ADL,architecture description language)中支持非功能属性的描述. 软件体系结构描述语言 ADL 数量众多,但直接支持非功能属性的仅有少数几个. MetaH[21,35]和 UniCon[36]是其中比较著名的 2 个. MetaH 主要应用于嵌入式实时电子控制系统,提供对性能、可靠性和安全的描述与分析. UniCon 适合于异构系统之间的互连,支持对性能的建模和分析. 然而它们只支持特定的非功能属性,因此不是普遍意义上的非功能软件体系结构.

设计一个全新的支持功能和非功能描述的体系结构描述语言可能并不现实. 实际上,很多学者在研究如何将非功能属性引入到现有的大多数体系结构描述语言(ADL)中. 他们通常是定义一种非功能属性描述语言,这种语言本身就考虑到了非功能属性到体系结构元素如组件、连接器中的对应. 如前面提到的 X. Franch 的 NoFun[12]和 N. S. Rosa 的 Process-NFL[11]等都是这样的例子.

当在软件体系结构中考虑非功能属性时,非功能属性的求精过程需要和软件体系结构的求精过程统一起来. N. S. Rosa 等人研究了这个问题,提出了相应的支持非功能属性的软件体系结构的求精规则[11]. 但 N. S. Rosa 没有提出如何运用这些规则的进一步指导和相应例证,这些规则是否成立也值得商榷.

需要特别指出的是,体系结构风格对于满足不同的非功能需求具有重要影响. 如何根据不同的非功能要求选择合适的体系结构风格,是非功能软件体系结构设计中的重要一环. 然而,现有的方法中没有对这个问题予以充分讨论.

2.2.4 软件非功能属性实现机制和实现技术

L. Chung、L. M. Cysneiros 以及 N. S. Rosa 等人关于非功能实现的思路都是基于这样一个设想:通过增加功能部分(如增加新的类或者在已有类中增加新的属性和操作等)来实现非功能需求. 采用这种思路,非功能实现技术和功能实现技术是完全一样的.

然而,将非功能属性单独分离出来进行实现是可能的. 这是基于“关注分离[37](separation of concerns)”这样一个重要思想. 所谓“关注分离”,就是将系统中需要关注的某些方面从其他

方面中分离出来予以单独考虑，从而减轻复杂度. 许多技术，如 AOP (aspect-oriented programming)[38,39]、反射[35,40,41]以及分布式组件(CORBA/CCM[42]和 J2EE/EJB[43])中的容器模型等，都是这一思想的体现.

(1) 基于特定技术的实现机制

这里介绍 AOP 和反射技术. 当然，这些技术适合于其他需要“分离”的场合，但应用在非功能与功能的分离中是非常合适的选择.

AOP[38,39]是代码级别的技术，它提供一种机制，使得非功能代码和功能代码可以分开编写，同时又能组合起来运行. AOP 的支持者正在努力扩展现有的编程语言，使得它们能够直接支持 AOP 技术. 目前已有多个这样的实现，例如 AspectJ[44]就是 Java 语言的 AOP 扩展，而 AspectC++[45]是 C++语言的 AOP 扩展.

反射的概念最初由 Smith[40,41]在 1982 年提出. 抽象地讲，反射是系统的一种推理和作用于自身的能力. 一个反射系统分为基层(base-level)和元层(meta-level)，两者之间的界面可以通过“元对象协议(MOP，meta-object protocol)[35]”进行定义. 使用反射技术实现非功能属性时，可以让基层实现功能部分，而让元层实现非功能部分. 改变元层将使得基层获得不同的非功能特性. M. O. Killijian 等人就使用了反射技术实现分布式系统的容错[46].

AOP 和反射技术的应用将可以极大地方便非功能的实现. 而且，采用这些技术本身就是提高系统可重用性、可适应性的手段. 但需要指出的是，并不是所有的非功能属性都适合采用这种技术实现，许多非功能属性事实上并不能和功能部分分开考虑，例如数据完整性. 同时，采用这些技术虽然方便了某些非功能属性的实现，但是可能影响其他非功能属性，例如性能. 因此，设计者在选择这些技术时需要慎重权衡.

