УДК 681.03

ОБЕСПЕЧЕНИЕ КАЧЕСТВА ФОРМИРОВАНИЯ КОНЦЕПТУАЛЬНОЙ МОДЕЛИ ТРЕБОВАНИЙ К ПРОГРАММНОМУ ОБЕСПЕЧЕНИЮ

Н.А. БАЖЕНОВ, Б.Н. СОКОЛОВ

Рассмотрена проблема формирования концептуальной модели требований к программному обеспечению (ПО) на основании требований, полученных от заинтересованных лиц. Модель формируется на основании метода семантико-синтаксического представления текста. Описана процедура контроля качества, состоящая из анализа релевантности документа, PAS (Predicate Argument Structure) анализа и валидации модели заинтересованными лицами. Приведен пример заполнения глоссариев.

ВВЕДЕНИЕ

Процесс программной инженерии принято разделять на последовательные функциональные стадии — процессы жизненного цикла программной системы [1]. На рис. 1 показаны 4 процесса (в соответствии с [2]), рассматриваемые в работах нашей научной группы. Два из них являются объектом данного исследования — это процессы сбора требований [3], анализа и обработки требований [4].

Этапы тестирования, поддержки, сопровождения, а также другие процессы организационной группы, касающиеся аспектов организации, управления и контроля разработкой в данном исследовании не рассматриваются, и на рис. 1 не представлены.

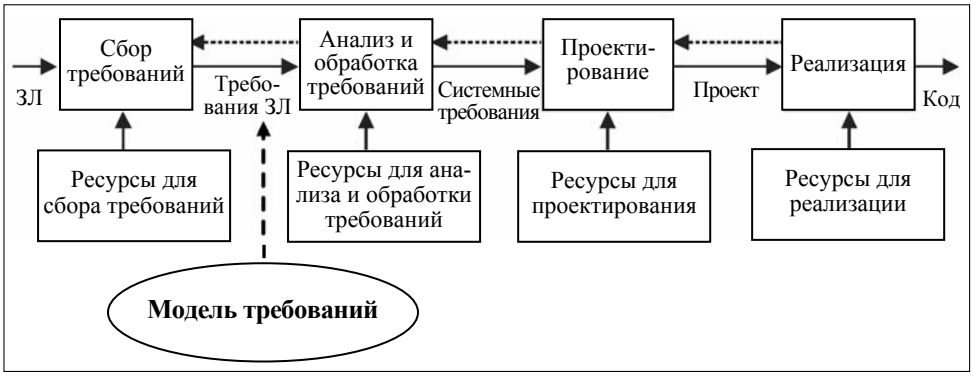

Рис. 1. Процессный подход к управлению качеством ПО

Качество программной системы зависит от качества проведения каждого процесса жизненного цикла, а значит и качества функциональных элементов, входных и выходных данных, ресурсов, необходимых для осуществления процесса. Таким образом, управление качеством процесса разработки проецируется на управление качеством отдельных процессов, а также на их элементы. В данной работе исследуется качество перехода от процесса сбора требований к процессу анализа и обработки, т.е. выходная информация одного процесса в тоже время является входной информацией последующего процесса.

Цель работы — разработка процедуры преобразования структурированных текстовых требований в концептуальную модель с обеспечением контроля качества данного преобразования на всех этапах данного процесса.

Для достижения необходимых результатов ставятся следующие задачи:

- разработка блока автоматизированной обработки текстов требований к ПО [5];
- создание концептуальной модели требований с дифференцированием функциональных и нефункциональных требований;
- разработка процедуры интерпретации структурированных требований и извлечения из них концептов модели требований;
- осуществление контроля качества на каждом шаге преобразования требований (в том числе для гибкого взаимодействия с другими процессами жизненного цикла).

Применению методов автоматизированной обработки естественного языка для обработки текстов посвящена работа [5]. Для решения задачи формирования модели требований в качестве базовых были использованы модели предложенные в [10–13]. В данной работе предлагается систематизировать результаты, получаемые на одних этапах, которые затем обрабатываются другими; разработать процедуру формирования модели требований; предложить подходы для контроля качества на всех внутренних блоках сбора, анализа и обработки требований. Научная новизна работы заключается в применении многошаговой стратегии, позволяющей повысить вероятность создания более качественной и жизнеспособной программной системы путем выявления ее узких мест на ранних стадиях разработки.

ФУНКЦИОНАЛЬНАЯ СХЕМА

На рис. 2 изображена схема взаимодействия рассматриваемых этапов жизненного цикла.

Процедура сбора требований представляет собой множество мероприятий по сбору информации от заинтересованных лиц (ЗЛ) и преобразованию ее в текстовую форму. Методы сбора могут быть различными: анкетирование, опрос, интервью и т.д.

