文章编号:1001-506X(2011)01-0233-04

软件可靠性度量实例研究

石 柱,郑 重
(中国航天科技集团公司软件评测中心,北京 100048)

摘 要:为适应软件的不同特点和使用阶段,必须根据软件的具体情况,考虑在软件开发的整个生命周期中,从不同角度对可靠性进行度量和评价。在现有软件可靠性度量的基础上,选择改造了 9 个可用于评价软件可靠性的度量,在航天软件中进行了应用,并详细阐述了各个度量的目标、方法和过程,最后对度量结果进行了综合分析以及给出了对软件的改进建议。

关键词:软件可靠性;软件度量;软件可靠性评价
中图分类号:TP 311.5 **文献标志码**:A **DOI**:10.3969/j.issn.1001-506X.2011.01.47

Case study on software reliability measurement

SHI Zhu, ZHENG Zhong
(*Software Evaluation and Testing Center of China Aerospace Science and Technology Corporation, Beijing 100048, China*)

Abstract: In order to adapt the different features and use stages of software, the reliability of software must be measured and evaluated from different aspects based on the specific situation in the entire software development life cycle. 9 metrics suitable to the evaluation of software reliability are chosen and constructed on a basis of existing software reliability metrics and are used in an aerospace software. The goals, methods and processes of each metric are described in detail, the results of measurements are analyzed synthetically, and the improvement suggestions are given.

Keywords: software reliability; software metric; software reliability evaluation

0 引 言

可靠软件的开发需要从项目一开始就对开发过程和软件产品进行监控,软件度量是一种对软件可靠性进行监控的重要手段[1]。现用于评价软件可靠性的度量大都仅限于应用可靠性模型,这些可靠性模型严重依赖于在开发结束时才适用的软件测试数据,在航天等领域中,由于失效较少,成本昂贵,很难或者无法收集这些数据,在分析和评价软件的可靠性时可以采用不同的可靠性度量指标来评价软件产品的可靠性。

本文度量的对象是一个航天实时嵌入式软件,采用自顶向下结构化的设计方法,依据功能依次分解为相应的功能模块。整个软件由万余行代码,共 80 多个模块构成,其中编程语言是 C 和汇编,此次度量工作只考虑 C 语言部分的代码。

1 度量的选取

截至目前,已有近百种有关可靠性度量的文献发表,在文献[2-6]中也提出了近百种可靠性度量指标,但在实际应用中,使用所有度量是不现实的,也不必要,必须考虑数据收集的困难程度,以及经费、进度问题。另一方面,这些标准文献中的很多度量都没有实际应用过,缺乏实际经验数据的支持。因此,在使用这些度量前,必须根据实际项目的特点和需求选择并改造这些度量,以满足实际项目应用。

本文对度量的选取及改造的原则如下[7-8]:

(1) 所选度量之间没有交叠信息。如在文献[6]中的度量"故障天数"是"平均故障间隔时间"的一个子度量,即故障天数除以故障数就可作为平均故障间隔时间的一个粗估计值;

(2) 度量的结果是唯一的,没有二义性,即指在相同环境和相同规则下,相同或不同人使用该度量所得的结果都一样;

(3) 所选择的度量要覆盖软件的整个生存周期。

根据以上度量选取原则,考虑到该航天软件的特点,经过选择和改造,本文应用以下 9 个度量:需求可追踪性、需求错误率、圈复杂度、信息流(或数据流)复杂性、需求覆盖

收稿日期:2009-06-04; **修回日期**:2009-12-11。
基金项目:国防科技工业局民用航天科研预先研究项目(B2120080307)资助课题
作者简介:石柱(1963-),男,研究员,博士,主要研究方向为软件工程、软件可靠性、软件安全性和软件过程改进。
E-mail:shizhu12s@sohu.com

率、缺陷密度、基于软件故障树分析(software fault tree analysis, SFTA)的缺陷指数、基于软件失效模式与影响分析(software mode effects analysis, SFMEA)的故障避免率、平均失效前时间(mean time to failure, MTTF)。

