УДК 512.086

Б.Н.СОКОЛОВ, НТУ «ХПИ»

О ПОДХОДАХ К УЧЕТУ ТРЕБОВАНИЙ КАЧЕСТВА ПРИ КОНЦЕПТУАЛЬНОМ ПРОЕКТИРОВАНИИ И РЕАЛИЗАЦИИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Процес розробки, вибору та розміщення програмних систем традиційно супроводжується контролем якості. Для того, щоби бути конкурентоспроможними на ринку програмного забезпечення країни, у якій вони розробляються та на світовій арені, програмні системи повинні бути розробленими у відповідності до одного з визнаних стандартів якості. У наші дні значно зростають витрати на супровід програмних систем через недостатній ступень врахованості вимог якості. Використання одного з підходів до врахування вимог якості сприятиме зниженню витрат на супровід складних систем та збільшить їх конкурентоспроможність на ринку.

The process of development, choosing and deployment of software is traditionally followed by the quality control. To be competitive on the domestic and world software market, software products should be developed corresponding to one of the world-accepted quality standards. Nowadays costs for maintenance of software products are extremely growing because of insufficient level of suitability of quality requirements. Using one of this approaches will contribute to decreasing maintenance charges and increase competitiveness on software market.

1. Введение. В настоящее время к программным системам и к технологическим процессам их производства применяются различные методики контроля качества (стандарты ISO, EFQM [1,2]; модели зрелости CMMI [3,4]) для повышения конкурентоспособности этих систем на рынке. Требования к программным системам принято делить на функциональные (некоторые требования к функциональности системы в соответствии с областью ее применения) и нефункциональные. Следует отметить, что корректней рассматривать функциональные и не функциональные требования вместе, так как пренебрежение определенной группой требований может привести к неконкурентоспособности системы на рынке программного обеспечения.

Подходы к учету требований качества при концептуальном проектировании и реализации программного обеспечения могут быть разделены на три группы. К первой группе можно отнести походы, сводящиеся к выбору архитектуры разрабатываемой системы из множества эталонных архитектур, заранее оцененных по ряду критериев. К этой же группе могут быть отнесены подходы, при которых на основе заданных требований качества и имеющихся шаблонов программного кода, осуществляется «сборка» каркаса программной системы. Ко второй группе могут быть

отнесены подходы, позволяющие отображать заданные требования качества в некоторые элементы программного кода. Третью группу образуют подходы, позволяющие преобразовывать требования качества в некоторое множество внешних или внутренних тестов. В соответствии с типом программной системы могут применяться различные вариации и комбинации подходов к учету требований качества.

План работы. В разделе 2 мы рассмотрим подходы к выбору архитектуры программной системы на основании требований качества. В разделе 3 мы обсудим возможность отображения требований качества в программном коде системы. Раздел 4 посвящен разработке программного обеспечения, управляемой тестами, созданными на основе требований качества.

2. Подходы к выбору архитектуры программной системы на основании требований качества. Здесь требования качества представляются в качестве некоторых параметров некоторой задачи выбора либо одной из эталонных архитектур программной системы, либо композиции каркаса программной системы на основе готовых блоков.

В первом случае необходимо иметь множество уже готовых *эталонных архитектур*, которым в соответствие поставлены некоторые оценки их характеристик качества. Для этого могут применяться различные методы оценки архитектур программных систем, такие как ATAM [5]. В зависимости от того, в каком виде получены эти оценки (например, нечеткие, вероятностные) могут применяться соответствующие методы принятия решений. Данные подходы скорее работают на уровне общей концепции, чем уже завершенного программного продукта. Они могут быть использованы как первый шаг при создании программного обеспечения на основе требований качества. Примеры применения данных подходов рассмотрены в [6,7].

Во втором случае нужны некоторые, настраиваемые в соответствии с определенными файлами настроек, *шаблоны функциональных блоков* программной системы. Такими функциональными блоками могут быть например настраиваемые блоки доступа к данным или некоторые элементы пользовательского интерфейса [8]. Главная проблема здесь – оценка этих функциональных блоков, т.к. необходимо учитывать, как добавление того или иного блока скажется на всей системе. Следует отметить, что применение метрик качества непосредственно программного кода (таких как расширяемость, переносимость и т.д.) здесь вряд ли позволит получить какие-либо выгоды от применения данной группы подходов. Поэтому функциональные блоки должны быть логически обособленными от других

участков программного кода и оцениваемыми в соответствии с выдвигаемыми требованиями качества.