Требования на естественном языке, представленные в текстовом виде, далее обрабатываются и трансформируются в структурированный семантико-базированный вид. Данная процедура осуществляется с помощью техник автоматизированной обработки текста на естественном языке [5]. Из структурированного семантико-базированного вида требования преобразуются

в концептуальную модель требований. Модель должна пройти верификацию ЗЛ. В случае, когда модель слишком сложна для адекватного восприятия ЗЛ, осуществляется ее преобразование к адаптированной форме. Если верификация не прошла успешно, цикл «сбор требований — преобразование требований — верификация требований» выполняется снова, либо полностью, либо частично (в зависимости от результатов верификации).

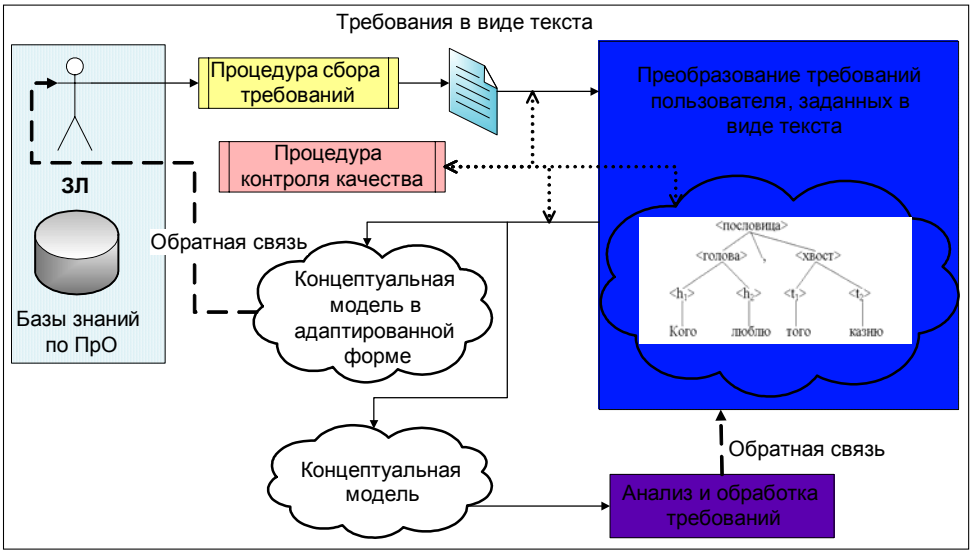

Рис. 2. Функциональная схема взаимодействия этапов

Концептуальная модель (без каких-либо адаптаций) поступает на вход процесса анализа и обработки требований, где с ней происходят дальнейшие преобразования. В начале данного процесса осуществляется анализ корректности требований, состоящий из проверки полноты и корректности требований, их тестируемости, осуществимости проектирования архитектуры системы, возможности использования и сопровождаемости системы [4]. Далее требования качества обрабатываются и анализируются, тем самым приводятся в форматы, которые было бы удобней использовать при проектировании и реализации системы. Так помимо различных спецификаций требования качества могут находить отражение в юнит-тестах, а также в метаинформации внедряемой непосредственно в код системы (например, в виде аннотаций языка java), которая, к примеру, может генерировать ошибки компиляции при нарушениях определенных требований (например, паттерн не был реализован) [6].

МОДЕЛЬ ТРЕБОВАНИЙ К ПО

Требования к ПО разделяются на два основных класса [7, 8] — требования к функциональным возможностям разрабатываемой системы и нефункциональные требования, т.н. требования качества (характеристики разрабатываемой системы необходимые для ее соответствующей работоспособности [9]). Таким образом, концептуальная модель требований представляется в виде модели функциональных требований и модели качества.

Модель функциональных требований. Используемая в данной работе модель функциональных требований базируется на подходе, разработанном в Клагенфуртском университете [10, 11]. В основе данного подхода лежит вовлечение ЗЛ на ранних этапах разработки ПО путем коммуникации и взаимодействия специалиста конкретной предметной области, не обладающего знаниями в области разработки ПО, со специальными интерфейсами, доступными этому специалисту. В качестве таких интерфейсов разработчиками подхода [10, 11] было предложено использование табличного представления концептуальных сущностей предметной области в виде глоссариев таких концептов. Метамодель модели функциональных требований изображена на рис. 3.

