支持运行监控的可信软件体系结构设计方法

文 静[1)] 王怀民[2)] 应 时[1)] 倪友聪[1)] 王 涛[2)]

[1)](武汉大学软件工程国家重点实验室 武汉 430072)

[2)](国防科学技术大学计算机学院网络与信息安全研究所 长沙 410073)

摘 要 近年来,软件的可信性成为软件质量的焦点,对软件可信性的分析、度量和应用支撑成为热点问题.对软件实施有效的监控是提升软件可信性的一种重要途径.然而目前的研究工作主要集中在软件编码以及相关技术的实现层,缺乏一套系统的软件体系结构设计方法以指导、支持运行监控的可信软件的分析和设计.通过引入面向侧面的软件体系结构设计方法及其相关概念,文中提出一种支持运行监控的可信软件体系结构设计方法.在支持运行监控的可信软件构造模型 TSCM 的基础上,利用一种面向侧面的体系结构描述语言 AC2-ADL 描述具有监控能力的软件体系结构,试图为分析和设计具有监控能力的系统的软件体系结构提供一种有效的解决方案.通过结合网上拍卖系统的案例展示该方法的主要步骤和结果,讨论了研究中存在的问题和进一步的工作.

关键词 面向侧面的体系结构描述语言;面向侧面的软件开发方法;支持运行监控的可信软件构造模型;软件工程;可信软件开发

中图法分类号 TP311 **DOI 号**: 10.3724/SP.J.1016.2010.02321

Toward a Software Architectural Design Approach for Trusted Software Based on Monitoring

WEN Jing[1)] WANG Huai-Min[2)] YING Shi[1)] NI You-Cong[1)] WANG Tao[2)]

[1)](*State Key Laboratory of Software Engineering*, *Wuhan University*, *Wuhan* 430072)

[2)](*Institute of Network Technology and Information Security*, *School of Computer*, *National University of Defense Technology*, *Changsha* 410073)

Abstract Nowadays, the trustworthiness has become the noticed focus in qualities of software, which emphasizes analysis, measurement, corresponding applications and support environments for the trustworthiness quality. Injecting monitoring capacity into software can be one of the approaches to improve the credibility of software. However, the current work primarily concentrates on the programming and coding level, lacking a systematic architectural design approach to guide the whole design and analysis process for the trusted software. By introducing aspect-oriented architectural design approach and relevant techniques into the design and analysis of software, this paper tries to offer an effective and systematic solution for the trusted software system. Based on trusted software constitution model with the capability of monitoring, the proposed approach utilizes a special kind of aspect-oriented architecture description language AC2-ADL to depict such software system, contributing to comprehension, evolution and reusability of software architectural design decisions. Summarily, the whole designing process of the approach is discussed systematically through a case study in e-business domain.

收稿日期:2008-11-28;最终修改稿收到日期:2010-08-02.本课题得到国家"八六三"高技术研究发展计划项目基金(2003AA142O1O,2006AA01Z168)和国家自然科学基金(60773006)资助.**文 静**,女,1982 年生,博士研究生,主要研究方向为体系结构、面向方面软件开发、人机交互. **王怀民**,男,1962 年生,教授,博士生导师,主要研究领域为分布式计算、网络与信息安全. E-mail: jingwen_1982@yahoo.com.cn. **应 时**(通信作者),男,1965 年生,博士,教授,博士生导师,主要研究领域为面向对象软件工程方法、基于组件的软件工程方法、软件体系结构和模式、软件的可重用性与互操作性等. E-mail: yingshi@whu.edu.cn. **倪友聪**,男,1979 年生,博士研究生,主要研究方向为体系结构、面向方面软件开发、软件工程. **王 涛**,男,1984 年生,博士研究生,主要研究方向为分布式计算和面向方面编程.

Keywords aspect-oriented architecture description language; aspect-oriented software architecture design; trusted software constitution model based on monitoring; software engineering; trusted software development

1 引 言

软件无所不在.然而,软件的应用需求愈来愈多,复杂度愈来愈高,可用性要求愈来愈强,日趋庞大的软件系统并不总是可以让人信任.随着对软件质量要求的日益提高,"可信"成为了当前软件质量研究和实践的关键特性.可以把"软件可信性"定义为软件的行为和结果符合使用者预期的程度,而可信软件生产技术则是以提高软件可信性为主要目的的软件生产技术.

支持运行监控的可信软件构造模型 TSCM (Trusted Software Constitution Model based on Monitoring)[1]是一种提高软件可信性的软件生产技术.该模型认为对软件实施有效的监控以获得软件的运行状态,并通过将监控结果与预期行为进行比较,可以有效地掌握软件行为的可信程度,且有助于准确分析和定位软件故障.因此在软件中以适当的方式注入监控能力,将是提升软件可信性的一种重要途径.但是软件的可信性是贯穿软件生产和运行全过程的质量属性,通常是软件系统的全局约束,需要伴随着软件开发过程逐步地设计出来[2].而目前的研究工作主要集中在软件编码以及相关技术的实现层[3-5],缺乏一套系统的软件体系结构设计方法以指导支持运行监控的可信软件的分析和设计.

近年来,软件体系结构(Software Architecture, SA)的研究受到了广泛的关注和重视.在 SA 的研究中,通过显式地表示系统的体系结构以及其中的组成元素,如构件、连接件等,可以在高抽象层次处理诸如全局组织和控制结构、功能到计算单元的分配、计算单元间的高层交互等设计问题[6].然而常规的软件体系结构设计方法几乎都是以业务过程和用例脚本为主要关注点,设计出软件体系结构的基本结构.然后在软件体系结构中以直接横切构件和连接件的形式,将与系统相关的可信性设计方案加入到基本结构中去.由于没有简单规范地考虑满足系统可信性的设计步骤,使得设计过程变得复杂起来.此外由于非规范地加入满足这些设计方案,模糊了构件和连接件的角色与边界,使得设计结果也变得难以理解.因此,运用分而治之的策略,构建可信软件系统的体系结构,是设计可信软件的关键.

随着面向侧面的软件开发(AOSD)方法的出现以及相关技术的不断发展与成熟,该方法成为可信软件生产技术发展的契机.该方法充分体现了关注点分离(Separation of Concerns, SoC)的思想,通过使用侧面(aspect)及其相关概念封装和管理分布在软件开发周期各个阶段的横切属性,有利于提高软件产品的质量,并减少适配、演化和维护的开销.因此,在软件体系结构层,使用侧面建模和管理软件的可信性,可进一步提高软件体系结构设计方案的可理解性、可演化性和重用性.

本文为分析和设计具有监控能力的软件系统的软件体系结构提供了一种有效的解决方案.在支持运行监控的可信软件构造模型的基础上,该方法利用一种面向侧面的体系结构描述语言 AC2-ADL 描述具有监控能力的软件系统,并研究结合电子商务领域的网上拍卖系统,讨论了主要设计过程,对可信软件的体系结构设计具有一定的参考作用.

2 相关研究工作

首先介绍面向侧面的软件体系结构设计领域中主要的研究工作,然后给出 TSCM 模型的基本思想.

2.1 面向侧面的软件体系结构设计方法

随着面向侧面的程序设计 AOP(Aspect-Oriented Programming)[7]技术的成熟,在软件生命周期早期阶段进一步研究和开发侧面技术[8],逐渐引起人们的重视.在软件体系结构设计阶段以及在软件体系结构的抽象层次上,研究和应用侧面技术对研究关注点的建模具有重要意义.

首先,利用面向侧面的设计方法设计出的软件体系结构可以解决软件体系结构设计方案中各种设计决策、行为和特征混杂与分散在软件体系结构层多个构件和连接件中的问题.这些横切的内容可以单独地被模块化,使软件体系结构关注点能被直接和明确地体现和跟踪,可以提高软件体系结构的可理解性和可维护性,有利于软件体系结构的演化和重用.其次,利用面向侧面的设计方法设计出的软件体系结构,可以为后续阶段使用 AOP 实现技术提

供获取和标识代码层侧面的线索，有利于 AOP 编码技术的高效应用，有利于 AOP 编码阶段与软件体系结构设计阶段更顺利地衔接.

