

面向问题域的领域建模方法

刘楚达 楚旺 刘轶 钱德沛
(西安交通大学计算机系,西安 710049)

E-mail: bigliu168@163.com

摘要 阐述了形式化方法优点、描述语言及其应用现状,分析了面向问题域的开发方法的研究的必要性,如何用形式化的方法对问题域进行建模,以及面向问题域的领域建模方法的作用。

关键词 领域工程 需求工程 形式化 领域建模 软件工程

文章编号 1002-8331-200305-0017-04 文献标识码 A 中图分类号 TP391 TP393

Method from the Problem Region Oriented Domain Modeling

Liu Chuda Chu Wang Liu Yi Qian Depei

(Department of Computer, Xi'an Jiaotong University, Xi'an 710049)

Abstract: This paper discusses the excellence, description language and the application status of formalized measure. It analyses the necessity of exploitation research that faces to problem region and how to use formalized measure to build the problem region oriented domain modeling and describes the function of it.

Keywords: Domain Engineering, Requirement Engineering, Formalization, Domain Modeling, Software Engineering

1 引言

软件开发已经成为一个非常重要的工业领域,对软件工程的研究也进行了几十年的努力,并取得了一定的进展。而软件开发的工业化生产的模式仍未形成,目前面临的问题与20世纪60年代初出现“软件危机”时几乎相同,即软件质量难以保证,开发周期无法控制,开发效率难以提高。

几十年来,学术界对软件开发的研究主要集中在工程化管理方法、软件形式化及计算机支持下的辅助工具等领域,并取得了许多重要的进展。如提出了瀑布式周期模型、快速原形法、螺旋式模型等开发方法。设计了许多形式化语言(如Z、VOM)等,但在应用方面并未取得令人满意的效果。其重要原因在于:一是目前的研究是以问题的解决方案为中心的,忽略了对问题域的研究,所以用户与开发者之间缺乏共同的交流语言;二是软件形式化研究大多以数学为中心展开的,各种语言符号难以理解,面向问题的抽象层次太低,很难在工业界大规模应用;三是对软件自身的认识还不成熟,尽管面向对象的思想已得到了广泛的应用(主要用于软件设计和编程),但在整个开发周期内应用面向对象的思想还存在许多问题;四是缺乏有效的辅助工具对软件的重要特征(如一致性、正确性、完整性等)进行自动化处理。

为了摆脱上述困境,开展面向问题域的开发方法的研究是十分必要的。通过形式化的方法对问题域进行建模,建立完整、准确的领域模型,使软件开发成为一个可定义、可控制的过程。对问题域形式化方法的研究要以问题域描述和软件开发的共同本体为中心,消除用户与开发者之间、开发者与开发者之间的交流障碍。通过工程化方法的研究,保证领域建模的质量和效率。因为领域建模是勾通用户BPR(业务过程重构)以及软件开发过程的重要桥梁,对领域建模方法的研究应该成为软件工

程研究的一个重要领域。

2 领域工程、需求工程及软件工程的关系

2.1 领域工程

领域是指一组具有相似或相近需求和功能的应用系统所覆盖的区域^[1]。领域内的应用系统一般都具有许多相似的特性。现实世界问题领域的解决方法是充分内聚和充分稳定的,从而决定了同一领域内各系统的需求和功能具有显著的共性,其实现也必然具有共性。

领域模型是领域中各系统的共同需求的描述。它描述了领域内系统需求上的共性,称领域模型所描述的需求为领域需求,它是通过考察领域中已有的系统获得的。当领域中存在大量系统时,需要选择它们的一个子集作为样本系统。对样本系统需求的考察将显示领域需求的一个变化范围。一些需求对所有被考察的系统是共同的,而一些需求是单个系统所独有的。依据已获取的领域需求,就可以建立起领域模型。

领域工程可以分为三个主要的阶段:即领域分析、领域设计和领域实现。领域分析是分析领域中系统的需求,确定哪些需求是被领域中的系统广泛共享的,从而建立领域模型;领域设计根据领域模型获得领域设计模型;领域实现是依据领域设计模型得到领域实现模型。