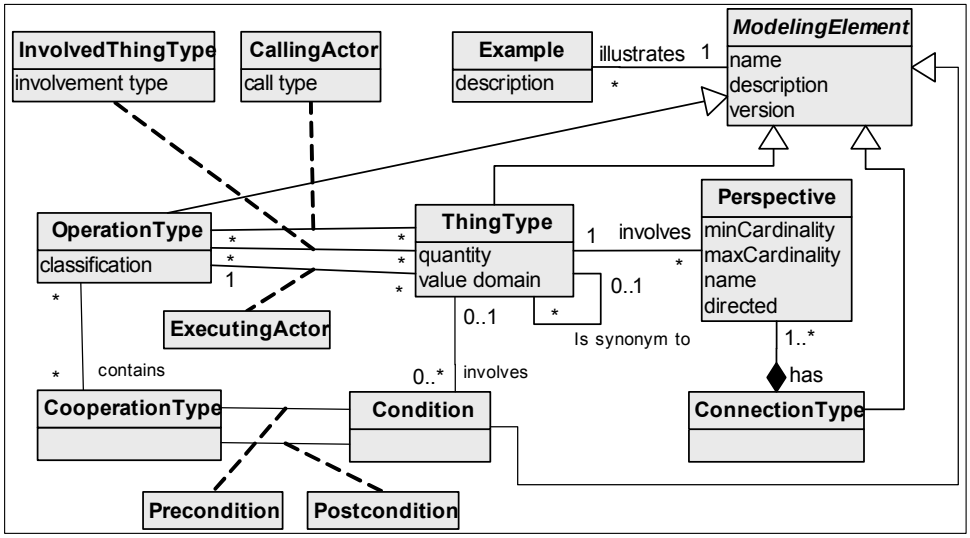

Рис. 3. Модель функциональных требований согласно [9]

Абстрактной сущностью модели, наследуемой всеми основными элементами модели, является концепт *ЭлементМодели* (ModelingElement). Каждый концепт, наследующий ЭлементМодели, может иметь множество *примеров* (*example*). Данная зависимость отображается на диаграмме с помощью связи «*illustrates*». Модель включает в себя статическую и динамическую части, содержащие процессно-инвариантную и процессно-зависимую информацию соответственно.

Тип-сущность (*thing-type)* и *тип-связь* (*connection-type*) составляют статическую подмодель. Тип-сущность обобщает понятие класса, атрибута, значения. Примерами таких концептов служат объекты, люди, ресурсы, параметры, характеристики объектов, абстрактные понятия и другие «сущности». В текстах требованиях типы-сущности представлены именными фразами, составными существительными и отдельными существительными. Может иметь атрибуты ожидаемой количественной величины (*quantity*) и значения допустимой области определения (*value domain*). Например, тип-сущность «Order date» может иметь количественную величину равную 365 и значение области определения «Дата». Если несколько типов-сущностей имеет синонимические значения с точки зрения предметной области, кото-

рую описывают данные требования, то между ними возникает зависимость *синоним* (*is synonym to*).

Типы-связи необходимы для отображения взаимосвязей между типами-сущностями. Два или более типа-сущности могут образовывать тип-связь. Для полного определения связи между сущностями необходимо описать ее с точки зрения всех сущностей, участвующих в данной связи. Из этого следует определение концепта *проекция* (*perspective*). Таким образом, каждый тип-сущность, состоящий в связи, имеет свою проекцию, которая может характеризироваться кардинальными числами (*minCardinality, maxCardinality*), именем и другими атрибутами. В свою очередь, из множества таких проекций образуется тип-связь.

Рассмотрим приведенное выше словосочетание «Order date». Из него можно извлечь следующую информацию: тип-сущность заказ (order) имеет атрибут дату заказа (order date), из этого следует, что между ними образуется тип-связь «Владение атрибутом» (attribute possessing), при этом типы-сущности участвуют в проекциях «имеет атрибут» (has an attribute) и «является атрибутом» (is an attribute of).

Тип-операция (*operation-type*) и *тип-кооперация* (*cooperation-type*) образуют динамическую подмодель. Тип-операция используется для определения разрешенных операций, их действующих лиц, объектов и параметров (*ExecutingActor, CallingActor, InvolvedThingType*). Данные динамические элементы могут иметь атрибут *классификация* (*classification*), показывающий к какой категории в данном домене относится тип-операция. Субъекты, объекты и параметры типов-операций представлены типами-сущностями.

Для моделирования бизнес-процессов и сервис-процессов используется элемент тип-кооперация. Он представляет собой триаду множеств <Prc, {A,O}, Poc>, где Prc — множество *предусловий* (*Precondition*), Poc — множество *постусловий* (*Postcondition*), {A,O} — множество пар, состоящих из типа-операции и субъекта, осуществляющего данную операцию. Каждое *условие* (*Condition*) может быть пред- и/или постусловием одного или более типа-кооперации. Условия могут быть выражены свойствами/состояниями объекта, событиям, завершенностью какой-либо деятельности, обстоятельственными/временными ограничениями, т.е. определяют начальное и конечное состояние объекта. Данное свойство условий влечет за собой наличие связи с объектами типов-сущностей, что отображено на диаграмме с помощью зависимости «*участвует*» (*involves*). Типы-кооперации могут быть сгруппированы сравнением их пред- и постусловий. Таким образом, образуется сеть, показывающая потоки процессов, условия их выполнения, а также результирующие условия. Множества типов-кооперaций могут представляться как в табличном варианте, так и в визуальном представлении в виде графа, что улучшает понимание и взаимодействие с ЗЛ на каждой итерации разработки.

