第16卷 第4期
2006年4月

计算机技术与发展
COMPUTER TECHNOLOGY AND DEVELOPMENT

Vol. 16 No. 4
Apr. 2006

基于 Viewpoint 的软件构架描述

章 静,李心科,吴 蕾
(合肥工业大学 计算机学院,安徽 合肥 230009)

摘 要:软件构架作为系统开发的蓝图,已成为软件工程领域的热点。在软件开发实践中,许多项目更为关注的是软件构架的设计,而忽略了软件构架的描述。事实上软件构架的详细精确描述是成功开发软件的根本保证。文中以 UML 为构架描述语言,引用 IEEE 1471 标准中 Viewpoint 的概念,提出了基于 3 大 Viewpoint 的软件构架描述方法。基于 Viewpoint 的软件构架描述方法有助于开发人员在大型项目开发中从更高层次把握系统构架,从而保证软件开发过程的顺利进行。

关键词:软件构架;viewpoint;UML;视图;描述
中图分类号:TP311.5 **文献标识码**:A **文章编号**:1005-3751(2006)04-0022-03

Software Architecture Description Based on Viewpoint

ZHANG Jing, LI Xin-ke, WU Lei
(Computer School, Hefei University of Technology, Hefei 230009, China)

Abstract: Serving as a blueprint for system development, software architecture has become the center of a frenzy of attention these days. At present, much attention has always been given to the design of software architecture. However, software developers often ignore the great importance of software architecture description. In fact, software architecture description in full detail without ambiguity is a leading factor in successful software development. With UML as architecture description language, this paper has proposed an approach to describe software architecture on a basis of three viewpoints proposed by the IEEE 1471 standard. Software architecture description based on viewpoint is very helpful for software developers to master the system architecture in large-scale software development. Therefore, the software development process can be run smoothly.

Key words: software architecture; viewpoint; UML; view; description

0 引 言

随着软件系统的规模越来越庞大、复杂程度越来越高,软件构架在软件开发中的作用也越来越突出。对于大型软件系统而言,软件构架已经成为系统和系统开发的蓝图,指导开发后期的维护和挖掘工作[1]。软件构架描述是软件架构的主要步骤之一,如果表达不明确,再完美的软件构架也毫无用处[2]。对软件构架进行详细、明确和有组织的说明可有助于其他开发人员获取所要的信息,促使开发过程顺利进行。

作为软件构架描述的基本单元,视图是对一组系统元素和它们间的关系的表示。视图从不同角度反映了系统的各个侧面。2000 年,IEEE 1471 标准引用了"Viewpoint"的概念来描述许多系统的通用描述性框架[3]。Viewpoint 是起草可重复使用、特定领域体系结构描述标准的工具。它建立了用来创建一个视图的语言或概念用于解释它的约定以及用于视图的任何相关分析方法。每个 Viewpoint 有其具体的建模目标和系统相关者。与 Viewpoint 相比,视图是一个能描绘整个系统的一个方面的模型的集合,仅应用于一个系统,而不能在许多系统上普及。Viewpoint 则从更抽象的角度来描述软件构架。Viewpoint 和视图的关系类似于面向对象中类与对象的关系。

文中以 POS 系统为例,从 IEEE 1471 标准[3]提出的 Viewpoint 出发,利用 UML[4]对软件构架进行了描述。

1 描述系统构架的三大 Viewpoint

构架师通常会从以下 3 个角度来考虑开发的软件:

(1)作为一个实现单元集,它是如何构造的;

(2)作为一个具有运行时行为和交互作用的元素集,它是如何构造的;

(3)它是如何在自己所处的环境内与非软件结构产生联系的。

这 3 个方面分别对应着:Module Viewpoint, Component &Connector Viewpoint, Allocation Viewpoint[5]。每个 Viewpoint 的不同风格限制了其对应视图中存在的元素集和关系集。

1.1 Module Viewpoint

系统软件通常可分解成许多可管理的功能单元。通

收稿日期:2005-07-22
基金项目:安徽省教育厅自然科学研究项目(2004kj136zd)
作者简介:章 静(1982-),女,安徽绩溪人,硕士研究生,研究方向为软件工程、软件体系结构;李心科,副教授,研究方向为软件工程。

