计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2020,29(5):270−274 [doi: 10.15888/j.cnki.csa.007386]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

面向回归测试的代码变更影响度量模型①

周海旭

(中国民航信息网络股份有限公司 北京市民航大数据工程技术研究中心, 北京 101318)
通讯作者: 周海旭, E-mail: hxzhou@travelsky.com

摘　要: 软件待测版本相对上一个版本的代码变更, 会对已有特性带来潜在的质量风险, 这一风险水平直接与回归测试用例的优先级相关联. 回归测试设计过程中的一个重要问题是如何衡量代码变更对回归测试用例优先级的影响. 本文在回归测试用例优先级评估模型的基础上, 从测试覆盖的角度建立起回归测试用例与代码变更的直接关联, 从代码整体耦合性的角度建立起回归测试用例与代码变更的间接关联, 分析了代码变更对回归测试的显性影响和隐性影响, 进而结合回归测试用例优先级的评估要求提出了一个新的度量模型. 实验结果显示, 使用该模型度量代码变更对回归测试用例优先级的影响水平, 可以得到比较全面和客观的定量结果, 从而为回归测试用例优先级的评估提供有效的支持.

关键词: 回归测试; 代码变更影响; 度量模型; 测试覆盖; 显性影响水平; 隐性影响水平

引用格式: 周海旭.面向回归测试的代码变更影响度量模型.计算机系统应用,2020,29(5):270–274. http://www.c-s-a.org.cn/1003-3254/7386.html

Code Change Impact Metric Model for Regression Test

ZHOU Hai-Xu

(Beijing Engineering Research Center of Civil Aviation Big Data, China Civil Aviation Information Network Inc., Beijing 101318, China)

Abstract: Code change which introduces risk in software quality is associated with regression test case prioritization. It is an important topic that evaluates the code change impact on regression test case prioritization, which plays a significant role in software quality assurance. This study analyzes the relationship between regression test case and code change from a test coverage and coupling perspective based on the test case prioritization evaluating model, and presents a new code change impact metric model which introduces both the dominant and recessive impact level. Experiments indicate that, the quantitative metric results of code change impact to regression test cases prioritization computed by this model are comprehensive and objective, and could provide effective support for regression test case prioritization evaluation.

Key words: regression test; code change impact; metric model; test coverage; dominant influence level; hidden influence level

回归测试指的是对软件待测版本中相对上一个版本不变的特性进行验证. 在软件的版本更迭过程中, 回归测试是必不可少的质量保障手段. 回归测试用例一般继承自产品用例库, 对于实际的软件项目来说, 回归测试用例的数量往往很大. 由于测试资源的限制, 一般不可能执行回归测试用例集中的全部用例, 而是要予以适当的取舍. 通常的做法是, 首先确定每个回归测试用例的优先级, 然后再根据可用资源的规模, 挑选一部分优先级较高的用例进行回归测试. 因此, 用例优先级的评估, 是回归测试设计中的一个核心问题[1−6].

① 基金项目: 国家核高基专项 (2014ZX010450101); 国家发展和改革委员会 2014 年云计算工程项目 (发改办高技[2014]1799 号)
Foundation item: National Science and Technology Major Program (2014ZX010450101); Year 2014, Cloud Computing Project of National Development and Reform Commission ([2014]1799)
收稿时间: 2019-09-27; 修改时间: 2019-10-22; 采用时间: 2019-11-05; csa 在线出版时间: 2020-05-07

软件待测版本相对上一个版本的代码变更, 会对已有特性带来潜在的质量风险, 这一风险水平直接与回归测试用例的优先级相关联. Orso 等[7]使用用户现场数据分析代码变更与回归测试用例选择的关系; Pfleeger 等[8]通过依赖追踪图建立起代码变更与回归测试用例的追踪框架; Ryder 等[9]提出了基于原子变更的分析方法, 可以找出所有可能受一组代码变更影响的回归测试用例, 同时也可以找出可能影响到某个回归测试用例的代码变更. Tyagi 等[10]考虑了缺陷严重性对回归测试用例优先级的影响; Mahmood 等[11]综合了需求、代码复杂性、历史信息等 16 个影响因素, 得到回归测试用例优先级的排序结果; Arar和 Ryu 等[12,13]使用成本敏感型机器学习算法来预测代码变更对软件质量及回归测试用例的影响; Di Nardo 等[14]对测试用例在软件历史版本上的执行信息进行分析, 并结合代码变更情况进行回归测试用例优先级排序. 然而这些已有的研究基本都没有涉及到代码变更对回归测试用例优先级影响水平的定量度量问题.

