基于知识的软件可信性需求获取

刘 春[1],王 越[1],金 芝[1,2]
(1.中国科学院数学与系统科学研究院,北京 100190;2.北京大学信息科学与技术系,北京 100871)

摘 要: 针对软件系统的可信性需求获取,本文提出了基于知识的需求获取方法.该方法认为软件系统给环境所带来的问题是导致软件系统不可信的原因,因而系统的可信性需求就是为避免软件系统给环境带来问题而定义的对策,并且现实世界积累的软件系统失效和其引发问题的相关知识,可以帮助用户来识别软件系统可能带来的问题以及相应的对策.该方法可以弥补分析人员在获取软件需求时的知识不足,从而帮助分析人员发现更多的软件系统的可信性需求.

关键词: 软件可信性; 软件失效知识; 可信性需求
中图分类号: TP311 **文献标识码:** A **文章编号:** 0372-2112 (2010) 2A-188-06

Eliciting Dependability Requirements: a Knowledge-based Approach

LIU Chun[1], WANG Yue[1], JIN Zhi[1,2]
(1. *Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing* 100190, *China*;
2. *School of Information Science and Technology, Peking University, Beijing* 100871, *China*)

Abstract: This paper proposes a method for the system dependability requirement elicitation based on experiential knowledge about how the software often causes issues to the environment. The basic idea of this approach is to capture deniability requirements by identifying the issues that the system may bring to the environment and determining the countermeasures to tackle them. To identify the potential issues and determine the countermeasures, we propose to utilize the accumulated knowledge about how the system fails and brings issues to the environment. By taking advantage of this kind of knowledge, the proposed approach can help the analysts to find more dependability requirements for the software to be developed.

Key words: software dependability; software failure knowledge; dependability requirement

1 引言

软件在现实社会中扮演着越来越重要的角色.它在给我们带来巨大的利益的同时,也可能会因为其失效而带来巨大的危害.因此,软件的可信性已经引起越来越多人的重视[1~4].可信性被认为是一个集合性的概念,它包含了一些传统的质量属性,比如可用性(availability)、可靠性(reliability)、可靠安全性(safety)、保密性(confidentiality) 等.国际信息处理联合会(IFIP)定义可信性为系统的可信度[2],一个可信的系统可以让人们信赖它所提供的服务.

为了建立一个可信的软件系统,其中一个基本问题就是获取软件系统的可信性需求.而为了获取软件系统的可信性需求,首先理解软件系统为什么不可信无疑是有意义的.对此,我们认为正是由于系统在服务的过程中给环境所带来的各种问题才是导致软件系统不可信的原因.用户对软件系统的可信性需求正是为了避免系统失效可能会造成的各种问题.因此,为了获取软件系统的可信性需求,了解软件系统通常如何失效、如何给环境带来问题是必要的.虽然现在有一些软件公司和研究人员研究了软件的相关缺陷[5~8],但是他们往往只是关注软件的某一个方面,我们认为,软件所引发的问题已不仅仅是由软件本身的缺陷所引起的,现在的软件系统越来越多的处在复杂的环境下,环境给软件系统所带来的威胁也已成为软件引发问题所不可忽视的因素,它们常包括人为的操作失误、环境灾难、各种恶意的攻击等等.另一方面,各个软件企业之间也很少有关于软件失效的相关数据的共享,对软件而言,还没有一个业界共享的关于软件所引发问题的相关数据收集和分析.

由于软件系统的复杂性,一个系统的失效模式可能不能被用于另一个系统的失效分析,但是我们必须承认的是有很多的软件系统的失效是有着相同的原因.因

收稿日期:2009-09-07;修回日期:2009-12-28
基金项目:国家自然科学基金(No.90818026;60736015);国家基础研究发展规划 973 项目(No.2009CB320701);国家杰出青年基金(No.60625204)

此,收集软件的失效数据,并进行分析最终可以得到关于软件失效的一般知识,这些知识可以帮助我们获取新的系统需求使得未来开发软件系统避免给环境带来类似的问题.这一做法其实在软件领域之外已得到人们的应用,比如美国的国家公路交通安全管理局就管理者几个数据库[3],来记录全国的公路所发生的交通事故的相关信息,同时该数据库也记录着来自各大汽车生产商所提供的关于保险理赔、用户投诉等相关的信息.这些信息可以被研究人员用来分析交通事故和汽车设计特征之间的关系,进而提高汽车的安全性.因此,本文的目标就是讨论如何来组织有关软件失效以及其所引发问题的相关知识,它包括软件所引发问题的原因、这些原因所导致的问题、以及相应的对策等,并如何利用这些知识来帮助可信性需求的获取.

