文章编号:1001-506X(2010)04-0877-04

基于混合体系结构的软件可靠性评估方法与应用

魏 颖[1,2,3],沈湘衡[2]

(1. 北华大学计算机学院,吉林 吉林 132013;
2. 中国科学院长春光学精密机械与物理研究所,吉林 长春 130033;
3. 中国科学院研究生院,北京 100049)

摘 要:基于体系结构的软件可靠性模型建立在软件研制周期的初期阶段,能够对软件进行较早的可靠性分析,对早期软件结构的变更以及后期软件的更新与升级都提供了一定的指导依据。然而,早期的基于体系结构的软件可靠性模型只对单一的软件结构进行分析,这显然不满足如今同时存在多种体系结构风格的复杂软件的需求。分析了目前常用的软件体系结构风格,在基于混合体系结构的软件可靠性模型的基础上,阐述了应用体系结构模型进行评估的步骤,并结合实例进行了分析与验证。

关键词:软件可靠性;混合体系结构;软件可靠性评估

中图分类号:TP 31 **文献标志码**:A

Software reliability estimation and application based on heterogeneous architectures

WEI Ying[1,2,3], SHEN Xiang-heng[2]

(*1. Computer Inst. of Beihua Univ., Jilin 132013, China;*
2. Changchun Inst. of Optics, Fine Mechanics and Physics, Chinese Academy of Sciences, Changchun 130033, China; 3. Graduate School of Chinese Academy of Sciences, Beijing 100049, China)

Abstract: Architecture-based software reliability models are established during the early phase of software development. The models can perform a prior analysis of software reliability. It is a basic factor for software architecture change, edition update and upgrade. However, the early architecture-based models are most applied to homogeneous software architectures, which could not satisfy the requirement of multiple architecture software. Multiple architecture styles are firstly analyzed. Then, a heterogeneous architecture-based software reliability model is proposed. Finally, an available example is presented with the model according to the estimation steps.

Keywords: software reliability; heterogeneous architecture; software reliability estimation

0 引 言

随着现代计算机技术的不断发展,应用软件的复杂化、体系化以及规范化日趋显著。软件可靠性指标作为衡量软件质量的一个重要属性,也越来越受到人们的关注。确立软件可靠性模型是计算软件可靠性的基础与核心,在过去几十年的研究中,许多专家学者不断研究和改进了上百种的软件可靠性模型[1-2],这些模型大部分是基于软件的测试数据进行分析,建模的时间也倾向于软件研制周期的中后期。另外,这些模型大部分采用黑盒研究方法,也就是说将软件可靠性进行整体分析而不考虑模块的可靠性以及模块之间的结构联系。因此,一个小的结构上的改动就会导致系统的重新测试,软件可靠性也将重新进行分析。

1980 年,Cheung 等人提出了 Cheung 模型,该模型采用了白盒方法,是最早将模块的使用情况及可靠度的预测结合到系统软件可靠性评估过程中的模型之一。Cheung 模型假设应用程序的控制流具备唯一的入口结点和出口结点。软件模块相互独立,并且模块间的控制转换方式满足离散时间的马尔科夫过程(discrete time Markov chain, DTMC)。因此,该模型通常用来对符合 DTMC 的单一体系结构的软件进行可靠性分析,包括顺序结构、分支结构和循环结构等软件。如今,随着软件功能的日益强大,系统会

收稿日期:2008-10-08; **修回日期**:2009-06-08。

基金项目:国家高技术研究发展计划(863 计划)(2007AA703112)资助课题

作者简介:魏颖(1979-),女,讲师,博士,主要研究方向为软件可靠性评估技术与方法研究。E-mail:weiyingjl@yahoo.cn

同时采用多种不同的体系结构风格，来满足软件的高质量高性能需求。例如，并行计算方法通常被用来提高系统性能；容错设计也通常被用来满足系统的容错冗余需求。因此，单一结构的软件可靠性分析模型已经不能满足当前复杂软件的需求。

1 基于混合体系结构的软件可靠性评估模型

1.1 基于体系结构的软件可靠性模型的前提

在可靠性状态模型里，状态是指系统运行在一定条件下的一组情况或属性的集合。状态转移路径是指从一个状态到另外一个状态的控制转换，而状态转移概率则是指执行响应转移路径的概率。依据以上的定性说明，定义如下[3]：

