知识引导的软件可信性需求的提取

王 越[1] 刘 春[1] 张 伟[2),3)] 金 芝[1),2),3)]

[1)](中国科学院数学与系统科学研究院 北京 100190)

[2)](高可信软件技术教育部重点实验室(北京大学) 北京 100871)

[3)](北京大学信息科学技术学院软件研究所 北京 100871)

摘 要 软件系统的可信性已经成为一个受到广泛关注的焦点问题.开发可信的软件系统的前提是在需求阶段提取恰当的可信性需求.能否提取出足够好的软件可信性需求,不仅依赖于需求工程师对未来软件系统可能面临的威胁的认识,还依赖于其对各种威胁有效的应对措施的了解和掌握.目前缺少系统化的方法指导软件可信性需求的提取.文中提出一个软件可信性需求上层本体作为软件可信性需求的概念框架.在此框架的基础上,开发了一个软件可信性需求知识库,定义了软件可信性需求模式框架以及如何根据知识库的内容进行模式实例化的过程,帮助提取可信需求.最后利用一个股票交易系统作为案例展示了该方法的可行性.

关键词 可信需求;可信需求模式;可信需求获取;需求工程

中图法分类号 TP311 **DOI 号**: 10.3724/SP.J.1016.2011.02165

Knowledge Guided Software Trustworthiness Requirements Elicitation

WANG Yue[1)] LIU Chun[1)] ZHANG Wei[2),3)] JIN Zhi[1),2),3)]

[1)](*Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing* 100190)

[2)](*Key Laboratory of High Confidence Software Technologies of Ministry of Education (Peking University), Beijing* 100871)

[3)](*Institute of Software, School of Electronics Engineering and Computer Sciences, Peking University, Beijing* 100871)

Abstract As software systems are used widely and deeply in society, trustworthiness, which refers to the ability to operate correctly under any situation, becomes one of the hot topics in practice and research. To develop trustworthy software system needs the elicitation of trustworthiness requirements. But the elicitation of the trustworthiness requirements needs the requirements engineer to possess lots of knowledge about undesired conditions a software system may face and the corresponding countermeasures the software system could take to deal with them. There still lacks a systematic method to guide the elicitation of trustworthiness requirements. This paper introduces a meta-model of trustworthiness, founds a knowledge base according to this meta-model, presents a trustworthiness requirements pattern and a method about how to generate patterns from the knowledge base to help eliciting trustworthiness requirements. Finally a stock trading system is used to show the feasibility of this method.

Keywords trustworthiness requirements; trustworthiness requirements pattern; trustworthiness requirements elicitation; requirements engineering

---

收稿日期:2011-08-29;最终修改稿收到日期:2011-09-20.本课题得到国家自然科学重点基金(90818026,60736015)、国家"九七三"重点基础研究发展规划项目基金(2009CB320701)资助.**王 越**,男,1983 年生,硕士研究生,主要研究方向为基于知识的需求工程.E-mail:wangyue@amss.ac.cn.**刘 春**,男,1982 年生,博士研究生,主要研究方向为需求工程.**张 伟**,男,1978 年生,博士,讲师,主要研究方向为软件复用、需求工程.**金 芝**,女,1962 年生,博士,教授,主要研究领域为需求工程、软件工程和知识工程.

1 引 言

随着软件应用范围的不断扩大,软件系统的可信性成为一个广泛关注的焦点问题.特别在那些与用户利益具有紧密关系的应用领域内(例如网上银行、电子商务等领域),软件系统的异常行为会对用户造成无法容忍的损失.因此,如何确保一个软件系统的可信性,成为对软件学术界和产业界面临的一个重要挑战.

要构建一个可信的软件系统,首先需要捕获软件系统的需求.一般而言,首先需要捕获软件系统的功能需求,也即软件系统将做什么,将向外提供什么样的服务.然而,在很多应用场景下,仅仅保证功能正确性和有效性是不够的.比如,在网络环境下软件系统在向外提供服务的过程中,可能面临各种干扰,如拒绝服务攻击、病毒、未预测的负载等来自于环境的各种威胁.

特别在目前 Internet 技术日益普及的情况下,一方面,软件系统所处的环境逐渐呈现出动态开放的特性.为了完成特定业务功能,软件系统可能需要与环境中存在的各种对象(包括人、软件对象、硬件对象、传感器等)密切交互.而环境对象的多样性以及环境对象自身特性的差异性,都会对软件行为的可预期性带来不可预期的负面影响.另一方面,环境的动态开放性也使得具有不良意图的环境对象能够通过与软件系统的交互对其正常行为产生负面影响,从而导致软件系统可信性的降低.因此,要构建一个可信的软件系统,除了正确的功能性需求之外,还需要识别软件系统将面临哪些威胁,并引入防御抵抗措施来应对这些威胁.这些对应对措施的需求构成软件系统的可信性需求.

如何系统、有针对性地识别软件系统将要面临的威胁,并确定合适的应对策略是需求工程师所面临的一个难题.不同于功能性需求,软件系统的可信性需求是由各种可能威胁所驱动的,而用户对软件系统可能面临的威胁往往并不十分清楚.因此,可信性需求的捕获不是来自于用户的业务逻辑,而来自于需求工程师对软件将面临的威胁的洞察力和对处理威胁的技术的应用能力.

目前关于安全和可信性工程的研究和实践积累了一些关于软件环境威胁相关的知识.例如,一些研究者提出了描述环境威胁的指导性框架[1];一些研究者则在实现层上对程序代码的缺陷给出了相应的分类[2];另外有些研究者针对软件系统的安全性威胁给出了相应的安全性需求[3].这些知识可以组织成可信需求模式,帮助需求工程师识别软件将面临的威胁以及制定应对策略.

目前,已经有一些关于可信需求模式的工作.比如,安全需求模式[10]从系统工程的角度出发用模式表达和组织关于安全需求的知识.非功能需求模式[14]描述了用模式来提取非功能需求的过程,但并不涉及如何复用可信需求知识来提取需求.当前对于可信需求模式的研究中,可信性需求模式都是利用手工建立的.当出现新知识时,它需要工程人员人工建立新模式,并且不能复用已有知识.

