软件可靠性工程学综述

郑艳艳[1] 郭 伟[2] 徐仁佐[1]
(武汉大学软件工程国家重点实验室 武汉 430072)[1] (武汉大学计算机学院 武汉 430072)[2]

摘 要 软件可靠性工程学定量评价软件系统的可靠性,是开发并维护软件系统的一种工程学技术。回顾了软件可靠性工程学的历史、目前的趋势以及现存的问题和困难。提出了软件可靠性工程学中未来可能的方向和有前景的研究课题。

关键词 软件可靠性工程学,容错,建模,数据收集,度量,测试

Overview of Software Reliability Engineering

ZHENG Yan-yan[1] GUO Wei[2] XU Ren-zuo[1]
(State Key Lab of Software Engineering, Wuhan University, Wuhan 430072, China)[1]
(School of Computer, Wuhan University, Wuhan 430072, China)[2]

Abstract Software reliability engineering is focused on engineering techniques for developing and maintaining software systems whose reliability can be quantitatively evaluated. We reviewed the history of software reliability engineering, the current trends and existing problems, and specific difficulties. Possible future directions and promising research subjects in software reliability engineering were also addressed.

Keywords Software reliability engineering, Fault tolerant, Modeling, Data collecting, Metric, Testing

1 引言

早在上世纪60年代,就有人提出了"软件危机",而软件不可靠是造成软件危机的重要原因[1]。软件可靠性工程学为理解软件的可靠性,管理软件质量,保证商业产品竞争性优势给出了希望。但现今的软件可靠性工程学还没有真正达到我们所期望的目标[2]。

软件可靠性工程学的核心是软件可靠性。软件可靠性定义为:给定时间内、给定环境下软件无故障运行的概率[3,4]。软件可靠性对软件故障进行了量化,是软件质量中的主要因素。软件可靠性工程学(SRE)是对基于软件的系统操作行为的定量研究,且与用户的可靠性需求相关[5-8]。

现有的SRE技术存在一些缺点,如故障数据的收集在时间上太迟;实验室内收集的故障数据不够精确;SRE技术或建模方法基于一些不现实的假设,使得结果过于乐观。虽然多个行业都有软件可靠性模型的成功实例,但对于现代软件系统,仍然迫切需要可信的软件可靠性技术。

在下面的章节中,我们将讨论软件可靠性工程学的过去、现在和将来。首先概述一些已有的技术和应用,然后介绍目前的趋势以及所关注的问题。最后,我们提出了软件可靠性工程学将来可能的方向。

2 历史上的软件可靠性工程学技术

国内软件可靠性研究始于上世纪80年代初,徐仁佐在软件可靠性建模等方面做了有益的探索,并开发了软件可靠性专家系统SRES[9]。黄锡滋、蔡开元、何国伟等在软件可靠性理论和工程实践等方面也做了研究,包括软件可靠性分配、软件可靠性管理、软件可靠性模型应用选择、软件可靠性测试、软件可靠性度量体系等[10]。图1为软件可靠性工程过程概图。

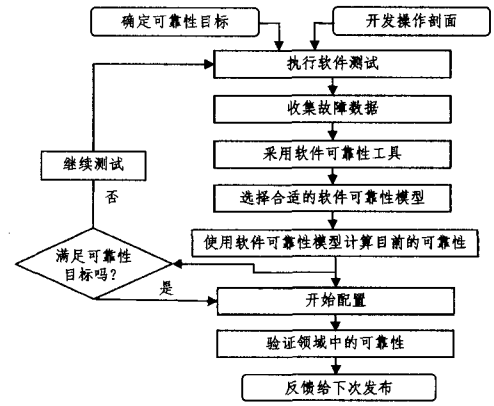

图1 软件可靠性工程过程概图

国外基于软件故障生命周期已经提出了若干技术来解决软件可靠性工程学问题。包括故障生命周期技术及软件可靠性和度量。文献[11]总结了故障生命周期的4种技术:错误预防;错误排除;错误容忍;错误/故障预测。软件可靠性建模的主要任务是错误/故障预测,即估计(度量目前状态)和预计

到稿日期:2008-03-03
郑艳艳(1975-),女,博士研究生,研究方向为软件可靠性、软件工程,E-mail:zhengyy03@shou.com;**郭 伟**(1966-),男,博士,高工,研究方向为软件工程、计算机网络;**徐仁佐**(1946-),男,教授,博士生导师,研究方向为软件可靠性、软件测试、软件工程等。