定义 1 在 m 个模块具备相同触发条件的前提下，可靠性模型的状态表示为 $s_i=\{C_i\}$，其中，C_i 表示 m 个软件模块的集合 $\{c_1^i, c_2^i, \cdots, c_m^i\}$。

定义 2 可靠性模型 S_M 由一个四元组 $(Q, s_1, s_k, \boldsymbol{M})$ 表示，其中，Q 是 k 个状态的有限集合 $\{s_1, s_2, \cdots, s_k\}$；$s_1$ 是转换过程中的初始状态；s_k 是在 Q 集合中最后到达的状态；$\boldsymbol{M}$ 是 $k\times k$ 转换矩阵，矩阵元素 $M(i,j)$ 表示从状态 s_i 成功转移到状态 s_j 的转换概率。

构造矩阵 $\boldsymbol{M}$ 是建立软件体系结构可靠性模型最重要的一个步骤，需要同时考虑到软件的结构和软件的运行方式。在 Cheung 模型中矩阵 $\boldsymbol{M}$ 被表示为

$$\begin{cases} M(i,j)=R_iP_{ij}, \text{状态 } s_i \text{ 可达状态 } s_j, & 1\leqslant i,j\leqslant k \\ M(i,j)=0, \text{其他} \end{cases} \tag{1}$$

式中，R_i 表示状态 i 的可靠度；P_{ij} 表示从状态 i 到状态 j 的转移概率。当状态 s_i 成功转移到状态 s_j 时不只需要 s_i 的成功运行，同时还要执行 i 到 j 的转移路径，因此，当状态 i 可达状态 j 的概率即：$M(i,j)=R_iP_{ij}$。

系统的可靠度表示从开始状态 s_1 成功到达最终状态 s_k 的所有可能性。因此系统可靠度的计算过程为

$$R=Q(1,k)R_k \tag{2}$$

式中，R 为系统可靠度；R_k 是结束状态 s_k 的可靠度；$Q(1,k)$ 表示从状态 s_1 运行到状态 s_k 的所有可能性，即

$$Q(1,k)=\sum_{k=0}^{\infty}M^k(1,k)=(-1)^{k+1}\frac{|(\boldsymbol{I}-\boldsymbol{M})_{k,1}|}{|\boldsymbol{I}-\boldsymbol{M}|} \tag{3}$$

式中，$(\boldsymbol{I}-\boldsymbol{M})_{k,1}$ 表示矩阵 $(\boldsymbol{I}-\boldsymbol{M})$ 去掉第 k 行第 1 列后的矩阵。根据式(2)和式(3)可推导出系统软件可靠度为

$$R=(-1)^{k+1}\frac{|(\boldsymbol{I}-\boldsymbol{M})_{k,1}|}{|\boldsymbol{I}-\boldsymbol{M}|}R_k \tag{4}$$

具体的推导步骤参见文献[1]。

1.2 各种体系结构风格的模块到状态的转换

1.2.1 批处理序列结构

批处理序列是一种最常见、结构最为简单的软件体系结构。批处理中的每个模块都是一个独立的程序，数据以整体的形式在模块间传输。在此结构中模块是顺序执行的，在某一执行时间内只有一个模块在运行，当前模块执行完以后，下一个模块才会运行。批处理结构中的模块与可靠性模型中的状态采取一一对应的关系。即状态转移矩阵 $\boldsymbol{M}$ 表示为

$$\begin{cases} M(i,j)=R_iP_{ij}, \text{状态 } s_i \text{ 直接可达状态 } s_j, & 1\leqslant i,j\leqslant k \\ M(i,j)=0, \text{其他} \end{cases} \tag{5}$$

式中，状态可靠度 R_i 与模块 c_i 的可靠度 r_i 相等；状态转移概率 P_{ij} 指的是模块 i 到模块 j 的转移概率。

1.2.2 并行处理结构

并行处理结构是指在并行的执行环境中，模块得以并发执行来提高软件的性能。它既可以指多处理器环境下软件的运行，也包括单处理器中多进程的并发执行。在并行处理结构中，存在一组模块并行工作来完成任务。在构建可靠性模型时，将这组模块合并成一个状态来分析。状态的运行范围从这组并行模块第一个执行开始到最后一个模块结束运行而终止。状态 s_i 的可靠度计算为

$$R_i=\begin{cases} r_a, \text{状态 } s_i \text{ 中只包含模块 } c_a \\ \prod r_a, \text{状态 } s_i \text{ 中包含并行模块 } \forall c_a \end{cases}, \quad 1\leqslant i\leqslant m \tag{6}$$