目前，面向侧面的软件体系结构研究已有了一些代表性的工作. Garcia[9] 等人提出了一个面向侧面设计与特殊领域相结合的典型案例. 他们将面向侧面的体系结构引入到多 agent 的系统设计中，使用侧面来表达那些正交和横切于系统的 agent 的功能. 此外，Pinto[10] 等人提出了基于构件和侧面的软件体系结构描述语言 DAOP-ADL. 该语言根据组成软件系统的构件、侧面及其交互关系，描述一个应用系统的软件体系结构，并通过动态的面向侧面平台 DAOP 支持构件中侧面的动态织入. Batista[11] 等人则通过扩展 ACME 语言来描述侧面连接件以呈现体系结构层侧面与构件之间复杂的交互. 北京大学的梅宏教授[12] 等人于 2003 年 4 月在《软件学报》上发表了论文“ABC：基于体系结构、面向构件的软件开发方法”. 该论文为了更好地基于构件描述软件体系结构，应用了面向侧面的方法和技术. 广东工业大学的张立臣教授[13] 等人从结构建模、行为建模、侧面织入及代码产生等几个方面研究了面向侧面的建模方法.

在这些研究工作的基础上，本文引入一种面向侧面的体系结构设计方法，并利用面向侧面的描述语言 AC2-ADL 描述面向侧面的软件体系结构. 该方法既为后续阶段的软件开发提供有力的线索和保证，也为最终软件在运行时的监控结果的分析和验证提供可追踪的依据.

2.2　支持运行监控的可信软件构造模型

支持运行监控的可信软件构造模型 TSCM 是由国防科学技术大学可信软件研究组研发的一种可信软件生产方法：通过注入机制为软件注入监控能力，并以此为基础通过软件运行监控机制提供软件运行的监控能力. TSCM 可以为一类原本没有监控能力的软件系统注入监控能力，并能在系统运行时有效地对系统的行为、状态进行监控. 此外 TSCM 还利用源代码表示技术和 AOP 技术，依靠一系列软件开发工具，帮助编程人员表达监控需求，自动生成实现监控的侧面代码，并最终将监控代码织入软件系统内部，使得软件系统具备监控能力.

图 1 展示了该模型的总体结构，TSCM 分为两大部分共 5 个模块：源代码分析模块、监控需求表达模块、监控代码生成模块和监控代码织入模块组成第一部分，主要完成监控能力注入的功能；支撑工具为第二部分，主要为其它 4 个模块功能的实现提供模板和工具支持：

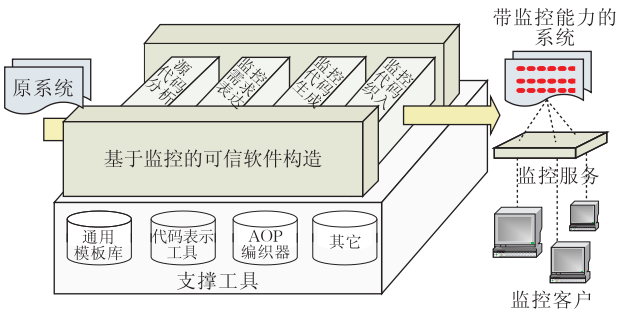


图 1　基于监控的可信软件构造模型 TSCM

(1) 源代码分析模块. 负责提取待监控软件系统的源代码信息. 通过利用代码表示工具对原系统代码进行分析，源代码分析模块将其转化为 XML 的中间形式，并利用 XML 解析器将代码中的类、函数、变量等信息提取出来. 最后以比较直观的方式将代码信息完整地展现出来，为监控需求表达模块功能的实现提供支持.

(2) 监控需求表达模块. 在源代码分析模块的支持下，确定所需要监控的对象如函数变量等内容，并根据支撑工具模块提供的监控需求表达式模板帮助编程人员完成监控需求表达式的描述，将监控需求保存为 XML 需求文档，为监控代码的生成提供依据.

(3) 监控代码生成模块. 在监控需求表达模块以及软件体系结构的支持下，负责根据监控需求表达式的描述，调用监控代码生成工具和通用模块库中的监控代码模板，生成适合所选用的 AOP 编织器所需的侧面监控代码.

(4) 监控代码织入模块. 在监控代码织入模块的支持下，负责根据原系统的类型和待监控软件系统的类型，从支撑工具模块中选用合适的监控服务和监控客户. 并调用监控代码生成模块选用的 AOP 编织器，将侧面代码中的监控探针和监控服务一起织入到待监控的软件系统内部，使其具备可监控的能力.

(5) 支撑工具模块. 负责为其它模块功能的实现提供工具和模板支持. 如可以分别为源代码分析、监控需求表达、监控代码生成和监控代码织入模块提供代码分析工具、监控需求描述工具、监控代码生成工具和 AOP 编织器. 通用模板库不仅可以为描述监控需求和生成监控代码提供标准的格式模板，还可以为不同类型的目标系统提供监控服务和监控客户.

当注入了监控能力的软件系统运行时,可以通过织入的监控探针获取该软件系统的实时状态和行为,由监控服务对监控到的信息进行统一存储和管理,并通过监控客户端向用户展示系统的状态.

TSCM 在监控技术的基础上,为可信软件的构造提供了一种可行的方法.然而,目前该工作仍停留在软件开发的实现阶段,缺乏一套系统的软件体系结构设计方法以指导支持运行监控的可信软件的分析和设计.因此,本文采用面向侧面体系结构设计方法描述与分析具有监控系统能力的软件体系结构,形成一套面向侧面的可信软件体系结构设计方法,从而为生产可信软件提供可靠的保证,使得所生成的具有监控能力的软件系统,反过来又能为软件的可信性提供有力的证据.下面,本文将重点介绍该设计方法以及根据该方法定义的一种面向侧面的体系结构描述语言 AC2-ADL,并结合电子商务领域中的网上拍卖系统,说明该方法的实施过程.

3 面向侧面的可信软件体系结构设计方法

下面首先阐述面向侧面的软件体系结构设计方法的基本思想以及面向侧面的体系结构描述语言 AC2-ADL 的概念框架以及主要的语法元素,然后给出面向侧面的可信软件体系结构设计方法.

3.1 基本思想

根据关注点分离认为:区分、捕获和建模小的软件实体的功能比作用于整个应用程序要容易得多.可以把整个软件系统的设计作为一个较大的单个关注点,它由不同的小的关注点组合而成.每个小的关注点是不同的涉众根据自己的兴趣所关注的某一个方面.在这些关注点中,有些关注点只关注于自身的业务逻辑和功能,而另一些关注点,如非功能性关注点,往往趋向于分散和混杂在系统其它的关注点之中.这种横切的结构不但造成体系结构层中构件和连接件之间的紧耦合,而且还使得构件和连接件中的内容变得混杂.它们所代表的概念变得复杂,它们的边界也变得模糊起来.如何使用适当的体系结构元素描述和建模这些不同性质的关注点以及如何提供有效的组合方式将这些体系结构元素组装成最终的系统软件体系结构是本文研究体系结构设计方法的重点.

传统的体系结构设计方法缺乏相应的机制显式地描述这些横切的关注点以及各种关注点之间的组合关系.本文基于 AOSD 以及 EA(Earlier Aspect,早期侧面)的思想,在软件的体系结构层引入一种新的构件类型——侧面构件可对横切关注点功能进行建模,同时保留传统的构件概念以建模系统的功能性关注点(这里称之为系统的主关注点).通过抽象出编织机制中编织器的功能,将其作为体系结构中的一种新的连接件类型——侧面连接件,来呈现 SA 元素之间复杂的交互,以解决不同关注点的混杂和散列等问题.最后,通过显式地表达软件体系结构语境中的注入点,从而在体系结构的配置/拓扑层中表达各元素实例的组合关系.

尽管同时引入体系结构侧面构件、侧面连接件及其相关概念既会增加体系结构中的基本元素,也会增加元素之间关系的复杂性,但这样做可以极大地提高软件体系结构的设计效率并改进其质量.因此这样做是十分必要的,其理由是:侧面构件与传统构件类型的本质区别在于侧面构件的批量注入特性(quantification)和不知觉注入特性(obliviousness)[14],即侧面构件能以一种不知觉的方式影响系统中多种构件的结构和行为.而传统构件只能通过明确的接口定义,向外部发布/定制相应的服务.此外,抽象出侧面连接件的目的在于将注入规则从侧面构件规范中分离出来,使得侧面构件只关注于自身的横切功能而无需考虑注入的位置和注入的时机,有利于侧面构件的重用并为实现层的编织机制提供明确的线索,因而不会给构造和理解体系结构设计方案增添负担.图 2 展现了该方法的基本过程.