常把这些功能单元称为模块。Module Viewpoint[5]关注的是软件功能的划分。一般不用 Module Viewpoint 来分析性能、可靠性和其他运行时的属性。

Module Viewpoint 的组成元素是模块。模块间的关联关系包括部分关系、依赖关系、特化关系。视点中的视图可采用 UML 中的类图、包、接口以及 UML 中的关联关系来描述。该视点的主要目的是分解系统功能。Module Viewpoint 中的视图通常会与组件和连接器 Viewpoint 中的视图进行直接映射。

Module Viewpoint 有 4 大常用风格:

①分解风格:用于集中处理模块间的包容关系。

②使用风格:表示模块间的功能依赖性关系。

③泛化风格:表示模块间的特化关系。

④分层风格:表示模块间受限制的"允许使用"关系。

1.2 Component&Connector Viewpoint(C&C Viewpoint)

组件和连接器是软件体系结构中的首要概念。连接器是一种特殊的构件,负责构件与构件间的交互。构件和连接器视图描绘了一幅运行时实体及它们交互机制的操作图。这些实体包括进程、对象、客户机、数据存储器等等。

C&C Viewpoint[5]忽略了有关构件分配和线程实现方面的问题;集中于构件间功能的分配问题以及构件间的接口问题。

C&C Viewpoint 的组成元素为组件和连接器。元素间的关联体现在组件的端口与特定的连接器间的关联。该视点的主要目的在于描述实现时期构件间的连接和通信,用于性能分析和后期的过程交互设计。

通常利用以下 3 种策略描述 C&C Viewpoint 中的视图:

(1)以 UML 类表示组件类型,对象表示组件实例。

(2)将组件类型表示为 UML 子系统,组件实例表示为子系统实例。

(3)采用提供第一种策略变体的 UML 语义框架。

常用的 C&C 风格有:管道过滤器风格;共享数据风格;发布订阅风格;客户机服务器风格;对等连接风格;MVC 风格。

1.3 Allocation Viewpoint

Allocation Viewpoint 关注的是软件元素到环境结构的映射,其中最常用的是部署视图。部署视图描述了组件连接器到硬件节点的一一映射关系。尽管通常认为部署视图是系统结构的一部分,而不属于软件构架考虑的范畴。但硬件往往会影响组件的设计和结构,所以开发团队在开发时通常会考虑硬件因素。

Allocation Viewpoint 的组成元素包括软件元素和环境元素。软件元素通常被分配到相应的环境元素中去。该视点的主要目的在于把软件构架的元素映射到硬件上,有助于系统性能分析和项目的管理。Allocation Viewpoint 的部署视图是建立在组件和连接器视图的基础之上的。通常部署风格则用 UML 中的部署图表示,而实现风格和工作任务风格常常用非正规表示法。

Allocation Viewpoint 的 3 类主要风格:部署风格、实现风格、工作任务风格。

2 POS 系统的构架描述

POS 系统(Point of Sale)是一种常用于零售业记录销售信息和处理支付过程的计算机应用。该系统包括计算机、条码扫描仪等硬件和系统软件。POS 系统还可为不同的服务应用程序提供接口,有较强的容错能力。同时,POS 系统有较强的可用性及良好的用户界面,从而方便收银员高效地进行业务操作。

下面利用三大 Viewpoint 描述 POS 系统架构。

2.1 Module Viewpoint

应用 Module Viewpoint 的分解风格,把整个系统进行逐步划分。图 1 把 POS 系统分解成 5 个子系统。图 2 对 DomainModel 子系统进行进一步的分解。而图 3 标识了 Payments 子系统的主要构件集合。(由于篇幅的限制,没有列出所有的子系统的分解图)

