# 非功能需求的结构化定义以及概念性追踪管理框架*

孙连山，王今雨
（陕西科技大学 电气与信息工程学院 计算机系，西安 710021）

**摘　要**：针对需求工程中非功能需求概念非常模糊甚至相互矛盾、非功能需求与其他非功能需求及功能需求之间的关系繁复而难以分析和建模、非功能需求与设计阶段制品之间的追踪关系模糊而不易记录和维护等问题，分析了与非功能需求相关的概念在需求分析阶段和体系结构设计阶段的表现形式，给出了一个结构化的非功能需求定义；规范了不同类型需求之间的各种复杂关系，建立了一个跨越分析和设计阶段的概念性非功能需求追踪管理框架，规范了需求分析和体系结构设计阶段与非功能需求相关的概念和制品之间的关系。提出的结构化定义以及概念性追踪管理框架明确地刻画了非功能需求概念的外延，为简化需求模型以及进一步研制系统化、实用化的非功能需求建模及追踪管理技术奠定了理论基础。

**关键词**：非功能需求；功能需求；软件体系结构；追踪性；设计决策
**中图分类号**：TP311.5　　**文献标志码**：A　　**文章编号**：1001-3695(2012)12-4598-06
doi:10.3969/j.issn.1001-3695.2012.12.051

# Structured definition and conceptual traceability management framework of nonfunctional requirements

SUN Lian-shan, WANG Jin-yu
(*Dept. of Computer Science, College of Electrical & Information Engineering, Shaanxi University of Science & Technology, Xi'an* 710021, *China*)

**Abstract**: The definitions of nonfunctional requirements (NFRs) in literature are ambiguous and even conflicting with each other. In real software development and evolution, the relationships among NFRs and functional requirements (FRs) are too complex to model and manage and the traceability from NFRs to design phase artifacts are often too elusive to record and maintain. This paper analyzed concepts related to NFRs and their representations in phases of requirements analysis and software architecting and presented a structured definition of NFRs. This definition normalized the relationships among FRs and NFRs. Furthermore, it built a conceptual traceability management framework to relate NFRs to concepts in software architecture. The framework normalized the relationships among architectural concepts and NFRs. The formal definition and the conceptual traceability management framework of NFRs together clarify the conceptual denotation of NFRs and form a solid theoretical base for simplifying the requirements model and developing systematic and practical approaches to modeling and managing nonfunctional requirements.

**Key words**: nonfunctional requirements; functional requirements; software architecture; traceability; design decision

## 0　引言

软件需求分为功能需求和非功能需求[1,2]。功能需求描述了系统必须具备的功能和行为。广义上，非功能需求概括了除功能需求之外的所有其他需求，如项目需求、过程需求、接口需求、质量需求以及各类约束等[3,4]。广义的非功能需求的内涵驳杂，很难用于指导实际开发活动。狭义上，非功能需求描述目标系统必须具备的质量属性、相关的实现策略以及其他设计和实现约束[5]，体现了人们在需求分析阶段开发高质量软件系统的努力（若不特别指出，下文均指狭义的非功能需求）。显式地建模非功能需求并管理其与其他软件制品之间的追踪关系，对开发和维护高质量软件系统具有重要意义[6-8]。虽然人们对于狭义上的非功能需求内涵的理解基本一致，但对于非功能需求的外延，即非功能需求与其他功能需求和非功能需求的关系、非功能需求与设计阶段制品之间的关系、非功能需求与领域独立或领域特定的非功能需求建模知识以及典型质量属性术语之间的关系并无明确的认识。而非功能需求的外延极大地影响着软件开发方法的形态，如需求模型的结构、需求模型与设计模型之间的追踪关系以及需求分析和设计过程等。一方面，非功能需求外延的混乱源于非功能需求本身的相对性、散布性以及交互性[3]等。非功能需求的含义取决于具体的环境和具体的用户；非功能需求可能被精化为具体的可操作化的措施，往往与功能需求相互混淆；非功能需求的实现通常散布在多个不同的模块当中；非功能需求可能与其他非功能需求相互影响。另一方面，非功能需求外延的混乱源于非功能需求与部分设计制品之间界限的模糊[9]，以及非功能需求建模过程与体系结构设计过程的迭代[10]。非功能需求是驱动软件体系结构设计的重要因素，必须在体系结构设计阶段被持续

**收稿日期**：2012-05-24；**修回日期**：2012-06-30　**基金项目**：国家自然科学基金资助项目(61202019)；陕西科技大学博士科研启动基金资助项目(BJ09-13)

**作者简介**：孙连山(1977-)，男，博士，主要研究方向为需求工程、软件体系结构、软件产品线、软件安全工程(sunlianshan@gmail.com)；王今雨(1988-)，男，硕士研究生，主要研究方向为非功能需求建模、软件产品线.

精化[11]。

本文给出非功能需求的一个结构化定义以及一个概念性追踪管理框架。结构化定义形式地规范了需求阶段非功能需求与功能需求之间的关系、非功能需求与非功能需求之间的关系,以及非功能需求与质量属性等建模知识之间的关系;而追踪管理框架则规范了需求分析和体系结构设计阶段与非功能需求相关的概念和制品之间的关系。两者相辅相承,明确地刻画了非功能需求的外延,为进一步研制系统化、实用化的非功能需求建模和管理技术奠定了理论基础。

