面向方面的软件非功能特性研究

杨汉明,张立臣
(广东工业大学 计算机学院,广东 广州 510090)

摘 要:软件非功能特性的处理一直以来都是困扰着研究人员和开发人员的难题。将面向方面的编程思想运用于软件非功能性的研究打破了传统的软件设计过程,这种研究方法在软件需求分析的初期分离出系统的非功能特性并将这些非功能特性做一个整体的处理,加快了软件开发的进度,提高了生产效率。文中基于面向方面思想,利用著名的非功能特性描述框架 NFR 对软件非功能特性进行描述,解决了软件关注点组合时产生的冲突问题,为软件非功能特性的处理提供了很好的解决办法。

关键词:面向方面编程;横切关注点;非功能需求框架;关注点分离
中图分类号:TP311.5 **文献标识码**:A **文章编号**:1673-629X(2011)02-0112-04

Research of Software Non-functional Requirements Based on Aspect-oriented

YANG Han-ming, ZHANG Li-chen
(Faculty of Computer Science, Guangdong University of Technology, Guangzhou 510090, China)

Abstract: How to address non-functional properties in software system has afflicted various stakholders for a long time. Non-functional Requriements based on Aspect-oriented breaked the traditional sofeware design process. This approach isolated non-functional characteristics of the system early in the software requirements analysis and done a whole handling of these non-functional properties. This approach speeded up the progress of software development and improved production efficiency. This paper based on Aspect-oriented, described non-functional properties by the Non-functional Requirements Framework and solved the conflicts of concerns. This is a better way to address non-fronctional properties.

Key words: aspect-oriented programming; crosscutting concern; non-functional requirements framework; separation of concerns

0 引 言

面向方面编程(Aspect-oriented Programming,简称 AOP)是在面向对象编程(Object-oriented Programming,简称 OOP)的基础上发展起来的一种编程思想,它的核心思想是将系统的非功能性横切关注点抽象成方面,模块化非功能性需求。AOP 解决了横切关注点的有效分离,通过将横切关注点局部化、模块化,达到对分散行为的控制和追踪[1]。

AOP 的出现并不是要取代 OOP,而是互补,它既保留又发展了 OOP 的特性,优化了对实时约束、安全策略、异常处理、日志记录等横切整个系统的关注点的处理。在传统的 OOP 中,这些关注点都分散在各个类中,造成代码的重用性差,代码纠缠交织,给系统开发和维护带来了许多的困难,AOP 则将它们以统一的形式定义为方面(Aspects), Aspects 无须在类中具体描述,它们被通过一个称为编织(Weaving)的过程统一织入到代码中[2]。

不同的软件有不同的非功能特性,如何处理软件非功能特性[3],并对其进行可视化建模是关系到整个软件系统质量的头等大事。传统的软件设计中,软件非功能特性一般都是在系统设计或实现的后期发现并补充,能否在软件需求分析的初期分离出系统的非功能特性并对这些非功能特性做一个整体的处理将有助于加快软件开发的进度,提高生产效率。

1 关注点概述

软件设计无非是对业务关注点的提取和封装,因此必须首先要明白什么是关注点,什么是横切关注点。所谓关注点就是一个所要解决的需求问题,它是软件分析、设计、演化以及重用过程中所关心或感兴趣的事

收稿日期:2010-06-08;**修回日期**:2010-09-25
基金项目:国家自然科学基金重大研究计划(90818008);国家自然科学基金项目(60774095,60474072Z);广东省自然科学基金项目(07001774,04009465)
作者简介:杨汉明(1986-),男,硕士研究生,研究方向为实时系统,面向方面研究;张立臣,教授,研究方向为实时系统,分布式系统。

物,它可以分为核心关注点(Primary Concern)和横切关注点(Crosscutting Concern)。核心关注点封装了软件系统的功能成分;横切关注点封装了软件系统的非功能或约束成分。图 1 为关注点横切关系图。

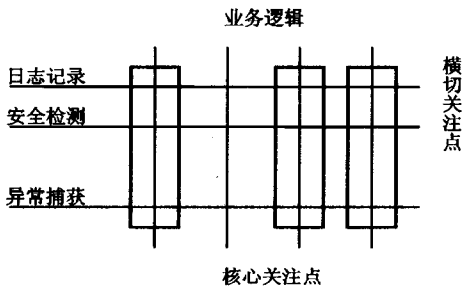

图 1　关注点横切关系图

2　非功能需求与 NFR 框架

需求通常分为功能性需求(Functional Requirements,简称 FRs)和非功能需求(Non-functional Requirements,简称 NFRs)。FRs 描述了软件系统必须实现的功能,它描述的是软件"做什么"这一概念;NFRs 是一类特殊的需求[4],可理解为实现功能性需求的"配套设施",它描述的不是软件"做什么",而是"如何去做",例如软件的安全性,可靠性,可预测性,外部接口需求等系统开发的主要非功能需求。FRs 描述了核心关注点的性质,而 NFRs 则描述了横切关注点的特性。