(评价将来状态)软件系统的可靠性。文献[12,13]叙述了对软件可靠性模型的历史回顾及模型应用的观点。

这里主要讨论软件容错技术和软件可靠性模型[14-16]。

容错是得到高可靠软件的主要方法之一。有两类错误容忍技术:单版本和多版本软件容错技术[17]。前者包括程序模块化、系统终止、原子行为、错误检测、意外处理、检查点和重启、过程对和数据多样性[18];后者称为设计多样性,由不同的程序团队使用不同的设计方法独立开发多个软件版本,但根据相同的需求说明,它们提供同样的服务。多版本软件方法的主要技术是恢复块、N-版本编程、N-自检编程和基于这3种基本技术的其他变种。

可靠性模型试图估计多版本中的相同故障的概率,Eckhardt和Lee[19]提出了设计多样性错误关联的第一个可靠性模型,基于需求空间难度差异的假设,来观察版本故障间的积极关联。Littlewood和Miller[20]提出,基于强制的设计多样性,可能存在消极的错误关联。Dugan和Lyu[21]提出一个马尔可夫奖励模型,来比较由设计多样性方法得到的系统可靠性。Tomek和Trivedi[22]为软件容错提出了统计学奖励网络模型。Popov,Strigini等[23]基于需求空间的子域概念,估计了设计多样性故障概率的上下限。关于容错软件的详细总结和它的可靠性建模方法能够在文献[24]中找到。经过实验对模型的比较和估计结果列在文献[25,26]中。

3 软件可靠性工程学的现状和问题

软件可靠性中的挑战不仅来自不同领域的软件应用程序的规模、复杂度、难度和新颖性,也与相关软件工程师的知识、培训、经验和特点有关[27-31]。我们从软件可靠性工程学的某些角度讨论当前趋势和问题。

3.1 软件可靠性和系统可靠性

软件可靠性的理论基础来自硬件可靠性技术。通过采取与硬件类似的数学建模方法,能够建立软件可靠性框架。用故障率作为度量,可直接用于模型;综合硬件和软件可靠性,就形成系统可靠性模型度量形式统一;框图设计的系统可靠性模型直观,易于迅速识别可靠性关键构件部件。

但故障模式影响分析(FMEA)和故障模式影响关键性分析(FMECA)对硬件系统应用很好,对软件系统的认识还不充分。

3.2 软件可靠性建模

SRGM几乎是所有的软件可靠性模型中最成功的,但它们的主要缺点包括:软件测试期间的观察可能不能直接扩展到操作使用中;收集到的故障数有限,可靠性预计未必准确;SRGM的某些假设不真实[32]。但这些问题可通过合适的方法来克服,比如收集数据时,满足模型假设,则可得到精确的可靠性估计。

添加影响软件质量的重要因子,能够进一步改善这些SRGM模型的预计精度。文献[33]提出了一种技术,将代码覆盖对可靠性影响结合进传统的软件可靠性模型中,使用时间和代码覆盖度量来预计可靠性。故障/错误数目和测试覆盖之间的关系,可由不同分布的模型来确定。文献[34]提出,这个关系是Rayleigh分布的变量。文献[35]说明,假设错误覆盖和测试覆盖遵循与执行时间相关的对数NHPP增长模型,那么它能够表示为对数-指数形式。如何将更多度量结合进这类建模,有待进一步探索。

早期的软件可靠性预计模型,通常很不充分,可执行程度低,不够形式化而难以分析,且与目标系统无关。它们对可靠性预计的影响也十分有限。

3.3 软件可靠性度量

软件复杂度和软件质量与软件可靠性密切相关,软件可靠性模型也包括了复杂性度量,如初始的软件错误密度和故障率。

SRGM中与可靠性相关的两个度量是时间区间内的故障数和故障间隔时间。CPU时间比日历时间更合适且更精确,CPU时间能够如实地表示软件的实际执行时间[36]。软件可靠性建模已合并了测试效果的其他度量,以改善预计的精确程度[37,38]。

关键问题是对软件度量的定义和解释不一致,迫切需要识别、描述、合并和理解统一的软件可靠性度量。

3.4 数据收集和分析

数据收集对软件可靠性度量的成功起关键作用。

基于错误的数据是静态的,可使用维护源代码的配置管理工具来收集。基于故障的数据很难收集。故障发生后,必须记录故障的出现时间。手工记录日历时间,会使建模不够精确;记录CPU时间,特别是在多个CPU并行地执行软件的分布式系统和网络环境中,很难收集。如果自动地收集数据,缺点是比较具有干扰性,如在线收集数据能够引起对测试中系统的中断。

