全生命周期软件过程风险管理模型研究

张子剑[1,3] 曹 静[2] 张 丽[4] 饶国政[2]
(天津大学软件学院 天津 300072)[1] (天津大学计算机科学与技术学院 天津 300072)[2]
(天津航海仪器研究所 天津 300131)[3] (天津科技大学经济与管理学院 天津 300222)[4]

摘 要 首先从软件过程的角度重新定义了过程风险的基本概念;然后结合CMMI和软件的生命周期分别从软件过程、生命周期和风险管理构建了三维结构的软件项目风险管理模型,提出了基于CMMI的全生命周期的过程风险策划、识别、评估和监控全过程模型;最后建立了软件研制风险管理系统,该系统在实际应用中取得了较好的效果。

关键词 软件风险管理,CMMI,软件过程,全生命周期
中图法分类号 TP311.5 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2014.11.031

Research on Software Process Life Cycle Risk Management Model

ZHANG Zi-jian[1,3] CAO Jing[2] ZHANG Li[4] RAO Guo-zheng[2]
(School of Software, Tianjin University, Tianjin 300072, China)[1]
(School of Computer Science and Technology, Tianjin University, Tianjin 300072, China)[2]
(Tianjin Navigation Instrument Research Institute, Tianjin 300131, China)[3]
(School of Economics and Management, Tianjin University of Science and Technology, Tianjin 300222, China)[4]

Abstract Firstly, the software process risk was defined from the perspective of the software process based on the basic risk concept. Secondly, the three-dimensional software project risk management model based on software process, life cycle and risk management was proposed combined with the full life cycle and CMMI. The full life cycle risk identification, assessment and monitoring model based on CMMI was presented. Finally, a software development risk management system was implemented, which achieves good practice results.

Keywords Software risk management, CMMI, Software process, Life cycle

1 引言

风险管理是指辨识、分析和控制风险的活动,这组活动不是孤立的,而是一组系统化、持续化的过程[1]。软件项目风险管理是指贯穿于软件项目生命周期,保证项目按计划进行的策略、方法、技术和工具的集合,它含有风险辨识、评估、排序、计划、监督和控制活动,并成为软件项目管理的主要部分[2]。

从软件项目风险管理的发展历史看,Boehm于1989年出版的专著《软件风险管理》[3],奠定了该领域的理论基础。在随后近30年中,又陆续出现了几种框架体系:

1)Boehm模型:Boehm思想的核心是10大风险因素列表,针对每个风险因素,Boehm都给出了一系列的风险管理策略。在实际操作时,以10大风险列表为依据,总结当前项目具体的风险因素,评估后进行计划和实施,在下一次定期召开的会议上再对这10大风险因素的解决情况进行总结,产生新的10大风险因素表,依此类推。

2)Charette的风险管理框架[4]:直接将其分为风险分析和风险管理两部分,其中风险分析包括识别、估算和评价,风险管理包括计划、控制和监控。

3)CRM模型:它是Higuera和Haimes提出的软件项目风险管理模型,该模型将风险管理划分为风险识别、分析、计划、跟踪、控制5个步骤,风险管理的方式是连续循环的,其核心是风险沟通。它要求在项目生命期的所有阶段都关注风险管理,即所谓的持续风险管理(CRM)框架模型[5]。

4)SEI的模型:它在Boehm和Charette的模型基础上有所改进,注重了软件项目的过程特点。但这一模型只是在理论上对风险管理的过程有了初步认识,而如何把风险管理演绎成一个动态、持续的风险管理过程,其未作详细阐述。

5)Leavitt模型:SEI和Boehm的模型都以风险管理的过程为主体,研究每个步骤所需的参考信息及其操作。而Aalborg大学提出的思路则是以Leavitt模型为基础,着重从导致软件开发风险的不同角度出发探讨风险管理。