下面,本文将首先介绍软件失效的相关经验知识,包括知识的元模型、知识的组织与管理;然后介绍如何根据这些知识来获取可信性需求.

2　失效知识

本节我们将首先介绍软件失效相关知识的分类,然后介绍知识的元模型,最后介绍知识的组织与管理.

2.1　知识分类

对知识的分类可以有助于人们理解软件是如何失效的,更有助于收集新的知识.软件失效的相关知识通常包括失效的原因、结果、对策等.其中认识软件失效的原因对我们尤为重要,所以下面我们将按照软件失效的原因来对软件失效的相关知识进行分类.

在文献[6]中,Landwehr 等认为,分类不是对样本的一个简单的组织,它隐含着分类者对样本所在的空间的一个认识.依据 Michael Jackson 的机器-环境模型[9],软件系统的需求(R)、规格说明(S),领域知识(D)之间存在如下关系:

$$S, D \vdash R$$

上式说明系统需求的满足需要系统的行为满足规格说明,领域满足领域知识的所描述的特征.这就意味着当系统的行为不满足需求规格说明,或者领域的特征不满足领域知识的描述时,用户的需求就得不到满足,软件就失效了,就可能会给环境带来问题.因此,导致软件诱发问题的原因可以分为软件本身的缺陷和环境的因素.我们将前者称为内部威胁,后者称为外部威胁.

内部威胁:内部威胁指的是软件在开发过程中或者维护中所引入的缺陷,比如不完全的参数验证、逻辑错误等.这类威胁更加关注于设计或者实现过程中的错误,其中有很多文献都总结了常见的该类威胁[5~8].在文献[8]中,作者列举了七类该种威胁:

(1)输入验证和表示

(2)滥用应用程序接口

(3)安全特征

(4)时间和状态

(5)错误

(6)代码质量

(7)封装

外部威胁:外部威胁指的是环境中存在的各种外在因素.由于软件系统必须要与现实中的环境中的各种实体交互,交互过程中的环境实体的行为和环境本身的变化都可能对软件系统造成威胁,它常包括:

(1)恶意的攻击,

(2)环境实体的输入性错误,包括用户的操作错误、硬件平台的故障、其它软件系统的缺陷等

(3)环境的突发事件(比如火灾、地震、断电等)

内部威胁和外部威胁的最大区别在于,内部的威胁是在开发或者维护中所引入的各种错误,它们在软件系统存在以前是不存的,而外部的威胁是存在于环境之中的,不会因为软件系统的存在而存在.但是,内部威胁和外部威胁二者之间往往也存在着联系,有时外部威胁之所以对系统产生影响是因为内部威胁的存在,比如外部威胁环境实体的输入性错误,它之所以对系统构成威胁是因为系统的输入验证存在着缺陷.

因此,根据上面的分类,我们对软件系统失效的相关知识可以分类为:跟系统本身相关的知识,它们包括系统内部威胁,以及相应的对策等;跟环境相关的知识,它们包括系统所处环境的外部威胁,所造成的问题,以及通常可以采取的对策等.

2.2　知识元模型

为了有效的利用我们所搜集和积累的知识,对知识的合理组织是必要的.为此,我们提出了如图 1 的知识的元模型.

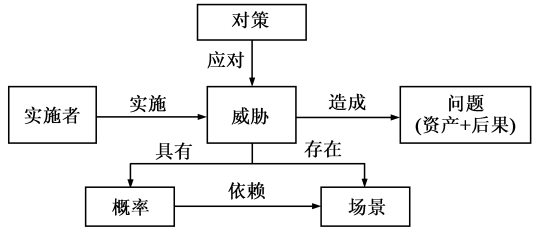


图1　软件失效知识的元模型

软件系统的失效往往是由于某些威胁所引起的,并且会导致某些问题的发生.问题往往同一定的资产相联系,并通过对该资产所造成的后果来表达,这里的资产是对利益相关者来说重要的需要保护的对象,比如时间、数据、设备、人的生命、环境本身等.同时,每种威胁都会存在一定的实施者,且都会在一定的场景下

以一定的概率发生,在不同的场景下相同的威胁发生的概率可能不同.为了应对威胁并减少软件系统失效所造成的损失,对每种威胁及相应的问题往往也存在着一定的对策.

