用基于阶段的方法创建高可靠性软件

香港中文大学计算机科学与工程系
吕荣聪 教授

1.介绍

人们对于复杂的软/硬件系统的需求日益增长,而我们设计，实现，测试和维护的能力却相对落后。当我们越来越需要并且依赖计算机的时候，计算机故障所引起的危机也就越来越严重。这些冲突渗透在各个领域，例如家用电器出现故障给人们带来的不便，银行业务系统中断所造成的经济损失，航空系统或医疗软件故障造成的人员伤亡等等。因此，计算机系统的可靠性就成为社会关注的一个主要问题。

然而计算机革命的进程并不平衡：基础薄弱的软件相比硬件要承受更大的负担。与快速发展的硬件技术形成鲜明对比的是软件开发在其各个方面，包括质量，生产力，生产成本以及性能上，都无法与硬件保持同步。软件的滞后和高成本经常使一些现代化的复杂项目陷入危险的境地。软件已经成为整个系统发展的瓶颈。

后来，许多软件公司看到一个项目开发的成本主要集中在设计，实现和确保软件的可靠性方面。他们认识到在一个系统内，大量需要能确保软件可靠性的系统化方法。显然，当前以及未来几十年，开发能满足软件可靠性工程所需的技术就成为硬件，软件以及各相关学科的工程师们所面临的主要挑战。

2.基于阶段的方法：概述

软件可靠性工程的核心是可靠性，它也是软件属性中很重要的方面。软件可靠性被定义为：在指定的环境和时间段里，软件运行出现任意故障的可能性。它是软件质量的一种属性，这种属性受到很多其它因素的影响，如功能性、可用性、性能、适用性、用基性、可于阶性，可维护性及文段等。因此，软件可靠性工程包的以方内基：

法创以软件可靠性建高为基础的软件可靠性可量,包括对可靠性的靠算和性测。

法创项目的设计、开发软程、件系香港、运行环境等与可靠性相关属性的定义及可量。

法检运用可靠性可量的技术来文大和指学软件的香港、开发、测试、确认、使用及维护。

我认为我们计算机科个阶段与工软件可靠性工程的问题程1系建建和机吕阶段荣程2系设计和实现阶段荣程中系测试和聪教阶段。所授这些阶段都 及到软件故障的1. 。介一个阶段，绍人系统香港和可能的故障们设建对起可靠性建高,其主要工复是建对系统的故障建高，并且杂出一系/ 硬如统...,需会...求的问题。可日使用的建建方法授益香港增机吕建高、长, 可而我建高、故障设机吕建高、,实测试和建高等。介维个阶段，在系统内建对可靠的护件，其核心是实现软件的能力和基力。我们主要却相使用可复用的基力例程和以多种方落设计的可以基力的软件。介科个阶段，后软聪靠和可量技术来聪教软件系统的可靠性，主要是。力和性力。与此相关的技术授对当人越的测试、可靠性可量和软件可靠性测试工来等。方面科要机需并且依赖这些技术。

中阶段一：建建和机吕阶段

为时在前候设计中进行可靠性的机吕和建建，以需求为基础的整个系统香港可以用以方几种技术建建。这些建建方法授香港增机吕建高、长，可而我建高、故障设机吕建高、，实现测试和建高。这些方法可用于为系统建对可靠性和性能建高，以便故障在各种所引中系统的运行起危。如系统的可靠性可以后软给定的件系香港也当被就严地测算出来，重方来就可以使用这些性机吕冲出对系统突关重要的也当以及系统中的关渗要机。透实上在这个阶段建对的可靠性建高在经软在各和个化后领可以用于系统设计后候对系统的机吕和聪教。

我们域例使用系统香港增对实如项目进行建建和机吕，香统家明软件的可靠性已经成为系统是电可靠的关渗因素。重方来我们器明时故障设是如出被用于对关渗性计用中故障建落的定性和定量的机吕，这是一种能给对学带系统失不的各种因素的便银的当学和增形化的行述，使用它能给使我们业入地时与系统内与可靠性务断相关的造在弱成。在这个阶段使用各种工来来经成建建和机吕是济要的，我们认为损定或和 医现时析是其中亡为例进的工来。

等阶段维：设计和实现阶段

在设计和实现阶段，软件可靠性在系统内要因以实现，其核心是实现软件的能力和基力。能力是许多软件工程技术的主题，本文不此依赖。本文需重成为在基力方面的问题上，并需社会在关一注本和多种注本的环境中基力技术的使用。