(2) 基于特定平台的实现

这里主要讨论分布式中间件平台中的实现. CORBA/CCM[42]和 J2EE/EJB[43]是 2 种主流的分布式中间件平台，同时也是分布式组件平台. 这 2 种组件都使用了容器模型. 容器是现代组件模型的一个显著特点. 组件总是在容器中运行，容器是组件的一个执行环境. 容器承担了组件的大部分非功能性需求(如并发、事务、安全、持久性等公共服务)，而组件只需集中完成功能性任务即可，这给组件的开发和部署带来了极大的便利.

然而，容器模型的一个致命缺陷是所有非功能能力都是预先规定的. 当组件需要其他非功能特性时，容器则无法提供. 这需要修改组件本身以使它具备这个能力. 但是如果使用的是一个第 3 方的组件，通常并不能修改它. F. Duclos 等人[13]研究了这个问题，通过在组件中引入 AOP 技术，可以在不需要修改组件的情况下使它获得所期望的其他非功能特性.

### 2.2.5　支持非功能属性的软件开发过程

非功能属性应该在软件开发过程中予以考虑. 这个开发过程应该全面考虑非功能需求、非功能设计(包括与功能部分集成)、非功能分析和验证、非功能实现等. “面向流程”的方法企图在这方面作出努力，但是已有的方法都存在很大缺陷.

L. Chung 等人的 NFR 计划只考虑了需求和设计，而没有任何分析和验证. 尽管也考虑到实现方法，但仅仅是高层的指导，没有考虑到具体平台的实现机制.

N. S. Rosa 等人提出的名为 Parmenides[11]的非功能软件开发框架，是一个以软件体系结构为中心的支持非功能属性的开发框架，包括非功能需求定义、与功能部分集成、与具体实现平台的映射等. 但也没有考虑分析和验证问题，而且，以软件体系结构为中心只能解决部分高

层设计中“结构性”的问题,对于与系统行为相关的非功能属性则是无能为力.

J. Skene 和 W. Emmerich 等人提出使用 MDA(model driven architecture)[3]框架作为非功能分析的手段[32]. 然而,除了使用 UML Profile 做了非功能设计到分析之间的映射外,没有做更进一步的工作.

另外,现有的方法也没有考虑如何与传统的软件开发周期模型相结合的问题. 传统的软件开发周期模型如瀑布模型、迭代模型、增量模型、快速原型模型等,主要考虑的是功能部分,引入非功能部分后会产生什么样的影响,是否需要新的模型等,都需要进一步研究.

2.2.6 软件质量评价和度量

软件开发完成后,需要评价它的质量. 软件质量评价和度量就是从整体上评价一个软件产品的质量. 当使用“度量(metric)”一词时,则是强调“定量”评价. 自从 1976 年 Boehm 提出定量评价软件质量的概念以来,出现了各种各样的软件评价方法和软件质量模型,其中影响较大的有 McCall 的 3 层次质量度量模型[47]和 ISO 9126[7,8]软件质量模型.

McCall 的 3 层次质量度量模型由质量要素(quality factor)、评价准则(criteria)和度量(metric)构成. McCall 认为,“质量要素”是从用户的角度观察到的软件质量特征,但并不是软件质量的原始属性. 软件产品本身具有一些客观、独立的原始质量属性,它们构成了软件的“评价准则”,而这些准则可以定量“度量”. McCall 提出了 11 个软件质量要素,如可靠性、灵活性、可重用性以及 23 个评价准则(准确性、简单性、模块性等). 一个质量要素可以通过多个评价准则来反映,一个评价准则也可以和多个质量要素相关联.

ISO 9126 软件质量模型是在 McCall 的 3 层模型基础上制定的,分为高、中、低 3 层,其中高层称为“外部属性”;中层称为“内部属性”,也称为“子属性”;低层为“度量”. 外部属性是最终用户所观察到的软件产品的属性,而内部属性是软件设计人员所感知的属性. 内部属性影响外部属性. ISO 9126 提出了 6 个外部属性(功能性、可靠性、易用性、可维护性、可移植性、效率)以及 21 个子属性.

根据 ISO 9126 质量模型,软件质量定义为:与软件产品满足明确或隐含需求的能力有关的特征或特性的总和. 从这个定义以及 6 个外部属性可以明显看出,软件非功能属性与软件质量属性的关系是,软件非功能属性是软件质量属性除去“功能性”后的一个子集.