6)Hall的六学科模型:Hall的六学科风险管理模型[6],将风险管理分解为预想(E)、计划(P)、工作(W)、度量(M)、改进(I)和发现(D)6个学科,它通过对不确定性的评价和对困惑的思考,考虑机会和风险的均衡,预先指导计划和规划的改变。

到稿日期:2013-07-30 返修日期:2013-09-25 本文受国家自然科学基金(61373165),天津市高等学校人文社会科学研究项目(20082110)资助。
张子剑(1969-),男,博士,正高级工程师,主要研究方向为软件风险与软件质量,E-mail:zzj1969@yahoo.com.cn;曹 静(1967-),女,博士生,高级工程师,主要研究方向为软件测试、电子政务等;张 丽(1977-),女,博士,副教授,主要研究方向为多Agent、风险管理,E-mail:zhangli@tju.edu.cn(通信作者);饶国政(1977-),男,博士,讲师,主要研究方向为知识工程、软件工程。

7)基于 CMM/ CMMI 的软件项目风险管理框架：文献[7]提出了基于 CMMI 的软件项目风险管理框架，对软件项目风险管理理论作了进一步研究和扩展。在 CMMI 中，风险管理作为第 3 级中的一个独立的关键过程域，是软件工程管理的一个重要方面，体现了风险管理的过程特点，从而使在过程中进行风险管理的原则得以真正体现[8]。基于 CMM/ CMMI 的软件项目风险管理的研究，推动了风险管理理论与以软件过程改进为主导的软件工程实践的融合，使软件项目风险管理朝着可预测、有规律、可量化的管理方向发展。

分析以上模型发现，研究者建立的模型都多少受到 Boehm 模型的影响，都是从风险因素的识别开始的[9]，而对软件过程风险并不太关注，尽管随着 CMMI 的出现，研究者更加关注软件的过程，但多过程风险并没有一个完善的模型，其实每个软件过程中的风险是完全不相同的，软件开发过程中每项实践活动都是可能导致风险的因素。

目前随着 CMMI 在企业广泛实施[10]，组织和机构不断探索如何实施软件过程改进[11]，针对不同的目的应该采取怎样的策略，已经有越来越多的企业逐渐积累了企业的最佳实践，在这种情况下，风险发生的概率变得可预期和可以进行改进，而实际上一个具体的软件开发过程在组织和机构总结的最佳实践指导下风险概率也会出现偏离，偏离会有正偏离，即在软件开发周期中的某个节点，软件风险发生的概率高于最佳实践条件下的软件风险发生概率，另一方面也会产生负偏离，这一般是由于持续不断的过程改进使得风险发生的概率降低。

基于此，本文给出软件的过程风险的定义。

定义 1(软件过程风险) 在软件生产过程中，我们将偏离最佳实践从而导致软件生成过程中产生的风险称为软件过程风险。

不同于 CMMI 中的软件风险的定义，它将过程客观存在的不确定性识别为风险因素，软件过程风险内涵如下：

1)软件过程风险是指软件开发过程中客观存在偏离最佳实践的不确定性；

2)这种不确定性对主体的决策和价值目标构成了潜在威胁或可能造成损失；

3)不同主体对同样风险的承受能力与收益大小、投入多少、项目活动的主体地位和拥有的资源有关。

在不改变风险以上内涵的情况下，随着 CMMI 的广泛实施，软件开发组织过程能力不断提升，但是每个软件的开发过程是不尽相同的，过程改进得到软件开发的最佳实践在每个软件开发实施过程中会出现偏差，这给实施 CMMI 企业的软件开发过程带来了客观存在的不确定性，因此我们给出软件过程的风险内涵如下：

