doi: 10.7690/bgzdh.2014.08.025

# 软件可测试性评估框架构建

刘刚[1,2]，黎放[1,2]，狄鹏[1,2]
（1. 海军工程大学管理工程系，武汉 430033；2. 海军工程大学舰船综合保障工程研究室，武汉 430033）

**摘要：**为了更加合理地开展软件可测试性评估工作，构建一种软件可测试性评估框架，框架从 7 个阶段对软件可测试性进行量化评估。通过分析软件可测试性评估相关理论，确定了软件可测试性评估框架的理论基础，提出了软件可测试性评估工作应贯穿于软件的开发周期，采用因素分解、度量机制和定性评价等措施对软件可测试性进行量化评估，并运用鱼骨模型强化设计阶段软件可测试性评估的重要性。研究结果证明：该框架减少了测试工作的盲目性，有利于指导软件开发人员有针对性地开展测试工作。

**关键词：**可测试性；评估；框架
**中图分类号：**TP306　**文献标志码：**A

# Software Testability Estimation Framework Construction

Liu Gang[1,2], Li Fang[1,2], Di Peng[1,2]
(1.*Department of Management Engineering*, *Naval University of Engineering*, *Wuhan* 430033, *China*;
2. *Lab of Vessel Integrated Logistics Support*, *Naval University of Engineering*, *Wuhan* 430033, *China*)

**Abstract:** In order to carry out the task of software testability estimation reasonably, a software testability estimation framework is constructed, analyzed form seven stages. By analyzing correlative theory of software testability estimation, confirm the theory basis, and the estimation task should through the developing period of software design, use methods of factor decomposition, measurement mechanism and qualitative assessment to estimate the software testability. Moreover, the importance of testability estimation is consolidated by the fish-bone model. The research results indicate that the framework can reduce the blindness of test task, is a useful method for software developer.

**Keywords:** testability; estimation; framework

## 0　引言

“可测试性”的概念最早是在 20 世纪 70 年代初针对硬件测试提出来的[1]。当时，由于一些硬件电路系统膨胀到一定的数量级，对其进行测试就显得异常复杂，众多学者纷纷提出了对硬件电路的可测试性度量方法，于是形成了硬件测试的一个重要分支——可测试性分析。

硬件可测试性分析的含义是:“对一个已初步完成设计的电路，或者待测电路，不具体生成测试码就能定量地估算出电路测试难度的一类方法”。到了 20 世纪 90 年代，人们逐渐把硬件的可测试性分析研究应用到软件上，软件可测试性研究日益成为人们关注的焦点[2]。对软件而言，其产品的可靠性至关重要。近年来，软件应用环境的复杂性日益增加，然而，大多数的软件开发者为了准时交货而减少测试数量和时间，这样就会增加软件潜在的缺陷，导致产品质量不高。软件测试是使用人工或自动手段来运行或测定某个系统的过程，其目的在于检验它是否满足规定的需求或是比较预期结果与实际结果之间的差别。软件测试涉及到软件程序代码，执行环境需求规范，设计文档，操作手册等诸多方面内容。一个完整的软件测试过程包括测试计划、测试准则、测试目标、测试用例、驱动模块、测试环境等许多相关内容。

在软件开始测试之前，进行有选择性的测试分析，比盲目进行测试有着明显的优点[3-5]。可测试性就是分析软件特定模块在测试过程中发现错误的容易程度并预见软件达到规定可靠性所需测试强度的一种能力，由此可见：依据可测试性分析得出的结论而进行的软件测试，可以更为合理地安排软件测试的方向以及所需测试的强度，在保证软件可靠性的基础上，减少了测试的数量，从而也节省了测试费用，提高了测试效率。近几年来，随着软件工程的发展，软件的可测试性已成为一个很重要的研究内容。为了减少软件的错误，软件的可测试性规定了软件测试的强度和难度，其重要目标是减少因软件设计不足而导致的错误的数量。