本文提出了一个软件系统可信性需求模型的概念框架.在对软件系统可信性的相关概念以及这些概念之间的相互联系进行了系统调研和整理的基础上,采用特征建模方法[5],设计并开发了一个软件可信性需求特征知识库,提出了软件可信性需求的模式框架,并在此基础上提出了一个基于知识生成软件可信性需求的方法.

本文第 2 节介绍可信需求概念框架以及根据这个概念框架建立的可信需求知识库;第 3 节介绍可信需求模式,给出可信需求模式的框架;第 4 节给出从可信需求知识库中生成可信需求模式的步骤;第 5 节给出一个案例来验证生成的可信需求模式的适用性;第 6 节对本文的重要相关工作进行说明与分析;最后,第 7 节对全文进行总结,并指出下一步的研究工作.

2 软件可信性需求知识库

本节首先给出可信需求概念框架,然后介绍根据该概念框架建立的可信需求知识库.

2.1 软件可信性需求概念框架

综合考虑软件的可信性,我们提出一个软件可信性需求的上层本体作为软件可信性需求概念框架(如图 1 所示).它包含 5 个相互关联的概念类以及 7 个概念类之间关联.下面我们介绍每个概念类以及概念类之间的关联.

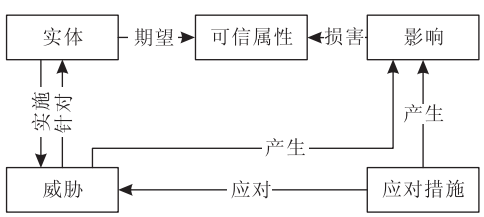


图 1 软件可信性需求概念框架

2.1.1　概念类

我们定义如下 5 个概念类：

(1) 实体. 实体是指对利益相关者具有价值的事物，比如数据、通道、软件及服务、硬件、人员等.

(2) 可信属性. 可信属性是系统利益相关者期望实体具有的一些属性. 例如，用户期望某些数据具有保密性和完整性，期望某些服务具有可用性和一定的时间性能要求.

(3) 威胁. 威胁是对系统的一种潜在攻击. 它由环境中的实体来实施，而且通常利用系统或者环境中实体存在的缺陷来实施. 例如，由黑客发动的分布式拒绝服务攻击(DDoS)通过产生大量的服务请求来消耗掉系统的计算或者带宽资源，使得正常的服务请求得不到满足.

(4) 应对措施. 应对措施是指用来防范威胁的可能手段. 它通过防范对系统的威胁使系统更加安全可信，通常包括用户个人的习惯和安全意识，组织的策略以及系统需求等. 例如，为了防止 SQL 注入恶意输入损害系统的正常运行，而采取输入验证这一应对措施.

(5) 影响. 影响是指对系统可能产生的负面作用. 它通常是由对系统实施威胁之后所产生的结果，主要是对于系统所应有的可信属性的破坏. 例如，当系统遭受到拒绝服务攻击的时候，正常的用户请求长时间才能得到满足或者不能得到满足，则造成运行时间加长或者损失可用性；但一些系统所采用的应对措施也会产生一些影响. 例如，为了增加存储文件的保密性，对存储文件进行加密，而加解密的过程会增加系统的运行时间.

2.1.2　概念类关联

我们定义概念类间的如下 7 种关联.

〈实体，实施，威胁〉

其描述的是实体发起一项威胁. 通常威胁是由环境中的实体发起的. 例如，用户会实施恶意输入这项威胁.

〈实体，期望，可信属性〉

其描述的是系统的利益相关者期望实体拥有一项可信属性. 例如，系统的用户会期望数据拥有私密性.

〈威胁，针对，实体〉

其描述的是一项威胁针对某个实体为目标. 例如，DDoS 是针对软件为攻击目标.

〈威胁，产生，影响〉

其描述的是一项威胁会对系统产生一个负面影响，如 DDoS 会使系统失去可用性.

〈应对措施，产生，影响〉

其指当系统采用一些应对措施时会对系统带来一些负面影响，如当系统采用输入验证时，会造成系统处理时间变长这样的影响.

〈应对措施，应对，威胁〉

其描述的是用一项应对措施来阻止/预防一个威胁. 例如，利用输入验证来应对用户的恶意输入.

〈影响，损害，可信属性〉

其描述的是影响会破坏某些可信属性. 例如失去可用性会损害可用性.

2.2　可信需求知识库

可信需求知识库的构建是在深入系统地研究一组安全需求相关的代表性工作的基础上进行的，其中包括一些标准，如卡内基·梅隆大学软件工程实验室推出的关于安全功能需求集和安全保证需求集的共同标准[3]、微软针对网络程序的威胁以及应对措施的工作[9]、美国国家标准技术局中关于计算机系统安全的标准①等以及一些具有影响的研究成果，如 MOQUARE(Misuse-Oriented Quality Requirements Engineering)方法[7-8]、Donald Firesmith 的开放过程框架等②、Mead 等人[4]撰写的 SQUARE(System Quality Requirements Engineering)的报告及其案例研究报告③等.

在本文研究中，我们根据上述可信性需求概念框架，采用基于特征建模的技术来构建知识库. 特征建模通过对特定领域中的系统或概念的共性和变化性特征进行分析，获得对领域知识的系统化理解.

在充分研究和深入分析上述成果的基础上，根据我们提出的可信需求概念框架，通过对上层概念的精化，识别这些特征具有的更为具体的子特征，并建立特征精化关系图；然后在此基础上，分析这些特征之间存在的各种依赖关系. 需要指出的是，在不同的概念中，特征的表现形式会体现出多样性. 例如，在实体这个概念中，特征所指代的是一种特定类型的实体，表现为名词形式；而在威胁这个概念中，特征指代的是一种特定类型的危险，表现为动词短语的形式. 同样，不同概念的特征之间存在的关系也不尽相同.

