大型复杂软件系统安全需求的体系结构模型*)

谭 良[1,2] 周明天[2]
(四川师范大学四川省软件重点实验室 成都 610066)[1]
(电子科技大学计算机科学与工程学院 成都 610054)[2]

摘 要 在开发基于Internet的大型复杂软件系统时,应该在体系结构层次上考虑业务需求和安全需求,而传统的体系结构没有专门针对安全需求的构件、连接件和体系结构风格的描述,因此在体系结构层次上描述安全需求还比较困难。本文首先论述了在体系结构层次上描述安全需求的必要性。然后,在传统体系结构单元——部件/连接件的基础上,引入了安全构件、半安全构件、安全连接件、半安全连接件等新的设计单元,并给出了这些设计单元形式化的语义和约束以及图形模型,解决了软件系统安全需求的构件表示方式。最后,用一个实例展示了软件系统安全需求的体系结构模型。

关键词 体系结构,安全构件,半安全构件,安全连接件,半安全连接件,体系结构模型

Security Requirements Architecture Model for Large and Complex Software Systems

TAN Liang[1,2] ZHOU Ming-Tian[1]
(School of Comp. Sci. & Engn., Univ. of Electronic Sci. & Tech. of China, Chengdu 610054)[1]
(college of Electronic Engineering, Sichuan Normal University, Chengdu 610066)[2]

Abstract It is imperative to considerate the functional requirements and security requirements on architecture level when developing the large and complex software system in Internet. Because traditional architecture has no direct component, connector and style for security requirements, it is difficult to descript these security requirements on architecture level. In this paper, necessities for security requirements architecture model are discussed, and then, some new fundamental units based on the traditional software architecture, such as security component, security connector, half-security component and half-security connector, and so on, are presented, moreover, the formal semantic elements, constraints and graph for these new units are established, so the representation of security requirements by component-fashion is resolved. Finally, with an example to show the security requirements architecture model for large distributed software systems in Internet.

Keywords Software architecture, Security component, Security connector, Half-security component, Half-security connector, Architecture model

由于Internet天生的开放和动态特性导致在安全和可靠方面的欠缺,因此在开发基于Internet、大型复杂的软件系统时,除了考虑软件系统的业务需求外,必须考虑和确定软件系统的安全需求,开发过程中需要软件工程和信息系统安全工程共同作为指导。由于这类系统的复杂性,在系统设计时,应该考虑系统中的计算元素和它们之间交互的高层组织,在高抽象层次处理诸如全局组织和控制结构、功能到计算元素的分配、计算元素间的高层交互等设计问题。如果忽略了这一点,则在开发过程中将系统需求(包括业务功能需求和安全需求)转换为设计非常困难,业务功能与安全功能之间的配置与交互关系模糊不清,并且相似系统之间的复用程度很低。这类系统的开发周期长、代价高和质量低的问题仍然存在[1]。

近几年,软件体系结构SA(software architecture)已成为软件工程研究的热点之一。它作为软件的蓝图,为人们宏观把握软件的整体结构,实现需求到设计的平滑过渡,提高系统的复用粒度等提供了一条有效途径。经过10多年的研究,取得了一系列成果,出现了大量的体系结构描述语言,典型的有:C2[2]、ACME[3]、Unicon[4]、Rapide[5]、SADL[6]、Wright[7]、Darwin[8]、ABC/ADL[9]、FRADL[10]、A-ADL[11]、XYX/ADL[12]、Tracer[13]等。尽管这些语言描述的体系结构及其风格各有特点和侧重,但都认为SA是组成系统的构件以及构件与构件之间交互作用关系(连接件)的高层抽象。

到现在为止,很少有专门针对软件系统安全需求的构件、连接件和体系结构风格的描述。这样,在开放的Internet平台上开发大型复杂软件系统时,很难在体系结构层次上完整地描述这类系统的体系结构。本文首先论述了在体系结构层次上描述安全需求的必要性。然后,在传统体系结构单元——部件/连接件的基础上,引入了安全构件、半安全构件、安全连接件、半安全连接件等新的设计单元,并给出了这些设计单元形式化的语义和约束以及图形模型,解决了软件系统安全需求的构件表示方式。最后,用一个实例展示了软件系统安全需求的体系结构模型。

1 在体系结构层次上描述安全需求的必要性

在开放的Internet平台上开发大型复杂软件系统时,满足其安全需求的方法通常是针对安全需求单独建安全模型。

*)国家863宽带VPN项目863-104-03-01课题资助;2003年度四川省科技攻关项目03GG007-007支持。谭 良 博士,主要研究方向为信息安全、中间件;周明天 教授,博士生导师,主要研究方向为网络计算、信息安全、分布并行处理。