Приведем примеры динамических элементов модели. Следующие предложения иллюстрируют некоторый бизнес-процесс поступления заказа: «*The order comes in… If the order comes in, the bookkeeping department checks the payment for possible authorization. If the payment is authorized and all ordered items are on stock, then the order department confirms the order. If the*

payment is not authorized, then the order department must reject the order…» [9]. Типом-операцией является оплата проверка оплаты «check payment» с выполняющим субъектом бухгалтерия «bookkeeping department» и вовлеченным параметром оплата «payment», которые являются типами-сущностями. Предусловием для проверки оплаты является поступление заказа «order comes in» с вовлеченным типом-сущностью заказ «order». Первым постусловием является проведенная оплата «*payment is authorized*», а альтернативным отклоненная оплата «*payment is not authorized*». В обоих случаях вовлеченным типом-сущностью является оплата «*payment*». Полученные постусловия, в свою очередь, является предусловиями для следующих типов-кооперций в иерархии, и так далее до окончательного формирования сети бизнес-процесса. Более детальное описание данного подхода дано в [9].

Модель требований качества. Представленная модель требований качества основывается на модели *QAPM-S* [12, 13] и расширяет, используемую в данной работе, модель *KCPM* [10, 11]. Метамодель модели концептов, отвечающих за характеристики качества, изображена на рис. 4. Связь с метамоделью функциональных требований осуществляется посредством общего мета-концепта *ЭлементМодели* (ModelingElement), а также типа-операции и типа-кооперации, задающих контекст использования.

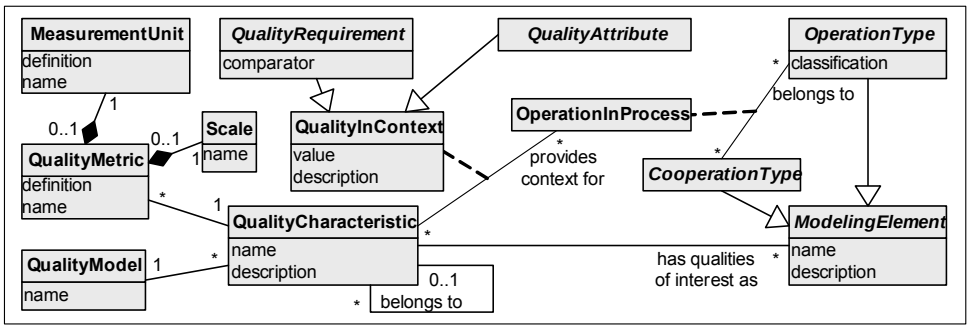

Рис. 4. Модель требований качества

Данный подход базируется на понятиях «категория качества», «характеристика качества», «метрика качества» и др., определенных в стандартах ISO [7, 8].

Требования качества находятся в зависимости от динамического контекста использования. Соответственно, контекст в каждом случае может быть определен, как пара $<E, O^E>$ (*OperationInProcess* в метамодели), где E — экземпляр *типа-кооперации* (*CooperationType*), а O^E — экземпляр *типа-операции* (*OperationType*), принадлежащего данному типа-кооперации. *Качество-в-контексте* (*QualityInContext*) имеет две специализации: *ТребованияКачества* (*QualityRequirements*) и *АтрибутыКачества* (*QualityAttribute*).

Например, у интернет-магазина могут существовать различные группы клиентов, но при этом мы можем получать различные контексты. Таким образом, в контексте «обычный клиент» *характеристика качества* (*QualityCharacteristic*) «доступность сервиса добавления заказа», соответствующая *метрики качества* (*QualityMetric*) «доступность» (Availability) может быть положена 95 %. В то же время для «VIP-клиента» она может быть равна 99,9 %.

ПРЕДЛОЖЕННАЯ МОДЕЛЬ NLP-ОБРАБОТКИ ТЕКСТОВ ТРЕБОВАНИЙ

Структурированное семантико-синтаксическое представление текста спецификаций требований

Согласно [3] спецификации требований к ПО после определенных процедур первичной обработки поступают на вход блока автоматизированного лингвистического анализа. Результатом работы данного блока является структурированный текст древовидного формата, с корневой вершиной, промежуточными узлами, характеризующими предложения и различные фразы, и с листьями, представленными отдельными лексемами с наборами определенных атрибутов. В данном исследовании для лингвистического анализа используется подход, основанный на теории контекстно-свободных грамматик с использованием статистического морфологического анализатора и основанного на правилах синтаксического парсера [14, 15]. Ограничением данного подхода является наличие практической применимости только для доменов с требованиями на английском языке, хотя возможность его практической адаптации к какому-либо природному языку присутствует. Общую схему структурированного предложения упрощенно можно показать на рис. 5. На данной схеме приведены следующие обозначения для фраз и частей речи: n0, n2, n3 — имя существительное и именные фразы; v0 — глагол и глагольные фразы; v0(aux) — вспомогательный глагол; pron0 — местоимение; a0, a2 — имя прилагательное и прилагательная (атрибутивная) фраза; adv0, adv2 — наречие и фраза обстоятельства; q0, q2 — имя числительное и фраза числительного; p0, p2 — предлог и предложная фраза; pt0 — частица; det0 — артикль, определитель; rel0 — относительное местоимение; sentence (inf) — инфинитивная группа; sentence (relative) — определительное придаточное предложение; sentence (subord) — придаточное предложение.

