

一种基于风险分析的软件集成测试方法

胡文生^{1,2,3}, 赵 明³, 杨剑峰^{2,3}, 贾国荣¹

(1 贵州商业高等专科学校 计算机科学与技术系, 贵州 贵阳 550005;

2 贵州大学 计算机科学与信息学院, 贵州 贵阳 550025;

3 贵州省可靠性工程中心, 贵州 贵阳 550025)

摘 要: 基于统一建模语言(UML)用例图和序列图在描述系统的功能及完成相应功能方面的特点, 结合风险分析技术, 提出了以用例的使用概率为基础, 按照完成用例功能的场景即序列图中所涉及到的组件使用概率和风险程度, 从中选出使用概率最高的和风险程度最高的组件为核心, 依照三明治的软件集成方式逐个将其他的组件集成进来从而完成整个系统的集成测试. 这种测试方式对系统的风险进行控制, 实现了部分的可靠性测试.

关键词: UML; 用例; 使用概率; 风险

中图分类号: TP312

文献标识码: A

文章编号: 1000—7180(2012)10—0030—04

A Integration Test Method Based on Risk Analysis

HU Wen-sheng^{1,2,3}, ZHAO Ming³, YANG Jian-feng^{2,3}, JIA Guo-rong¹

(1 Commercial College of Guizhou, Guiyang 550005, China;

2 College of Computer Science & Information, Guizhou University, Guiyang 550025, China;

3 Reliability Engineering Center of Guizhou, Guiyang 550025, China)

Abstract: This paper proposes a method based on the features of UML (Unified Modeling Language) use case diagram and sequence diagram by risk analysis techniques and using probability of use cases. One component is selected as core in each use case according to this component using probability and risk degree. To implement integration testing of whole system, the component with the highest using probability and the highest risk is regard as the core, the other components are gradually involved step by step according to sandwich integration testing. This testing method can control the risk of system and implement a part of reliability testing.

Key words: UML; use case; using probability; risk

1 引言

软件测试是软件质量保证的重要手段. 现有的软件测试已经突破传统的基于代码运行的单一的软件测试手段, 呈现出测试方法的多样化, 除了有白盒测试、黑盒测试外, 还有灰盒测试, 测试的目的性更加明确, 为了适应软件的质量要求, 软件的可靠性、安全性测试变得越来越重要.

按照软件测试的过程分类, 可以分成单元测试、集成测试、系统测试、验收测试(也叫确认测试). 以

上四个过程相互独立且依照软件开发的各个过程依次进行. 其中最著名的 V 模型就是上述思想的最好体现. V 模型最早是由 Paul Rook 在 20 世纪 80 年代后期提出来的, 该模型反映出了软件测试活动与软件分析设计活动的关系. 如图 1 所示.

每个阶段的软件测试的依据是软件开发过程中所形成的各种文档、文件、模型, 其中统一建模语言(UML)所建立的各个模型图作为软件测试的依据是目前软件测试研究领域中的重点, 从各种文档、文件中提炼出软件测试用例的工作比较繁杂, 容易遗

漏一些重要信息,而 UML 模型图能直观、真实、全面地反映出整个软件静态和动态的体系结构.表 1 描述了各个 UML 模型与软件开发过程中的对应关系.

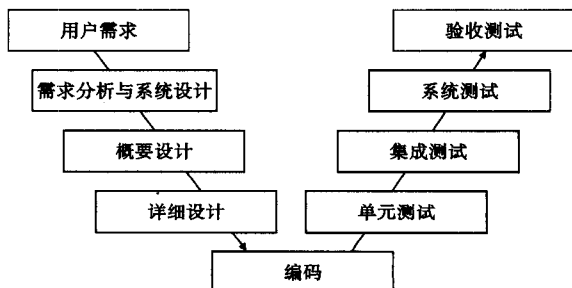


图1 软件测试的 V 模型

表1 UML 模型图与软件开发过程

软件开发过程	UML 模型图
用户需求	用例图
需求分析与系统设计	用例图、包图、部署图
概要设计	组件图、顺序图、交互图、状态图、活动图
详细设计	类图、对象图

结合图1与表1,将基于 UML 模型的软件测试的 V 模型称为 UML-V 模型,如图2所示.