式中，m 表示状态的总数。当 s_i 状态中只包含一个模块 c_a 时，s_i 的可靠度 R_i 等于模块 c_a 的可靠度 r_a；当 s_i 状态中包含多个并行模块时，R_i 等于多个并行模块可靠度的乘积。也就是说，只有当状态内所有模块都正常运行的前提下，该状态才是可靠的。由此可以推出转换矩阵的计算式为

$$\begin{cases} M(i,j)=R_iP_{ij}=r_aP_{ij}, & s_i \text{ 中只包含模块 } c_a \\ M(i,j)=R_iP_{ij}=\prod_{a=1}^{k}r_a, & s_i \text{ 含 } k \text{ 个并行模块}, 1\leqslant i,j\leqslant m \\ M(i,j)=0, & \text{其他} \end{cases} \tag{7}$$

1.2.3 容错结构

容错结构是指软件的一种冗余设计，它是用多个功能相同但设计不同的模块来同步执行相同的任务，以此保证软件运行的可靠性，提高软件的工作质量。在容错结构中，存在一组功能相同的模块，这组模块有主份和备份之分。在程序运行过程中当主份模块失效时，备份模块及时顶替主份模块，使软件继续正常运行。如果新的主份模块也失效，则继续由其他备份模块依次替换。由于这组模块采用不同算法或结构进行设计，因此模块的可靠度也可以是不同的。容错结构模块与状态之间的转换方法类似于并行处理结构。对单一执行的模块用单一的状态来表示；否则，对一组容错执行的模块则合并成一个状态来表示。其中，状态运行范围从主份模块开始到所有备份模块结束。

状态对应的可靠度计算为

$$R_i=\begin{cases} r_a, s_i \text{ 中只包含模块 } c_a \\ 1-\prod(1-r_a), s_i \text{ 含容错模块 } \forall c_a \end{cases}, \quad 1\leqslant i\leqslant m \tag{8}$$

当 s_i 中包括多个容错模块 c_a 时,$R_i=1-\prod(1-r_a)$,只要该组内存在一个正常运行的容错模块,则状态 s_i 就是可靠的。由此可以推出转换矩阵的计算式为

$$\begin{cases} M(i,j)=R_iP_{ij}=r_aP_{ij},\ s_i \text{ 只包含模块 } r_a \\ M(i,j)=R_iP_{ij}= \\ \left(1-\prod_{a=1}^{k}(1-r_a)\right)P_{ij},\ s_i \text{ 含 } k \text{ 个容错模块} \\ M(i,j)=0,\ \text{其他} \end{cases} \tag{9}$$

式中,$1\leqslant i,j\leqslant m$。

2　基于体系结构模型的软件可靠性评估

2.1　基于体系结构模型的可靠性评估过程与步骤

基于体系结构的软件可靠性模型评估的一般步骤为:确定软件结构,分解并划分系统功能模块;定义模块的失效行为,估计模块的可靠度及模块间的状态转移概率;结合软件可靠性模型与软件的体系结构对系统可靠性进行评估。

为了获得稳定而有效的测试数据,通常在软件开发周期的系统测试阶段进行可靠性评估。图 1 细化了软件可靠性评估的具体步骤,由于基于体系结构的可靠性分析需要动态执行软件,因此在确定软件结构并分解系统模块之后需要执行测试用例。图 1 中,左分支通过测试工具的记录与计算,估计出模块间的控制转移概率;右分支表示分析导致系统失效的测试用例的执行情况,将错误定位到具体失效模块,累计模块的失效数,从而估计模块的可靠度。最后,应用可靠性分析模型,评估软件可靠性。