可信需求知识库可以表示为

KB=〈Term，Relation，Constraint〉，

① http://csrc.nist.gov/publications/nistpubs/800-12/handbook.pdf
② http://www.opfro.org/
③ http://www.cert.org/archive/pdf/05sr005.pdf

其中 Term 为特征概念集，Relation 为特征概念关系集，Constraint 为关系约束集. 目前为止，所建立的可信需求知识库中包含 121 个特征概念，719 条特征概念之间的关联以及 5 类概念间关联约束. 知识库的特征概念层次及相互关系如图 2 所示. 图 3 展示了知识库的特征概念浏览界面.

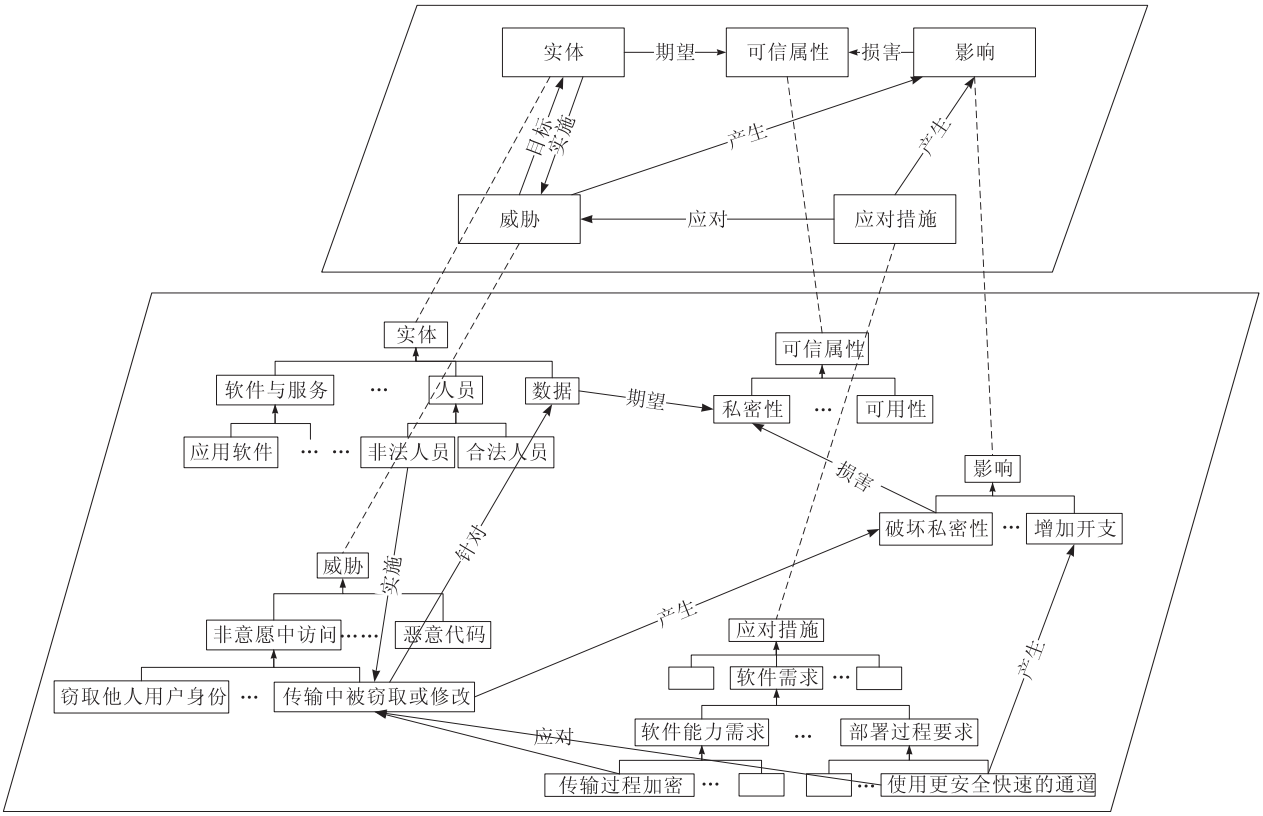


图 2 知识库概念层次及相互关系示意图

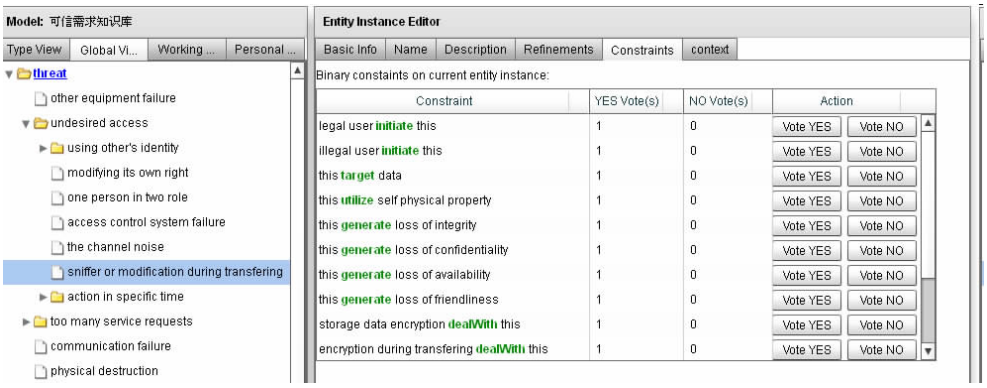


图 3 可信需求知识库的特征概念浏览视图

2.2.1 特征概念

特征概念集由 5 个特征概念子集组成：

$$Term = Entity \cup Threat \cup Impact \cup Property \cup Countermeasure,$$

其中 Entity 为实体特征集，Threat 为威胁特征集，Impact 为影响特征集，Property 为可信属性特征集，Countermeasure 为应对措施特征集合. 我们目前的知识库中包含 13 种实体概念，31 种威胁概念，14 种影响概念，12 种可信属性概念，51 种应对措施概念.

2.2.2 特征概念关联

特征概念关联集由两部分构成：

$$Relation = Rintra \cup Rinter,$$

其中 Rintra 为同一类特征概念间的实例关联集，Rinter 为不同类特征概念间的实例关联集.

