第十一章　软 件 可 靠 性

内 容 提 要

§11-1 软件可靠性的基本概念

一、软件的环境条件　　二、时间的度量

三、 软件的故障　　四、影响软件可靠性因素

§11 - 2　软 件 可 靠 性 的基 本 特 征 量

1. 系统不工作次数
2. 系统平均不工作间隔时间（MTBD）
3. 有效性 (A)
4. 平均修复时间（MTTR）
5. 平均不工作时间（MDT）　6. 初期故障率
7. 偶然故障率　　8. 使用方误用率
9. 用户提出补充要求数　　10. 处理能力

第十一章　软 件 可 靠 性

随着计算机软件的飞速发展，软件可靠性已变得越来越重要。据统计，计算机系统中，由于软件错误引起的故障占所有故障的65％。

究其原因是软件太复杂了，一个小小的程序，其可能的路径可以是天文数字，以致于在软件开发过程中难以对其作穷尽的测试，或者说难于完全排除软件缺陷。

为了说明软件的复杂性，让我们考虑一个**由10至20条高级语言构成的程序，其控制流程图如图11-1所示。**

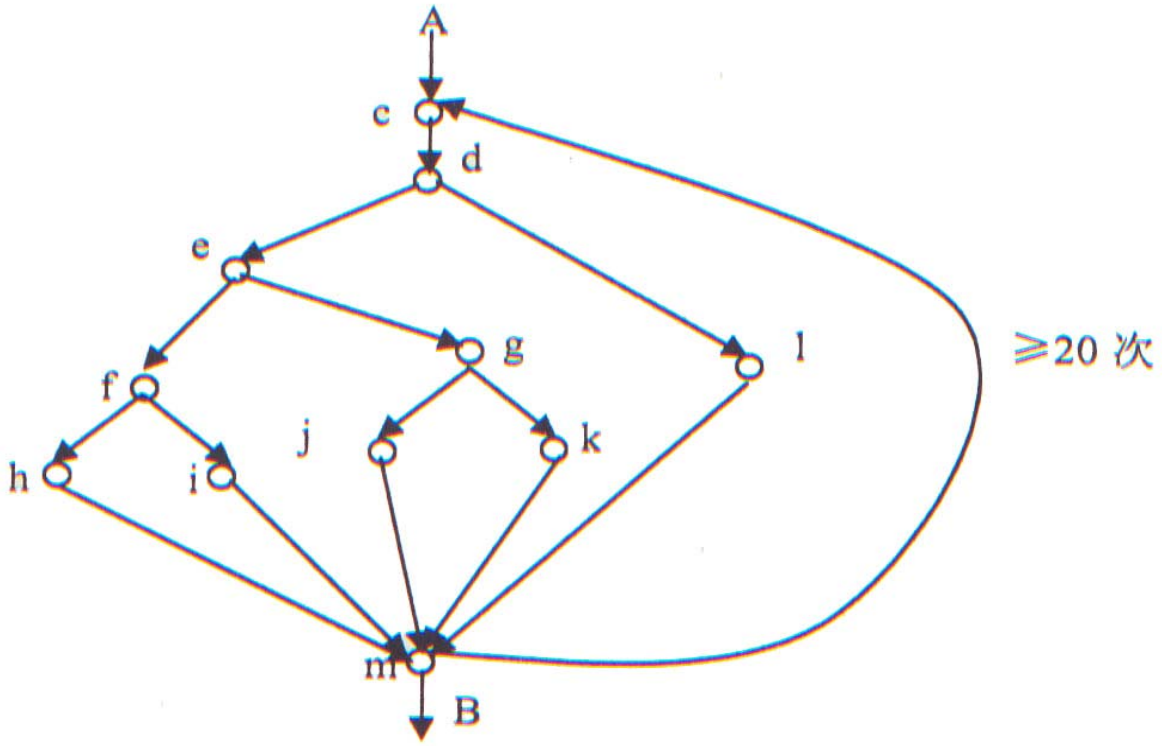

图11-1简单程序的控制流程图

图11-1中每个结点或圆圈代表一段可能以转移语句结束的顺序执行语句，每条弧代表两段程序间的控制转移。程序含有一个最少重复20次的循环语句，而在循环体内，则有一些嵌套的条件语句。假设程序中所有判断都是相互独立的，由于有**5条**贯穿循环体的路径：