目前,软件质量评价和度量已经成为软件工程中的一个重要研究方向. 我国在此领域的研究起步较晚,其中最引人瞩目的是上海软件中心提出的 SSC(Shanghai software center)软件质量评价体系[48]. SSC 模型在 ISO 9126 三层次模型的基础上做了修改和改进,采用了其 6 个质量属性和 21 个子属性,同时提出了 71 个度量,并在国内首次提出了模型分级的设想,完成了部分软件类型的模型分级. 依据这个评价体系,上海软件中心已经开发出了相应的“软件质量评价平台”,并获得了初步应用[48].

软件质量评价和度量体现了“面向产品”的非功能属性研究方法,一方面提供了对软件质量的评价方法;另一方面可以作为软件设计和开发的高层指导. 但是,它并不能提供具体的开发流程上的指导,使最终产品能够满足特定的质量要求,这是“面向流程”方法需要解决的问题.

然而,需要指出的是,“面向产品”和“面向流程”的方法是可以进行结合的. 这种结合不仅应该体现在软件开发完成之后,也应该体现在软件设计之初的需求描述阶段. 例如,ISO 9126 就提出了质量属性“求精”的概念,这与前面提到的非功能需求的“求精”概念是类似的,只是后

者更为复杂. 同时，质量模型中“内部属性”对“外部属性”的影响也可以直接用来进行非功能属性的“相关性”研究. 目前进行这方面研究的只有 N. S. Rosa 等人. 在他们提出的 Parmenides 框架[11]中，同时包含面向流程的 Process-NFL 语言和面向产品的 Product-NFL 语言. 但是，这种结合仅仅是在产品开发完成之后，并没有体现在早期需求阶段.

## 3　软件非功能属性研究中存在的问题

自从 Boehm 等人开创“面向产品”的研究方法和 L. Chung 等人开创了“面向流程”的研究方法以来，软件非功能属性的研究取得了重要进展. 但同时也可以看到，软件非功能属性的研究还很不充分，存在很多问题和不足：

(1) 没有一个对软件的非功能属性的统一和一致的定义，对软件非功能属性涉及的范围没有一致的认识，对各单独属性的定义也不尽相同，从而造成研究、理解和沟通上的困难.

(2) 非功能属性求精方法和相关性研究有待深入. “求精知识库”和“相关性知识库”非常缺乏.

(3) 非功能与功能部分的集成有待进一步研究.

(4) 非功能设计和非功能分析结合的研究刚刚起步，还有许多问题有待解决.

(5) 在软件体系结构级别支持非功能属性的研究仍然欠缺，一个原因是软件体系结构的研究方兴未艾，人们对软件体系结构本身就缺乏一致的认识.

(6) 特定技术和平台相关的非功能实现机制有待进一步研究. 缺乏从高层设计到底层实现映射的研究.

(7) 现有的方法偏重于通用方法的研究，考虑各非功能属性差异性的研究较少. 将各特定领域(如性能、可靠性、安全等领域)的研究成果引入到通用方法中的研究非常欠缺.

(8) 没有一个支持非功能属性的完整的开发过程. 已有的方法中要么侧重于早期需求阶段(如 L. Chung 等人的方法)，要么侧重于分析(如 J. Skene 和 W. Emmerich 等人)，要么是具体的实现机制(如反射和 AOP).

(9) 面向流程的方法和面向产品的方法缺乏有效地结合.

(10) 仍然缺乏有效的工具支持.

## 4　结　论

软件的非功能属性对于保证一个软件系统的质量具有重要意义. 在软件开发完成后测试和评价一个软件产品固然重要，然而在开发过程中显式的考虑非功能属性却显得更为重要. 尽管软件非功能属性的研究已经取得一些重要进展，但无论是“面向流程”的方法还是“面向产品”的方法，都有待进一步完善，还有很多问题需要深入研究.

随着软件体系结构研究的深入，在软件体系结构级别支持非功能属性的研究必将是未来软件非功能属性研究的一个主要方向之一.

需要特别提及的是关于 MDA 和 UML Profile 用于非功能属性的研究的趋势[32]. 尽管目前这方面的研究刚刚起步，但鉴于 MDA 良好的多视点架构和 UML 在面向对象的功能建模领域的领导地位，极有可能成为将非功能和功能完美结合起来的最佳方式.

