基于用例和Z的一种软件体系结构建模的方法

刘妮娜,周 枫
(昆明理工大学 信息工程与自动化学院,云南 昆明 650051)

摘 要:随着计算机软硬件技术和网络技术的发展和应用的不断深入,软件复用技术,特别是软件体系结构的设计和复用是克服软件危机行之有效的手段。用例在面向对象的软件开发过程中起着重要的作用,但它是一种半形式化的方法,缺乏精确的语义描述,难以保证软件体系结构模型的正确性,形式化的方法恰好能够弥补这方面的不足。针对如何构建软件体系结构的问题,提出了一种基于用例和Z语言的形式化构建软件体系结构的方法,将迭代增量技术运用到体系结构的设计开发中,并就相关问题进行了分析探讨。

关键词:软件体系结构;用例;Z语言;迭代增量;形式化
中图分类号:TP311.5 **文献标识码**:A **文章编号**:1673-629X(2012)02-0037-04

A Modeling Method of Software Architecture Based on Use Case and Z Specification

LIU Ni-na,ZHOU Feng
(Faculty of Information Engineering and Automation,Kunming University of Science and Technology,
Kunming 650051,China)

Abstract:With the development and deepgoing application of computer hardware and software technology,the technology of software reuse,especially the design and reuse of the software architecture,was an effective method to overcome the crisis of software. The use case played an important role in the object-oriented software development process,but it was a half-formal representation,and it lacked of precise semantics description. The use case can't assure the accuracy of the software architecture model,but the formal method can make up for this deficiency. On how to construct the software architecture,it gives a kind of method based on the use case and the formal Z specification to construct the software architecture. This method adopts the technology of iterative increment during the design and development of the software architecture,and discusses the related issues.

Key words:software architecture;use case;Z specification;iterative increment;formalization

0 引 言

随着国内外软件产业的蓬勃发展,人们对软件危机日益重视。软件复用技术已成为克服软件危机的有效手段,它可以极大地提高软件生产率,降低软件成本,保证软件产品的质量。近年来,人们对软件复用的研究,主要集中于源代码级的功能部件的复用上。然而,随着用户需求的提高和软件技术的迅猛发展,对规模大、复杂程度高的软件系统而言,开发者不仅要考虑到功能部件的设计和复用,而且更要考虑到软件体系结构的设计和复用问题。软件体系结构是有效实现大粒度软件复用的重要方法,是系统分析和设计的高层复用。它可以从全局、整体的角度去把握系统各部分的内在联系,并在更高的抽象层次上去分析和理解整个系统的行为和特性,从而解决开发大型、复杂软件系统存在的难题。但具体如何实现软件体系结构一直没有得到有效的解决。文中就如何构建软件体系结构的问题,提出了一种基于用例和Z语言的软件体系结构建模的方法,将迭代增量技术运用到体系结构开发设计中。同时,分析了增量迭代的过程和相关问题。

收稿日期:2011-06-16;**修回日期**:2011-09-21
基金项目:云南省教育科学研究基金(07C10799)
作者简介:刘妮娜(1985-),女,湖北随州人,硕士研究生,主要研究方向为软件工程及其应用;周 枫,副教授,硕士,主要研究方向为软件工程及其应用。

1 软件体系结构和Z语言的相关概念

软件体系结构(Software Architecture,简称SA)首次是在1969年的NATO软件工程会议上提出。IEEE 610.12—1990软件工程标准词汇中定义:体系结构是

以构件、构件之间的关系、构件与环境之间的关系为内容的某一系统的基本组织结构，以及指导上述内容设计与演化的原理[1]。

从体系结构构造的角度来看，软件体系结构是由构件、连接件以及它们之间的配置约束构成的。它为软件系统提供了一个结构、行为和属性的高级抽象，它不仅指定了系统的基本组织结构，而且显示了构成系统的元素和系统需求之间的对应关系，以及指导上述内容设计与决策的基本原理。