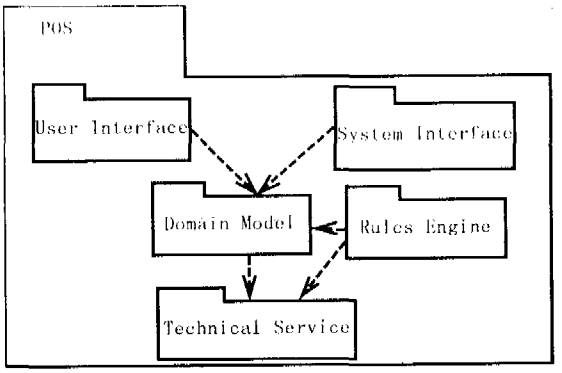

图 1　POS 系统领域包图

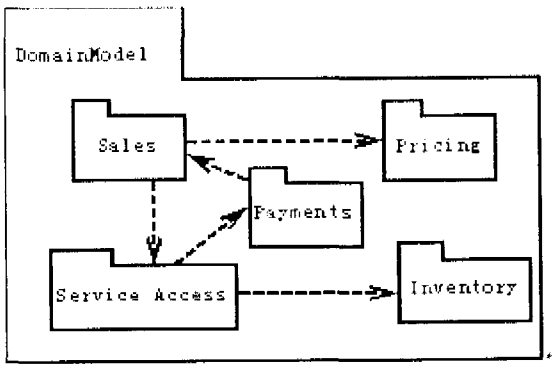

图 2　领域模型的分解图

分解风格是几乎所有的软件构架开始时都要采用的风格。分而自治的原则在这里得到了充分的体现。

2.2 C&C Viewpoint

如图 4 所示,POS 系统主要由 4 个功能组件组成:

●Barcode Scanner:

1)控制并与扫描仪进行通信;

2)当扫描仪硬件扫描时可把商品的序列号反馈给

POS 系统。

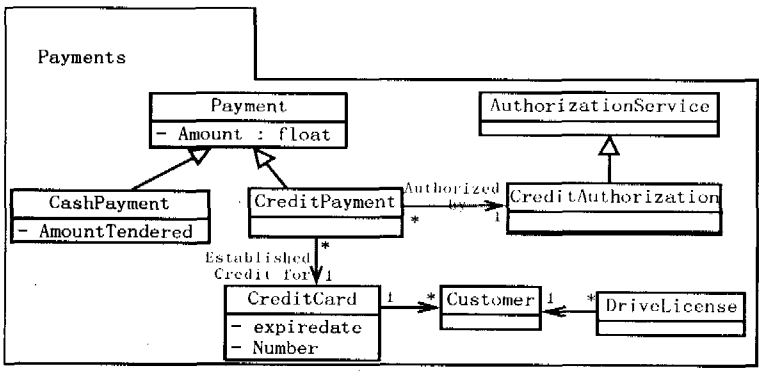

图 3 Payments 子系统的分解

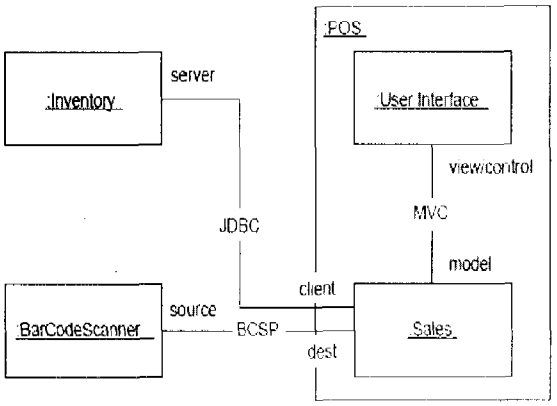

图 4 POS 的 CAC 视图

●Sales:

1)记录扫描过的商品的数量、价格,并计算销售总额;

2)初始化和结束 sales 过程。

●Presentation:

1)在终端展示商品的名称、数量、单价和总价。

2)可以打印小票。

●Inventory:

1)记录商店的库存;

2)可以在 ID 与商品名称和单价两者间进行映射。

在图 4 中,3 个连接器负责了上述组件间的交互。

①MVC:MVC 模式定义了该连接器应遵循的协议. Sales 构件在系统中充当的是 Model 角色;Presentation 则负责控制和视图(Control 和 View)。