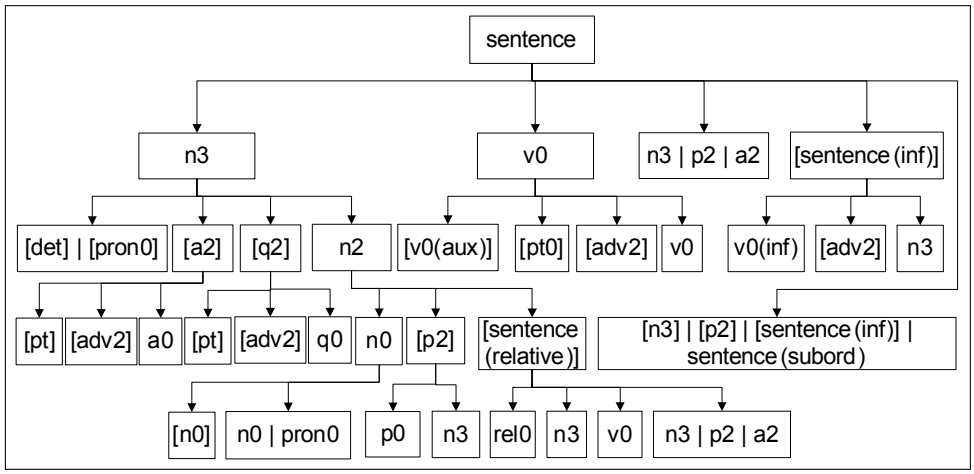

Рис. 5. Обобщенная структура предложения, получаемого в результате обработки

С помощью разработанных эвристических правил, описанных в работах [5, 15], осуществляется построение дерева текста требований с узлами в виде предложений, морфосинтаксических групп, фраз, словосочетаний и листьев, представленных отдельными лексемами.

Пример обработки предложения-требования «*The order department for each ordered item checks its availability on stock*» NLP-блоком приведен на рис. 6.

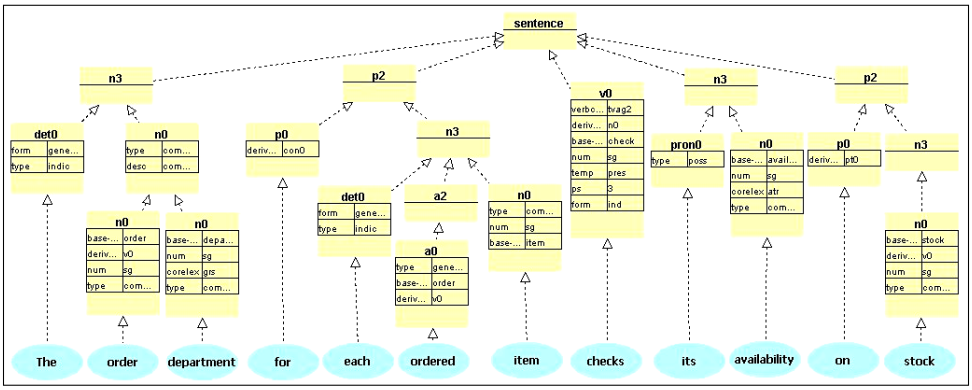

Рис. 6. Древовидное представление предложения

ИНТЕРПРЕТАЦИЯ ДРЕВОВИДНЫХ СТРУКТУР ТРЕБОВАНИЙ И ЗАПОЛНЕНИЕ ГЛОССАРИЕВ МОДЕЛИ ТРЕБОВАНИЙ

Формат структурированных данных требований далее используется для формирования концептуальной модели требований. Для выделения концептов и заполнения ими глоссариев модели необходимо использовать правила интерпретации. В данном исследовании предлагается взять в качестве базовой характеристики каждого предложения его предикатно-аргументную структуру (PAS — predicate argument structure), основанную на транзитивности и др. глагольных категориях [5]. Определяющие лексемы именных групп и части составных существительных формируют множество типов-сущностей. Отношения «включения» и «атрибутивности» образуют множество типов-связей. Семантико-синтаксическая роль глаголов в предложении определяет трансформацию динамической части модели. Если глагол имеет субъект-агент, то он указывает о наличии типа-операции, при этом существительные, вовлеченные в его управление, в зависимости от семантической роли (агент, субъект, цель) указывают на вовлеченные в процесс типы-сущности (*InvolvedThingType*), которые также попадают в глоссарий типов-операций. Если глагол не имеет субъекта-агента, то он становится кандидатом в пред- или постусловия для типов-кооперации.

Таким образом, сформируем базовый набор правил трансформации лингвистических элементов в элементы модели требований, используя [16–18]. В табл. 1 представлены правила интерпретации, необходимые для заполнения модели функциональных требований, а в табл. 2 — для модели требований качества.