1)软件过程风险是指软件开发过程中客观存在偏离最佳实践的不确定性；

2)这种不确定性对主体的决策和价值目标构成了潜在威胁或可能造成损失；

3)不同主体对同样风险的承受能力与收益大小、投入多少、项目活动的主体地位和拥有的资源有关。

2 全生命周期软件过程的风险管理和控制模型

本文构建了一个三维(过程维、时间维和逻辑维)模型，来管理软件的风险，如图 1 所示。

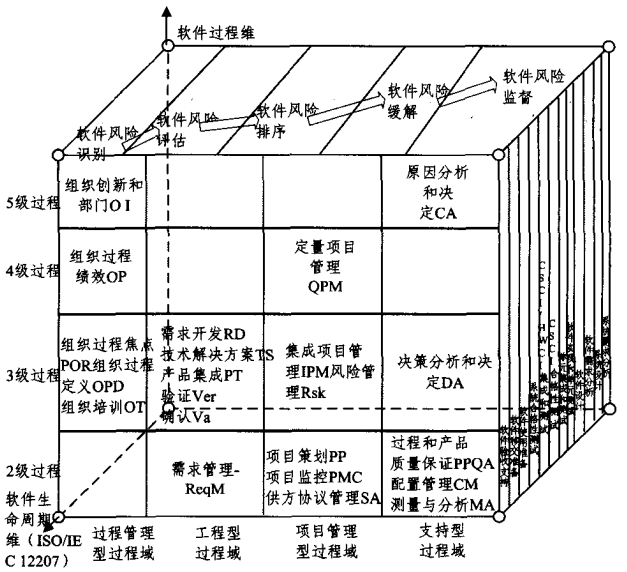

图 1 软件的全生命周期的软件过程风险管理和控制关系模型

2.1 软件过程维

从项目管理角度讲，风险识别依据有：合同、项目计划、工作任务分解 WBS、各种历史参考资料(类似项目的资料)、项目的各种假设前提条件和约束条件。从软件开发的生命周期看，每个阶段的输出(各种文档)都是下一阶段进行风险识别的依据，许多技术风险都可据此来分析。

本项目的面向过程的风险管理的模型是建立在 CMMI 的过程域模型之上，软件风险识别依据是过程域，CMMI 标准中的过程域的组成模型部件如图所示，它是一个领域内的一组相关的实践，若这些实践中的风险被全部处于控制之下，就能达到对该过程域风险进行有效管理的目的。因此关注过程风险就是要关注过程域中每个子实践的风险。

这些子实践中，公用实践的风险可以看成是每个过程域的通用问题，统一进行考虑，而专用实践则需要为每个专用实践建立风险清单。

风险识别过程的活动是将项目实施中的不确定性转变为明确的风险陈述。系统地识别风险是这个过程的关键，识别风险不仅要确定风险来源，还要确定何时发生、风险产生的条件，并描述过程风险特征和确定哪些风险事件有可能影响本项目。本项目风险识别是面向过程的，以过程域、关键实践以及子实践为依据，并以面向软件过程的度量指标作为依据，将一、二、三级指标作为风险的基础度量指标。风险识别不是一次性的活动，应当在项目执行过程中自始至终定期进行。

2.2 逻辑维——项目风险管理维

逻辑维是指分析问题解决问题的逻辑思维过程。软件项目面临众多错综复杂的风险因素，风险层次、各种风险之间的相互关系错综复杂，造成了项目风险的客观性和多样性。而为控制项目风险所采取的任何措施都会给项目带来正反两方面的影响：1)风险控制措施能降低风险对项目成本和工期的影响；2)风险控制行动需要付出额外的费用和时间，这些影响不仅取决于管理者是否采取行动，还与采取风险控制措施的强度以及在什么时点采取措施有关，因此软件项目的风险管理应当以项目的总成本和工期最小化为目标，根据对风险因

素的分析并结合主体的风险接受准则来决定是否采取以及何时采取某项风险控制措施，只要将风险控制在可接受和合理的范围即可，而不是盲目地采取措施控制项目中的所有风险。

软件风险管理则指对软件项目进行风险管理时一般采用的工作步骤。如前所述在本项目中我们采用通用的风险管理5步骤：风险识别、风险评价、风险排序、风险缓解和风险监督。

2.3 时间维——软件项目开发过程基本活动

时间维度选取ISO/IEC 18019中软件开发过程中的基本活动，并按软件生命周期时间先后顺序来组织这些活动。

全生命周期的软件过程风险管理的三维结构模型旨在增进风险应对决策的能力；降低意外事件的发生和由此造成的成本或损失；识别和管理贯穿于软件项目全过程的风险；提供对多重风险的整体应对，从而在对项目总体需求的有效评估中改善资本调配能力。