②JDBC:JDBC 连接器规范了标准 SQL 查询应遵循的协议。

③BSCP:BSCP 连接器规定了主系统与扫描仪连接应遵循的协议。

在开发实践中,开发者可借助 UML 中的时序图来进一步描述有关连接器应遵循的协议。

2.3 Allocation Viewpoint

这里主要考虑的是部署风格。

图 5 展示了 POS 系统的 UML 部署图。该部署图包含下列硬件元素和软件元素。

* 硬件元素。

①扫描仪:用来输入商品的有关信息。它通过 RS232 连接器与 POS 终端相连。

②销售终端:用户与系统的主要交互平台。

③应用服务器:向所有的应用终端提供应用服务。

④数据库服务器:提供对数据的存储。

* 软件元素。

①POS executable 部件:运行在 POS 系统的客户端。负责与外部设备进行交互。它通过 RMI 与应用服务器进行通信。

②JBoss:一个开放的应用服务器,它主要完成应用逻辑层的功能,通过 JDBC 与数据库服务器进行通信。

③MySQL:一个开放的 SQL 数据库,主要完成数据库的通用功能。如存储数据、事务处理、并行控制等。

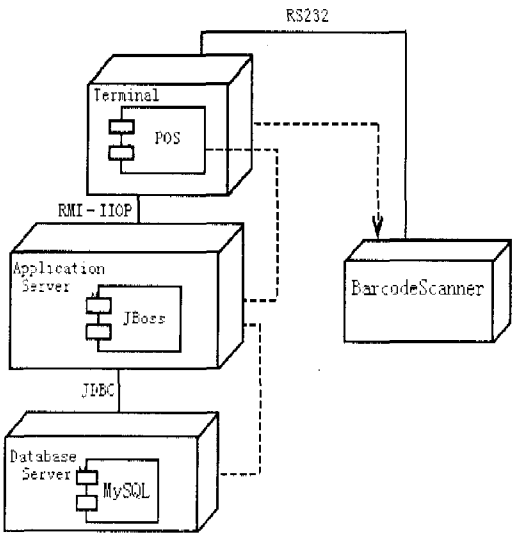

图 5 POS 系统的部署图

该部署图是典型的 3 层部署图:界面、应用逻辑、存储。3 层架构的显著特点是分离了应用逻辑,将其作为独立的逻辑中间层软件。应用逻辑运行在 J2EE 的平台上。

3 结 论

文中从 Viewpoint 的角度出发,分析了如何用通用的 UML 来描述软件构架。作为 IEEE 1471 标准[3]的核心概念,Viewpoint 是一种通过规定视图的用途和创建分析视图的技术开发单个视图时的模式或模板。视图是根据其 Viewpoint 的需求对系统的表示,是描述软件构架的基本单元。任何视图都隶属于上述 3 大 Viewpoint[5] 中的一类。作为视图的抽象,Viewpoint 在描述软件构架时也会出现信息冗余的情况。

UML 是软件开发业中广泛应用的标准建模语言。它的易用性、通用性得到许多业内人士的认同。由于本身缺少分析软件构架的语义,UML 并不是软件构架建模的最佳选择。但是 UML 的通用性促使它成为一种广泛的软

(下转第 27 页)

$[\lceil \log l \rceil]$,投影 $\mathrm{pr}(X_1, F)$ 由 $s(a)$ 的每一个集合 X 的副本组成,集合 X_1 对应于一个自然数 a 的二进制编码,每一个 $X(a) \subseteq X_1$,且 $a \in A = [3l]$。投影 $\mathrm{pr}(X_2, F)$ 由 $s(a)$ 的每一个集合 X 的副本组成,集合 X_2 对应于一个自然数 a 的二进制编码,每一个 $X(a) \subseteq X_1, b \in [l]$。

显然,对于 3 个部分问题(A, B, s)存在的充要条件是在两个投影 $\mathrm{pr}(X_1, F_1)$ 和 $\mathrm{pr}(X_2, F_2)$ 中,每一个都存在着一个具有 $3l$ 个不同集合的兼容数据集。

4 结　论

分析了频繁集挖掘的可计算复杂度问题,给出了在许多情况下问题的计算是困难的。对于已知的频繁集,用户不能计算出与已知频繁集兼容的不同数据集的数量,甚至不能决定是否存在着相容数据集。从隐私保持的观点看,频繁集的传递不会引起严重的隐私威胁,但另一方面反频繁集挖掘的计算是困难的。

参考文献:

[1] Mannila H. Local and global methods in data mining: Basic techniques and open problems[A]. In: Widmayer P, Triguero F, Morales R, et al. Automata, Languages and Programming, volume 2380 of Lecture Notes in Computer Science[C]. [s. l.]: Springer - Verlag, 2002. 57 - 68.