即 c→d→e→f→h→m；

c→d→e→f→i→m；

c→d→e→g→j→m；

c→d→e→g→k→m；

c→d→l→m。

那么从点A到点B的所有独立路径数为：$5^{20}+5^{19}+\ldots+5^{1}$，约为$10^{14}$或$10^{16}$亿。如果考虑程序输入数据的变化，那情况就更为复杂了。

可见，软件可靠性问题在软件工程实践中极为重要，对软件可靠性问题的研究在国际上已十分活跃。

§11-1 软件可靠性的基本概念

关于 软件可靠性的确切定义，国际学术界曾经有过长期的争论。对软件可靠性定义的理解有广义和狭义两种：

广义的可靠性：

是指一切旨在避免、减少、处理、度量软件故障（错误、缺陷、失效）的分析、设计、测试方法、技术和实践活动。

与之相关的内容有软件可靠性度量、软件可靠性设计、软件可靠性建模、软件可靠性测试和软件可靠性管理等。

狭义的可靠性：

是指软件无失效运行的定量度量。

与之相关的内容有软件可靠性度、软件失效强度和软件平均失效时间等。

狭义的可靠性定义

通常用可靠度代替，在工程上是软件在规定的运行环境中和规定的时间内无故障运行的概率。

一、软件的环境条件

环境条件包括与程序存储有关的计算机及其操作系统。

例如计算机型号、字长、内存容量、外存介质的数量及容量、输入和输出设备的数量、通信网络、操作系统和数据管理系统、编译程序及其他支持软件等。

这些因素对程序的运行有很大的影响，但在使用中一般没有变化。

环境条件还包括软件的输入分布。

软件的输入有外部和内部输入：

程序在启动运行时，需要给变量赋值,即给程序提供输入数据,输入的数据可能由外部设备输入，也可能由早已存储在计算机内等待读取。

程序运行一次所需的输入数据构成程序输入空间的一个元素，这个元素是一个多维向量。全部输入向量的集合构成程序的输入空间。

一组输入数据经过程序处理后得到一组输出数据，这些输出数据构成一个输出向量，全部输出向量的集合构成程序的输出空间。

程序输入空间的元素数量非常庞大，程序运行中每个元素被选用的概率各不相同，形成一定的概率分布，我们称此为程序运行剖面，程序的不同的运行状态，对应于不同的运行剖面。

二、时间的度量

1. 日历时间

软件的测试和运行以日、周、月、年等为计时单位。

2. 时钟时间

软件从运行开始，到运行结束以时、分、秒为计时单位。其中包括等待时间和其他辅助时间，但不包括停机占用时间。

3. 执行时间

计算机在执行程序时，实际占用中心处理器（CPU）的时间，又称CPU时间。

三、 软件的故障

软件可靠性工程的主要目标是保证提高软件可靠性。为达到这一目标，显然首先要弄清软件为什么会出现故障。只有这样，才有可能在软件开发过程中减少导致软件故障的隐患，且一旦出现什么故障，有可能采取有效措施加以清除。

弄清软件故障机理是软件可靠性分析的根本目标。由于软件内部逻辑复杂，运行环境动态变化，且不同的软件差异可能很大，因而软件故障机理可能有不同的表现形式。

譬如有的故障过程比较简单，易于追踪分析，而有的故障过程可能非常复杂，难于甚至不可能加以详尽描述和分析。尤其是运行于高度复杂实时环境中的大型软件。

但总的说来，软件故障机理可描述为：软件缺陷、软件错误和软件故障。

缺陷 → 错误 → 故障(失效)

