О Методах Верификации и Мониторинга Программных Продуктов

Рамиз Шыхалиев
Институт Информационных Технологий НАНА, Баку, Азербайджан
ramiz@science.az

***Аннотация*— В работе представлен анализ методов верификации и мониторинга программных продуктов. Рассматриваются метрики программных продуктов и основные методы верификации программных продуктов, в частности, методы статической, формальной и динамической верификации.**

Ключевые слова— программные продукты, мониторинг программных продуктов, верификация программных продуктов, динамическая верификация, статическая верификация, формальная верификация

I. ВВЕДЕНИЕ

Сегодня, с ростом уровня информационных технологий, выросла и уровень технологий разработки программных продуктов. Вместе с тем, расширились области применения программных продуктов и намного вырасло их роль и важность в жизни общества. В результате этого, жизнь общества становиться зависимым от программных продуктов. Программный продукт – это набор компьютерных программ, процедур, документации и данных [1].

Однако, несмотря на рост уровня технологий разработки программных продуктов, все же в них случаются ошибки. Одной из главных причин появления ошибок в программных продуктах является рост их сложности. Так как, исходные коды программных продуктов могут состоят из миллонов строк. Поэтому обнаружить ошибки программных продуктов становиться очень сложной задачей. Примерно 80% затрат осуществляемой на разработку программных продуктов тратятся на выявление и устранения ошибок [2].

В зависимости от области и масштаба применения программных продуктов, их ошибки могут привести к последствиям различной тяжести. Если ошибки случаются в программных продуктах предназначеных для критических систем, например, в медицинсих, энергетических, технологических и др., то последствия этих ошибок могут быть катастрофическими и даже фатальными [3, 4]. Кроме того, ошибки программных продуктов также могут привести к очень большим экономическим потерям [2, 5].

Для гарантирования того, что программные продукты работают адекватно поставленным задачам, то есть для обеспечения гарантированного высокого качества программных продуктов, необходимо провести верификацию, который позволить выявлять ошибки на разных этапах процесса разработки программных продуктов.

Верификация программных продуктов – это мероприятие по обеспечению качества программных продуктов, которое направлено на то, чтобы программные продукты разрабатывались в соответствии с процессом разработки [6].

Целю статьи является анализ основных принципов методов верификации и мониторинга программных продуктов. Анализ этих методов поможет понять их цели и назначения и выбрать необходимый метод для мониторинга программных продуктов.

II. МЕТРИКИ ПРОГРАММНЫХ ПРОДУКТОВ

Для мониторинга любой области нобходимо иметь некоторые метрики. Целью метрик мониторинга программных продуктов является выявление и измерение основных параметров, влияющих на разработку программных продуктов.

Метрики программных продуктов собираются на разных этапах разработки программных продуктов и используются для оценки их качества. В частности, эти метрики позволяют количественно оценить и прогнозировать стоимость программных продуктов, измерять и прогнозировать производительность, качество и сложность программных продуктов и т.д. Метрики программных продуктов также могут быть использованы для идентификации программных модулей потенциально подверженных ошибкам [7, 8, 9]. При этом, для статистического анализа метрик программных продуктов используются такие методы, как логистическая регрессия, линейная регрессия наименьших квадратов, регрессия Пуассона и т. д.

В общем, метрики программных продуктов – это числовые величины, которые извлекаются из программного проекта. Существует два типа метрик, а именно метрики продукта и метрики процессов [10]. Метрики продукта – это числовые величины, которые извлекаются из некоторого документа или части исходного кода, а метрики процессов отображают числовые значения прогаммных процессов, например, значение времени, требуемое для отладки модуля программного продукта.