Z语言是一种应用广泛、非常有效的形式描述规范语言，具有“状态——操作”风格。它以一阶谓词逻辑和集合论作为形式语义基础，将函数、映射、关系等数学方法用于规格说明[2]，具有精确、简洁、无二义性等优点，可有效支持体系结构的求精、验证。Z语言借助于模式（Schema）来表达系统结构。模式有水平和垂直两种形式。

水平形式：

Schema_Name ≙ Schema_EXP

垂直形式：

Schema_Name

Declarations

Predicate

一个模式由变量说明和谓词约束两部分组成，可用来描述系统的状态和操作，即：模式＝声明＋谓词。声明部分引入变量，谓词部分表示了关于变量值的要求。

2 基于用例和Z的映射方法

一般意义上，软件需求获取（Software Requirement Elicitation）是软件系统开发过程中最为困难也是最为重要的部分，只有真正满足用户需求的软件产品才能为用户接受。需求模型[3]的建立主要是识别系统的用例和角色，系统用户按角色（Role）进行划分，明确各类角色的目标（Goal），接着分析系统中各角色和用例间的联系，也就是从外部用户的角度观察，系统应该支持哪些功能，它是对系统功能的宏观描述。对复杂系统而言，系统建模往往需要进行严格的语义分析，但用例却缺乏精确的语义描述，它是一种半形式化的描述方法，对系统模型难以进行正确性分析和一致性检查，形式化的方法恰好能够弥补这方面的缺陷。用例和形式化描述语言Z在软件体系结构需求建模中结合使用，寻求一种在软件体系结构建模过程中用例到形式描述语言Z的映射[4]与转换表达方法。

用例图主要描述执行者（角色）和用例之间的关系，通常包括角色、用例、用例之间的关系（关联、泛化、包含和扩展）和系统，所以对用例图的形式化工作就转化为对用例图元素的形式化[5]描述。

1）角色。

Actor

rolename:Name

attribute: $\mathbb{F}$ *UMLAttribute*

operation: $\mathbb{F}$ *UMLOperation*

$\forall$ *a1,a2:attribute•a1.name=a2.name⇒a1=a2*

$\forall$ *op1,op2:operation•(op1.name=op2.name∧#op1.parameters=#op2.parameters∧*

∀i:1..#op1.parameters•op1.parameters(i).name=op2.parameters(i).name∧

op1.parameters(i).type=op2.parameters(i).type)⇒op1=op2

角色是用例的发起者，是为了完成一个事件而与系统交互的实体，它可以由人承担，也可以是其它系统、硬件设备，甚至时钟。

2）用例。

ExtensionPoint

location: $\mathbb{N}$

size: $\mathbb{N}$

usecase:UseCase

UseCase

name: Name

attribute: $\mathbb{F}$ *UMLAttribute*

operations: seq *Operation*

extensionpoint: $\mathbb{P}$ *ExtensionPoint*

baseextend: Extend

extensionextend: Extend

baseinclude: Include

additioninclude: Include

$\forall$ *e: extensionpoint•e.usecase=self*

用例是一个外部可见的系统功能单元，它展示了参与者（角色）如何使用系统。为了体现用例间包含和扩展的依赖关系，在用例的形式化模型中引入扩展点的概念。

3）系统。

System

usecase: $\mathbb{F}$ *UseCase*

generalization: $\mathbb{F}$ *Generalization*

include: $\mathbb{F}$ *Include*

extend: $\mathbb{F}$ *Extend*

{i:include•{i.parent , i.sub}}⊂UseCase

{e:extend•{e.parent , e.sub}}⊂UseCase

系统主要体现的是需求规范中需要提供的功能及这些功能之间的组织关系，它是由用例及用例之间的关系组成的集合。

4）关系。

AttachedKind::=Actor | UseCase

AssociationEnd

rolename: Name
multiplicity: P N
attached: AttachedKind

multiplicity≠{0}

Association

name: Name
e1 , e2: AssociationEnd

e1.rolename≠e2.rolename
e1.rolename∈{ca:e1.attached.attribute•ca.name}
e2.rolename∈{ca:e2.attached.attribute•ca.name}
∀ a1 ,a2: Association | a1≠a2•{a1.e1.attached , a1.e2.attached}= {a2.e1.attached , a2.e2.attached}⇒a1.name≠a2.name