另外，非功能需求不是一成不变的，即便在设计阶段充分考虑了各种需求，系统开发完成之后的需求仍然可能发生改变，因此，如何支持这种非功能需求的改变或者说保证非功能属性

的演进,显得非常重要. 目前这方面的研究还非常少,它将成为今后软件非功能属性研究中的另一个重要课题.

最后需要指出的是,国内在软件非功能属性方面的研究非常缺乏. 除了上面提及的在"面向产品"的研究上(软件质量评价和度量)有一定进展外,"面向流程"的研究则基本停留在"需求描述"阶段[49],没有大的突破. 因此,迫切需要国内专家和学者参与到软件非功能属性的研究中来,为提高我国软件产品的质量提供理论和方法学指导以及工具支持,促进我国软件产业的发展.

## 参考文献:


[1] Finkelstein A, Dowell J. A comedy of errors: the London ambulance service case study[A]. Proceedings of the 8th International Workshop on Software Specification and Design[C]. 1996. 2-4.

[2] Lindstrom D R. Five ways to destroy a development project[J]. IEEE Software, 1993, 10(5):55-58.

[3] OMG. Model driven architecture(MDA)( ormsc/2001-07-01)[S].

[4] Boehm B W, Brown J R, Lipow M. Quantitative evaluation of software quality[A]. Proceedings of 2nd International Conference on Software Engineering[C]. San Francisco, CA: IEEE Computer Society, 1976. 592-605.

[5] Boehm B W. Characteristics of software quality[M]. Amsterdam:North-Holland, 1978.

[6] Losavio F, Chirinos L, Perez M A. Quality models to design software architectures[A]. Proceedings of Technology of Object-Oriented Languages and Systems(TOOLS)[C]. IEEE, 2001. 123-135.

[7] ISO/IEC IS 9126—1991, Information technology-software product evaluation: quality characteristics and guidelines for their use[S].

[8] ISO/IEC FCD 9126-1. 2—1998, Information technology-software product quality. part 1: quality model(draft)[S].

[9] Mylopoulos J, Chung L, Nixon B A. Representing and using nonfunctional requirements: a process-oriented approach[J]. IEEE Transactions on Software Engineering, 1992,18(6):483-497.

[10] Chung L, Nixon B A, Yu E, et al. Non-functional requirements in software engineering[M]. Kluwer Academic Publishers,1999.

[11] Rosa N S, Justo G R R, Cunha P R F. A framework for building non-functional software architectures[A]. Proceedings of the 2001 ACM symposium on applied computing[C]. ACM Press, 2001.

[12] Franch X, Botella P. Putting non-functional requirements into software architecture[A]. Proceedings of Ninth International Workshop Software Specification and Design[C]. IEEE, 1998. 60-67.

[13] Frédéric Duclos, Jacky Estublier, Philippe Morat. Describing and using non functional aspects in component based applications[A]. Proceedings of the 1st International Conference on Aspect-Oriented Software Development[C]. ACM, 2002.

[14] Luiz Marcio Cysneiros, Julio Cesar Sampaio do Prado Leite. Non-functional requirements: from elicitation to modelling languages[A]. Proceedings of the 24th International Conference on Software Engineering[C]. ACM, 2002.

[15] Roman G C. A taxonomy of current issues in requirements engineering[J]. IEEE Computer, 1985,18(4):15-23.

[16] Mostow J. Towards better models of the design process[J]. AI Magazine, 1985, 6(1):44-57.

[17] Keller S E. Specifying software quality requirements with metrics, in tutorial:system and software

engineering[M]. IEEE Computer Society Press, 1990. 145-163.

[18] Franch X. The convenience for a notation to express non-functional characteristics of software components [A]. Foundations of Component-Based Systems Workshop (FoCBS) [C]. Zurich, Switzeland, 1997. 101-109.

[19] George R, Ribeiro-Justo, Ahmed Saleh. Non-functional integration and coordination of distributed component services[A]. Proceedings of the Sixth European Conference on Software Maintenance and Reengineering (CSMR. 02)[C]. IEEE, 2002.

[20] Skene J, Emmerich W. A model driven architecture approach to analysis of non-functional properties of software architectures [A]. Proceedings of the 18th IEEE Conference on Automated Software Engineering[C]. Montreal, Canada: IEEE Computer Society Press, 2003.