由于 NFRs 非常特殊和抽象,很少有形式化的描述,缺乏相关的工具支持,与 FRs 分离不易确定等特点,L. Chung 在 1992 年首次提出对非功能需求进行表示、组织和分析的 NFR 框架。

NFR 框架[5]包括以下的几个概念[6]:"软目标"、"相互依赖性""软目标相互依赖关系图"。其中"软目标"表示非功能需求的基本单元,指的是一类没有明确的定义和标准来证明其是否得到满足的目标。在 NFR 框架中有三类软目标[7],分别是 NFR 软目标(NFR Softgoal),操作性软目标(Operationalizing Softgoal)和断言软目标(Claim Softgoal)。NFR 软目标表示的是系统必须满足的高层次非功能需求;操作性软目标则指的是可能的解决方案(如操作、过程,数据表示等)或有助于实现 NFR 软目标的设计选择;最后,断言软目标则是对软目标及其相互依赖性的上下文进行解释的基本原理。在 NFR 框架中,每个软目标有一个相关的类型(type)指示软目标的意义,一个或多个主题(subject)指示软目标的信息项[8]。软目标的紧急程度由"!"表示,"!"表示紧急,"!!"表示非常紧急,可操作性由"√"表示被接受,"×"表示被拒绝;"相互依赖性"指的是软目标间的相互依赖关系,包括父子软目标间的精化关系(采用第一类基值表示,通过 AND/OR 将一个父软目标分解为一组子软目标)和操作性对相应软目标实现所起的作用(采用第二类基值表示,包括强正相关++,正相关+,负相关-,强负相关--);"软目标相互依赖关系图"表示软目标及其相互依赖性关系的图形,如图 2 所示。图 2 中的圆形表示软目标,软目标之间的单圆弧表示"与"关系,双圆弧表示"或"关系。

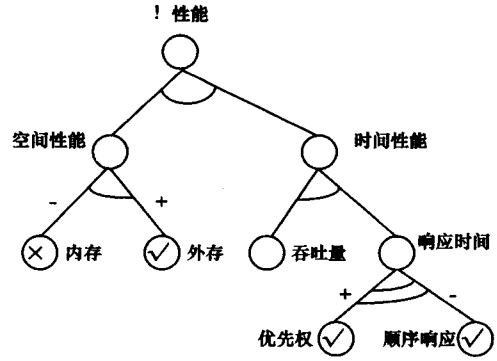

图 2　软目标相互依赖关系图例子

3　关注点分离

关注点分离(Separation of Concerns,简称 SoC)是软件开发中最重要的事情之一,也是软件工程最重要的原则之一。一般来说,关注点分离指的是确定、封装和操作那些与特定概念、目标、任务或目的相关的软件部件的能力[9]。近年,随着 AOP 技术的发展及早期方面(early aspect)的提出和深入研究,已经为关注点分离技术提供了重要的理论基础和技术支持。

3.1　关注点分离模式

关注点分离体现的是一种分而治之的思想,即是对于一个复杂的问题 P,文献[10]将其解决途径归纳为如下步骤[10]:

1) 将待解决问题 P 分解为不同的关注点 $P_1,P_2,\cdots,P_n$,即:$P \rightarrow D(P_1,P_2,\cdots,P_n)$,$D$ 表示问题分解策略,即关注点分离方法;

2) 对 $P_1,P_2,\cdots,P_n$ 分别考虑,求得各自的解 $S_1,S_2,\cdots,S_n$;

3) 通过合成 $S_1,S_2,\cdots,S_n$,求得问题 P 的解 S,即:$C(S_1,S_2,\cdots,S_n) \rightarrow S$,$C$ 表示解的合成方法。

3.2　关注点分离过程

上面的分解步骤并不是针对某个特定问题的完整解决方案。对于具体问题,可以结合问题的背景,运用相应的分离策略对关注点进行分离。关注点的分离策略有很多种,如基于概念空间,基于需求规约,基于用例等等。下面就基于需求规约和用例,描述软件系统