## 1　非功能需求概念的内涵和外延

区分概念的内涵(connotation)和外延(denotation)对理解并应用概念非常重要。根据韦氏词典,内涵是指一个概念所概括的思维对象本质特有的属性的总和,而外延是指一个概念所概括的思维对象的总和。

非功能需求的内涵在于其描述了目标系统必须满足的各种质量属性或约束[5],非功能需求的外延则为所有具体非功能需求的总体。非功能需求概念的内涵说明了非功能需求的作用,而它们的外延则应界定哪些软件开发制品是非功能需求,应该采用非功能需求的建模和管理技术加以处理,哪些开发制品不是非功能需求,不应采用非功能需求建模技术对其进行处理。

现有的非功能需求定义侧重于使用抽象的术语(如属性、约束、质量等)刻画非功能需求的内涵[3~5],但缺乏对非功能需求概念外延的准确刻画。例如现有文献对功能需求与非功能需求之间关系的描述就非常模糊甚至相互矛盾:

a)功能需求描述了系统应该做什么(what),而非功能需求则限制了应该怎么做(how)[12]。

b)非功能需求描述了系统的整体属性和约束,与具体的功能需求无关。可独立于功能需求获取和建模非功能需求[4,13]。

c)部分非功能需求依附于单个、部分或全部功能需求。需要在获取和建模功能需求之后才能获取和建模相应的非功能需求[14]。

d)功能需求和非功能需求相互交织,独立处理非功能需求和功能需求是研发适当的非功能需求实现技术的重要障碍[15]。

e)功能需求对非功能需求具有正面或负面的影响[3]。

f)部分需求,如访问控制,有时被看做功能需求,而有时则被认为是非功能需求[5]。

对功能需求与非功能需求之间关系的这种模糊甚至相互矛盾的认识,不能指导开发者明确地区分功能需求和非功能需求,给显式建模非功能需求并进而构造高质量的软件系统带来了挑战。而建立在不同概念基础上的研究工作往往致力于解决软件生命周期不同阶段的局部问题,很难相互集成以满足实际软件开发的需要[11]。

## 2　结构化的非功能需求定义

本章给出一个结构化的非功能需求定义,明确地刻画非功能需求在需求分析阶段的外延,并定性地分析结构化的非功能需求定义对非功能需求建模方法的影响。

### 2.1　结构化的非功能需求定义

在总结现有非功能需求定义在术语运用、分类体系、表示形式等方面问题的基础上,Glinz 将软件需求分为系统需求、项目需求和过程需求。其中项目需求和过程需求不属于非功能需求,而非功能需求属于系统需求,是目标系统的质量属性以及各类约束[5]。该定义较好地刻画了非功能需求的内涵。

本文认为系统是软件开发活动的最终制品,在需求分析阶段并不存在。直接将非功能需求刻画为不存在的系统的质量属性或约束,暗示开发者将非功能需求看做系统的全局属性,并进而独立于功能需求处理抽象的非功能需求,甚至在实现功能需求之后才考虑如何实现非功能需求。这就易于导致非功能需求和功能需求混淆、非功能需求与质量属性等领域无关或领域相关的建模知识之间的混淆、非功能需求不完整、功能需求与非功能需求之间以及非功能需求与非功能需求之间关系非常复杂而难以管理等问题[14]。事实上,功能需求与非功能需求紧密相关[1,4],在分别建模功能需求和非功能需求之后,开发者往往还须分析并建模两者之间的关系。区分功能需求和非功能需求、管理两者之间的复杂关系,为正确、高效地管理需求模型带来了挑战[7]。

从模型精化的角度来看,软件开发的过程就是从需求分析开始不断地建立各种模型并对其进行精化直至获得目标系统的过程。而不同阶段的模型,如需求分析模型、设计模型等则是对目标系统在相应阶段的抽象描述。需求分析模型是系统在需求分析阶段的抽象实现。本文采用需求分析阶段可用的、具有结构的需求分析模型代替需求分析阶段不可用的、无结构的系统黑盒定义非功能需求,并给出如下结构化的非功能需求定义。

**定义 1**　在特定环境中,功能需求描述了目标系统所必须实现的功能和行为;在内涵上,非功能需求描述了系统的质量属性或其他约束;在外延上,非功能需求包括对紧密相关的功能需求集合的质量约束以及其他设计和实现约束或对已知非功能需求的进一步精化或约束。