另外,不同组织之间所需收集数据的数量和类型也各不相同,要根据SRE目的搜集类似的数据。

对领域故障数据的分析,有5步:1)数据预处理;2)数据分析;3)模型结构识别和参数估计;4)模型求解;5)模型分析。

3.5 方法和工具

除了软件可靠性增长建模外,SRE还有许多其他方法。接下来给出方法和工具的一些实例。

故障树提供了图形化和逻辑框架,系统地分析故障模式。故障树技术起源于硬件系统,扩展到软件系统,它能够为硬件/软件协同设计提供统一的建模方案。为硬件-软件交互的可靠性建模是目前一个研究热点[39]。

SRE仿真技术假设给定的环境代表了影响因素,仿真方法的结果就会代表实际软件环境中的结果[40]。仿真技术通过实验来分析软件故障和其他软件度量之间关系的本质。另一种可靠性模拟方法[13]产生与可靠性相关的行为和事件的时间轴模拟。

已经开发了不同的SRE度量工具来进行数据收集、可靠性分析、参数估计、模型的应用和可靠性估计。改进这些工具,也是对SRE的改进。需要提供工具和环境,支持开发人员建立可靠的软件。工具、环境和技术的划分,应该反映目前最好的SRE实践的合理应用。

3.6 测试效果和代码覆盖

软件测试是软件可靠性工程学中排除错误的典型机制,它已成为工业界中确保质量和改善可靠性的广泛实践。有人提出用代码覆盖指示软件测试的充分程度,它也广泛用于表示错误覆盖。

尽管代码覆盖和错误覆盖呈现出同时增长的现象,但这到底能够归结为代码覆盖和错误检测之间的因果依赖,还是

两种度量的累计本质的巧合？在对此问题的研究中，进行了与蒙特卡罗模拟有关的实验，假设在代码覆盖和错误检测之间没有因果依赖[41]。发表的数据的测试结果支持在代码覆盖和错误覆盖之间无因果依赖。但许多研究人员将覆盖作为软件测试效果的可靠指标。

3.7 测试和操作剖面

操作剖面有助于确定测试时间，产生测试用例，选择测试运行。很多工业项目中能够见到采用操作剖面的好处的实例[42]。使用操作剖面来指导系统测试，确保了在终止测试和由于时间限制必须发布软件时，使用最多的操作受到的测试最多，可靠性最大。

存在的问题主要有：很难开发某些应用的操作剖面，特别是分布式软件系统，如 Web 服务；软件不能复制操作剖面来加速测试；单元测试时可使用操作剖面，而系统测试或回归测试时不适用。处理不正确的操作剖面和操作剖面间的依赖，是操作剖面技术中的两个主要问题。

3.8 工业实践和关心的内容

管理软件可靠性工程学有一些成功的例子，但并没有得到广泛的应用。工业中对 SRE 缺乏热情的主要原因是它的成本效益不明显。许多公司不愿意花太多精力收集故障数据，更不用说数据共享。因此，不能比较可靠性的结果，得不到基准可靠性，也难以积累过去的经验。大多数软件实践人员采用直接的方法和度量来控制产品可靠性。产品的可靠性属性优先级常常比功能性或创新性要低。产品发布时间很紧时，可靠性常常是第一个被压缩的元素。

系统体系结构框图的可靠性预计和 FME(C)A 已成功地用于硬件可靠性工程学技术，而仍不能直接用于软件可靠性工程学。如果软件构件可以被分解，或者能够清楚地识别构件之间的依赖，并正确建模，则这些硬件的普遍技术可以用于软件，并广泛应用于工业中。

3.9 软件体系结构

过去 10 年对软件体系结构的系统化研究非常活跃。在将来的 10 年，它将仍然是研究的中心[43]。软件体系结构设计不仅影响软件开发行为，也影响 SRE 的效果。不同软件块之间的依赖减少，这样它们的可靠性不会互相影响。设计软件体系结构主要考虑错误隔离。这些基于构件的软件开发方法需要不同的框架、质量保证模式[44]和从传统软件开发中得到的可靠性建模方法[45]。

新兴的软件体系结构包括跨平台的技术[46]、开放世界软件[47]、面向服务的体系结构[48]和 Web 应用[49]。虽然已有一些建模方法用来估计特定 Web 系统的可靠性[50]，但对一般 Web 服务和其它面向服务体系结构的 SRE 技术还需要做更多研究。

4 未来的发展趋势

4.1 软件体系结构和现成构件的可靠性

由于现代软件越来越多地开发并使用可重用的构件，软件开发的可靠性工程学主要集中于两点：软件体系结构和基于构件的软件工程学。

软件体系结构是最终软件产品的基础，设计和管理软件体系结构是软件可靠性工程学研究中的支配因素。清晰的、精心组织的体系结构必须事先考虑高内聚、低耦合、模块的分离、适当的系统终止、简明的接口、避免复杂性等等，这些将最终有助于增加软件可靠性并减少操作和维护成本。