1 代码变更与回归测试用例优先级的关系

回归测试的目的是验证待测版本的变更是否会对已有特性产生不良影响. 从风险、价值和成本的角度, 可以归纳出回归测试用例优先级的启发式评估原则如下:

(1) 用例对应的需求特性受代码变更的影响越大, 则该用例的优先级应该越高;

(2) 用例对应的需求特性越重要, 则该用例的优先级应该越高;

(3) 执行用例所需的成本 (包含运行用例和维护用例的成本) 越低, 则该用例的优先级应该越高.

记回归测试用例总集为 S, 其中用例数量为 N, 假设 S 由一组用例子集构成, 记为$\{S_i\}$, $i=1,2,\cdots,n$. S_i中用例数量为N_i. 记产品需求特性总集为 U, 假设 U 由一组需求特性子集构成, 记为$\left\{U_{\hat{i}}\right\}$, $\hat{i}=1,2,\cdots,m$. 且S_i与$U_{\hat{i}}$存在测试覆盖的映射关系, 记为$S_i \to U_{\hat{i}}$. 用a_i表示待测版本的代码变更对需求特性子集$U_{\hat{i}}$的影响水平. 用v_i表示$U_{\hat{i}}$的价值水平, 即$U_{\hat{i}}$在功能或非功能上的重要程度. 用e_i表示运行S_i(手工或自动化方式) 所需人天. 用m_i表示维护S_i(维护手工用例或自动化脚本) 所需人天. 用 R 表示可投入到此次回归测试中的人天资源总数.

不失一般性, 假设S_i中用例数量$N_i=1$, 则S_i代表回归测试用例总集中的第 i 个用例. 用c_i表示该用例的优先级, 则[3]:

$$c_i=\frac{N\cdot R\cdot v_i\cdot a_i}{(e_i+m_i)\cdot\sum_{i=1}^{N}(e_i+m_i)\cdot\sum_{i=1}^{N}\frac{v_i\cdot a_i}{e_i+m_i}} \tag{1}$$

上式说明, 代码变更与回归测试用例优先级的关系, 集中体现在代码变更对相关需求特性的影响程度上. 数值上要求$a_i>0$.

2 面向回归测试的代码变更影响度量模型

根据式 (1), 在回归测试设计中进行用例优先级的评估时, 需要首先计算a_i的值, 也就是要定量评估代码变更对需求特性的影响程度. 目前度量a_i的主要手段是依靠研发人员的主观判断和经验, 结果并不精确. 本文将从测试覆盖的角度建立起需求特性与代码变更的关联, 综合显性和隐性影响水平两方面的度量指标, 提出一个新的代码变更影响度量模型, 为回归测试用例优先级的评估提供量化支持.

2.1 基于测试覆盖的度量模型设计

需求特性与测试用例之间存在测试覆盖的映射关系. 需求特性是测试用例的覆盖目标, 测试用例是需求特性的实现反映. 假设回归测试用例S_i覆盖需求特性子集$U_{\hat{i}}$, 那么$U_{\hat{i}}$的实现程度只能通过S_i的执行结果来体现, 如果代码变更影响了$U_{\hat{i}}$的实现, 必然影响S_i的执行结果. 也就是说, 代码变更对测试用例的影响程度, 可以反映出代码变更对需求特性的影响程度. 图 1 描述了代码变更与需求特性、测试用例的关系.