非功能需求对系统的实现方式的约束可分为两类,即质量约束以及其他设计和实现约束。$Q$ 表示各类软件成分(如文档、代码、运行系统)应该具有的质量属性及其子属性,如性能、易用性、安全性、可维护性等所构成的约束集合;$C$ 是各类其他设计和实现约束构成的集合,如实现语言、运行平台环境,甚至部分实现方案等,例如某些 Web 系统开发项目要求复用指定的 Web 服务器和构件中间件平台。令 $\pi$ 表示从需求到体系结构的各种可能映射所构成的集合,即对任意 $T \in \pi$,则 $T(F)$ 定义了一个实现 $F$ 中所有功能需求的系统实现方案。使用这些记号,下面的定义 2 给出了非功能需求的结构化定义的形式化表述。

**定义 2**　形式地,假设 $F$ 表示所有功能需求构成的集合,记特定环境中系统的所有非功能需求所构成的集合为 $N$,则有 $N = N_1 \cup N_2 \cup N_3$。其中:

$N_1 = \{n = \phi(A) \mid A \in {}_2{}^F; \phi \in Q \cup C\}$,$\phi(A)$ 表示由约束 $\phi$ 作用于特定功能需求集合 $A$ 之上所形成的非功能需求,即功能需求集合 $A$ 的实现方案必须满足 $\phi$ 规定的约束,若 $\exists T \in T: T(A) \rightarrow \phi$,则 $T$ 即为可行的实现方案。例如电子商务系统中的用户登录功能必须具有安全性,形成名为安全登录的非功能需求,防止恶意攻击者采用机器人暴力破解用户登录名和密码。

$N_2 = \{n = \psi(n_q) \mid \forall n_q \in N, \psi \in Q \cup C\}$,$\psi(n_q)$ 表示对非功能需求 $n_q$ 的实现方式施加由 $\psi$ 所表示的约束而构成的非功能

需求,即满足 $n_q$ 的实现方案必须同时满足 $\psi(n_q)$。如安全登录需要满足性能和易用性约束。

定义映射 $\Pi$: $N\rightarrow_2{}^F$ 计算受给定非功能需求 $n$ 约束的功能需求集合 $A$,即 $\Pi(n)=A$。若 $\exists T\in\pi: T(\Pi(n_q))\rightarrow(n_q\wedge\psi(n_q))$,则 $T$ 即为可行的实现方案。将功能需求集合 $\Pi(n_q)$ 上相互冲突的约束对记为$\langle\psi(n_q),n_q\rangle$。若存在$\langle\psi(n_q),n_q\rangle$,则 $\nexists T\in\pi: T(\Pi(n_q))\rightarrow(n_q\wedge\psi(n_q))$。

$N_3=\{n\in\Theta(n_p)\mid\forall n_p\in N\}$,映射 $\Theta: N\rightarrow_2{}^N$ 用于计算对给定非功能需求 $n_p$ 进行精化所得到的需求集合 $\Theta(n_p)$。根据定义1,精化非功能需求 $n_p$ 所得的需求集合 $\Theta(n_p)$ 均为非功能需求。若 $\forall n\in\Theta(n_p)$, $\exists T\in\pi$: $T(\Pi(n))\subseteq T(\Pi(n_p))$,则 $T$ 即为可行的实现方案。

定义1、2表明:

a)明确了非功能需求与功能需求的关系。一方面,非功能需求是对特定功能需求集合的实现方式的约束。非功能需求依存于功能需求,若没有功能需求,则非功能需求也无从谈起。实现非功能需求意味着以某种方式实现相应的功能需求,并保证功能需求的实现满足非功能需求所施加的额外约束。这也同时意味着获取和精化非功能需求之前,必须完成或部分完成功能需求建模。例如安全需求依存于其所保护的敏感功能和数据。另一方面,非功能需求不必是系统的全局约束。例如安全性往往被认为是系统的整体质量属性,但实际开发中并不是所有数据都是需要保护的敏感信息,也并不是所有服务都需要访问控制。例如,电子商务应用中,敏感的支付信息需要严格保护,而浏览商品目录以及管理购物车等功能则不需要访问控制。

b)规范了非功能需求之间的相互关系,统一将其建模为新的非功能需求。若非功能需求 $n_q$ 的实现须满足特定的约束 $\psi$,则构成新的非功能需求 $\psi(n_q)$,该非功能需求实际上是对满足 $n_q$ 的系统实现的进一步约束,而$\langle\psi(n_q),n_q\rangle$构成可能冲突的约束对。例如,基于生物特征的认证技术可用于增强用户登录子系统的安全性,但基于生物特征的认证技术的实现需要进一步满足性能以及易用性等方面的约束。这时在用户登录子系统中,安全认证与易用性就可能构成了相互冲突的约束。事实上,非功能需求的一个重要特性就是相互之间可能相互影响甚至冲突,研究指出非功能需求之间的冲突是相对的[16]。定义1、2进一步指出冲突源于两个非功能需求同时限制同一功能需求集合的实现方式,因此又是局部的。通常可通过采取适当的实现策略,如体系结构风格以及可操作的非功能需求等,避免或降低非功能需求之间的冲突。例如,登录功能必须具有安全性但也要具有较高的易用性,使用多重密码固然能够增加安全性,但要求用户记忆多重密码,易用性受到较大的影响。但若在不考虑实现成本的前提下采用基于生物特征识别的登录技术,则可能达到既安全又易用的效果。将非功能需求之间的相互影响关系转变为单独的非功能需求,有助于开发者管理这些需求到设计模型的追踪关系。