核心问题是设计故障恢复体系结构。设计必须确保软件执行时构件是分离的。构件故障一旦出现，就能被迅速识别并将其完全隔离。如何确保应用之间的分离，分离方法如何混合应用才能得到更好的可靠性结果，是将来研究的一个方向。

在设计故障恢复体系结构时，除了容忍构件故障之外，还常常结合额外的资源和技术，例如错误检测、诊断、隔离和恢复过程的错误处理机制。应避免额外的可靠性增长程序引入不必要的复杂性。执行恢复功能时，它们不应该有干扰性。设计简明、简单而有效的机制来执行错误检测和回复，是活跃的研究主题。

软件体系结构代表了软件系统的产品观点，而基于构件的软件工程学强调了软件工程学的过程观点。需要研究的问题有：怎么识别和设计可靠的、通用的可重用构件？如何修改现存构件，以便于重用？如何为构件提供清楚的接口设计，使得它们的交互完全在控制中？怎样给构件提供防御机制，保护构件，而不引起大的故障？怎样确定构件是否无风险？怎样在没有测试而可预见的操作条件下评价构件的可靠性？如果构件不是独立的，怎样对构件交互建模？基于构件的软件工程学也可以实现基于结构的可靠性，这有利于运行和测试软件前的可靠性设计。主要是需要适当地捕获构件间的依赖并对之建模。

4.2 实现设计可靠性

可靠的系统设计需要的容错机制包括错误限制、错误检测、诊断、重新配置、恢复、重启、修理和重组。这些阶段挑战性的研究问题，如下所述：

1)错误限制。实现方式包括自检接受测试、意外处理例程、一致性检查机制和多请求/确认。关键是减少依赖。这已是且将仍然是软件可靠性工程学中的开放问题。

2)错误检测。有离线和在线错误检测。离线技术如诊断程序能够提供全面的错误检测，但系统在测试状态下不能工作。在线技术，如看门狗监控器或冗余配置，与任务并行执行，提供了实时的检测能力。

3)诊断。诊断提供故障位置和/或属性的信息。在线的故障预防诊断是目前的研究趋势。

4)重新配置。成功地重现配置要求健壮灵活的软件体系结构和相关的重新配置计划。

5)恢复。恢复技术基于错误掩盖、重试和回滚。错误掩盖利用冗余来掩盖故障的影响，如 N 版本编程。重试，即尝试第二次操作，可用恢复块重试。回滚，从检查点重新开始错误检测和操作，回滚必须足够远。N 版本程序设计和恢复块体现了设计的多样性，这仍然是争论的热点。

6)重启。热启动能够从错误检测点处恢复所有操作。只在系统没有损害时，才可能热启动。暖启动中，只有某些过程能够被无损恢复；而在冷启动中，所有进程被杀死，执行的系统完全重载。

7)修理。有离线和在线修理。在线修理时，构件可以用后备备件立即替换，或没有错误构件(例如掩盖冗余或优雅降级)继续操作，系统操作不会被干扰；但实现完全和无缝的修理对研究人员来说提出了很大挑战。

8)重组。将修理了的模块重组合进系统。在线修理时，重组不能干扰系统操作。

能够进一步探索可靠性设计技术的4个领域是：错误预防、错误检测、掩盖冗余和动态冗余。非冗余系统不容忍错误，通常由错误预防获得可靠性。冗余系统一般使用错误检测、掩盖冗余和动态冗余，自动处理错误。软件容错设计主要考虑的是成本效益。产生的设计必须能够有效地提供更好的可靠性，同时不引入过多的成本，包括性能降低和复杂度升高的成本。

4.3 可靠性估计评价测试

传统的软件测试和软件可靠性结合得不好，将来两者的结合会是重要的研究点。使用度量测试压缩因子，能够根据测试中估计的软件可靠性来预计领域中的软件可靠性。将与测试有关的其他因素结合进软件可靠性建模，也能够基于测试结果得到精确的量度。

最近研究了代码覆盖对错误检测的影响[51]。需要实验数据来评价代码覆盖，看它能否说明测试集的有效性。在不同的测试剖面下，代码覆盖对错误检测的影响不同。为有效地发现错误，白盒测试和黑盒测试应该相互校验，从而增加可靠性。