安全模型是指在一定的环境里，为保证提供一定级别的安全保护所奉行的基本思想，它表示安全服务和安全框架是如何结合的。目前，在建立安全模型方面已取得了诸多成果，如自主访问控制模型、强制访问控制模型、基于角色的访问控制、基于任务的访问控制、BLP 模型、Biba 模型、Lipner 模型和 Clark-Wilson 模型等；另外，也包括一些综合性模型，如中国长城模型、临床信息系统安全政策模型和创建者控制访问控制模型等。因此，在开发这类系统时，可以在某些情况下复用这些安全模型。用安全模型解决系统的安全需求最大好处在于可以采用形式化的方法进行验证。

但是，在怎样对待安全模型方面有两种不良倾向：一是重模型建立，轻代码实现。很久以来，很多学术文章都在讨论安全模型问题，包括建立模型的方法、模型的安全性、完备性以及形式化证明等。但是，安全模型要起作用，必须将安全模型转换为程序代码。一个好的理论模型，如果它不能用代码实现，或用代码来实现很困难，这样的模型也只能是理论摆设而已，对满足用户的实际安全需求帮助很小。如要将 BLP 模型和 Biba 模型同时在 OS 中实现，不得不对模型进行修改，否则 OS 会产生死锁[14]。另一种倾向是重模型复用，轻代码复用。实际上，要将一个安全模型用代码来实现，花费的人力物力并不比同等规模的一般功能要少。如在 Grsecurity 中[15]，对强制访问控制的实现，用了将近 1 万多行代码，按一个程序员平均 15 行/天来计算，那么 5 个程序员要将近半年的时间才能实现，这还不包括对模型的分析、理解和程序员之间沟通和个体差异。因此，模型的代码复用非常重要。

那么应该在什么层次上复用模型代码呢？如果在一般的模块（函数）层次上复用，则重用的级别比较低，重用的适用性有限，重用的效果也受到影响。而基于构件的软件复用和软件体系结构是近年来软件工程研究的重点之一，是提高软件生产率和软件质量的有效途径。因此，应该在软件体系结构层次上展开研究，用构件、连接件来描述安全需求（模型）。安全需求（模型）在软件体系结构层次上的描述有两个重要的意义：一方面，它可以解决安全需求（模型）向安全设计与实现平滑过渡的问题，提高复用粒度；另一方面，从软件体系结构角度来看，忽略了安全需求（模型）的软件体系结构是有缺陷的、不完整的。缺少了安全需求的软件体系结构不能在较高抽象层次上展示整个需求的全景视图。而且，在体系结构层次上人为地将业务需求和安全需求分开，不利于展现业务需求与安全需求之间的依赖关系。

实际上，在软件体系结构层次上描述安全需求（模型）的工作正在逐步展开，已有不少工作涉及到这一问题。文[9,16,17]在阐述 ABC/ADL（architecture description language，简称 ADL）、ABC 方法中，为了更好地描述系统的结构和行为，引入了 Aspect[18] 规约，并在实例——火车售票系统中将 Aspect 应用于系统的鉴别和权限控制。Aspect 是指软件系统中一些贯穿全局（cross-cutting）的特性，例如事务、日志等。面向 Aspect 的软件开发可以提供一种机制对一些贯穿全局的特性进行建模，使之与系统的实现模型分离，并且能够在组装时将这些特性插入到具体的系统中，使得系统的结构和行为更利于理解，更利于系统的设计、开发、维护和演化。但是，ABC/ADL 和 ABC 方法引入 Aspect 的目的并不是专门解决软件系统安全需求（模型）的体系结构描述。尽管采用 Aspect 的思想来描述软件系统安全需求的体系结构有它的合理性，但并没有体现出安全需求在基于 Internet 大型复杂系统中的重要地位，对此类软件系统的安全需求还缺少系统的体系结构方法研究。

除此以外，在国内外的安全产业界，厂商们均提出了各自的整体安全解决方案或安全产品解决方案，由于用户不同、应用不同、需求不同，导致没有任何一种安全产品或安全方案能解决所有的安全问题。必须通过安全产品的综合集成，才能提供针对特定应用需求的安全解决方案，因此安全系统的综合集成就显得尤为重要。但是，安全系统的综合集成是一个非常复杂的技术问题，必须有一套完整的综合集成方法和技术体系框架，将不同的安全产品通过综合集成的方式最大程度地发挥系统的整体防护效能。目前很多安全产品之间存在功能重叠的现象，例如一些防病毒软件集成了部分主机防火墙功能，一些防火墙产品又集成了部分入侵检测功能。一个安全产品既想集成尽可能多的功能，但又无法覆盖用户的全面需求。因此在安全系统建设过程中，用户购买了很多产品，但这些产品的综合效益得不到充分发挥。