Вместе с тем, метрики программных продуктов классифицируются как метрики результатов и метрики

DOI: 10.25045/NCSoftEng.2017.09

прогнозов. В свою очередь, метрики прогнозов являются метриками продуктов, которые могут использоваться для прогнозирования значений других показателей. При этом, использование характеристик системной спецификации для прогнозирования количества ресурсов, необходимых для программного проекта, может служит примером метрик продукта (системная спецификация), который используется для прогнозирования метрики результата (ресурса проекта). Метрики могут быть использованы для измерения различных показателей программных продуктов, которым можно отнести следующие:

- оценка затрат и усилий на разработку программных продуктов;
- контроль и обеспечение качества программных продуктов;
- оценка производительности программных продуктов;
- оценка надежности программных продуктов;
- оценка структруы и сложности программных продуктов [11];
- оценка алгоритмической и вычислительной сложности программных продуктов;
- и т.д.

Несмотря на то, что сегодня разработаны различные метрики программных продуктов, нет единой метрики или комбинации метрик которые могли бы прогнозировать качество программных продуктов в целом. Так как, склонность программных модулей к ошибкам, отличается от системы к системе и в такой ситуации традиционные статистические методы анализа метрик программных продуктов являются неадекватными. Для решения этого вопроса используются методы машинного обучения [12], которые позволяют более точно определить качество программных продуктов, чем линейные модели.

III. МЕТОДЫ ВЕРИФИКАЦИИ ПРОГРАММНЫХ ПРОДУКТОВ

Существует три основных подхода к верификации программных продуктов [13]:

✓ ***статическая верификация*** – используетя для доказательства корректности программных продуктов;

✓ ***формальная верификация*** – это математическая спецификация или проектирование верификации. Формальные методы верификации используются только при разработки программных продуктов;

✓ **динамическая верификация** – используетя для поиска ошибок в программных продуктах.

A. Статическая верификация

Статическая верификация – это процесс проверки соответствия программных продуктов требованиям путем проверки кодов до их запуска. Например:

– верификация соглашений о кодировании. Соглашении о кодировании представляют собой набор рекомендаций для конкретного языка программирования, которые включают стили, инструкции и методы программирования;

– обнаружение неправильного использования;

– расчет характеристик программных продуктов.

B. Формальная верификация

Существует два основных подхода к формальной верификации. Первый подход основан на методах доказательства теорем [14, 15, 16], то есть проверка соответствия программных продуктов требованиям и проектным решениям основывается на теоретическом доказательстве. В этом подходе системная спецификация состоит из набора декларативных операторов или декларативных предложений и обычно определяются свойства реальных и/или системных сущностей или объектов, их поведения и взаимодействия.

Второй подход к формальной верификации основан на модельной проверке [17, 18, 19, 20], ее также назвают теоретико-модельным подходом. В этом подходе система представляется операторной моделью, которая обычно отображает поведение системы и в качестве модели используется конечный автомат, состоящий из вершин, представляющих системные состояния и направленные ребра, представляющие поведение системы, вызывающее переходы состояний. При этом, каждое состояние системы определяется логическим или условным выражением, то есть система находится в том или ином состоянии тогда и только тогда, когда используя системные атрибуты условие оценивается как истинное. Формальная верификация на основе модельной проверкой начинается с исходного состояния системы и применяя операции генерирует состояния, При этом, необходимые свойства или ограничения проверяются для каждого из созданных состояний и в результате выявлются нарушении.

C. Динамическая верификация

Динамическая верификация осуществляется во время выполнения программных продуктов и динамически проверяет их поведения. Она также известная как тестирование программных продуктов и в зависимости от целей тестов могут быть классифицированы следущим образом:

✓ частичный тест, который проверяет одну функцию или класс (*unit test*);

✓ полный тест, который проверяет группу классов, например:

- тестирование модуля (одного модуля);
- интеграционное тестирование – это этап тестирования программного продуктов, в котором отдельные программные модули объединяются и тестируются как группа;
- тестирование системы (всей системы).

DOI: 10.25045/NCSoftEng.2017.09

✓ приёмочный тест – является формальным тестом и служить для проверки критериев приемлемости программных продуктов:

✓ функциональный тест;

✓ нефункциональный тест (производительность, стресс-тест).

