面向不确定需求的适应性软件体系结构设计

付 赟 李敏强 陈富赞
(天津大学管理学院 天津 300072)

摘 要 需求不确定性是软件开发的主要风险来源之一。首先讨论了不确定需求的含义、产生的原因及其影响;然后分析了需求与软件体系结构的关系;指出了软件体系结构对满足需求和控制软件风险的重要性;提出了在不确定需求的情形下,需求分析与体系结构设计的协同建模方法及适应性软件体系结构的设计思想;论证了不确定需求情形下软件体系结构的演化过程,以保障生命周期内软件产品质量的开发思路。

关键词 不确定需求,软件体系结构,适应性设计,系统演化
中图法分类号 TP311 **文献标识码** A

Adaptive Software Architecture Design Oriented to Requirements Uncertainty

FU Yun LI Min-qiang CHEN Fu-zan
(School of Management, Tianjin University, Tianjin 300072, China)

Abstract Requirements uncertainty is recognized as a major source of risk to software development process. A new idea about software development, called the adaptive software architecture design, was proposed in this paper, which offers a feasible way to guarantee the quality of software product during the life-cycle. Firstly, the concept of requirement uncertainty was defined; its causes and impacts were discussed. Secondly, the relationship between the software requirement and the software architecture was analyzed, and the adaptive software architecture approach ws put forth to handle the variable users' requirements and control process risks. Finally, the evolution process of software architecture was characterized with regard to the requirements uncertainty, and a collaborative modeling method was presented to achieve adaptive design between requirements and architecture.

Keywords Requirements uncertainty, Software architecture, Adaptive design, System evolution

1 引言

20世纪80年代以来,我国软件产业呈现出迅猛、广泛、深化的发展趋势,全球范围内的经济和社会的信息化更催化了对软件产品的巨大需求。随着软件产品规模的不断增长,其复杂性也在不断增加。软件虽然不是采用传统物理方法进行制造,但仍是以具体应用目的为导向被"设计"出来的,且其设计贯穿于开发和维护的全生命周期过程。软件体系结构是软件整体的关键内容和最重要的质量载体,软件设计也是以软件体系结构为核心而展开的。如果把软件设计过程看作是一个复杂系统工程,那么软件体系结构必然随着需求或环境的改变而演化。这个演化的过程也是"设计产物"最重要的特征。

根据ANSI/IEEE Std 729-1983[1],软件质量被定义为"与软件产品满足规定的和隐含的需求的能力有关的特征或特性的全体"。国际化标准组织ISO在ISO/IEC9126[2]中将软件质量定义为"反映软件产品满足规定需求和潜在需求能力的特征和特性的组合"。不难看出,软件需求是度量软件质量的基础。衡量软件质量的标准,根本就在于软件是否能够满足用户的需求和在多大程度上满足用户的需求。在需求获取过程中,用户往往会遗漏一些隐含的需求,或者是随着软件开发过程变更需求[3]。据CHAOS报告[4],仅仅20%的软件项目能够按预定工期结束,主要原因是非预期的需求变更,特别是需求形态的变化和增加新的需求。既然需求本身是不确定的,如果软件只满足那些精确定义了的需求,而没有满足隐含需求和变更的需求,软件质量也不能得到保障。

另外,软件开发过程中进行的工作可谓是"此呼彼应"。这一环节的工作会影响到其他若干环节,其他环节的工作又反过来影响它。开发过程中的每一环节都决定着最后软件产品的质量。需求作为衡量软件质量的唯一标准,贯穿于软件开发的始终。但是,传统软件设计所依赖的确定性需求只是对问题的静态描述,没有充分考虑软件设计和开发过程中发生的需求变化或变更。研究显示[5],在整个软件开发过程中,70%的错误是由需求和软件设计阶段引入的。要保证软件产品的质量,提高需求分析和软件设计的质量就成了关键。我们认为,软件产品演化的根本在于需求变化。如何在不确定需求的条件下构建出适应性软件体系结构,并最终保证高质量的软件产品,将是软件需求分析、系统设计、开发和实施的

到稿日期:2010-01-26 返修日期:2010-04-15 本文受国家自然科学基金(No. 70771074)资助。
付 赟(1982-),女,博士生,主要研究方向为信息系统等,E-mail:fy8266@163.com;李敏强(1965-),男,博士,教授,博士生导师,主要研究方向为进化计算、数据挖掘和机器学习等;陈富赞(1972-),女,博士,副教授,主要研究方向为知识发现与数据挖掘。

关键成功因素。

本文试图从理论上探寻在不确定需求情形下，利用需求分析与体系结构设计的交互性，进行需求分析与体系结构设计的协同建模，以支持适应性软件体系结构的演化，最终提高生命周期内软件产品的质量。

2 不确定需求特征

随着系统科学理论的发展，对不确定性的研究越来越多，软件工程中的需求不确定性也引起了广泛关注[6-8]。

2.1 不确定需求的含义

IEEE 标准[9]把需求定义为“用户为了解决问题或达到某些目标所需的条件或能力”。《现代汉语词典》中“确定”是明确、肯定的意思；所谓“不确定”即不明确、不肯定。把“不确定”和“需求”的定义相结合，似乎能够说明不确定需求的意思，但是仍不足以表示出它的含义。可参考 Free On-line Dictionary of Computing 关于“需求”词目的一条补充解释：“几乎所有软件的一个共性就是需求会在整个生命周期中发生改变”[10]。

