基于 MUC 的安全攸关系统危害识别方法①

霍方方, 姚淑珍

(北京航空航天大学 计算机学院, 北京 100191)

摘 要: 安全攸关系统在工业领域应用广泛, 其设计的首要任务是识别系统危害, 而已有的危害识别方法在表达方式、创造力等方面存在局限性. 本文基于误用例(Misuse case, 简称 MUC)提出了一种易用的系统危害识别方法, 这种方法有助于系统工程师在系统需求阶段获取系统潜在危害. 首先, 根据系统功能得到 MUC, 然后,使用文本误用例(Textual Misuse case,简称 TMUC)模板对每个 MUC 进行危害分析, 最终,获得系统的潜在危害.

关键词: 安全; 危害; MUC; TMUC

Hazard Identification Method of Safety Critical System Based on MUC

HUO Fang-Fang, YAO Shu-Zhen

(School of Computer Science and Engineering, Beihang University, Beijing 100191, China)

Abstract: The Safety-critical systems are widely used in industry. The first task of Safety-critical system design is to identify system hazards. However, the commonly used methods of system potential hazards identification have limitations in expression, creativity, etc. Based on the Misuse case (MUC), this article puts forward a kind of method of system hazard identification, which is easy to use. This approach helps system engineers to acquire system potential hazards in system requirements phase. First of all, you must acquire MUC based on the system function. Then, the textual misuse cases (TMUC) template is used to analysis the hazard of each MUC. Finally, the potential hazards in the system are obtained.

Key words: safety; hazard; MUC; TMUC

近年来, 电子技术和计算机的快速发展, 系统结构也变得越来越复杂, 系统的安全性问题得到了更多重视. 安全攸关系统是指错误的功能或失效会导致人员伤亡、财产损失、环境破坏等严重后果的系统[1]. 在应用领域有很多著名的例子, 如医疗设备系统, 飞机控制系统, 武器系统等. 而安全攸关系统设计的重点是识别系统中的潜在危害.

危害是一种能够导致人的生命损伤或者系统损失的条件, 这种条件是系统状态和外界环境的结合[2]. 危害识别是指对在生产活动、服务过程中所有对健康、安全、环境造成影响的因素进行识别[3]. IEC61508[4]定义了危害的四种严重级别: 灾难性的、关键的、一般的、可忽略的. 安全工程师必须注意系统灾难性的危害,并识别这类危害.

工业生产中, 系统发生危害的原因之一在于开发阶段的前期对安全性问题关注太少. 为了提高开发过程的效率, 安全性分析应该在系统开发阶段的前期进行. 危害识别是安全性分析的核心内容, 因此, 危害的识别也应该在系统开发过程的起始阶段进行[5]. 然而, 系统开发者并不熟悉系统安全分析过程, 系统工程与安全工程在方法、符号和工具方面存在差距, 安全性分析过程一般由安全工程师负责, 大部分系统需求工程师无法简单快捷的获取安全性需求. 传统安全性分析方法(如 FMEA, PHA, HAZOP)大多使用工作表的表达方式分析系统的潜在危害, 其可视性差, 并且这些方法过多的依靠于专家的经验.

本文从系统建模的角度出发, 以UML用例为基础, 基于 MUC[5]研究一种系统危害识别方法. 这种方法可

① 收稿时间:2015-08-08;收到修改稿时间:2015-09-17

以帮助系统工程师在系统需求阶段捕获和精确表达系统的安全性需求, 获取系统潜在危害, 并以可视化的方式展示系统的安全性需求, 为系统工程师提供简单易用的危害识别方法.

1 相关工作

安全性分析, 也称危害分析, 它是安全性工程的核心内容. 系统安全工程师常用的安全性分析方法有失效模式与影响分析(FMEA)和危险与可操作分析(HAZOP)等, 这些危害识别方法的弊端在于它们使用表格的表达方式, 可视化程度低, 并且过多的依靠于安全专家的经验.

作为用例的扩展, 误用例 (Misuse case, MUC)的提出最初是为了抽取安全性需求, 后来 MUC 用来发现系统中的潜在危害[6]. 与传统的安全性分析方法相比, MUC 在分析与用户交互相关的失效模式方面优于 FMEA. MUC 基于图表的表达方式, 可视化程度高, 在减少疑惑, 表达方式上更有优势[7]. 文献[8]中将 MUC 与一些形式化分析方法作对比, 发现 MUC 在与用户沟通、发挥分析人员创造力、可追踪性等方面具有明显优势. 正如文献[9]中提到的传统的危害分析方法存在诸多限制, 因此有理由相信 MUC 能够提供一些有价值的安全信息, 至少可以作为已经存在的技术的补充.