根据图 1,我们提出软件系统失效的相关知识可以通过如下形式的断言来表达:

Knowledge(K_i, *Cau*, *Pro*, *Act*, *Sce*, *Iss*, *Cou*)

其中 K_i 是该条知识的标识,*Cau* 代表威胁,*Pro* 代表威胁发生的概率,*Act* 是威胁的实施者,*Sce* 是威胁发生的场景,*Iss* 是给环境带来的问题,*Cou* 是相应的对策.

在知识库中,软件系统的失效知识应该按照不同的类别进行组织.同时,知识的组织应该使得用户可以从威胁、问题、实施者、对策这些概念中的任何一个开始,去检索用户需要的知识.因此,知识库应该为上述概念中每一个概念建立一个视图,表 1 展示了在信息安全领域围绕信息泄露问题知识库可以采用的问题视图.围绕这些知识,下面我们将介绍如何基于这些知识来获取软件系统的可信性需求.

表 1 知识库的问题视图

问题:信息泄露					
威胁	类型	实施者	场景	概率	对策
嗅探	外部威胁	黑客	数据在网络传输,嗅探在靠近客户端或者服务器的网络端口监听数据传输	0.5	数据加密
木马	外部威胁	黑客	入侵客户端主机或者服务器,收集数据	0.7	数据加密
仿冒和篡改输入控件	外部威胁	黑客	控件在下载到客户那里之前已经被仿冒或者篡改了,用户安装之后输入信息就会被记录下来	0.2	代码签名
……	……	……	……	……	……

3 基于知识的可信需求获取过程

为了获取软件系统的可信性需求,我们认为,软件系统在服务中给环境所带来的问题是导致软件系统不可信的原因,那么软件系统的可信性需求就是为了避免软件系统给环境带来问题所定义的对策.因此,软件系统可信性需求的获取就应该围绕软件系统可能引发的各种问题来展开.

但是,从 IFIP 对可信性的定义可以看出,可信性是一个主观性的概念,它反映的是软件系统的利益相关者对软件系统的信任程度.当一个问题发生时,它是否是我们所应该关注的这取决于利益相关者.因此,软件系统对环境所造成的问题是相对于利益相关者而言的.因此,我们认为,可信性需求的获取过程应该从识别不同角色的利益相关者开始.对每一个角色的利益相关者,需求工程师同他们沟通,并在知识库的支持下,识别出系统可能会引发的问题,以及造成这种问题的潜在威胁,然后针对这些威胁与问题定义应对策略,这些对策就构成了未来软件系统的可信性需求.对此,我们提出了如图 2 所示的基于知识的可信需求获取过程.

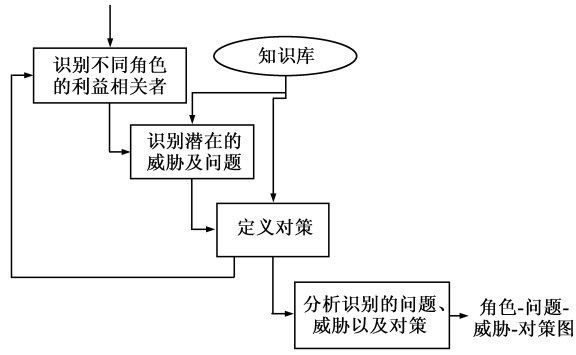


图2 可信性需求获取过程

下面本文将借助于网上银行系统来详细的介绍该过程中的每个不同的活动.

网上银行系统:由于网络环境的复杂性,在开展网上银行业务时,一个可信的网上银行系统是必不可少的.在使用网上银行服务时,用户往往需要在本地启动连入互联网的机器,然后开始输入个人信息,输入的信息通过网络传到银行服务器,服务器处理后将结果还回给用户.下面我们将借助于该系统来介绍可信性需求获取过程的各个活动.

(1)识别软件系统不同角色的利益相关者: 软件系统的利益相关者指的是与软件系统存在联系的人,它通常包括以下四类不同角色的人[10]:①用户,②赞助方,③规划人员,④其生活将会受到软件系统影响的人.