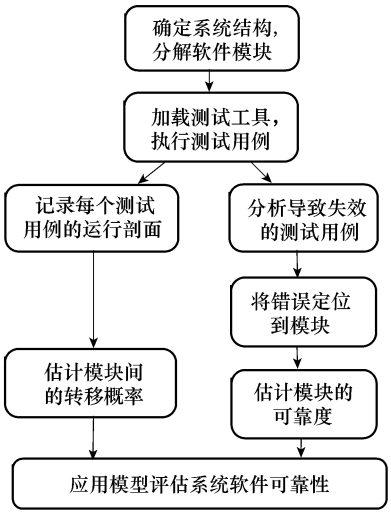

图 1　软件可靠性模型评估的步骤

2.2　实例分析

超光谱通信监控软件是某航空载荷系统的一部分,是连接超光谱仪与总线控制器之间的承接软件,负责完成超光谱仪的通信与监控任务。该通信监控软件的主要功能模块有:指令处理、接口控制、子系统控制等模块。其中,指令处理模块又被分为:任务调度、数据注入处理和程控指令接收部分。子系统控制模块除了接收来自任务调度模块输出的子系统任务调度信息,还有程控指令接收模块的程控指令。子系统控制模块采取了并行结构设计,可并行监控和处理两个子系统的状态与命令。接口控制模块接收数据注入处理模块发送的数据注入信息与接口控制命令。接口设备在硬件上采取了主备份工作模式,在软件上也采用了功能相同的容错冗余设计。软件的系统结构如图 2 所示。

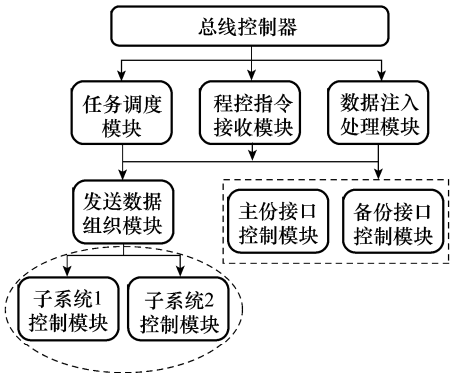

图 2　超光谱通信监控软件主要功能框架

构造软件的体系结构需要考虑模块的划分以及模块之间的调用关系。因此,在软件研制初期要考虑两个因素:一个是软件的结构风格是否满足状态模型的条件,它们之间的对应关系如何;另一个是要权衡软件模块在应用中的划分。模块划分得过细过多而产生的庞大状态空间,会导致软件可靠性模型参数很难确定从而增加评估难度;而模块划分得过粗过少也会造成由于忽略模块的分布差异而使失效数据丢失,从而影响可靠性评估的准确性。依据这两个因素,将软件划分为 10 个模块,如图 3 所示。其中 C_5 和 C_6 是容错冗余模块,C_8 和 C_9 是并行执行模块,其他模块均顺序执行。

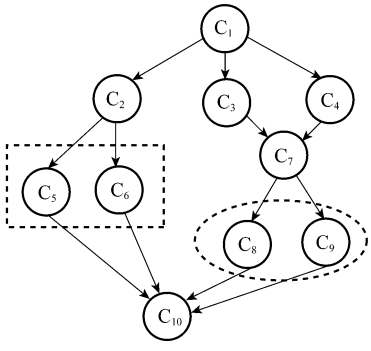

图 3　软件结构图

依据 1.2 节中提到的模块与状态的转换关系,可以将软件的结构框图转化为状态转移框图,如图 4 所示,其中将 C_5 和 C_6 合并为一个状态,C_8 和 C_9 合并为一个状态,其他的模块与状态一一对应。

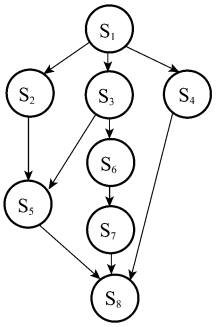

图 4 状态转换图

转移概率 P_{ij} 是指从模块 C_i 到模块 C_j 的控制转换概率。在串行结构中，$P_{ij}=n_{ij}/n_i$，其中，n_{ij} 为软件单次运行过程中模块 C_i 到 C_j 的转换次数，$n_i=\sum_j n_{ij}$。在并行结构和容错冗余结构中，将具备相同性质的模块组合成一个状态，即假设模块 C_i（状态集合外的模块）到模块 C_j（状态集合内的模块）的转移概率为 1。根据以上求法得到系统各模块的转移概率如下：

$$P_{1,2}=0.3,\ P_{1,3}=0.6,\ P_{1,4}=0.1,\ P_{2,5}=P_{2,6}=1,$$
$$P_{3,5}=P_{3,6}=0.2,\ P_{3,7}=0.8,\ P_{4,7}=0.6,\ P_{4,10}=0.4,$$
$$P_{7,8}=P_{7,9}=1,\ P_{5,10}=P_{6,10}=P_{8,10}=P_{9,10}=1$$