Верификационный мониторинг относится к динамическим методам и состоит в протоколировании функционирования программных продуктов в обычных режимах работы и оценки их соответствия требованиям и проектным решениям, а также выявление и регистрация ошибок, которые были допущены во время разработки или модификации программных продуктов. Вместе с тем, верификационный мониторинг также позволить обеспечивать поддержку планирования, контроля и улучшения процесса разработки программных продуктов.

Эффективность мониторинга программных продуктов может быть обеспечена только тогда, когда он осуществляется в рамках конкретных целей, то есть он должен затрагивать важные вопросы процесса разработки и создания программных продуктов. При мониторинге программных продуктов используются различные методы верификации. Эти методы различаются по своему назначению, видам регистрируемой информации, способу получения данных о работе программных продуктов и их анализа и т.д.

Имитационный мониторинг (*simulation-based, hypervisor-based*) основан на протоколировании работы проверяемых программных продуктов имитатором, на котором выполняется. В работе [21] авторами представлена система мониторинга программ, использующая автоматное моделирование основанное на графе состояний извлеченных из программ на языке С посредством статического анализа. Для построения полного графа состояний предлагается метод анализа указателя псевдонима, чтобы решить функции указателей и получить фактические потоки управления. После компиляции программы оснащаются зондами для сообщения о внутренних состояниях при их запуске. В ядре системы Linux встроен программный монитор, который следит за состоянием программ из зондов и проверяет пути их выполнения. Предложенная система мониторинга программных продуктов может активно реагировать на ненормальное поведение программы и позволит защитить системы и программы от дальнейшего ущерба.

Для поддержания выполнения имитационого мониторинга, существуют аппаратные решения.

Профилирование. При профилировании (*profiling*) программных продуктов, обычно измеряются показатели производительности программы. Профилирование представляет из себя динамический анализ программ, при котором измеряетя, например, обьем (занимаемый память) или сложность программы (время выполнения), использование определенных команд или частота и продолжительность вызовов функций и т.д. Чаще всего профилирующая информация служит для оптимизации программных продуктов. Для профилирования программных продуктов используется исходный код программы или ее исполняемый двоичный код. Для этого, с помощью так называемого профилировщика (или профайлера кода) код преобразуется в некоторый инструмент. При этом, профилировщики могут использовать различные методы, такие как событийно-основанный метод, статистические, инструментальные и имитационное моделирование.

Мониторинг событий. Используемые для проверки качества программных продуктов методы, такие как модельная проверка, статический анализ кода и модульное тестирование и т.п., не имеют возможности для исследования и использования полного дерева состояний. Это особенно заметно в программных продуктах, которые имеют сложную систему функций состоящих из более простых взаимодействий, где изучение пользователем пространства взаимодействия является ключевым моментом. Для этого исползуется мониторинг основаный на событиях (*Event-Driven Monitoring*) [22, 23].

Методы мониторинга программных продуктов основанные на событиях, для получения метрик используют полную запись событий и их атрибутов (времени, содержания, задействованных объемов памяти и кэшей различных типов и пр.). Использование такой архитектуры может помочь гарантировать, что созданные программные продукты работают в пределах спецификации, повышая надежность такого программного обеспечения.

ЗАКЛЮЧЕНИЕ

Сегодня, с увеличением масштаба и сложности программных продуктов, становиться трудно гарантировать их высокое качество. Для решения этой проблемы необходимо провести верификационный мониторинг програмных продуктов. В зависити от оцениваемых характеристик качества программных продуктов, существуют различные методы верификационного мониторинга.

Работа посвящена анализу методов верификации и мониторинга программных продуктов. Правильный выбор методов верификационного мониторинга может привести к повышению качества программных продуктов и снижению количества появления возможных ошибок, а также к эффективному управлению процессом разработки программных продуктов.