我们认为，不确定需求主要包含了以下几个方面：

(1)不一致需求

知识、信息的不一致普遍存在于客观世界之中，软件需求又是来源于用户或其他利益相关者对现实世界的认识和理解。因此，从可能不一致的信息中总结的需求也可能存在不一致。此外，需求往往是一个源自诸多视角的综合产物，不同用户有不同的观点和兴趣，因此需求出现不一致也就不可避免。

(2)不完整需求

软件开发之初，由于用户对问题认识的不完全，总会有一些需求被遗漏，这些都会造成需求的不完整。

(3)不准确需求

用户用语言来描述的需求总是模糊的。通常来说，描述需求的语言多是定性的，更多地是表示需求被满足的程度。

(4)不稳定需求

软件开发过程中，需求不是固定不变的。所谓不稳定需求，就是指需求会随开发进程而改变。这种改变包括对原有需求的增加、删除或者修改。Jones[11]将需求的不稳定性视为软件开发的一个顽疾。Isaac Asimov 给出了对需求更为形象的断言：“唯一不变的就是变化”[12]。

软件系统演化是软件生命周期内的一个重要特征。软件产品的演化根本在于需求变化。需求的变化必然引起整个软件系统设计与实现的相应改变，于是这个过程又形成了一个新的与软件开发有关的周期。我们关注软件系统演化，在不确定需求中更重视不稳定需求，即需求变化。

2.2 不确定需求产生的原因及影响

需求分析阶段的工作就是为了得到能够准确反映用户需求的、确定的规格说明。但是一直以来，完全明确的需求几乎是一项不可能完成的任务。大部分的需求规格不能完全、准确地反映用户需求，而且很难应对需求的变化。不确定需求是软件开发过程中面临的最大问题，它直接关系着需求规格的完整性、一致性、可追溯性和可度量性[13]，也是影响软件质量的根本原因。

首先，认识会带来需求的不确定。需求是用户对客观世界的一种主观描述，而人类对客观世界的认识又是不完全的，这是导致需求不完整、不一致的根源。同时，人类认识客观世界的能力也会受其经验、知识等影响，这就使得用户在对客观世界的认识过程中也是不确定的。Jones[11]把软件需求的获取过程比作是在迷雾中远行，只有迷雾渐渐散去，才能更加看清前方的道路。这告诉我们，软件需求不可能在开发初期就完全确定，而是随着开发的进行不断清晰。

其次，描述会带来需求的不确定。需求是用自然语言来描述的用户或其他利益相关者对系统的要求。通常，对需求的描述多是定性语言，其用来表示需求被满足的程度。因此，不同的人对定性语言的不同理解会导致对需求理解的偏差，从而导致不确定的需求。

再次，变化会带来需求的不确定。客观存在不是一成不变，而是运动的。需求规格也一样会随着时间、空间的改变而有所变化。Brooks 认为[14]“软件开发过程是艰难的，没有银弹”。而 Berry 更加明确地指出[15]之所以不存在“银弹”，就是因为需求发生了改变，需求的不断变化使开发人员面临巨大的困境。Boehm[16]也认为 21 世纪软件工程将面临快速变化带来的强有力挑战，这就要求组织以更快、更合适的方式来适应这种变化。

不确定的需求存在于任何软件开发环境中。它的产生原因或有不同，但它带来的影响却是相似的——不能实现项目开始时制定的目标(包括开发周期、开发成本及软件质量)。不确定需求对软件项目的影响有以下几个方面[8]：不适当的项目计划、项目的持续变化、项目延期、项目缺陷、配置问题以及软件产品不能满足用户的需求等。

关于需求，也有学者指出[17]风险就存在于需求的语境之中(Arisk is a requirement in context)。软件项目的风险同质量一样，都是全生命周期的。每个 IT 项目管理者，无论自己意识与否，都被 IT 项目背后涌动的巨大风险所“胁迫”[18]。不确定需求不可避免，需求变化也被看作是软件开发的最主要风险来源之一[14]。为了提高软件产品质量，越来越多的学者把目光投向了软件开发的源头——需求。但由于需求分析过程的独特性和重要性，软件需求迄今仍是软件工程领域最复杂、最困难的问题之一。

2.3 不确定需求的度量

度量是一切工程学科的基础。随着软件产业的迅猛发展，技术和管理上的复杂性都给软件工程带来了巨大的不确定性和风险，因此客观地评价软件产品是否满足相应的评价标准就显得极为重要。

度量作为一种方法应该是软件开发过程中的重要组成部分，而不是作为一种附属物存在。比如，ANSI/IEEE Std 610.12-1990[19]关于软件工程的定义就表述为：“将系统化的、规范的、可度量的方法应用于软件开发、运行和维护的过程。”

软件度量是对软件开发项目、过程或资源的一些内在和外在性质进行量化和预测，其目的在于对软件开发过程进行有效的管理[20,21]。从经济学的角度来看，软件开发中的错误发现得越晚，则挽救该错误及其影响的代价就越大[22]。需求分析是软件开发的首要阶段，因此软件度量首先就要关注与需求相关的度量。Ali[23]从需求的角度对软件质量的度量指标进行了总结，并指出需求度量是软件开发过程中的重要组