2.2 需求工程及软件工程

2.2.1 需求工程定义

需求工程是指应用已证实有效的技术、方法进行需求分析,确定用户需求,帮助分析人员理解问题并定义目标系统的所有外部特征的一门学科。它通过合适的工具和记号系统来描述开发系统及其行为特征和相关约束,形成需求文档,并对不断变化的需求演进给予支持。

基金项目:国家“十五”863高技术发展研究计划项目:“新一代互连网技术综合实验环境”资助

作者简介:刘楚达(1967-),男,博士研究生,主要研究方向:网络拥塞控制、IP QoS、软件工程、需求工程。楚旺(1966-),男,博士研究生,主要研究方向:软件工程、需求工程。钱德沛,男,教授,博士生导师。

需求工程是一个不断反复的需求定义、文档记录、需求演进的过程。它由五阶段生命周期组成:需求定义分析、需求决策、形成需求规范、需求实现与验证、需求演进管理。需求工程的主要结果是形成软件需求规范,软件需求规范指出了用户解决问题或达到系统目标所需要的条件,是外部行为和系统环境接口的完整的描述性文档。好的软件需求规范应具有:正确无二义性、完整性、一致性、可验证性、可修改性、可跟踪性、易理解性以及有良好的注解等。

2.2.2 需求工程方法

目前有两种需求工程方法比较引人注目,一是面向目标(Goal-oriented)的需求工程方法,二是面向情景的(Senario-oriented)需求工程方法。

在面向目标的需求工程方法中,需求工程的目的是获取应用系统的高层次目标。通过对这些目标进行细化,形成服务规范和约束条件,将相应的职责分配给相应的代理者,如人、设备、软件等。在目标获取时,由于对行为的假设(包括对环境的假设)是理想的,一些无法预知的代理者的行为使系统的鲁棒性受到影响。同时,也可能获取的目标存在着一些矛盾或冲突,所以在形成软件规范前必须进行验证。Axel Van Lamsweerde等人在文献[2]中提出了相应的解决方法。

Scenario可以被看作是系统使用方面的规范,是一种十分直观的需求获取方法。但是该方法也存在许多问题,如:如何确定Scenario已经完整地描述了软件需求,各个Scenario之间是如何交互的。Scenario通常用自然语言进行描述,难以分析规范的完整性和一致性。

Colette Rolland在文献[3]中指出“尽管面向目标的方法是一种识别需求的有效途径,但是领域专家难以处理一个模糊的目标。如果目标没有给出,那它们将来自何处呢?此外,经验表明目标发现也不是一件容易的工作,而且不直观,所以应该将基于Scenario的方法与面向目标的方法结合起来。”

另一个与需求工程相关的重要研究领域是业务过程重构(BPR)。它的目标是通过过程重构,提高业务过程的效果或实现过程自动化。这一过程是由用户主导进行的,需要对目前的业务过程进行建模、分析、评价,并寻找利用IT技术的机会,尽可能地消除那些不能创造价值的环节。Gary Katzenstein等人在文献[12]中将业务过程的建模方法分为三类,一是使用传统的开发信息系统的方法,提供一些关于信息的冗余、不足、延迟等方面的问题;二是协作模型,将组织过程看作类似于Petri网,进行软件形式化的分析;三是社会技术系统模型,提供了一些有用的概念,如个人及组织目标、因果关系等。

为了便于对业务过程进行分析,一些学者提出利用软件形式化方法对业务过程进行描述,通过可执行的模型对过程进行仿真、验证和量化分析。Keith Thomas Phalp等人在文献[13]中提出了基于角色的(Role-Based)建模方法,而Michalis Glykast等人则将面向对象的思想引入业务过程建模,提出了代理关系模型(ARM)和代理/对象生存周期模型。

2.2.3 软件工程

而软件工程则是应用计算机科学、数学及管理科学等原理,以工程化原则、方法解决软件问题的工程。其中,计算机科学、数学用于构造模型与算法,工程科学用于制定规范、设计范型、降低成本及确定权衡,管理科学用于计划、资源、质量、成本等管理。