此外，不同的覆盖度量之间具有差异，应该去除掉冗余的度量。代码覆盖和其它度量的关系，能够指导选择和评价测试用例，并得到带有可预测的可靠性结果的系统化测试计划。

减小测试集的规模是软件测试的关键目标。应该在不同的测试场景下，评价不同的测试度量，看它们在减小测试集的同时是否得到了相同的可靠性[52]。如果能够定量确定测试用例的效果，在创建测试集合时就预计可靠性的增长，就能促进早期的可靠性预计和反馈控制。

4.4 可靠性预计的度量

多个组织之间，很难收集与系统故障有关的工业软件工程学的历史数据。在不同的项目和应用之间，收集度量数据是一个大问题。多个组织共同研究建立可信的可靠性分析过程，要定义：①考虑到域的灵敏性、可访问性、保密和效用，要收集哪些数据；②如何根据工具和技术来收集数据；③怎么使用现存的技术解释和分析数据。

除了工业数据收集，改善可靠性预计的新方法也是活跃的研究问题。例如，贝叶斯信任网络(BBN)从度量数据中提取丰富的信息，使用合理的概率统计基础，为研究软件工程学提供了一种有希望的方向[53]。对于评价软件系统的可靠性，发现预计软件缺陷方面的应用，预计软件可靠性和确定无法控制的项目[54]，软件度量是有力的工具。

将测试完整性或有效性的度量，如代码覆盖和传统的基于测试时间的度量结合进来，能够加强传统的可靠性模型。测试时间和覆盖两个维度之间的依赖程度，是研究人员的一个开放问题。将时间和覆盖度量合并进新的可靠性模型来预计可靠性，会是有前景的方向。

目前度量和数据收集过程是单向、开环的，不能提供关于如何进展的反馈过程，这是一个缺点。传统的软件可靠性模型将度量(例如，故障密度或故障间隔时间)作为输入，产生可靠性的量作为输出。将来迫切需要是反向的：给定一个可靠性目标，可靠性过程(和产生的度量)会怎么样？通过提供这样的反馈，期望闭环过程能够为得到可预测的可靠性软件提供信息，并对软件可靠性工程学有益。

4.5 新兴软件应用的可靠性

新兴的软件应用将为特定领域的可靠性工程学创造大量的机会。

服务业是软件中大量存在的关键产业。面向服务的设计也被应用于软件工程学领域。基于Web应用开发的未来趋势，如文献[56]中展示的那样，软件可靠性工程学对这项新兴的技术提出了巨大的挑战和机会。可靠的Web服务的设计和评价Web服务的可靠性是新的和开放性的研究问题。无缝的Web服务组合是可靠性工程学的核心问题。也需要在Web服务可靠性度量领域中进行广泛的实验。

研究人员已提出将发布/订阅模式作为中间件平台的基础，中间件支持高度演化和动态构件联合的软件应用。文献[57]中介绍了基于中间件的软件工程学技术中的未来趋势。

开放系统的方法是软件应用中的另一个趋势。世界本质上是开放的，特别是在普适计算的背景下。应用覆盖了很宽范围的领域。在开放的世界中，环境不断地变化。软件必须适应变化，并动态地对变化作出反应。在执行时，系统能够发现这样的构件以及动态地将它绑定到应用上，因此软件必须展示出自组织能力。软件设计人员首先导出变更需求，区分优先次序并详细说明、设计、运行和测试，然后重新配置软件，这种传统的解决方法不再可行。为对软件演化做出快速的反应，我们需要更灵活和动态的可调节的可靠性工程学模式。

结束语 本文回顾了软件可靠性工程学的历史、目前趋势和现存问题以及特有的困难。也着重叙述了软件可靠性工程学中将来可能的方向和有前景的研究问题。为了满足工业以及顾客的需要，我们展现了软件可靠性工程学目前的和将来可能的趋势。从软件体系结构、测试技术和软件故障表现机制等角度，我们建立了新的软件可靠性工程学模式，也提供了关于新兴的软件应用的一些想法。

参考文献

[1] 蔡开元. 软件可靠性工程基础[M]. 北京：清华大学出版社，1995

[2] Lyu M R. Software Reliability Engineering: A Roadmap, Future of Software Engineering// IEEE, 2007. Future of Software Engineering, 2007. Digital Object Identifier 10. 1109/FOSE. 2007. 24, FOSE '07 23-25, May 2007: 153-170