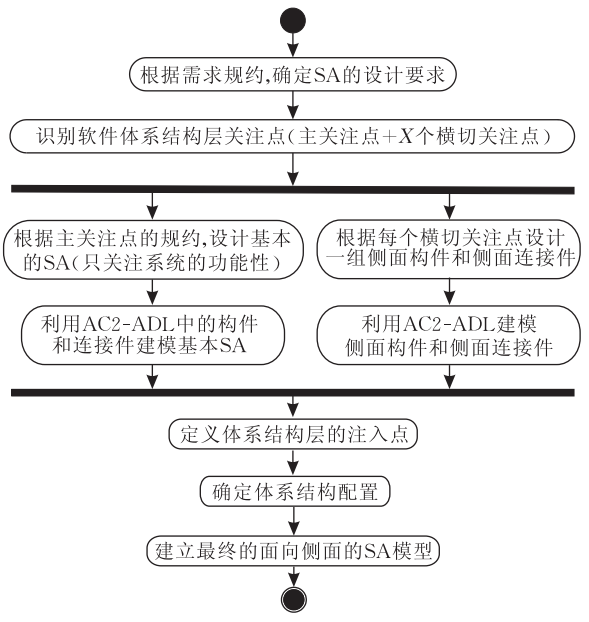


图 2 面向侧面的软件体系结构设计方法

3.2　面向侧面的体系结构描述语言

体系结构描述语言为体系结构的设计提供一种规范化的描述，是基于体系结构软件开发的基础. 为了刻画面向侧面的体系结构，AC2-ADL 除了传统的 ADL 中所包含的概念元素以外，还引入了侧面构件(aspectual component)和侧面连接件(aspectual connector)作为体系结构层的第一类元素. 然而，一个好的 ADL(Architecture Description Language)不仅要有较好的抽象表达能力，还应能够在体系结构设计阶段对有关性质进行分析和推理. 因此，AC2-ADL 结合一种时序逻辑语言 XYZ/E[15] 对各种构件的行为进行逐步求精，并结合面向侧面的软件开发中特有的性质和概念，扩展了 XYZ/E 中类其中主要元素的相关语法.

侧面构件. 不同于传统构件，侧面构件作为一种新的元素，在属性和功能上有了较大的改变. 为了体现自身的横切特征，侧面构件定义一种特殊的接口称其为横切接口(crosscutting interfaces)，它用于规范侧面构件能够提供的计算委托及其用途上的约束. 通过不同的横切接口，侧面构件可以对不同的构件进行行为和结构上的影响. 在 AC2-ADL 中，侧面构件的基本语法由图 3 以扩展 BNF 形式给出，其中加粗的符号为终结符.

```
aspectComponent ::= %aspectComponent component_name == [
    [SharedItemDeclPart;]
    [methodDeclPart;]
    [crosscuttingInterfaces;]
    [%subArchtecture subArchtecture_name == [
     architetureDeclPart | null;]]
    [%mappingDecl == [[mapping_expression | null;]]]
    [%internalProcess == □[conditionalelement_expression;]]
    [%constrains == [WherePart | null;]]]
```

图 3　侧面构件的基本语法规范

根据该语法，下面给出侧面构件的一些设计规则：

(1) 侧面构件由横切接口、侧面构件内部行为的描述、数据类型定义和方法以及相关约束等元素组成.

(2) 构件内部和方法中的行为以及约束条件都由 XYZ/E 中的条件元表达式(conditional element expression)描述.

(3) 根据系统的复杂性，侧面构件可由多个不同的功能单一的侧面构件复合而成，因此可以含有相应的子体系结构. 通过映射声明(mapping declaration)，外部的横切接口可以委托子体系结构中元素提供的相应横切服务.

(4) AC2-ADL 的构件以及侧面构件的接口由一组端口(port)和操作(operation)组成，操作定义了构件可以提供或定制所需要的服务，可由构件中定义的方法和条件元表达式组合来完成，端口是操作和外部环境的交互点.

图 4 描述了 AC2-ADL 中 3 种接口类型的定义. 其中横切接口中操作被分为两种不同的类型，它们分别被表示为 add_Operation 和 introduce_Operation. 其语义可解释为：通过 add_Operation 操作，横切接口可以把该操作中的方法添加到目标构建中，以增强目标构件行为；而 introduce_Operation 操作体现了侧面构件的静态横切的特性，即侧面构件可以通过该操作为目标构建引入相关的属性、方法、甚至接口，从而影响目标构建的结构. 这种面向侧面所特有的特征使得软件架构师能用一种真正面向对象的方法有效地建立复杂系统的模型，以一种本质上更优雅、更逼真于现实结构的方式，注入跨越整个系统的公共行为.

```
crosscuttingInterfaces ::= %crosscuttingInterface name == [
   [%port port_name:PortType]
   [%add_Operation Operation_name == □[
         conditionalelement_expression;]]]
requiredinterfaces ::= %requiredInterface name == [
   [%port port_name:PortType]
   [%Operation Operation_name == □[
              conditionalelement_expression;]]]
providedInterfaces ::= %providedInterface name == [
   [%port port_name:PortType]
   [%Operation Operation_name == □[
              conditionalelement_expression;]]]
method ::= %method name == □[conditionalelement_expression;]
```

图 4　AC2-ADL 中接口及方法的语法规范

侧面连接件. 随着侧面构件的出现，如何表达侧面构件对软件体系结构中元素的影响是面向侧面的体系结构设计中一个重要的需要解决的问题. 通过定义侧面连接件、建模侧面构件与基本构件间的横切关系，进而可以在体系结构层中显示地表达出面向侧面技术中编织器的功能.

图 5 提供了侧面连接件的主要语法，其相应的主要设计规则如下：

(1) 与构件相似，侧面连接件也由一组接口组成，这些接口通常被称之为角色.

(2) 侧面连接件包括两种角色，分别为注入点角色(pointcutRole)和横切角色(crosscuttingRole). 它们定义了该连接件所表示交互的参与者，如注入点角色对应一系列被横切的对象，而横切角色的扮演者往往是侧面构件的横切接口.

(3)为了保证体系结构中的构件连接以及它们之间通信的正确,并能够推导出所期待的连接服务,每个角色由一组端口和行为(behavior)组成了该角色的扮演者需要参与的活动.在实际的连接中,角色中的每个行为可由接口中的操作所对应的方法来扮演.

(4)通过连接协议(crosscutting protocol),连接件可以定义角色中行为之间交互的规范.在 AC2-ADL 中分别有 4 种横切交互协议类型,它们分别为 before、after、around 以及 replace.

```
aspectConnector::=%aspectConnector name==[
    %pointcutRole name==[
    [%port port_name:PortType]
    [%behaviors name;] ]
    %crosscuttingRole name==[
    [%port port_name:PortType]
    [%behaviors name;]
    [%constrains==[[WherePart|null;] ]
  %crosscuttingProtocols==[[crosscuttingProtocol_expressions;]]
   ]
crosscuttingProtocol_expressions::=
        crosscuttingRole_name.behavior_name
        crosscuttingProtocol_types
        pointcutRole_name.behavior_name
crosscuttingProtocol_types::=before|after|around|replace
```

图 5 侧面链接件的主要语法规范

体系结构层中的注入点.在面向侧面的实现阶段,注入点(joinpoint)是程序执行中的一组精确执行点,例如类中的方法调用.而在体系结构层,对注入点的描述将会比在实现层中的描述更加抽象,并且所囊括的概念也更加广泛.它可以是接口中的操作,也可以是构件的接口,甚至是体系结构中的某个连接件或构件,这也意味着侧面不仅可以影响构件还能影响构件间的交互.因此,AC2-ADL 明确地定义了注入点的类型,此外为了体现横切接口的批量注入机制,该语言定义了一组切点指示(Pointcut Designer,PCD).下面给出 PCD 的定义:

(1)PCD 可分为原子 PCD 和复合 PCD,前者是不含任何逻辑操作符的 PCD,后者是用"and"、"not"将其它 PCD 连接起来所形成的 PCD.