Generalization

parent , child: UseCase

(∀g:Generalization•{g.parent , g.child}) ∩ id(UseCase)=∅*

Include

name: Name
base: UseCase
addition: UseCase

Extend

name: Name
base: UseCase
extention: UseCase

关联(Association)表示了用例与角色、角色与角色之间存在某种语义上的联系,它提供的是一种通讯的路径;泛化(Generalization)可以表示角色与角色之间、用例与用例之间的一般或特殊化关系;包含(Include)描述的是被包含的用例(基用例)与包含用例之间的依赖关系;扩展(Extend)描述的是扩展用例与基用例之间的依赖关系。

5)用例图。

用例图是由角色、用例、系统及四种关系组成[6],它是对系统的行为进行建模,所以只要将角色、用例、系统及四种关系的形式化描述组织起来,就可以得到该用例图的形式化描述。

RELATIONSHIP::=Association | Generalization | Include | Extend

UMLUsecaseDiagram

Actor
UseCase
System
Relationship: RELATIONSHIP

3　软件体系结构建模方法

用例是能够向用户提供有价值结果的系统中的一种功能,也是对象建模活动的起点[7]。对用例进行形式化描述,可以克服用例对系统模型描述的不一致性问题,能更好地获取系统的功能需求。

用例与软件体系结构之间存在着紧密的关系:希望系统支持的用例会影响软件体系结构的设计,用例是软件体系结构的驱动力(软件体系结构还受到系统软件、中间件、遗留系统、标准和政策、非功能性要求、分布需求等影响);此外,用例也受到软件体系结构的影响,软件体系结构是在细化阶段[8]的迭代过程中被创建的。

依据软件体系结构和用例之间的相互关系,可以将迭代增量的技术运用到软件体系结构的构建中,具体方法为:

(1)首先,不考虑具体的用例,在充分了解、明确领域范围的基础上建立一个临时的软件体系结构;

(2)其次,选取几个重要的用例,并结合对用例的 Z 语言形式化描述,使软件体系结构能够支持被选取的用例;

(3)进一步选取更多能够支持体系结构的用例,构建更加完美的软件体系结构,重复以上过程。

每一次迭代过程中,选取并实现一组能够支持体系结构的用例,结合对用例的 Z 语言描述来确认体系结构。如有必要,再进一步对体系结构进行改进、完善。随着每次迭代的进行,可在所选取用例的基础上,进一步实现软件体系结构的某些专门应用部分。体系结构模型的构建方法如图 1 所示。

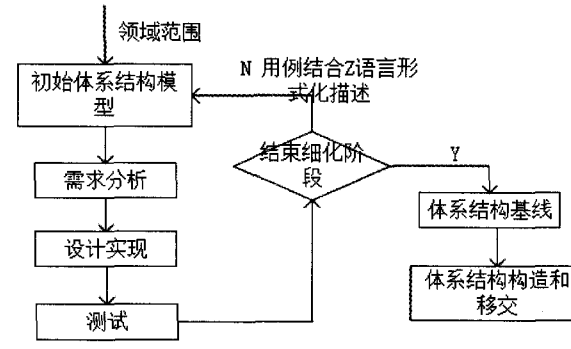

图 1　基于用例和 Z 的软件体系结构迭代增量构建模型

软件开发生命周期中,对一个软件项目,可以粗略划分为四个阶段:先启、精化、构建和产品化。各阶段的主要目标如下:

先启(Inception)——确定项目开发的目标和范围,并建立初始业务案例,从业务的角度表明项目的可行性;

精化(Elaboration)——确定系统架构,建立体系结构基线,明确大多数的需求;

构建(Construction)——实现系统的功能,确保产品达到最初的可操作能力,并可以开始移交给客户;

产品化(Transition)——完成软件的产品化工作,

确认得到一个准备向用户发布的产品，将系统移交给客户，并培训用户如何使用该软件。

软件体系结构是在精化阶段的迭代过程中被创建的，它最终得到的是一条软件体系结构基线[9]（即一个只有些许软件"肌肉"的系统骨架）。整个迭代过程包含5种核心工作流：需求、设计、实施（编码）、部署和测试，且不同的阶段，迭代侧重点是不同的。在先启和精化阶段中，绝大部分的工作主要集中在获取需求，进行初步的分析和设计上；在构建阶段中，重点则转移到软件详细设计、实施、部署和测试上。每次迭代完成之后需要对迭代完成的结果进行评估，其中一个目标就是判定用户是否提出了新的需求，或者已有的需求变化是否会影响到后续的迭代，以此为依据来制定下一次迭代的目标。一次迭代完成会产生一个增量结果，一个增量便是一次迭代的内部版本与下一次迭代的内部版本之间区别的标志。

4 实 例

文中将以传统的编译器来介绍基于用例和Z的软件体系结构映射方法。编译器是一种能将高级语言程序翻译成低级语言（机器语言、汇编语言）目标程序的语言处理程序。

传统的编译器，一般由词法分析、语法分析、语义分析与中间代码生成、中间代码优化和目标代码生成等阶段组成。传统编译器的用例图如图2所示。

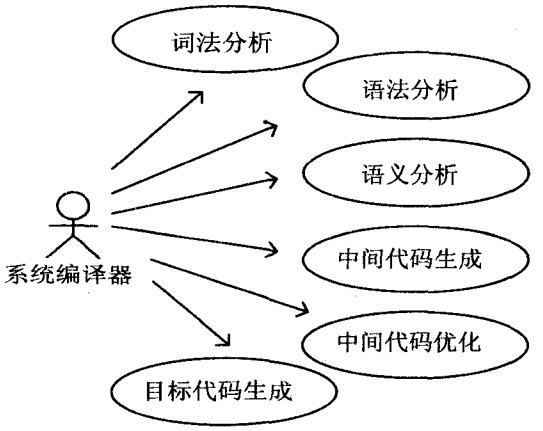

图2 编译器的用例图

传统编译器的Z语言形式化描述如下：

图形化规格说明语言容易书写和理解，但缺乏精确性和形式的语义；形式规格说明技术精确，无歧义，但难以理解和构造。在此，结合两者的优点，描述了传统编译器的功能需求，并以管道—过滤器风格[10,11]来构造编译器的体系结构。基于篇幅有限，在此暂不谈及编译器体系结构迭代增量构建的具体过程，只是给出体系结构的描述。

```
Compiler_UsecaseDiagram
role_name: RName
role_attribute: F UMLAttribute
role_operation: F UMLOperation
usecase_name: P UName
usecase_attribute: F UMLAttribute
usecase_operation: seq Operation
Relationship:RName +→UName

Relationship::=Association | Generalization | Include | Extend
UName::=Accidence | Phraseology | Semantic | Mid-code-switch
| Mid-code-optimize | End-code-switch
```