成部分。

我们认为，在软件开发过程中对不确定需求进行抽象和度量，对软件质量的控制、评价和预测等有着重要的意义。首先，度量的目的是为了更好地控制开发过程，对不确定需求进行量化可以为开发过程提供必要的反馈信息。其次，度量的结果使得我们可以对开发过程和产品的质量进行评价，从而做出正确的判断。最后，度量的结果可用于预测某些属性的未来表现，为评价或决策提供必要的信息。

2.4 国内外对不确定需求的研究

关于用户需求的获取、建模、验证和变化管理等专题研究，起始于 1993 年 IEEE 组织的两年一次的需求工程国际讨论会(International Symposium on Requirements Engineering, ISRE)。软件工程学科定义了软件需求工程子学科(Requirement Engineering, RE)[24,25]，以更好地实现用户需求的获取、建模、验证和变化管理等。Springer-Verlag 也于 1996 年创办了“Requirements Engineering”学术期刊。目前，在需求工程中关于需求变化的研究已经取得了初步成果。

Strens 等[26]指出，软件需求变化的风险不仅体现在变化本身对设计的影响，还包括由变化所引发的不可预知的一系列影响。他认为分析需求变更要从敏感性分析和影响分析两方面入手。O'Neal 和 Carver[27]以需求的可追溯性为基础对需求变化类型进行优先排序，并对开发过程的需求变化进行了影响分析。Schooff 等[28]针对螺旋式和原型法软件开发过程，提出了软件评估的两个动态模型：线性正态动态模型(linear-normal dynamic model)、非线性状态和观察方程(nonlinear state and observation equations)，其用于解释需求变更、系统设计及其它策略的动态性及对项目目标(成本和进度)的影响。Burgess 等[29]采用影响图来描述软件需求变化，并识别关键需求的风险因子。Cheng 等[30]针对需求和环境的变化，特别强调了软件系统适应性的重要性，并指出通过自我管理(self-management)来改善软件系统的适应性，理想情形是软件系统可以识别需求变化和环境变化并实现自我进化。

在国内学者的研究中，王青、李明树[31]基于统计过程来控制度量、分析和限制需求变化，以保障软件过程能力和产品质量。熊伟等[32]从考虑需求向软件设计过程映射的角度，采用模糊层次分析法来定量判断客户需求的重要性，从而实现最大的顾客满意度。黄蒙等[33]考虑了针对需求变更和环境改变的迭代开发过程，建立了基于项目计划和风险管理的需求优先级评价模型，并结合多方系统参与者之间的协商，动态调整需求优先级，强化迭代开发过程对需求变化的管理和风险控制。王映辉[34]从软件变化控制和软件演化的角度研究了功能需求变化，提出了需求变化表示和追踪方法及其在软件体系结构中的波及影响，分析了在软件生命周期中软件变化转播的一些重要性质。

通过对相关文献的分析我们发现，对不确定需求的研究仍然是定性多于定量。当然，这与需求分析活动本身的特点是分不开的。需求是一种用自然语言表达用户或其他利益相关者对系统要求的主观描述。同时，需求伴随着软件开发的全过程，需求分析过程是个不断反复的、螺旋型进程。这些都为量化需求的变化带来了困难。但如何量化需求变化的关联性、动态性或时序性，的确是一个值得我们深入探索的问题。而关注需求风险，设计出最终满足用户需求的软件产品，就需要开发人员设计的软件系统是一个可适应需求变化的系统。

3 软件需求与软件设计

软件体系结构对需求工程具有积极作用已成为一种共识[35]。要在不确定需求条件下构建出适应性的软件体系结构，软件需求与软件设计之间的沟通和协调就显得尤为重要。软件的演化性取决于需求波动情况下体系结构的稳定性[36]，因此软件需求与软件设计的关系就成了处理不确定需求条件下软件体系结构演化的基础。

3.1 软件体系结构

软件设计过程是以软件需求规格(软件系统信息、功能和行为的描述)为基础，来完成软件体系结构(架构)设计、数据结构设计和程序过程设计。软件设计是软件产品质量和可信性形成的源头，只有确保软件设计质量，才能保障最终软件产品的质量和可信性的获得，其中软件体系结构设计又是软件设计的核心。

由于需求和应用环境的变化及软件技术的不断进步，软件系统变得日益复杂，开发投入、周期和质量更加难以管理。为了控制日益增长的软件过程风险，保障软件系统的可信性，软件体系结构(software architecture, SA)逐渐成为软件工程的一个热点问题和关键技术[37,38]。软件体系结构是软件系统的组织结构和拓扑结构(组成元素及联系)、系统属性和行为的抽象描述，提供了软件设计和实施决策的依据。它是用户需求与软件系统之间的一个桥梁，是需求变化管理、软件设计与复用的平台，也是软件过程多方参与者交流的开发过程平台、软件编码和测试的平台[39]。软件体系结构在整个软件系统和过程中的核心地位越来越清晰，逐渐形成了在软件生命周期中以软件体系结构为核心的开发方法[38,40]。我们所关注的不确定需求对软件设计的影响，主要就是指软件体系结构。

3.2 需求与设计的关系

我们认为，需求阶段的需求规格描述的是问题空间，而设计阶段的体系结构描述的是解空间。问题空间关注的是系统的行为，包括功能性需求和非功能性需求；解空间关注的是系统的结构以及关于系统的设计决策。于是，需求与设计的关系可以用图 1 来描述。