同一类特征概念之间只存在 is-a 关联，用来表

示同一类特征概念之间的父概念/子概念关系.

Rintra={⟨x,is-a,y⟩|x∈Term,y∈Term}.

例如：

⟨人员,is-a,实体⟩

⟨非意愿中的访问,is-a,威胁⟩

⟨传输过程中被截获,is-a,非意愿中的访问⟩

等是同一类概念中的关联.

不同类概念间的关联有“实施(initiate)”,“期望(deserve)”,“针对(target)”,“应对(dealwith)”,“产生(generate)”,“损害(jeopardize)”共 6 种.

Rinter={⟨x,R,y⟩|x∈Term,y∈Term,
R∈{initiate,deserve,target,dealwith,
generate,jeopardize}}.

例如：

⟨输入验证,dealwith,用户的误操作⟩

⟨恶意代码,target,应用软件⟩

等是不同类特征概念间关联.

特征概念关联的结构和含义如表 1 所示.

表 1　特征概念间的相互关系

关联	形式化表示	解释
is-a	Entity $\xrightarrow{n\ \ 1}$ Entity Threat $\xrightarrow{n\ \ 1}$ Threat Impact $\xrightarrow{n\ \ 1}$ Impact Countermeasure $\xrightarrow{n\ \ 1}$ Countermeasure Property $\xrightarrow{n\ \ 1}$ Property	用来表示同一类特征概念之间的父概念/子概念关系. 每一类概念中的概念通过该关系形成一棵树,这些树的根分别是概念框架中的概念:“实体”,“威胁”,“应对措施”,“影响”,“可信属性”. 我们定义 Root={“实体”,“威胁”,“应对措施”,“影响”,“可信属性”}
initiate	Entity $\xrightarrow{n\ \ n}$ Threat	表示用实体来发起某项威胁
deserve	Entity $\xrightarrow{n\ \ n}$ Property	表示实体被期望拥有某项属性
target	Threat $\xrightarrow{n\ \ 1}$ Entity	一项威胁以某个实体为目标
dealwith	Countermeasure $\xrightarrow{n\ \ n}$ Threat	表示应对措施阻止/预防某类威胁
generate	Threat $\xrightarrow{n\ \ n}$ Impact Countermeasure $\xrightarrow{n\ \ n}$ Impact	表示威胁可以产生某类影响 表示采用一些应对措施时会对系统带来一些负面影响
jeopardize	Impact $\xrightarrow{n\ \ n}$ Property	表示影响会损害到某些期望属性

2.2.3　特征概念关系约束

Rintra 中的关系需要满足如下两类约束.

约束 1(父特征唯一).　如果⟨x_1,is-a,x_2⟩∈Rintra,且⟨x_1,is-a,x_3⟩∈Rintra,则 $x_2=x_3$.

这个约束表明,每一个特征概念最多有一个父特征概念.

约束 2(根确定性).　对于∀x_1∈Term,且 $x_1\notin$Root,则∃x_2∈Term,满足⟨x_1,is-a,x_2⟩∈Rintra.

这个约束表明在每一个概念特征集合中的元素,除了根节点,都有一个父节点. 约束 1 和约束 2 一起规定了每一类概念特征集合中的元素构成了一棵树.

Rinter 中的关系需要满足如下 3 类约束.

约束 3(关联的向上传递 1).　如果 x_1,x_2,y∈Term,⟨x_1,is-a,x_2⟩∈Rintra,⟨x_1,_,y⟩∈Rinter,则⟨x_2,_,y⟩∈Rinter.

约束 4(关联的向上传递).　如果 x,y_1,y_2∈Term,⟨y_1,is-a,y_2⟩∈Rintra,⟨x,_,y_1⟩∈Rinter,则⟨x,_,y_2⟩∈Rinter.

例如,如果

(1)⟨使用更安全快速的通信,dealwith,传输过程中被截获和修改⟩∈Rinter;

(2)⟨传输过程中被截获和修改,is-a,非意愿中的访问⟩∈Rintra,⟨非意愿中访问,is-a,威胁⟩∈Rintra,

则根据约束 4,得到

(1)⟨使用更安全快速的通信,dealwith,非意愿中访问⟩∈Rinter;

(2)⟨使用更安全快速的通信,dealwith,威胁⟩∈Rinter;

而另一方面,因为

(1)⟨使用更安全快速的通信,is-a,部署过程需求⟩∈Rintra;

(2)⟨部署过程需求,is-a,软件需求⟩∈Rintra;

(3)⟨软件需求,is-a,应对措施⟩∈Rintra;

根据约束 3,可以得到

⟨应对措施,dealwith,威胁⟩∈dealwith.

约束 5(完整性).　∀x∈Threat,且!∃x_1∈Threat 满足⟨x_1,is-a,x⟩∈Rintra,则∃y∈Countermeasure,满足⟨y,dealwith,x⟩∈Rinter.

这类约束表明,每一个处于叶子节点的威胁都必须有一个应对措施来处理,否则遇到这种威胁时

不知道如何处理.

3 可信需求模式

模式是针对普遍存在的问题所提出的解决方案,它是一种有效的知识复用方法.可信需求模式复用人们所积累的软件可信性知识,包括影响软件系统可信性的相关威胁以及相应的对策等.在这一节中,我们给出可信需求模式的框架.

3.1 可信需求模式框架

可信需求模式框架如表 2 所示,其中各项的含义为

Name:该模式的名字,通常用一个自然语言语句来表达.

Goal:该模式所要保护的东西,用

实体概念+可信属性概念

表示,其中,实体概念∈Entity,可信属性概念∈Property.

Issue:该模式所要处理的问题,用

威胁概念

表示,其中,威胁概念∈Threat.

NegativeEffect:该模式中的问题所产生的后果,用

影响概念

表示.其中,影响概念∈Impact.

Countermeasure:该模式推荐的解决该问题所使用的应对措施,用

应对措施概念

表示.其中,应对措施概念∈Countermeasure.

SideEffect:采取该模式中的应对措施会带来的一些副作用,用

影响概念

表示其影响概念∈Impact.