文献[10]通过实验将图表与文本表达形式进行对比, 并公开宣布图形表达方式的诸多优势. 文献[11]在系统危害识别实验中对比文本用例(TUC)和图形误用例(MUC), 得出的结论是文本用例分析能够包含更多危害信息. 因此, 本文将基于图形和文本两种方法, 利用它们各自的优势, 提出一种在需求阶段用图形和文本误用例识别系统潜在危害的方法.

2 MUC和TMUC实现

2.1 MUC

在需求工程中, 用例图适于描述系统的功能性需求, 但它并不支持非功能需求(如安全性需求)的描述[12].

Sindre & Opadah 最早提出 MUC 的概念[13], MUC 是用例面向安全性领域的扩展. MUC 是一种特殊的用例, 它描述系统所有者不希望系统发生的行为. 但它具有普通用例所具有的关系属性,例如, “包含”、“扩展”、“泛化”等. 误用例角色(Mis-actor)是一种特殊的角色, 它可以发起误用例(MUC). 安全需求场景中误用例角色不一定是人, 误用例角色可以是失效的设备或者是外界不可控因素(如恶劣天气)[14]. MUC 这种直接的表达形式能够帮助安全领域专家简单正确地分析系统危害.

用例和 MUC 的关系如图 1 所示.

(1) MUC 威胁着用例. 系统的误用例角色做出对系统有危害的行为, 即误用例角色发起误用例, MUC 又威胁着用例.

(2) 用例缓解 MUC. 角色发起的用例可以是为了缓解 MUC 的一个解决措施.

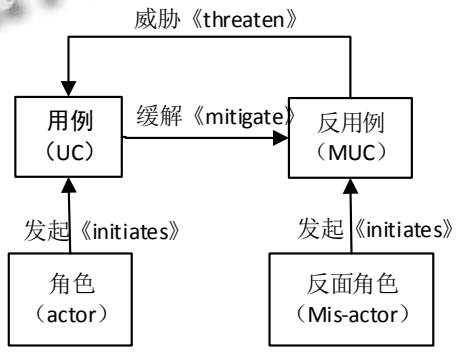

图 1 用例和 MUC 的关系图

2.2 TMUC

TMUC(文本误用例)能够获取更详细的 MUC 描述, TMUC 模板详细刻画了 MUC 的信息, 主要包括 MUC 的名称、一般描述、前置条件、基本路径、可替代路径、扩展点等[15].

3 基于MUC和TMUC获取系统潜在危害

用例图描述的是系统的主要功能, MUC 主要用来发现对系统构成威胁的用例. MUC 用图形的方式描述系统的安全性需求, 但 MUC 比较局限于高级安全性需求, 用于分析系统主要功能面临的危害, 无法分析系统内部潜在的危害. 而 TMUC 的优势在于发现系统内在详细的危害. 因此, 本文将重新定义 TMUC 模板, 并将 TMUC 与 MUC 相结合, 发现系统中潜在的危害.

文献[15]中提出的“TMUC 模板”, 仅仅是对误用例详细信息的描述, 无法用于识别系统危害. 本文根据识别危害的需要, 定义出可以识别危害的表格模板, 主要包括用户行为、系统响应、潜在危害、解决措施

四列, 如表 1 所示. 本文这样设计 TMUC 的目的是帮助 MUC 找到系统内部隐含的危害. 误用例角色发出破坏系统的行为后, 系统本身和系统用户都会做出相应的响应动作. 而在用户行为和系统响应的过程中, 又可能发出威胁系统的动作. 这些危害动作以及对应的预防措施便是系统的低级安全性需求. 如果将每一个 MUC 都用 TMUC 进行分析, 并对得到的系统危害进行汇总, 这对于系统工程师和安全工程师分析设计安全攸关系统意义重大.

表 1 TMUC 模板

MUC 名称			
用户行为	系统响应	潜在危害	解决措施

通过 MUC 和 TMUC 获取系统危害的具体方法分为如下步骤:

(1)识别出系统中的用例和 MUC

根据使用系统的角色以及系统的功能, 找出系统中的正面角色及用例. 通过分析对系统安全构成威胁的误用例角色(Mis-actor)以及它们的行为, 找出系统中的 MUC.

识别安全性需求是一个复杂的任务. 为了充分利用 MUC 技术, 规定使用 MUC 必须满足以下前置条件:
a)首先必须找到系统最小的用例集合;
b)为了不对识别潜在 MUC 产生约束, 需要假设系统暂时并不存在安全性需求.

(2)使用 TMUC 获取系统潜在危害

根据步骤 1 中的结果, 对每一个 MUC 使用定义的 TMUC(主要包含用户行为、系统响应、潜在危害和解决措施)进行分析, 找到每一个 MUC 潜在的危害.