c)规范了精化非功能需求的语义。即非功能需求可被精化,但精化所得仍然是非功能需求,而不应被精化为功能需求。事实上,非功能需求最初体现了人们对目标系统的质量期望,而对非功能需求的精化仍然表达人们对目标系统的质量期望,只是更具体,更易于分析、设计、实现和测试。因此,本文认为非功能需求的精化所得应仍为非功能需求,而不应为功能需求。即一个需求是非功能需求与否,取决于引出此需求的最初关注点,而不取决于其表示为何种形式、是否可操作、是否具有贯穿性等。例如安全性是典型的质量属性,安全需求可能被精化为加/解密、访问控制等。根据定1、2,精化安全性所得的访问控制需求应被建模为非功能需求。规范非功能需求的精化语义,事实上将所有影响非功能需求的其他需求都建模为非功能需求,从而消除了功能需求对非功能需求的影响关系,有利于简化需求模型,便于分析需求模型变化对非功能需求的影响。

### 2.2 应用分析

非功能需求框架技术[4]是最为学术界所广泛引用(被引次数高达1 700余次)的非功能需求建模方法。该方法认为:a)非功能需求能够精化为功能需求(可操作的实现策略)或其他非功能需求;b)功能需求能够影响一个或多个非功能需求的实现程度;c)非功能需求可能与其他非功能需求冲突。若完全描述不同类型需求之间的这些复杂关系,即便是小型项目也将产生非常复杂的非功能需求模型。如文献[7]中给出的ice breaker系统的部分非功能需求就包含各种目标、软目标以及操作化措施等40多个,包含各类关系近50个。对用户而言,建立如此复杂的需求模型、分析各类变化对非功能需求的影响并保证模型的完整性和一致性是非常困难的。本节分析2.1节给出的结构化的非功能需求定义在简化传统的非功能需求模型结构、规范不同类型需求之间的关系、管理非功能需求建模知识等方面的作用。

#### 2.2.1 简化非功能需求模型的结构

定义1和2假设功能需求通过精化关系组织成一个层次结构,层次结构中父节点将精化为一个或多个子节点。开发者可描述不同粒度的功能需求集合的非功能需求。例如,对层次结构根节点的约束表示全局的非功能需求。

开发者可通过定义依附于不同粒度的功能需求集合的非功能需求模型,将巨大而复杂的传统非功能需求模型分解成为多个简单的子模型。每个子模型内部仅包含针对特定功能需求的多种不同类型的质量属性约束或其他设计与实现约束。依附不同粒度的功能需求所形成的层次结构,非功能需求子模型也将构成层次结构,最终形成如图1所示的模型结构。若两个功能需求是松耦合的,则依附于它们的非功能需求子模型通常也呈现为松耦合的关系,或至少有可能通过精心的设计消除耦合,如采用不同的措施实现不同功能模块的相似非功能需求。

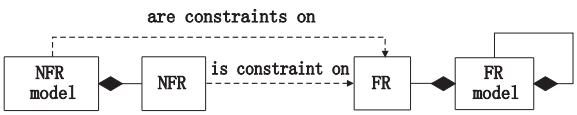


**图1 服从结构化的非功能定义需求模型结构**

#### 2.2.2 规范不同需求类型之间的关系

结构化的非功能需求定义消除了传统模型中功能需求对非功能需求的影响关系以及非功能需求之间的冲突关系,并直接将之建模为相应的非功能需求。

深入考察传统需求模型中功能需求对非功能需求的影响关系不难发现,这些所谓功能需求往往是精化非功能需求所得的操作化措施(operationalizations,它们类似于功能需求,能够被实现为具体的构件或连接子)[4]。这些操作化措施可能影响其他非功能需求 $M$,而与此同时它们也在一定程度上实现非功能需求 $N$。这时非功能需求 $N$ 与非功能需求 $M$ 就可能相互冲突,即实现非功能需求 $N$ 同时也必须实现非功能需求 $M$。显然,允许将非功能需求精化为功能需求,是导致传统需求模

型中不同类型需求之间关系复杂而难以理解的根源。

本文规范了非功能需求的精化语义,将精化抽象的非功能需求所得的操作化措施称为可操作的非功能需求,以区别于普通的非功能需求和功能需求(均可操作)。本文将这类需求称为可操作的非功能需求以表达两个内涵:a)它们源于非功能需求关注点;b)它们是可操作的,并在一定程度上实现抽象的非功能需求且同时能够被实现为具体的体系结构成分,如访问控制、加密/解密、撤销/重做等。