[21] Binns P, Engelhart M, Jackson M, et al. Domain-specific software architectures for guidance, navigation, and control [J]. International Journal of Software Engineering and Knowledge Engineering, 1996, 6(2): 66-68.

[22] Cohen D, Goldman N, Narayanaswamy K. Adding performance information to ADT interfaces[A]. Proceedings of the Interface Definition Languages Workshop[C]. ACM Sigplan Notices, 1994.

[23] Janet E B, David C B. NFR's: Fact or Fiction? [EB/OL]. http://www.cs.wpi.edu/~dcb/Papers/CASCON03.pdf, 2003.

[24] Booch G, Rumbaugh J, Jacobson I. The unified modeling language user guide[M]. Addsion Wesley Longman, 1999.

[25] Leite J C S P. Enhancing a requirements base line with scenarios [J]. Requirements Engineering Journal, 1997, 2(4): 184-198.

[26] Chung L, Nixon B A. Dealing with non-functional requirements: three experimental studies of a process-oriented approach [A]. Proceedings of IEEE 17th International Conference on Software Engineering[C]. Seattle, 1995. 25-37.

[27] Issarny V, Bidan C, Saridakis T. Achieving middleware customization in a configuration-based development environment: experience with the Aster prototype[A]. Proceedings of 4th International Conference on Configurable Distributed Systems[C]. Maryland: Annapolis, 1998. 207-214.

[28] Zarras A, Issarny V. A framework for systematic synthesis of transactional middleware [A]. Proceedings of Middleware'98[C]. England: The Lake District, 1998. 257-272.

[29] Saridakis T, Issarny V. Fault tolerant software architectures[Z]. Technical Report 3350, I NRIA, 1998.

[30] Pooley R, King P. The unified modeling language and performance engineering [A]. IEEE Proceedings of Software[C]. 1999. 2-10.

[31] OMG. UML profile for schedulability, performance, and time specification[S].

[32] Skene J, Emmerich W. A model driven architecture approach to analysis of non-functional properties of software architectures [EB/OL]. http://www.cs.ucl.ac.uk/staff/w.Emmerich/publications/ASE2003/index.html, 2003-09.

[33] Rodrigues G N, Roberts G, Emmerich W, et al. Reliability support for the model driven architecture [A]. Workshop on Software Architectures for Dependable Systems (WADS), in Conjunction with ICSE '03[C]. Portland: ACM Press, 2003.

[34] Torsten L, David B, Jürgen D. SecureUML: a UML-based modeling language for model-driven security[A]. Proceedings of the 5th International Conference on the Unified Modeling Language[C]. 2002. 426-441.

[35] Gregor Kiczales, Jim des Rivi'eres. The art of the metaobject protocol[M]. Massachusetts: MIT Press, 1991.

[36] Shaw M, DeLine R, Zelesnik G. Abstractions and implementations for architectural connections[A]. Proceedings of the Third International Conference on Configurable Distributed Systems[C]. 1996.

[37] Lopes C V, Hursch W L. Separation of concerns[M]. Boston: College of Computer Science, Northeastern University, 1995.

[38] AOP Home Page[EB/OL]. http://www.aosd.net/.

[39] Kiczales G. Aspect-oriented programming[A]. Proceedings of the European Conference on Object-Oriented Programming (ECOOP)[C]. Springer-Verlag, 1997.

[40] Smith B C. Reflection and semantics in a procedural language[Z]. Technical Report, MIT Laboratory of Computer Science, 1982.

[41] Smith B C. Reflection and semantics in Lisp[A]. Proceedings of ACM Symposium on Principles of Programming Languages[C]. 1984. 23-35.

[42] OMG. CORBA Components v3.0 (formal/02-06-65)[S].

[43] Sun. Enterprise JavaBeans specification, version 2.1, proposed final draft 2(ejb-2—1-pfd2-spec)[EB/OL]. http://java.sun.com/webapps/download/Redirect/60366713/ejb-2_1-pfd2-spec.pdf, 2003-06-02.

[44] AspectJ home page[EB/OL]. http://aspectj.org/.

[45] Aspect C++ home page[EB/OL]. http://www.aspectc.org/.