2 度量的应用

在进行度量之前,首先应该明确管理人员或开发人员对可靠性目标的要求。通过与用户沟通和项目需求,此次度量的目标如表 1 所示。从表 1 可知,有些度量没有给出具体的目标值,是因为该软件还没有度量方面的经验数据,只能给出目标趋势;再者,有些度量贯穿于软件开发的整个生命周期的多个阶段,是连续性度量,可以根据两个阶段度量值的对比,看其有无提高的趋势。

表 1 度量目标

度量	目标
需求可追踪性/(%)	100
需求错误率	低
圈复杂度	≤10
数据或信息流复杂性	低
需求覆盖率/(%)	100
缺陷密度	低
基于 SFTA 的缺陷指数	低
基于 SFMEA 的故障避免率/(%)	100
MTTF/d	180

注:MTTF 的置信度为 0.6。

2.1 需求可追踪性

为了解用户需求是否在设计中全部体现,需要考察该软件的需求可跟踪性。通过对软件需求规格说明和概要设计的审查,发现概要设计中的需求项(共 24 项)完全覆盖了用户的初始需求(共 24 项),故需求可跟踪性 X 为

$$X = \frac{24}{24} \times 100\% = 100\% \tag{1}$$

从度量结果可以看出,需求的可跟踪性很好,概要设计已经完全覆盖了用户的初始需求,有利于进一步的详细设计。

2.2 需求错误率

设 N 为总需求数,并将需求错误类型分为三种,即与初始需求不一致、未准确反映软件需求的,不完备的、未完全反映软件需求的,以及曲解的、不正确反映软件需求的,分别表示为 N_1、N_2 和 N_3。

总需求数 N 为 24,通过对需求规格说明和各个模块所实现的需求进行对比,发现与初始需求不一致的地方 11 处,不完备的地方 0 处,曲解的地方 2 处,需求错误率 R 为

$$R = \frac{N_1 + N_2 + N_3}{N} = \frac{13}{24} = 54.2\% \tag{2}$$

分析度量结果可以看出,软件实现的功能中与初始需求不一致的现象较为严重,而从软件的需求可跟踪性中可以看出,需求与概要设计的一致性较好,因此可以确定软件开发人员在实现过程中没有严格遵循前期的概要设计。

2.3 圈复杂度

本度量可以确定已编码模块的结构复杂性,开发者根据这个度量可以限制模块的复杂性,保持模块大小在可理解性、易编程的范围内,从而有利于提高软件的质量和可靠性。

分别对该软件模块的圈复杂性进行度量,结果发现 58 个模块的圈复杂性都在 10 以内,25 个模块的圈复杂度超过了 10,因此需要对代码进行进一步的模块化,特别是圈复杂度较高的模块。

另外需要考虑的是,圈复杂度是用来限制模块内的复杂性,因而可能会增加完成给定功能所需的模块数,从而增加模块间的复杂度,所以应该同时考虑模块内和模块间的复杂性,并进行优化,而不是只考虑某一方面。

2.4 信息流(或数据流)复杂性

这个度量是用来指示子系统之间关系的简单程度,通过此度量,开发者能够为了获得可靠系统而力求较低的信息流复杂性,计算方法为

$$IFC = (fanin \times fanout)^2 \tag{3}$$

$$WIFC = length \times (fanin \times fanout)^2 \tag{4}$$

式中,IFC(information flow complexity)是信息流复杂性;$WIFC$(weighted information flow complexity)是加权的信息流复杂性;$length$ 是源程序可执行代码行数(不包括注释);$fanin$ 是模块的扇入数;$fanout$ 是模块的扇出数。

通过检查各个模块的扇入和扇出,计算出各个模块的信息流复杂性,如表 2 所示。

表 2 信息流复杂性表

模块	可执行代码行数	扇入	扇出	信息流复杂性	加权的信息流复杂性
1	16	1	0	0	0
2	112	1	0	0	0
3	100	1	1	1	100
4	21	0	1	0	0
5	47	1	2	4	188
6	72	1	3	9	648
7	163	0	15	0	0
8	379	1	23	529	200 491
9	225	0	13	0	0
10	50	1	4	16	800
…	……	……	……	……	……