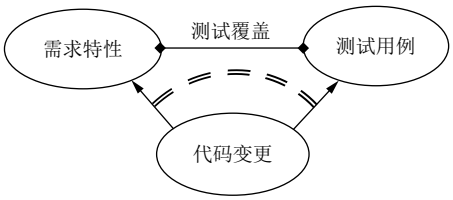

图 1 代码变更与需求特性、测试用例的关系

基于需求特性与测试用例之间的测试覆盖关系, 可以通过度量代码变更对测试用例的影响水平, 来评估代码变更对需求特性的影响水平.

进一步考虑如何度量代码变更对测试用例的影响. 回归测试用例S_i在执行过程中会覆盖被测软件中的一部分代码, 将这部分代码行的集合记为L_i. S_i的执行结果是否正确, 在代码层面完全取决于L_i的实现是否正

确. 因此, 代码变更对S_i的影响程度, 等同于代码变更对L_i的影响程度. 从局部的角度看, 发生变更的代码行集合和L_i都是被测代码的子集, 这两个子集的关系代表了代码变更对L_i的显性影响; 从整体的角度看, 代码总集内部的全局耦合性代表了代码变更对L_i的隐性影响. 本文提出的度量模型综合了局部和整体两方面的考虑.

2.2 显性影响水平

工程实践中, 绝大多数软件项目都使用配置管理工具来进行代码变更的标识、组织和控制. 流行的配置管理工具, 如 Git、SVN 等, 都可以很方便地输出待测版本的代码变更信息, 包括发生变更的类、代码行、变更类型、变更内容等.

测试用例与代码的关联关系可以通过代码覆盖来进行分析和描述, 基本步骤是:

(1) 对被测代码进行插桩, 即插入一些用于信息采集的探针代码, 并保证语义及功能与原始代码完全等效;

(2) 对插桩后的代码进行编译、链接, 获得可执行程序;

(3) 执行测试用例, 对插桩输出的信息进行处理, 获得用例对代码的覆盖结果, 也就是用例与代码的关联关系.

插桩有手工和自动两种方式. 手工方式指的是在代码需要获取运行覆盖信息的地方写入信息采集代码段, 相对灵活, 但对源代码有修改, 引入了额外的风险; 自动方式则通常是基于代码覆盖分析工具, 对编译链接产生的中间文件进行插桩修改, 相对安全且高效. 目前, 采用自动方式进行代码覆盖的分析已经是业界的主流选择, 流行的代码覆盖分析工具包括 Jacoco、EclEmma、gcov 等.

考虑待测版本代码变更对回归测试用例S_i的影响. 将待测版本的上一个版本的代码行总集记为 L. 借助代码覆盖分析工具, 依据上述分析步骤, 容易得到S_i覆盖的上一个版本的代码行集合, 即L_i, L_i中的元素可以用“命名空间 + 文件名 + 代码行号”的方式来描述; 借助配置管理工具, 容易得到待测版本相对上一个版本发生变更的代码行集合, 记为L_D, L_D采用与L_i同样的元素描述方式. 本文仅讨论待测版本发生代码变更的情形, 即$L_D \neq \emptyset$. 从代码行的角度看, 代码变更主要有三种情形: 修改某一行, 删除某一行, 或在某一行之后新增若干行. 若上一个版本的第j行被修改、删除, 或者在第j行之后新增了若干行, 则$j \in L_D$.

如果L_i与L_D存在交集, 说明测试用例S_i覆盖的代码在待测版本中发生了变更, S_i的测试执行结果将受到这一变更的直接影响. L_i与L_D交叠的程度越大, 代码变更对S_i的影响也越大. 基于这一结论, 我们使用L_i与L_D的 jaccard 相似度来度量代码变更对测试用例S_i的显性影响水平, 即:

$$J(L_i, L_D) = \frac{|L_i \cap L_D|}{|L_i \cup L_D|} \tag{2}$$

根据式 (2) 可知, $0 \le J(L_i, L_D) \le 1$. 当$J(L_i, L_D) = 0$时, 与S_i有关的代码没有发生变更; $J(L_i, L_D)$的数值越大, 说明与S_i有关的代码发生变更的比例越大, S_i检出缺陷的概率也越大.