这里,用户指的是将要与系统直接交互的人员.赞助方指的是支持并赞助系统开发的人员.规划人员则指的是关注软件系统对社会所产生的影响的人员,他们一般是政府部门,或者是标准化组织.而生活将要受到软件系统影响的人,一般指的是不直接与软件系统交互,但是软件系统可能会对他们的生活造成影响的人.利益相关者又可以分为用户和非用户两类,而后者就包括软件系统的赞助方、规划人员和其生活将要受到软件系统影响的人.

在获取软件系统可信性需求时,需求工程师可以按照上述不同角色的人群去识别潜在的利益相关者.对网上银行系统而言,不同角色的利益相关者主要有:银行客户、银行的管理者、政府部门.

(2)识别潜在的威胁以及问题:对于用户和非用

户的利益相关者,我们采用不同的方法来识别问题以及潜在的威胁.

对于用户,他们要与系统进行交互,因此我们可以识别他与系统交互的场景,然后在知识库中匹配相似的场景,并依据相似的场景来发现潜在的威胁以及可能的问题.但是,知识库所给出的潜在的威胁和问题最终还需要用户来决定是否采纳.而对于非用户的利益相关者,他们不与软件系统交互,但他们仍然不希望软件系统带来他们所不期望的问题.对于他们所关注的问题,我们可以从他们的关注点出发,首先识别他们的关注点,然后根据关注点可能受到的损害来确定他们所关注的问题.这里,关注点就是利益相关者特别强调的特征,它往往也可以通过利益相关者所关注的资产和该资产所应具有的约束来表达.对于引发这类问题的威胁的识别,我们可以首先识别利益相关者关注的资产所涉及的场景,然后通过在知识库中匹配相似的场景,来识别潜在的威胁.

针对每种威胁和相应的问题,需求工程师可以和相应的利益相关者协商,来确定在相应场景中威胁发生的概率 P,以及问题所带来的损失 L.获取威胁发生的概率和问题所带来的损失的精确值是困难的,为此我们可以根据当前的场景与知识库中描述的场景的匹配程度来决定当前威胁所发生的概率,而对于问题所带来的损失,我们则可以估计它们的相对值,比如取值为[0,1]之间的值.

对于网上银行系统,其用户就是银行客户,他们与系统交互的场景可以用图3来表示,这一场景是网络应用中常见的网络传输场景.该场景涉及的实体主要有:用户、本地机器、个人信息、网络、银行主机.对于该场景,我们通过网络传输场景来查询知识库,可以得知本地机器和银行主机将受到木马入侵,而网络则可能受到嗅探监听,这些都可以导致问题"用户个人信息发生泄漏".银行主机还可能受到 DOS 攻击,从而引发另外一个问题"用户请求的响应时间延迟".

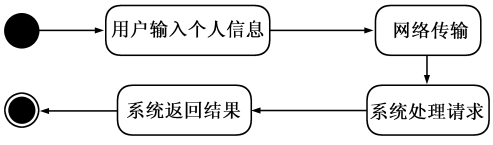


图3　用户办理网上业务场景

对于非用户,银行管理者则是系统开发的赞助方和支持者,他们关注的也是客户的信息不能被泄露,否则他们将要承担法律责任.而政府部门则是规划人员,他们关注的也是居民的个人信息不能被泄露.因此,对于银行的管理者和政府部门他们关注的问题都是客户的个人信息泄露.而这一问题所涉及的资产就是用户的个人信息,而该资产涉及的场景也是网络传输.对于网上银行系统,当前所识别的问题以及相应的威胁,以及问题所来到的相对损失和威胁发生的概率如图5所示.

(3)定义对策:对策是为了控制威胁和相应问题所带来的风险,它们构成了软件系统的可信性需求.对于对策的定义,我们提出了图4所示的过程.

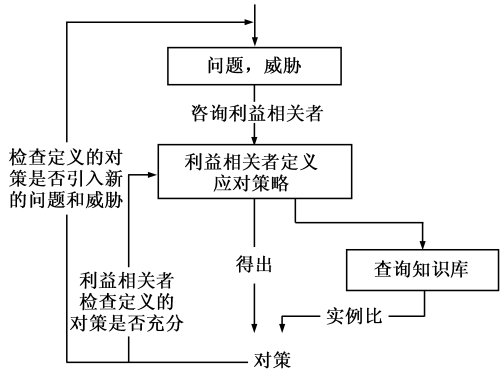


图4　定义应对策略

如图4所示,在定义应对策略的时候,首先是针对每一个威胁和相应的问题,咨询相关的利益相关者,然后由利益相关者提供他们所能提供的对策,但是对于一些威胁和问题由于利益相关者可能并不太了解,此时就需要查询知识库,然后将知识库中的相应策略实例化即可作为对策.但是,最终的应对策略仍然需要利益相关者来检查确认是否充分.同时,由于所定义的对策可能会引入新的问题和威胁,在这种情况下就需要对定义的每种对策,重复前面的两步活动以检查是否存在新的问题和威胁.