## 1　软件可测试性评估

软件可测性评估有利于改进软件测试过程，近年来，提出了很多可测试性评估的方法，包括基于规划的可测试性度量、基于模型的可测试性度量、基于可信性的可测试性评定。近年来发表的论文中

**收稿日期：**2014-03-20；**修回日期：**2014-04-02
**基金项目：**国家部委基金资助项目(4314231428)；海军工程大学自然科学基金项目(HGDQNEQJJ13016)
**作者简介：**刘刚(1982—)，男，湖北人，硕士，讲师，从事装备综合保障工程、可靠性、维修性、测试性研究。

提出了几个软件可测试性测量的内在机制，但这些机制、方法仅适用于软件开发的后期。学者和研究人员提出了很多不同的可测试性方法改进软件设计，Voas 和 Miller[6]提出了基于软件结构输入、输出划分的可测试性分析机制，通过运用 PIE(传播、影响、执行)分析工具来改善软件的可测试性评估，但 PIE 计算量大，应用于可测试性评估很困难；Bruce 和 Haifeng[7]分析了目标导向影响可测试性的因素，提出了基于软件测试的故障—错误模型，通过软件部分可测试性评估来进行软件总体的可测试性评估，但是这种评估容易以偏概全，遗漏软件中部分内容；Bruntink 和 Deursen[8]通过资源代码定义了 Java 软件的可测试性评估机制，但这种机制仅适用于 Java 软件系统；Jungmayr[9]提出了软件综合测试性点的观点，这种观点关注软件内部的依赖；Gao 和 Ming[10]提出基于五边形模型的软件测量方法；Mouchawrab 等[11]提出了基于 UML 的可测试性评估模型框架，对软件每个特征进行了假设，分析软件各个部分测试的期望关系，但是这种假设主要以经验为主；Mulo[12]提出了通过严控软件的开发过程来改善软件的可测试性。

通过对软件可测试性评估相关理论进行分析，可以知道现在的研究成果基本上都是对软件局部的可测试性进行研究，对软件整体进行可测试性评估的很少；因此，对软件可测试性评估方面还需做很多工作，很多评估进程、指南和工具还是很缺乏的。

## 2　框架构建理论基础

软件测试是发现并及时纠正软件错误，从而提高软件质量的一种重要手段。近年来，已经提出了许多软件测试方法，这些方法对软件的测试提供了重要的思路。但到目前为止，软件测试有些基本问题尚未解决：一是软件测试的故障模型是什么？二是一定的软件测试方法对软件中存在故障的检测效果如何？三是软件的可测试评估与软件质量之间的关系是什么？而这些问题中，软件的可测性评估与软件质量之间的关系是提高软件质量的本质问题。

软件开发者普遍认为，在软件设计阶段，对软件的可测试性进行整合是很有必要的，这样做能避免资源浪费和促进软件的持续改进。一般认为，用于可测试性评估的机制一般应用于软件开发周期的后期，因此，这种评估机制对软件的改进效果不明显，同时如果在此阶段对软件进行改动，花费是很大的，有时甚至得不偿失；另一方面，设计阶段不考虑软件可测试性，要想在开发阶段进行补充有时也是不可能的。但是至今仍未有广泛全面的模型或框架来用于软件可测试性评估，因此，制定广泛适合各类软件可测试性评估的框架，来指导软件设计阶段的可测试性评估是很有必要的。

在软件设计阶段，提出一种能体现软件可测试性的系统方法是很急迫的，而且效果是显而易见的。这样能避免资源浪费，促进软件可测试性的持续改进。前面提到的研究成果和软件可测试性现状，为整合软件设计阶段可测试性评估提供了坚实的理论基础。笔者基于软件内在的设计特性，运用一种目标导向方法来构建软件可测试性评估框架。该框架可在目标导向软件阶段量化软件的可测试性，是很值得做的，也是富有成效的。