软件工程的形成和发展。大型软件系统的开发需要包括分析、设计、编程、测试、维护在内的一整套的软件工程理论与技

术体系。

2.3 三者的关系

软件开发中领域工程、需求工程和软件工程之间的关系如图1所示;

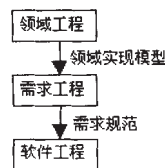


图1 领域工程与需求工程和软件工程间的关系

领域工程通过领域分析、领域设计和领域实现,分别得到领域分析、设计和实现模型。开发应用系统时,通过将领域工程和应用工程相结合的双工程方法,将领域分析、设计和实现模型人分别与应用工程中的分析、设计和实现模型相对应,它们在不同阶段、不同抽象级别,描述了反映领域本质特征的系统成分。

需求工程以领域模型为输入,确定应用系统的软、硬件结构,划分软件边界。由于领域模型已经对应用系统的环境进行了完整、准确地描述,所以最后形成的需求规范是比较稳定的。除了用户界面可能会产生修改外,其它需求是明确的,软件工程的输入就比较稳定。这样在进行项目规划时,进度将是可控的。

3 形式化方法

形式化是用数学的方法来描述系统的特性,而不是繁琐地刻划如何实现,即形式化的方法只描述系统要干什么而不涉及怎么干。典型的以形式化规约语言给出,提出一系列精确定义的概念,如:一致性和完整性,以及更进一步,定义规范实现和正确性。形式化是一个非常活跃的研究领域,目前对形式化的应用还存在着很大的分歧,但不可否认形式化具有以下优点:

- (1) 增加对复杂系统的功能和行为的理解;
- (2) 促进准确地描述需求规范;
- (3) 可以对系统、规范、实施特性(包括一致性、完整性、可靠性、正确性等)进行严格的分析;
- (4) 可以利用计算机进行辅助处理。

Jonathan P.Brown等人在文献[4]中提出了使用形式化的三个层次:

一是形式化规范,形式化语言可以使规范更加简洁和明确,在保持一定的抽象水平的前提下,消除模糊性。

二是形式化开发/验证,它包括形式化的描述系统,证明某些特征是可满足的,并且应用进化算子将抽象规范逐步变换为更有效、更具体的表示,最终的表示就是可执行的代码。

三是机器证明,用于一致性和完整性的证明,这对于安全系统来说最为重要。

目前用于形式化的语言主要有Z/Object-Z、VDM、CSP、RSML、CTL、Wright Darwin等等,其中有的语言用于描述状态特征(如Z/Object-Z、VDM),有的用于描述行为特征(如CSP、RSML),有的则用于描述结构特征(如Wright Darwin),还有的用于描述不变量特征(如CTL),这些形式化语言从不同的侧面描述软件的多种属性。

形式化除了使软件规范的描述严格、无歧义外,还可以用于自动验证。目前在这一方面也有所进展,例如,Marsha

chechick 等人在文献[5]中提出对利用 SCR 模型描述的软件需求规范与系统设计之间的一致性进行自动分析。而 Joanne M. Altee 等人在文献[6]中提出利用模型检查技术 (Model-checking) 对用 SCR 描述的软件需求规范的安全特征进行验证。比较有影响的模型检查工具包括 SPIN、PVS 等, 这些工具都是对有限状态系统进行检查。

测试是软件开发过程中成本最高的一个环节, 它贯穿于软件开发的全过程, 是产品质量保证的重要手段。形式化方法在该领域有很大的应用潜力, 软件测试目前主要集中在代码级。由于缺乏形式化的需求规范, 很难在开发的早期阶段开展基于规范的测试。现在利用形式化规范进行测试用例的生成的研究有一定的进展。Flaine Weyuker 等人在文献[7], Kai H chang 等人在文献[8], Nigel Tracey 等人在文献[9]中提出了利用软件规范生成测试用例的方法。

