基于构件软件的可靠性通用模型*

毛晓光+, 邓勇进

(国防科学技术大学 计算机学院,湖南 长沙 410073)

A General Model for Component-Based Software Reliability

MAO Xiao-Guang+, DENG Yong-Jin

(School of Computer, National University of Defense Technology, Changsha 410073, China)

+ Corresponding author: Phn: +86-731-4573673, Fax: +86-731-4573637, E-mail: xgmao@nudt.edu.cn, http://www.nudt.edu.cn

Received 2003-01-09; Accepted 2003-06-30

Mao XG, Deng YJ. A general model for component-based software reliability. *Journal of Software*, 2004,15(1):27~32.

http://www.jos.org.cn/1000-9825/15/27.htm

Abstract: The approach of aggregating components into complex software systems is maturing with the rapid development of component technology. How to analyze software reliability from system architecture and components' reliabilities should be answered. Software is static, while development process is dynamic. To enable reliability tracing through a dynamic process, this paper presents a general model for component-based software reliability-component probability transition diagram-based on function abstractions. Different from other related work emphasizing on mathematical modeling, the model presented here focuses on reliability tracing through the dynamic development process.

Key words: component-based software; software component; software reliability; model

摘 要: 随着软构件技术的快速发展,聚集软构件设计复杂软件系统的软件开发方法日趋成熟.如何利用系统架构和软构件的可靠性分析软件系统的可靠性成为一个亟待解决的问题.软件是静态的,而开发过程是动态的.为了在动态的开发过程中跟踪可靠性,以函数抽象为基础,提出了基于构件软件的一个可靠性通用模型——构件概率迁移图.与偏重于数学建模的相关工作不同,该模型更关注于动态开发过程中的可靠性跟踪.

关键词: 基于构件软件;软构件;软件可靠性;模型

中图法分类号: TP311 **文献标识码**: A

随着软构件技术的快速发展,聚集软构件设计大型复杂软件系统的软件开发方法日趋成熟.但是,目前的工作更多地集中在软构件开发以及软构件的复用技术等方面,而互补的可靠性等质量方面则较少得到关注[1].如何利用系统架构和软构件的可靠性估计软件系统的可靠性成为一个亟待解决的问题.

* Supported by the National Natural Science Foundation of China under Grant No.60233020 (国家自然科学基金); the National High-Tech Research and Development Plan of China under Grant No.2001AA113190 (国家高技术研究发展计划(863))

作者简介: **毛晓光**(1970－),男,浙江江山人,博士,副教授,主要研究领域为软件可靠性,软件工程;**邓勇进**(1976－),男,助教,主要研究领域为软件工程.

通常,软件系统的可靠性评估方法主要有 3 类[2]:基于操作剖面的模型、基于状态的模型和基于路径的模型.基于操作剖面的模型基础是用户使用软件的操作及其频率信息.基于状态的模型通常假设软件的控制转移具有 Markov 性质,但基于构件的软件系统难以保证 Markov 模型需要的构件独立性假设.基于路径的模型通常在实现后对软件系统可靠性进行评估.

本文提出了基于构件软件的一种可靠性通用模型——构件概率迁移图.该模型跟踪动态开发过程中基于构件软件的可靠性,符合基于构件软件及软件开发的特征,具有较广的覆盖面和较强的兼容性.

1 基于构件软件的开发特征

构件是指封装了数据和功能、在运行时能够通过参数进行配置的模块.通常构件由第三方开发,具有清晰的接口描述.除了常见的功能接口描述以外,构件还应有表明该构件使用场合和可靠性等性能指标的描述.以此为依据,基于构件的软件设计师才能够判断该构件是否适合当前系统.但由于软件可靠性与使用该软件的环境相关,因此,技术指标中的可靠性表述不仅仅是某个具体的指标参数,而是一组指标参数,可用"映射"关系来表示:给定一个使用环境,则返回一个该环境下的可靠性情况.为了在具体的使用环境下对构件的可靠性作出判断,构件生产者需要提供什么样的数据呢?目前还没有这方面的标准[1].通常考虑两个因素:数据必须足够在具体环境下对软构件可靠性作出估计;开发者对这些数据的收集也要在可以接受的代价范围内.