对于不同的项目,可操作的非功能需求是否被建模为非功能需求,将取决于谁、在何时、面向何种关注点获取并规约这些需求。例如,若用户一开始就明确地要求编辑器必须具有撤销/重做功能,那么撤销/重做就可以被认为是功能需求;若用户只要求编辑器完成基本编辑操作并具有较高的易用性需求(一种非功能需求),而后开发者和用户共同精化易用性需求得到撤销/重做需求,这时撤销/重做就应被看做非功能需求。将可操作的非功能需求简单地看做功能需求,一方面可能混淆开发关注点,另一方面导致开发者必须额外规约这些所谓功能需求对非功能需求的影响[4]。

#### 2.2.3　集成非功能需求建模知识

长久以来,开发者通常使用一些固定的术语如安全、可靠性、易用性等,概括性地描述语义相似的非功能需求及其常用实现手段。例如,使用安全性指代认证以及访问控制等。不恰当地使用这些抽象的词汇是导致需求文档中非功能需求语义模糊的重要原因[4]。但这些词汇蕴涵了理解相关非功能需求的大量知识。非功能需求建模通常需要建立并管理特定类别的、领域无关的非功能需求知识库,甚至建立并使用特定领域的非功能需求知识库。实际上,定义 2 中 $Q$ 和 $C$ 代表了领域无关或领域相关的非功能需求类型知识,如典型的质量属性以及典型的设计和实现约束等。定义 1、2 表明,只有将各种约束施加到具体的功能需求集合之上才能获得非功能需求,而约束本身不应被看做非功能需求。

## 3　概念性追踪管理框架

软件体系结构是实现非功能需求的重要场所[17,18],在软件开发的全过程中起着重要的作用[19]。非功能需求的获取和精化活动在软件体系结构设计过程中将持续进行。例如,若一些设计方案生产的中间数据能解决泄露敏感数据的信息,则有必要定义相应的安全需求保护这些中间数据。因此,实际开发过程中往往需要回答两个问题:a)非功能需求将实现为(追踪到)体系结构中的哪些成分;b)对体系结构模型所表示的系统抽象模型的质量约束以及其他设计和实现约束是否应建模为非功能需求。本章分析非功能需求在体系结构设计阶段的外延,建立非功能需求的概念性追踪管理框架来回答这两个问题。

### 3.1　以设计决策为中心的软件体系结构模型

传统的软件体系结构定义将构件以及连接子(构件之间的连接)作为体系结构模型的基本成分[18],而新一代的软件体系结构定义则围绕体系结构设计决策建立体系结构模型[20,21],认为软件体系结构是由一系列设计决策构成的集合,并给出了多种设计决策元模型。文献[21]指出体系结构模型可分为体系结构设计决策模型和体系结构制品模型。制品模型即为包含构件及连接子的传统的软件体系结构模型;而设计决策模型则包括问题(issue)、方案(solution)、决策(decision)和理由(rationale)等概念。问题源于需求,开发者针对问题制订问题解决方案,多个问题解决方案融合形成体系结构解决方案;设计师可以根据设计理由作出体系结构设计决策选择体系结构解决方案形成总体设计方案,而体系结构制品模型则是总体设计方案的一个实例[21]。

### 3.2　非功能需求概念性追踪管理框架

基于本文给出的非功能需求的结构化定义以及文献[21]中的体系结构元模型,本节建立了如图 2 所示的概念性非功能需求追踪管理框架,明确地说明非功能需求将实现为哪些体系结构成分。下面分别说明非功能需求与不同体系结构模型成分之间的追踪关系。

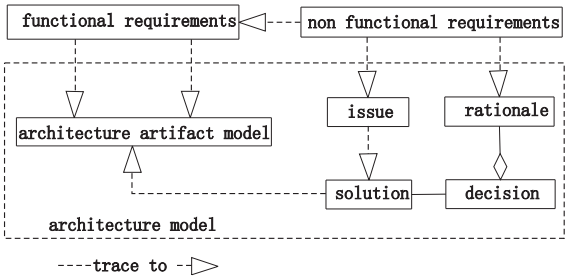


**图2　概念性非功能需求追踪管理框架**

#### 3.2.1　直接追踪到体系结构制品

在需求分析阶段,持续精化抽象的非功能需求通常最终能得到部分可操作的非功能需求。这些可操作的非功能需求能够被实现为系统中的具体构件和连接子,在一定程度上满足抽象的非功能需求[3]。例如,Web 应用系统往往需要满足安全性约束,保证数据传输安全以及限制非法用户活动系统授权等。这些非功能需求可实现为目标系统中的支持安全套接层(SSL)协议的连接子以及相关的访问控制措施[22]。

