计算机科学 1995 Vol. 22№. 3

软件需求工程述评

张家重 吕 建 徐家福
(南京大学计算机软件研究所 南京 210093)

摘 要 本文对软件需求工程的有关研究内容进行了剖析。给出了需求工程的概念,讨论了其基本活动和信息类型;分析与评述了问题分析的基本原则、基本技术及面向对象需求方法;说明了软件需求规约的作用和内容,并对需求规约语言的几种形式、涉及内容进行了讨论。

关键词 软件需求工程,面向对象,需求规约语言。

1. 引 言

需求分析与定义处于整个软件开发过程的第一阶段,其成功与否对整个软件的开发和维护有着举足轻重的作用。在计算机发展的早期,由于所求解的问题规模较小,需求分析与定义也因此而被忽视。自1968年在北大西洋公约学术会议上为克服"软件危机"而提出软件工程以来,人们已逐渐认识到需求分析与定义的重要性,并随着软件系统复杂性的提高和规模的增大,需求分析与定义愈加困难和耗时,使得它在软件开发中所占的工作量更加突出,从八十年代中期以来,由于人们对需求分析与定义研究的重视,而形成了作为软件工程子领域的需求工程。

值得注意的是,近年来国际上兴起了需求工程的研究热潮,1993年召开了第一届需求工程国际研讨会,1994年召开了第一届需求工程国际会议,IFIP也成立了第一个关于需求工程的工作小组,即IFIP WG2.9。可见,需求工程的研究已得到国内外计算机界的广泛重视。

2. 需求工程

需求工程(Requirements Engineering,RE)是为实现需求分析、形成需求文档、支持需求演进而进行的基本原则、技术、语言和支撑工具的系统化应用。象其它工程一样,需求工程也遵从成熟且系统的途径,而非随机的或零乱的方式。RE使用的基本原则是一般规则或宗旨;技术是渐进过程或方案;语言用来表达需求的语法和语义,可为形式化的,也可为非形式化的;工具是辅助或自动实施原则、技术和语言和支撑软件系统。需求分析指需求的诱导获取;需求文档指需求规约或记录;需求演进指对用户要求变动的支持。

RE可用于包括软件和硬件的整个系统,也可只用于软件系统。前者称为系统需求工程,后者称为软件需求工程,本文以下仅指后者。

2.1 基本活动

RE的领域包括问题域和刻画软件系统外部行为的问题解域。相应地,RE涉及两类活动:问题(需求)分析和软件系统外部行为规约(定义),二者之间的活动类型是不同的。

在问题分析过程中,分析人员首先会见具有领域知识的人员,然后构造原型,并识别关于问题解的所有可能约束。其中主要难点是寻找一种权衡约束与过多信息的组织方法。问题分析只有在完全理解了欲解问题以后才能完成。

行为规约(软件需求规约)就是对所期望的软件产品的描述。事实上,行为规约即是对被理解了的问题表述其外部行为,消除矛盾,删除不一致的和模糊的成分。此间,需要对哪些部分可以自动化作出困难的决策。

必须指出的是,这两个活动并不是时序的和互斥的,因为:(1)一些软件开发需要很少或甚至不需

张家重 博士生,副教授。吕 建 博士,副教授。徐家福 教授,博士生导师,所长,计算机软件国家重点实验室主任。

问题分析。特别地，问题分析只是用于新的、困难的，或目前仍未解决的问题。(2)对于新的未解问题，也并不是完全分析好整个问题后才开始描述规约。事实上，只要部分问题被理解，对应的软件规约就可描述出来，即可以边分析问题边形成规约。

2.2 信息类型

需求活动中涉及的信息主要有三类：对象、功能和状态。下面从需求角度讨论其含义。

(1)对象：涉及需求领域的一切被明确定义边界的现实世界实体。"涉及需求领域"指排除实现时刻的对象(如，堆栈)。"明确定义边界"指对象须与待解问题有关，排除哪些被误认为是对象的情况。

对象往往被归为某对象类，一对象类是具有共同属性、函数及与其它对象的共同关系的对象的集合。对象是对象类的成员，它继承对象类的有关属性、函数及关系等。

(2)功能：即一项任务、服务、处理、数学函数，或是以下两种情况的活动：在现实世界中正在进行的；或要求系统将要实施的。功能常借助分解方法被分成若干层次，而得到一功能分解树结构。其根代表最抽象的功能，叶代表最具体的功能。需指出的是，这种层次是对功能的分解与简化，而非对软件的设计。

(3)状态：帮助系统、对象和功能决定在具体的问题环境中如何动作的条件。当为系统的状态时，它表示系统响应的集合，一般由需求规约来反映，而不显式地给出。

3. 问题分析

问题分析包括如下主要活动：(1)待解问题的学习；(2)理解用户的要求；(3)弄清真正的用户是谁；(4)获取与解有关的所有约束。

假定需求阶段的最终结果是形成软件产品外部行为的完整描述，即软件需求规约，那么问题分析可认为是定义软件产品空间，亦即，所有可能满足问题约束的结果范围。该过程应遵循一定的原则并利用相应的技术来进行。