我们认为,问题的关键是构件开发者与构件使用者的分离.两者的分离为基于构件软件的可靠性评测带来了新的挑战:构件开发者不能预知构件具体的使用环境;构件使用者无法明了构件的内部细节.为了解决该问题,很多研究人员进行了大量的工作[1,3~8],其中文献[1]以输入子域的角度研究了基于构件的软件可靠性理论基础.

追踪构件开发和使用的过程,我们可以从理论角度给出基于构件软件的可靠性工程过程:

- 构件开发者定义一种空间划分Π;
- 构件开发者定义一种度量(或性质)M;
- 构件开发者对每个划分块计算构件关于该划分块的度量,即,$\forall s\in\Pi$,计算关于s的度量$M(s)$;
- 构件开发者收集各划分块及其度量,形成一个离散映射函数$f=\{\langle s,M(s)\rangle|s\in\Pi\}$;
- 系统设计者确定使用构件的软件系统的架构,然后使用构件开发者提供的数据和软件系统的使用环境,计算软件系统可靠性;
- 如果某构件可靠性不能满足需求,则可以重新选择或开发更合适的构件,也可以调整软件系统架构,甚至两者兼用.

但由于构件概念的多样性以及构件关联的复杂性,从理论到实践需要进行大量的工作.本文尝试了一种紧密结合动态开发过程的方法.

2 基于构件软件的可靠性通用模型

2.1 基于构件软件的可靠性问题分析

下面是两种对基于构件软件进行可靠性估计的方法:

(a) 分析软件的使用模型或操作剖面,据此实施可靠性测试,最后应用可靠性模型于测试结果数据计算软件可靠性;

(b) 分析软件中构件及其使用频度,结合构件开发者提供的可靠性数据计算软件可靠性.

那么,如何判断哪种方法更适合基于构件软件的可靠性估计呢?

其根源在于构件开发者和构件使用者的分离.对方法进行评价的关键是分析方法能否充分利用已知信息提高可靠性估计的准确性,能否自然、方便地使用已知信息,对基于构件软件的开发过程覆盖面如何等.

从最终产品——基于构件软件来看,可见的是新开发的构件、以前开发现在复用的构件、第三方构件接口和连接这些构件的胶合逻辑(和代码),对于第三方构件除接口外的内部细节则不可见.

从产品开发过程来看,基于构件软件采用迭代增量式开发,早期可见的是产品的构架/体系结构、以前开发现在复用的构件、第三方构件接口和最终用户对产品的使用模式;随着项目的进展,会出现新开发完成的构件的逐步加入、产品构架的细微调整、复用构件的重选等已知信息体系的变化.

从已知信息的利用来看,方法(a)视整个软件为黑盒,仅利用了用户对软件的使用模式信息;方法(b)则进一步穿透软件,利用了软件中出现哪些构件以及这些构件的使用频度信息.从使用涉及信息的自然性和方便性方面来看,两种方法都不错;但两者均未充分利用已知信息.比如,方法(b)对如何获取各构件的使用频度缺乏足够的信息来源,也就不能保证其估计的准确性;方法(a)则是在开发阶段后期的可靠性估计方法,此时设计、开发工作已基本完成,根据"可靠性是设计到软件中的"这一观点,此时发现的可靠性问题带来的变更成本偏高.因此,这两种方法都不是评估基于构件软件可靠性的最佳选择.

2.2 基于构件软件中的函数

在讨论模型之前,我们作两个假设:

- 假设所有的构件都是第三方构件.对于自主开发的构件,其可靠性完全可以采用传统方法进行估计.因此,如同第三方构件,它们的细节处理被排除在外.
- 假设胶合逻辑完全可靠.较大规模的胶合逻辑可以抽象为独立的构件参加到软件产品中,剩下较小规模的胶合逻辑则假设足够简单.