一个很好的解决途径是将各种安全产品和安全软件的功能构件化，安全产品不再以独立的产品形态出现，而是以安全构件形态出现。每个安全构件都提供较为单一的安全功能，相互之间没有功能重叠。用户在建设安全系统时，根据需求从安全构件库中选取特定的安全构件，将这些安全构件通过一个综合构件平台加以集成。这些安全构件运行在安全构件平台上，对安全构件进行统一分发、加载、配置、管理和升级。基于这种思路，就可以快速地构建适合本单位需求的安全系统，而且构件之间相互联动，可以发挥最大的效益，既降低了投资，又增强了系统的整体性，同时便于统一维护和管理。但是，安全功能构件化需要安全需求的体系结构作为基础。因此，在体系结构层次上描述安全需求是安全功能构件化的前提，也是满足用户需求，提高安全产品开发效率、节约投资、避免安全产品功能重叠的需要。

目前，Internet 的迅速发展越来越深刻地影响着人们的日常生活。社会各个领域的应用越来越多地基于 Internet 实现和开展，包括金融、电信、宇航、电子商务、电子政务甚至军事等。在开发这些领域的大型复杂软件系统过程中，鉴于安全需求的重要性，在体系结构层次上研究安全需求（模型）的描述、表示和复用就显得十分紧迫而又必要。

2 软件系统安全需求的体系结构单元

软件系统安全需求的体系结构是整个体系结构的一部分。对于体系结构中的业务需求，我们采用传统体系结构中的基本元素：构件和连接件来描述；而对于安全需求，我们引入以下设计单元：安全构件（Security Component）、安全连接件（Security Connector）、半安全构件（Half-Security Component）和半安全连接件（Half-Security Connector）。用户可通过这些设计单元的组合来构建软件系统安全需求的体系结构。下面对引入的基本设计单元进行介绍。

2.1 安全构件

构件模型是构件的本质特征及构件间关系的抽象描述，它将构件组装所关心的构件类型、构件形态和表示方法加以标准化，使关心和使用构件的外部环境（如使用构件构造出的

应用系统、构件组装辅助工具和构件复用者等)能够在一致的概念模型下观察和使用构件。传统的构件表示形式如图1。

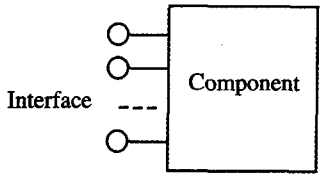

图1 传统构件模型的表示

安全构件是指系统中较为独立的安全功能实体,是软件系统中安全需求的结构块单元,是软件安全功能设计和实现的承载体。由于传统的构件模型仅强调统一的操作接口和界面,对构件的内部结构、与其它构件的依赖关系没有明确的语义和规约,不能完全表达软件系统的安全需求。因此,我们提出了安全构件的新模型。

定义1 安全构件是一个安全需求数据单元或一个安全需求计算单元,它由安全接口和实现模块组成。安全接口是安全构件与外部接触点的集合,即$\langle Security_Port_1, Security_Port_2, \cdots, Security_Port_n\rangle$,而每一个接触点 $Security\_Port_i$ 是一个十二元组$\langle SID, Pub_Security_i, Ext_Fun_entry_i, Ext_Sec_entry_i, Ext_Sec_Dependency_i, Ext_Enviroment_i, Ext_Sec_Enviroment_i, Pri_Sec_attribute_i, Sec_Beha_i, Msgs_i, Cons_i, Non\text{-}Func_i\rangle$,其中:

SID 是安全构件的标识;

$Pub\_Security_i$ 是安全构件第 i 个接触点能提供给环境或其它构件的安全功能集合;

$Ext\_Fun\_entry_i$ 是安全构件第 i 个接触点运行时所需要的其它构件的功能集合;

$Ext\_Sec\_entry_i$ 是安全构件第 i 个接触点运行时所需要的其它安全构件的功能集合;

$Ext\_Sec\_Dependency_i$ 是安全构件第 i 个接触点运行时与其它安全构件的依赖关系;

$Ext\_Enviroment_i$ 是安全构件第 i 个接触点运行时所需的环境要求集合;

$Ext\_Sec\_Enviroment_i$ 是安全构件第 i 个接触点运行时所需的安全环境要求集合;

$Pri\_Sec\_attribute_i$ 是安全构件第 i 个接触点的私有属性集合;

$Beha_i$ 是安全构件第 i 个接触点安全行为语义描述;

$Msgs_i$ 是安全构件第 i 个接触点所产生消息的集合;

$Cons_i$ 是对安全构件第 i 个接触点安全行为约束,它通常包括构件运行的初始条件、前置条件和后置条件。有时为了明确表示这3个条件,可把它写成 $Cons(init, pre\text{-}cond, post\text{-}cond)$,$init$,$pre\text{-}cond$ 和 $post\text{-}cond$ 分别表示初始条件、前置条件和后置条件的集合;