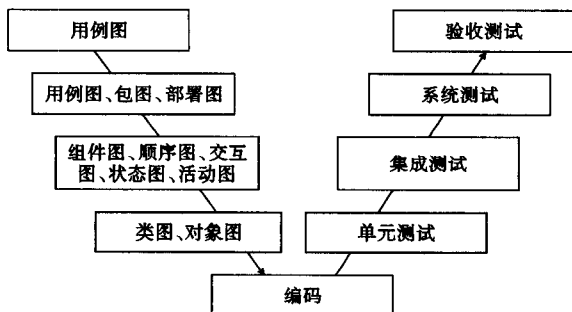


图2 软件测试的 UML_V 模型

文中主要研究基于 UML 模型的软件集成测试,与以往文献研究集成测试的方式不同,该文主要以用例的优先级别和组件的风险程度结合 UML 的用例图、顺序图、组件图来研究软件的集成测试,提出了一种新的、更能保证软件质量的软件集成测试^[1].

2 基于 UML 模型的软件集成测试的策略分析

软件集成测试的实施方案有很多,主要包括非增量式集成测试、增量式集成测试、三明治集成测

试、核心集成测试、分层集成测试、高频集成测试等.本文提出的集成测试,是综合上述几种集成测试方案,再结合组件的风险分析、用例的优先级别的一种综合的软件集成测试方案.该测试策略的最大优点是始终以优先级别最高的用例中所包含的风险最大的组件为核心,然后再逐步集成其他的组件.由于把组件的风险因素也考虑进去了,所以在软件集成测试的过程中,也在一定程度上对软件进行了可靠性测试.

2.1 用例的优先级别的确定

按照用例的使用情况把用例分成四个等级,分别是关键用例、重要用例、一般用例、轻微用例.具体描述如下:

① 关键用例:使用概率在 0.75~1.0 之间,该种类型的用例为用户经常使用的用例,在该系统中占有重要的地位.

② 重要用例:使用概率在 0.50~0.75 之间,该种类型的用例为用户比较常用的一些用例,在短期内可以没有.

③ 一般用例:使用概率在 0.25~0.50 之间,该种类型的用例为用户不常使用的用例,在较长时间内可以没有.

④ 轻微用例:使用概率在 0.00~0.25 之间,该种类型的用例为用户很少使用的用例,一般为一些特殊的用户而设置,在整个系统中只是起一些辅助作用,在测试过程中可以长时间内无须出现.

上述用例分类是按照该用例的使用概率来区分的,有两种特殊的情况必须剔除,一种是使用概率在 1.0 的用例,出现这种情况表示该系统只有一个用例,也就是说该系统只能提供单一的功能,目前,这样的系统还是比较少的,不是本文研究的范围.另一种是使用概率为 0.00 的,这表示该用例没有用户使用,根本没有存在的价值,本文主要研究使用概率大于 0 而小于 1 的用例.

用例的使用概率可以参考文献[2]所提出的方式进行计算.假设一个系统有 m 种用户类型,有 n 个用例,用 $p_i (i=1,2,\dots,m)$ 表示第 i 种用户类型的用户数在使用系统的用户总数中所占的比例, $p_{ij} (j=1,2,\dots,n)$ 表示第 i 种用户类型使用用例 j 的概率,于是用例 x 的使用概率为:

$$P(x) = \sum_{i=1}^m p_i \cdot p_{ix} \quad (1)$$

其中, $\sum_{i=1}^m p_i = 1, \sum_{j=1}^n p_{ij} = 1, P(x)$ 表示用例 x

的使用概率。

2.2 用例执行场景中组件及组件连接器的风险程度分析

用例图中每个用例的执行场景可以用顺序图或交互图来表示出来,顺序图或交互图通过信息在组件之间的交互、传递来描述每个用例所表示的功能是怎样实现的.在顺序图或交互图中每个组件会随着时间或事件的交互进行而发生状态的变化,由此可以推导出每个组件的状态图.从顺序图或交互图中也可以得到每个用例执行场景下的组件图.所以可以通过顺序图、交互图、状态图、组件图来计算每个组件以及组件之间连接在用例的具体执行场景下的风险程度.具体计算过程如下^[3]:

第一步:度量组件的动态复杂度(即组件失效概率).

$$dco_i^x = t_i^x - c_i^x + 2 \quad (2)$$

其中 dco_i^x 表示组件 i 在用例执行场景 S_x 下的动态复杂度; c_i^x 表示的是组件 i 在执行场景 S_x 下组件 i 的状态数; t_i^x 表示组件 i 在执行场景 S_x 下状态之间的转移数.