[2] Evfimievski A, Srikant R, Agrawal R, et al. Privacy preserving mining of association rules[A]. In Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining[C]. Edmonton, Alberta, Canada: ACM Press, 2002. 217 - 228.

[3] Oliverira S R M, Zaï O R. Privacy preserving frequent itemset mining[A]. In: Clifton C, Estivill - Castro V. IEEE ICDM Workshop on Privacy, Security, and Data Mining, volume 14 of Conferences in Research and Practice in Information Technology[C]. Maebashi City, Japan: [s. n.], 2002. 43 - 54.

[4] Gouka K, Zaki M J. Efficiently mining maximal frequent itemsets[A]. In: Cercone N, Lin T Y, Wu X. Proceedings of the 2001 IEEE International Conference on Data Mining[C]. Washington, DC: IEEE Computer society, 2001. 163 - 170.

[5] Gunopulos D, Khardon R, Mannila H, et al. Discovering all most specific sentences[J]. ACM Transactions on Database Systems, 2003, 28(2): 140 - 174.

[6] Calders T, Goethals B. Minimal k - free representations of frequent sets[A]. In: Lavrac N, Gamgerger D, Todorovski L, et al. Knowledge Discovery in Databases: PKDD 2003, volume 2838 of Lecture Notes in Artifical Intelligence[C]. [s. l.]: Springer - Verlag, 2003.

[7] Garey M R, Johnson D S. Computers and Intractability: A Guide to the Theory of NP - Completeness[Z]. New York - San Francisco: W. H. Freeman and Company, 1979.

[8] Knuth D E. Sorting and Searching, volume 3 of The Art of Computer Programming (2nd ed)[M]. Reading, Massachusetts: Addison - Wesley Publishing CO., 1998.

[9] Jukan S. Extremal Combinatorics: With Applications in Computer Science[A]. EATCS Texts in Theoretical Computer Science[C]. Berlin: Springer - Verlag, 2001.

[10] Gali Z. Efficient algorithms for finding maximum matchings in graphs[J]. ACM Computing Surveys, 1986, 18(1): 23 - 38.

(上接第 24 页)

件构架建模语言[6]。

软件构架不仅是系统开发项目的蓝图,而且也是将项目所有阶段结合在一起的概念纽带。对软件构架进行准确详尽的描述,是构建软件构架的最重要的步骤之一[7]。但在开发实践中,开发人员往往更关注于软件构架的设计,忽视了软件构架的描述。对再完美的软件构架设计如果没有进行详细明确有组织的描述,这个软件构架对整个开发过程也是没有太大帮助的。

Module, C&C, Allocation Viewpoint[5]是对描述软件构架的视图的高度抽象和概括。对于大型项目来说,Viewpoint 可以有助于开发人员从更高层次上把握整个系统构架,也更易对架构进行详尽准确的描述。

随着 UML2.0[8]的颁发和 IEEE 1471 的采用,基于 Viewpoint 的软件构架描述将更为广泛地应用于软件开发实践中。

参考文献:

[1] 万建成,卢　雷. 软件体系结构的原理、组成与应用[M]. 北京:科学出版社,2002.

[2] 冯　冲,江　贺. 软件体系结构理论与实践[M]. 北京:人民邮电出版社,2004.

[3] IEEE Computer Society. IEEE recommended practice for architectural description of software - intensive systems[R]. IEEE Std 1471, 2000.

[4] OMG (2003). *Unified Modeling Language specification 1.5* [EB/OL]. http://www.omg.org/uml/, 2001.

[5] Clements P, Bachmann F, Bass L. Documenting Software Architectures: Views and Beyond[M]. [s. l.]: Addison - Wesley publication house, 2002.

[6] Medvidovic N, Rosenblum D S, Redmiles D F. Modeling Software Architectures in the United Modeling Language[J]. ACM Transactions on Software Engineering and Methodology, 2002, 11(1): 52 - 57.

[7] Clements P, Kazman R, Klein M. Software Architecture in Practice(2 edition)[M]. [s. l.]: Addison - Wesley publication house, 2003.

[8] OMG. United Modeling Language[EB/OL]. http://www.omg.org/uml/, 2003.