Используя приведенные правила, механизм формирования модели требований заполняет глоссарии концептами модели. На рис. 7 схематически показан механизм заполнения глоссария типа-сущностей: в первом случае типами-сущностями становятся как части сложного существительного «order», «services», так и все сложное существительное в целом «order processing services»; во втором случае ведущая лексема («stock clerk») именной группы «(the/det0 (stock/n0 clerk/n0)/n0comp)/n3» становится типом-сущностью. При этом атрибут существительного «corelex=hum» помогает

определить классификационный признак типа-сущности «person». Процесс интерпретации не является автоматическим, а допускает изменение пользователем настроек правил по умолчанию. После выполнения обработки с помощью стандартного набора правил, если пользователь не удовлетворен результатами интерпретации, то он может или изменить правила и повторить процесс обработки, или управлять контентом моделей непосредственно через глоссарии концептов.

Таблица 1. Некоторые правила интерпретации для модели функциональных требований

№	Правило	Примечание	Тип элемента	Пример
1	n0 → thing type	n0.corelex → thing type.classification	Thing type	Order
2	n0(desc=compound), n0.child$_j$ → thing type	Составное существительное	Thing type	Order processing services
3	v0.verbclass={locV, possV, copV}→ property of thing type	Свойство типа-сущности	Thing type	Accessibility is highest possible
4	n3, n2.child=p2, p0=’of” →perspective “attribute”	2 типа-связи	Connection type	Default qualities of interest
5	n0(desc=compound) → perspective “containment”	2 типа-связи	Connection type	Order item
6	v0.verbclass≠{aux, eV, iv}→ connection type	По умолчанию	Connection type	
7	v0.verbclass=tv3 →operation type, n3$_{subject}$ →executing, n3$_{object}$ →calling, n3$_{goal}$ | p2 | sentence → parameter	Битранзитивный глагол	Operation type	Order processing services should allow the stock clerk to replenish the items in stock
8	v0.verbclass=tvag2 →operation type, n3$_{subject}$ →executing, n3$_{object}$ →parameters	Глагол с субъектом-агентом	Operation time	The order department for each ordered item checks its availability on stock
9	v0.verbclass=tv2 → condition, n3 → involvedTypes	Пред./пост. условие для типа-кооперации	Cooperation type	Payment is authorized
10	con0 n3 v0 [n3 | p2 | sentence], adv2 n3 v0 [n3 | p2 | sentence]	Конструкция «если, то»	Cooperation type	If each ordered item is on stock, then order department relates that item to the order

Таблица 2. Некоторые правила интерпретации для модели нефункциональных требований

№	Правило	Примечание	Тип элемента	Пример
1	n3$_{subject}$ v0.verbclass=copV n3$_{object}$, n3$_{object}$.child=q2 → n3$_{subject}$ -QualityMetric, QualityInContext.value=q2	Значение показателя	Quality Metric	The response time to the stock items replenishment is below 2 seconds.
2	n3$_{subject}$ v0.verbclass=copV a2 → n3$_{subject}$ -QualityMetric, QualityInContext.description=a2	Характеристика показателя	Quality Metric	Accessibility is highest possible

ПРОЦЕДУРА КОНТРОЛЯ КАЧЕСТВА

Процесс управления качеством формирования модели требований состоит из нескольких этапов. На первоначальном этапе до NLP-обработки осуществляется анализ релевантности документа/текста требований. Чтобы предотвратить использование текстов непригодных для онтологии доменных потребностей, тексты требований проходят процедуру фильтрации, при которой осуществляется поиск ключевых фраз и слов, отвечающей данной предметной области (ПрО). Используя разные стратегии фильтрации, на первом шаге в тексте идентифицируются ключевые слова, которые считаются важными для данной ПрО. После этого предложения содержащие данные ключевые элементы проходят дальше в обработку. Обязательным условием является тот факт, чтобы данные предложения представляли собой связанный смысловой блок. Более детально данный процесс описан в работе [19].