CausalityDiagram:一个有向图,指的是该模式的问题以及应对措施所产生的总体关系,利用三元组$\langle a,r,b\rangle$来表示,a,b 为模式中的元素,r 为它们之间的关系.包含的关系有

$\langle i,\text{hurt},\text{Goal}\rangle$

其中 $i\in$NegativeEffect 表示负面影响,即 i 会损害 Goal.

$\langle \text{Issue},\text{generate},i\rangle$

其中 $i\in$NegativeEffect.

$\langle \text{Countermeasure},\text{dealwith},\text{Issue}\rangle$

$\langle \text{Countermeasure},\text{generate},se\rangle$

其中 $se\in$SideEffect.

表 2 可信需求模式的框架

槽名	解释	形式
Name	模式的名字	句子
Goal	模式所要保护的目标	实体概念+可信属性概念
Issue	模式要处理的问题	威胁概念
NegativeEffect	式中的问题产生的影响	影响概念集合
Countermeasure	模式推荐的解决该问题使用的应对措施	应对措施概念
SideEffect	采用该应对措施会带来的副作用	影响概念集合
CausalityDiagram	该模式的问题以及应对措施所产生的总体关系	三元组$\langle a,r,b\rangle$的集合,其中 a,b 为模式的不同部分,r 为两者之间关系

4 模式生成过程

在本节中,我们提出一个从知识库中自动生成可信需求模式的过程.生成模式的过程如下.

过程 1. Generating Pattern Procedure.

输入:a,其中 $a\in$Entity,且$!\exists a_1\in$Entity 满足$\langle a_1,$is-a,$a\rangle\in$Rintra. 即 a 为实体特征树的叶子节点

输出:Pattern

1. Identify Goal.

$P=\{p\mid p\in\text{Property},!\exists p_1\in\text{Property},\langle p_1,\text{is-a},p\rangle\in\text{Rintra},\langle a,\text{deserve }p\rangle\in\text{Rinter}\}$

Select $p\in P$

Goal$=(a,p)$

2. Identify Issue and NegativeEffect

$I=\{i\mid i\in\text{Impact},!\exists i_1\in\text{Impact},\langle i_1,\text{is-a},i\rangle\in\text{Rintra},\langle i,\text{jeopardize},p\rangle\in\text{Rinter}\}$,

$T=\{t\mid t\in\text{Threat},!\exists t_1\in\text{Threat},\langle t_1,\text{is-a},t\rangle\in$ Rintra,满足$\langle t,\text{target},a\rangle\in$Rinter,且$\exists i\in I$,满足$\langle t,\text{generate},i\rangle\in$Rinter$\}$

Select $t\in T$

$I'=\{i\mid i\in I,\langle t,\text{generate},i\rangle\in\text{Rinter}\}$

Issue$=t$

NegativeEffect$=I'$

3. Determinate Countermeasure

$C=\{c\mid c\in\text{Countermeasure},!\exists c_1\in\text{Countermeasure},\langle c_1,\text{is-a},c\rangle\in\text{Rintra},\langle c,\text{dealwith},t\rangle\in\text{Rinter}\}$

Select $c\in C$

Countermeasure$=c$

4. Propose SideEffect

$SE=\{se\mid se\in\text{SE},!\exists se_1\in\text{Impact},\langle se_1,\text{is-a},se\rangle\in\text{Rintra},\langle c,\text{generate},se\rangle\in\text{Rinter}\}$

SideEffect＝*SE*

5. Generate CausalityDiagram

CD＝{}

For each *i*∈NegativeEffect

CD＝*CD*∪{⟨*i*,hurt,Goal⟩,⟨Issue,generate,*i*⟩}

CD＝*CD*∪{⟨Countermeasure,dealwith,Issue⟩}

For each *se*∈SideEffect

CD＝*CD*∪{⟨Countermeasure,generate,*se*⟩}

ausalityDiagram＝*CD*

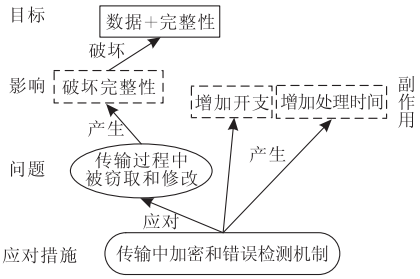
6. Determine Name

Name＝利用 Countermeasure 防范 Issue 的模式.

第 1 步根据输入的实体类型，利用 deserve 关系找到该实体类型所期望的可信属性. 然后从中选择一个形成目标. 第 2 步利用目标中的实体类型和可信属性以及 generate 和 jeopardize 关系找到一个威胁集，根据具体的场景，从中选择一个作为 Issue，并找出它所产生能够破坏目标中的可信属性的影响作为 NegativeEffect. 第 3 步利用 Issue 中的威胁和 dealwith 关系得到可以采用的应对措施集合，从中选择一个作为 Countermeasure. 第 4 步根据上一步的 Countermeasure 和 generate 关系找出 SideEffect. 第 5 步根据所得到的模式中各个元素产生 CausalityDiagram. 第 6 步利用规则生成模式的名字.

假设应用场景为网络数据传输. 利用实体类型“数据”为输入，第 1 步得到 *P*＝{可用性，私密性，完整性}，我们在这里选取完整性，得到 Goal＝(数据，完整性)；第 2 步得到 *I*＝{破坏完整性}，产生 *I* 且针对数据的威胁集 *T*＝{来自人员的破坏，提供假的设备，恶意代码，非法修改自己权限，访问控制系统被破坏，传输过程中被截获和修改，信道噪音干扰信息内容，拒绝服务攻击，地震海啸等，用户的误操作，恶意输入，与其它软件的交互问题，与其它硬件的交互问题}，因为应用场景为在网络中传输数据，所以我们选择 Issue＝传输过程中被截获和修改，则相应 NegativeEffect＝{破坏完整性}；第 3 步得到可以阻止传输过程被修改的应对措施集合 *C*＝{存储数据加密，传输过程中加密和错误检测机制，消息日志，使用更安全快速的通信}，我们选择 Countermeasure＝传输过程中加密和错误检测机制；第 4 步利用 generate 关系得到传输过程中加密和错误检测机制产生的副作用为 SideEffect＝{计算时间增长，增加开支}；第 5 步我们根据上面的元素画出 CasulityDiagram；然后最后一步生成模式名字. 上面步骤产生的模式如表 3.