## 3　软件可测试性评估框架

设计早期用于对软件可测试性进行量化评估。框架可见于图 1，图 2 鱼骨模型用于强化设计阶段软件可测试性评估的重要性。框架包含 7 个阶段。

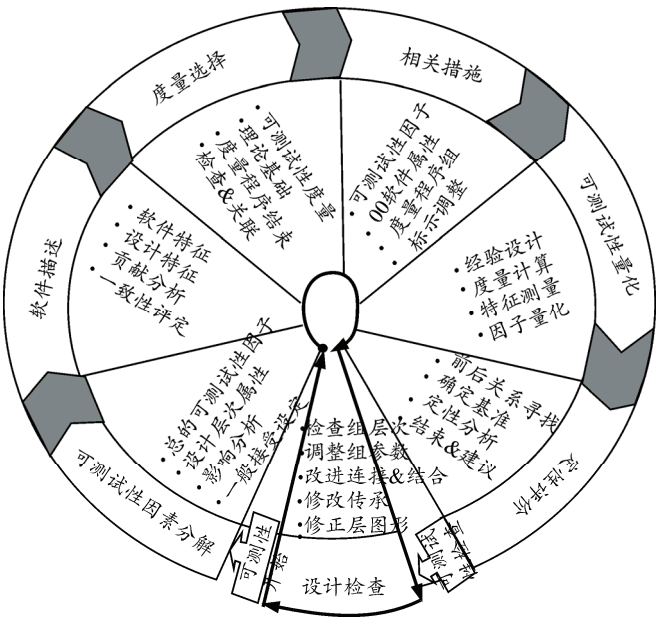


**图 1　软件可测试性框架**

1) 可测试性因素分解。

可测试性是软件设计质量的顶层指标。为了对软件可测试性进行量化评估，就必须对可测试性的因素进行分解。在此阶段，需要确定总的可测试性因子，分析软件设计层次的属性，以及分析这些属性对软件可测试性的影响，最终设定软件可测试性评估的一般接受准则。同时设计层次属性的分析要贯穿于软件设计的全过程。

2) 软件描述。

对软件的质量而言，不同软件的可测试性描述对其影响是一样的。但为了准确分析所需进行可测试性评估的软件的特性，将进行目标导向软件的可

测试性描述，也是为后续工作打下基础。在此阶段，需要分析软件的特征，以及对其进行设计的特征，分析可测试性评估对软件设计的贡献，最终确定软件可测试性评估的一致性评定准则。

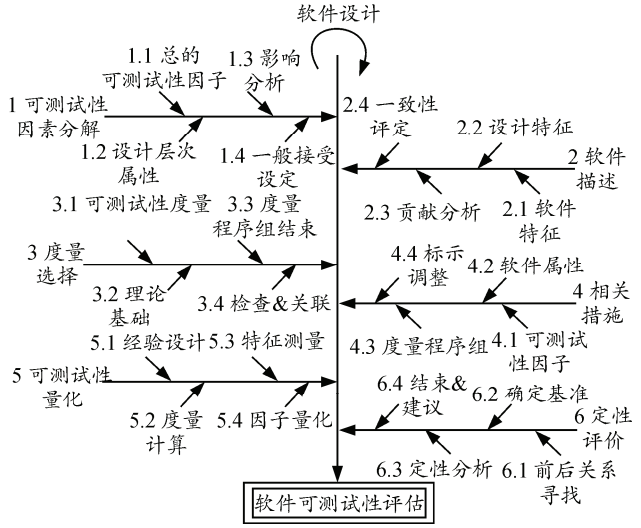


**图 2 软件可测试性框架鱼骨模型**

3) 度量选择。

这是进行软件可测试性评估的重要步骤之一，在此阶段，提出一套软件可测试性评估度量机制，并对其理论基础进行分析，待度量程序结束，检查软件可测试性评估度量效果，分析度量机制与软件可测试性评估的关联。

