Estimation of Program Reverse Semantic Traceability Influence at Program Reliability with Assistance of Object-Oriented Metrics

Rogalchuk
Viacheslav Vladimirovich
Petersburg Passageway State University
email: vyacheslav.rogalchuk@gmail.com

Tyrva
Aleksey Vladimirovich
Petersburg Passageway State University
email: altyr@mail.ru

Khomonenko
Anatoly Dmitrievich
Petersburg Passageway State University
email: khomon@mail.ru

Abstract

In article the approach to the estimation of program reverse semantic traceability (RST) influence on program reliability with assistance of object-oriented metrics is proposed. In order to estimate reasonability of RST use it is naturally to define, how it influences on major adjectives of software development project: project cost, quality of the developed application, etc.

At present object-oriented metrics of Chidamber and Kemerer are widely used for predictive estimation of software reliability at early stage of life cycle. In row of works, for example, [6-7], it is proposed to use logistic regression for estimation of probability π (that a module will have a fault). The parameters of this model find by maximal likelihood method with calculation of object-oriented metrics.

In a paper it is shown how to change the software reliability model parameters, that was received using logistic regression, in order to estimate influence of program RST on program reliability.

***Keywords**: Object-Oriented Complexity Metrics, Program Reliability, Reverse Semantic Traceability*

Оценивание влияния обратной семантической трассировки программ на их надежность с помощью объектно-ориентированных метрик

Рогальчук
Вячеслав Владимирович
Петербургский государственный университет путей сообщения
email: vyacheslav.rogalchuk@gmail.com

Тырва
Алексей Владимирович
Петербургский государственный университет путей сообщения
email: altyr@mail.ru

Хомоненко
Анатолий Дмитриевич
Петербургский государственный университет путей сообщения
email: khomon@mail.ru

Аннотация

В статье рассматривается подход к учету влияния метода обратной семантической трассировки программ на прогнозные значения характеристик надежности, получаемые с помощью объектно-ориентированных метрик сложности. Чтобы оценить целесообразность применения метода обратной семантической трассировки, естественно определить, как он влияет на важнейшие характеристики проекта разработки программного обеспечения: стоимость проекта, качество разрабатываемого приложения и др.

В настоящее время для получения прогнозных оценок надежности программных продуктов на ранних этапах жизненного цикла широко используются объектно-ориентированные метрики ПО Чайдембера и Кемерера. В ряде работ, например, [6-7], вероятность отказа модуля π предлагается оценивать с использованием логистической регрессии. Параметры этой модели находятся методом максимального правдоподобия с учетом объектно-ориентированных метрик.

В статье показывается, как изменить параметры модели оценки надежности программ, полученной с помощью логистической регрессии, чтобы учесть влияние обратной семантической трассировки на надежность программы.

Ключевые слова: объектно-ориентированные метрики сложности, надежность программ, обратная семантическая трассировка программ

1. Введение

При разработке программного обеспечения может использоваться метод обратной семантической трассировки с целью сокращения расходов на сопровождение и рефакторинг (улучшение характеристик) уже созданных программ [1-2]. В частности, этот метод целесообразно использовать в случае, когда в процессе жизненного цикла создания программного проекта имеет место изменение состава сотрудников в команде разработчиков.

Чтобы оценить целесообразность применения метода обратной семантической трассировки, естественно определить, как он влияет на важнейшие характеристики проекта разработки программного обеспечения: стоимость проекта, качество разрабатываемого приложения и др. В частности, в [3] рассматривается влияние метода обратной семантической трассировки на стоимость разработки программного обеспечения, оцениваемой с помощью модели COCOMO II [4].

Мы сфокусируем свое внимание на оценивании влияния метода обратной семантической трассировки на одну из важнейших характеристик — надежность программного продукта.

По сути метод обратной семантической трассировки хорошо подходит к разработке программ с помощью объектно-ориентированного подхода. При этом с целью обеспечения возможности трассируемости программы при разработке выполняется описание отношений между ее объектами. В связи с этим для оценки влияния обратной семантической трассировки на характеристики надежности ПО воспользуемся объектно-ориентированными метриками.

Применение средств и методов обеспечения качества ПО наиболее эффективно на ранних этапах жизненного цикла. После выпуска продукта стоимость обнаружения и исправления ошибок увеличивается многократно. Соответственно, обнаружение потенциально ненадежных модулей на этапе проектирования позволяет выполнить перепроектирование или выделить больше ресурсов для их разработки и тестирования.