3. Подходы, позволяющие отображать заданные требования качества в некоторые элементы программного кода. В данном случае требования качества добавляются некоторым образом в программный код. В данном случае возникает не только проблема формализации требований качества, но и способов их отображения в программном коде, а также интерпретации этого отображения. В большинстве своем подходы данной группы ориентированы на языки программирования, в программный код на которых есть возможность добавлять такие объекты как *аспекты* и *аннотации*. В качестве примера можем привести такие языки программирования как Java и C#.

Наиболее простым способом отображения является добавление в каркас программного кода некоторых комментариев, содержащих информацию о соответствующих ограничениях. В данном случае функцию интерпретатора выполняет сам разработчик. Отличие от записи данных требований в проектной документации заключается в том, что комментарии могут находиться в области зрения программиста непосредственно в процессе разработки соответствующих элементов системы.

Более сложным в реализации является преобразование требований в некоторую интерпретируемую метаинформацию. Одним из возможных вариантов решения является применение аннотаций. Аннотации можно поставить в соответствие конкретным элементам программной системы. В соответствии с информацией, заключенной в аннотации к элементу, возможны, например, различные способы его обработки, передачи по сети, хранения и т.д.

Также определенный интерес представляют генерация предупреждений и ошибок компиляции. Безусловным минусом тут будет то, нам необходимо ориентироваться на механизмы генерации предупреждений и ошибок компиляции, которые могут достаточно сильно отличаться друг от друга. Не стоит, однако, полагать, что в данном случае мы обречены на работу непосредственно с компилятором и должны работать исключительно на «низком» уровне. Механизмы объявления предупреждений и ошибок компиляции предоставляет нам аспектно-ориентированное программирование. Например, такие механизмы есть в AspectJ [9].

При использовании генерации предупреждений и ошибок компиляции внимание разработчика все время будет привлекаться к требованиям качества, которым должен соответствовать программный продукт им разрабатываемый. Генерация предупреждений и ошибок компиляции предоставляет достаточно широкое поле для внедрения требований качества в программный код вплоть до проверок на соответствие части программного кода определенным шаблонам проектирования.

4. Подходы, использующие различные виды тестов, разработанных на основе требований качества. В данном случае можно говорить о разработке, *управляемой тестами* (этот общепринятый сейчас термин впервые стал применяться в экстремальном программировании [10]). В данном случае этап разработки системы тестов предшествует разработке непосредственно программного кода. Это позволяет выявлять несоответствия требованиям качества на ранних стадиях разработки.

Тесты в значительной степени зависят от архитектуры разрабатываемой системы. Создание тестов для распределенных приложений представляется достаточно проблематичным [11].

Тесты можно условно разделить на внешние и внутренние. Внешние тесты применяются, как правило, к собранным элементам системы или ко всей системе. Внутренние тесты применяются для тестирования определенной функциональности системы, путем создания и последующей работы с экземплярами элементов системы.

Внешнее тестирование программного обеспечения, управляемое требованиями качества, подразумевает настройку уже существующих систем внешнего тестирования или создание необходимых внешних тестов. Среди достоинств внешних тестов можем указать отсутствие жесткой привязки к конкретному языку или среде программирования, обособленность по отношению к тестируемой системе и возможность их повторного использования. На данный момент внешние тесты получили достаточно широкое распространение. В соответствии с целью тестирования их можно разделить на несколько подклассов. Граничные внешние тесты призваны генерировать для тестируемой системы проблемные ситуации и отслеживать обработку системой этих ситуаций. Функциональные тесты проверяют, насколько хорошо тестируемая система соответствует поставленным перед ней функциональным требования (простой пример: существует входной файл с некоторыми исходными данными, которые задаются тестируемой системе, и эталонный выходной файл, который сравнивается с выходом системы). Тесты графического интерфейса имитируют работу пользователя с графическим интерфейсом программного обеспечения. Тесты на загруженность системы позволяют определить максимальную загруженность системы, при которой она работает стабильно. Стрессовые тесты близки по воздействию на тестируемую систему к тестам на загруженность, однако, их целью является выявление слабых мест в системе.

Внутренние тесты в наше время принято выделять в отдельные структурные элементы проекта, которые бы можно было безболезненно исключить из поставляемых конечным пользователям вариантов системы. В этой области наиболее часто применяются Unit-тесты, реализации которых существуют в большинстве языках программирования высокого уровня. Ранее также использовались специальные функции для тестирования внутри программного кода самой системы (примером может служить main-тестирование на Java), однако, в настоящее время применение подобной методики тестирования считается дурным тоном. Преимуществом внутренних тестов является то, что они частично или полностью могут быть сгенерированы на основании требований качества. Тестирование с помощью Unit-тестов может быть разделено, в зависимости от уровня доступа к внутренним элементам тестируемого класса, на тестирование «белого ящика» и тестирование «черного ящика».