1．软件缺陷

软件开发中残留的内在缺陷称为软件缺陷。这些缺陷可以在软件生存期的各个阶段被引入。

在软件开发的各阶段，软件始终离不开人的参与，而人难免会犯错误，这样就必然给软件留下不良的痕迹。

例如一段程序进行某些数据处理，若在处理过程中就产生软件错误，则说明这段程序存在缺陷或缺少一个程序段。

软件缺陷是一个静止的现象，只在一定的输入条件下才能被激活导致软件错误，而且软件错误也不一定导致软件故障。

比如容错软件中的错误就可以被检测出来并可纠正或避免，而不导致故障。

2．软件错误

软件缺陷在一定条件下暴露并导致系统在运行中出现可感知的不正常、不正确、不按规范执行的内部状态，则认为软件出现“错误”，简称出错。

所谓不正确的内部状态，是指在此状态下，当正常的算法继续下去时，就会发生软件故障。软件错误是由于软件缺陷造成的。

一个错误可能是多个故障源。

例如，在求最大值的程序中，设计人员由于疏忽将求得的平均值作为最大值，这就是一个软件错误。

3．软件故障

在对错误不作任何纠正和恢复的情况下，导致系统的输出不满足用户提供的正式文件上指明的要求，或双方协议的条款，称为软件的一次故障。

软件故障是由于软存错误造成的一种外部表现，它是动态的、程序执行过程中出现的行为表现。

综上所述，软件缺陷是人为错误。

当一个软件缺陷被激活时，便产生一个或多个软件错误；

当软件错误不加以纠正时，便不可避免地产生软件故障。

同一个软件缺陷下可能产生不同的软件故障。

四、影响软件可靠性因素

软件可靠性因素：软件生存期内影响软件可靠性的因素。显然，有许许多多因素可以影响软件可靠性，包括技术的、社会的、经济的、甚至文化的，因为在软件生存期的各个阶段均有人的干预，而人的行为受到各方面因素的影响。

但从**技术角度**来看，影响软件可靠性的因素主要包括：

1. 运行环境(剖面)

软件可靠性定义相对于运行环境而言，同一软件在不同运行剖面下，其可靠性行为可能极不相同。

让我们考虑一个极端例子。

我们知道，软件故障是软件缺陷在一定输入情况下被激活的结果。于是可以将软件输入域划分为两个部分(G和F)：

G 中的输入**不会激活**软件的缺陷，**F** 中的输入**恒激活**软件缺陷。如果运行剖面**不包含F**中的输入，则软件不会出现故障，**其可靠性恒为1**。

反之，如果运行剖面**不包含G**中的输入，则每一输入情况下均出现故障。如果没有容错措施，则导致软件故障，**软件可靠性恒为0**。

2.软件规模

如果软件只含一条指令，那么谈论软件可靠性问题便失去意义。随着软件规模的增大，软件可靠性问题愈显突出。

在我们考虑软件可靠性问题时，软件一般是指中型以上软件(**4000～5000条以上语句**)，这时可靠性问题难以对付。

软件工程实践的一个侧面可以反映这一点，即单元测试一般由编程人员本人进行，而综合测试则需独立的测试人员。软件可靠性增长模型也主要应用于综合测试阶段。

3. 软件内部结构

软件内部结构一般比较复杂，且动态变化，对可靠性的影响也不甚清楚。

但总的说来，结构越复杂，软件复杂度越高，内含缺陷数越多，因而软件可靠度越低。

4. 软件可靠性设计技术。

关于软件可靠性设计技术的外延并不明确，但一般是指软件设计阶段中采用的用以保证和提高软什可靠性为主要目标的软件技术。如故障模式与影响分析(FMECA)、故障树分析(**FTA**)等。显然采用或不采用软件可靠性设计技术对软件可靠性必有影响。