2. Объектно-ориентированные метрики

В настоящее время для получения прогнозных оценок надежности программных продуктов на ранних этапах жизненного цикла широко используются объектно-ориентированные метрики ПО Чайдембера и Кемерера [5-8]:

- Response for a Class (RFC) – отклик на класс. RFC определяется как количество методов, которые могут вызываться экземплярами класса, путем суммирования количества локальных и удаленных методов.
- Depth of Inheritance Tree (DIT) – длина наибольшего пути дерева наследования.
- Number of Children (NOC) – количество классов-наследников для заданного класса.

- Weighted Methods per Class (WMC) – взвешенное количество методов класса. Определяет меру сложности класса на основе цикломатической сложности каждого его метода. Класс с более сложными методами и большим количеством методов является более сложным.
- Lack of Cohesion in Methods (LCOM) – отсутствие сцепления в методах. Определяется как разность между количеством пар методов класса, использующих общие объекты, и количеством пар методов, не использующих общие объекты. Если разность отрицательна, то метрика считается равной нулю.
- Total Number of Attributes (NAI) – количество атрибутов классов.
- Coupling Between Object Classes (CBO) – связность между объектами класса (класс связан с другим классом, если методы одного класса используют методы или атрибуты другого класса и наоборот). CBO определяется как количество других классов, с которыми связан некоторый класс.

Ряд объектно-ориентированных метрик взаимосвязанности (OCAIC, DCAEC, OCAEC, ACMIC, OCMIC, DCMEC, OCMEC) более точно определяет отдельные свойства связанности классов, чем метрика CBO [8-9]. При определении наименований метрик используются следующие обозначения:

- Первая буква указывает отношение: A – взаимосвязь данного класса с классами-предками. D – взаимосвязь данного класса с классами-потомками. O – другие виды взаимосвязи.
- Следующие две буквы обозначают тип взаимодействия между классами C и D: CA (Class-Attribute) – существует взаимодействие класс-атрибут между классами C и D, если C имеет атрибут класса D. CM (Class-Method) – существует взаимодействие класс-метод между классами C и D, если C имеет метод с параметром типа класс D.
- Последние две буквы указывают направление взаимодействия между классами: IC (Import Coupling) – взаимосвязь импорта учитывает количество классов, от которых зависит данный класс. EC (Export Coupling) – взаимосвязь экспорта учитывает количество классов, которые зависят от данного класса.

В частности, метрика OCMEC (Other Class-Method Export Coupling) определяет количество классов, на которые воздействует (экспортирует влияние) данный класс через связь класс-метод.

Значения метрик собираются из документов проектирования – диаграмм унифицированного языка моделирования UML (диаграмм классов, последовательностей и др.), причем этот процесс может быть автоматизирован. На основе полученных метрик решается задача классификации модулей с целью выявления потенциально вызывающих ошибку. Построение такой модели осуществляется с использованием средств математической статистики или искусственного интеллекта (нейронные сети, генетические алгоритмы).

3. Прогнозирование надежности программ

В ряде работ, например, [6-7], вероятность отказа модуля π предлагается оценивать с использованием логистической регрессии и выражать в виде:

$$\pi = \frac{1}{1+e^{-(\beta_0+\sum \beta_i x_i)}}.$$

Здесь x_i – значение i-й метрики сложности, β_i – коэффициент этой метрики. Параметры β_i модели находятся методом максимального правдоподобия [10].

В [7] приводится пример и результат такого моделирования, для чего авторами проанализировано готовое коммерческое приложение, программные модули классифицированы как ненадежные (при

тестировании обнаружены ошибки) и надежные, определены проектные метрики сложности и построена регрессионная модель:

$$\pi = \frac{1}{1 + e^{-(-3.97 + 0.0464 \cdot NAI + 1.47 \cdot OCMEC + 1.06 \cdot DIT)}}. \quad (1)$$

Например, вероятность обнаружения отказа в производном классе (DIT=1) с тремя атрибутами (NAI=3) и при условии, что в модуле есть два других класса с параметрами методов типа данного класса (OCMEC=2):

$$\pi = \frac{1}{1 + e^{-(-3.97 + 0.0464 \cdot 3 + 1.47 \cdot 2 + 1.06 \cdot 1)}} = 0.5422. \quad (2)$$

Эту модель можно использовать на этапе проектирования для классификации модулей на надежные (например, $\pi < 0.33$) и ненадежные (соответственно, $\pi \geq 0.33$). Отметим, что сложностью применения данной модели является то, что параметры определяются с использованием статистики тестирования уже готового приложения, а расчеты по модели производятся при проектировании других приложений, таким образом, вносится некоторая погрешность. Указанное замечание не ограничивает применение описанной модели, так как результаты расчетов можно уточнить на более поздних этапах жизненного цикла (например, при анализе сложности исходного кода или статистики отказов при тестировании).

