НАДЕЖНОСТЬ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Лекция 1. Введение. Основные понятия. Верификация и валидация

Информационные технологии являются одним из основных элементов инфраструктуры современного общества. Они служат базой для экономической деятельности и социального и культурного развития человечества, обеспечивая людям доступ к огромным массивам разнообразной информации и связывая их друг с другом, где бы они не находились.

Любая информационная система состоит из аппаратного и программного обеспечения (ПО). В начале развития компьютерной техники аппаратная часть была более сложной и значительно более дорогостоящей, стоимость программной части оценивалась примерно в 5% стоимости всей системы. Однако гибкость программного обеспечения и (как оказалось впоследствии, обманчивая) простота внесения в него изменений побуждали использовать его для решения разнообразнейших задач на одном и том же или стандартизированном аппаратном обеспечении. Поэтому постепенно ПО усложнялось, приобретало все большую ценность, и в последние десятилетия его стоимость достигает от 30% до 90% стоимости систем, в зависимости от их типа [1]. Совокупные затраты на создание, развитие и поддержку ПО уже превосходят соответствующие затраты на аппаратное обеспечение [2]. Сложность же современных программных комплексов такова, что многие исследователи считают их самыми сложными системами, созданными человеком [3].

Возрастающая сложность ПО приводит к увеличению количества ошибок в нем, а одновременный рост количества и критичности выполняемых им функций влечет рост ущерба от этих ошибок. Оценки потерь одной экономики США от некачественного программного обеспечения дают около 60 миллиардов долларов в год [4]. Известны также примеры серьезных ошибок в ПО, приведших к потере человеческих жизней, космических аппаратов или к масштабным нарушениям работы инфраструктурных сетей [5-11]. Одна из первых хорошо описанных ошибок такого рода — ошибка в системе управления космическим аппаратом Mariner 1 [5], которая привела к потере этого аппарата 22 июля 1962 года. Ошибка заключалась в том, что в одном месте была пропущена операция усреднения скорости корабля по нескольким последовательно измеренным значениям. В результате колебания значения скорости,вызванные ошибками измерений, стали рассматриваться системой как реальные, и она попыталась предпринять

корректирующие действия, которые привели к полной неуправляемости аппарата. Именно после этого инцидента управление военно-воздушных сил США приняло решение использовать в процессе разработки ПО экпертизу кода — его просмотр и анализ другими людьми, помимо самого разработчика.

При построении систем определенного уровня сложности люди в принципе не могут избежать ошибок, просто потому, что им вообще свойственно ошибаться, а возрастающая сложность предоставляет все больше возможностей для ошибок, при этом затрудняя их быстрое обнаружение. Для обеспечения корректности и надежности работы таких систем большое значение имеют различные методы верификации и валидации, позволяющие выявлять ошибки на разных этапах разработки и сопровождения ПО, чтобы последовательно устранять их.

Цель нашего лекционного курса — представить обзор разнообразных методов верификации ПО. Но прежде, чем перейти к самому обзору, необходимо напомнить определения основных используемых понятий и определить место верификации среди других видов деятельности, используемых при разработке и сопровождении ПО.

Основные понятия

Под ***жизненным циклом программного обеспечения*** обычно понимают весь интервал времени от момента зарождения идеи о том, чтобы создать или приобрести программную систему для решения определенных задач, до момента полного прекращения использования последней ее версии. Жизненным циклом этот период назван по аналогии с циклом жизни растения или животного, которое рождается, проходит определенные фазы роста и развития и в итоге погибает, давая жизнь новым существам, проходящим через те же стадии.

Описать общую структуру жизненного цикла произвольной программной системы, по-видимому, невозможно — слишком сильно отличается разработка и развитие ПО, предназначенного для решения разных задач в различных окружениях. Однако можно определить набор понятий, в терминах которых описывается любая такая структура — это, прежде всего, виды деятельности, роли и артефакты.

