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-

альном виде. В качестве одного из таких направлений выделяют анализ спецификаций требований.

Был предложен подход, предполагающий идентификацию требований качества к программному обеспечению по спецификациям на естественном языке. В основе данного подхода лежит универсальная классификация требований качества, состоящая из четырех граней: вид, представление, удовлетворение, роль [2]. Полная классификация представлена ниже [3].

Грань вида представлена следующими классификаторами: функциональность, надежность, применимость, эффективность, сопровождаемость, переносимость.

- Функциональность включает пригодность для применения, точность, способность к взаимодействию, защищенность.
- Надежность включает готовность, устойчивость к дефектам, восстанавливаемость.
- Применимость включает понятность, простоту в изучении, простоту использования, привлекательность.
- Эффективность включает временную эффективность и использование ресурсов.
- Сопровождаемость включает удобство анализа, изменяемость, стабильность, тестируемость.
- Переносимость включает адаптируемость, простоту установки, сосуществование, замещаемость.

Грань представления представлена классификаторами «оперативное», «количественное», «качественное», грань удовлетворения - классификаторами «жёсткое» и «мягкое», грань роли - классификаторами «предписывающая» и «допускающая».

Используя данную классификацию, мы можем четко разделить функциональные и нефункциональные требования. Главный критерий в этом процессе – грань вида. Согласно с ней первые три пункта функциональности являются функциональными требованиями (пригодность для применения, точность, способность к взаимодействию), остальные – нефункциональными.

Другие грани более точно определяют важность требований с точки зрения проектирования системы.

2. Процесс идентификации требований. Процесс идентификации требований качества в соответствии с [3] состоит из двух этапов. На первом происходит заполнение словаря требований. Исходные спецификации на

естественном языке с помощью подсистемы лингвистического разбора преобразуются в набор слов и связей между словами, обладающих лингвистическими характеристиками. Далее производится оценка пользователем исходных предложений, то есть отнесение их к определенному типу требований для каждой грани. После этого происходит расчет весов, характеризующих принадлежность каждого элемента к каждому типу требований по каждой грани. Полученные результаты сохраняются в базе данных, образуя словарь требований.

Второй этап состоит в идентификации требований качества на основе словаря требований. После получения обработанных подсистемой лингвистического разбора спецификаций для каждого элемента предложения находится множество элементов словаря, соответствующих данному элементу. На основе этой информации рассчитываются весовые коэффициенты элементов. Тип требований, получивший наибольший вес для каждой грани, и считается результирующим для данного элемента. Установка значения элемента для каждого среза предполагает сохранение этой информации во внутреннем формате программы.

3. Лингвистический разбор текста. Данное действие подразумевает под собой применение различных методов и алгоритмов с целью получения некоторого формализованного и легко поддающегося обработке представления натурального текста. Суть проблемы кроется в существовании множества различных неоднозначностей внутри естественных языков.

Как показано в работе [4], мысль при передаче от одного субъекта к другому в виде речи на естественном языке терпит два преобразования «один ко многим». Первый переход производится при трансляции источником смысла в текст и обеспечивается механизмом формирования синонимов. Второй переход осуществляется при трансляции приемником текста в смысл и обеспечивается существованием нескольких значений у одного слова или фразы.

Тем не менее, существуют различные гипотезы, которые пытаются так или иначе формализовать процесс моделирования текста в однозначной, математической форме.

Так, например, в [3] предложение рассматривается, как система слов и связей, смысл которой ищется по критерию наибольшей согласованности между словами. Под смыслом в данном случае понимается множество связей, которые присутствуют в предложении. В начале работы метода производится морфологический анализ слов и выделения свойств, которые характеризуют некоторые аспекты смысла и грамматики данного слова. Например, слово может быть существительным, родительного па-

дежа, женского рода, во множественном числе. Далее выполняется сравнение слов между собой и расчет несогласованности, который может характеризовать одновременно, как особенности человеческого мышления данной языковой группы, так и несовместимость слов по критериям оценки. После проведенных расчетов решается оптимизационная задача, которая находит наилучшее сочетание связей, при котором достигается наименьшая суммарная энтропия связей между словами. Также на решение накладывается следующее условие: граф, содержащий в качестве узлов - слова, а в качестве ребер – связи, должен быть связанным деревом.