编译器的体系结构描述如图3所示。

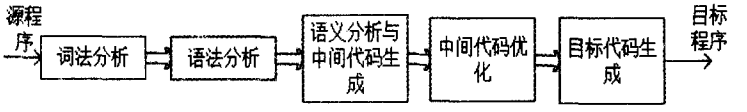

图3 编译器的体系结构

5 结束语

文中结合一个实例研究了如何通过用例和Z的映射方法，运用迭代增量的方式得出系统的软件体系结构。与传统的映射方法相比，基于用例和Z的映射方法更好地保证了软件系统模型的正确性。迭代增量的软件体系结构构建方法比较符合人们的认识和思维方式，它对软件体系结构的构建具有很好的指导作用。文中对于如何选择与体系结构相关的用例没有具体讨论，有待于进一步研究的问题还包括：如何保证每次迭代的增量和最后收敛，以及迭代用例的重要性排序和相关性等。进一步的工作还包括：完善从用例到形式描述语言Z映射的形式规约，且通过定理证明或模型检测[12]等方式给出其形式验证，以及对软件体系结构的形式化验证，确保其满足系统的需求。

参考文献：

[1] 余雪丽. 软件体系结构及实例分析[M]. 北京：科学出版社，2004.

[2] 张晓莺，朱关铭，缪淮扣. 规格说明语言Z的类型检查[J]. 计算机应用与软件，2000，17(2)：1-9.

[3] Scott R. UML用例驱动对象建模—种实践方法[M]. 徐海，周 靖，陈华伟，译. 北京：清华大学出版社，2003.

[4] 汤小康. 基于UML和Z的需求分析到软件体系结构的映射研究[D]. 长沙：湖南师范大学，2007.

[5] Sengupta S，Bhattacharya S. Formalization of UML Use Case Diagram-A Z Notation Based Approach[C]//International Conference on Computing and Informatics：ICOCI 2006. Washington，DC：IEEE Computer Society，2006：1-6.

[6] 袁红超，郝克刚，葛 玮. 用例技术的研究和应用[J]. 计算机技术与发展，2006，16(2)：96-97.

（下转第44页）

RSSI算法要大。图1表示对同一个节点仿真10次过程中用QPSO-RSSI算法和对RSSI求均值方法得到的LQI值和实际距离对应的LQI值的比较，可以看出运用QPSO-RSSI算法求得的LQI值都在实际距离LQI值附近，要比对RSSI求均值得到的LQI值更好。

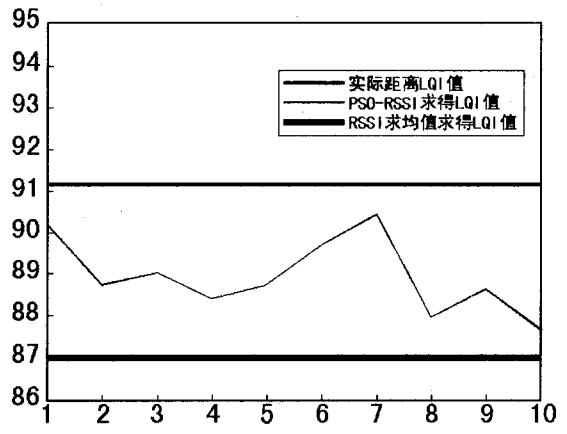

图1 两种算法得到LQI值比较图

通过极大似然法[11,12]即可实现节点的定位，运用MATLAB做仿真定位图，其中运用求RSSI均值法后再进行定位，得到理论坐标点偏离实际坐标点距离较大，而从图2中可以看出运用QPSO-RSSI算法后再进行定位理论坐标点和实际坐标点的相差距离很小，精度比较高。