(2)PCD 有 ComPCD、ConPCD 和 and(ComPCD,ConPCD)三种形式.ComPCD 由构件级(componentJP)、接口级(interfaceJP)、操作级(operationJP)三种不同抽象层次及由它们所形成的复合 PCD(and(ComPCD,ComPCD))所组成.

(3)ConPCD 由连接件级(connectiorJP)、角色级(roleJP)、行为级(behaviorJP)三种不同抽象层次及由它们所形成的复合 PCD(and(ComPCD,ComPCD))所组成.and(ComPCD,ConPCD)表示一种既横切构件又横切连接件的混合 PCD.

图 6 展示了 PCD 基本语法,其中符号 comInstName、intfName、optName 分别表示构件实例名、接口名、操作名;conInstName、roleName、behaName 分别表示连接件实例名、角色名、行为名;保留字 PROINTF、REQINTF、PROROLE、REQROLE 分别表示提供类型接口、请求类型接口、提供类型角色、请求类型角色.通过引入逻辑操作符、保留字及通配符"*"来方便、精确地描述扮演侧面连接件的同一基本角色.例如:PCD:and(comInstName.PROINTF,*.REQROLE)表示名为 comInstName 的构件实例所有提供接口以及所有连接件实例的请求角色都为 SA 层的注入点,在这些位置上都要受到横切影响.

```
PCD::=ComPCD|ConPCD|and(ComPCD,ConPCD)
ComPCD::=operationJP|interfaceJP|componentJP|
            and (ComPCD,ComPCD)
operationJP::=comInstName.intfName.optName|not(operationJP)
interfaceJP::=comInstName.intfName|comInstName.PROINTF|
            comInstName.REQINTF|*.PROINTF|*.REQINTF|
            not(InterfaceJP)
componentJP::=comInstName|*|not(componentJP)
ConPCD::=behaviorJP|roleJP|connectorJP|and(ConPCD,ConPCD)
behaviorJP::=conInstName.roleName.beharName|not(behaviorJP)
roleJP::=conInstName.roleName|conName.PROROLE|
            conInstName.REQROLE|*.PROROLE|*.REQROLE|
            not(roleJP)
connectorJP::=conInstName|*|not(connectorJP)
```

图 6 切点指示器语法规范

面向侧面的体系结构配置.体系结构配置描述了运行时刻各种构件和连接件之间的拓扑结构.如图 7 所示,AC2-ADL 在配置(configuration)中指明:

(1)系统中的哪些注入点(通过保留字 playPCD)扮演了侧面连接件中的哪些注入点角色;

(2)哪些横切接口(通过保留字 play)扮演了侧面连接件中的哪些横切角色;

(3)系统元素中的操作与哪些角色中的哪些行为绑定关系(通过保留字 attachment)来完整地描述系统的拓扑结构.

```
Configuration::=
   %configuration==[[ConfigurationBody;]]
ConfigurationBody::=pcd playPCD
   aspectConr_Instance_name.baseRole_name|
   aspectCompt_Instance_name.crosscutting_Interfaces_name
   plays aspectConr_Instance_name. crosscuttingRole_name
   [%attachment==[attachment_Expressions;]]
attachment_Expressions::=method_name attaches behavior_name
```

图 7 体系结构配置的语法规范

此外，在设计系统的配置时，也需要遵循下面的规则：

(1) 类型的一致性，即各种构件的接口与其相连的连接件的接口的类型必须一致；

(2) 行为的一致性，即各构件接口中的行为应该与其相连的连接件角色的行为保持一致

最后，图 8 给出了面向侧面的体系结构描述规范，包括其定义的各种构件以及这些构件的实例和实例之间的配置等元素. 综合上述，本文认为将这种面向侧面的软件体系结构设计方法引入到支持运行监控的可信软件体系结构的设计之中，为指导支持运行监控的可信软件的分析和设计是行之有效的. 下面我们给出了一种基于侧面的，支持运行监控的可信软件体系结构设计方法.

```
Architecture::=%architecture architecture_name==[
    [component_list]
    [aspectComponent_list]
    [connector_list]
    [aspectConnector_list]…
    %component_instances==[component_instance_list]
    %aspectCompt_instances==[aspectComponent_instance_list]
    %connector_instances==[connector_instance_list]
    aspectContr_instances==[aspectConnector_instance_list]
    [%architectural_Configuration ==[Configuration]]
]
component_list::=[component;]
component_instance_list::=[instance_name:component_name;]
aspectComponent_list::=[aspectComponent;]
aspectComponent_instance_list::=
[aspectComponentInstance_name:aspectComponent_name;]
connector_list::=[connector;]
aspectConnector_list::=[aspectConnector_name;] …
```

图 8　体系结构语法规范

3.3　基于侧面的、支持运行监控的可信软件体系结构设计方法

监控在可信方面的目的更多地集中在验证程序的安全性、正确性、可靠性以及软件质量和信任评估上. 为了监控软件系统，首先应该清楚地知道需要针对系统的哪些与可信相关的关注点进行监控，然后根据这些关注点确定系统的监控目标，最后归纳出系统中需要监控的对象. 下面对该过程进行详细的阐述.

该方法归纳了近 70 个关注点，分离关注点的方法及过程可以参照我们用于体系结构设计的关注点多维分离模型[16]的相关工作. 通过研究软件需求工程及其相关研究[17-18]，以分组的方式将关注点之间关系较为紧密的归为一个维度，目前共归纳出 13 个维度. 在此基础之上建立通用多维模型以及管理这些关注点之间的若干维度. 该模型可为体系结构设计人员建立特定软件体系结构基本的体系结构模型提供指导.

根据该方法，首先可以建立与可信相关的通用关注点集合. 沿着与软件可信性质相关维度可分析出以下 4 个的主要横切关注点：① 可靠性(reliability)；② 可靠安全性(safety)；③ 保密安全性(security)；④ 生存性(survivability). 这些横切关注点被建模为侧面构件. 每个侧面构件封装了与之相关的监控目标和监控方法. 通过相应的横切接口以及其中的操作，侧面构件可以根据与之相连的侧面连接件将相应的监控方法编织到目标构件中去，从而完成自身的监控任务. 根据这些关注点，TSCM 模型能够确定所需完成的监控任务，并把它们分为以下几个主要部分：

(1) 系统优化. 系统的性能和资源在大多数情况下会相互冲突. 通过监控搜集到原始数据并生成有关状态报告，从而显示出对于当前系统性能状况以及资源消耗情况的综合性描述，再通过深入的分析，可以及时准确地发现系统的性能瓶颈或资源缺乏等问题.

(2) 系统可靠性. 软件可靠性不仅与软件中存在的缺陷有关，而且与系统输入和系统的使用有关. 通过监控的手段，对于软件系统中发生的各种由以上原因引起的异常事件进行监视，并且采用安全报告、实时处理等方式消除危害.

(3) 质量评估. 软件自身质量的评估是一个比较复杂，而且难以解决的问题. 通过在现有监控技术的基础上建立一个质量评估的平台，对于构件库中的各种实体(如构件和服务)用相对统一的方式来进行质量方面的评估，为每个实体建立相应的质量评估或者信任等级，最终为可信构件库建立的信任机制奠定一定的基础.

(4) 系统控制. 通过各种监控手段，能够掌握系统的具体状态. 掌握特定模块(线程、接口等)的当前状态，从而能够把握整个系统的历史和实时状态，并进一步决策出对系统的控制操作. 尤其是在系统出现某些错误或者性能降低等事件的时候，监控机制应该能够通过系统运行参数的调节而达到系统控制、管理的目的.

总之，可信的概念赋予了监控技术更多的内涵和思想，也极大地增大了监控的应用领域. 随着监控技术的发展和不断完善，TSCM 模型在可信软件领域也将发挥越来越大的作用.

然而抽象的监控任务难以自动地映射到软件开

发后续阶段.因此为适应开发可信软件的需要并提供具体的监控需求表达机制以及尽量保证系统具有较好的通用性和可扩展性,本文在以前的工作基础上[19-20]总结出4种主要的监控类型:进程监控类型(process)、线程监控类型(thread)、方法监控类型(method)和域监控类型(field).在实现层,每一种监控类型与一组实际的监控对象相对应.例如域监控类型对应着应用程序中一组实际的变量字段.根据这四种主要的监控类型,设计者可以针对目标系统的进程、线程、函数和关键变量进行监控,并对监控到的关键信息记录日志.在软件的体系结构层,不同的监控类型定义了对应的监控方法,如监控进程、监控线程、监控方法调用和监控关键域字段等方法.每个监控方法将抽象的监控任务转化为具体的监控目标和相应的监控属性.图9列出了主要的监控目标与相应的监控属性.