5. 软件可靠性测试

研究表明，软件测试方法与资源投入对软件可靠性有不可忽视的影响。

6. 软件可靠性管理

软件可靠性管理旨在系统管理软件生存期各阶段的可靠性活动。

使之系统化、规范化、一体化，这样就可以避免许多人为错误，以提高软件可靠性。

7. 软件开发人员能力和经验

显然，软件开发人员（包括测试人员)的能力愈强，经验愈丰富，所犯错误便可能愈少，所得软件产品质量愈高，相应的可靠性也愈高。

8. 软件开发方法

软件工程表明，开发方法对软件可靠性有显著影响。与非结构化方法比较，结构化方法可以明显减少软件缺陷数。

9. 软件开发环境

研究表明，程序语言对软件可靠性有影响。譬如，结构化语言Ada优于Fortran语言，而软件测试工具优劣则影响测试效果。

总之，有许许多多的因素影响软件可靠性，在软件设计时应尽量采用有利于提高软件可靠性的手段和方法。

§11-2 软件可靠性的基本特征量

软件质量主要由以下几个方面因素决定。

时间因素：包括平均故障间隔时间（MIBF）、平均失效前时间（MTTF）、平均系统不工作间隔时间（MTBD）、平均修复时间（MTTR）。

缺陷频数：包括软件缺陷数、文件缺陷数和用户提出的补充要求数等。

与软件可靠性有关的百分率：主要包括可靠性、有效性、可维护性、故障率、不合格率、延迟率、错误操作率、原因不明率、同故障事件率、可靠性经济率等。

对软件的投入：包括完成软件用了不同水平的工作人员的工作日数或工时数和对软件的检查项目数及对用户提出的要求采取对策的费用等。

软件特征：包括软件的复杂性、标准化程度、寿命周期、结构及规模大小等。

使用方特征：包括使用软件的系统的特点（如实时系统、嵌入式系统），需根据软件的特点，选择适当的可靠性参数作为软件质量指标。

显然，软件质量是众多因素及指标的综合反映，在软件可靠性评估中，需根据软件的特点，选择适当的可靠性参数作为软件质量指标，软件常用的可靠性参数有以下几种：

1. 系统不工作次数

在一定时期内，由于软件故障而停止工作，必须由操作者介入再启动才能继续工作的次数称为系统不工作次数。

2. 系统平均不工作间隔时间（MTBD）

系统平均不工作间隔时间反映了系统的稳定性。

即 $$\mathrm{MTBD} = \frac{T_v}{(d+1)}$$

式中 T_v——软件系统正常工作的总时间(h)；

d——系统由于软件故障而停止工 作的次数。

3. 有效性 (A)

有效性A综合反映了系统的可靠性和维修性。

即 $$A = \frac{T_v}{T_v + T_D}$$

亦可表达为 $$A = \frac{\text{MTBD}}{\text{MTDB+MDT}}$$

式中 T_v——软件系统正常工作的总时间 (h) ;

T_D——由于软件故障使系统不工作的时间（h）；

MDT——平均不工作的时间（h）。

4. 平均修复时间（MTTR）

平均修复时间反映了出现软件缺陷后采取对策的效率。在一定程度上也反映了软件企业对社会服务的责任心。

5. 平均不工作时间（MDT）

平均不工作时间是指由于软件故障，系统不工作的均值。

6. 初期故障率

一般以软件交付使用方后的三个月内为初期故障期。

初期故障率以**每100h的故障为单位**，用它来评价交付使用时的软件质量和预测什么时候软件可靠性基本稳定。

初期故障率的大小取决于软件的设计水平、检查项目数、软件规模、软件调试彻底与否等因素。

7. 偶然故障率

一般以软件交付给使用方四个月后为偶然故障期。偶然故障率一般以每1000h的故障数为单位，它反映了软件处于稳定状态下的质量。

8. 使用方误用率

使用方不按照软件规范及说明等文件使用造成的错误叫“使用方错误”。在使用次数中，使用方误用次数占的百分率叫“使用方误用率”。

造成使用方法误用的原因之一是使用方对“说明”理解不深，操作不熟练，但也有可能是说明书没有讲清楚而引起误解。

另外还有软件系统的可操作性还应改进、对使用方的使用培训还不够深入等。

生产方有责任及时调查使用方误用的原因，对软件功能加以改进。

9. 用户提出补充要求数

用户提出补充要求数主要是反映软件未能充分满足用户的需要，有些要求是特定用户的特殊要求，生产方为了更好地为社会服务，应该尽力满足他们的要求。

有些要求是带有普遍性的，这就要求给予足够重视，因为它反映了原来的软件功能还不够全面。

10. 处理能力

处理能力有各种指标，例如，可用每小时平均处理多少文件、每项工作的反映时间多少秒等来表示，具体情况根据需要而定。

在评价软件及系统的经济效益时需用这项指标。

中国可靠性网

http://www.kekaoxing.com

感谢 kingdodoo 分享