4. Оценка влияния обратной трассировки на надежность ПО

Вполне очевидно, что использование обратного трассирования оказывает двоякое влияние на объектно-ориентированные метрики сложности разрабатываемого программного обеспечения:

1. С одной стороны, увеличивается сложность разрабатываемого программного кода ввиду внесения дополнительной информации. Тем самым должно происходить снижение надежности программного обеспечения.

2. С другой стороны, улучшается сопровождаемость программного обеспечения, что должно приводить к улучшению надежности создаваемого ПО.

Первый фактор влияния обратной трассировки на надежность программного обеспечения в формуле (1) учитывается автоматически ввиду того, что используемые при описании отношений между объектами связи учитываются при расчете объектно-ориентированных метрик сложности.

Второй фактор влияния обратной трассировки на надежность программного обеспечения требует внесения поправки в значение коэффициента β_0 в формуле (1).

Чтобы учесть влияние метода обратной трассировки предлагается следующий подход. На основании данных по статистике ошибок, выявляемых при разработке программного обеспечения, производится оценивание доли ошибок разного класса. Для этого можно воспользоваться, например данными по статистике ошибок, представленными в [11-12] и др. Далее с помощью экспертных оценок определяется процент ошибок, нейтрализуемых с помощью метода обратного трассирования программ. На этой основе делается заключение о том, насколько снижается количество ошибок в программе благодаря проведению обратного трассирования программы.

Для примера приведем вариант классификации и распределения ошибок, описанный в [11]. В табл. 1 приведено распределение числа ошибок, выявленных при проведении системного тестирования (End to End), а также экспертные оценки возможного влияния обратного трассирования программы на устранение ошибок.

Таблица 1. Распределение ошибок, выявленных при системном тестировании

Тип ошибки	Доля P_E ошибок каждого типа, %	Оценка влияния I_T трассировки на устранение ошибок, %	Доля ΔP_E выявленных ошибок, %
Ошибки обработки исключений	2	50	1
Не оптимизированный код	3	50	1,5
Некорректная логика кода	12	75	9
Ошибки планирования	7	10	0,7
Ошибки статических данных	22	10	2,2
Ошибки сценариев оболочки	14	10	1,4
Ошибки баз данных	7	25	1,75
Ошибки среды (окружения)	25	50	12,5
Ошибки файлов данных	5	80	4
Ошибки отображения (построения схемы программы)	3	50	1,5
Итого	100		35,55

Применительно к методу обратной семантической трассировки можно предположить, что для приведенных типов ошибок он обеспечивает предотвращение или устранение указанных типов ошибок на некоторый процент I_T. Приведенные в табл. 1 значения I_T определялись методом экспертных оценок. Значения доли выявленных ошибок каждого типа вычисляются по формуле вида:

$$\Delta P_E = \frac{P_E \cdot I_T}{100}.$$

Таким образом, для приведенных в табл. 1 данных в итоге получаем, что использование метода семантического трассирования программ приводит к выявлению $\sum \Delta P_E = 35{,}55\,\%$ от общего числа ошибок всех типов.

Полученная оценка влияния метода обратного трассирования на число выявляемых ошибок позволяет оценить поправку, которую нужно внести в значение коэффициента β_0 в формуле (1).

Для примера на рис. 1 приведен график вероятности π отказа в производном классе (DIT=1) в зависимости от значений π β_0. При этом в [7] рассматривался модуль с тремя атрибутами (NAI=3) и при условии, что в модуле есть два других класса с параметрами методов типа данного класса (OCMEC=2). Напомним, что стандартное значение, при котором метод обратного трассирования не выполняется, есть β_0=-3.97. При этом вычисленная по формуле (2) вероятность отказа есть $\pi = 0.5422$.