从表 2 中可以看出,该软件的模块扇入值普遍偏小,而模块扇出值普遍偏大,这与“多扇入,少扇出”的模块设计原则不相符。模块的扇入大说明该模块的复用性越好;模块的扇出大可能在设计该模块时需要考虑的问题就越多,因而复杂性越高。而且个别模块的信息流复杂性很高,说明该模块过分复杂,需要进行重点测试或重新设计。

另一方面,通过对代码走查发现,该软件模块的功能封装性差,功能不单一,有可能是造成模块信息流复杂度大的重要原因。

2.5　需求覆盖率

需求覆盖率是测试充分性的一个重要指标,是功能测试要达到的目标之一。需求覆盖率是指在软件功能测试中是否将用户的需求项全部包含。经过审查发现软件需求的总数为 24,已被测试的需求项为 24,故该软件的需求覆盖率 TC 为

$$TC = 24/24 \times 100\% = 100\% \quad (5)$$

2.6　缺陷密度

软件缺陷是存在于软件中的、不期望的或不可接受的偏差,其结果是当软件运行于某一特定条件时将出现软件故障(即软件缺陷被激活)。软件缺陷以一种静态的形式存在于软件的内部,是软件开发过程中人为错误的反映。本文若无特殊声明,主要是指程序中的缺陷。

通过对软件进行代码走查和功能测试,共发现缺陷 164 个,每个模块的缺陷密度如图 1 所示。

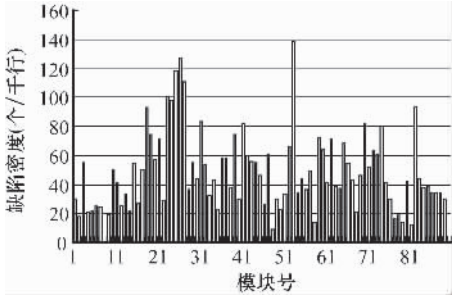

图 1　各个模块的缺陷密度分析

由图 1 发现,模块 53 的缺陷密度最大,达到 138,有 4 个模块的缺陷密度达到 100 以上,需要重点分析。

通过对缺陷的产生原因进行分析,发现其主要包括以下 7 种类型:抗单粒子翻转设计、共享资源冲突、读写越界、计算异常、算法设计、软硬件接口、编码规范性。按照缺陷的类别进行统计,缺陷分布如图 2 所示。

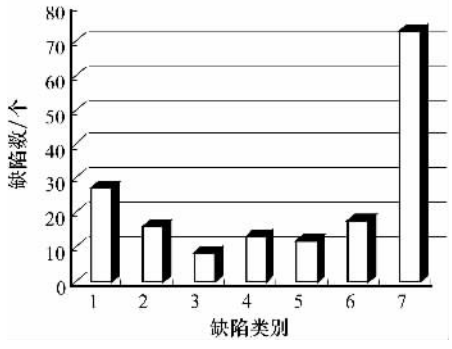

图 2　缺陷分布图

由图 2 可以看出,大部分的缺陷是由编程不规范造成的,提示开发人员应该严格依据编程规范改进程序中的不符合项。

由于对于该类型软件目前没有缺陷密度的可靠性范围值的经验,因此此阶段的缺陷密度可作为一个基线值,在对软件进行修改后做回归审查和测试后,再对其缺陷密度值进行度量,以监测可靠性是否增长。若各个模块的缺陷密度有明显降低,则可认为修改过程是成功的,软件的可靠性实现了增长。