2.3 隐性影响水平

通过评估测试用例覆盖代码行集合与变更代码行集合的交叠程度, 可以衡量代码变更对测试用例的显性影响程度. 然而, 由于代码耦合性的广泛存在, 即使测试用例覆盖的代码行均未发生变更, 该用例也可能受到变更的隐性影响.

假设待测版本代码中, 类的总集为$C = \{c_1, c_2, \cdots, c_{|C|}\}$. 借助代码覆盖分析工具, 容易得到测试用例$S_i$覆盖的类集合, 记为$C_{S_i} = \{c^{S_i}{}_1, c^{S_i}{}_2, \cdots, c^{S_i}{}_{|C_i|}\}$. 借鉴软件设计度量中耦合因子的概念[15], 定义面向测试用例的耦合因子如下:

$$COF(S_i) = \frac{\sum_{m=1}^{|C_{S_i}|}\sum_{n=1}^{|C|} isclient(c^{S_i}{}_m, c_n)}{|C_{S_i}| \cdot (|C| - 1) - \left(2\sum_{m=1}^{|C_{S_i}|}\left|Descendents\left(c^{S_i}{}_m\right)\right|\right)} \tag{3}$$

其中, $Descendents\left(c^{S_i}{}_m\right)$表示与$c^{S_i}{}_m$有派生关系的类集合; $isclient(c^{S_i}{}_m, c_n)$是一个函数, 当$c^{S_i}{}_m$引用了$c_n$中的某个方法或变量, 且$c^{S_i}{}_m$与$c_n$没有派生关系, 且$c^{S_i}{}_m \neq c_n$时, $isclient(c^{S_i}{}_m, c_n)$取值为 1, 否则取值为 0. 易知$0 \le COF(S_i) \le 1$.

$COF(S_i)$刻画了代码全局耦合性中与测试用例S_i有关的部分, 使用这个指标可以在一定程度上表征代码变更对测试用例的间接影响. 但是该指标与代码变更程度的关联较小, 只要代码没有类级别的变更, 比如类的新增、删除、派生关系、类间引用的改变等, $COF(S_i)$就是一个定值. 而在直观认知中, 代码修改得越多, 对测试用例执行结果和相关需求特性产生间接影响的可能性就越大. 因此, 我们需要针对代码变更程

度补充另一个度量维度.

考虑待测版本的上一个版本的代码行总集 L 和变更代码行集合 L_D 的关系. 从 L 和 L_D 的定义易知, $L \supseteq L_D$. 使用 L 和 L_D 的 jaccard 相似度来评估代码变更程度, 即:

$$J(L,L_D)=\frac{|L\cap L_D|}{|L\cup L_D|}=\frac{|L_D|}{|L|} \tag{4}$$

$$a_i=\lambda\cdot J(L_i,L_D)+COF(S_i)+J(L,L_D)=\lambda\cdot\frac{|L_i\cap L_D|}{|L_i\cup L_D|}+\frac{\sum_{m=1}^{|C_{S_i}|}\sum_{n=1}^{|C|}isclient(c^{S_i}{}_m,c_n)}{|C_{S_i}|\cdot(|C|-1)-\left(2\sum_{m=1}^{|C_{S_i}|}\left|Descendents\left(c^{S_i}{}_m\right)\right|\right)}+\frac{|L_D|}{|L|} \tag{5}$$

其中, λ 是一个可定制的常量, 用于调节显性影响水平和隐性影响水平的数值比例, 在同一软件的不同版本中可以取不同的值. 配置 λ 的经验原则是, 调节后的平均显性影响水平应比平均隐性影响水平高一个数量级, 以保证显性影响水平在度量中处于主导地位.

因为 $0\le J(L_i,L_D)\le 1$, $0\le COF(S_i)\le 1$, $J(L,L_D)>0$, 所以 $a_i>0$, 满足式 (1) 的数值要求.

该度量模型的构成如图 2 所示.