Вид деятельности в жизненном цикле ПО — это набор действий, направленных на решение одной задачи или группы тесно связанных задач в рамках разработки и сопровождения ПО. Примерами видов деятельности являются анализ предметной области,

выделение и описание требований, проектирование, разработка кода, тестирование, управление конфигурациями, развертывание.

Роль в жизненном цикле ПО — это профессиональная специализация людей, участвующих в работах по созданию или сопровождению ПО (или затрагиваемых ими) и имеющих одинаковые интересы или решающих одни и те же задачи по отношению к этому ПО. Примеры ролей: бизнес-аналитик, инженер по требованиям, архитектор, проектировщик пользовательского интерфейса, программист-кодировщик, технический писатель, тестировщик, руководитель проекта, пользователь, администратор системы.

Артефактами жизненного цикла ПО называются различные информационные сущности, документы и модели, создаваемые или используемые в ходе разработки и сопровождения ПО. Так, артефактами являются техническое задание, описание архитектуры, модель предметной области на каком-либо графическом языке, исходный код, пользовательская документация и т.д. Различные модели, используемые отдельными разработчиками при создании и анализе ПО, но не зафиксированные в виде доступных другим людям документов, не могут считаться артефактами.

Верификация и валидация

Верификация и валидация являются видами деятельности, направленными на контроль качества программного обеспечения и обнаружение ошибок в нем. Имея общую цель, они отличаются источниками проверяемых в их ходе свойств, правил и ограничений, нарушение которых считается ошибкой.

Верификация проверяет соответствие одних создаваемых в ходе разработки и сопровождения ПО артефактов другим, ранее созданным или используемым в качестве исходных данных, а также соответствие этих артефактов и процессов их разработки правилам и стандартам. В частности, верификация проверяет соответствие между нормами стандартов, описанием требований (техническим заданием) к ПО, проектными решениями, исходным кодом, пользовательской документацией и функционированием самого ПО. Кроме того, проверяется, что требования, проектные решения, документация и код оформлены в соответствии с нормами и стандартами, принятыми в данной стране, отрасли и организации при разработке ПО, а также — что при их создании выполнялись все указанные в стандартах операции, в нужной последовательности. Обнаруживаемые при верификации ошибки и дефекты являются расхождениями или противоречиями между несколькими из перечисленных документов, между документами и реальной

работой программы, между нормами стандартов и реальным процессами разработки и сопровождения ПО. При этом принятие решения о том, какой именно документ подлежит исправлению (может быть, и оба) является отдельной задачей.

Валидация проверяет соответствие любых создаваемых или используемых в ходе разработки и сопровождения ПО артефактов нуждам и потребностям пользователей и заказчиков этого ПО, с учетом законов предметной области и ограничений контекста использования ПО. Эти нужды и потребности чаще всего не зафиксированы документально — при фиксации они превращаются в описание требований, один из артефактов процесса разработки ПО. Поэтому валидация являетсяменее формализованной деятельностью, чем верификация. Она всегда проводится с участием представителей заказчиков, пользователей, бизнес-аналитиков или экспертов в предметной области — тех, чье мнение можно считать достаточно хорошим выражением реальных нужд и потребностей пользователей, заказчиков и других заинтересованных лиц. Методы ее выполнения часто используют специфические техники выявления знаний и действительных потребностей участников.

Различие между верификацией и валидацией проиллюстрировано на Рис. 1.