$Non\text{-}Func_i$ 是构件第 i 个接触点非功能说明,包括构件服务特性(如吞吐量、延迟等)、可靠性要求等。

将 $Ext\_Sec\_Dependency_i$、$Ext\_Enviroment_i$、$Ext\_Sec\_Enviroment_i$、$Pri\_Sec\_attribute_i$、Sec_Beha_i、$Msgs_i$、$Cons_i$ 和 $Non\text{-}Func_i$ 统称为安全构件的 *Internal specification*。为了使安全构件更加形象直观,便于用户理解,将安全构件用图符表示出来,如图2。$Security\_Port$ 用环表示;Ext_Fun_entry 用单框表示;而 $Ext\_Sec\_Dependency$ 用复框表示。

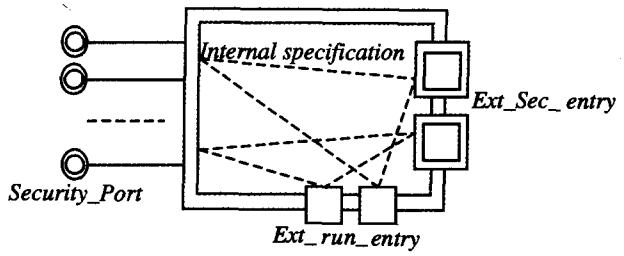

图2 安全构件的图形表示

2.2 半安全构件

软件系统在实现用户的安全需求时,必然有部分传统构件与安全构件存在交互和配置关系。如为了实现用户身份认证,登录构件(传统构件)与身份鉴别构件(安全构件)需要相互传递用户身份信息。这部分构件虽然不提供安全服务,但它们是大量传统构件和安全构件连接的桥梁和纽带,含有安全构件所需要的大量敏感信息,这些信息同样需要保护,这类构件应与一般的传统构件区别开来。因此,我们提出了半安全构件的新模型。

定义2 半安全构件是一个与安全构件有交互和配置关系的数据单元或一个计算单元,它由接口和实现模块组成。接口是构件与外部接触点的集合,即$\langle Port_1, Port_2, \cdots, Port_n\rangle$,而每一个接触点 $Port_i$ 是一个十三元组$\langle HSID, Pub_i, Ext_Fun_entry_i, Ext_Sec_entry_i, Ext_Sec_Dependency_i, Ext_Enviroment_i, Ext_Sec_Enviroment_i, Pri_attribute_i, Beha_i, HS_Beha_i, Msgs_i, Cons_i, Non\text{-}Func_i\rangle$,其中:

$HSID$ 是半安全构件的标识;

Pub_i 是半安全构件第 i 个接触点能提供给环境或其它构件的功能集合;

$Pri\_attribute_i$,是半安全构件第 i 个接触点的私有属性集合;

$Beha_i$ 是半安全构件第 i 个接触点一般行为语义描述;

HS_Beha_i 是半安全构件第 i 个接触点与安全构件的安全行为语义描述;

其它分量的含义与安全构件相同。将 $Ext\_Sec\_Dependency_i$,$Ext\_Enviroment_i$,$Ext\_Sec\_Enviroment_i$,$Pri\_attribute_i$,$Beha_i$,HS_Beha_i,$Msgs_i$,$Cons_i$ 和 $Non\text{-}Func_i$ 统称为半安全构件的 *Internal specification*。为了使半安全构件更加形象直观,便于用户理解,将半安全构件用图符表示出来,如图3。$Port$ 用圆表示;Ext_Fun_entry 用单框表示;而 Ext_Sec_entry 用复框表示。

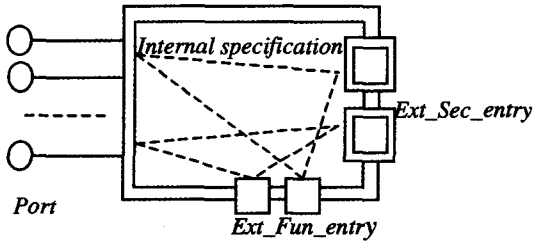

图3 半安全构件的图形表示

同时,安全构件和半安全构件还可以拥有自己的内部体系结构,这样的构件称为复合安全构件和复合半安全构件。利用复合安全构件和复合半安全构件的概念,开发人员可以逐步精化系统的安全体系结构模型,更好地进行设计与开发。

2.3 安全连接件和半安全连接件

连接件(connector)是软件体系结构的一个组成部分,它通过对构件间的交互规则的建模来实现构件间的连接。与构件不同,连接件不需编译。连接件模型是 SA 研究的重要贡献之一,它显式地描述了构件之间的交互关系或交互协议。而在传统的程序设计中,构件间交互的描述往往散布在发生交互的各个构件之中。通过连接件,SA 提供了一种在较高抽象层次观察、设计系统并推理系统行为和性质的方式,也提供了设计和实现可复用性更好的构件甚至复用连接件的途径。传统连接件模型的表示形式如图 4。