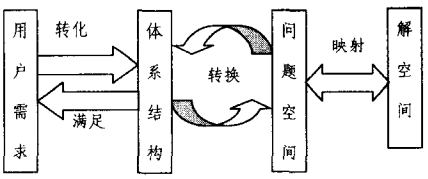

图 1 需求与设计的关系

传统的软件开发模型将需求分析与体系结构设计视为独立的两个阶段。如经典的“瀑布模型”[41]就将软件开发的生命周期分为软件计划、需求分析与定义、软件设计、软件实现、软件测试和测试维护。这种划分直接将需求分析和体系结构设计分别置于问题空间和解空间。问题空间只需要描述系统需求，解释“做什么”，而解空间必须明确“如何做”。不可否认，问题空间和解空间的划分使得我们对问题的理解更为清晰，但这种分割却将问题空间和解空间看成两个独立的领域。两个领域分别使用不同的术语、概念和符号标记来建构模型，从而使得两个领域的差异形成了概念上的差距。同时，这种

分割也导致针对二者的技术、方法等自成体系，难以统一。正如 Grünbacher 指出的那样[45]，需求和体系结构之间的沟壑，一部分就是由于各自使用适合自身的不同的术语和概念去建模而造成的。

瀑布模型把上阶段的结果作为下阶段的输入，只有当上阶段工作完成后才能进入下阶段。可现实是我们很难在各阶段之间划出一条明显的界限。需求分析将于何时何处结束，体系结构设计于何时何处开始，这本身就不可能有非常精确的答案。问题空间毫无疑问地决定了解空间，但是解空间也可能反过来影响问题空间。因此，对于问题空间的分析一定不能脱离解空间而单独存在。

问题空间和解空间的关系应该是联系的、互补的，而不是完全割裂的。将需求视为问题空间、体系结构视为解空间，这种划分是出于对问题理解的必要，却不代表需求和体系结构之间的绝对差异。Swartout 和 Balzer 指出[42]软件的需求规格说明(Specification)和软件开发执行过程(Implementation)将不可避免地会交织在一起。这与我们的观点也是一致的。将需求和体系结构视为独立的阶段，对开发过程中使用的方法、技术等都有很不利的影响，这种分割会在整个软件开发过程中产生很大的问题[43-45]。而且，从需求到体系结构设计的过程仍然是一种非正式的，缺乏必要的指导原则和有效的方式，这在很大程度上使得软件产品最终不能满足用户的需求[43-46]。

为了协调软件需求与软件设计之间的关系，Nuseibeh 提出了“双峰模型”(The Twin Peaks Model)[47]，其基本思想就是需求与体系结构设计同时进行，如图 2 所示。该模型不再将软件开发看成是线性的“瀑布模型”，而将开发过程视为螺旋型的，需求分析和体系结构设计都是随开发的进行而不断清晰的过程。

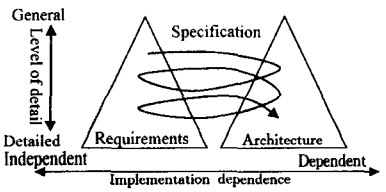

图 2　双峰模型

对于软件开发的这种螺旋式进程，Kozaczynski 指出[48]这是由结构性需求(Architectural Requirements)、开发风险和体系结构三者共同作用的结果。只有那些结构性需求才会对体系结构的构建产生真正意义上的影响。在不确定需求条件下进行软件开发，根本就在于把握需求、体系结构和风险之间的相互作用。图 3 描述了这三者之间的关系。

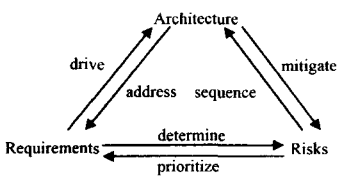

图 3　需求、风险和体系结构之间的关系

除了上面提到的“双峰模型”，Jackson 提出的“问题框架”(Problem Frames)[49]以及 Grünbacher 等提出的 CBSP 方法[45]都对需求和体系结构之间的关系进行了分析和研究。Diallo 等[50]也从实证角度对“需求-结构”(Requirements-Architecture Gap)的研究进行了比较和总结。作者认为，虽然对“需求-结构”的研究日益增多和深入，但其中的绝大部分仍然只是单独针对需求或者体系结构。

毫无疑问，对于任意的产品来说，当发生变化时，设计人员都需要在“完全不变”和“完全改变”这两种极端情况的中间做出权衡。这个选择依赖于特定环境因素、内部资源限制、开发过程的决策有效性等。在这个过程中应用的所有决策辅助方法都是为了实现用最小的代价满足最大的期望。

3.3　面向不确定需求的软件体系结构设计

前面提到，对需求的分析是否完善，很大程度上决定了软件质量的优劣，需求因素是软件开发中最大的风险来源之一。于是，要提高软件开发的质量，需求自然而然就成为了一个重要的议题。但是，简单的就事论事可以治标却难以治本。只从需求出发，仅仅改善需求分析过程，而忽视需求与体系结构之间的联系，抛开解空间对问题空间的影响，仍然不能有效地解决软件开发过程中的不确定需求问题。