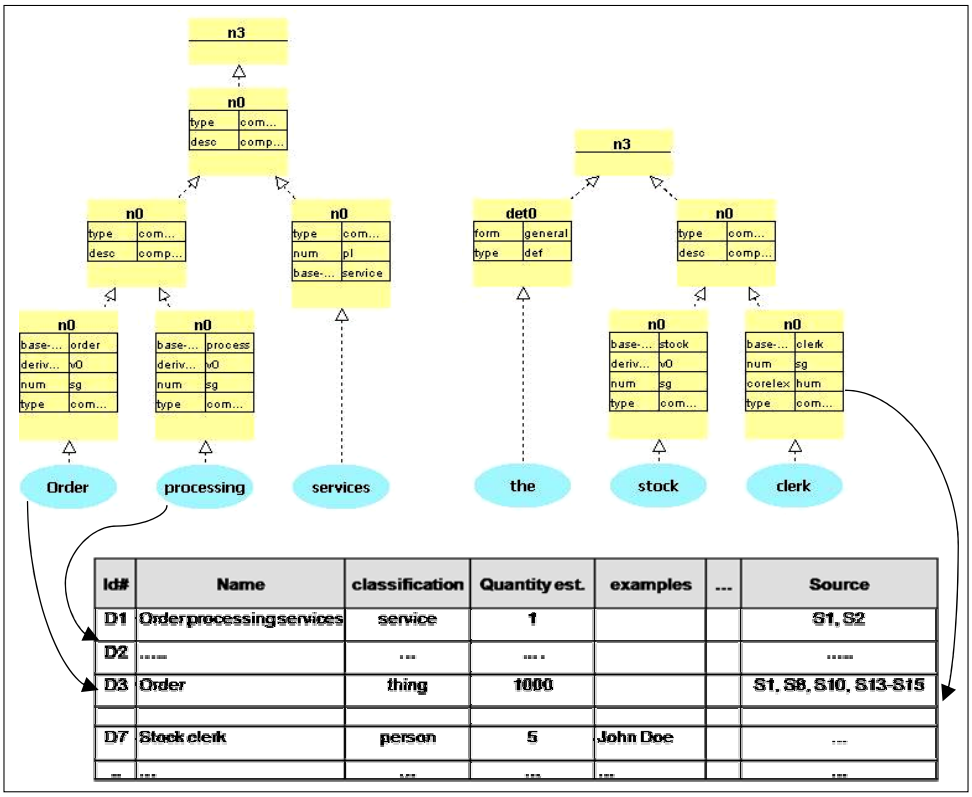

Рис. 7. Заполнение глоссариев модели требований

Второй этап контроля качества предусмотрен перед этапом конвертации древовидных структур текста в концепты модели требований с помощью правил интерпретации. Семантическая информация в предложении несет информацию об его предикатно-аргументной структуре. Если глагол идентифицирован как битранзитивный, т.е. «tv3», то у него должны быть субъект-агент, объект-тема и объект-цель или объект-причина. Если какой-либо из атрибутов отсутствует, то либо формирование фразовых структур предложения произведено неверно, либо глаголу присвоена ошибочная метка транзитивности.

Заинтересованные лица могут иметь различные мнения по одним и тем же вопросам, допускать неточности при ответах на вопросы во время сбора требований и т.д. Также требования могут быть ошибочно интерпретированы во время процедур сбора требований. Для того, чтобы минимизировать возможные рассогласованности и неточности проводится валидация модели ЗЛ: модель (либо модель в адаптированной форме, если полная модель слишком сложна для восприятия) демонстрируют всем ЗЛ и определяют их степень удовлетворенности. В случае, если все ЗЛ удовлетворены моделью, ее можно передавать для использования в дальнейших этапах. Если какие-то из ЗЛ остались неудовлетворенными, проблемный участок анализируется и в зависимости от допусков по неточностям, временных и финансовых ограничений может быть принято решение о частичном или полном пересмотре модели. В случае есть противоречивые требования, полученные от разных ЗЛ, можно либо осуществлять процедуру координации конфликтных позиций до достижения некоторого равновесного состояния, либо организовать непосредственное общение между ЗЛ, при котором они сами должны будут прийти к компромиссу. Примером применения подобного подхода может служить [20].

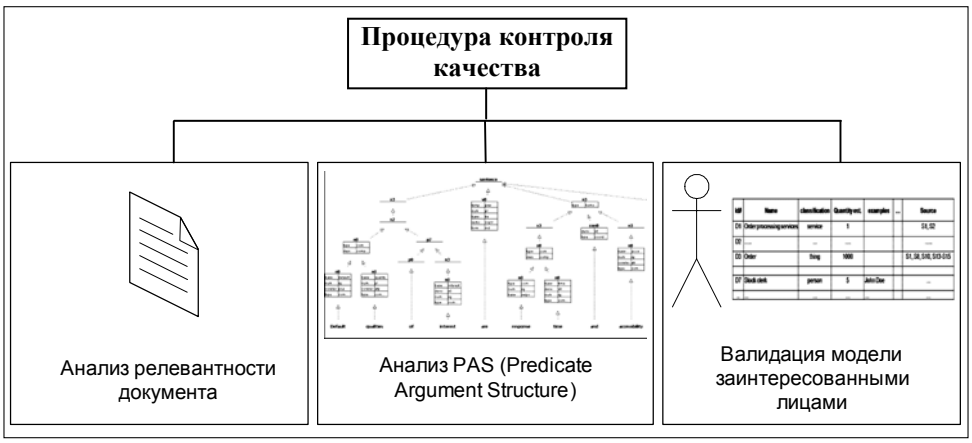

Рис. 8. Процедура контроля качества

ВЫВОДЫ

В работе рассмотрен процесс формирования модели требований к ПО. Рассмотренная двухкомпонентная модель требований является переходным звеном между процессом сбора, процессом анализа и обработки требований к ПО. Поэтому обеспечение качества формирования этой модели позволяет сделать весомый вклад в интегральный показатель качества всего процесса разработки. Поскольку табличное представление данной модели, прежде всего, направлено на активное вовлечение ЗЛ и/или будущих пользователей системы, то осуществляется коррекция требований к разрабатываемому ПО «на лету».