我们后软使用一主的力问题测和然复测严可以实现关一注本软件环境中的基力。可使用的革严包括建命化、进包的性、平复的衡：化、础行前的确认和薄常弱．等。使用可复用的软件基力例程是一种比承的方法。

中间件平受包括一系/ 可复用护件法测试和快，实现或展，试技术和 成机测判别。它们础行一些力问发生前及发生后的基力任务。使用这些可复用护件的硬件平受是一个是开其的计算机护成的和面，和面中包一受计算机都杂日对和面中其他计算机的括质，并是护件来杂日题量成、日生、产本、以测、及上、重都和故障然复等机无来能保力问。

持同，我们需在一个多注本环境的基力大例中展步软件基力技术的滞常软程。

使阶段科：测试和可量阶段

在这个阶段软件可靠性需被聪靠和代化。一些测试和可量技术可以用来经成这个目开。测试技术是为时。力，可靠性聪靠技术是为时性力。一些测试方项和工来可在这个阶段用来进行测试和可靠性可量。

软件测试授很多方法，所目的功能测试、及陷测试、集成测试、入险测试、关境测试、地已测试、确认测试等整是我们常瓶技术中的一要机。颈许程或地已系测试是一种基于软件内要多公香港的测试，用来可量测试的质量，包括司看地已、到件地已、集确地已以及引入力问测试。我们需并且器明这些测试方项。我们领他认步大时 航测在集确地已测试方项中的用法和引入力问测试的软程。

持一方面，软件可靠性可量是显用软件运行和测试软程中所产生的故障当人进行统计前赖的软程。我们已经建对起一个用于软件可靠性可量目的基本未几，包括方面十个主要要机:程1系可靠性目开荣程2系年复概满荣程中系可靠性建高及可量荣程等系可靠性确认。

我们设计并实现时一个软件可靠性建高工来，命足为硬计算机师临软件可靠性聪靠系统程内损或求 可以系统化地进平聪靠软件可靠性。我们需挑例器明并滞步这个工来。

战衡文：

A Phase-Based Approach to Creating Highly Reliable Software

Michael R. Lyu
Computer Science & Engineering Department
The Chinese University of Hong Kong
Shatin, Hong Kong
lyu@cse.cuhk.edu.hk

1. Introduction

Our demand for complex hardware/software systems has increased more rapidly than our ability to design, implement, test, and maintain them. When the requirements for and dependencies on computers increase, the crises of computer failures also increases. The impact of these failures ranges from inconvenience (e.g., malfunctions of home appliances), economic damage (e.g., interruptions of banking systems), to loss of life (e.g., failures of flight systems or medical software). The reliability of computer systems has become a major concern for our society.

Within the computer revolution progress has been un-even: software assumes a larger burden while based on a less firm foundation than hardware. In stark contrast with the rapid advancement of hardware technology, proper development of software technology has failed to keep pace in all measures, including quality, productivity, cost, and performance. Software has become the bottleneck of system development, and its delay and cost overrun have often put modern complex projects in jeopardy.

To this end, many software companies see a major share of project development costs identified with the design, implementation, and assurance of reliable software, and they recognize a tremendous need for systematic approaches to assure software reliability within a system. Clearly, developing the required techniques for software reliability engineering is a major challenge to computer engineers, software engineers, and engineers of various disciplines for now and the decades to come.

2. Phase-based approach: an overview

Software reliability engineering is centered around a very important software attribute: reliability. Software reliability is defined as the probability of failure-free software operation for a specified period of time in a specified environment. It is one of the attributes of software quality, a multi-dimensional property including other factors like functionality, usability, performance, serviceability, capability, installability, maintainability, and documentation. Software reliability engineering therefore includes:

(1) software reliability measurement, which includes estimation and prediction, with the help of software reliability models established in the literature;

(2) the attributes and metrics of product design, development process, system architecture, software operational environment, and their implications on reliability; and

(3) the application of this knowledge in specifying and guiding system software architecture, development, testing, acquisition, use, and maintenance.