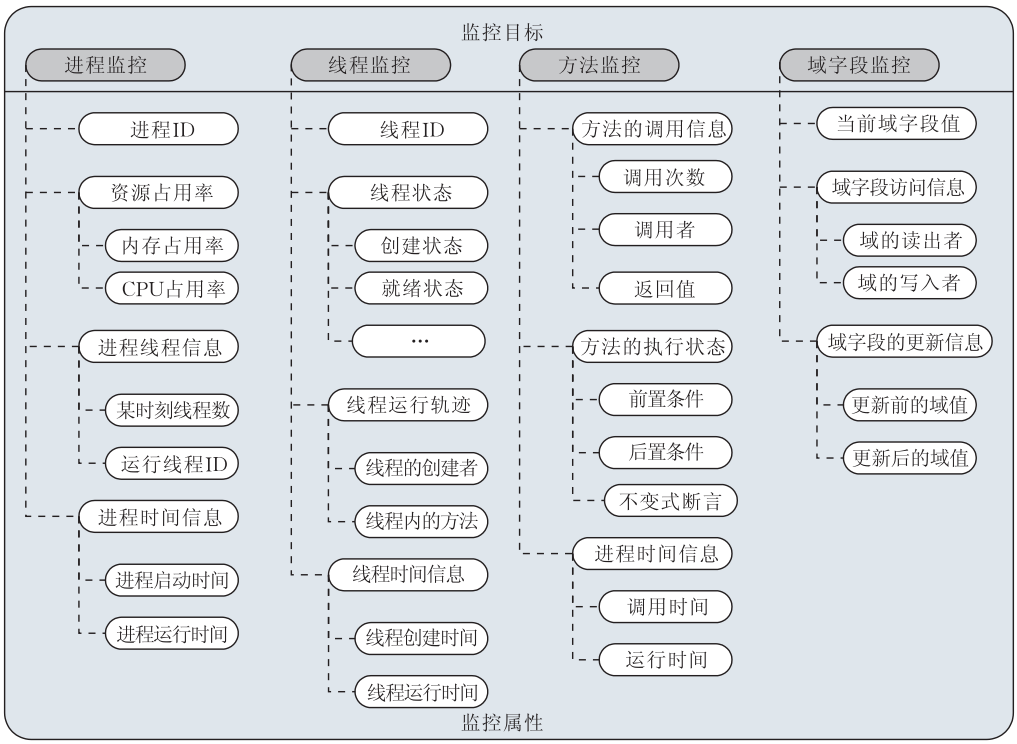


图9 监控目标结构图

根据可信软件体系结构设计的需要,设计人员可以为每个与软件可信性监控相关的侧面添加这四类方法,以获得系统运行的相关信息,如当前进程的名称、进程ID、进程开始执行时间、进程执行时间、运行线程信息、执行函数信息、线程数量等等.根据这些信息,监控侧面构件可以及时做出相应的判断,采用恰当的策略完成自身的监控目标或任务,从而提高系统的可信性.需要注意的是,本文所归纳的监控目标和监控属性是监控范畴的一个基本集合,并不能涵盖所有的监控需求.因此,针对不同的需求分析和设计规范,设计人员可根据3.3节提出的设计方法,为不同的侧面构件添加额外的监控目标、监控属性以及相应的监控方法.下面结合基于侧面的、支持运行监控的可信软件体系结构设计方法,讨论电子商务领域中网上拍卖系统支持运行监控的可信软件体系结构设计过程,为该方法的应用提供一定参考.

4 案例应用

案例研究的是一个基于英式风格的网上拍卖系统:系统为每个买家、卖家设立虚拟账户,提供银行帐号资金和虚拟账户钱币之间的划拨.游客可以注册成为买家或卖家.卖家拍卖自己的物品,买家参与物品的竞拍.针对每一轮的拍卖,系统对拍卖过程中买家的叫价进行仲裁.一旦拍卖结束,系统计算最高竞标价,然后将扣除了最高价中拍卖系统委托服务费之后的部分,存入卖家的信用卡中,并从买家的虚拟账户上扣除成交价与佣金的总和.

根据本文提出的方法,首先应识别出相应的与软件可信性相关的横切关注点,并且针对每个横切关注点分析其监控的目的,然后分别描述系统中各种体系结构元素,最后给出系统的软件体系结构.

4.1　识别与软件可信性相关的横切关注点

通过分析网上拍卖系统软件的需求描述，可以确定系统是一个大型的分布式的软件系统，来自Internet上的不同位置，有着不同需要的用户，都可以进入该系统. 此外，就拍卖领域本身的要求而言，系统也要保证公平性和实时性. 因此参考电子商务领域的相关工作[21]以及前面所确定的与可信相关的通用关注点集合，建立一个具体的网上拍卖系统的多维关注点模型. 由于篇幅的限制，本文的重点主要集中在体系结构的设计层与可信性相关的部分，详细的多维关注点模型方法可参看文献[16]. 下面仅给出了网上拍卖系统中，与可信性相关的主要横切关注点以及相关描述：

(1) 安全性监控关注点. 由于互联网上的买家和卖家是相互透明的，安全性监控关注点是网上拍卖系统的一个重要的关注点. 它既要验证用户的合法性，也要确保在Internet上数据的安全性.

(2) 性能监控关注点. 主要用来协调网上拍卖系统中各种资源和性能之间的冲突. 如以系统的数据库为例，性能调整主要包括内存的优化与调整、数据库Cache的调整、SQL语句的优化与调整以及数据库动态参数的调整等. 通过对系统整体的全局监控并进行性能调整，可使数据库达到最佳性能以满足用户的需要.

(3) 可靠性监控关注点. 该关注点是本系统的重点，涉及到与其相关的很多方面，如通过同步对象的状态，确保拍卖系统从客户端接收的数据的一致性；针对各种异常情况，如通信、数据存取、输入输出等问题采取相应的处理措施等等.

(4) 并发和并行性监控关注点. 用户可能同时参与多个商品的拍卖，也可同时对同一个商品投标. 在这种多线程并行环境中，由于资源竞争以及调度等原因，并发程序的各个并行执行语句之间的相对执行次序是不确定的，导致不同的执行结果. 通过监控能够及时发现运行时的错误.

当然，不同的需求，其关注点也是的不同的. 设计人员可以根据具体的需求，确定系统所需要监控的关注点. 针对案例中与可信相关的横切关注点，本文利用AC2-ADL描述网上拍卖系统的软件体系结构，并给出该体系结构的设计过程.

4.2　网上拍卖系统的软件体系结构

根据第3节给出的面向侧面的软件体系结构设计方法，可以得出如图10所示的面向侧面的网上拍卖系统的软件体系结构.