Данный подход интересен упрощенным, с точки зрения программной реализации, процессом поиска решения, однако имеет ряд трудностей связанных с организацией процесса расчета и сравнения морфологических и семантических характеристик слов.

Существуют также и другие методологии, которые описывают различные формы представления лингвистических конструкций. Например, теория «Смысл-текст» [4], универсальная грамматика [5], основанные на грамматике Хомского (например, грамматика, описанная в [6]) и много других. Однако целью данной статьи является не описание существующих средств, а формулировка сложившейся проблемы и постановка задач для исследований в данной области.

4. Об универсальном представлении текста. Проблемность ситуации заключается не только в разработке подсистем лингвистического разбора текста или анализа смысловых конструкций для извлечения из них требований качества, но и в согласованности данных подсистем. Эта согласованность заключается в формировании общего формата представления семантики. Простейшим способом представления может быть дерево, которое используется в [3], но, например, в [4] используется многоуровневая структура представления, где каждый уровень может описывать одну из следующих составляющих смысла лингвистической конструкции:

- семантика;
- глубокий синтаксис;
- поверхностный синтаксис;
- глубокая морфология;
- поверхностная морфология;
- глубокая фонология;
- поверхностная фонология.

Таким образом возникает задача выбора, которая требует отдельного решения.

Следует отметить, что данный аспект автоматизированной системы сбора требований качества к программному обеспечению на основе спецификаций может привести к потере или искажению данных и при этом внешне создавать иллюзию корректной работы.

Выбор способа представления смысла должен выполняться в соответствии с методологическими средствами, которые реализуются в подсистемах лингвистического разбора и выявления требований качества.

5. Цели исследований и основные задачи. Автоматизированный анализ требований качества является перспективным направлением, которое может открыть новые возможности в области разработки программного обеспечения.

Целью исследовательских групп в данном направлении должна стать разработка наиболее адекватной, достоверной и производительной системы сбора требований качества на основе текста спецификаций.

Можно сформировать постановки задач, которые необходимо выполнять исследовательским группам в данной области для достижения поставленных целей.

Задача 1. Разработать теоретико-практические инструменты для реализации подсистемы лингвистического разбора текста. Результатом выполнения данной задачи должны стать:

- математическое обеспечение морфологического анализа;
- математическое обеспечение синтаксического анализа;
- алгоритмические решения наиболее сложных задач;
- при необходимости: словари, статистическая информация, результаты опроса экспертов.

Задача 2. Разработать теоретико-практические инструменты для реализации подсистемы выявления требований качества. Результатом выполнения данной задачи должны стать:

- математическое обеспечение расчета коэффициентов принадлежности частей текста к грани;
- алгоритмическое обеспечение идентификации частей текста
- при необходимости: словари, статистическая информация, результаты опроса экспертов.

Задача 3. Разработать способ представления программным обеспечением семантики текста. В результате выполнения данной задачи необходимо получить спецификацию модели данных, которая в состоянии адекватно представлять семантику лексических единиц и их совокупности в аспекте разработанного математического обеспечения.

Задача 4. Используя результаты предыдущих задач описать систему выявления требований качества на основе текста спецификаций, как целостной системы «разбор-представление-выявление». Тут необходимо провести операцию интеграции подсистем посредством универсального представления семантической информации. В качестве результата данной задачи должна быть спецификация системы в целом, а также точные спецификации всех подсистем.

Задача 5. Реализовать подсистемы, используя разработанное математическое и алгоритмическое обеспечение. Результатом данного этапа должна стать готовая к внедрению и эксплуатации система выявления требований качества из текстов спецификаций написанных естественным языком.