在体系结构模型中通常假设软件各模块是相互独立的，并且模块的失效会导致系统失效。因此在实际应用中，可以通过已有软件测试工具测试并记录模块的失效数 f_i 和模块在系统单次运行过程中的调用次数 n_i。计算模块可靠度的简单公式即为 $R_i=1-f_i/n_i$。该软件各模块的可靠度预测值如下：

$$R_1=0.998,\ R_2=0.975,\ R_3=0.995,\ R_4=0.980,$$
$$R_5=0.985,\ R_6=0.990,\ R_7=0.996,$$
$$R_8=0.990,\ R_9=0.990,\ R_{10}=0.992$$

依据式(5)～式(9)，计算转换概率矩阵 $\boldsymbol{M}$ 值如下：

$$M(1,2)=R_1P_{1,2}=0.998\times0.3=0.299\ 4$$
$$M(1,3)=R_1P_{1,3}=0.998\times0.6=0.598\ 8$$
$$M(1,4)=R_1P_{1,4}=0.998\times0.1=0.099\ 8$$
$$M(2,5)=R_2P_{2,5}=0.975\times1=0.975$$
$$M(3,5)=R_3P_{3,5}=0.995\times0.2=0.199$$
$$M(3,6)=R_3P_{3,7}=0.995\times0.8=0.796$$
$$M(4,6)=R_4P_{4,7}=0.980\times0.6=0.588$$
$$M(4,8)=R_4P_{4,10}=0.980\times0.4=0.392$$
$$M(5,8)=(1-(1-R_5)(1-R_6))P_{5,10}=0.999\ 85$$
$$M(6,7)=R_7P_{7,8}=0.996\times1=0.996$$
$$M(7,8)=R_8R_9P_{8,10}=0.990\times0.990\times1=0.980\ 1$$

$k=8$，$|(\boldsymbol{I}-\boldsymbol{M})_{k,1}|=-0.972\ 7$，$|(\boldsymbol{I}-\boldsymbol{M})|=1$，$R_{10}=0.992$

系统可靠性的值为

$$R=(-1)^{k+1}\frac{|(\boldsymbol{I}-\boldsymbol{M})_{k,1}|}{|\boldsymbol{I}-\boldsymbol{M}|}R_k=0.964\ 9$$

3 结束语

本文提到的混合体系结构软件可靠性模型改进了以往只能分析单一顺序结构的软件可靠性模型，符合目前复杂系统的软件可靠性需求，尤其适合一些基于组件的实时监控系统。但是，该模型还存在着一定的不足。由于对体系结构分析时假设软件运行符合 Markov 特性，模块之间的转移是随机的，而现实的软件中会存在很多模块运行不随机的情况，例如：一个模块会按照事先的顺序依次调用其他模块。另外，软件模块可靠度以及转移概率的准确性也直接影响了系统的可靠性。

参考文献：

[1] Katerina G P, Trivedi KS. Architecture-based approach to reliability assessment of software systems[J]. *Performance Evaluation*, 2001(45): 179 - 204.

[2] Lyu M R. *Handbook of software reliability engineering*[M]. IEEE Computer Society Press and McGraw-Hill Book Company, 1996.

[3] Wang W L, Dai P, Chen M H. Architecture-based software reliability modeling[J]. *Journal of Systems and Software*, 2006 (79): 132 - 146.

[4] Wang W L, Tang M H, Chen M H. Software architecture analysis: a case study[C]// *Proc. of 23rd IEEE International Computer Software and Applications Conference*, 1999: 265 - 270.

[5] Hamlet D, Mason D, Woit D. Theory of software reliability based on components[C]// *Proc. of 23rd International Conference on Software Engineering*, 2001: 361 - 370.

[6] Gokhale S S, Wong W E, Horgan J R, et al. An analytical approach to architecture-based software performance and reliability prediction[J]. *Performance Evaluation*, 2004(58): 391 - 412.

[7] Katerina G P, Hamill M, Perugupalli R. Large empirical case study of architecture-based software reliability[C]// *Proc. of the 16th IEEE International Symposium on Software Reliability Engineering*, 2005: 43 - 53.

[8] Reussner R H, Schmidt H W, Poernomo I H. Reliability prediction for component-based software architectures[J]. *Journal of Systems and Software*, 2003(66): 241 - 252.