[46] Killijian M O, Fabre J C, Ruiz-Garcia J C, et al. A metaobject protocol for fault-tolerant CORBA applications[A]. Proceedings of IEEE SRDS'98[C]. Indiana: West Lafayette, 1998. 127-134.

[47] McCall J. The automated measurement of software quality[A]. Proceedings of 5th COMPSAC'81[C]. 1981.

[48] **叶言苓, 朱三元. 软件质量评价体系及其实现**[J]. **计算机应用与软件**, 2001, 18(1):26-33.
Ye Yanling, Zhu Sanyuan. Software quality evaluation system and its implementation[J]. Computer Application and Software, 2001,18(1):26-33.

[49] **雍信阳, 施伯乐. 非功能需求跟踪**[J]. **计算机研究与发展**, 1998, 35(7):584-588.
Yong Xinyang, Shi Bole. Non-functional requirements traceability[J]. Computer Research and Development, 1998,35(7):584-588.

# 软件非功能属性研究



作者： 杨放春， 龙湘明
作者单位： 北京邮电大学,交换技术与通信网国家重点实验室,北京,100876
刊名： 北京邮电大学学报 ISTIC EI PKU
英文刊名： JOURNAL OF BEIJING UNIVERSITY OF POSTS AND TELECOMMUNICATIONS
年，卷(期)： 2004,27(3)
被引用次数： 8次

## 参考文献(49条)

1. Finkelstein A;Dowell J A comedy of errors:the London ambulance service case study 1996

2. Lindstrom D R Five ways to destroy a development project[外文期刊] 1993(05)

3. OMG.Model driven architecture(MDA)(ormsc/2001-07-01) 2001

4. Boehm B W;Brown J R;Lipow M Quantitative evaluation of software quality 1976

5. Boehm B W Characteristics of software quality 1978

6. Losavio F;Chirinos L;Perez M A Quality models to design software architectures[外文期刊] 2001

7. ISO/IEC IS 9126-1991,Information technology-software product evaluation:quality characteristics and guidelines for their use

8. ISO/IEC FCD 9126-1.2-1998,Information technology-software product quality.part 1:quality model(draft)

9. MYLOPOULOS J;Chung L;Nixon B A Representing and using nonfunctional requirements:a process-oriented approach 1992(06)

10. CHUNG L;Nixon B A;Yu E Non-functional requirements in software engineeringKluwer Academic Publishers 1999

11. Rosa N S;Justo G R R;Cunha P R F A framework for building non-functional software architectures 2001

12. Franch X;Botella P Putting non-functional requirements into software architecture[外文会议] 1998

13. Frédéric Duclos;Jacky Estublier;Philippe Morat Describing and using non functional aspects in component based applications 2002

14. Luiz Marcio Cysneiros;Julio Cesar Sampaio do Prado Leite Non-functional requirements:from elicitation to modelling languages 2002

15. Roman G C A taxonomy of current issues in requirements engineering 1985(04)

16. MOSTOW J Towards better models of the design process 1985(01)

17. Keller S E Specifying software quality requirements with metrics,in

tutorial:system and software engineeringIEEE Computer Society Press 1990

18. Franch X The convenience for a notation to express non-functional characteristics of software components 1997

19. George R;Ribeiro-Justo;Ahmed Saleh Non-functional integration and coordination of distributed component services 2002

20. Skene J;Emmerich W A model driven architecture approach to analysis of non-functional properties of software architectures 2003

21. Binns P;Engelhart M;Jackson M Domain-specific software architectures for guidance,navigation,and control 1996(02)

22. Cohen D;Goldman N;Narayanaswamy K Adding performance information to ADT interfaces 1994

23. Janet E B;David C B NFR's:Fact or Fiction? 2003

24. Booch G;Rumbaugh J;Jacobson I The unified modeling language user guideAddsion Wesley Longman 1999

25. Leite J C S P Enhancing a requirements base line with scenarios[外文期刊] 1997(04)

26. Chung L;Nixon B A Dealing with non-functional requirements:three experimental studies of a process-oriented approach 1995

27. Issarny V;Bidan C;Saridakis T Achieving middleware customization in a configuration-based development environment:experience with the Aster prototype 1998