以软件项目全生命周期集成的三维体系，要求根据项目的特点，以时间维划分不同阶段，在各生命周期阶段，按照逻辑维顺序，结合阶段特点用知识维对项目风险进行集成。

2.4 软件风险分析的流程

通常项目计划人员与管理人员、技术人员一起，进行风险分析，该过程是一个不断重复的过程。为了有计划、有规律地进行风险分析，本文给出分析流程，如图2所示，具体活动内容如下：

A. 风险识别依据。在本项目中，风险识别依据是软件生命周期各阶段相关的过程域所相关的子实践。在风险管理中，也必须对每个相关的子实践的风险进行识别。在这一阶段，要将评价的过程域分解为共用实践、专用实践以及相关的子实践等若干可以评价的层次。

B. 风险识别。软件项目风险管理中的一个共同的问题是要识别出各类风险来源及其影响程度。在这一阶段，系统各风险因素应该尽可能被识别出来。识别风险要从项目全生命周期各阶段，使用多种识别方法以及方法间的组合去实现，并分析这些风险是由哪些具体风险源构成的，以及风险源之间的关系。

C. 风险评价。风险评价是指用具体的方法评价具体的风险，能否有效降低不确定性是指导这一过程的主要标准。在风险评价时，首先应根据风险发生的概率和对项目的影响程度，对每一风险做出非正式的估测，而项目内在不确定性的概率和影响程度也可以根据经验和相似项目或项目单元参照获得。其次是分析各阶段风险的特点，建立其风险评价指标体系并识别出适用的评价方法，包括非正式的猜测、正式的数理模型和实践经验。

D. 风险排序。风险整体集成是指为获得各风险源的概括认识对风险进行的适当的整合，即综合评价这些风险对项目的总体影响，根据帕累托80%原理，项目所有风险中只有小部分对项目威胁较大。因此，需要对风险进行排序，集中力量对威胁较大的几个风险加强管理。

F. 风险缓解。即制定并实施控制风险的计划，确定降低风险发生的可能性并减少其不良影响的方法。风险应对计划是针对风险量化结果，为降低项目风险的负面效应制定风险应对策略和技术手段的过程。

G. 风险监督。风险监督就是在风险发生时实行原先制定的风险处置措施，并根据项目环境的变化情况，适度修改或重新采取新的风险处置策略。除跟踪已识别风险外，还要对剩余风险进行监控，一旦出现症状就应该立即修改风险应对计划，保证项目正常进行。

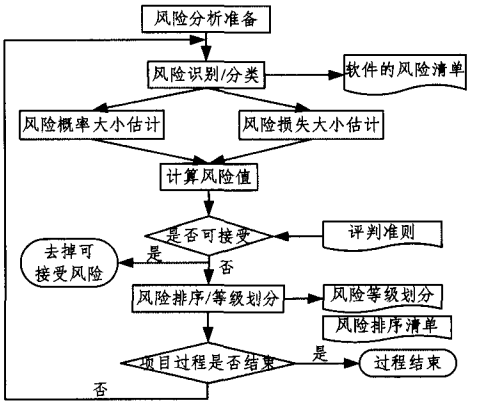

图2 软件的风险分析流程

3 软件风险因素分析体系及裁剪

根据前述三维模型，将三维体系进行映射，如表1所列（由于篇幅所限，软件生命周期及软件过程域等没有全部列出），首先建立了软件的生命周期及软件开发文档要求和CMMI的关系映射表，然后给出每个过程域的主要活动和活动的输出，根据定义1，通过对每个过程域中的活动和输出与最佳实践进行比较，就能识别过程中的活动及其输出是否出现偏离，通过对这些偏离进行评估，就能实现对软件过程风险的识别、评价、排序、缓解和监督。