3.1 基本原则

R. Yeh 等人在文[1]中首次提出了问题分析的三个结构化的基本原则，分别为：

(1)分解原则(Partition)：用以获得问题领域中对象之间、功能之间和状态之间的部分关系(part of)或聚合关系(aggregation of)等结构关系。

(2)抽象原则(Abstraction)：用以获得问题领域中对象之间、功能之间和状态之间的一般/具体关系(general/specific of)或实例关系(instance of)等结构关系。

(3)投影原则(Projection)：用以获得问题领域中对象之间，功能之间和状态之间的视图关系(view of)。

上述原则定义了对象之间、功能之间和状态之间的三种类型的结构关系或层次关系。不同的关系对于同一问题有着不同的解释。如，"A 从属于 B"可解释为：对于原则(1)，A 是 B 的部分；对于原则(2)，A 是 B 的实例，从而 A 继承 B 的所有属性；对于原则(3)，A 是 B 的视图。

3.2 基本技术

由于问题分析涉及到用户、系统分析员和软件开发者多类人员，从而带来了许多困难的工作。如：组织所有信息，处理不同人员的看法，发现并消除矛盾等。还有一个主要困难是如何避免"设计斜偏"，因为问题分析与软件设计活动极为相似，而且有许多技术方法和工具对二者均有效，只是其目的不同而已。

一种好的分析技术方法应至少具备如下特征：

(1)用一种易于理解、便于通讯的语言；

(2)提供定义系统边界的手段；

(3)具有定义上述几种结构关系的方法；

(4)引导分析员根据实际问题本身，而非目标软件系统进行分析和组织文档；

(5)易于发现不一致的成分；

(6)使分析员易于修改相关知识。

至今已形成的结构化技术(SA)和面向对象技术是问题分析方法的两个谱系。结构化技术是我们熟知的方法，而面向对象是较新提出且尚未成被广泛承认的方法，以下对其作一简要评述。

3.3 面向对象需求技术

面向对象的概念源于程序语言 Smalltalk，而作为需求技术，它主要从如下四个方面演变而来。

(1)源于面向对象设计(OOD)：OOD 主要是使系统具有模拟现实世界结构的系统结构。在设计时

刻，一个对象一般被认为是一个抽象数据类型，因而对象是数据和相关方法的封装。由于问题分析的目的是作为一个模型来描述现实世界，因此它使用对象就显得很自然。从OOD演化而来的面向对象需求技术(OORA)不区分二者中的对象，区别只在于其详细程度。

优点：1)问题对象比领域功能更易于定义且稳定；2)从OOD到OOA的过渡更直接。

缺点：难于区别分析时刻与设计时刻的对象选择准则，因为分析时刻的选择准则应考虑现实世界物理实体的某些含义，这对于理解问题很重要，但它却可能与设计构模无关。

(2)源于数据库设计：实体-关系模型(ERM/ERD)用于数据库设计，来识别实体、关系和属性。源于ERD的OORA较少注意，甚至忽略了作用到实体之上的功能或处理。

优点：1)同上1)；2)需求之后的数据库设计相对比较直接。

缺点：为理解问题而被构模的许多现实对象并不对应于数据库。

(3)源于一般需求技术：这种方法采用非形式化的技术，通过分析现实世界实体来理解问题，且强调问题领域中专有对象的创建。

优点：1)同上1)；2)只有与现实问题有关的对象被构模；3)对象的属性及其操作被封装。

缺点：从分析到设计的过渡不易。

(4)源于结构化方法：一些被改造的SA方法也称为OORA，它只是将数据流图(DFD)的形式改变一下，结果并无信息隐蔽和数据封装的含义，有的仅是简单的功能。这种方法对于面向对象来说谈不上有什么优点，因为它本质上并没有面向对象的思想。

面向对象的主要动机是当由于功能要求发生变化使系统需要演进时，因其对象保持不变，而易于维护。大多面向对象途径都强调现实世界中对象、对象类及其关系的定义与精化，有关细节可参阅文[2]。

若以需求为目的，对象应从以下几方面定义：1)现实世界实体；2)与问题领域有关；3)具有明确边界定义；4)属性和行为应易于理解，并对其封装。

4. 软件需求规约

需求工程的主要结果是软件需求规约(Software Requirements Specification，SRS)，是对将要构造的软件系统的外部行为的描述文档，即涉及目标软件与环境(硬件、软件和人)的所有接口的完整描述。SRS的主要作用有三：1)它可作为系统开发者和用户之间的合约，使得双方对待解问题有一个共同理解，实现用户，分析员和设计人员之间通讯；2)它可作为系统的开发依据，开发人员以此为基础进行软件的设计和实现；3)它可作为系统确认和验证的基础，一方面，可以确认需求规约是否满足用户的要求；另一方面，可验证软件的设计与实现是否正确。