4) 相关措施。

这是进行软件可测试性评估的关键步骤，在此阶段，需要明确软件属性，确定软件设计属性与可测试性因子的关联；在度量机制的基础上，画出标准线，用于对软件可测试性进行量化评估，同时编写度量程序组，对程序组中可测试性因子进行标示，并在编写的过程中随程序的调整而不断调整。

5) 可测试性量化。

分析软件结构参数和设计层次，确定软件可测试性量化的因素。在此阶段，经验设计发挥很大作用，此外还需进行软件可测试性度量的计算，确定软件特征测量要素，以及将可测试性因子进行量化。

6) 定性评价。

在对软件进行可测试性量化的基础上，执行可测试性定性评价。在此阶段，对软件进行可测试性评估前后关系进行分析，用于回顾和修改给定的设计；确定软件可测试性评估的基准，以及对软件可测试性作定性分析。

7) 设计检查。

在定性评价阶段的基础上，检查和修改给定软件可测试性设计，严格核查软件设计结构，并进行有针对性的调整，以获取结构更好的软件的可测试性。在此阶段，需要对软件程序组的层次进行检查，并调整程序组的参数，改进程序与软件可测试性评估之间的连接或者结合，对软件程序组之间的传承进行修改，最后修改软件可测试性层次图形设计。

## 4 结束语

目前在软件的设计中，可测试性评估是周期较长、问题较多的一个环节，需要不断改进与软件可测试性有关的各种解决方法。软件可测试性评估框架设计是现代系统方法在软件设计中的具体体现，构建的软件可测试性评估框架从软件设计的开始就考虑软件的可测试问题，并在软件的开发周期内持续对可测试性进行评估，有利于指导软件开发人员有针对性地开展测试工作，减少了测试工作的盲目性，对提高测试效率、提升软件质量是必要的。

## 参考文献：

[1] Bart Broekman, Edwin Notenboom. Testing Embedded Software[M]. 张军施, 张思宇, 周承平, 译. 北京: 电子工业出版社, 2004: 1-5.

[2] Ron Patton, 软件测试[M]. 周予滨, 姚静, 等. 译. 北京: 机械工业出版社, 2002: 10-19.

[3] 宫云战, 刘海燕, 万琳, 等. 如何设计一个易测试的软件[J]. 质量与可靠性, 2001, 13(2): 29-31.

[4] 张华, 管红根, 桂成兵. 基于 LabVIEW 的火炮振动测试性分析系统[J]. 兵工自动化, 2012, 31(4): 75-78.

[5] 董成基, 齐杏林, 吕静, 等. 飞行控制软件测试用例生成技术[J]. 兵工自动化, 2012, 31(9): 93-96.

[6] Voas J.M, Miller D.J. Semantic metrics for software testability[J]. Journal of Systems and Software, 1993, 20 (3): 207-216.

[7] Bruce W.N, Haifeng S. A preliminary testability model for object-oriented software[J]. in Proc. International Conf. on Software Engineering, Education, Practice, IEEE 1998, 17(15): 330-337.

[8] Bruntink M, Deursen A.V. Predicting class testability using object-oriented metrics[J]. in Proc. IEEE international Workshop on Source Code Analysis and Manipulation, 2004, 9(2): 136-145.

[9] Jungmayr S. Testability Measurement and Software Dependencies[J]. In Proceedings of the 12th International Workshop on Software Measurement, 2002, 25(9): 179–202.

[10] Gao J, Ming C.S. A component testability model for verification and measurement[J]. In Proc. of the 29th Annual International Computer Software and Applications Conference, IEEE 2005, 33(12): 211–218.

[11] Mouchawrab S, Briand L.C, Labiche Y. A measurement framework for object-oriented software testability[J]. Information and Software Technology, 2005, 47(15): 979-997.

[12] Mulo E. Design for Testability in Software Systems[J]. Master's Thesis, 2007, 35(10): 101-107.