[3] 徐仁佐. 广义软件可靠性工程学[M]. 北京：清华大学出版社，2006

[4] 王纬. 软件工程与软件可靠性[J]. 质量与可靠性，2001(4)：31-34

[5] 向宝坚，孙晓. 软件可靠性工程展望[J]. 株洲工学院学报，1999，13(6)：42-44

[6] 陈光宇，黄锡滋. 软件可靠性学科发展现状及展望[J]. 电子科技大学学报：社科版，2002，4(3)：99-102

[7] 王纬. 软件可靠性工程发展现状[J]. 装备质量，2000(5)：52-56

[8] 黄锡滋. 软件的可靠性与安全性[M]. 北京：科学出版社，1993

[9] 徐仁佐. 软件可靠性专家系统[M]. 北京：清华大学出版社，1996

[10] 刘云，赵纬. 软件可靠性研究与进展[J]. 微机发展，2003，13(2)：12-15

[11] Bertolino A. Software Testing Research: Achievements, Challenges, Dreams// Briand L, Wolf A, eds. Future of Software Engineering 2007. IEEE-CS Press, 2007

[12] Cheng J, Bell D A, Liu W. Learning Belief Networks from Data:

An Information Theory Based Approach // Proceedings of the Sixth International Conference on Information and Knowledge Management. Las Vegas, 1997: 325-331

[13] Lyu M R. Handbook of Software Reliability Engineering. IEEE Computer Society Press and McGraw- Hill, 1996

[14] 崔玉宝,曲凤娟. 软件可靠性工程研究[J]. 科技情报开发与经济,2006,16(6):224-226

[15] 徐仁佐. 软件可靠性模型及应用 [M]. 北京:清华大学出版社,1994

[16] 党涛立,潘新祥,赵海涛. 软件可靠性工程综述[J]. 鱼雷技术,2005,13(6):13-16

[17] Lyu M R, Cai X. Fault-Tolerant Software // Benjamin Wah, ed. Encyclopedia on Computer Science and Engineering. Wiley, 2007

[18] Torres-Pomales W. Software Fault Tolerance: A Tutorial. TM-2000-210616. Hampton, Virginia: NASA Langley Research Center, Oct. 2000

[19] Eckhardt D E, Lee L D. A Theoretical Basis for the Analysis of Multiversion Software Subject to Coincident Errors. IEEE Transactions on Software Engineering, 1985, 11(12): 1511-1517

[20] Littlewood B, Miller D. Conceptual Modeling of Coincident Failures in Multiversion Software. IEEE Transactions on Software Engineering, 1989, 15(12): 1596-1614

[21] Dugan J B, Lyu M R. Dependability Modeling for Fault-Tolerant Software and Systems // Lyu M R, ed. Software Fault Tolerance. New York: Wiley, 1995: 109-138

[22] Tomek L A, Trivedi K S. Analyses Using Stochastic Reward Nets // Lyu M R, ed. Software Fault Tolerance. New York: Wiley, 1995: 139-165

[23] Popov P T, Strigini L, May J, et al. Estimating Bounds on the Reliability of Diverse Systems. IEEE Transactions on Software Engineering, 2003, 29(4): 345-359

[24] Lyu M R, Cai X. Fault-Tolerant Software // Benjamin Wah, ed. Encyclopedia on Computer Science and Engineering. Wiley, 2007

[25] Cai X, Lyu M R, Vouk M A. An Experimental Evaluation on Reliability Features of N-Version Programming // Proceedings 16th International Symposium on Software Reliability Engineering (ISSRE'2005). Chicago, Illinois, Nov. 2005

[26] Cai X, Lyu M R. An Empirical Study on Reliability and Fault Correlation Models for Diverse Software Systems // Proceedings 15th International Symposium on Software Reliability Engineering (ISSRE'2004). Saint-Malo, France, Nov. 2004: 125-136

[27] 张亚明,王凌. 软件可靠性工程管理[J]. 数量经济技术经济研究,2003 (12):117-121

[28] 赞岳,毛晓光,王戟,等. 一个集成的软件可靠性工程环境[J]. 计算机工程与科学,2007,29(7):116-118

[29] 李明强,刘治德,马聪颖. 机载武器系统软件的可靠性工程管理[J]. 电子产品可靠性与环境试验,2001(6):13-16

[30] 何国伟. 软件可靠性[M]. 北京:国防工业出版社,1998

[31] 陶克强,陶捷,王纬. 软件可靠性工程的研究与发展方向[J]. 质量与可靠性,1992(4):39-41

[32] Shooman M L. Reliability of Computer Systems and Networks: Fault Tolerance, Analysis and Design. New York: Wiley, 2002

[33] Chen M, Lyu M R, Wong E. Effect of Code Coverage on Software Reliability Measurement. IEEE Transactions on Reliability, 2001, 50(2): 165-170