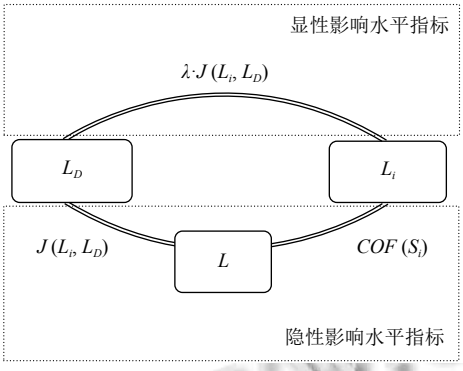

图 2 度量模型构成

模型由显性影响水平指标和隐性影响水平指标两部分组成. 显性影响水平刻画了用例覆盖代码发生变更的程度, 是度量模型的主要指标, 通过变更代码集合与用例覆盖代码集合的 jaccard 相似度进行度量; 隐性影响水平刻画了用例覆盖代码藉由代码总集内部耦合性与变更代码产生间接关联的程度, 是模型的次要指标, 通过面向测试用例的耦合因子、变更代码集合与代码总集的 jaccard 相似度进行度量.

由于模型全面考虑了代码变更对测试用例可能产生的显性影响与隐性影响, 在以下各类情形中, 模型都可以给出合理的度量结果:

(1) 回归测试用例覆盖的代码发生了变更. 这时, 模型中显性影响水平指标大于 0, 而且居于主导地位;

(2) 回归测试用例覆盖的代码未发生变更. 这时, 模式中显性影响水平指标等于 0, 度量结果取决于隐性影响水平指标, 即用例覆盖的代码与变更代码的耦合程度.

在后续的实验分析中, 可以看到度量模型上述特性的实际效果.

因为 $L_D\ne\emptyset$, 所以 $J(L,L_D)>0$.

结合面向测试用例的耦合因子、代码变更程度两方面的度量, 可实现对隐性影响水平相对完整的评估.

2.4 度量模型的构成与特性

综合以上对显性影响水平和隐性影响水平两方面的分析, 可以得到面向回归测试的代码变更影响度量模型如下:

3 实验分析

实验项目的一个迭代版本中共包括 11 个类, 1454 行代码. 回归测试用例集中共有 16 个用例, 各自对应一个需求特性. 该版本有两个需求特性发生了变更, 对应的回归测试用例分别是 TC05 和 TC06. 如果根据主观经验来分析代码变更对测试用例优先级的影响程度, 只能简单地给这两个用例赋予一个较高的影响水平, 而给其它用例赋予一个较低的影响水平.

基于本文提出的代码变更影响度量模型, 可得到度量结果如表 1 (λ 取值为 10). 影响水平排在前两位的是 TC06 和 TC05, 说明模型的度量结论与主观经验判断相符; 另外, 影响水平排在 3~5 位的依次为 TC01、TC14 和 TC16, 且 TC01 与 TC05 的影响水平很接近. 通过检查各测试用例的具体内容可以发现, TC01 和 TC14 虽然并不是针对发生变更的两个特性所设计的用例, 但是在用例的结果校验部分涉及了变更特性的功能, 所以受到了代码变更的直接影响程度较高; 通过检查各测试用例的代码覆盖情况可以发现, TC16 所覆盖的代码存在较多的类间变量引用, 在类层面的耦合性较高, 因此受到代码变更间接影响的可能较大. 由此

可见, 本文提出的模型可以对代码变更影响回归测试用例的程度给出更客观、更全面的定量度量结果.

表 1 代码变更对回归测试用例的影响水平度量结果

回归测试用例编号	$J(L_i,L_D)$	$COF(S_i)$	$J(L,L_D)$	a_i
TC01	0.11	0.41	0.01	1.52
TC02	0	0.11	0.01	0.12
TC03	0	0	0.01	0.01
TC04	0	0	0.01	0.01
TC05	0.15	0.03	0.01	1.54
TC06	0.25	0.16	0.01	2.67
TC07	0	0	0.01	0.01
TC08	0	0	0.01	0.01
TC09	0	0	0.01	0.01
TC10	0	0.07	0.01	0.08
TC11	0	0	0.01	0.01
TC12	0	0	0.01	0.01
TC13	0	0.21	0.01	0.22
TC14	0.04	0	0.01	0.41
TC15	0	0	0.01	0.01
TC16	0	0.32	0.01	0.33