与 MUC 类似, 规定 TMUC 应用前必须满足以下前置条件:

a)首先必须找到系统最小的用例集合;

b)为了不对识别潜在 MUC 产生约束, 需要假设系统暂时并不存在安全性需求.

c)将 MUC 作为 TMUC 的输入, TMUC 可以从用户和系统对 MUC 的响应细节中识别系统的危害.

4 实例

本文采用锅炉供热系统案例说明论文中用到的技术. 锅炉供热系统可以自动控制舱内水位, 并为工业生产提供蒸汽. 该自动系统包含两个阀门, 一个增加阀门用来控制向锅炉内的水槽加水, 一个减少阀门用来控制清除水槽内的水. 如果锅炉内的压力过大, 超过操作者提前设置的压力临界值后, 减少阀门应该自动打开. 操作者也可以手动清空水槽, 例如当水槽内压力过大时, 或者手动加水过程中, 自动判断水位失效, 减少阀门无法工作时.

本案例使用 MUC 和 TMUC, 期望达到两个目标:

(1)识别锅炉供热系统的 MUC

(2)识别锅炉供热系统中的潜在危害

下面通过 MUC 和 TMUC 找出系统潜在的危害.

3.1 识别出系统的用例和 MUC

通过对锅炉供热系统分析, 可以得到系统需要提供的功能有: 设置关键压力值、手动加水、设置温度、手动清空水槽、读取温度值、读取水位值、读取压力值.

为了识别系统中潜在的 MUC, McGraw 指出可以通过提问“此处可能有哪些人或环境导致系统出错”[16], 帮助需求工程师像专业安全工程师一样思考系统的安全性问题. 针对锅炉供热系统, 可以想到误用例角色可能包括“不合格的操作者”和“失效的系统”. 通过分析它们可能产生的危害行为, 可以得到 MUC, 如图 2 所示.

为了将用例和误用例, 角色和误用例角色区分, MUC 和误用例角色一般用黑色框图表示. “设置过高温度”威胁着一个(如设置关键压力值)或多个用例. 这种简洁的技术非常适合系统工程师发现系统高层需求中的危害.

3.2 使用 TMUC 获取系统潜在危害

TMUC 采用的文本表达形式可以帮助识别 MUC 中的潜在危害, 在识别危害时, TMUC 只关注单个 MUC, 而不是所有的用例. 因此, TMUC 避免了将所有功能聚集在一起产生的复杂性.

由于篇幅限制, 下面将仅以“设置过高压力”这个 MUC 为例, 分析当压力值过高时, 用户和系统分别做出什么动作, 并分析动作中包含的潜在危害, 以及每个危害的解决措施.

经过表 2 中逐步的分析可以看出, 仅仅一个 MUC“设置过高压力”就产生了 7 个潜在的危害, 甚至有些潜在的危害还未被发现, 这体现了 TMUC 在分析系统潜在危害方法的可用性. 如果将所有 MUC 都用 TMUC 分析, 并去掉最后重复的危害, 最后汇总出的潜在危害将对系统安全产生重大价值.