关注点分离的一般过程：识别关注点、描述关注点、组合关注点。如图3所示。

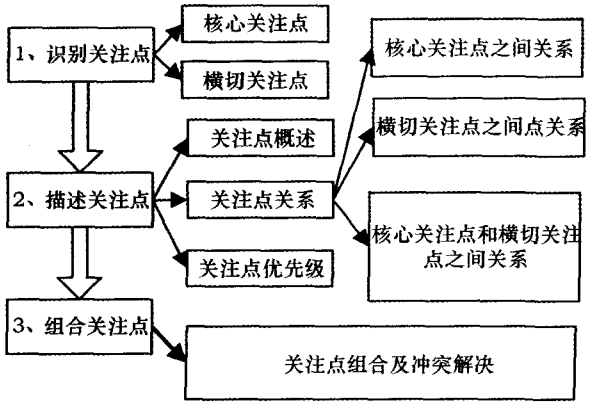

图3 关注点分离一般过程

3.2.1 关注点识别

从上面3.1的关注点分离模式来看关注点分离过程，识别关注点就是识别步骤1中的关注点 $P_1,P_2,\cdots,P_n$，关注点的识别是建立在需求规约信息的基础之上。

关注点识别的方法也有很多，其中比较有效的是基于用例的关注点识别。正式地讲，用例是一个类元构造，它描述了特定参与者（用户）类型是如何使用系统的所有相关用法的集合[11]。一个用例就是用户与系统的一次完整的交互过程，它从执行者的角度观察系统需要提供的功能，描述了系统的一个方面。所有的用例组合起来就可以得出系统所提供的完整功能。基于用例的关注点识别过程可以归结为：分析需求规约、确定系统用例、分析用例关系并标识关注点。

step1：identify NFR_{decom}，$SG_{top}=NFR$；

step2：$SG_{top}\xrightarrow{and/or}\{SG_1,\cdots,SG_i\}(i\leqslant 1)$，
num：=i；

step3：if(i≤num & SG_i，need_decom) then{
$SG_i\xrightarrow{and\ i\ or}\{SG_{num+1},\cdots,SG_{num+j}\}$；
num：=num+1；}
else if (SG_i not need_decon) then{
i：=i+1；goto step3；}
else if (i>num) then goto step4；

step4：stop.

图4 NFR分解算法步骤

功能性关注点从用例中很容易分析，本文重点是关注非功能性关注点的处理。为了较好的标识并模块化NFRs，可以运用文献[6]提供的分解算法[6] NFR_DECOM对初步确定的NFRs进行细分，其中 SG_{top} 表示顶级软目标，SG_i 表示分解后的软目标，num表示分解的数量，如图4所示。

非功能性关注点在分解后需要确定其可能的操作性，其确定的步骤和上面的分解过程比较相似，也是一个迭代的分解过程。

3.2.2 关注点描述

关注点的描述可以对应上面步骤2的 $S_1,S_2,\cdots,S_n$，关注点描述清楚就能初步确立问题的解。关注点描述问题包括对各个关注点之间关系的描述，其中包括功能性关注点之间的关系、非功能性关注点之间的关系以及功能性和非功能性关注点之间的关系。确定非功能关注点相互关系及其相关性问题可以用下面的矩阵来表示，其中“+”表示正相关，“-”表示负相关（见表1）：

表1 非功能需求影响矩阵

NFR \ NFR	NFR1	NFR2	…	NFRn
NFR1		-		+
NFR2	-			-
…				
NFRn	+	-		

提取出上面的相互冲突的非功能性关注点，其对功能性关注点的影响用[0,1]之间的数值表示，根据最后影响数值的大少来确定其优先权的大少，数据越大优先权越大，反之亦然（见表2）。

表2 冲突的非功能需求优先权矩阵

FR \ NFR	FR1	FR2	…	FRn
NFR1	a_1	b_1	…	n_1
NFR2	a_2	b_2	…	n_2
…	…	…	…	…
NFRm	a_m	b_m	…	n_m

注：$0\leqslant a_i\leqslant 1,0\leqslant b_i\leqslant 1,\cdots,0\leqslant n_i\leqslant 1$，其中 i 为整数，$1\leqslant i\leqslant m$。

3.2.3 关注点组合

关注点的组合可以对应上面步骤3的中的 S，组合的过程其实就是如何确定连接点的过程。连接点模型定义了以下一些连接点[9]：

（1）方法或构造器的执行。
（2）通知的执行。
（3）对方法或构造器的调用。
（4）对一个域的读或写方法。
（5）异常处理（catch块）的执行。
（6）类的静态初始化。
（7）对象或方法的初始化。

如果有多于一个非功能需求应用于给定的连接点并且所涉及的非功能需求互相起否定影响的话将会产生冲突的行为[12]。冲突行为可以通过确定优先权的

办法来解决,如果冲突的非功能需求优先权相同则可以通过与用户协商进行解决。

4 结束语

软件非功能特性的处理长期以来困扰研究人员和开发人员,成为软件工程领域一个迫切解决的问题,文中通过利用面向方面的思想,对软件进行关注点的分离,对分离处理的关注点利用了著名的 NFR 框架进行描述处理,通过这些方法在一定程度上很好的解决了软件非功能特性的处理。后期工作是利用 UML Profile 扩展机制对 NFR 框架的描述方法进行补充[8],使其图形化的同时不失语义的精确性,也可以对其进行必要的定量描述;非功能需求的形式化描述也是本文后期的主要研究工作。

参考文献:

[1] GRADECKI J D, LESIECKI N. 精通 AspectJ[M]. 王欣轩, 吴东升, 等译. 北京:清华大学出版社, 2005.