Из недостатков следует отметить языковую зависимость блока автоматизированной обработки текста требований. На данном этапе он адаптирован к текстам на английском языке. Для обработки требований на других

языках требуется подключение соответствующих лингвистических ресурсов. В дальнейшем планируется внедрение более гибких методов и техник, использующих собственную внутреннюю семантико-синтаксическую структуру. Таким образом, адаптировав блок интерпретации под такую структуру, можно будет обрабатывать тексты требований на различных языках одной ПрО, имея при этом одну общую базу концептов моделирования.

ЛИТЕРАТУРА

1. *Андон Ф.И., Коваль Г.И., Коротун Т.М., Лаврищева Е.М., Суслов В.Ю.* Основы инженерии качества программных систем. — К.: Академпериодика. — 2007. — 680 с.
2. *ISO/IEC* 12207:2008 System and software engineering — Software life cycle processes. — International Organization for Standartization, 2008. —123 p.
3. *Баженов Н.А.* Модели управления качеством сбора требований к программному обеспечению на основе текстов спецификаций // Вестн. НТУ «ХПИ». — 2010. — № 9. — С. 35–42.
4. *Соколов Б.Н.* Модели управления качеством обработки требований к программному обеспечению // Вестн. НТУ «ХПИ». — 2010. — № 9. — С. 43–50.
5. *Bazhenov N.A.* Combining probabilistic tagging with rule-based multilevel chunking for requirements elicitation // Искусственный интеллект. — 2010. — № 2. — С. 6–14.
6. *Соколов Б.Н.* Подходы к учету требований качества при концептуальном проектировании и реализации программного обеспечения // Вестн. НТУ «ХПИ». — 2008. — № 5. — С. 133–138.
7. *ISO/IEC* 9126-1:2001 Software engineering — Product quality. — Part 1: Quality model. — International Organization for Standardization, 2001. — 25 p.
8. *ISO/IEC* 25030:2007 Software engineering — Software product Quality Requirements and Evaluation (SQuaRE). — Quality requirements. — International Organization for Standardization, 2007. — 42 p.
9. *Shekhovtsov V., Kaschek R.., Kop C., Mayr H.C.* Relational service quality modeling // Developing effective service oriented architectures: concepts and applications in service level agreements, quality of service and reliability. — NY: IGI Global. — 2010. — P. 172–193.
10. *Mayr H.C., Kop C.* Conceptual predesign — Bridging the gap between requirements and conceptual design // Proc. ICRE. — 1998. — P. 90–100.
11. *Kop C., Mayr H.C.* Mapping functional requirements: from natural language to conceptual schemata // Proc. SEA. — 2002. — P. 82–87.
12. *Shekhovtsov V., Kop C., Mayr H.C.* Towards quality-aware predesign model // Вісн. НТУ «ХПІ». — 2008. — № 5. — P. 17–31.
13. *Shekhovtsov V., Kop C., Mayr H.C.* Capturing the semantics of quality requirements into an intermediate predesign model // Proc. SIGSAND-EUROPE Symposium, Lecture Notes in Informatics (LNI) P-129. — Bonn: GI-Edition. — 2008. — P. 25–37.
14. *Vöhringer J., Gälle D., Fliedl G., Kop C., Bazhenov M.* Using linguistic knowledge for fine-tuning ontologies in the context of requirements engineering // International Journal of Computational Linguistics and Applications. — Bahri Publications. — 2010. — **1**. — P. 249–267.
15. *Fliedl G., Kop C., Mayr H.C., Winkler C., Weber G., Salbrechter A.* Semantic tagging and chunk-parsing in dynamic modeling // Proceedings of the 9-th Interna-

tional conference on applications of natural language processing and nformation systems (NLDB'2004). — Salford UK, Springer LNCS 3316. — 2004. — P. 421–426.

16. *Fliedl G., Kop C., Mayr H.C., Hölbling M., Horn T., Weber G., Winkler C.* Extended tagging and interpretation tools for mapping requirements texts to conceptual (predesign) models // Proc. Of 10-th Int. Conf. on Applications of Natural Language to Information Systems (NLDB'2005). Lecture notes in computer science. — Springer, Heidelberg. — 2005. — **3515**. — P. 173–180.
17. *Fliedl G., Kop C., Mayerthaler W., Mayr H.C., Winkler C.* Guidelines for NL-Based requirements specifications in NIBA // Proc. 5-th Int. Conf. on Applications of Natural Language to Information Systems (NLDB'2000). Lecture otes in computer science (LNCS'1959). — Springer Verlag. — 2000. — P. 251–264.
18. *Perkonigg M.* Linguistische aspekte des attempto controlled english (ACE). Masterarbeit. — Klagenfurt: Alpen-Adria-Universität Klagenfurt. — 2009. — 215 p.
19. *Turban B., Kucera M., Tsakpinis A., Wolff C.* Bridging the requirements to design traceability gap // Intelligent Technical Systems. — Springer Netherlands. — 2009. — **38**. — P. 275–288.

Поступила 07.06.2010