2.7　基于 SFTA 的缺陷指数

本度量作为一个过程度量,是随着软件错误的不断排除而不断变化的。本次 SFTA 的顶事件是一个失效,在其最小割集中,只要有一个或一个以上的底事件发生,则顶事件必发生。设在 SFTA 阶段中所有的 58 个底事件中,能导致一个以上的顶事件发生的底事件(X_1)为严重缺陷,能导致一个顶事件发生的底事件(X_2)为次要缺陷,与其他底事件组合才能导致顶事件发生的底事件(X_3)为轻微缺陷。X_1 的个数为 8 个,X_2 的个数为 47 个,X_3 的个数为 3 个。本文采用推荐的缺陷权值,即 X_1 的权值为 10,X_2 的权值为 3,X_3 的权值为 1,则基于 SFTA 的缺陷指数 P 为

$$P = (8 \times 10 + 47 \times 3 + 3 \times 1)/58 = 3.86 \quad (6)$$

该缺陷指数的目标是:随着软件缺陷的排除,其值越来越小,可将此缺陷指数值作为一个基线值,在后续版本更新中判断缺陷排除过程是否充分有效的指标。

2.8　基于 SFMEA 的故障避免率

在设计阶段,对该软件进行了 SFMEA,共提出了 64 种故障模式,其中Ⅰ类(关键的)故障模式 9 个,Ⅱ类(重要的)故障模式 10 个,Ⅲ类(一般的)故障模式 17 个,Ⅳ类(轻度的)故障模式 28 个。本文重点对关键的和重要的故障模式进行分析,发现已经采取措施避免失效发生的故障模式数为 12,据此计算故障避免率 F。计数避免的故障模式数并与要考虑的故障模式数比较。

$$F = A / B = 12/19 = 63.1\% \quad (7)$$

式中,A 是在设计/编码过程中已经采取措施避免的故障模式数;B 是要考虑的故障模式数。

从中可以看出,该软件的故障避免程度还有待提高,应该根据 SFMEA 后的结果和建议对程序的设计和代码进行修改,以消除这些故障模式。

2.9　平均失效前时间

通过对软件实际运行情况的监测,该软件在 6 个月时间内共发生 5 次失效,其失效数据集合如表 3 所示。

表 3　失效数据集

失效数	失效间隔时间/d	失效累积时间/d
1	23	23
2	71	94
3	7	101
4	1	102
5	4	106

在计算 MTTF 时,本文采用 J-M(Jelinski-Moranda)模型和 G-O(Goel-Okumoto)模型,这两个模型是马尔可夫过程类模型和非齐次泊松过程类模型的典型代表,与其他同类模型有较大相似性,且在以往的可靠性预测工作中有较好表现,应用广泛[9]。为选择预测能力最好的模型,需要检验模型的预测质量,本文采用序列似然度比率来检验不同

模型的预测精度，u-结构图和 y-结构图分别检验模型的偏差和偏差趋势[10]，这三种值越小表示模型的预测能力越好，最后得出的结果如表 4 所示。

表 4 度量结果

模型	失效率	平均失效前时间/d	预测精度	偏差	偏差趋势
J-M	0.712	34.582	511.691	0.355	0.319
G-O	0.894	27.932	481.312	0.532	0.297

由表 4 可见，G-O 模型的预测质量比 J-M 模型好，因此选择 G-O 模型的预测值作为此次度量的结果，即失效率为 0.894，平均失效前时间为 27.932。

3 度量结果分析

通过以上度量的应用，该软件总的度量结果如表 5 所示。

表 5 度量结果列表

度量指标	度量结果
需求可追踪性/(%)	100
需求错误率/(%)	54.2
圈复杂度	部分模块过大
信息流(或数据流)复杂性	部分模块过大
需求覆盖率/(%)	100
缺陷密度	部分模块过大
基于 SFTA 的缺陷指数	3.86
基于 SFMEA 的故障避免率/(%)	63.1
MTTF/d	27.932

由于该软件缺少可靠性度量方面经验，无法给出这些度量的目标值，结合该项目的特点给出对该软件可靠性的定性评价和改进建议：

(1) 经过统计，缺陷密度较大的模块的信息流复杂性也较大，但这些模块的圈复杂性并不大，因此后期改进时需要对其进行模块化分解，使得各个模块的功能尽量单一，也有利于以后的代码维护工作。

(2) 结合缺陷类别的分布图可以发现，由于编程规范而造成软件缺陷的比例较大，在对软件进行修改的同时，开发人员的编程规范也需要改进，在考虑成本的情况下，应开展对开发人员的编程能力培训。但这并不代表由编程规范造成的缺陷对可靠性的影响大，还应该考虑缺陷的严重性程度，由前阶段的 SFMEA 发现，共享资源冲突和读写越界缺陷对可靠性影响较大，对其修改可使可靠性有显著增长。