对公式(2)进行标准化如下:

$$DOC_i^x = \frac{dco_i^x}{\sum_{k \in S_x} dco_k^x} \quad (3)$$

其中 DOC_i^x 的取值在 0 到 1 之间,可以看成是组件 i 的失效概率,公式中的分母表示在场景 S_x 下所有活动的组件复杂度总和.

第二步:度量组件之间链接器的动态耦合度(即链接器失效概率).

$$EOC_{ij}^x = \frac{|MT_{ij}^x|}{|MT^x|} \quad (i, j \in S_x, i \neq j) \quad (4)$$

EOC_{ij}^x 表示两个组件 i 和 j 之间交互的动态耦合度, MT_{ij}^x 表示场景 S_x 执行期间从组件 i 发送到组件 j 的信息集的基数, MT^x 表示在场景 S_x 执行期间所有活动的组件相互交互的所有信息集的基数, EOC_{ij}^x 的值在 0 到 1 之间,它可以用来表示两个组件之间链接器失效的概率.

第三步:确定组件的风险因素.

组件 i 在场景 S_x 中的风险因素 rf_i^x 可以从下式计算:

$$rf_i^x = DOC_i^x * svt_i^x \quad (5)$$

其中 DOC_i^x 是第 i 个组件在场景 S_x 下的标准的复杂度,它的取值 $0 \leq DOC_i^x \leq 1$; svt_i^x 是第 i 个组件在场景 S_x 下的严重等级,它的取值 $0 \leq svt_i^x \leq 1$;

第四步:确定组件之间链接器的风险因素.

用 rf_{ij}^x 表示组件 i 和组件 j 在场景 S_x 下的风险因素分析,可以用下式计算:

$$rf_{ij}^x = EOC_{ij}^x * svt_{ij}^x \quad (6)$$

其中 EOC_{ij}^x 是第 i 个组件与第 j 个组件之间的链接器在场景 S_x 下的标准的耦合度,它的取值 $0 \leq EOC_{ij}^x \leq 1$; svt_{ij}^x 是第 i 个组件和第 j 个组件之间的链接器在场景 S_x 下的严重等级,它的取值 $0 \leq svt_{ij}^x \leq 1$.

2.3 确定每个组件的使用频率

通过用例的执行场景即用例对应的顺序图^[4-5].计算各个组件在该执行场景下的执行频率,用下面公式进行计算:

$$p(C_i) = \frac{C_i^x}{\sum_{j=1}^n C_j^x} \quad (7)$$

其中 C_i^x 表示组件 C_i 在用例执行场景 S_x 下的执行的次数, $\sum_{j=1}^n C_j^x$ 为用例执行场景 S_x 下的所有组件执行次数的总和.

根据组件的执行频率可以对某个用例执行场景下的组件进行分类:

- ① 使用频繁高的组件:执行频率在 0.60~1.0 之间.
- ② 使用频繁较高的组件:执行频率在 0.20~0.60 之间.
- ③ 使用频繁一般的组件:执行频率在 0.00~0.20 之间.

2.4 基于 UML 模型及风险分析的软件集成测试的算法描述

按照三个层次对组件进行划分,第一个层次从用例层次,第二个层次是从组件在用例执行场景下的执行情况,第三个层次是从组件的风险程度.三个层次的划分如图 3 所示.图 3 中具体展开情况如图 4 所示.

基于风险的软件集成测试的基本思想是:根据用例使用概率从高到低对用例进行排序,然后对用例中每个组件进行排序,按照组件的使用概率和用例的风险程度进行从高到低的进行排序,以使用概率最高的用例中所涉及到的组件开始集成,按照组件的使用概率和风险程度从高到低的集成,使用概率高的用例集成完成后开始进行下一个使用概率排在第二的用例中所涉及的组件开始集成,集成的原则也是按照组件的使用概率和风险程度从高到低的

集成,但是第二次集成时必须剔除前面集成过程中已经集成过的组件,只对前面没有集成过的组件进行集成.这样一直进行下去,直到所有的用例中所涉及到的所有组件全部集成为止.核心思想是以使用概率最高的用例中使用概率和风险程序最高的组件为核心,以该组件为中心再按照各外围软件组件的使用概率和风险程序逐个集成到核心系统中,必要时可以使用驱动模块和桩模块,采用三明治集成测试完成整个软件的集成测试.