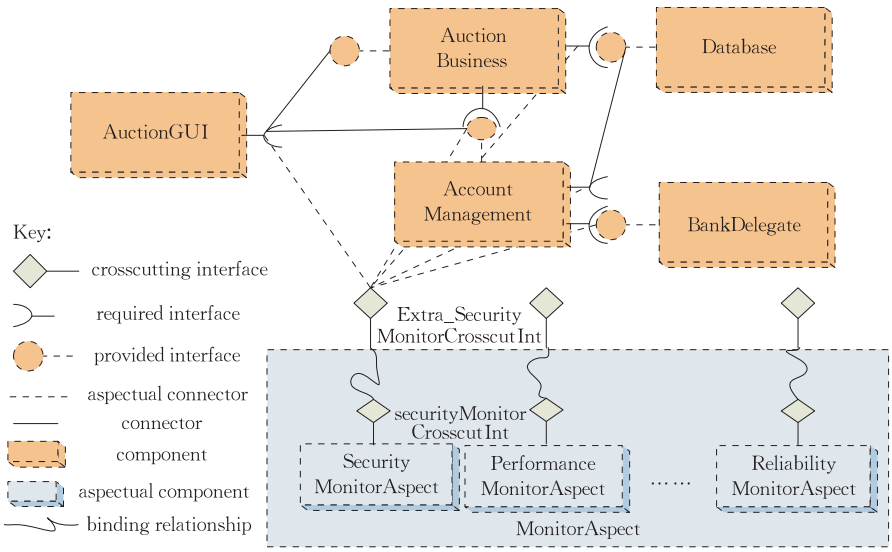


图10　支持运行监控的网上拍卖系统软件体系结构

该系统的核心功能构件分别由提供网上拍卖系统的用户图形界面接口构件AuctionGUI、封装了拍卖业务逻辑功能的构件AuctionBusiness和账户管理构件Account Management以及数据库构件Database和银行提供的代理构件BankDelegate所组成. 而侧面构件MonitorAspect封装了需要对系统注入监控能力的相关横切功能. 为了能满足各种监控目标，该侧面构件由多个监控侧面构件复合而成，其中包括安全监控侧面SecurityMonitorAspect、性能监控侧面PerformanceMonitorAspect、可靠性监控侧面ReliabilityMonitorAspect等. 每个侧面构件都有相应的横切接口. 如图10所示，安全监控侧面SecurityMonitorAspect提供了安全监控横切接口SecurityMonitorCrosscutInt对系统某些敏感点注

入相应的监控探针以监控系统的安全.这些内部模块的接口可以通过与复合构件外部的接口进行相应的绑定,从而和该复合构件以外的元素交互.由于篇幅的限制下面只呈现了由 AC2-ADL 描述的部分体系结构元素,并以安全监控侧面 SecurityMonitorAspect 作为案例,来说明面向侧面的体系结构设计方法在开发过程中的应用.详细的描述可参看相关文献[22].

首先需要用 AC2-ADL 建模体系结构中的各种元素如构件、侧面构件等;然后将它们按照一定的规范组合起来如体系结构的配置,从而得到最终的具有监控能力的网上拍卖系统的软件体系结构蓝图.图 11 给出 SecurityMonitorAspect 的相关描述.这里需要对时序逻辑语言 XYZ/E 做一个简要的介绍,XYZ/E 中的表达式有条件元组成,其表现形式主要有两种:

$$LB=y\wedge R\Rightarrow \$O v=e\wedge \$OLB=z \quad (1)$$

$$LB=y\wedge R\Rightarrow @(Q\wedge \$OLB=z) \quad (2)$$

形式(1)所示的条件元直接定义了程序相邻状态之间的转换关系,形式(2)所示条件元表示程序抽象规范,其中符号@可以是下一时刻算子 $O 或最终时刻算子◇;两种形式的条件元都是时序逻辑公式,其语义即为此时序逻辑式的语义模型.此外,XYZ/E 也提供输入命令和输出命令:

$$LB=y\wedge R\Rightarrow ChNm?x\wedge \$OLB=z,$$

$$LB=y\wedge R\Rightarrow ChNm!e\wedge \$OLB=z,$$

其中"ChNm"是通道名,x 是变量,e 是表达式."ChNm?x"表示通过通道 ChNm 将数据送入变量 x;"ChNm!e"表示将 e 的值经过通道 ChNm 送出.从图 11 可以看到 SecurityMonitorAspect 主要包含的方法,由于篇幅的限制,这里结合 XYZ/E 中条件元表达式主要描述了线程监控方法 threadMonitoring() 和权限验证方法 verification() 部分行为.其中 threadMonitoring() 能获得线程当前的状态,并为新运行的线程的状态信息的存储分配一定内存空间,以便后续的工具可以对其进行管理或分析.而 verification() 是系统安全性必不可少的一个组成部分,该方法为新开启的线程赋予相应的权限,并通过 check() 方法检查当前运行的线程是否在其相应的权限之内,以确保系统的安全性.

图 11 也给出该构件体内部分处理过程,通过时序逻辑语言中通道以及消息的输入/输出命令,构件中的方法按照相应的交互协议有续的联系构成构件的总体行为.

```
%aspectComponent SecurityMonitorAspect==[
TYPE THREADSTATE:RECORD[name: STRING;ID:INT;…]
  THREADSTATE list:RECORD[threadState:THREADSTATE ;
                  next:POINT(TSTATEList);…] …
%Methods threadMonitoring(%IOP TState:THREADSTATE )==
  □[
  %LOC[i:THREADSTATE];
  LB=Start⇒currState=getThreadState()∧$OLB=L1;
  LB=L1∧(currThread==NewThread)⇒
      i=currThread∧$OLB=L2;
  LB=L1∧~(currThread==NewThread)⇒$OLB=L3;
  LB=L2⇒threadStateList.add(i)∧$OLB=L3;
  LB=L3⇒TState=currState∧$OLB=EXIT;];
%methods verification (%INP threadState:THREADSTATE;
  %OUTP verificationResult:BOOL)==□[
  LB=Start⇒currID=getID(threadState)∧$OLB=L1;
  LB=L1∧(currID==NewThreadID)
    ⇒setPermission (currID)∧$OLB=L2;
  LB=L1∧~(currSID==NewThreadID)⇒$OLB=L2;
  LB=L2⇒perm=getPermission(currID)∧$OLB=L3;
  LB=L3⇒result=check(perm)∧$OLB=EXIT;];
%methods processMonitoring==□[…];
%methods functionMonitoring==□[…];
%methods variableMonitoring==□[…];…
%CrosscuttingInterface SecurityMonitorCrosscutInt==□[
    %port tStatePort;THREADSTATE;
    %port verificationPort;THREADSTATE;
    %port resultPort:BOOL; …
    %add_Operation threadMonitoring==□[
      LB=Start⇒tStatePort?TState$OLB=L1;
      LB=L1∧~(TState==null)⇒
          threadMonitoring()∧$OLB=L1;
      LB=L2∧(TState==null)⇒$OLB=EXIT];
    processMonitoring==□[…];
    fieldMonitoring==□[…];
    methodMonitoring==□[…]; …]
%internalProcess==□[
    ||[LB=Start⇒tStatePort?TState∧$OLB=L11;
    LB=L11∧~(TState==null)⇒
        threadMonitoring()∧$OLB=L12;
    LB=L12⇒tStatePort?TState$OLB=∧L11;
    LB=L11∧(TState==null)⇒$OLB=EXIT];
    [LB=L21⇒verificationPort!threadState∧$OLB=L22;
     LB=L22⇒verification()∧$OLB=L23;
     LB=L23⇒resultPort!verificationResult∧$OLB=L21]…]
  %constrains==□[TState==threadState;…]…
  ]
```

图 11 SecurityMonitorAspect 的 AC2-ADL 描述

对于侧面连接件的描述,图 12 给出了安全监控侧面连接件(SecurityAspectConnector)的模型.如图 12 所示,该连接件注入点角色包括用户登录角色(LoginRole)、消息传输角色(TransmissionRole)等,而横切角色有安全监控角色(SecurityMonitorRole)等,其中它们分别含有各自的一系列行为,如 SecurityMonitorRole 可能具有监控变量、监控函数、监控线程等行为.在横切协议(crosscuttingProtocol)中 SecurityMonitorRole 的监控函数和监控变量的行为必须在用户登录行为发生之前先发生(由保留字 before 说明),从而可以在用户登陆时对用

户的登录和输入的数据进行监控；另一方面，监控线程行为可能需要对系统中的各种线程（如买家或买家线程等）中的各种接收或发送行为（如发出叫价消息或接收商品当前信息等）进行监控.

```
aspectConnector SecurityAspectConnector==[
%pointcutRole LoginRole==[
        %port loginPort:THREADSTATE;…
        %behaviors logging;…]
 %pointcutRole TransmissionRole==[
        %port transmissionPort:Data;…
        %behaviors sendMassage, receiveMassage;…]…
%crosscuttingRole SecurityMonitorRole==[
        …
        %behaviors monitorVar, monitorFun,…;]
%crosscuttingProtocol==[
  (SecurityMonitorRole.monitorFun and monitorVar)
  before Login.logging;
  SecurityMonitorRole.monitorThread before
  (Transmission .sendMassage and receiveMassage);…]]
```

图 12　SecurityAspectConnector 的 AC2-ADL 描述

在侧面构件以及相应的侧面连接件设计完成之后，随之而来的问题就是如何合理地组合这些体系结构元素. 根据面向侧面的软件体系结构设计方法，首先需要确定软件体系结构层的注入点，并定义注入点表达式 PCD. 然后根据这些 PCD 定位与之相应的侧面构件，并用适合的侧面连接件将侧面构件与 PCD 所指定的构件组合起来，形成体系结构的配置. 换句话说，这些 PCD 是侧面连接件中注入点角色的扮演者，而侧面构件以及它们的横切接口，则是相关横切角色的扮演者. 下面给出一个定义 PCD 的例子：由于在基于 internet 的网上拍卖系统中，各个构件之间的交互往往与安全性密切相关，每个请求接口中的方法都应该是被监控的对象，当它们向外界请求服务的过程中，它们的行为、状态等应该受到监控. 因此一个与安全监控侧面有关的 PCD 可以定义为如下形式：

```
AuctionGUI_instances.  AllrequestInterface.* and
AuctionBusiness_instances. AllrequestInterface.* and
AccountManagement_instances. AllrequestInterface.*
```

最后，在确定了体系结构层的注入点、安全侧面构件以及安全侧面连接件后，一段与安全监控相关的配置可描述为如图 13 所示.