软件体系结构的构建过程其实也就是设计的选择决策过程。这个选择不仅会影响到它能满足的用户需求，也可能影响到它不能完全满足的需求，甚至对某个特定问题的解还会引入新的问题，从而形成新的需求。需求与体系结构不能被人为地割裂，它们是相互联系、互为因果的。这也与 Remco de Boer 等[51]的观点不谋而合。

不确定需求的存在已不可改变，但在不确定需求条件下对软件体系结构的研究却能从以下几个方面带来好处：

(1)理解性(Understanding)

软件体系结构是对系统的较高层次抽象。它不但揭示了系统设计的高层原则，也可以帮助我们更深刻地理解需求。

Mead 向我们呈现了两种需求分析场景[35]：一种是在完全不与体系结构设计人员沟通的情况下将需求过分细化，导致软件产品设计成本激增而且难以满足需求，最后再返回到需求阶段，花费极大的代价。另一种是不考虑体系结构的实现可能，只关注用户的需求，使得需求分析的结果过于理想化而不具可行性。这两种都属于不考虑体系结构进行需求分析的情况。对于第一种，可以在需求分析的早期就由体系结构设计人员参与其中，从体系结构设计的角度保证需求分析结果在一个合理的抽象层次上，不致于过于细化也不会太过抽象。对第二种，可以通过体系结构的设计向用户提供原型(prototype)，这样既有利于开发人员和用户理解需求，又不会出现需求不可行的状况。

在需求分析阶段综合考虑软件体系结构所获取的需求比单纯的需求分析更加有效、一致、完整和可行。

(2)演化性(Evolution)和重用性(Reuse)

需求规格描述是系统功能的一种静态表征，软件的开发过程却是设计人员创造性的体现。软件的演化一方面来源于用户对问题空间的理解，另一方面来源于用户对解空间的要求。演化是必然的，重要的是如何保证软件产品不会在需求变化的情况下不堪一击。软件体系结构的一个重要作用就是揭示系统有可能演化的方面。在需求阶段考虑软件体系结构可预先定义系统的演化参数，及早明确那些可能的演化，并对未来变化的趋势和影响进行分析和预测，以评估变化可能引起的成本及相关的维护费用。

由于软件体系结构是通过组件(或构件)及组件之间的关

系来进行描述，因此可以在保持功能的条件下，通过改变连接件机制实现演化和重用。软件体系结构改变了软件的开发模式，这种模式的最大特点就是可替换性。软件体系结构的开发侧重于对组件的分割和组合，这与软件重用是一致的。而且，软件体系结构的描述支持不同层次上的重用。软件体系结构的应用不仅提高了重用的抽象层次，而且提高了重用的粒度。

(3)管理性(Management)和分析性(Analysis)

软件体系结构提供了对系统进行各种分析的平台，包括系统的一致性检验、是否满足软件质量要求、独立性分析以及特定领域的领域分析[52]等。

对软件开发项目的管理贯穿于整个软件开发过程中。实践表明，那些成功的软件项目都是把构建了可行的软件体系结构作为开发过程中的重要里程碑[52]。对软件体系结构做出的重要评价，会让我们对需求、执行策略，甚至潜在的风险都有更清晰的认识。

(4)决策性(Decision-Making)

软件产业的迅猛发展在很大程度上是由于市场对软件的需求。同时，在市场利益驱动下，开发人员也不得不面临缩短发布周期、降低开发成本的问题。只有既缩短开发周期又保证软件质量，才能获得高投资回报率，进而赢得市场。这些都是在需求分析阶段不可回避的问题。尤其是商业现货软件(Commercial-off-the-shelf，COTS)产品的大量涌现，使得软件生产向软件消费发生着悄然的转变，越来越多的软件开发者会从 COTS 中选择合适的产品来支持自己的开发过程。已发布的软件产品会因为需求的变化而不断投放新的版本，这类产品将会面临“重用”(Reuse)的选择，原有版本中内容或是直接应用于新版本中，或是修改之后再应用，或是不适合需要重新开发。在新产品的开发过程中，部分功能实现可以由第三方产品，即 COTS 实现，这时就必须在购买和自主开发之间做出选择。

若软件开发与市场的联系越来越紧密，不确定需求对软件项目计划的影响也越来越大，则软件体系结构应对市场影响的优越性也越来越明显。软件体系结构有助于软件开发方建立有效的集成生产机制以缩短开发周期，降低开发成本。而集成开发组件和购买组件正是软件生产向软件消费的重要体现。

4 适应性软件体系结构

软件的设计目标是高质量的软件产品，软件的质量以满足用户需求为衡量标准，软件设计又以软件需求为基础。需求本身作为一种偏向于主观描述的产品特征或特性，具有不确定性。不确定需求的演化必然导致软件体系结构的设计的相应改变，从而影响到软件的整体质量。

从本文的图 1 看出，在软件开发中，需求与设计之间的一致性是通过二者的转换和映射来实现的。需求与设计的转换和映射是软件开发中的核心问题。把握“需求到体系结构”这个过程，保证需求模型和设计模型之间的映射一致性，就成了基于软件体系结构开发的关键。这要求我们充分考虑到需求分析与体系结构设计的交互性，在不确定需求条件下设计出适应性软件体系结构。

系统适应本是由环境变化引起，离开变化讨论适应性问题将毫无意义。需求变化对软件设计的影响，以及软件设计对变化的适应能力，与体系结构有着密切的关系。软件体系结构也成为处理适应性问题的基础。