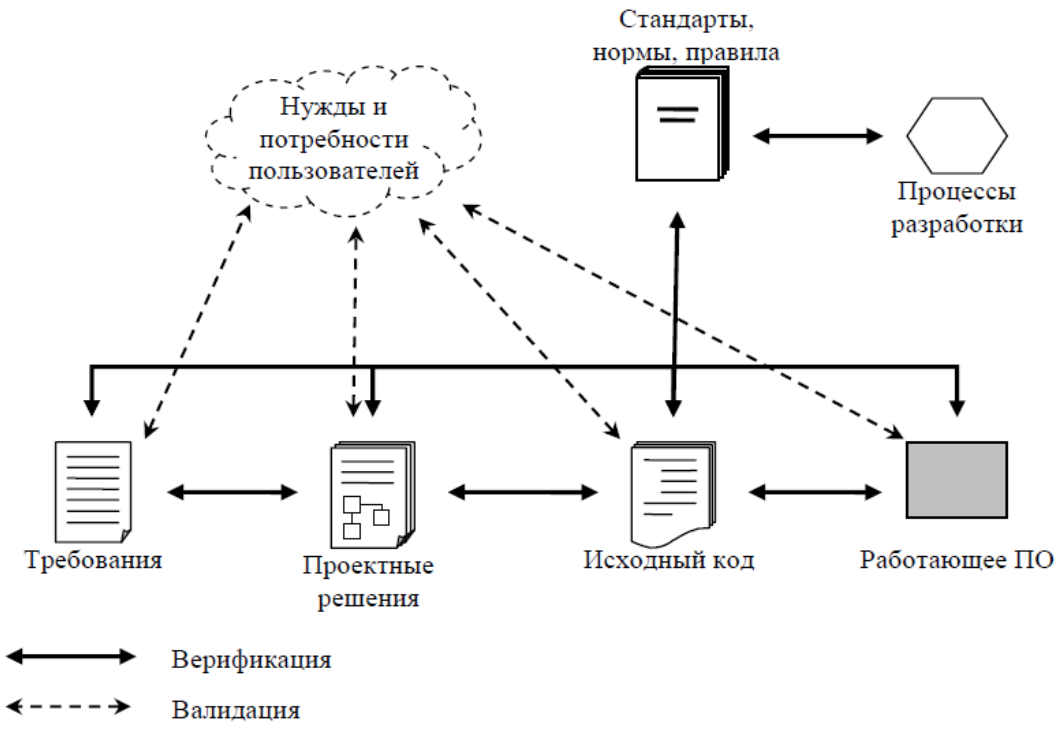

Рисунок 1. Соотношение верификации и валидации.

Литература

[1] B. W. Boehm. *Software Engineering Economics*. Englewood Cliffs, NJ: Prentice-Hall PTR, 1981. Русский перевод: Б. У. Боэм. *Инженерное проектирование программного обеспечения*. М.: Радо и связь, 1985.

[2] H. Miller, J. Sanders. *Scoping the Global Market: Size Is Just Part of the Story*. IT Professional, 1(2):49-54, 1999.

[3] F. P. Brooks. *No Silver Bullet — Essence and Accidents of Software Engineering*. Proceedings of the IFIP 10-th World Computing Conference, pp. 1069-1076, 1986. Издана по-русски в сборнике Ф. Брукс. *Мифический человеко-месяц, или Как создаются программные системы*. СПб.: Символ-Плюс, 1999.

[4] *The Economic Impacts of Inadequate Infrastructure for Software Testing*. NIST Report, May 2002. http://www.nist.gov/director/prog-ofc/report02-3.pdf

[5] http://nssdc.gsfc.nasa.gov/nmc/tmp/MARIN1.html

[6] N. Levenson, C. S. Turner. *An Investigation of the Therac-25 Accidents*. IEEE Computer, 26(7):18-41, July 1993.

[7] R. Z. Sagdeev, A. V. Zakharov. *Brief history of the Phobos mission*. Nature 341:581-585, 1989.

[8] G. N. Lewis, S. Fetter, L. Gronlund. *Casualties and Damage from Scud Attacks in the 1991 Gulf War*, 1993. http://web.mit.edu/ssp/Publications/working_papers/wp93-2.pdf.

[9] http://www.ima.umn.edu/~arnold/disasters/ariane5rep.html

[10] *Mars Climate Orbiter Mishap Investigation Board Phase I Report*, 1999. ftp://ftp.hq.nasa.gov/pub/pao/reports/1999/MCO_report.pdf

[11] http://www.nyiso.com/public/webdocs/newsroom/press_releases/2005/blackout_rpt_final.pdf