软件可靠性度量的是用户对软件运行可靠程度的感知.可靠性的一种原始计算是所有运行路径出现频度与该路经可靠性乘积的累加($R=\sum_{P_i\in\varphi}(R_{P_i}\times F_{P_i})$,其中 R 为可靠性,φ 为所有运行路径的集合,R_{P_i} 和 F_{P_i} 分别为路径 P_i 的可靠性和出现频度).对于基于构件软件,由于采用内部细节不可见的第三方构件,所以可以把构件作为路径中的节点.该计算方式的模型是运行路径的树型结构.该模型有两个问题:一是经常会出现无穷长路径(也会有无穷多分支节点),尤其是对于反应式软件;二是模型中的路径是一种带历史的概念,复杂度较高,其频度信息难以准确获取.

路径无穷长问题可以采用自动机或迁移图的模型解决,这时,路径类似于可识别的字.但迁移图中的迁移仅与当前状态相关,不能感知历史,于是,带历史的路径的频度信息只好以统计结果的形态集中出现在没有历史感知能力的迁移上.同时,以统计形态出现的路径的频度也就转换成了构件之间迁移发生的(局部)频度.后者的获取比前者相对容易些.

以一个只有 3 个构件 C_1,C_2,C_3 的软件为例,其构件迁移图粗略地如图 1 所示.其中 $R_{C_j}(C_i)$ 表示当 C_i 迁移到 C_j 时,C_j(在该使用环境下)的可靠性;$P(C_i,C_j)$表示 C_i 迁移到 C_j 的概率.

上述模型的核心是两个函数:

- $f:C\to(C\to R)$,其中R为可靠性,它用来描述一个构件在其他构件上下文中所表现出来的可靠性,$f(C_i)$表示的是C_i的可靠性函数,在不同构件(如C_j和C_k)上下文中,它表现的可靠性也有所不同(如$f(C_i)(C_j)$和$f(C_i)(C_k)$).
- $g:C\times C\to P$,其中$P=\{p|0\le p\le 1\}$.$g(\langle C_i,C_j\rangle)$表示构件$C_i$迁移到$C_j$的概率.

但是构件开发者与构件使用者分离的特性,使得构件开发者不能具体给出构件 C_i 的 $f(C_i)$这一可靠性映射函数.在真实环境中,构件开发者给出构件如何被使用以及会表现出怎样的可靠性的信息比较合理.因此,上述 f 函数需要加以变化,具体被分解为两个函数(如图 2 所示):

- $f_1:C\times C\to M$,其中M为使用模型,$f_1(C_i,C_j)$表示从C_i迁移到C_j时C_j的使用模型,即此时C_j如何被使用.
- $f_2:C\to(M\to R)$,其中M为使用模型,$f_2(C_i)$为C_i的可靠性映射,只要知道C_i如何被使用(如$m\in M$),就可以知道这种使用方式下C_i体现的可靠性(如$f_2(C_i)(m)$).