表 3　防范传输过程截获和修改的模式

槽名	内容
Name	利用传输中加密和错误检测机制防范传输过程截获和修改的模式
Goal	数据＋完整性
Issue	传输过程截获和修改
NegativeEffect	破坏完整性
Countermeasure	传输中加密和错误检测机制
SideEffect	计算时间增长； 增加开支
CausalityDiagram	

我们在提取可信需求的时候，首先要找出系统以及它所在环境中的实体以及它们的类型(类型是指在知识库中的实体概念)，然后利用这些类型作为输入生成可信模式.

5　案例研究

在本文中我们采用一个股票交易系统作案例，对比利用本文所提出的方法生成的可信需求模式以及这个系统的需求文档中相关的需求，来考察所提出的方法的适用性. 这个股票交易系统的角色是电子经纪人. 通过股票交易系统进行交易的流程是：用户将订单提交给经纪人，然后经纪人将订单提交给系统，系统将接收到的订单发送给证券交易所. 对于其中某些类型的订单，系统需要从股票信息提供者那里及时获得股票信息来更新订单信息. 订单一旦在股票交易所完成执行，交易所就会将结果返回给系统. 系统通知经纪人并且更新他的可以交易余额. 他和其它需要交互的实体的上下文图如图 4 所示.

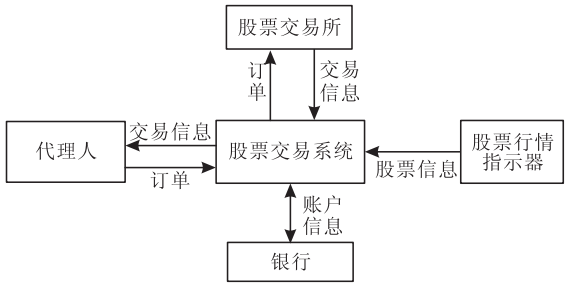


图 4　股票交易系统的上下文图

原来的股票交易系统支持同时在线用户的能力不足,随着公司规模扩展,系统访问量增加,系统经常发生崩溃,需要对系统进行重构.我们利用可信需求模式来获得新系统的一些可信需求.

我们首先识别该股票交易系统中的实体以及它们的类型,得到的结果如表4.

表4 股票交易系统的实体

实体	类型
订单	数据
交易结果	数据
股票行情	数据
账户	数据
交易员	人员
股票交易系统	软件及服务

5.1 股票交易系统的可用性模式

股票交易系统是软件及服务类型的实体.

第1步,对于这样类型的实体从知识库中可得它的可信属性,包括:可用性、易用性、时间要求方面等,但是在这里主要是它的可用性的问题.所以以它的可用性为目标来生成模式,得到目标为"股票交易系统的可用性".

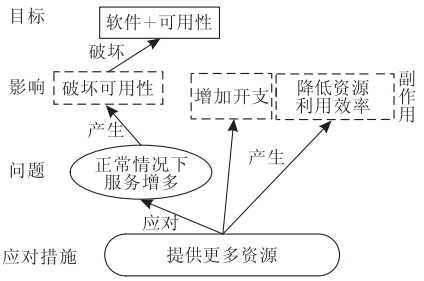
第2步,从知识库中得到威胁,包括:提供假的设备,恶意代码,正常情况下服务增多,拒绝服务攻击,用户的误操作,恶意输入,与其它软件的交互问题,与其它硬件的交互问题,其它硬件设备失效,通信失效,计算机硬件故障.在这里我们考虑"正常情况下服务增多",指的是随着公司的发展而造成的访问请求的增多.它所产生的影响为"损失可用性".

第3步,从知识库中得到应对措施,包括:在受攻击的时候提供必要的服务,灾难恢复机制,配置管理,入侵检测系统,冗余的硬件设备,提供更多的计算资源,考虑计算设备的处理能力,使用更安全快速的通信,考虑计算的扩展性.我们在这里采取的措施为"提供更多的资源",而在实际重构的时候采取的措施是将系统结构从单机系统变为了分布式系统.

第4步,从知识库中得到该应对措施产生的副作用为:降低资源利用效率;增加开支.

具体可用的模式如表5所示.

表5 防范正常情况下的服务增多的模式

槽名	内容	实例化模式内容
Name	利用提供更多资源来防范正常情况下的服务增多的模式	
Goal	软件+可用性	股票交易系统的可用性
Issue	正常情况下的服务增多	由于用户大量访问造成系统崩溃
NegativeEffect	损失可用性	系统崩溃;用户访问响应时间增长
Countermeasure	提供更多资源	采取分布式处理系统架构
SideEffect	降低资源利用效率;增加开支	重构的费用会非常高;计算资源利用效率会减低
CausalityDiagram		

5.2 数据的私密性模式

订单,交易结果,股票行情,账户都是数据类型的实体.

第1步,对于这样类型的实体,可以从知识库得到它的可信属性,包括:私密性、可用性、完整性.而这些数据一旦泄露出去就会造成巨大的经济损失,所以它们的私密性就变得非常的重要,所以以它们的私密性为目标来生成模式.目标为"数据的私密性".

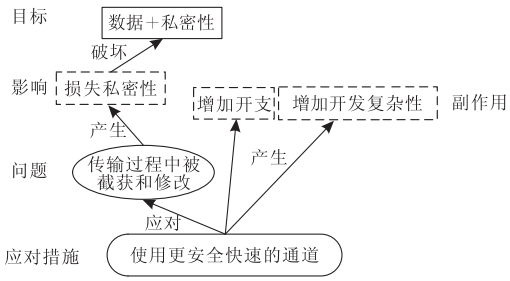
第2步,从知识库中得到威胁,包括:恶意代码,非法修改自己权限,占据两个角色的同一个人,访问控制系统被破坏,恶意输入,提供假的设备,传输过程中被截获和修改.因为数据主要在网络当中传输,所以选取威胁为"传输过程中截获和修改",指在传输的过程中被非法的截取或者是修改.而相应的影响为"损失私密性".