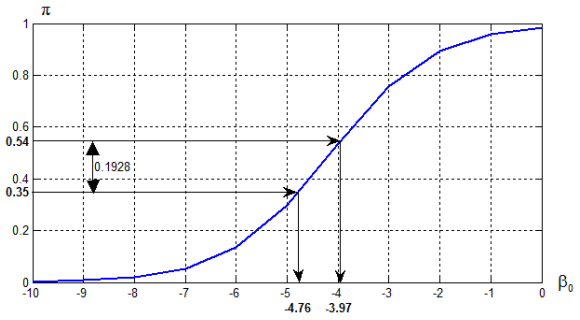

Рисунок 1. График вероятности отказа

Соответственно, для определения искомой поправки в значение коэффициента β_0 в формуле (1) выполним следующее:

1. Вычислим вероятность отказа при использовании метода обратной семантической трассировки как разность 0.5422 – 0.3555*0.5422 = 0. 5422 – 0.1928 = 0.3494. Первый член разности – исходная вероятность возникновения отказа, а второй член разности – нормализованное значение доли выявленных ошибок всех типов при использовании метода обратной трассировки.

2. По вычисленному значению 0.3494 вероятности π отказа в производном классе определим соответствующее ему значение коэффициента β_0 в формуле (1). Для приведенного на рис. 1 графика получим приближенное значение $\beta_0 \approx -4{,}76$.

5. Заключение

Модели, аналогичные описанной нами, можно построить и для других наборов метрик сложности, в том числе на более поздних этапах жизненного цикла. Например, метрики Холстеда и МакКейба, функциональные точки и др. После начала отладки, когда появляются данные о распределении отказов, становится возможным учитывать их в прогнозной оценке надежности программ, в том числе программ, разрабатываемых с помощью обратной семантической трассировки.

В предложенном подходе наиболее проблематичным представляется получение точной экспертной оценки того, насколько метод обратной семантической трассировки влияет на возможность уменьшения соответствующего типа ошибок. В связи с этим представляется целесообразным продолжить исследования в направлении экспериментальной оценки влияния обратной семантической трассировки на надежность программ.

Кроме того, представляется важным теоретическое обоснование правомочности использования предложенного подхода путем исследования свойств объектно-ориентированных метрик, например, по аналогии с подходом, использованным в работе [13].

Литература

1. Aizenbud-Reshef N., Nolan B. T. Model traceability / IBM Systems Journal, July-Sept. 2006, P. 515-526.
2. Zhereb K., Pavlov V., Doroshenko A. and Sergienko V. Using Reverse Semantic Traceability for Quality Control in Agile MSF-based Projects. Материалы конференции/Conference Materials. SEC(R) 2008, Software Engineering Conference (Russia). Moscow, Russia, October, 2008. – pp. 165-181.
3. Рогальчук В.В., Хомоненко А.Д. Метод обратной трассировки и оценивание его влияния на стоимость разработки программного обеспечения. Научно-технические ведомости СПбГПУ / Информатика. Телекоммуникации. Управление. 2008. № 4 (62). – С. 146-151.
4. Boehm B. «COCOMO II. Model definition manual» / http://sunset.usc.edu/csse/research/COCOMO II/cocomo2000.0/CII_modelman2000.0.pdf / 2000.
5. Chidamber S.R., Kemerer C.F. A metrics suite for object oriented design. Software Engineering, IEEE Transactions on Volume 20, Issue 6, Jun 1994. – pp. 476-493.
6. Basili V.R., Briand L.C., Melo W.L. A validation of object-oriented design metrics as quality indicators - IEEE Transactions on

Software Engineering, Volume 22, Issue 10, Oct 1996. – pp. 451-761.
7. El-Emam K. Object-oriented metrics: A review of theory and practice. National Research Council of Canada, Canada. NRC 44190, March 2001. – 32 p.
8. El-Emam K., Melo W. The prediction of faulty classes using object-oriented design metrics. National Research Council of Canada, Canada. NRC 43609, November 1999. – 24 p.
9. Романов В.Ю. Анализ программного обеспечения с использованием объектно-ориентированных метрик. Обзор метрик. МГУ им. Ломоносова. http://old.master.cmc.msu.ru/ romanov/russian/pub/OOMetrics-Report.htm.
10. Hosmer D. W., Lemeshow S. Applied Logistic Regression, 2nd ed. New York: Wiley, 2000.
11. Churamani N. Using Orthogonal Defect Classification for Defect Analysis. NIIT Technologies, New Delhi, India. 2007.
12. Ицыксон В.М. , Моисеев М.Ю., Цесько В.А., Карпенко А.В. Исследование систем автоматизации обнаружения дефектов в исходном коде программ. Научно-технические ведомости СПбГПУ / Информатика. Телекоммуникации. Управление. 2008. № 5 (65). – С. 119-127.
13. Misra S. and Ibrahim Akman I. Measuring Complexity of Object Oriented Programs. O. Gervasi et al. (Eds.): ICCSA/ Springer-Verlag Berlin Heidelberg, 2008, Part II, LNCS 5073, pp. 652–667.