此外, 利用可执行的形式化规范进行仿真也是一个重要的研究方向, 通过仿真在系统开发前, 验证系统规范的正确性。通过可视化方法, 可以及早地获取用户的反馈, 并消除规范中的错误。Derrick Morris 等人在文献[10]中介绍了他们在仿真领域的进展。

4 面向问题域的领域建模方法

4.1 必要性

软件开发面临着两个重要的问题: 一是用户与开发者之间、开发者与开发者之间的交流存在困难; 二是形式化方法难以广泛应用。这两个问题也是软件工业发展过程中的最大障碍。对这两个问题的解决, 需要对问题域, 对软件本身进行新的认识。目前的软件开发方法仍将焦点集中在软件自身 (即解决方案本身) 的结构及特征方面, 而不是问题域方面。Jawed Siddigi 等人在文献[11]中指出, 软件是某种目标机器的描述, 它的开发就是这种机器的构建, 其需求描述了目的, 一个机器的目的是在机器之外发现的, 即在问题域的上下文中发现的, 所以需要转向一种面向问题的开发方法。

采用面向问题域的方法, 可以将应用系统的需求看作是问题域中现象之间的关系, 而需求规范就是一种限定性的需求。对问题域中的现象进行表达, 需要对问题域本身进行建模。而建模语言属于其开发者的术语, 用户与开发者之间由于没有使用共同的“语言”存在着交流障碍。尽管已经出现了以用户为中心的设计和 demand analysis, 但其效果并不明显。原因是这种“以用户为中心”并不是实质性的, 仍然是以开发者为导向的, 是为获取信息模型 (E-R 图、类图)、DFD/STD 等为主的“软件描述”, 这是开发人员理解的“语言”。而用户关注的是由人和应用系统共同构成的环境如何完成他们的业务过程, 并满足预期的目标, 而不是软件设计方案, 需要的是用问题域的术语所表达的系统行为及其特征。

4.2 方法

如何建立、描述领域模型, 寻找用户与开发者的共同的本体语言, 这是目前的需求工程和软件工程领域被忽视的问题。

问题域建模要涉及人们认识事物, 解决问题的方法。Jeffroy Parsons 等人在文献[12]中指出, 认识事物有两种方法, 一是本体论方法, 本体论主要研究存在 (Being) 及其关系; 二是分类理论方法。

本体论的观点认为:

(1) 现实世界 (Real-World) 是由具有不同特征的事物

(Thing) 所组成, 事物的特征受其内在规律 (Law) 的制约;

(2) 事物的特征可以通过属性 (Attribute) 来表征, 每一个属性都是一个状态变量, 它们在某一时刻的值就构成了事物的状态 (state);

(3) 每个事物都在变化, 这种变化通过状态的改变体现出来, 事物的规律决定了允许的状态变化;

(4) 事物之间存在交互, 这种交互影响其它事物的状态演化;

(5) 多个事物可以组合成一种新的事物, 新的事物有新的特征。

分类方法是人们通过对相似事物的分类构建知识结构的认知方法, 根据分类理论:

(6) 一个实例表明世界上一种事物的存在;

(7) 事物的特征构成了关于事物的知识;

(8) 一个概念 (或类) 是由一组实例的公共特征所定义的

(9) 一个实例可由其它实例所构成。

利用本体论的方法对问题域进行分析是比较有效的。用户的业务环境可以看作是一个社会系统, 组织理论和角色理论是进行社会系统行为及特征分析的重要方法。组织的存在是因为人们为了完成单个人无法完成的任务而形成的一种相互依赖的关系。角色理论告诉人们相互依赖意味着一个人的活动可能受其他人活动的影响, 这种相互依赖、相互影响的关系受组织内部多种规则的约束, 这种约束是保证组织有效运转的重要前提。组织规则表达了通用的、全局的需求, 组织结构定义了组织的类型及控制方式。所以组织中的本体是角色、结构及规则, 这是领域建模的要素。