第3步,从知识库中得到应对措施,为:使用更加安全快速的通道,传输过程中加密和错误检测机制.我们采取"使用更加安全快速的通道",而在实际重构的时候采取的措施包括:在系统和股票交易所、系统和股票行情指示器以及系统和银行之间使用专用的通信通道,而在用户和系统之间传输采取SSL协议.

第4步,从知识库中得到的副作用为:增加开发复杂性;增加开支.

具体的模式如表6所示.

表 6　防范传输过程中截获和修改的模式

槽名	内容	实例化模式内容
Name	利用使用更安全快速的通道来防范传输过程中截获和修改的模式	
Goal	数据＋私密性	订单、交易结果、股票行情、账户数据的私密性
Issue	传输过程中被截获和修改	在传输过程中截获和修改订单、账户和交易信息
NegativeEffect	损失私密性	数据泄漏给不期望被看到的人，对用户造成巨大的经济损失
Countermeasure	使用更安全快速的通道	使用更安全快速的通道：系统和股票交易所之间使用专用通道；在系统和股票行情指示器之间使用专用通道；系统和银行之间使用专用通道；在用户的浏览器和系统之间采取 SSL 协议
SideEffect	增加开发复杂性；增加开支	开发费用会非常高；增加了开发的难度
CausalityDiagram		

5.3　人员的责任性模式

交易员是人员类型的实体.

第 1 步，对于这样类型的实体，可以从知识库得到它的可信属性，包括责任性. 责任性指的是不会否认曾参与过的动作或者声称参与过实际没有参与的动作. 以它的责任性为目标. 所以目标为“交易员的责任性”.

第 2 步，从知识库中得到威胁为：用户否认具体的操作. 所以选取威胁为用户否认具体的操作. 而相应的影响为损失责任性.

第 3 步，从知识库中得到应对措施为：用户操作的日志功能.

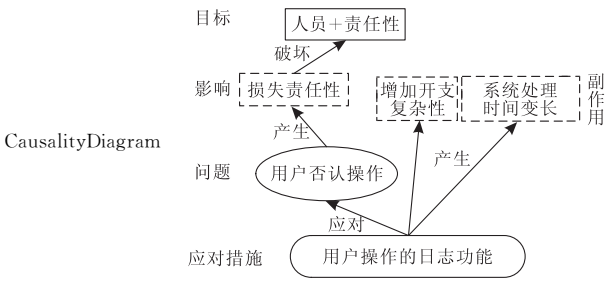
第 4 步，从知识库中得到该应对措施产生的副作用为：增加开发复杂性；系统处理时间变长.

最后得到的具体的模式如表 7 所示.

表 7　防范用户否认具体的操作的模式

槽名	内容	实例化模式内容
Name	利用用户操作的日志功能来防范用户否认具体的操作的模式	
Goal	人员＋责任性	交易员的责任性
Issue	用户否认操作	交易员否认他曾经进行过某次交易
NegativeEffect	损失责任性	对用户造成巨大的经济损失，且对于造成的损失没有人负责
Countermeasure	用户操作的日志功能	记录交易员的每一笔交易
SideEffect	增加开发复杂性；系统处理时间变长	增加了开发的难度；因为对于每一次交易都需要存储会增加系统处理时间

（续　表）

槽名	内容	实例化模式内容
CausalityDiagram		

6　相关工作

目前有许多可信需求知识源或者部分可信需求知识，例如安全性等. 卡内基・梅隆大学的软件工程实验室在信息安全方面给出利用“拥有者、应对措施、风险、资产、威胁主体、威胁”这些概念以及它们之间的关系，描述安全需求的方法. 它给出了一系列具体的安全功能需求集和安全保证需求集[3]. 微软公司针对网络程序的安全问题，给出了描述攻击的一般步骤，而且给出了网络威胁的一个分类“STRIDE”，分别是“欺骗、篡改、否认、信息曝光、拒绝服务、提高权限”，而且也根据网络程序的一般结构给出了相应的防护措施[6]. MOQUARE[7-8] 提供了一个基于误用例的方法来提取质量需求，而且它以表的形式提供关于商业目标、商业损害、资产、质量属性、威胁、误用者、缺陷、应对措施等方面的知识. 相对于以上的工作，我们所提出的可信

需求知识库更加具有系统性，为复用提供了很好的帮助.

模式也被广泛应用在需求工程阶段.模式可以分为两类，关于制品的模式和关于过程的模式.关于制品的模式包括："需求模式"，"安全模式"，"设计模式"和"分析模式"等；关于过程的模式有"基于目标的需求获取的形式化精化模式"和"非功能需求模式"等.

在关于制品的模式中，需求模式[9]提出了 37 个需求模式，涵盖了商业系统的近半数需求，它也给出了怎样写需求的细节，但是需求模式没有给出是否需要在这些模式间进行取舍以及如何在这些模式之间进行取舍.安全模式[10]是关于系统安全的模式，它主要站在系统工程的角度给出了资产识别、威胁评估、缺陷评估、风险确定等过程模式，而且也提供了 25 个关于体系结构和设计层的模式以及 7 个安全网络程序模式.设计模式[11]是关于软件设计过程中常出现的问题的通用解法，它包括 3 大类：创建模式、结构模式和行为模式.后来又提出一个新模式：同步模式[12].设计模式主要关注软件工程的设计阶段.

在关于过程的模式中，形式化精化模式[13]给出了怎么精化一个目标的模式.这种模式是用来描述如何对目标进行分解，而对于如何得到分解后的目标，仍然需要需求工程人员的知识.非功能需求模式[14]提供了 4 类模式：目标模式、问题模式、方法模式以及选择模式.用来描述怎样提取非功能需求的过程，同样缺少对分解过程所需知识的支持.关于过程的模式将需求分析的过程利用模式来表示，但是这个过程所需要的关于软件领域的知识都依赖于需求工程人员自身的知识储备.