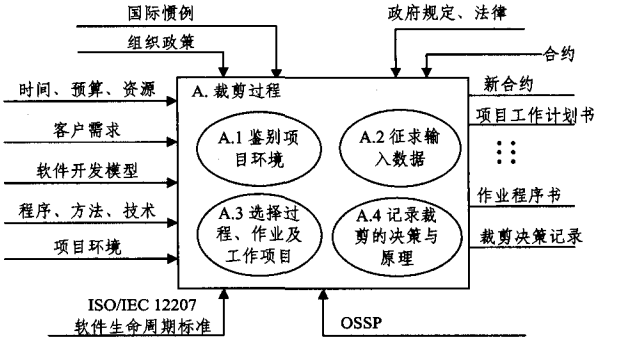

图3 风险因素裁剪

如图3所示，在制定备件软件组织的标准过程时，首先应当参照通用软件过程的通用模型及国际、国内标准，然后根据软件研发软件的规定需求（行业、规模和类型等）、组织规模和组织类型（中间产品供应商、产品供应商和软件服务提供商等），选择合适的软件过程标准并进行剪裁，以建立适合自己组织的软件过程风险因素。同时，软件组织有时可能会被要求强制遵守一些特定的标准，这时就必须要以相应的强制标准为基础并在其上制定组织的标准过程。同样，项目经理也要以组织的标准过程为基础并参照相关裁剪原则和标准来制定项目的过程。

表1　基于三维模型的风险因素分析体系

活动类型	软件开发生命周期 ISO/IEC 12207	软件开发文档要求 ISO/IEC 18019	CMMI 关键过程域													
			工程型过程域		项目管理型过程域						支持型过程域					
			需求管理		项目策划		项目监控		供方协议管理		过程和产品质量保证		配置管理		测量和分析	
			主要活动	活动的输出	主要活动	活动的输出	主要活动	活动的输出	主要活动	活动的输出	主要活动	活动的输出	主要活动	活动的输出	主要活动	活动的输出
软件开发的基本活动（软件生命周期基本过程）	系统需求分析	a)运行方案说明 b)系统/子系统规格说明 c)接口需求规格说明	分配需求确认	分配需求文档，分配需求评审报告	软件项目早期策划项目建议书	项目开发计划，项目对外诺声明文件										
	系统设计	…														
	…	…														
	软件验收支持	…														
软件开发的支持活动	软件配置管理	h)软件配置管理计划											创建和维护项目SCM计划	软件配置管理计划		
	…															
软件开发管理活动	项目策划和监控	g)软件开发计划 j)软件安装计划 k)软件移交计划 l)软件测试计划			策划跟踪和测量	软件测量计划	策划跟踪和测量	软件测量计划								
	…															

通过裁剪就能在一个通用的风险因素模板中，快速定义每个具体软件的过程风险因素，并在过程中实现对软件过程活动风险的监控。

4　模型实现及应用实践

天津航海仪器研究所隶属于中国船舶重工集团公司，是国内较早实施 CMMI 的装备软件制造企业之一，承担了大量的船舶装备软件的研发，企业在进行风险管理过程中发现风险管理成本非常高，随着装备软件规模越来越大、复杂性越来越高，风险管理的难度也在不断增大。针对这一问题，首先应当参照通用软件过程的通用模型及国际、国内标准和军用软件的相关标准（如 GJB5000A-2008《军用软件研制能力成熟度模型》、GJB438B-2009《军用软件开发文档通用要求》、GJB 2786A-2009《军用软件开发通用要求》、HJB 111-1994《舰船指挥控制系统应用软件测试》HJB 221-2000《舰艇作战指挥系统维修性规范》和 HJB 103-1993《海军指挥自动化综合标准化工作指南》等）的内容，根据三维模型建立如图 4 所示的软件过程风险管理系统。该系统可以从配置管理系统中导入相应的软件生产的各项数据并接收软件过程改进小组的数据，在不增加更多成本的条件下实现了对软件过程风险的管理，软件的生成质量得到了较好的保证，而且风险管理难度也大幅度下降。