近年来流行的面向对象技术就是分类理论的具体运用。其思想符合人们认识事物的基本方法, 所以被广泛认同。同时, 其中的数据封装、继承、多态性等概念简化了复杂系统的设计, 有利于代码的复用和维护, 成为编程和软件设计的主流方法。

面向对象的思想正逐渐应用于系统分析。由于它主要用于编程和软件设计, 软件设计的目标是模块化和代码复用, 产生良好的类树结构。而系统分析的目的是根据特定的需求识别问题域中的各种构件以及各构件之间的交互关系和约束条件。随着应用系统变得越来越复杂, 对相互依赖的构件集合的动态特征进行建模的需要也在增加, 而传统的面向对象技术在这一方面的建模能力有限, 不能满足需要。

应用系统的开发必须准确反映用户的环境, 即在应用系统中体现 (或实现) 相关的业务规则。传统的建模方法, 只是强调了静态特征, 通过 E-R 图或类图反映问题域中的数据之间的约束关系, 属于数据驱动的方法, 而对动态行为规则并未给予足够的重视。从目前的开发周期模型看, 无论是原形法还是螺旋模型都试图通过“试错”的方式不断获取问题域的规则。这种方式会造成大量的代码修改, 使系统的质量及可维护性, 需求的可跟踪性受到影响。正是由于在开发的前期没有对问题域的规则进行获取分析、验证, 才使后期的开发成为一种不可预期的过程。

面向对象技术在对系统动态进行建模时存在着一定的局限性, 对“软件”的认识方法也应该在面向对象的基础上进行扩展。类的概念仍然是软件开发的基础, 它表达了软件空间内各种概念之间的关系, 这是实现代码复用及可扩展性的保证。但同时要增加对象间相互协同的规则, 这种规则对应于问题域中的相应规则及其衍生规则, 这样可以把“软件”看作是由多种具有一定功能的对象按照一定的规则相互协作, 完成既定目标的

系统。

软件系统可以看作是一个人工的社会系统,对象对应于角色,所以软件领域中的本体也是角色、结构、及规则。因此角色、结构及规则可以看作是领域建模及软件开发的共同本体。这些概念简单、易懂,符合人们的思维习惯,同时在由领域模型向软件模型的演化过程中,保持表达方式及语义的一致性。

4.3 作用

对开发者来说,面向问题域的领域建模可以提供一般化的框架。它由相应的构件类型及结构所组成,通过实例化可以作为一个具体问题供解决方案。领域模型是一种抽象,它由那些与特定目标有关的领域知识所构成,领域知识包括:结构、规则、过程、数据关系等等。在领域模型中,概念关系是一个重要的组成部分,它主要用于对业务术语的解释。在一个具体的业务环境中,对同一概念可能有不同的术语,对同一术语可能有不同的解释。这种现象的存在,在建模过程中容易产生冗余或冲突,并影响人员之间的交流,而且还会影响规范或模型的重用。因此领域模型中的概念关系,就是该领域的共同术语或称本体。

利用角色理论对问题域和软件进行建模,可以使领域模型及软件模型基于共同的本体的概念,容易理解,而且问题域的描述及解决方案的描述是同构的不必进行语言的转换,所以更加直接地导向解决问题的方向。

面向问题域的领域建模还可以大大改善需求的可跟踪性。这在传统的开发方法中一直难以解决,因为最初的需求只是一个简单的、自然语言描述的陈述。在需求工程中获取的比较详细的需求,通常也是以自然、语言表达的,要将软件功能与需求直接相关联很不自然。有的则可能无法建立这种关联,且联的维护也十分困难。

需求可跟踪性是为了保证系统的开发与用户需求相一致,帮助开发者理解如何及为什么产品能满足用户的需要。从管理的角度看,可跟踪性有助于说明每一个需要是否已经得到满足;从系统维护的角度看,维护人员及用户可以准确地评估系统的改变所产生的影响及维护请求是否可行。

正是基于以上的要求,需求的可跟踪性一直是软件工程质量评价的重要因素。通过建立领域模型可以使这一问题得到较好的解决,因为软件的开发过程就是从领域模型开始,逐步演化形成软件设计方案,演化的轨迹通过相应的模型可以清晰、准确地记录演化的因果联系,关联的维护可以利用计算机自动进行。