ЛИТЕРАТУРА

[1] ISO/IEC 25000:2014 – Systems and software engineering -- Systems and software Quality Requirements and Evaluation (SQuaRE) -- Guide to SQuaRE.

[2] G. Tassey, The Economic Impacts of Inadequate Infrastructure for Software Testing, Final Report, NIST, 2002.

[3] https://www.cs.tau.ac.il/~nachumd/horror.html

[4] http://www.popmech.ru/technologies/46176-top-6-katastrof-proizoshedshikh-po-vine-programmnogo-obespecheniya/

[5] https://dev.by/lenta/main/10-samyh-dorogih-oshibok-v-razrabotke-po

[6] D. R. Wallace, R. U. Fujii. Software verification and validation: an overview, IEEE Software, vol. 6, no. 3, pp. 10-17, 1989.

DOI: 10.25045/NCSoftEng.2017.09

[7] N. Fenton, S. L. Pfleeger. Software Metrics: A Rigorous and Practical Approach, International Thomson Computer Press, London, UK, second edition, 1997.

[8] C. Lewerentz, F. Simon. A product metrics tool integrated into a software development environment. In S. Demeyer and J. Bosch, editors, Object-Oriented Technology (ECOOP'98 Workshop Reader), LNCS 1543, pages 256 - 257. Springer-Verlag, 1998.

[9] L.C. Briand, W.L. Melo, J. Wust, Assessing theapplicability of Fault-Proneness Models across Object-Oriented Software Projects, IEEE Trans. Software Eng., vol. 28, no. 7, pp. 706-720, 2002.

[10] D. Ince, Software Metrics, Introduction Infonnation And Software Technologx, vol. 32, no. 4, pp. 297-303, 1990.

[11] Mrinal Kanti Debbarma, Swapan Debbarma, Nikhil Debbarma, Kunal Chakma, Anupam Jamatia, A Review and Analysis of Software Complexity Metrics in Structural Testing, International Journal of Computer and Communication Engineering, vol. 2, no. 2, pp. 129-133, 2013

[12] I.Gondra, Applying machine learning to software fault-proneness prediction, System and Software, pp. 186-195, 2008.

[13] https://en.wikipedia.org/wiki/Software_verification

[14] Monty Newborn, Automated Theorem Proving: Theory and Practice, Springer, 2000.

[15] C.A.R. Hoare, An axiomatic basis for computer programming, CACM vol.12, no.10, 1969. pp. 576 - 583.

[16] C. A. R. Hoare, et al, Laws of programming, CACM, Vol. 30, No. 8, August 1987. pp. 672 - 687.

[17] M Edmund, Jr. Clarke, Orna Grumberg, D.A. Peled, Model Checking, The MIT Press, 1999.

[18] E. M. Clarke, Automatic Verification of Finite-State Concurrent Systems Using Temporal Logic Specifications, ACM Transactions on Programming Languages and Systems, Vol. 8, No. 2, pp. 244 -263, April, 1986.

[19] Gerard J. Holzmann, The Model Checker SPIN, IEEE Transactions on Software Engineering, vol. 23, No. 5, May 1997.

[20] T. Henzinger et al., Symbolic Model Checking for Real-Time Systems, Proceedings, The Seventh Annual IEEE Symposium on Logic in Computer Science, pp. 394-406, 1992.

[21] Xiang LingBo Huang Guoqing Wu, Program monitoring based on automaton simulation, Wuhan University Journal of Natural Sciences, vol. 18, no. 2, pp 102-108, 2013.

[22] Gerard J. Holzmann, Margaret H. Smith, A practical method for verifying event-driven software, Proceedings of the 21st international conference on Software engineering pp. 597-607, 1999.

[23] Chris Lewis, Jim Whitehead, Runtime repair of software faults using event-driven monitoring, Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering, vol. 2, pp. 275-280, 2010.

DOI: 10.25045/NCSoftEng.2017.09