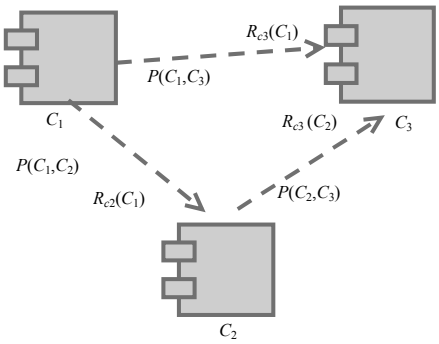

Fig.1 An example of component-based software
图 1 基于构件软件示例

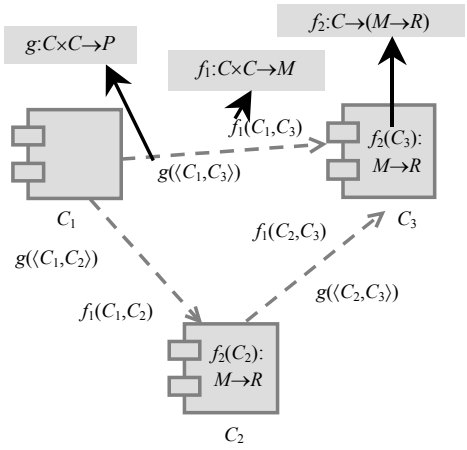

Fig.2 Functional abstraction
图 2 函数抽象

2.3 基于构件软件的可靠性通用模型——构件概率迁移图

定义 1(构件概率迁移图). 构件概率迁移图 G 是一个十元序偶$\langle C,T,h,S,E,M,P,R,g,f\rangle$,其中 C 为构件集合,T 为迁移集合,全函数 $h:T\to C\times C$,$S\subseteq C$ 为起始构件集合,$E\subseteq C$ 为终止构件集合,M 为使用模型集合,R 为可靠性度量论域,P 为概率论域,全函数 $g:T\to M\times P$,$f:C\to(M\to R)$.通常,P 为集合$\{x|0\le x\le 1\}$.

定义 2(运行). 在构件概率迁移图 $G=\langle C,T,h,S,E,M,P,R,g,f\rangle$中,若路径 $c_0t_1c_1t_2c_2\ldots t_nc_n$ 满足:

i) $\forall 0\le i\le n, c_i\in C$;

ii) $\forall 1\le i\le n, t_i\in T$;

iii) $c_0\in S, c_n\in E$;

iv) $\forall 1\le i\le n, h(t_i)=\langle c_{i-1},c_i\rangle$,

则称该路径为构件概率迁移图 G 的一个运行.

2.4 通用模型实例化及可靠性估计方法

构件概率迁移图通用模型中的 f 由构件开发者提供,使用模型的描述方法根据具体应用也可能有所不同.本节给出一个通用模型的具体实例.

由于构件的属性可以通过属性访问操作取值和置值(如 Void Set(ATTR x),ATTR Get()),因此我们不妨假设对构件的使用均通过具体的操作进行,即构件只提供操作接口而不提供属性接口.

定义 3(使用模型). 假设 OP_{C_i} 为构件 C_i 的接口操作集合,那么一个满足 $\sum_{op\in OP_{C_i}} m(op)=1$ 的全函数 $m:OP_{C_i}\to P$ 就是构件 C_i 的一个使用模型,其中 $P=\{p|0\le p\le 1\}$为出现概率的论域.

在构件概率迁移图中,我们将 C_i 到 C_j 迁移上出现的使用模型限定为 C_j 的使用模型,每个构件的出迁移概率之和为 1,而且构件开发者在构件接口中提供如下形式的可靠性映射:

定义 4(可靠性映射). 假设 OP_{C_i} 为构件 C_i 的接口操作集合,那么全函数 $r:OP_{C_i}\to R$ 就是构件 C_i 的一个可靠性映射,其中 $R=\{r|0\le r\le 1\}$.

根据上述可靠性映射可以得到函数 f.

注意,下面如图 3 所示的实例中其操作也可进一步细化,即引入操作的状态空间.

设起始构件为附加构件,它不做任何动作,可靠性为 100%,则在上述通用模型实例下估计软件系统可靠性的基本步骤如下:

i) 生成测试路径

根据构件概率迁移图即软件系统的使用模型随机(遵循迁移概率)生成测试路径.在构件概率迁移图中,测试路径就是运行,可以用手动或自动的方式生成.

ii) 计算测试路径的可靠性

假设测试路径 P 为 $c_0t_1c_1t_2c_2...t_nc_n$,那么该路径可靠性可以计算如下:

$$R_P = \prod_{1\le i\le n} f(C_i)((g(t_i))_1),$$

其中, $g(t_i)$ 是 $M\times P$ 上的二元序偶, $(g(t_i))_1$ 表示取 $g(t_i)$ 的第 1 个元素.

iii) 计算软件系统的可靠性

软件系统的可靠性可以计算如下:

$$R = \sum_{1\le i\le n}(R_{P_i}\times F_{P_i}) / \sum_{1\le i\le n} F_{P_i},$$

其中 R_{P_i} 和 F_{P_i} 分别为路径 P_i 的可靠性和出现频度.若测试路径 P 为 $c_0t_1c_1t_2c_2...t_mc_m$,则该路经的出现频度 $F_P = \prod_{1\le i\le m}(g(t_i))_2$,其中, $g(t_i)$ 是 $M\times P$ 上的二元序偶, $(g(t_i))_2$ 表示取 $g(t_i)$ 的第 2 个元素.