[34] Vouk M A. Using Reliability Models During Testing With Non-operational Profiles // Proceedings of 2nd Bellcore/Purdue Workshop on Issues in Software Reliability Estimation. October 1992: 103-111

[35] Malaiya Y K, Li N, Bieman J M, et al. Software Reliability Growth with Test Coverage. IEEE Transactions on Reliability, 2002, 51(4): 420-426

[36] Musa J D, Iannino A, Okumoto K. Software Reliability: Measurement, Prediction, Application. New York, NY: McGraw-Hill, Inc., 1987

[37] Cai X, Lyu M R. The Effect of Code Coverage on Fault Detection Under Different Testing Profiles // ICSE 2005 Workshop on Advances in Model-Based Software Testing (A-MOST). St. Louis, Missouri, May 2005

[38] Huang C Y, Lyu M R. Optimal Release Time for Software Systems Considering Cost, Testing-Effort, and Test Efficiency. IEEE Transactions on Reliability, 2005, 54(4): 583-591

[39] Teng X, Pham H, Jeske D. Reliability Modeling of Hardware and Software Interactions, and Its Applications. IEEE Transactions on Reliability, 2006, 55(4): 571-577

[40] von Mayrhauser A, Chen D. Effect of Fault Distribution and Execution Patterns on Fault Exposure in Software: A Simulation Study. Software Testing, Verification & Reliability, 2000, 10(1): 47-64

[41] Briand L, Pfahl D. Using Simulation for Assessing the Real Impact of Test Coverage on Defect Coverage. IEEE Transactions on Reliability, 2000, 49(1): 60-70

[42] Musa J D. Operational Profiles in Software Reliability Engineering. IEEE Software, 1993, 10(2): 14-32

[43] Taylor R, van der Hoek A. Software Design and Architecture: The Once and Future Focus of Software Engineering // Briand L, Wolf A, eds. Future of Software Engineering 2007. IEEE-CS Press, 2007

[44] Cai X, Lyu M R, Wong K F. A Generic Environment for COTS Testing and Quality Prediction // Beydeda S, Gruhn V, eds. Testing Commercial-off-the-shelf Components and Systems. Berlin: Springer-Verlag, 2005: 315-347

[45] Yacoub S, Cukic B, Ammar H. A Scenario-based Reliability Analysis Approach for Component-based Software. IEEE Transactions on Reliability, 2004, 53(4): 465-480

[46] Bishop J, Horspool N. Cross-Platform Development: Software That Lasts. IEEE Computer, October 2006: 26-35

[47] Baresi L, Nitto E, Ghezzi C. Toward Open-World Software: Issues and Challenges. IEEE Computer, October 2006: 36-43

[48] Margaria T, Steffen B. Service Engineering: Linking Business and IT. IEEE Computer, October 2006: 45-55

[49] Jazayeri M. Web Application Development: The Coming Trends // Briand L, Wolf A, eds. Future of Software Engineering 2007. IEEE-CS Press, 2007

[50] Wang W, Tang M. User-Oriented Reliability Modeling for a Web System // Proceedings of the 14th International Symposium on Software Reliability Engineering (ISSRE'03). Denver, Colorado, November 2003: 1-12

[51] Lyu M R, Huang Z, Sze S, et al. An Empirical Study on Testing and Fault Tolerance for Software Reliability Engineering // Proceedings 14th IEEE International Symposium on Software Reliability Engineering (ISSRE'2003). Denver, Colorado, November

2003:119-130

[52] Cai X, Lyu M R. The Effect of Code Coverage on Fault Detection Under Different Testing Profiles// ICSE 2005 Workshop on Advances in Model-Based Software Testing (A-MOST). St. Louis, Missouri, May 2005

[53] Cheng J, Bell D A, Liu W. Learning Belief Networks from Data: An Information Theory Based Approach // Proceedings of the Sixth International Conference on Information and Knowledge Management. Las Vegas, 1997:325-331

[54] Amasaki S, Mizuno O, Kikuno T, et al. A Bayesian Belief Network for Predicting Residual Faults in Software Products// Proceedings of 14th International Symposium on Software Reliability Engineering (ISSRE2003). November 2003:215-226

[55] Jazayeri M. Web Application Development: The Coming Trends // Briand L, Wolf A, eds. Future of Software Engineering 2007. IEEE-CS Press, 2007

[56] Issarny V, Caporuscio M, Georgantas N. A Perspective on the Future of Middleware-based Software Engineering. Briand L, Wolf A, eds. Future of Software Engineering 2007. IEEE-CS Press, 2007

(上接第 4 页)