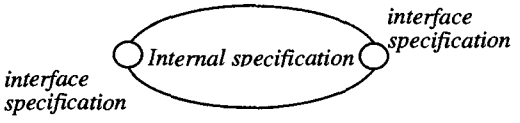

图 4 传统连接件模型的图形表示

定义 3 安全连接件是安全构件与安全构件之间维持行为关联和信息传递的途径。它是一个六元组〈*SID*, *Sec_Role*, *Sec_Beha*, *Msgs*, *Cons*, *Non-Func*〉,其中:

SID 是安全连接件的标识;

Sec_Role 为安全连接件与安全构件的交互点的集合,每个 *Sec_Role*=〈*SRId*, *Action*, *Event*, *LConstrains*〉组成。其中:*SRId* 是 *Sec_Role* 的标识;*Action* 是 *Sec_Role* 活动的集合,每个活动由事件的连接(谓词)组成;*Event* 是 *Sec_Role* 产生的事件集合;*LConstrains* 是 *Sec_Role* 的约束集合。把 *Sec_Role* 从安全连接件的其它属性分开来描述的目的是突出连接件的多态性,即一个安全连接件可同时与多个安全构件相连。

Msgs 是安全连接件中各 *Sec_Role* 中事件产生的消息的集合;

Sec_Beha 是安全连接件行为的语义描述;

Cons 是安全连接件约束的集合,它包括连接件的初始条件、前置条件和后置条件。有时为了明确表示这 3 个条件,可把它写成 *Cons*(*init*, *pre-cond*, *post-cond*),*init*, *pre-cond* 和 *post-cond* 分别表示初始条件、前置条件和后置条件的集合。

Non-Func 是安全连接件的非功能说明,包括延迟、可靠性说明等。

将 *Sec_Beha*、*Msgs*、*Cons* 和 *Non-Func* 统称为安全连接件的 *Internal specification*。为了使安全连接件更加形象直观,便于用户理解,将安全连接件用图符表示出来,如图 5。*Sec_Rolet* 用环表示;*Internal specification* 用椭圆环表示。

图 5 安全连接件的图形表示

定义 4 半安全连接件是安全构件与半安全构件之间维持行为关联和信息传递的途径。它是一个八元组〈*HSID*, *Role*, *Sec_Role*, *Beha*, *Sec_Beha*, *Msgs*, *Cons*, *Non-Func*〉,其中:

HSID 是半安全连接件的标识;

Role 为半安全连接件与半安全构件的交互点的集合,每个 *Role*=〈*HSRId*, *Action*, *Event*, *LConstrains*〉组成。其中:*HSRId* 是 *Role* 的标识;*Action* 是 *Role* 活动的集合,每个活动由事件的连接(谓词)组成;*Event* 是 *Role* 产生的事件集合;*LConstrains* 是 *Role* 的约束集合;

Beha 是半安全连接件行为的语义描述;

其它分量的含义与安全连接件相同。将 *Beha*、*Sec_Beha*、*Msgs*、*Cons* 和 *Non-Func* 统称为半安全连接件的 *Internal specification*。为了使半安全连接件更加形象直观,便于用户理解,将安全连接件用图符表示出来,如图 5。*Sec_Role* 用环表示;*Role* 用园表示;*Internal specification* 用椭圆环表示。

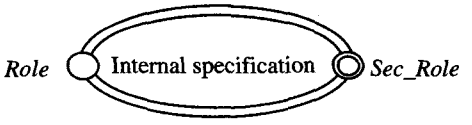

图 6 半安全连接件模型的表示

半安全连接件与安全连接件不同,安全连接件两端只能和安全构件连接,而半安全连接件一端只能连接半安全构件,另一端只能连接安全构件。

安全连接件和半安全连接件也可以拥有自己的内部结构(具有内部结构的安全连接件或半安全连接件称为复杂安全连接件或复杂半安全连接件)。这样可以为设计人员提供更强的抽象描述能力,并且能够将高层的复杂连接件逐步精化到最终的实现上。

3 软件系统安全需求的软件体系结构模型

定义 5 软件系统安全需求的体系结构模型:

$$Sec\_Arch=\bigcup_{i=0}^{i_1} HS_i+\bigcup_{j=0}^{j_2} S_j+\bigcup_{m=0}^{m} HSC_m+\bigcup_{n=0}^{n} SC_n+\bigcup_{k=0}^{k} C_k\ (S\rightarrow S)+\bigcup_{l=0}^{l_4} HC_l\ (HS\leftrightarrow S)$$

其中:

i, j, k, l, m, n 是非负整数;

S_j、HS_i 是安全构件和半安全构件;

HSC_m、SC_n 是安全连接件和半安全连接件;

$C_k(S\leftrightarrow S)$、$HC_l(HS\leftrightarrow S)$ 是安全连接件与安全连接件、半安全连接件与安全连接件之间的连接约束。