综合对软件适应性的若干代表性观点[53,54]，并结合本文研究的实际，我们采用文献[54]中对适应性的定义：适应性是指当软件生存环境发生变化时，软件实体的结构或行为可以随之改变并满足新环境需要的特性。适应性分两个层次：一类是实体本身能够适应环境的变化，称为完全自适应；另一类需要对其进行修改和配置后才能适应环境的变化，称为可适应。从高层次讲，软件体系结构不仅要具有一定的稳定性，而且要具有适应性调整的能力。理论上，把具有上述特性的软件体系结构称为适应性软件体系结构。

软件体系结构要适应不确定需求，就要求对需求有足够的了解，并且具备及时发现软件设计背离需求的机制。传统的软件工程中，软件需求与软件设计和执行相互独立，这种分割使得需求及其规格说明在软件中表现得并不明显[55]。正如前面的分析，要实现软件体系结构的设计随不确定需求而演化，就要将软件体系结构设计与软件需求分析相结合，这也是适应性软件体系结构与传统软件设计的最重要区别。适应性软件体系结构可以在软件设计阶段提供及时应对需求变化的响应机制。另外，相较于传统方法以功能性需求为主的观点，适应性软件体系结构的构建还重视系统的非功能性需求。

软件体系结构通过组件及其之间的连接来描述，因此在设计过程中，软件的整体需求必然要从系统层次(高层次)分解到组件层次(低层次)。适应性软件体系结构在这个分解过程不仅要保证高层次需求得到满足，而且要在高层次需求与低层次需求之间建立明确的映射关系，使得分解的低层次需求与高层次需求保持一致。另外，由于软件本身是一种逻辑结构，检验这种逻辑结构的有效性却要通过在计算机上实现的操作，这是一个与需求分解相反的过程。不仅每个单独的组件要实现各自的需求，而且组件的组合要满足软件的整体需求。于是，适应性软件体系结构除了要在高层次与低层次需求之间建立映射关系，还要能够解释这种映射关系。从适应性软件体系结构入手，增强软件对需求和结构变化的适应能力，使得软件一方面能够容纳一定范围的需求变化和结构变动，另一方面又在变化超出一定范围时能够方便地进行适应性修改。

事实上，对于软件体系结构而言，适应性的最重要体现并不是组件本身，而是组件之间的相互联系。由需求变化带来的诸多体系结构调整都要通过组件之间的交互关系的改变来实现。适应性软件体系结构所强调的适应性，不是组件所要实现的具体功能，而是所有组件相互联系形成体系结构时最终表现出来的一种外部属性。

图 4 给出了我们对适应性软件体系结构研究的一个参考模型。不确定需求环境是适应性问题研究的根源。在适应性软件体系结构开发过程中，需求和软件体系结构因不确定需求环境而交互，并在一定程度上对不确定需求环境造成新的影响。同时，根据需求分解的思想，将软件体系结构模型分为 3 个层次：系统层、连接层、组件层。适应性将体现在不确定需求与这 3 个层次间的联系之中。

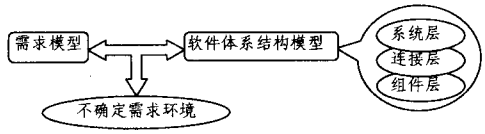

图 4 适应性软件体系结构研究模型

软件开发过程中对变化的管理一直被视为一项基础性和决定性的工作。既然需求本身不可能是封闭的，那么需求确定也不能是封闭的过程。我们将着眼点放在"适应性软件体系结构"的开发上，通过需求和体系结构的共同作用，在变化发生之时就及时调整设计方案，并反馈到开发过程。从问题空间出发，及早探寻解空间，通过解空间的反馈来加深对问题空间的理解。这样就可以在开发的早期同时引入需求和软件体系结构，并对风险进行度量。要实现适应性软件体系结构这个目标，还需要辨识出系统最核心的需求。核心需求通常都是用户非常关注的关键性需求，也是软件产品所必须具备的属性或功能。核心需求最能体现需求价值和产品价值。如果软件产品不能满足这类需求，用户满意度就会大幅降低，产品价值也将大打折扣。因此，根据核心需求设计出的软件体系结构才能真正保证软件产品的质量。我们希望能够通过对解空间的了解来增加对问题空间的把握，从而在整体上使得软件体系结构能够满足需求，以开发出高质量的软件产品。

结束语 本文从理论上讨论了在不确定需求条件下设计适应性软件体系结构，最终提高软件产品质量的实践性思路。在这个过程中，最根本的问题在于将需求分析与软件体系结构设计并重，关注问题空间与解空间的互动关系，考虑不确定需求对软件体系结构设计的影响。我们将"软件需求-软件设计-软件质量"作为研究主线，将不确定需求的演化分析贯穿其中。对于软件产品，如何在不确定需求条件下构建出相对稳定的、适应性的软件体系结构，最终保证高质量的设计结果，将是我们未来的研究方向。

参 考 文 献

[1] IEEE. ANSI/IEEE Std 729-1983. Standard Glossary of Software Engineering Terminology [S]. 1983

[2] ISO. International Standard, ISO/IEC 9126. Information Technology — Evaluation of Software Quality Characteristics and Guides for their use [S]. 1991