对网上银行系统的各个问题和威胁所定义的对策如图5所示.其中为了消除木马或者嗅探对客户个人信息的窃取,需要对用户输入的个人信息进行加密,此时就需要在用户的本地机器上安装加密控件.针对加密控件,网上存在可能模仿的控件,而且由于用户对网络知识的了解可能很少,或者由于用户缺乏耐心或者一时疏忽等,很可能用户从网上下载并安装了模仿的加密控件,此时如果用户输入个人信息那么就会发生用户的个人信息被泄露的问题.虽然这一问题我们已经识别了,但是通过对引入的对策进行分析,我们发现了新的威胁"加密控件被模仿,并被用户下载安装".

(4)分析识别的问题、威胁和定义的对策:由于不同角色的利益相关者的需求之间可能存在着冲突,也可能因为资金和时间的因素当前的开发过程不允许考虑所有的问题和威胁,在这种情况就需要对前面活动所得到的问题、威胁和对策进行进一步的分析.首先需要做的是识别相互冲突的问题,由于问题可以表达为资产和相应的后果,那么对冲突问题的识别可以将涉及相同资产的所有问题聚在一起,然后分析各个问题

所表达的后果之间是否存在冲突即可.

解决冲突的问题之后,需要做的就是估计每个问题及相应威胁所带来的风险,并依据风险对各个问题进行优先级排序,只有这样才能识别那些重要的问题.风险的计算可以按照下式进行:

$$R = L \times \sum P_i$$

其中 R 是每个问题所带来的风险,L 是其带来的损失,P_i 是导致该问题的第 i 个威胁所发生的概率.

对于网上银行系统,其可能引发的问题的风险如图 5 所示.可信性需求获取过程最终可以得到如图 5 所示的角色-问题-威胁-对策关系图,处在最上层的是识别的角色,然后是角色关注的问题和导致这些问题的威胁,最后是对策.该图展示了为什么需要可信性需求,以及每个需求所涉及的利益相关者.

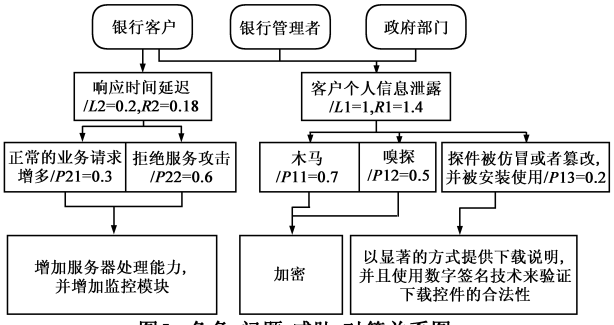


图5 角色-问题-威胁-对策关系图

4 相关工作

对软件失效的分析,很早就已经引起了人们的注意.在文献[5]中,作者列举了当时的操作系统所存在的缺陷.在文献[6]中,Landwehr 从三个方面来对软件系统所存在的漏洞进行分类:起源(漏洞是如何进入到系统中的),时间(它们又是什么时间进入系统中的),地点(它们又是在什么地方显露出来).该工作后来被 Viega[7]进行了扩展,它不仅按照起源、时间、地点进行分类,还按照它们造成的后果、发生的概率等其他信息对漏洞进行分类.在最近的工作[8]中,作者关注于具体的危及到系统安全的错误,并将它们分为七类.上述工作虽然从不同的角度对软件系统所面临的威胁进行了总结,但是他们往往只是关注于软件系统本身可能存在的缺陷,并没有讨论环境因素对软件系统可信性所构成的威胁.