#### 3.2.2　通过功能需求间接追踪到体系结构制品

非功能需求直接或间接地产生于功能需求。而功能需求将实现为体系结构构件和连接子。很多时候功能需求和非功能需求的实现往往相互混淆无法区分,独立于功能需求管理非功能需求到体系结构的追踪关系、测试和确认非功能需求都非常困难,甚至是不可能的[7,15]。例如,性能是信息检索系统的一个非常关键的非功能需求,为了提高信息检索的性能,必须设计特殊的存储结构,采用诸如索引、缓存乃至分布式以及并行等技术。这些措施往往没有形成单独的系统构件或连接子,而是与信息检索算法混杂在一起[23]。

本文认为应将非功能需求追踪到受其约束的功能需求集合所对应的体系结构片段,如图 2 所示。在实践中,开发者可基于功能需求到体系结构模型的追踪关系管理非功能需求到体系结构模型的追踪关系,在测试被约束的功能需求的过程中同步测试非功能需求。例如,文本编辑功能通常实现为给定的文本编辑控件,但如果待复用的文本编辑控件不能满足用户关于性能和易用性的需要,则开发者可能重新开发一个新的文本编辑控件,改变文档数据结构,实现缓存、撤销/重做等可操作的非功能需求,形成一个具有高性能且易用的文本编辑控件。

#### 3.2.3　通过设计问题间接追踪到体系结构制品

实现依附于功能需求的非功能需求意味着识别可能的设计问题,设计相应的解决方案,根据适当的理由作出选择决策。体系结构设计问题(issue)是体系结构设计所必须解决的关键问题。它们是从设计者角度考虑的设计问题,不同于从用户角

度表达的原始需求[21]。因此,并不是所有软件需求都导致设计问题。事实上,绝大多数功能需求甚至一些非功能需求都不会导致设计问题。它们的实现方式是被领域特定的开发者所熟知的,体系结构设计师不必重点关注这些需求的设计实现问题。但是绝大多数设计问题都来源于非功能需求。相对于功能需求而言,非功能需求的实现方式多样,不同实现方式实现非功能需求的程度不同,且可能需要同时满足其他非功能需求,因此非功能需求往往是产生体系结构设计问题的主要来源。

例如指挥显示系统[21]具有很多非功能需求,如需要具有实时性和可维护性等,但同时对系统成本的限制较低。基于这些非功能需求可以导出包括实时的数据获取、快速的历史数据查询、降低网络和数据库负载、可视化界面以及可维护性等在内的多个设计问题(详见文献[21]第 4 节)。一个或多个体系结构设计问题最终由一个或多个问题解决方案所解决,而多个问题解决方案最终将融合在一起构成体系结构制品模型。从这个意义上说,大部分非功能需求都能够沿着从设计问题到设计方案再到体系结构模型的路径间接地追踪到体系结构模型。

#### 3.2.4 直接追踪到设计理由

虽然大部分非功能需求都能够通过功能需求或设计问题间接地追踪到体系结构构件和连接子,但有些非功能需求表现为涉众的偏好而根本无法具体实现、测试和确认[15],如要求目标系统的成本尽可能低、性能的优先级高于安全等。显然,这些非功能需求通常不能追踪到具体的设计问题,而只能作为选择设计方案的决策理由而存在。文献[21]认为设计理由是构成体系结构决策的一个重要组成部分,但没有指出设计理由的来源。事实上,非功能需求是设计理由的重要来源,是设计者权衡不同设计方案的依据。例如方案 A 实现高性能而安全性较差,方案 B 则安全性高而性能差。这时作出设计决策的理由通常是一些全局的或局部的非功能需求,如某个模块的性能的优先级高于安全,或反之。

虽然组织运营策略、计算资源限制等很多其他因素都影响系统满足非功能需求的程度,但作为可操作的非功能需求直接追踪到体系结构构件和连接子、通过功能需求和设计问题间接地追踪到体系结构构件和连接子,以及直接追踪到设计理由是从非功能需求到软件体系结构的最主要的三类追踪关系。

### 3.3 体系结构设计过程中的非功能需求

3.2 节回答了非功能需求将实现为哪些体系结构成分的问题;本节讨论针对体系结构模型乃至实现模型的各类约束是否应建模为非功能需求。

一方面,软件需求描述用户的需要,是用户和开发者沟通的重要媒介,是评价目标系统成败的依据。因此,尽管对体系结构模型、实现模型、部署模型乃至运行模型的约束同样足以构成非功能需求,但若这些对系统解决方案的约束是由开发者基于系统的部分解决方案提出来的,不作为用户评价目标系统是否成功的依据,则不宜将其视为非功能需求,而更适于将其视为体系结构设计问题或设计方案。例如,若设计人员提出采用 Java 语言开发目标系统并将其部署到某个开源中间件上,则其仅仅是一个可选择的设计方案,而不是必须被满足的用户需求。另一方面,对于有用户参与的联合开发团队,在设计乃至实现阶段发现的对系统设计模型的约束,若由用户提出并确认作为评价目标系统是否成功的依据,则可以认为是非功能需求。例如,对于上述设计方案,若开发团队中的用户代表要求采用 Java 语言开发目标系统,并将其部署在公司维护人员熟知的可信的 Jboss 开源中间件上,则该设计方案就转变成了一个非功能需求。