上式表明,软件系统安全需求的体系结构由安全构件、半安全构件通过安全连接件、半安全连接件连接而成。将 $\bigcup_{j=0}^{n_2} S_j+\bigcup_{n=0}^{n} SC_n+\bigcup_{k=0}^{n_3} C_k(S\leftrightarrow S)$ 称为安全核;$\bigcup_{i=0}^{n_1} HS_i+\bigcup_{m=0}^{m} HSC_m+\bigcup_{l=0}^{n_4} HC_l(HS\leftrightarrow S)$ 称为安全边缘。安全核的主要功能是为安全需求提供所需要的全部安全服务;安全边缘的主要目的是展现业务需求的软件体系结构和安全需求的体系结构之间的交互和配置。由定义 5 可得安全需求的体系结构性质:

• 封闭性(envelopment)。即安全构件与安全构件、半安全构件与安全构件、安全构件与结构、半安全构件与体系结构、体系结构与体系结构连接后仍是一个体系结构。

• 层次性(hierarchy)。即安全需求的体系结构可由安全构件连接而成,而体系结构又可以再经过连接组成新的更大的体系结构。

• 可扩充性(expansibility)。即一个满足安全需求的新安全构件可以通过连接加入到结构中。

• 交互性(interaction)。即通过体系结构的安全边缘将体系结构的安全核和业务需求的体系结构连接起来。安全边

缘反映了业务需求和安全需求的配置和交互。

4 网上购物系统身份认证子系统的软件体系结构模型

在这一节中，我们结合“网上购物”系统展示安全需求的体系结构元素(安全构件、半安全构件、安全连接件、半安全连接件)和体系结构模型。根据定义5，“网上购物”系统身份认证子系统的体系结构模型如下：

```
Sec-Arch={
    S={
        认证构件(Authentication);
        用户身份信息构件(User Identity);
        密钥或口令构件(Key);
        用户访问属性构件(User Access Attribute);
        用户和主体绑定构件(User-Subject Banding)
      }
    HS={
        登录构件(Login)
      }
    SC={
        认证构件↔用户身份信息构件;
        认证构件↔密钥或口令构件;
        认证构件↔用户访问属性构件;
        认证构件↔用户和主体绑定构件
      }
    HC={
        登录构件↔认证构件
      }
}
```

为了更加形象直观，我们将网上购物身份认证子系统的软件体系结构用图表示出来，如图7。在图7中，①为登录模块(Login)，是一个半安全构件；③、⑤、⑦、⑨和⑪是安全构件，其中③是认证模块(Authentication)，⑤是用户身份信息模块；⑦是用户密钥或口令模块，⑨是用户和主体绑定模块，⑪是用户访问属性模块。②是半安全连接件，连接登录模块和认证模块；④、⑥、⑧和⑩是都安全连接件，④连接用户身份信息模块和认证模块，⑥连接用户密钥或口令与认证模块，⑧连接用户与主体绑定模块，⑩连接的是用户访问属性与认证模块。