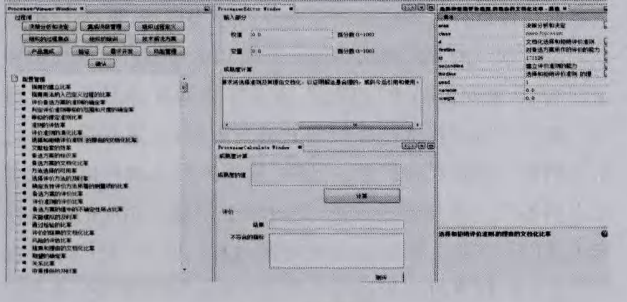

图4　系统界面

结束语　本文首先从软件过程的角度重新定义了过程风险的基本概念，并结合软件的全生命周期构建了三维结构的软件项目风险管理体系，提出基于 CMMI 的全生命周期的风险策划、识别、评估和监控全过程模型，并建立了软件研制过程风险因素分析体系。在天津航海仪器研究所的实践表明，有效地将 CMMI 和软件研制生命周期和软件配置文件管理进行结合，不仅降低了风险管理的难度，并且风度管理成本大幅度下降，这可为其他 CMMI 企业制定行之有效的风险管理实施方法提供有益参考。

参 考 文 献

[1] Software Engineering Institute. The SEI approach to managing software technical risks[R]. Bridge: Software Engineering Institute, 1992: 19-21

[2] Boehm B W. Software risk management: Principles and practices [J]. IEEE Software, 1991, 8(1): 32-41

[3] Boehm B W. Software risk management [M]. Piscataway: IEEE Computer Society Press, 1989

[4] Charette R. Software engineering risk analysis and management [M]. New York: McGraw-Hill, 1989

[5] Dorofee A J, Walker J A. Continuous risk management [R]. Pittsburgh: Carnegie Mellon University, 1996

[6] Hall E M. Managing risk: Methods for software systems development [M]. Addison-Wesley Publishing Company, 1998

[7] Surie D. Evaluation and integration of risk management in CMMI and ISO/IEC [OL]. 2006, http://www.cs.umu.se/~dipak/paper-cmmi.pdf

[8] Alfred B. Process-based software risk assessment[C]// Proc of the 4th European Workshop on Software Process Technology. Nordwijkerhout, 1995: 1-21

[9] Li Ming-lu, Li Jian-ping, Song Hao, et al. Risk Management in the Trustworthy Software Process: A Novel Risk and Trustworthiness Measurement Model Framework[C]// Fifth International

Joint Conference on INC, IMS and IDC, 2009(NCM'09). 2009: 214-219

[10] SEI. CMMI for Development, Version 1.2-Improving Processes for Better Products[S]. SEI, CMU, 2006

[11] Sujin C, Dae-Kyoo K, Sooyong P. ReMo: A recommendation model for software process improvement[C]// 2012 International Conference on Software and System Process (ICSSP). IEEE Press, Zurich, 2012: 135-139

(上接第 127 页)

并将其应用于异步系统(asynchronous systems)中。他们的工作使用事件结构作为形式化描述方式,该工作将控制事件和可观察事件区分开来,通过提取出的仅含有控制事件的序列来生成测试用例。该工作通过一些扩展机制,在现有测试工具基础上完成自己的功能并将其应用于真实系统中。该工作对于控制事件与可观察事件的区分与本文有相似之处,但其应用领域为纯粹的软件行业,显然不能满足高并发、高安全性要求的 CPS 系统。

在安全攸关的 CPS 系统中应用基于场景的测试技术需要充分考虑 CPS 系统与传统系统,如软件、嵌入式系统等之间的区别,突出 CPS 系统的特征以及面临的新的挑战。