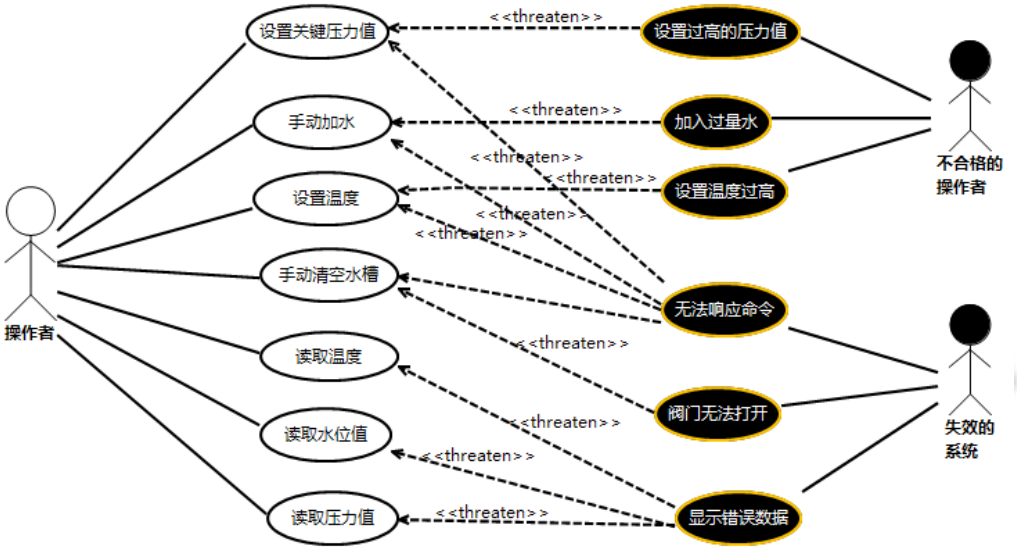

图 2 锅炉供热系统 MUC 图

表 2 分析“设置过高压力”的潜在危害

对“压力过高”的响应			
用户行为	系统响应	潜在危害	解决措施
	高压报警	系统无法报警; 操作者无法通知报警;	安装两个独立的警报; 同时使用音频和视觉线索; 在监控房间内外都安置警报;
操作者发出清空水槽的命令		操作者无法响应; (例如, 生病, 没意识到) 操作者发出错误命令; (例如, 加水)	通知备用的操作员; 自动智能检测, 在高压下不允许加水;
	系统打开出水阀门	系统无法响应命令; 阀门生锈, 无法打开;	操作人员定期检查; 去锈或更换新的阀门;
操作者读取压力值		操作者误读并过早停止清空;	保持报警状态直到恢复正常;

5 结语

本文的主要思路是使用 MUC 技术获取高层次的系统危害(即, MUC). 然后, 针对找到的每一个 MUC, 采用 TMUC 技术找到每个 MUC 内部潜在的危害, 即低层次的系统危害. 本文的价值在于提供了一种易用的系统危害识别方法——采用 MUC 和 TMUC 技术, 打破了只能利用传统的安全分析方法(如 FMEA、HAZOP)识别危害的局限性. 实验证明, 本文提出的 TMUC 模板, 能帮助系统工程师发现每个 MUC 内部的潜在危害, 并最终确定整个系统的潜在危害.

本文最终的输出是系统潜在的危害以及每种危害对应的解决措施. 需求阶段的危害识别确定以后, 我们将考虑设计阶段的安全性问题. 后续工作会把需求阶段得到的危害和解决措施用于系统安全性建模过程中, 进一步从设计阶段解决安全性问题.

参考文献

1 Knight J. Safety critical systems: Challenges and directions. Proc. of the International Conference on Software Engineering. Orlando, FL, US. 2002.

2 Thramboulidis K, Scholz S. Integrating the 3+1 SysML view model with safety engineering. IEEE International Conference on Emerging Technology and Factory Automation (ETFA 2010). Bilbao, Spain. 2010.

3 彭洪军.危害识别与风险评价方法探讨.商业时代,2011(16): 81–82.

4 IEC 61508. Functional Safety of Electrical/Electronic/ Programmable Electronic Safety-related Systems-Part 4:

Definitions and Abbreviations. 2010.

5 Tndel IA, Jensen J, Rstad L. Combining misuse cases with attack trees and security activity models. ARES'2010. IEEE Press. 2010. 438–445.

6 Stålhane T, Sindre G. A comparison of two approaches to safety analysis based on use cases. Submitted to ER'07, 2007.

7 Sindre G, Opdahl AL. Eliciting security requirements by misuse cases. Proc. 37th Conf. Techniques of Object-Oriented Languages and Systems, TOOLS Pacific 2000. 2000. 120–131.

8 Sindre G, Opdahl AL. Eliciting security requirements with misuse cases. Requirements Engineering Journal, 2005, 10(1): 34–44.

9 Matulevi˘cius R, Mayer N, Heymans P. Alignment of misuse cases with security risk management. Proc. of the ARES 2008 Symposium on Requirements Engineering for Information Security (SREIS 2008), IEEE Computer Society. Los Alamitos. 2008. 1397–1404.

10 Cheng PCH. Why diagrams are (sometimes) six times easier than words: Benefits beyond locational indexing. In: Blackwell AF, Marriott K, Shimojima A. eds. Diagrams 2004. LNCS (LNAI). Springer, Heidelberg. 2004, 2980: 242–254.

11 Larkin JH, Simon HA. Why a diagram is (sometimes) worth ten thousand words. Cognitive Science, 1987, 11.

12 Stalhane T, Sindre G. Safety hazard identification by misuse cases: experimental comparison of text and diagrams. Models, 2008: 721–735.

13 Sindre G, Opdahl AL. Capturing security requirements through misuse cases. NIK 2001, Norsk Informatikkonferanse 2001. http://www.nik.no/2001.

14 Alexander I. Misuse cases: Use cases with hostile intent. Software, IEEE 20.1. 2003: 58-66.

15 Sindre G, Opdahl AL. Template for misuse case description. Proc. of the International Workshop Requirements Engineering: Foundation for Software Quality (REFSQ 2001), 2001.

16 McGraw G. Software Security- Building Security In. Boston: Pearson Education, 2006.