4 结论与展望

本文首先从测试覆盖的角度建立起回归测试用例与代码变更的关联, 基于代码覆盖集合与代码变更集合的 jaccard 相似度来度量显性影响水平; 继而从代码整体系统性的角度, 使用面向测试用例的耦合因子、代码变更集合与代码总集的 jaccard 相似度来综合度量隐性影响水平. 本文提出的面向回归测试的代码变更影响度量模型, 同时考虑了显性影响水平和隐性影响水平两方面的因素, 能够对代码变更影响水平进行比较全面和客观的分析, 进而为回归测试用例优先级的评估提供有效的支持. 目前存在的问题是, 模型的显性影响水平指标中, 关于新增代码行这一情况的度量方式不够精细, 没有体现出代码增量大小带来的不同影响. 后续研究中将考虑予以改进.

参考文献

1 Wong WE, Horgan JR, London S, *et al*. A study of effective regression testing in practice. Proceedings the Eighth International Symposium on Software Reliability Engineering. Albuquerque, NM, USA. 1997. 264–274.

2 Epitropakis MG, Yoo S, Harman M, *et al*. Empirical evaluation of pareto efficient multi-objective regression test case prioritisation. Proceedings of the 2015 International Symposium on Software Testing and Analysis. Baltimore, MD, USA. 2015. 234–245.

3 周海旭. 回归测试用例优先级评估模型研究. 电子技术与软件工程, 2019, (1): 49–50.

4 Acharya AA, Mahali P, Mohapatra DP. Model based test case prioritization using association rule mining. Computational Intelligence in Data Mining, 2015, 3: 429–440.

5 Panichella A, Oliveto R, Penta MD, *et al*. Improving multi-objective test case selection by injecting diversity in genetic algorithms. IEEE Transactions on Software Engineering, 2015, 41(4): 358–383. [doi: 10.1109/TSE.2014.2364175]

6 Ranga KK. Analysis and design of test case prioritization technique for regression testing. International Journal for Innovative Research in Science and Technology, 2015, 2(1): 248–252.

7 Orso A, Apiwattanapong T, Law J, *et al*. Leveraging field data for impact analysis and regression testing. Proceedings of the 9th European Software Engineering Conference Held Jointly with 11th ACM SIGSOFT International Symposium on Foundations of Software Engineering. Helsinki, Finland. 2003. 128–137.

8 Pfleeger SL, Bohner SA. A framework for software maintenance metrics. Proceedings. Conference on Software Maintenance 1990. San Diego, CA, USA. 1990. 320–327.

9 Ryder BG, Tip F. Change impact analysis for object-oriented programs. Proceedings of the 2001 ACM SIGPLAN-SIGSOFT Workshop on Program Analysis for Software Tools and Engineering. Charleston, SC, USA 2001. 46–53.

10 Tyagi M, Malhotra S. An approach for test case prioritization based on three factors. International Journal of Information Technology and Computer Science (IJITCS), 2015, 7(4): 79–86.

11 Mahmood H, Hosain S. Improving test case prioritization based on practical priority factors. Proceedings of 2017 8th IEEE International Conference on Software Engineering and Service Science. Beijing, China. 2017. 899–902.

12 Arar ÖF, Ayan K. Software defect prediction using cost-sensitive neural network. Applied Soft Computing, 2015, 33: 263–277. [doi: 10.1016/j.asoc.2015.04.045]

13 Ryu D, Jang JI, Baik J. A transfer cost-sensitive boosting approach for cross-project defect prediction. Software Quality Journal, 2015, 25(1): 235–272.

14 Di Nardo D, Alshahwan N, Briand L, *et al*. Coverage-based regression test case selection, minimization and prioritization: A case study on an industrial system. Software: Testing, Verification and Reliability, 2015, 25(4): 371–396. [doi: 10.1002/stvr.1572]

15 Abreu FBE, Goulão M, Esteves R. Toward the design quality evaluation of object-oriented software systems. Proceedings of the 5th International Conference on Software Quality. Austin, TX, USA. 1995.