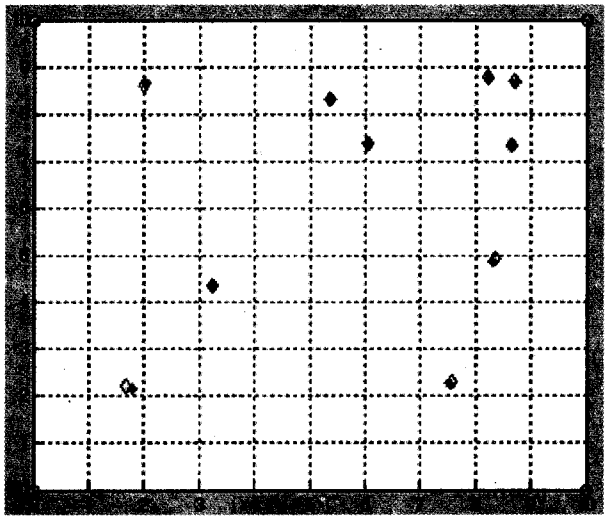

图2 QPSO-RSSI算法定位仿真图

3 结束语

本文提出了QPSO-RSSI自校正定位算法，仿真结果表明，QPSO-RSSI自校正定位算法能够更精确地对节点进行定位。此外，该自校正算法对硬件的要求不高，适合于无线传感器网络的低成本、低功耗的要求，是一种实用的优化定位。同时，量子离子群算法也解决粒子群算法会出现局部最优的问题，从而更好地提高定位性能。

参考文献：

[1] 孙利民，李建中，陈 渝，等. 无线传感器网络[M]. 北京：清华大学出版社，2005：5-15.

[2] 户晓玲，曾建潮. 基于微粒群模型的移动传感器网络部署研究[J]. 计算机技术与发展，2009，19(10)：81-88.

[3] 王书聪. 无线传感器网络分布式定位算法研究[J]. 计算机技术与发展，2009，19(5)：192-194.

[4] 唐 琳，高宝成. 无线传感网络节点自定位技术的研究[J]. 计算机与现代化，2010(2)：19-21.

[5] 唐 强，郭 庆，彭 保. 一种基于优化TDOA的无线传感网络定位算法[J]. 计算机工程与应用，2008，44(25)：98-99.

[6] 李士勇，李盼池. 量子计算与量子优化算法[M]. 哈尔滨：哈尔滨工业大学出版社，2009.

[7] 陈星舟，廖明宏，林建华. 基于粒子群优化的无线传感器网络节点定位改进[J]. 计算机应用，2010(7)：1736-1738.

[8] 刘子京，斐文江. 基于Zigbee协议的无线传感器网络研究[J]. 计算机技术与发展，2009，19(5)：192-194.

[9] He T，Huang C，Lum B，et al. Range-free localization schemes for large scale sensor networks[C]//Proc of ACM MobiCom. San Diego，CA：[s. n.]，2003.

[10] Deng Jing，Varshney P K，Haas Z J. A new backoff algorithm for the IEEE 802.11 distributed coordination function[C]// Proc CNDS'04. [s. l.]：[s. n.]，2004.

[11] Ledeczi A，Kiss G，Feher B，et al. Acoustic source localization fusing sparse direction of arrival estimates[J]. Intelligent Solutions in Embedded Systems，2006(6)：1-13.

[12] Mendalka M，Kulas L，Nyka K. Localization in wireless sensor networks based on Zigbee platform[C]//Proc of 17th International Conference on Microwaves，Radar and Wireless Communications (MIKON). [s. l.]：[s. n.]，2008：1-4.

(上接第40页)

[7] Bendraou R，Blanc X. A comparison of six UML-based languages for software process modeling[J]. IEEE Transactions on Software Engineering，2010，36(5)：662-675.

[8] Runeson P，Host M. Guidelines for conducting and reporting case study research in software engineering[J]. Empirical Software Engineering，2009，14(2)：131-164.

[9] Becker S，Koziolek H，Reussner R. The Palladio component model for model-driven performance prediction[J]. Journal of Systems and Software，2009，82(1)：3-22.

[10] 郭广义，李代平，梅小虎. Z语言与软件体系结构风格的形式化[J]. 计算机技术与发展，2009，19(5)：140-142.

[11] 李 辉，张广泉，陈枢茜. 基于UML和XYZ/E的ATM系统体系结构描述[J]. 微计算机信息，2010(9)：200-202.

[12] 罗巨波，应 时，叶 鹏. 一种支持软件体系结构重用的反射机制及其形式化[J]. 计算机科学，2009，36(8)：145-148.