5 结语与将来的工作

由于“软件危机”的出现,人们在软件开发中引入了工程化管理方法,并取得了一定的进展。但面临的问题并未得到根本的改观,所以在20世纪90年代中后期,又将软件需求的获取作为新的研究方向,提出了“需求工程”的概念,并取得了许多重要成果,在模型形式化、建模工具及建模方法等方面需要展开更加广泛和深入的研究。该文阐述了形式化方法优点、描述语言及其应用现状,分析了面向问题域的开发方法的研究的必要性,如何用形式化的方法对问题域进行建模,以及面向问题域的领域建模方法的作用。

领域建模方法的研究将主要在以下几个方面展开:

5.1 建立模型形式化的理论基础

从多个视角研究描述模型的形式化方法,为计算机辅助处

理建立良好的理论基础。

5.2 可执行模型研究

通过构建可执行的模型,可以对业务过程进行仿真和分析,采用可视化的手段,使业务过程更加直观和清楚,便于相关人员的交流和沟通。

5.3 模型完整性与一致性的验证

明确定义模型的完整性和一致性的含义,建立各种特征之间的关系,并设计相应的验证算法,

5.4 基于规则的特征证明

研究有效的推理方法和表达方法,验证模型能否满足指定特征的要求。

5.5 模型正确性验证

研究有效的验证算法,确认设计与需求是否一致,需求与领域模型是否一致。

5.6 测试用例的生成

以领域模型为基础,生成系统确认测试计划,并评价测试的有效性。

5.7 知识工程

研究的重点在于领域知识的表达,存贮、检索及管理,实现基于模型的复用。(收稿日期:2002年9月)

参考文献

1. Frederick H. Architecture based Acquisition and Development of System: Guidelines and Recommendations from ARPA Domain-Specific Software Architecture Program[R]. Ver 1.01, CMU/SEI, 1994
2. Axel Van Lamsweerd, Emmanuel Letier. Handling Obstacles in Goal-Oriented Requirements Engineering[J]. IEEE Transactions on Software Engineering, 2000, 26 (10): 978~1004
3. Colette Rolland, Carine Souveyet, Camille Ben Achour. Guiding Goal Modeling Using Scenarios[J]. IEEE Transactions on Software Engineering, 1998, 24 (12): 1055~1070
4. Jonathan P. Browne, Michael G. Hinchey. Ten commandments of Formal Methods[J]. IEEE computer, 1995, 56~62
5. Marsha chechik, John Gannon. Automatic Analysis of consistency between Requirements and designs[J]. IEEE Transactions on Software Engineering, 2001, 27 (7): 651~671
6. Joanne M. Altee, John Gannon. State-Based Model checking of Event-Driven System Requirements[J]. IEEE Transactions on Software Engineering, 1993, 19 (1): 24~40
7. Elaine weyuker, Tarak Goradia, Ashutosh Singh. Automatically Generating Test Data From a Boolean Specification[J]. IEEE Transactions on Software Engineering, 1994, 20 (5): 353~363
8. K. H. Chang, S. S. Liao, S. B. Seidman et al. Testing Object-Oriented programs from formal Specification to test scenario generation[J]. The Journal of systems and software, 1998, 42: 141~151
9. Nigel Tracy, John Clark, Keith Mander. Automated Program Flaw Finding Using Simulated Annealing[J]. ASM SIGSOFT Software Engineering Notes, 1998, 23 (2): 73~81
10. Derrick Morris. Simulating the Behavior of Computer Systems: Co-Simulation of Hardware/Software[J]. The Computer Journal, 1997, 40 (10): 617~629
11. Jawed Siddiqui, Chandra Shekaran. Requirements Engineering: The Emerging wisdom[J]. IEEE Software, 1996, 15~19
12. Jeffrey Parsons. Using Objects For System Analysis[J]. Communications of the ACM, 1997, 40 (2): 104~110