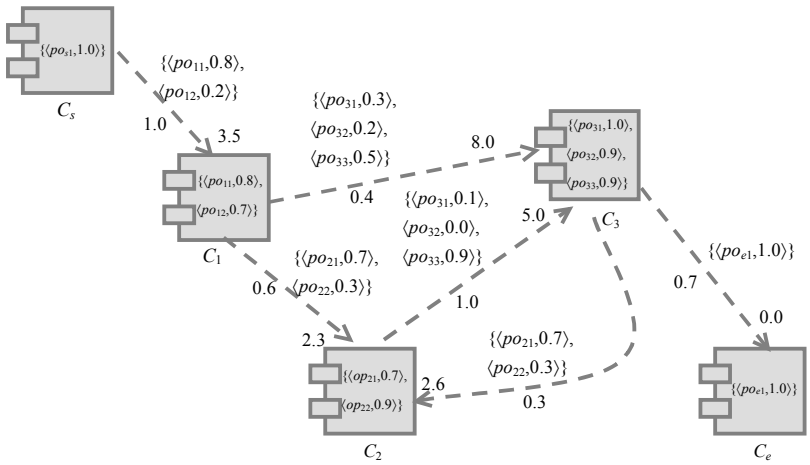

Fig.3 An example of component transition probability diagram

图 3 构件迁移概率图示例

2.5 模型特点

构件概率迁移图通用模型具有以下特点:

- 支持基于构件软件的可靠性指标分配.在可靠性需求阶段,利用构件概率迁移图中的构件关系,可以将软件系统的可靠性指标分解到具体的构件上.
- 支持基于构件软件的架构调整.软件架构信息可以依据使用关系反映到构件概率迁移图上,从而发现调整对软件可靠性的影响程度,根据影响可能需要重新分配可靠性指标.即使所有构件不变,软件架构的调整也会导致整体可靠性的提高或降低,原因在于架构调整改变了构件的使用关系,而可靠性与使用模型直接相关.
- 支持基于构件软件的增量、迭代式开发.构件只要选择好或开发完成,其可靠性需求指标值就可以替换为该构件的实际可靠性,然后根据软件整体可靠性的重新估计结果指导开发计划和需求指标的调整.
- 支持构件复用.由于构件采用函数型可靠性数据,保证了该构件在不同使用环境下复用时对其可靠性进行重计算的可能性,从而解决了构件复用时环境变更导致的可靠性变化问题.

2.6 原型系统CoBSRA

基于构件概率迁移图,我们已初步实现了基于构件软件的可靠性分析工具 CoBSRA,其工程流图和外观如图 4、图 5 所示.