其中复合侧面构建 MonitorAspect 的外部横切接口 extra_SecurityMonitorCrosscutInt，通过将其横切功能委托给内部的侧面构建 SecurityMonitorAspect 的实例 securityMAspect_instances 中的横切接口来完成自身的任务（由关键字 mappingDeclaration 给出），另一方面在配置中，接口中的方法和角色中

```
%architecture OAS==[
%component Auction_GUI==[…];
%component Auction_Business==[…]; …
%aspectComponent SecurityMonitorAspect==[…];
%aspectComponent MonitorAspect==[…
  %crosscuttingInterface extra_SecurityMonitorCrosscutInt; …
  %subArchtecture securityMonitor_SubStructure==[…
    %aspectComponent_instances
      securityMAspect _instances:SecurityMonitorAspect;…]
  %mappingDeclaration
 SecurityMonitorCrosscutInt binds extra_SecurityMonitorCrosscutInt;
 …]…
%aspectConnector SecurityAspectConnector==[…]; …
%component_instances
    AuctionGUI_instances:AuctionGUI;
%aspectComponent_instances
    monitorAspect _instances:MonitorAspect; …
%aspectConnector_instances
    securityAConr_instances: SecurityAspectConnector; …
%configuration==[
    AuctionGUI_instances.AllrequestInterface.* and
    AuctionBusiness_instances.AllrequestInterface.* and
    AccountManagement_instances.AllrequestInterface.*
  playPCD securityAConr_instances.TransmissionRole;
    monitorAspect_instances. extra_SecurityMonitorCrosscutInt
  plays securityAConr_instances. SecurityMonitorRole;
  attachment threadMonitoring attaches monitorThread;
             variableMonitoring attaches monitorVar;…]]
```

图 13　与安全监控相关的体系结构描述

的行为的粘合可在 attachment 段由关键字 attaches 进一步细化，以确保构建中的方法能够按照连接件中规定的行为进行交互. 按照上面的方式，一个完整网上拍卖系统的软件体系结构可以逐步地设计出来，它将指导软件后续阶段的开发.

4.3　基于软件体系结构的可信软件开发

设计良好的体系结构设计文档既为后续阶段的软件开发提供有力的线索和保证，也为最终软件在运行时的监控结果的分析和验证提供相关的依据.

基于支持运行监控的可信软件构造模型 TSCM，开发人员根据体系结构设计文档，在网上拍卖系统的源码中确定所需要监控的对象如方法、域字段等内容. 这里系统采用 Java 语言的实现方式. 根据 TSCM 支撑工具模块提供的监控需求表达式模板，可以采用 Java5.0 的注释机制（annotation）的描述具体的监控需求表达式. 该模板包括四类监控 annotation：@monitor_process，@monitor_thread，@monitor_method 和 @monitor_field. 并通过 annotation 的参数来对监控目标或属性的监控点进行选择. 当参数的真值为 TRUE 时，表示要对该监控点进行监控，为 FALSE 时则表示不进行监控. 如进程监控 annotation 可表示为：@monitor_process (Boolean Resources, Boolean Thread, Boolean Time, String Description)，其中 Resources 表示进程占用

资源的监控，Thread 表示进程内运行线程信息的监控，Time 表示进程运行时间监控，Description 参数提供给开发人员来对要监控的进程进行描述. 对进程 ID 的监控是默认进行的，Resources，Thread 和 Time 的缺省值为 FALSE. 如在本例中，针对构件 AuctionBusiness 中的用户登录，需监控系统的访问者，登录方法的调用信息等，可以用来分析用户权限等与安全性相关的可信属性. 图 14 给出部分监控注释应用代码示例.

```
/ * import code segment * /
public class userLogin {
  @monitor_field(RealValue=TRUE,…,UpdateInfor=TRUE,
  Description="This is a key factor of the system")
  public static string userName; …(Java code)…
  @monitor_method(CallInfor=TRUE, State=FALSE,
    Description="This is the method need to be verified")
    public void login(string userName,…) {…}
  …(Java code)…}
```

图 14 部分监控注释应用代码示例

在监控代码生成模块的支持下，监控需求表达式中的 annotation 被抽取出来，编译成与原系统兼容的侧面代码. 随后监控代码织入模块使用对应的 AOP 编织器进行编织，将侧面代码注入原网上拍卖系统，使其具备运行时可自我监控的能力. 图 15 给出监控需求表达式编译后生成的部分 AspectJ 代码框架.

```
public aspect SecurityAspect{
pointcut userCreateTime():within(User) &&execution(new(..));
pointcut userRun(): execution(void user.run());
pointcut userNameChange(): set(string userName);
pointcut methodCall(): call(void login(string userName,…));
pointcut process():execution(void Test.main(..)); …
before(): process(){
//start a thread to detect system resource usage and send information to the monitor service}
before(string userName,…):methodCall()&&args(userName,…){
//record the time while starting to call "login ()", get the parameter of it,
//acquire the thread context, and send it to the monitor service}
before():userCreateTime(){
//record the time when a Producer thread be created and send these
//information to the monitor service}
after():userRun(){
//record the time when a Producer thread stopped and send these
//information to the monitor service}
before(): userNameChange(){
//record the value of "user.name" before change,acquire the thread and
//function context and send these information to the monitor service}
after():userNameChange(){
//record the time when "user.name" be changed and record the value after //change, send these information to the monitor service}
……}
```

图 15 编译后生成的部分 AspectJ 代码框架

5 结束语

本文在支持运行监控的可信软件构造模型 TSCM 的基础上，引入了面向侧面的软件体系结构设计方法以及相关的面向侧面的软件设计语言 AC2-ADL，提出了一种面向侧面的可信软件设计方法；讨论了该方法的基本过程，并以网上拍卖系统为实例展示了该过程的主要步骤和实现层的结果. 与已有的工作相比，我们的工作支持基于软件体系结构的可信软件系统的分析、设计和实现，并且将面向侧面的思想及其相关技术应用于软件开发的各个阶段，对可信软件的开发是一种新的尝试.

然而，我们的研究工作仍存在着许多不足，控制作为监控的一个重要部分，仍然是下一步工作的重点. 拟解决的方案包括：进一步地完善 AC2-ADL，研究软件可信保障机制，通过结合命题投影时序逻辑（Propositional Projection Temporal Logic，PPTL）形式化方法[23-24]，分析和验证软件的可信性，以确保软件的健康运行.