[2] Filman R E. 面向方面软件开发[M]. 莫　倩等译. 北京:机械工业出版社, 2006.

[3] 龙湘明, 孙其博, 苏　森. 使用 UML 扩展机制描述软件非功能需求的新方法[J]. 计算机工程, 2005, 31(14):60-63.

[4] CHUNG L, NIXON B A, YU E, et al. Non-Functional Requirements in Software Engineering[M]. Norwell: Kluwer Academic Publishers, 2000.

[5] LIU XIAOMEI, LIU SHULIN, ZHENG XIAOJUAN. Adapting the NFR Framework to Aspectual Use-case Driven Approach[C]//2009 Seventh ACIS International Conference on Software Engineering Research, Management and Applications, 2009. USA: IEEE, 2009: 209-214.

[6] 郑旭飞. 一种面向方面的非功能需求框架 AONFRF 建模研究[D]. 重庆:西南师范大学, 2005.

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

[8] CHUNG L, MYLOPOULOS J, NIXON B. Representing and Using Non-Founctional Requirements: A Process-Oriented Approach[J]. IEEE Transactions on Software Engineering, 1992, 18(6):483-497.

[9] FILMAN R E, ELRAD T, CLARKE S, et al. 面向方面的软件开发[M]. 莫　倩, 王　恺, 刘冬梅, 等译. 北京:机械工业出版社, 2005.

[10] 何明昕. 关注点分离在计算思维和软件工程中的方法论意义[J]. 计算机科学. 2009, 36(4):60-63.

[11] JACOBSON I, NG PAN-WEI. AOSD 中文版——基于用例的面向方面软件开发[M]. 徐　锋译. 北京:电子工业出版社, 2005.

[12] 张琳琳, 应　时, 赵　楷, 等. 一种建模软件体系结构非功能属性的方法[J]. 计算机科学, 2009, 36(7):92-96.

(上接第 111 页)

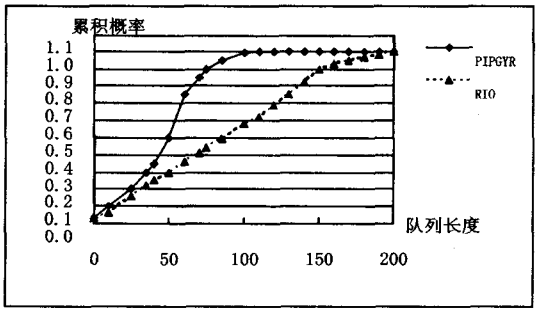

图 3　PIPGYR 和 RIO 队列长度比较

3 结束语

文中结合 PIP 算法和三色标记器的功能,提出了 PIPGYR 算法,仿真分析表明: PIPGYR 算法的队列长度比较稳定,平均分组丢失率低,同时能够更好地保护高优先级报文。

参考文献:

[1] 张明杰, 朱培栋, 卢锡城. DSOCC: 面向区分服务的拥塞控制[J]. 国防科技大学学报, 2004, 26(3):29-33.

[2] 章　森, 吴建平, 林　闯. P2I :一种新的主动队列管理算法[J]. 计算机学报, 2003, 26 (10):1288-1294.

[3] 林　闯, 单志广, 盛立杰. 区分服务及其几个热点问题的研究[J]. 计算机学报, 2000, 23(4):419-433.

[4] 李向丽, 陈　圆, 邱保志. 基于区分服务 AF PHB 的主动队列管理算法机制[J]. 郑州大学信息工程学报, 2004, 22(2):13-16.

[5] 钱光明. 基于业务的多优先级队列区别服务方案[J]. 计算机工程与应用, 2006, 42(10):118-120.

[6] 徐昌彪, 鲜永菊. 计算机网络中的拥塞控制与流量控制[M]. 北京:人民邮电出版社, 2007.

[7] Blake S, Black D, Carlson M, et al. An Architecture for Differentiated Services[S]. RFC 2475, 1998.

[8] Jacobson V, Nichols K. An Expedited Forwarding PHB[S]. RFC 2598, 1999.

[9] Heinanen J, Baker F, Weiss W, et al. Assured Forwarding PHB[S]. RFC2597, 1999.

[10] Clark D, Fang W. Explicit Allocation of Best Packet Delivery Service[J]. IEEE/ACM Trans. on Networking, 1998, 6(4): 362-373.

[11] Heinanmen J, Baker F, Weiss, et al. Assured Forwording PHB group[S]. RFC 2597, 1999.