由于可信性集合了多个质量属性,因此可以采用基于目标的方法来获取可信性需求,将可信性的每个属性看作一个目标,然后将它们操作化[11].但是,由于各个质量属性之间往往存在着复杂的相关性,理解并区分它们之间的关系对软件系统的利益相关者和需求工程师来说往往是很困难的.在本文中,我们认为软件系统所带来的问题是导致系统不可信的原因,并提出了从问题出发来表达可信性需求,这不仅避免考虑各个可信性属性的定义以及它们之间的关系,并且更易于利益相关者表达可信性需求.

与本文工作相近的是文献[12]提出的面向误用例的质量需求工程.该文也基于经验知识,他们将经验知识按照每个质量属性对应的威胁、威胁造成的后果、威胁的实施者、威胁的相应对策来组织成一览表(checklist).但是,作者仍然将各个非功能属性分开独立进行处理,并且根据各个不同的属性来对这些知识进行分类.本文关注的是软件系统可信性,并且认为问题的存在显示了可信性需求的必要性,所以我们将知识分类为系统相关的和环境相关的,而不去讨论具体的可信性属性.

6 总结

本文提出了基于知识的可信性需求获取方法.我们认为软件系统给环境带来的问题是导致软件系统不可信的原因,而软件系统的可信性需求就是为了避免软件系统带来这些问题.我们强调了现实世界中所积累的关于软件失效并给环境带来问题的相关知识对获取软件需求的重要性,为此提出了基于该类知识的可信性需求获取方法.

下一步我们将搜集整理关于软件失效及其所引发的问题的相关知识,并开发相应的工具来支持基于知识的可信性需求获取.

参考文献:

[1] Avi_zienis A, Laprie J C, Randell B, Landwehr C. Basic concepts and taxonomy of dependable and secure computing[J]. IEEE Transactions on Dependable and Security Computing, 2004, 1(1): 11 – 33.

[2] Donzelli P, Basili V. A Practical framework for eliciting and modeling system dependability requirements: experience from the NASA high dependability computing project[J]. The Journal of System and Software, 2006, 79: 107 – 119.

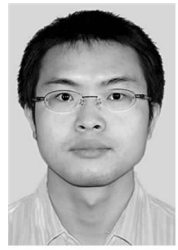
[3] Jackson D. A direct path to dependable software[J]. Communications of the ACM, 2009, 52(4): 78 – 88.

[4] 陈火旺,王戟,董威.高可信软件工程技术[J].电子学报,2003,31(12):1933 – 1938.
Chen H, Wang J, Dong W. High confidence software engineering technologies[J]. Acta Electronica Sinica, 2003, 31(12): 1933 – 1938. (in Chinese)

[5] Abbott R P, Chin J S, Donnelley J E, etc. Security analysis and enhancements of computer operating systems[R]. NBSIR 76 – 1041, National Bureau of Standards, ICST, Washington, D.C., 1976.

[6] Landwehr C E, Bull A R, McDermott J P, Choi W S. A taxonomy of computer program security flaws, with examples[J]. ACM Computing Surveys, 1994, 26(3): 211 – 254.
[7] Viega J. The CLASP Application Security Process, Volume 1.1 Training Manual.
[8] Tsipenyuk K, Chess B, McGraw G. Seven pernicious kingdoms: a taxonomy of software security errors [J]. IEEE Security & Privacy, 2005, 3(6): 81 – 84.
[9] Jackson M. The meaning of requirements [J]. Annals of Software Engineering, 1997, 3: 5 – 21.
[10] Alexander I F, Stevens R. Writing better requirements [M]. Addison-Wesley Professional, Great Britain, 2002.
[11] Cysneiros L M, do Prado Leite J C S. Nonfunctional requirements: from elicitation to conceptual models[J]. IEEE Transactions on Software Engineering, 2004, 30(5): 328 – 350.
[12] Herrmann A, Paech B. MOQARE: misuse-oriented quality requirements engineering[J]. Requirements Engineering, 2008, 13: 73 – 86.

作者简介:



刘　春　男,1982 年生于河南信阳,中国科学院数学与系统科学研究院博士研究生,研究方向为软件系统的可信性需求获取.
E-mail: liuchun@amss.ac.cn

王　越　男,1983 年生于河南昌,中国科学院数学与系统科学研究院博士研究生,研究方向为基于知识的需求工程.
E-mail: wangyue@amss.ac.cn

金　芝　女,博士,研究员,博士生导师,CCF 高级会员,主要研究领域为软件需求工程、领域建模和基于知识的软件工程.