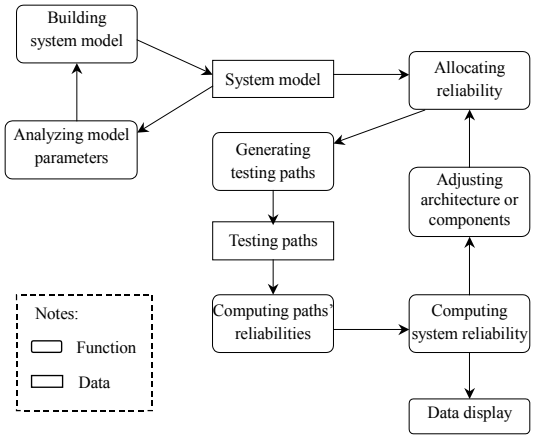

Fig.4 Engineering flow of CoBSRA
图 4 CoBSRA 工程流图

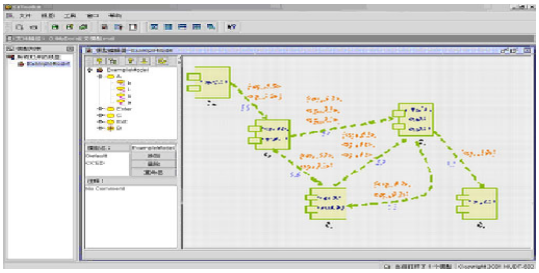

Fig.5 A snapshot of CoBSRA
图 5 CoBSRA 外观

3 结束语

20 世纪 90 年代以前,关于软件可靠性模型的研究多集中在黑盒方法[9]上,即利用使用模型或操作剖面对软件与外部环境之间的交互进行建模,软件内部的信息则不可见.这些模型以软件测试或使用期间获得的失效数据为研究对象,遵循统计学的观点对软件失效过程建模,从而估计或预测软件的失效行为.

随着 90 年代软件产业的快速发展,软件迅速脱离"一切从零开始"的开发模式,转向高级复用技术,如中间件、设计模式等.黑盒模型明显不能适应大型的基于软构件的新型软件开发模式,于是关于软件可靠性系统分析的方法得到重视,成为软件系统可靠性研究的主导方法.该方法根据软构件的第三方开发和基于构件的软件开发特点,虽然具体的软构件仍是黑盒,但要求系统内部的结构信息可见.围绕该方法提出了一系列基于结构的软件可靠性估计模型[3~5,8],其中大部分模型均可被视为构件概率迁移图这一通用模型或其扩展时间维后的实例,而基于通用模型建立的系统将可以兼容其所有实例集.

构件概率迁移图模型的出发点和落脚点是跟踪基于构件软件开发过程中的可靠性,而相关模型大多偏重于数学建模和变换,对动态的软件开发则考虑不多.因此,与相关模型的点上静态建模思想不同,构件概率迁移图模型采用的是线上动态建模的思想,更适合指导基于构件软件的开发工作.

References:

[1] Hamlet D, Mason D, Woit D. Theory of software reliability based on components. In: Proc. of the 3rd Int'l. Workshop on Component-Based Software Engineering. Toronto: IEEE Computer Society, 2001. 361~370.

[2] Goseva-Popstojanova K, Trivedi K, Mathur AP. How different architecture based software reliability models are related? In: Proc. of the Fast Abstracts 11th IEEE Int'l. Symp. on Software Reliability Engineering (ISSRE 2000). San Jose, California, 2000. http://www.chillarege.com/fastabstracts/issre2000/2000103.pdf

[3] Goseva-Popstojanova K, Trivedi K. Architecture-Based approach to reliability assessment of software systems. Performance Evaluation, 2001,45(2-3):179~204.

[4] Gokhale S, Lyu M, Trivedi K. Reliability simulation of component based software systems. In: Proc. of the 9th Intl. Symp. on Software Reliability Engineering (ISSRE'98). Paderborn: IEEE Computer Society, 1998. 192~201.

[5] Krishnamurthy S, Mathur AP. On the estimation of reliability of a software system using reliabilities of its components. In: Proc. of the 8th Int'l. Symp. on Software Reliability Engineering (ISSRE'97). Albuquerque, NM: IEEE Computer Society, 1997. 146~155.

[6] Mason D. Probabilistic analysis for component reliability composition. In: Crnkovic I, Schmidt H, Stafford J, Wallnau K, eds. Proc. of the 5th ICSE Workshop on Component-Based Software Engineering: Benchmarks for Predictable Assembly. Orlando, 2002.

[7] May J. Component-Based software reliability analysis. Technical Report, CSTR-02-002, Department of Computer Science, University of Bristol, 2002.

[8] Gokhale S, Wong W E, Trivedi K, Horgan JR. An analytical approach to architecture based software reliability prediction. In: Proc. of the 3rd Int'l. Computer Performance & Dependability Symp. (IPDS'98). Durham: IEEE Computer Society, 1998. 13~22.

[9] Musa JD, Iannino A, Okumoto K. Software Reliability: Measurement, Prediction, Application. New York: McGraw-Hill, 1987.