My position is that we should attack the problem of software reliability engineering in three phases: (1) Modeling and Analysis Phase, (2) Design and Implementation Phase, and (3) Testing and Measurement Phase. All these phases deal with the management of software faults and failures. In the Modeling and Analysis Phase, reliability of the software system is being modeled according to the structure of the system and possible fault scenarios. The key topic of this phase is to provide fault modeling of the system, and ask the ”what if” questions. The available modeling approaches include system reliability modeling block diagrams, reliability models by Markov chains, fault tree analysis, and stochastic Petri-nets. In the Design

and Implementation Phase, reliability of the software system is being achieved by reliable components built into the system. The key topic of this phase is to provide fault avoidance and fault tolerance. The available techniques we emphasize include reusable software fault tolerance routines, and software fault tolerance by design diversity. In the Testing and Measurement Phase, reliability of the software system is being evaluated and verified by measurement and evaluation techniques. The key topic of this phase is to provide fault removal and fault prediction. The available techniques include data flow testing, reliability measurement tasks, and software reliability tools. We discuss the details of these techniques in the following three sections.

3. Phase 1: modeling and analysis phase

To provide reliability modeling and analysis of a software system during the pre-design phase, the overall system architecture based on requirement can be modeled by several techniques. The available modeling approaches include system reliability modeling block diagrams, Makov-chains reliability modeling, fault tree analysis, and stochastic Petri-nets. These approaches can be used to establish system reliability and performance model for the study of system behavior under various scenarios. The reliability of the system, for example, can be predicted in a coarse basis for the overall system given its architectural options are defined. Sensitivity analysis can then be performed to locate important parameters of the system, and critical components of the system can be identified for enforcement of each component's individual reliability. Note that the reliability model established in this phase can be refined and revised for evaluation purpose in a post-design phase for the purpose of a fine prediction and estimation.

We first perform reliability modeling and analysis using block diagrams for an actual project. It is shown that software reliability has become a major critical factor in system reliability. We further show how fault tree models can be used for the qualitative and quantitative analysis of the failure modes of critical systems. A fault tree provides a mathematical and graphical representation of the combinations of events which can lead to system failure. The construction of a fault tree model can provide insight into the system by illuminating potential weaknesses with respect to reliability of the system. The usage of software tools is a must in the modeling and analysis phase. We consider SHARPE and UltraSAN as two leading tools in this arena.

4. Phase 2: design and implementation phase

In the Design and Implementation Phase, reliability of the software system is being achieved within the system. The key topic of this phase is to provide fault avoidance and fault tolerance. Fault avoidance is the subject of many software engineering techniques and is beyond the scope of this paper. Fault tolerance, on the other hand, is the focus of our discussion. We examine fault tolerance techniques used in single-version as well as multiple-version environments.

Software fault tolerance in single-version software environment is achieved by introducing special fault detection and recovery features, including modularity, system closure, atomicity of actions, decision verification, and exception handling. One successful approach is accomplished by reusable software fault tolerance routines.

A middleware platform containing a set of reusable software components `watchd, libft, REPL, libck`p, and `addreju`v) to perform these reactive and proactive software fault tolerance tasks will be described. The hardware platform for using these reusable software components is a network of standard computers where each computer provides a back-up facility for another one on the network. The components provide mechanisms to checkpoint, log messages, watch, detect, rollback, restart, and recover from failures and rejuvenate to avoid failures.

In addition, we will show the evolution of techniques for building fault-tolerant software out of simplex units, and a design paradigm in achieving multi-version software fault tolerance systems.

5. Phase 3: testing and measurement phase

In this phase the reliability of the software system should be evaluated and verified, and testing and measurement techniques are available to achieve this goal. Testing techniques are for fault removal purpose, and reliability assessment techniques are for fault prediction purpose. This includes schemes and tools for software testing and software reliability measurement.

There are many ways of testing software. The terms functional, regression, integration, product, unit, coverage, user-oriented, are only a few of the characterizations we encounter. White-box, or coverage, testing uses the structure of the software to measure the quality of testing. This structural coverage and its measurement is believed to be connected with reliability estimation. These testing schemes include statement coverage testing, decision coverage testing, data-flow coverage testing, and fault-injection testing. We will address these testing schemes in detail. In particular, we demonstrate the usage of ATAC for data-flow coverage testing schemes, and the procedure applied for fault-injection testing.

Software reliability measurement, on the other hand, is the application of statistical inference procedures to failure data taken from software testing and operation to determine software reliability. We have established a framework for software reliability measurement purpose, including four major components in this software reliability measurement process, namely,

(1) reliability objective, (2) operational profile, (3) reliability modeling and measurement, and (4) reliability validation.

We designed and implemented a software reliability modeling tool, called Computer-Aided Software Reliability Estimation (CASRE) system, for an automatic and systematic approach in estimating software reliability. This tool will also be illustrated and demonstrated.