28. Zarras A;Issarny V A framework for systematic synthesis of transactional middleware 1998

29. Saridakis T;Issarny V Fault tolerant software architectures 1998

30. Pooley R;King P The unified modeling language and performance engineering 1999

31. OMG.UML profile for schedulability,performance,and time specification

32. Skene J;Emmerich W A model driven architecture approach to analysis of non-functional properties of software architectures 2003

33. Rodrigues G N;Roberts G;Emmerich W Reliability support for the model driven architecture 2003

34. Torsten L;David B;Jürgen D SecureUML:a UML-based modeling language for model-driven security 2002

35. Gregor Kiczales;Jim des Rivi`eres The art of the metaobject protocol 1991

36. Shaw M;DeLine R;Zelesnik G Abstractions and implementations for architectural connections 1996

37. Lopes C V;Hursch W L Separation of concerns 1995

38. AOP Home Page

39. Kiczales G Aspect-oriented programming 1997

40. Smith B C Reflection and semantics in a procedural language 1982

41. Smith B C Reflection and semantics in Lisp 1984

42. OMG.CORBA Components v3.0 (formal/02-06-65)

43. Sun.Enterprise JavaBeans specification,version 2.1,proposed final draft 2(ejb-2_1-pfd2-spec) 2003

44. AspectJ home page

45. Aspect C++ home page

46. Killijian M O;Fabre J C;Ruiz-Garcia J C A metaobject protocol for fault-tolerant CORBA applications[外文会议] 1998

47. McCall J The automated measurement of software quality 1981

48. 叶言苓;朱三元 软件质量评价体系及其实现[期刊论文]-计算机应用与软件 2001(01)

49. 雍信阳;施伯乐 非功能需求跟踪[期刊论文]-计算机研究与发展 1998(07)

## 本文读者也读过(8条)

1. 方喜峰.赵良才.FANG Xi-feng.ZHAO Liang-cai 基于知识的CAQFD系统的开发[期刊论文]-华东船舶工业学院学报2000,14(1)

2. 徐莹 一种本体驱动的软件体系结构评价方法[期刊论文]-中国管理信息化2012,15(4)

3. 琚川徽.程勇.袁兆山 需求驱动的软件体系结构设计[期刊论文]-合肥工业大学学报(自然科学版)2002,25(3)

4. 李红卫.王映辉.LI Hong-wei.WANG Ying-hui 面向Internet/Web平台的大规模软件架构技术[期刊论文]-科学技术与工程2008,8(17)

5. 刘永纯.伦立军 化学抽象机在软件体系结构中的应用[期刊论文]-黑龙江科技信息2008(33)

6. 何文孝.HE Wen-xiao 软件非功能需求综合评判算法研究[期刊论文]-煤炭技术2010,29(12)

7. 韩强.冯翼.丁静.HAN Qiang.FENG Yi.DING Jing 基于SOA的商业银行电子渠道系统网络建设[期刊论文]-大连民族学院学报2008,10(3)

8. 廖日坤.纪越峰 基于嵌入式TCP/IP软件体系结构的优化设计与实现[期刊论文]-电子产品世界2007(2)

## 引证文献(9条)

1. 赵新辉.袁开银.吴尽昭 网络式软件非功能需求冲突消解[期刊论文]-计算机工程 2012(18)

2. 刘敬勇.钟勇.张立臣 软件非功能需求的面向方面建模[期刊论文]-计算机应用与软件 2010(12)

3. 江涛.段富 基于QoS的Web Services发现[期刊论文]-太原理工大学学报 2006(3)

4. 王磊.黄昌军 煤矿安全监控系统模型设计与自动分析研究[期刊论文]-榆林学院学报 2009(6)

5. 杨汉明.张立臣 面向方面的软件非功能特性研究[期刊论文]-计算机技术与发展 2011(2)

6. 李琳. 葛孝堃. 吴国文 基于缓存和代理的QoS服务发现机制[期刊论文]-计算机应用与软件 2008(12)

7. 尹祖全 基于Windows NT中央控制站实时引控软件体系结构研究[学位论文]硕士 2005

8. 张琳琳. 应时. 赵楷. 文静. 倪友聪 一种建模软件体系结构非功能属性的方法[期刊论文]-计算机科学 2009(7)

9. 江涛 基于QoS的Web Services发现[学位论文]硕士 2006