CPS 系统研究的先驱 Edward A. Lee[1]将 CPS 系统与当前的嵌入式系统作了比较,他强调了 CPS 系统具有与生俱来的高并发性,时间性质在 CPS 系统中具有绝对重要的地位,不可忽略。Frank Mueller[2]指出安全问题、时间相关分析和软错误保护是 CPS 系统研究面临的 3 大挑战。Lui Sha 等[4]将实时系统的抽象建模,CPS 系统的健壮性、防危性和安全性,系统服务质量组合,将系统工程化方法研究与系统可信作为 CPS 研究的主要挑战。Stankovic 等[5]认为相关的验证与检验方法仍是 CPS 系统研究的重点,其中形式化的建模与设计方法是重要的研究方向。

本文工作参考上述文献所述的 CPS 系统的特点,在模型构造和用例生成阶段充分考虑系统的时间特性,应用路径覆盖的思想达到对场景的覆盖,保证相对充分的测试,从而提高 CPS 的安全性。

结束语 本文立足于现有的 CPS 中安全攸关场景建模与测试用例生成技术上存在的不足,提出了一种基于 UML 活动图扩展的建模语言,并提供对此类模型进行测试用例自动化生成的算法。

本文工作的意义如下:

(1)在 UML 活动图的基础上,根据安全攸关场景建模的新需求,提供新的建模支持,尤其是在时间特征方面的支持。

(2)采用对安全攸关场景模型生成测试用例的过程,来帮助系统确定受控设备的范围,系统内部、系统与外部环境相互作用可能存在风险的点。进一步提高系统的安全性。

(3)测试用例的生成过程实现了自动化,极大减少了手工创建测试用例容易造成的遗漏,同时降低了人力成本。

参 考 文 献

[1] Lee E A. Cyber Physical Systems: Design Challenges[C]// Proc. 11th IEEE International Symposium on Object-Oriented Real-Time Distributed Computing. 2008: 363-369

[2] Mueller F. Challenges for Cyber-Physical Systems: Security, Timing Analysis and Soft Error Protection[C]// National Workshop on High Confidence Software Platforms for Cyber-Physical Systems(HCSP-CPS). Nov 2006

[4] Lui Sha, Gopalakrishnan S, Liu Xue, et al. Cyber-Physical Systems: A New Frontier[C]// Proc. 2008 IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing. 2008: 1-9

[5] Lee S J A, Mok I, Rajkumar A, et al. Opportunities and Obligations for Physical Computing Systems[J]. IEEE Computer, 2005, 38(11): 23-31

[6] IEC 61508 series[OL]. http://www.iec.ch/functionalsafety

[7] Castillos, Cabrera K, Botella J. Scenario based test generation using test designer[C]// 2011 IEEE Fourth International Conference on Software Testing, Verification and Validation Workshops (ICSTW). IEEE, 2011

[8] Utting M, Legeard B. Practical Model-Based Testing-A tools approach[J]. Elsevier Science, 2006, 550

[9] Marchetti, Eda, Schilders L, et al. Scenario-based testing applied in two real contexts: Healthcare and Employability[C]// 2011 IEEE Fourth International Conference on Software Testing, Verification and Validation Workshops (ICSTW). IEEE, 2011

[10] Dadeau, Frédéric, Tissot R. jSynoPSys-a scenario-based testing tool based on the symbolic animation of B machines[J]. Electronic Notes in Theoretical Computer Science, 2009, 253(2): 117-132

[11] Wang Lin-zhang, et al. Generating test cases from UML activity diagram based on gray-box method[C]// Software Engineering Conference, 2004. 11th Asia-Pacific. IEEE, 2004

[12] Ryser J, Berner S, Glinz M. On the State of the Art in Requirements-based Validation and Test of Software[M]. Universität Zürich, Institute für Informatik, 1998

[13] Salas, Pari P, Krishnan P. Automated software testing of asynchronous systems[J]. Electronic notes in theoretical computer science, 2009, 253(2): 3-19