很多体系结构层的设计问题乃至实现方案若由用户在需求分析过程中提出并规约,则构成需求分析阶段的非功能需求或实现抽象非功能需求的可操作的非功能需求。类似地,在实现阶段、部署阶段甚至运行阶段,对相应的实现模型、部署模型和运行模型的约束也可能被建模为设计问题或非功能需求。

## 4 相关工作及讨论

### 4.1 相关工作

相关工作来自非功能需求建模研究领域以及非功能需求追踪性研究领域。首先,完整、准确的非功能需求规约是开发高质量软件的重要前提。很多研究者给出了非功能需求的定义和相应的建模方法。如 Glinz[5]对非功能需求的定义进行了比较完整的搜集和分析,将非功能需求定义为系统需求的一部分,即系统的属性和约束。Chung 等人[3,4]提出的非功能需求框架技术被学术界广泛引用。但这些定义均侧重于描述非功能需求的内涵,而未能准确地刻画非功能需求的外延,即非功能需求与功能需求、非功能需求与其他非功能需求、非功能需求与建模知识,以及非功能需求与设计制品之间的区别和联系。而相应的建模方法倾向于将非功能需求看做系统的全局属性或约束,允许非功能需求精化为功能需求,导致功能需求和非功能需求相互混淆以及复杂的需求模型[3,4,7]。本文给出的结构化非功能需求定义则明确地刻画了非功能需求的外延,规范了非功能需求与功能需求之间的关系,为简化需求模型奠定了基础。

一些研究者讨论了追踪非功能需求的重要性以及实际软件项目中缺乏对非功能需求的追踪管理的原因[6~8]。其中 Cleland-Huang 等人[6,7]提出了以目标为中心的非功能需求追踪管理技术,即在非功能需求框架技术[3,4]的基础上研究功能需求变化对非功能需求的影响。然而非功能需求框架技术对功能需求与非功能需求之间的关系的定义往往导致需求模型非常复杂而难以分析。Kassab 等人[8]则给出了一个非功能需求追踪关系元模型,提出了所谓相关点(association point)的概念。相关点可能是功能需求、外部资源甚至是软件项目本身。非功能需求必定与一个或多个相关点相关,而与同一相关点相关的非功能需求则可能相互影响。这一观点与本文相同。但 Kassab 没有区分非功能需求的内涵和外延,也没有从外延的角度定义非功能需求,对非功能需求追踪关系的分析也仅停留在需求分析模型内部。相对地,本文关注于狭义的非功能需求,区分了非功能需求的内涵和外延,建立了非功能需求的概念性追踪管理框架,管理非功能需求与不同体系结构模型成分之间的追踪关系。

### 4.2 讨论

综合分析前文结构化的非功能需求定义以及概念性追踪管理框架,可以给出如下结论,即一方面总结本文的贡献,另一方面可以作为帮助开发者区分功能需求和非功能需求、关联非功能需求和设计模型的检查列表。

a)非功能需求直接或间接地产生于功能需求,描述了对功能需求实现方式的约束,或描述了对已知非功能需求的进一步精化或约束。

b)在需求工程过程中,用户和需求分析人员须首先获取功能需求,才能具体分析并建模非功能需求。

c)非功能需求依存于不同粒度的功能需求集合,而不必是对整个系统的全局约束。

d)非功能需求之间可能相互影响甚至冲突,但这种影响和冲突是相对的[16],甚至是局部的,特定于它们所依附的功能需求。

e)将非功能需求之间的相互影响关系转变为单独的非功能需求,有助于开发者管理这些需求到设计模型的追踪关系。

f)精化非功能需求所得必然是非功能需求。

g)非功能需求及其精化所得的可操作的非功能需求与设计问题和设计方案从内容到形式上都可能十分相似,但区别在于前者可作为用户评价项目成败的依据,而后者则不然。

h)类似于功能需求,可操作的非功能需求将实现并追踪到具体的体系结构构件和连接子。

i)依附于功能需求的非功能需求将间接地追踪到实现相应功能需求的体系结构构件和连接子。

j)蕴涵体系结构关键设计问题的非功能需求将间接地追踪到针对该设计问题的设计方案,即一系列构件或连接子或体系结构风格。

k)部分非功能需求将直接追踪到体系结构设计理由。

## 5　结束语

本文提出了一个结构化的非功能需求定义,即在内涵上,非功能需求描述了系统的质量属性或其他约束;在外延上,非功能需求包括对紧密相关的功能需求集合的质量约束以及其他设计和实现约束或对已知非功能需求的进一步精化或约束。该定义既体现了非功能需求作为系统的属性和约束的概念内涵,又明确地刻画了非功能需求在需求分析阶段的外延,为区分功能需求和非功能需求,并简洁、一致地建模两者之间的关系奠定了坚实的理论基础。本文进一步分析了体系结构阶段实现非功能需求的不同方式,建立了一个跨越分析和设计阶段的概念性非功能需求追踪管理框架。该框架说明了功能需求和非功能需求与各种体系结构层概念制品之间的关系,明确地刻画了非功能需求概念在设计阶段的外延,为研究系统化、实用化的非功能需求建模和管理技术奠定了理论基础。未来的研究工作包括在本文提出的非功能需求的结构化定义,以及概念性追踪管理框架的基础上,扩展典型的功能需求和体系结构建模技术,建立实用的非功能需求建模和追踪管理技术,结合现有文献中的案例和实际案例展开对比研究和应用研究。