4.1 基本内容

SRS的基本内容包括两部分：

* 行为需求规约：表达了软件系统"做什么"，描述系统的输入、输出及其关系的信息，完整地刻画系统的行为，因此它是整个软件需求的核心所在。亦称作行为或功能性规约。

* 非行为需求规约：定义系统的属性，包括系统的有效性、可靠性、安全性，易维护性和可见性等标准或限制。也称作非功能性规约。

良好的SRS应至少具备如下特征：

* 正确的；* 明确的；* 完整的；* 一致的；

* 可验证的；* 易理解的；* 易修改的；

* 可追踪的；* 设计无关的；* 注解好的。

欲使上述前五个特征满足，必须求助于形式化的方法，而不幸的是，形式化方法会失去一个重要特征，即对非计算机专业的用户或客户难于理解。一种折衷的途径是，采用形式化方法定义需求，但须开发必要的支撑软件工具用以实现将形式的SRS自动翻译成非形式的易于理解的SRS提供给用户。这样做的好处是它能满足以上两方面的要求，GIST项目[3]便是一例。

4.2 需求规约语言

需求规约语言是指用于书写用户对所需软件系统各种需求规约的语言。由于需求规约语言是需求工程的核心内容之一，它在软件开发与维护中有着十分重要的意义。

按照形式化程度，需求规约语言可分为非形式需求规约语言、半形式化需求规约语言以及形式化需求规约语言三类。

(1)非形式需求规约语言：指未作任何限制的自

然语言。其主要特征是非形式性，即歧义性、不完备性和模糊性。歧义性是指一句话可能会有多种不同的解释；不完备性是指问题描述中的某些侧面未完整地给出；模糊性是指某些语句含义不清，词难达意。一般说来，非形式需求规约语言具有易理解、易使用的特点，易于为一般用户所接受。但是由于自然语言的非形式性而使得需求规约中常出现错误，并且难以用计算机系统提供自动化的支持。

(2)半形式化需求规约语言：指在宏观上对语言的语法和语义有较精确的描述，而在某些局部方面则允许使用非形式的自然语言。通常有两种途径，其一采用图形化方法对语言的总体结构加以刻画。例如在SA中，用数据流图来刻画系统的总体结构，即系统由哪几部分组成、各部分之间的关系等，而系统中各个成分的含义则通常用自然语言规约；第二条途径是对系统需求规约所涉及的各个侧面加以总结、通过较精确的语法对之进行刻画。例如在PSL中，它对系统中所涉及的数据和加工的描述均有比较严格的规约；但在某些局部的侧面，则允许用户使用自然语言。这类语言的主要特点既便于用户表达和理解相应的需求规约，又可在某种程度上用机器对相应的需求规约进行管理和部分正确性检查。

(3)形式化需求规约语言：指其语法和语义均精确定义的语言。这类语言通常以一定的数学理论为基础。目前，常用的数学理论有两类，一类是逻辑，代表性的工作有Z语言和VDM-SL；另一类是代数，代表性的工作有OBJ语言和CLEAR语言。一般说来，形式化需求规约具有良好的数学性质，易于分析需求规约的各种性质，如一致性、完备性等，并且可用自动演绎机制对需求规约的分析提供自动化的支持，有利于在某种意义下保证后续开发步骤的正确性。然而，形式化需求规约语言往往要求用户具有较高的数学修养，且相对说来，所书写的需求规约较难理解。值得注意的是，目前，形式化需求规约语言主要集中于需求功能的刻画，而在非功能需求的形式化方面进展缓慢。

需求规约语言的研究主要涉及基本模型、语言结构和语用分析等[4]。基本模型是相应软件开发风范与方法在语言中的具体体现，它反映了软件需求的获取方式，决定了参加需求分析的各类人员的思维方式。两类重要的模型分别是功能分解模型和面向对象模型。

语言结构提供了用于刻画系统需求的具体手段，主要包括基本模型的描述、数据描述、控制描述、抽象机制、以及项目相关信息描述等，当然，不同语言的侧重点有所不同。

语用分析主要讨论需求规约语言的适用领域、易扩展性等性质，以及相应的方法与工具支持。例如，PSL主要适合于商业应用，RSL主要集中于实时系统。

5. 结束语

一般认为，1965-1975是程序设计突破的十年；1975-1985是软件设计突破的十年；并可望1985-1995是需求突破的十年。然而，需求工程技术至今尚无根本性的进展，原因是多方面的。其中最关键的是，需求工程与问题领域有关，且涉及人员繁杂。因此，为实现需求突破，开发出高质量的应用软件系统，除对需求工程技术本身进一步深化研究外，还应该广泛地研究其相关理论与技术，如，自然语言理解和领域分析(Domain Analysis)技术等。

参考文献

[1] R. Yeh, P. Zave, Specifying Software Requirements, Proceedings of the IEEE 68(9), 1980

[2] 张家重，王志坚，面向对象程序设计，《计算机科学技术百科全书》，清华大学出版社(待出)

[3] R. Balzer, et al., Operational Specifications as the Basis for Rapid Prototyping, ACM Software Engineering Notes, 7(5), 1982

[4] 吕　建，需求定义语言，《计算机科学技术百科全书》，清华大学出版社(待出)