参 考 文 献

[1] Li Ren-Jie. Research and implementation of the trusted software constitution based on monitoring [M. S. dissertation]. National University of Defense Technology, Changsha, 2007 (in Chinese)
(李仁杰. 基于监控的可信软件构造技术研究与实现[硕士学位论文]. 国防科学技术大学，长沙，2007)

[2] Chen Huo-Wang, Wang Ji, Dong Wei. High confidence software engineering technologies. Chinese Journal of Electronics, 2003, 31(12A): 1933-1938(in Chinese)
(陈火旺，王戟，董威. 高可信软件工程技术. 电子学报，2003，31(12A)：1933-1938)

[3] Guo Chang-Guo, Wang Tao. A method and framework for fetching softeware runtime state//Proceedings of the 2010 International Conference on Computer, In: Mechatronics, Control and Electronic Engineering (CMCE 2010). Changchun, 2010

[4] Musa John D, Anthony Iannino, Kazuhira Okumoto. Software Reliability Measurement, Prediction, Application. New York: McGraw-Hill Book Company, 1987(in Chinese)

[5] Wang Huai-Min, Tang Yang-Bin, Yin Gang, Li Lei. Trust mechansims of Internet software. Science in China, Series E, Information Sciences, 2006, 36(10): 1156-1169(in Chinese)
(王怀民，唐扬斌，尹刚，李磊. 互联网软件的可信机理. 中国科学 E 辑，2006，36(10)：1156-1169)

[6] Garlan D, Shaw M. An introduction to software architecture//Ambriola V, Tortota G eds. Advances in Software Engineering and Knowledge Engineering, Volume 1. New Jersey: World Scientific Publishing, Co., 1993

[7] Kiczales G, Lamping J, Mendhekar A et al. Aspect-oriented programming//Proceedings of the 11th European Conference on Object-Oriented Programming (ECOOP 1997). Lecture Notes in Computer Science. Springer-Verlag, 1997: 220-242

[8] Araujo J, Baniassad E, Clements P, Moreira A, Tekinerdogan B. Early aspects: The current landscape. Technical Report: CMU/SEI-2005-TN-xxx, Lancaster University, 2005

[9] Garcia A, Kulesza U, Lucena C. Aspectizing multi-agent systems: From architecture to implementation//Proceedings of the Engineering for Multi-Agent Systems III, Research Issues and Practical Applications. Lecture Notes in Computer Science 3390. Springer-Verlag, 2005: 21-143

[10] Pinto M, Fuentes L, Troya J M. DAOP-ADL: An architecture description language for dynamic component and aspect-based development//Proceedings of the International Conference on GPCE. Erfurt, Germany, 2003: 118-137

[11] Batista T, Chavez C, Garcia A et al. Aspectual connectors: Supporting the seamless integration of aspects and ADLs//Proceedings of the ACM SIGSoft XX Brazilian Symposium on Software Engineering (SBES'06). Florianopolis, Brazil, 2006

[12] Mei Hong, Chen Feng, Feng Yao-Dong et al. ABC: An architecture based, component oriented approach to software development. Journal of Software, 2003, 14(4): 721-732(in Chinese)
(梅宏，陈锋，冯耀东等. ABC:基于体系结构、面向构件的软件开发方法. 软件学报，2003，14(4)：721-732)

[13] Liu Rui-Cheng, Zhang Li-Chen. Aspect-oriented modeling method based on UML. Computer Science, 2005, 32(1): 204-213(in Chinese)
(刘瑞成，张立臣. 基于 UML 的面向方面建模方法. 计算机科学，2005，32(1)：204-213)

[14] Filman R E, Friedman D P. Aspect-oriented programming is quantification and obliviousness//Proceedings of the Workshop on Advanced Separation of Concerns (OOPSLA 2000). 2000

[15] Tang Zhi-Song et al. Temporal Logic Programming and Software Engineering. Beijing: Science Press, 2002(in Chinese)
(唐稚松等. 时序逻辑程序设计与软件工程. 北京：科学出版社，2002)

[16] Zhang Lin-Lin, Ying Shi et al. Model for architectural multi-dimensions separating of concerns. Computer Science, 2009, 36(3): 266-269(in Chinese)
(张琳琳，应时等. 一种软件体系结构关注点多维分离模型. 计算机科学，2009，36(3)：266-269)

[17] Barbacci M, Klein M H, Longstaff T et al. Quality attributes. SEI, 1995

[18] Eeles P. Capturing architectural requirements. The Rational Edge, 2001

[19] Haban D, Wybranietz D. A hybrid monitor for behavior and performance analysis of distributed systems. IEEE Transactions on Software Engineering, 1990, 197-211

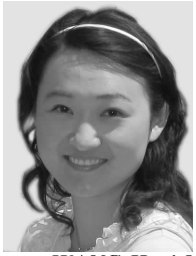
[20] Jeffery C, Zhou W, Templer K, Brazell M. A lightweight architecture for program execution monitoring. ACM Sigplan/Sigsoft, 1998

[21] Wurman P R. Online auction site management. In: The Internet Encyclo-pedia, 2003. http://www.csc.ncsu.edu/wurman/Wur-article.pdf

[22] Wen Jing, Ying Shi, Zhang Lin-Lin, Ni You-Cong. AC2-ADL: Architectural description of Aspect-Oriented System. International Journal of Software Engineering and Its Applications, 2009, 3(1): 1-10

[23] Tian Cong, Duan Zhen-Hua. Complexity of propositional projection temporal logic with star. Mathematical Structures in Computer Science, 2009, 19(1): 73-100

[24] Tian Cong, Duan Zhen-Hua. Model Checking Propositional Projection Temporal Logic Based on SPIN//Proceedings of the ICFEM2007. LNCS4789, Springer-Verlag, 2007: 246-265



WEN Jing, born in 1982, Ph. D. candidate. Her research interests include software engineering, aspect-oriented software development, human computer interaction.

WANG Huai-Min, born in 1962, professor, Ph. D. supervisor. His research interests include distributed computation, network and information security.

YING Shi, born in 1965, Ph. D., professor, Ph. D. supervisor. His research interests include software engineering, object-oriented software development, software architecture and pattern.

NI You-Cong, born in 1979, Ph. D. candidate. His research interests include software engineering, aspect-oriented software development.

WANG Tao, born in 1984, Ph. D. candidate. His research interests include distributed computation and object-oriented programming.

Background

Nowadays, the application of software is omnipresent. With the increasing requirement for functionality, complexity and usability of applications, the ever-growing scalable software systems are not always trustworthy. As a result, "Trustworthiness" has become a key characteristic for software quality research and practice. Despite without precise definition, the "Trustworthiness", featured as a gauge to evaluate and assess conformity of software behavior and out-

come with the expectation of users, has been supported and developed by the techniques of trustworthy software production, which emphasize mainly how to improve trustworthiness of software.

Trusted Software Constitution Model based on Monitoring (TSCM) is a method to improve software trustworthiness. In this method, monitoring mechanism was initiated to gather software operation status. The comparison between monitoring results and expected behavior could be used to evaluate the trustworthiness of software behavior, and will contribute to precisely analyzing and locating the malfunction of software. Consequently, injecting monitoring mechanism will be an important method to improve the trustworthiness of software. Nevertheless, software trustworthiness is a quality attribute not only through software production, but also cross the whole software running process. It is usually a global constraint in the whole software system and has to be designed step by step along with the software developing process. Unfortunately, most of recent researches focus on coding and implementing level of software development, lacking a systematic software architecture design method to support the analysis and design of trusted software with the capability of monitoring.

At present, the growing attention on Software Architecture (SA) has been paid by researchers. Through research on SA, a system SA and its composite elements, such as component, connector, could be explicitly expressed. Thus a series of design problems, such as global organization and control conditions, function assembly, computing unit deployment and relevant communication between them, could be handled in a higher abstract level. However, conventional SA design methods usually start from business processes and user cases to model the basic SA. As a result, implementation of requirement for trustworthiness is delayed to code in form of crosscutting into the basic SA units. These codes are always scattering and tangling with other modules, leading to high coupling and complexity. How to find a divide-and-handle method according to the nature of trustworthiness to control the complexity is the key to trusted software design.

Fortunately, the development and maturity of AOSD (Aspect-Oriented Software Development) and its related technology bring the new hope for the development of trusted software. This method takes fully advantage of the Separation of Concerns (SOC) theory. Through using Aspect and its related concept to encapsulate and manage crosscutting attribute spreading across all phases in the software developing cycle, it is helpful to improve software quality and also reduces the cost of adaptation, evolution and maintenance. Therefore, using Aspect to manage trustworthiness in the software architecture could further improve the comprehension, evolution and reusability of software architecture design.

In this paper, on the basis of TSCM, an Aspect-oriented SA design method and its related concept are introduced. An Aspect-oriented SA describing language AC2-ADL is used to describe software system with monitoring ability, aiming to provide an effective solution to the analysis and design of SA for software system with monitoring ability. A case study of auction system will demonstrate the procedure and result of the proposed method, and the paper is concluded with discussion and future work.