性,其采用的方法包括:基于 XML 的、基于 CORBA 的等。目前我国正在开展的联网审计也是持续审计的一种方式,它在技术实现上属于分离式[31]。

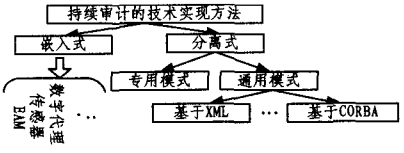

图 2 持续审计技术实现方法研究的分类

根据以上分析可知,随着信息技术的发展,持续审计软件的研制与应用是审计软件的一个重要发展方向。

另外,目前对于持续审计实现技术和应用的研究较多,但缺少关于持续审计实施绩效评价方面的研究与应用。而研究持续审计实施绩效,有助于重新认识目前的持续审计政策和持续审计实现技术,有助于审计机关更有效地制定持续审计对策。因此,关于持续审计软件实施绩效评价方面的研究与应用也是审计软件研究与应用的一个重要发展方向。

参 考 文 献

[1] Robert L B, Harold E D. Computer-assisted audit tools and techniques: analysis and perspectives[J]. Managerial Auditing Journal, 2003, 18 (9): 725-731

[2] INTOSAI audit committee. Principles of computer assisted audit techniques-student notes, 2004

[3] James A H, Schaller C A. Information systems auditing and assurance [M]. Cincinnati, OH: South-Western College Publishing, 2000

[4] 6th ASOSAI Research Project on IT Audit Guidelines. http://www.asosai.org, 2003

[5] 陈伟,张金城,Qiu Robin. 计算机辅助审计技术(CAATs)研究综述. 计算机科学[J], 2007, 34(10): 290-294

[6] Weidenmier M L, Herron T L. Selecting an audit software package for classroom use[J]. Journal of Information Systems, 2004, 18(1): 95-110

[7] Jackson R A. Get the most out of audit tools [J]. Internal Auditor, 2004(8): 36-47

[8] http://www.caseware-idea.com

[9] http://www.acl.com

[10] http://www.horwathsoftware.com

[11] http://www.paisleyconsulting.com

[12] http://www.pwcglobal.com

[13] http://www.pentana.com

[14] Chen Wei, Wang Hao, Zhu Wenming. Study on data-oriented IT audit used in China[A]// Zhu Qingsheng, ed. Proceedings of the 11th Joint International Computer Conference (JICC2005)[C]. Singapore: World Scientific Publishing, 2005: 666-669

[15] 国家 863 计划审计署课题组. 计算机审计数据采集与处理技术研究报告[M]. 北京:清华大学出版社, 2006

[16] 中华人民共和国审计署《AO》研发项目组. 现场审计实施系统实用手册[M]. 北京:中国时代经济出版社, 2005

[17] http://ytaudit.com

[18] http://www.auditsoft.com.cn

[19] http://www.tonsin.net

[20] http://www.auditcn.com

[21] http://www.zperp.com

[22] http://www.jinjiansoft.com

[23] 陈伟,Qiu Robin,刘思峰. 一种基于数据匹配技术的审计证据获取方法[J]. 计算机科学, 2008, 35(8): 183-187, 194

[24] Du H, Roohani S. A framework for independent continuous auditing of financial statements [A]// American Accounting Association 2006 Annual Meeting [C]. Washington, 2006

[25] Groomer S M, Murthy U S. Continuous auditing of database applications: an embedded audit module approach [J]. Journal of Information Systems, 1989, 3(2): 53-69

[26] Debreceny R S, Gray G L, Ng J, et al. Embedded audit modules in enterprise resource planning systems: implementation and functionality [J]. Journal of Information Systems, 2005, 19(2): 7-27

[27] Vasarhelyi M A, Halper F B. The continuous audit of online systems [J]. Auditing: A Journal of Practice and Theory, 1991, 10(1): 110-125

[28] Woodroof J, Searcy D. Continuous audit-Model development and implementation within a debt covenant compliance domain [J]. International Journal of Accounting Information Systems, 2001, 2(3): 169-191

[29] Rezaee Z, Sharbatoghlie A, Elam R, et al. Continuous auditing: building automated auditing capability[J]. Auditing: A Journal of Practice and Theory, 2002, 21(1): 147-163

[30] Murthy U S, Groomer S M. A continuous auditing web services model for XML-based accounting systems [J]. International Journal of Accounting Information Systems, 2004, 5(2): 139-163

[31] Chen Wei, Zhang Jin-cheng, Jiang Yu-quan. One continuous auditing practice in China: data-oriented online auditing (DOOA) [A]// The 7th IFIP International Conference on e-Business, e-Services, and e-Society (I3E2007) [C]. Boston: Springer, 2007: 521-528