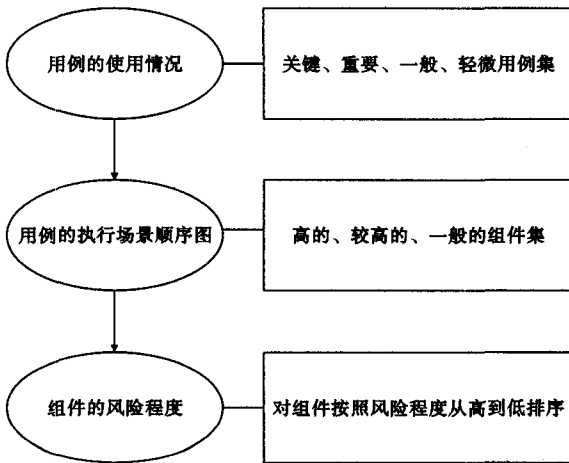


图3 三个层次的划分

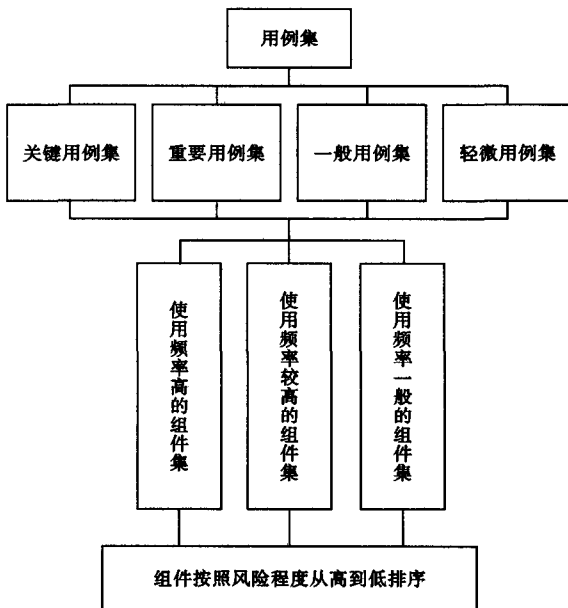


图4 集成测试展开示意图

3 结束语

文中利用 UML 用例图和序列图找出最核心的组件,以此为中心将软件系统的其他组件按照重要程度逐步进行集成测试.这种测试的优势是可以大大减少系统问题的发生,把容易发生问题的部分提前测试,使各种错误和缺陷能及时发现及时修改从而缩短了相应的测试时间,而且在集成测试过程中也部分完成了可靠性测试.

参考文献:

- [1] Wende Kong. Towards a formal and scalable approach for quantifying software reliability at early development stages[D]. College Park: Dissertation submitted to the Faculty of the Graduate School of the University of Maryland, 2009.
- [2] Singh H, Cortellessa V, Cukic B, et al. A bayesian approach to reliability prediction and assessment of component based systems[C]//Proc. of 12th International Symposium on Software Reliability Engineering (IS-SRE'01), 2011.
- [3] Ajith Reddy Guedem. Software architectural risk assessment[D]. Master Thesis submitted to the College of Engineering and Mineral Resources at West Virginia University, 2004.
- [4] Ahmed E Hassan. Architectural level risk assessment [D]. Dissertation submitted to the College of Engineering and Mineral Resources at West Virginia University, 2004.
- [5] Sherif M. Yacoub, Hany H Ammar. A methodology for architecture-level reliability risk analysis[J]. IEEE Transactions on software Engineering, 2002, 28(6): 529-547.

作者简介:

胡文生 男,(1969—),博士研究生,讲师.研究方向为软件可靠性.

赵明 男,(1962—),博士,教授,博士生导师.研究方向为软件、硬件可靠性、应用统计.

杨剑峰 男,(1986—),博士研究生.研究方向为软件可靠性.

贾国荣 男,(1958—),教授.研究方向为电子商务.