**参考文献:**


[1] SOMMERVILLE I. Software engineering[M]. 8th ed. New Jersey: Pearson Education. 2006.

[2] WIEGERS K. Software requirements[M]. 2nd ed. Washington: Microsoft Press,2003.

[3] CHUNG L, NIXON A, YU E, *et al*. Non-functional requirements in software engineering[M]. [S. l.]: Kluwer Academic Publishing, 2000.

[4] CHUNG L, LEITE J C P. On non-functional requirements in software engineering[M]. Berlin, Heidelberg: Springer-Verlag, 2009: 363-379.

[5] GLINZ M. On non-functional requirements[C]// Proc of the 15th IEEE International Requirements Engineering Conference. New Jersey: IEEE Press,2007: 21-26.

[6] CLELAND-HUANG J, SETTIMI R, BENKHADRA O, *et al*. Goal-centric traceability for managing non-functional requirements[C]// Proc of the 27th Internation Conference on Software Engineering. New York: ACM Press,2005:362-371.

[7] CLELAND-HUANG J, MARRERO W, BERENBACH B. Goal-centric traceability: using virtual plumblines to maintain critical systemic qualities[J]. IEEE Trans on Software Engineering, 2008, 34(5): 685-699.

[8] KASSAB M, ORMANDJIEVA O, DANEVA M. A traceability metamodel for change management of non-functional requirements[C]// Proc of the 6th International Conference on Software Engineering Research, Management and Applications. Washington DC: IEEE Computer Society,2008:245-254.

[9] REMCO P, HANS V. On the similarity between requirements and architecture[J]. Journal of Systems and Software,2009,82(3): 544-550.

[10] TAYLOR R N, MEDVIDOVIC N, DASHOFY E M. Software architecture, foundations, theory, and practice[M]. New Jersey: John Wiley,2010.

[11] MATOUSSI A, LALEAU R. A survey on non-functional requirements in software development process,TR-LACL-2008-7 [R]. Paris:Laboratory of Algorithmics, Complexity and Logic (LACL) University Paris 12 (Paris East), 2008.

[12] 杨放春,龙湘明. 非功能属性研究[J]. 北京邮电大学学报,2004,27(3):1-12.

[13] BARBACCI M, LONGSTAFF T H, KLEIN M H, *et al*. Quality attributes, CMU/SEI-95-TR-021[R]. Pittsburg:CMU, 1995.

[14] JACOBSON I, BOOCH G, RUMBAUGH J. The unified software development process [M]. New Jersey: Addison Wesley,1999.

[15] BURGE J, BROWN D. NFRs: fact or fiction? WPI-CS-TR-02-01 [R]. Worcester:WPI,2002.

[16] MAIRIZA D, ZOWGHI D. An ontological framework to manage the relative conflicts between security and usability requirements[C]// Proc of the 3rd International Workshop on Managing Requirements Knowledge. New Jersey: IEEE Press,2010:1-6.

[17] BASS L, CLEMENTS P, KAZMAN R. Software architecture in practice.[M]. 2nd ed. New Jersey: Addison-Wesley, 2003.

[18] 梅宏,申峻荣. 软件体系结构研究进展[J]. 软件学报,2006,17(6):1257-1275.

[19] 杨芙清,吕建,梅宏. 网构软件技术体系:一种以体系结构为中心的途径[J]. 中国科学:E 辑,2008,38(6):818-828.

[20] JANSEN A, BOSCH J. Software architecture as a set of architectural design decisions[C]//Proc of the 5th Working IEEE/IFIP Conference on Software Architecture. New Jersey: IEEE Press, 2005:109-120.

[21] 崔晓峰,孙艳春,梅宏. 以决策为中心的软件体系结构[J]. 软件学报,2010,21(6):1196-1207.

[22] SUN Lian-shan, HUANG Gang, SUN Yan-chun, *et al*. An approach for generation of J2EE access control configurations from requirements specification[C]//Proc of the 8th International Conference on Quality Software. Washington DC: IEEE Computer Society, 2008:87-96.

[23] GROSSMAN D, FRIEDER O. Information retrieval: algorithms and heuristics[M]. Dordecht:Springer,2004.

[24] 肖赛,崔晓峰,孙艳春,等. 一种软件体系结构设计决策的建模工具[J]. 计算机科学,2009,36(8):161-164,173.