Выводы. Таким образом, выявление требований качества является неотъемлемой частью разработки программного обеспечения. Однако реализация систем для автоматизации данной задачи весьма усложнена в области выявления требований качества на основе спецификаций написанных естественным языком.

Для создания средств автоматизированного сбора требований качества к программному обеспечению необходимо провести дополнительные исследования по направлениям:

- лингвистический разбор (включает в себя морфологический, синтаксический разборы, а также ряд других алгоритмов, набор которых зависит от конкретной методологии);
- представление семантической информации средствами программного обеспечения;
- автоматизированное исследование синтаксической информации.

В результате проведения данных исследований необходимо получить математическое и алгоритмическое обеспечение системы выявления требований на основе спецификаций с последующей интеграцией и реализацией в виде программного обеспечения.

Список литературы: 1. *M.Glinz.* Rethinking the Notion of Non-Functional Requirements // Proc. Third World Congress for Software Quality (3WCSQ 2005), p. 55-64 **2.** *Kazman, M.Klein, P.Clemens.* ATAM: Method of Architecture Evaluation. Technical Report // CMU/SEI-2000-TR-004, CMU, 2000. **3.** *A.Kostanyan, V.Shekhovtsov.* Towards entropy-based requirements elicitation // Proc. ISTA'2007, LNI P-107, 2007. – P. 105-116. **4.** *I. Mel'čuk.* Vers une linguistique Sens-Texte. Leçon inaugurale Paris: Collège de France 1997. **5.** *N. Chomsky.* Barriers. Linguistic Inquiry Monograph 13. Cambridge: MIT Press, 1986. **6.** *С.Пинкер.* Язык как инстинкт. М.: Едиториал УРСС, 2004.

Поступила в редколлегию 22.02.08

УДК 512.086

Ю.ТОМИЛКО, НТУ «ХПИ»

АВТОМАТИЗИРОВАННОЕ РЕЦЕНЗИРОВАНИЕ ИСХОДНОГО КОДА НА ОСНОВЕ ОНТОЛОГИИ ПРАВИЛ

В статті коротко розглянута проблема рецензування вихідного коду. Проведен огляд прикладної програми FxCop, яка дозволяє автоматизувати процес реценцування. Показані складності існуючого процесу та запропоноване рішення на основі використання онтології правил. Представлені його переваги та запропоновані майбутні напрямки роботи.

In the paper the problem of code review is briefly reviewed. The survey on FxCop tool that allows automating the process of code review is conducted. The difficulties of the existent process are shown; solution based on the use of rules' ontology is presented. Its advantages are emphasized and future directions of work are outlined.

Введение. Под рецензированием исходного кода (code review) будем понимать следующее [1]: это особый вид инспектирования (inspection), в процессе которого просматривается и проверяется исходный код программ с целью обнаружения и исправления в нём ошибок допущенных при разработке программного обеспечения. В стандарте [2] определены 5 типов рецензирования программного обеспечения (software review - согласно стандарту, процесс, в ходе которого программный продукт подвергается проверке персоналом проекта, менеджерами, пользователями, заказчиками, представителями пользователей и другими заинтересованными лицами для критического разбора или одобрения): административное рецензирование, техническое рецензирование, инспектирование, сквозной контроль и аудит. Рецензирование исходного кода следует рассматривать как вид инспектирования, а не технического рецензирования (technical review). В литературе также используется термин инспектирование программ, см. например [3], т.е. просмотр и проверка исходного кода программного обеспечения с целью обнаружения в нём ошибок. Под рецензированием исходного кода понимается, как правило, менее формальный процесс.

Доказано, что инспектирование является эффективным методом обнаружения ошибок. Оно дешевле тестирования программного обеспечения, в работах [4, 5] показано, что инспектирование программного кода является более эффективным и менее дорогостоящим, чем тестирование. Согласно [6] более 60% ошибок в программном обеспечении можно обнаружить с помощью неформального исследования (инспектирования). Процесс инспектирования также может оценить качественные характери-