关于提取可信需求方面，SQUARE 方法[4]提供了一个系统的步骤来提取安全质量方面的需求：统一定义，辨识安全目标，开发一些支持安全需求定义的制品，风险评估，选择提取技术，分类需求，需求优先级排序，需求审查.它主要关注安全方面的需求，而且在开发安全定义制品的时候会对一些安全定义制品（如系统结构图、用例、误用例、攻击树等）进行评估来决定究竟采用什么制品，在提取需求的步骤会对不同的提取方法进行比较来决定使用什么提取方法.但是对于提取安全需求时所需的知识仍是依赖于需求工程师自己的经验.MOQUARE[7-8]方法提供一个基于误用例的方法来提取质量需求，同时它也利用知识来帮助提取需求.Secure Tropos[15]在 Tropos 的基础上面添加了一些安全相关的概念，同样在需求提取过程仍需要需求工程师自身的经验.

7 结束语

软件系统的可信性已经成为一个受到广泛关注的焦点问题.开发可信的软件系统需要在需求阶段就提取出软件的可信性需求.提取可信性需求需要需求工程师掌握关于软件系统所面临的不期望的情况以及所采取的应对措施的知识.为了帮助需求工程师系统地分析软件可能面临的威胁，并确定有效的应对策略，本文构建了一个可信需求知识库，给出了一个可信需求模式框架，并提出了一种基于知识的方法来逐步引导需求工程师选择有效的可信需求模式.

下一步工作主要包括两个方面：一方面继续丰富和完善知识库的内容；另一方面在更多实际案例中检测本文方法的适用性.

参 考 文 献

[1] Howard J, Longstaff M. A common language for computer security incidents. Sandia National Laboratories, Sandia Report: SAND98-8667, 1998

[2] Tsipenyuk K, Chess B, Mcgraw G. Seven pernicious kingdoms: A taxonomy of software security errors. IEEE Security and Privacy, 2005, 3(6): 81-84

[3] Common Criteria for Information Technology Security Evaluation, Part 2: Security Functional Components. Version 3.1 Revision 2, 2007

[4] Mead N, Hough E, Stehney T. Security quality requirements engineering (SQUARE) methodology//Proceedings of the 2005 Workshop on Software Engineering for Secure Systems-Building Trustworthy Applications. New York, USA, 2005: 1-7

[5] Zhang Wei, Mei Hong. A feature-oriented domain model and its modeling process. Journal of Software, 2003, 14(8): 1345-1356(in Chinese)
(张伟，梅宏. 一种面向特征的领域模型及其建模过程. 软件学报，2003，14(8)：1345-1356)

[6] Meier J, Mackman A, Vasireddy S, Dunner M, Escamilla R, Murukan A. Improving Web Application Security, Threats and Countermeasures. USA: Microsoft Press, 2003

[7] Herrmann A, Paech B. MOQARE: Misuse-oriented quality requirements engineering. Requirements Engineering, 2008, 13(1), 73-86.

[8] Herrmann A, Paech B. Software quality by misuse analysis. University of Heidelberg, Technical Report SWEHD-TR-2005-01

[9] Withall S. Software Requirement Patterns (Best Practices). Washington: Microsoft Press, 2007

[10] Schumacher M, Fernandez-BuGlioni E, Hybertson D, Buschmann F, Sommerlad P. Security Patterns: Integrating Security and Systems Engineering. USA: Wiley, 2006.

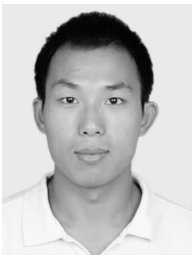
[11] Gamma E, Helm R, Johnson R, Vlissides J. Design Patterns: Elements of Reusable Object-Oriented software. New Jersey: Addison-Wesley, 1994

[12] Schmidt D, Stal M, Rohnert H, Buschmann F. Pattern-Oriented Software Architecture Volume 2: Patterns for Concurrent and Networked Objects. USA: Wiley, 2000

[13] Darimont R, Lamsweerde A. Formal refinement patterns for goal-driven requirements elaboration. ACM SIGSOFT Software Engineering Notes, 1996, 21(6): 190

[14] Supakkul S, Hill T, Chung L, Tun T, Leite J. An NFR pattern approach to dealing with NFRs//Proceedings of the 18th IEEE International Requirements Engineering Conference. Sydney, Australia, 2010: 179-188

[15] Mouratidis H, Giorgini P. Secure tropos: A security-oriented extension of the tropos methodology. International Journal of Software Engineering and Knowledge Engineering, 2007, 17(2): 285-309



WANG Yue, born in 1983, M. S. candidate. His research interests focus on knowledge based requirements engineering.

LIU Chun, born in 1982, Ph. D. candidate. His research interests focus on requirements engineering.

ZHANG Wei, born in 1978, Ph. D., lecturer. His research interests include software reuse, requirements engineering.

JIN Zhi, born in 1962, Ph. D., professor. Her research interests include requirements engineering, knowledge engineering and software engineering.

Background

This work is supported financially by the National Basic Research Program (973 Program) of China (grant No. 2009CB320701) and the Key Projects of National Natural Science Foundation of China (grant Nos. 90818026, 60736015).

Requirements engineering is a critical step for the software engineering. But as the software's environment is becoming more dynamic and open, and meanwhile software is used more widely and deeply, trustworthiness become one important property of software. But the elicitation of trustworthiness requirements needs the requirements engineer to know the knowledge about the threats that could hurt the software's trustworthiness, and the methods they could take to deal with them. And this may be difficult for them. In some research or practice, there are some knowledge sources about some trustworthiness properties for example, security. But these knowledge sources are scattering and are not computer readable. It is different to reuse.

In this paper, we suggest a trustworthiness requirement meta-model, and found a knowledge base based on this meta-model. We also introduce a trustworthiness requirements pattern template, and a method that could generate pattern from the knowledge base. And this provides a convenient way to reuse the knowledge and the pattern provide a tool to help eliciting trustworthiness requirements.