(3) 基于 SFMEA 的故障避免率没有达到 100%，预示着软件在预防失效方面需要加强，结合软件故障树分析结果，对可能造成失效的严重和重要缺陷加以改正，所有这些修改都应考虑成本和时间的约束，如：对该软件架构进行修改，则需充分考虑风险和成本问题。

(4) 平均失效前时间没有达到预定的要求 6 个月(约 180 d，置信度为 0.6)，需要进行软件可靠性增长测试，将以上发现的缺陷在测试中复现出来，并针对具体的问题和缺陷进行改正，最后再进行可靠性验证测试，检查可靠性指标是否达标。

(5) 需求覆盖率和需求可跟踪性只说明了设计、测试与需求的一致性较好，虽然达到了 100%，并不能说明软件需求不存在问题，而且从需求错误率可以看出，软件实现的功能与需求有出入，需要检查需求的正确性。

(6) 所有这些度量值可作为过程和产品的基线值，用于以后过程能力的改进，可能由于度量过程的不稳定，会带来度量值的振荡，但此时仅仅是将这些数据作为一个参考，用于组织目标的设定和改进，经过时间和数据的不断积累，基于稳定的过程基础之上会产生越来越多的可用的数据并生成准确而有参考意义的度量值，这些度量值可作为以后生产同等类型软件的可靠性目标。例如，对于圈复杂度来说，推荐的最佳圈复杂度是小于 10，但该软件 30%模块的圈复杂度都大于 10，而且缺陷密度也不高，因此可将圈复杂度的基线值作适当调整。

4 结 论

从第 3 节分析可知，推荐的软件可靠性度量对于该软件项目具有可操作性和实用性，为软件质量保证人员验证软件的可靠性提供了一条途径，也需要在以后的可靠性度量实践过程中不断完善软件可靠性指标体系。另一方面，软件度量的过程不是一个简单的单向过程，度量结果指示着对产品和过程的改进，而改进后的软件可靠性还需要度量的检验和监控，如此动态变更的过程，才能保证软件的可靠性确实得到提高。

参考文献：

[1] Fenton N E, Pfleeger S L. *Software metrics: a rigorous and practical approach*[M]. Beijing: Tsinghua University Press, 2003.

[2] Lyu M R. Software reliability engineering: a roadmap[C]// *Proc. of Future of Software Engineering*, 2007: 153-170.

[3] IEEE Std 982.1-2005. IEEE standard dictionary of measures of the software aspects of dependability[S]. New York: IEEE Computer Society Press, 2005.

[4] Krajcuskova Z. Software reliability models[C]// *Proc. of 17th International Conference*, 2007: 1-4.

[5] Ehrlich W K, Lee S K, Molisani R H, et al. Applying reliability measurement: a case study[J]. *IEEE Software*, 1990, 7(2): 56-64.

[6] Srivastava V K, Farr W, Ellis W. Experience with the use of standard IEEE 982.1 on software programs[C]// *Proc. of Engineering of Computer-Based Systems*, 1997: 121-127.

[7] 袁心成，石柱. 软件可靠性度量在航天控制软件中的应用[J]. 航天控制，2006，24(3)：74-80. (Yuan X C, Shi Z. Application of software reliability metrics in aerospace control software[J]. *Areospace Control*, 2006, 24(3): 74-80.)

[8] 石柱，袁心成，马卫华，等. 适用于航天软件开发的可靠性度量[J]. 航天控制，2004，22(3)：87-92. (Shi Z, Yuan X C, Ma W H, et al. Reliability metrices suitable for aerospace software development[J]. *Areospace Control*, 2004, 22(3): 87-92.)

[9] Almering V, Van Genuchten M, Ger C, et al. Using software reliability growth models in practice[J]. *IEEE Software*, 2007, 24(6): 82-88.

[10] Lyu M R. *Hand book of software reliability engineering*[M]. New York: McGraw-Hill, 1996: 168-189.