5. Выводы. Несмотря на то, что стандарты, применяемые при оценке качества программных систем и процесса их производства, используются и в других областях, многие подходы к учету требований качества при концептуальном проектировании и реализации программного обеспечения не имеют аналогов. Наиболее развитый аппарат имеют методы выбора архитектуры системы среди эталонных. Подходы их разных групп применяются на различных стадиях разработки программных систем и не исключается возможность их комбинирования.

Список литературы: 1. *В. Липаев.* Процессы и стандарты жизненного цикла программных систем. М: СИНТЕГ, 2006. **2.** *W.Suryn, A.Abran, C.Laporte.* A integrated lifecycle model for general public market software products // Proceedings of Software Quality Management XII Conference, 2004. **3.** *В. Липаев.* Модели зрелости программной инженерии — CMMI. Содержание и применение // Jet Info, Информационный бюллетень, № 6, 2006. **4.** *К. Мильман, С. Мильман.* CMMI – шаг в будущее // Открытые системы, № 5-6 (2005), № 2 (2006). **5.** *R.Kazman, M.Klein, P.Clemens.* ATAM: Method of Architecture Evaluation. Technical Report // CMU/SEI-2000-TR-004, CMU, 2000. **6.** *T.Al-Haeem, I.Gorton, V.Ali Babar, F.Rabchi, B.Bennatallah.* A Quality-Driven Systematic Approach for Architechting Distributed Software Applications // URL: http://ieeexplore.ieee.org/iel5/10409/33070/01553567.pdf. **7.** *M.Svahnberg, C.Wholin, L.Lundber.* A Quality-Driven Decision-Support Method for Identifying Software. In: Int. Journal of Software Engineering and Knowledge Engineering, 2003. 13(5): p. 547-573. **8.** *J.Herrington.* Code Generation in Action. Manning, 2003. **9.** *R.Pawlak, L.Seinturier, J-Ph.Retaille*. Foundations of AOP for J2EE Development. Apress, 2005. **10.** *K.Back*. Extreme Programming Explained: Embrace Changed. Addison-Wesley, 1999. **11.** *R.Johnson*. J2EE Design and Development. Indianapolis: Wrox, 2003.

Поступила в редколлегию 24.02.08

УДК 512.086

И. СУРЖЕНКО, О. ГОРЧЕНОК, ***Е. ЛЮТЫЙ,*** НТУ «ХПИ»

О ПОДХОДАХ К ОБРАБОТКЕ ТРЕБОВАНИЙ КАЧЕСТВА НА ОСНОВЕ АНАЛИЗА ЕСТЕСТВЕННОГО ЯЗЫКА

У роботі розглядаються можливі напрямки досліджень у галузі обробки вимог якості до системи на основі аналізу природньої мови. Сформульовано мету дослідження та основні задачі, зокрема, задачі лінгвістичного аналізу тексту, виділення вимог якості, програмного подання семантики тексту.

The paper is devoted to discussing possible research directions in the area of eliciting the quality requirements to the system based on natural language specifications. The main goal and the specific research problems are formulated, in particular, the problem of parsing the specification, eliciting and classifying the requirements, program representation of the requirements semantics.

1. Роль требований качества к программному обеспечению. По традиции главное внимание при разработке программных систем уделялось реализации их функциональных возможностей. Такой подход к проектированию систем можно назвать “разработкой системы, управляемой функциональностью”. В последнее время, однако, ряд исследователей придерживается мнения, согласно которому функциональные возможности можно считать только частью того, что должно управлять процессом разработки программного обеспечения. Наиболее важным управляющим фактором в данном случае выступает *системное качество*.

Качество программной системы в формулировке, использованной Международной организацией по стандартизации (ISO), может быть понято как свойство системы, которое делает эту систему соответствующей ее намеченному или фактическому использованию. Корректное выполнение необходимых функций теперь рассматривается как одна из характеристик качества системы.

Необходимо различать требования качества к системе и характеристики качества системы. Заинтересованные лица формируют требования качества. Разработчики программного обеспечения разрабатывают систему со специфическими качествами (характеристики качества). В пределах ограничений по бюджету, времени и доступным ресурсам создаваемая система должна иметь характеристики, соответствующие требованиям качества. Согласование решений может быть организовано, например, так, как это описано в [1].

Актуален вопрос о реализации автоматизированного сбора требований качества на основе различной информации представленной в матери-