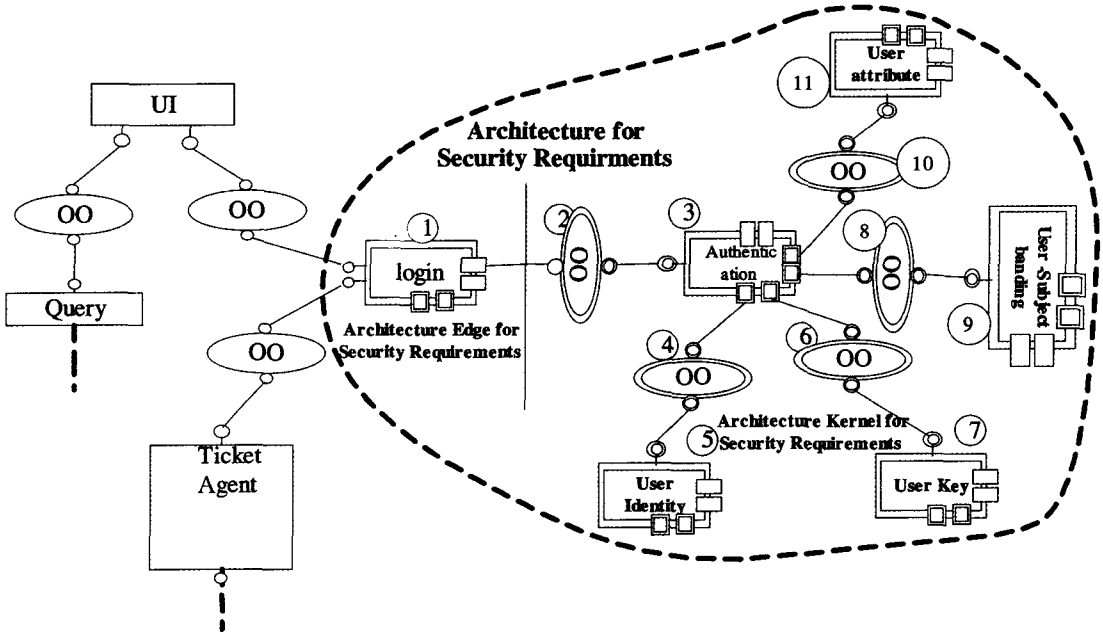

图7 网上购物认证子系统的软件体系结构

结论 软件体系结构是近年来软件工程界关注的重点。安全需求的软件体系结构具有重要的意义，主要表现在：

(1) 将安全需求与业务需求在软件体系结构层次上同时展现出来，这是对传统软件体系结构的扩展和补充，这样的软件体系结构才是完整的；

(2) 在体系结构层次上同时展现业务需求和安全需求，更有利于将这些需求(包括业务需求和安全需求)的全局组织和控制结构、功能到计算元素的分配、业务计算元素和安全计算元素间的高层交互等转化为设计；

(3) 软件系统安全需求的体系结构模型，是(满足安全需求的)安全模型的构件化表示，便于安全模型的精化、提高安全模型代码复用粒度；

(4) 将安全需求与业务需求在软件体系结构层次上同时展现出来，有利于清楚表达业务需求与安全需求之间的交互和依赖关系；

进一步的工作包括：①定义安全体系结构模型的描述语言。我们拟从两个方面着手，一是扩充现有的得到大家认同的ADL，使它能够描述系统的安全体系结构；另一方面是定义全新的ADL语言。②需要进一步研究安全体系结构的体系结构风格以及形式化。

参 考 文 献

1 孙昌爱，金茂忠，刘超. 软件体系结构研究综述. 软件学报，2002，13(7)：1228～1235

2 Medvidovic N, Rosenblum D S, Taylor R N. A language and environment for architecture-based software development and evolution. In: Proceedings of the 21st International Conference Software Engineering (ICSE'99), 1999. 44～53. http://www.ics.uci.edu/～dsr/icse99-dradel.pdf

3 Garlan D, Monroe R, Wilel D. ACME: An architecture description interchange language. The CASCON 97, Toronto, Ontario 1997

4 Shaw M, Deline R, Klein DV, et al. Abstractions for software architecture and tools to support them. IEEE Transaction on Software Engineering, 1995, 21(4): 314～335

(下转第277页)

可以检查对遗产软件进行重工程产生的 UML 交互模型是否完备。文[16]提出了一种基于场景的测试方法，它是基于活性顺序图(Live Sequence Charts, LSCs)[5]的一个简单子集的，但活性顺序图不能用于描述向后和双向强制一致性规约。一些验证 Java 程序的工作[13,11,22]是基于模型检验技术[4]的，但由于状态空间爆炸问题，该技术的应用受到了一些限制。

总结 本文提出了一种 UML 行为图驱动的对 Java 程序运行时验证。该工具以一个随机的测试用例集作为输入，运行经过插装的被测 Java 程序，得到一组用于验证的程序运行轨迹。通过对程序运行轨迹和 UML 行为图中合法的事件序列的比较，该工具可以对程序的动态行为规约进行检查。在本文中，我们具体介绍了该验证框架的结构，并对原型工具进行了实例研究。

该验证工具的主要优势在于产生随机测试用例的低成本。这一优势可以使我们得到一个自动化并易于使用的工具，大大降低验证工作的成本。因此，这一技术在中小软件企业中会有广泛的应用前景。

下一步的工作主要是对工具的功能进行扩充，特别是对包含一个类的多个实例的 UML 规约的验证，以及对并发场景的验证。另外，本工具中的随机测试用例生成器尚不能完全自动化，需要对随机测试用例的生成算法进行研究，以适应对输入域较复杂的系统的验证。

参 考 文 献

1 Bartetzko D, Fischer C, Moller M, et al. Jass-Java with Assertions. Electronic Notes in Theoretical Computer Science, 2001, 55(2)

2 Bjork R C. The Simulation of an Automated Teller Machine. http://www.math-cs.gordon.edu/local/courses/cs211/ATMExample/Links.html

3 Brorkens M, Moller M. Dynamic event generation for runtime checking using the jdi. Electronic Notes in Theoretical Computer Science, 2002, 70(4):21～35

4 Clarke E M, Grumberg O, Peled D A. Model Checking. MIT Press, 1999

5 Damm W, Harel D. LSCs: Breathing Life into Message Sequence Charts. Formal Methods in System Design, 2001, 19(1):45～80

6 Drusinsky D. Semantics and Runtime Monitoring of TLCharts: Statechart Automata with Temporal Logic Conditioned Transitions. Electronic Notes in Theoretical Computer Science, 2001

7 Finkbeiner B, Sankaranarayanan S, Sipma H. Collecting Statistics over Runtime Executions. Electronic Notes in Theoretical Computer Science, 2002, 70(4):1～19

8 Foundation for Intelligent Physical Agents. FIPA Iterated Contract Net Iteration Protocol Specification. http://www.fipa.org/specs/fipa00030/, 2002