[3] Nuseibeh B, Easterbrook S. Requirements Engineering: a Roadmap [C]// Proceedings of the Conference on the Future of Software Engineering. 2000: 35-46

[4] Little T. Schedule Estimation and Uncertainty Surrounding the Cone of Uncertainty [J]. IEEE Software, 2006, 23(3): 48-54

[5] 孙昌爱，金茂忠，刘超. 软件体系结构研究综述[J]. 软件学报，2002, 13(7): 1228-1237

[6] Davis G B. Strategies for Information Requirements Determination [J]. IBM Systems Journal, 1982, 21(1): 3-30

[7] Moynihan T. Coping with "Requirements-uncertainty": the Theories-of-action of Experienced IS/software Project Managers [J]. Journal of Systems and Software, 2000, 53(2): 99-109

[8] Ebert C, Man J D. Requirements Uncertainty: Influencing Factors and Concrete Improvements [C]// Proceedings of the 27th International Conference on Software Engineering (ICSE 2005). May 2005: 553-560

[9] IEEE. ANSI/IEEE Std 610. 12-1990. Standard Glossary of Software Engineering Terminology [S]. 1990

[10] Free On-line Dictionary of Computing [DB/OL]. http://foldoc.org/, 2010-1-23

[11] Jones C. Strategies for Managing Requirements Creep [J]. IEEE Computer, 1996, 29(6): 92-94

[12] Change, the Double-Edged Sward [DB/OL]. http://media.johnwiley.com.au/product_data/excerpt/18/04702594/0470259418.pdf, 2010-1-5

[13] Boehm B. Requirements that Handle IKIWISI, COTS, and Rapid Change [J]. IEEE Computer, 2000, 33(7): 99-102

[14] Brooks F P. No Silver Bullet: Essence and Accidents of Software Engineering [J]. IEEE Computer, 1987, 20(4): 10-19

[15] Berry D M. The Software Engineering Silver Bullet Conundrum [J]. IEEE Software, 2008, 25(2): 18-19

[16] Boehm B. Making a Difference in the Software Century [J]. IEEE Computer, 2008, 41(3): 32-38

[17] Kozaczynski W. Requirements, Architectures and Risks [C]// Proceedings of the 10th Anniversary Joint IEEE International Requirements Engineering Conference (RE'02). 2002: 6

[18] Demarco T, Lister T. 与熊共舞——软件项目风险管理[M]. 熊节，马姗姗，译. 北京：清华大学出版社，2004

[19] IEEE. ANSI/IEEE Standard 610. 12-1990, Standard Glossary of Software Engineering Terminology [S]. 1990

[20] Costello R J, Liu Dar-Biau. Metrics for Requirements Engineering[J]. Journal of Systems and Software, 1995, 29(1): 39-63

[21] Fenton N E, Neil M. Software Metrics: Roadmap [C]// Proceedings of the Conference on the Future of Software Engineering. June 2000: 357-370

[22] Boehm B W. Software Engineering Economics [J]. IEEE Transactions on Software Engineering, 1984, 10(1): 4-21

[23] Ali M J. Metrics for Requirements Engineering[D]. UMEA University, 2006

[24] van Vliet H. Software Engineering: Principles and Practice (3rd edition)[M]. Wiley, June 2008

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

[26] Strens M R, Sugden R C. Change Analysis: a Step Towards Meeting the Challenge of Changing Requirements[C]// IEEE Symposium and Workshop on Engineering of Computer Based Systems (ECBS'96). Mar 1996: 278-283

[27] O'Neal J S, Carver D L. Analyzing the Impact of Changing Requirements[C]// Proceedings of IEEE International Conference on Software Maintenance. 2001: 190-195

[28] Schooff R M, Haimes Y Y. Dynamic Multistage Software Estimation[J]. IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, 1999, 29(2): 272-284

[29] Burgess C J, Dattanl I, Hughes G, et al. Using Influence Diagrams to Aid the Management of Software Change[J]. Requirements Engineering, 2001, 6(3): 173-182

[30] Cheng B H C, Atlee J M. Research Directions in Requirements Engineering[C]// 2007 Future of Software Engineering. 2007: 285-303

[31] 王青，李明树. 基于 SPC 的软件需求度量方法[J]. 计算机学报，2003, 26(10): 1312-1317

[32] 熊伟，新藤久和，渡边喜道. 软件需求定量分析及其映射的模糊层次分析法[J]. 软件学报，2005, 16(3): 427-433

[33] 黄蒙，舒风笛，李明树. 一种风险驱动的迭代开发需求优先级排序方法[J]. 软件学报，2006, 17(12): 2450-2460

[34] 王映辉. 软件功能需求变化传播机理分析[J]. 计算机学报，2007, 30(11): 2025-2032

[35] Shekaran C, Garlan D, Jackson M, et al. The Role of Software

Architecture in Requirements Engineering[C]// Proceedings of the First International Conference on Requirements Engineering. 1994:239-245

[36] Hall J G, MistrIk I, Nuseibeh B, et al. Relating Software Requirements and Architectures[J]. IEE Proceedings Software, 2005,152(4):141-142

[37] Bass L, Clements P, Kazman R. Software Architecture in Practice(2nd Edition)[M]. SEI Series in Software Engineering, Addison-Wesley Professional, April 2003

[38] Kruchten P, Obbink H, Stafford J. The Past, Present, and Future for Software Architecture[J]. IEEE Software, Special issue on Software Architecture, 2006,23(2):22-30

[39] Perry D E. Software Engineering and Software Architecture[C]// Proceedings of the International Conference on Software: Theory and Practice. Beijing: Electronic Industry Press, 2000:1-4

[40] Booch G. The Economics of Architecture-first [J]. IEEE Software, 2007,24(5):18-20

[41] Royce W W. Managing the Development of Large Software Systems: Concepts and Techniques[C]// Proceedings of the 9th International Conference on Software Engineering (ICSE'87). 1987:328-338

[42] Swartout W, Balzer R. On the Iinevitable Intertwining of Specification and Implementation [J]. Communications of ACM, 1982,25(7):438-440

[43] Garlan D, Perry D. Software Architecture: Practice, Potential, and Pitfalls[C]// Proceedings of the 16th International Conference on Software Engineering. May 1994:363-364

[44] Rajasree M S, Reddy P J K, Janakiram D. Pattern-oriented Software Development: Moving Seamlessly from Requirements to Architecture[C] // Proceedings of the Second International Workshop on from Software Requirements to Architectures (ICSE'03). May 2003

[45] Grünbacher P, Egyed A, Medvidovic N. Reconciling Software Requirements and Architectures with Intermediate Models[J]. Software and System Modeling, 2004,3(3):235-253

[46] Chung L, Gross D, Yu E. Architectural Design to Meet Stakeholder Requirements[C]// Proceedings of the First Working IFIP Conference on Software Architecture (WICSA). San Antonio, Texas, USA, 1999

[47] Nuseibeh B. Weaving Together Requirements and Architectures [J]. IEEE Computer, 2001,34(3):115-119

[48] Kozaczynski W. Requirements, Architectures and Risks [C]// Proceedings of IEEE Joint International Conference on Requirements Engineering. 2002:6-7

[49] Jackson M. Problem Frames: Analyzing and Structuring Software Development Problems [M]. Addison-Wesley Longman Publishing Co., Inc. 2001

[50] Diallo M H, Sim S E, Alspaugh T A. Case Study, Interrupted: the Paucity of Subject Systems that Span the Requirements-architecture Gap [C]// Proceedings of the 1st ACM International Workshop on Empirical Assessment of Software Engineering Languages and Technologies: held in conjunction with the 22nd IEEE/ACM International Conference on Automated Software Engineering(ASE). 2007:43-48

[51] de Boer R C, van Vliet H. On the Similarity Between Requirements and Architecture[J]. Journal of Systems and Software, 2009,82(3):544-550

[52] Garlan D. Software Architecture: a Roadmap[C]// Proceedings of the Conference on the Future of Software Engineering. June 2000:91-101

[53] 杨红. 适应性软件体系结构评价方法研究[D]. 大连:大连理工大学,2007

[54] 李刚,金茂忠. 适应性软件体系结构研究[J]. 计算机科学,2002,29(2):90-93

[55] Jun H, Colman A. The Four Major Challenges of Engineering Adaptive Software Architectures[C]// 31st Annual International Computer Software and Applications Conference(COMPSAC). 2007:565-572

(上接第 98 页)

结束语 本文重点研究如何将工作流建模和分析的形式化模型 WF-net 转换为 PNML,以便解决 WF-net 和其他 Petri 网模型的互操作问题。

本文首先在 PNML 元模型的基础上,针对 WF-net 中的一些特殊元素,如控制流构造模块和变迁触发器等,提出了扩展的元模型,然后基于扩展的元模型,定义了 WF-net 模型元素到 PNML 的转换规则。最后,在基于 Java 语言的 Eclipse 平台和通用交换格式 XML 的基础上,设计并实现了一个支持工作流网编辑和 PNML 格式存储的软件工具,取得了较好的效果,从而证明了提出的 WF-net 的 PNML 元模型的正确性和转换规则的可行性。

本文为 Petri 网标准交换格式的制定做了有力的探索。进一步的工作是将工具做得更完善,研究工作可以侧重通用的 Petri 网到 PNML 的转换方法。

参 考 文 献

[1] Matthias J, Ekkart K, Michael W. The Petri Net Markup Language [EB/OL]. http://www.informadtik.hu-berlin.de/top/pnml/ download/ JKW_PNML.ps,2000

[2] 周必水,胡伟军. Petri net 可扩展性标记语言[J]. 系统仿真学报,2003,15(8):49-52

[3] Jonathan B, Soren C. The Petri Net Markup Language: Concepts, Technology, and Tools[C]// Proceedings in Applications and Theory of Petri Nets 2003: 24th International Conference, 2003. Berlin: Springer-Verlag, 2003:1023-1024

[4] ISO/IEC/JTC1/SC7. ISO/IEC 15909-2 Software and Systems Engineering-High-level Petri Nets Part 2: Transfer Format International Standard[S]. 2009

[5] 赵文,袁崇义,张世琨,等. 一种模型驱动的工作流过程定义途径[J]. 计算机科学,2006,33(12):10-15

[6] Aalst W M P. The application of Petri nets to workflow management [J]. The Journal of Circuits, Systems and Computers, 1998,8(1):21-66

[7] Thomas F, Andreas E. Workflow Petri Net Designer [EB/OL]. http://www.woped.org