9 Gutjahr W J. Partition Testing vs Random Testing: The Influence of Uncertainty. IEEE Trans on Software Engineering, 1999, 25(5):661～674

10 Hamlet R. Random Testing. Encyclopedia of Software Engineering, 1994. 970～978

11 Havelund K, Pressburger T. Model checking JAVA Programs Using JAVA PathFinder. Int J Software Tools for Technology Transfer, 2000(2):336～381

12 Havelund K, Rosu G. Monitoring Java Programs with Java PathExplorer. Electronic Notes in Theoretical Computer Science, 2001, 55(2):1～18

13 Holzmann G J, Smith M H. Software Model Checking: Extracting Vrificatioin Models from Source Code. In: Proc. 12th Int'l Conference on Formal Description Techniques (FORTE/PSTV '99), 1999

14 Huang J C. Program Instrumentation and Software Testing. Computer, 1978, 11(4):25～32

15 Kim M, Kannan S, Lee I, et al. Java-MaC: A Runtime Assurance Tool for Java Programs. Electronic Notes in Theoretical Computer Science, 2001, 55(2)

16 Lettrai M, Klose J. Scenario-based monitoring and testing of real-time UML models. In: Proc. 4th Int'l Conference on Unified Modeling Language (UML '01) Lecture Notes in Computer Science 2185, 2001. 45～80

17 Li X, Qiu X, Wang L, et al. UML State Machine Diagram Driven Runtime Verification of Java Programs. Manuscript, 2006

18 Li X, Wang L, Qiu X, et al. Runtime Verification of Java Programs for Scenario-Based Specifications. In: Proc. 11th Int'l Conference on Reliable Software Technologies (AE '06), 2006

19 Nortel Networks Corporation. FIPA-OS Distribution Notes. http://fipa-os.sourceforge.net, 2002

20 Object Management Group. UML Superstructure Specification, v2.0. http://www.omg.org/docs/formal/05-07-04.pdf, 2005

21 Oriat C. Jartege: a Tool for Random Generation of Unit Tests for Java Classes. In: Proc. 2nd Int'l Workshop on Software Quality (SOQUA '05) Lecture Notes in Computer Science 3712, 2005. 242～256

22 Park D Y, Stern U, Skakebak J U, et al. Java Model Checking. In: Proc. 1st Int'l Workshop on Automated Program Analysis, Testing and Verification, 2000

23 Peled D A. Software Reliability Methods. Springer, 2001

24 Rumbaugh J, Jacobson I, Booch G. Unified Modeling Language Reference Manual, 2nd ed. Addison-Wesley, 2004

25 Stolz V, Huch F. Runtime Verification of Concurrent Haskell Programs. Electronic Notes in Theoretical Computer Science, 2004

(上接第 264 页)

5 Luckham D C, Veral J. An event-based architecture definition language. IEEE Transaction on Software Engineering, 1995, 2(9):717～734

6 Taylor R, Medvidovic N, Anderson K, E, et al. A component and message-based architectural style for GUI software [J]. IEEE Transactions on Software Engineering, June 1996. 390～406

7 Allen R, Garlan D. A formal basis for architectural connection. ACM Transactions on Software Engineering and Methodology, 1997, 6(3):213～249

8 Magee J, Kramer J. Dynamic structure in software architectures. In: Kaiser G. E. ed. Proceedings of the ACM SIGSOFT'96: the 4th Symposium, Foundations of Software Engineering (FSE4), New York: ACM Press, 1996. 3～14

9 王晓光，冯耀东，梅宏. ABC/ADL：一种基于 XML 的软件体系结构描述语言. 计算机研究与发展，2004，141(1):1521～1531

10 冯铁，张家晨，陈伟，等. 基于框架和角色模型的软件体系结构规约. 软件学报，2000，11(8):1078～1086

11 马俊涛，傅韶勇，刘积仁. A-ADL：一种多智能体系统体系结构描述语言. 软件学报，2000，11(10):1382～1389

12 骆华俊，唐稚松，郑建丹. 可视化体系结构描述语言 XYZADL. 软件学报，2000，11(8):1024～1029

13 张家晨，冯铁，陈伟，等. 基于主动连接件的软件体系结构及其描述方法. 软件学报，2000，11(8):1024～1029

14 蔡谊，郑志蓉，沈昌祥. 基于多级安全策略的二维标识模型[J]. 计算机学报，2004，27(5):620～624

15 Grsecurity. http://www.grsecurity.net/

16 Mei Hong, Chang Jichuan, Yang Fuqing. Composing software components at architectural level. The Int Conf on SoftwareTheor yandPractice, Beijing, 2000

17 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

18 Kiczales G, Lamping J, Mendhekar A, et al. Aspect-Oriented programming. In: Proceedings of the European Conference on Object-Oriented Programming (ECOOP), LNCS 1241, Springer-Verlag, 1997. 220～242. http://citeseer.nj.nec.com/63210.html