Онтологический подход к поддержке процесса инженерии требований в SCRUM

М.Ш. Муртазина[1], Т.В. Авдеенко[1]

[1]Новосибирский государственный технический университет, проспект К. Маркса 20, Новосибирск, Россия, 630073

Аннотация. В статье представлен подход к организации поддержки процесса инженерии требований к программному обеспечению на основе онтологической модели. Анализируются особенности инженерии требований при управлении проектами разработки программного обеспечения в SCRUM. Исследуются критерии оценки качества пользовательских историй. Онтология разрабатывается в среде Protégé.

1. Введение

Согласно стандарту ISO/IEC/IEEE 29148-2011 «Системная и программная инженерия. Процессы жизненного цикла программных средств. Инженерия требований», инженерия требований (Requirements Engineering, RE) – это междисциплинарный подход, который связывает стороны поставщика и заказчика с целью определения и поддержания требований, которым должна соответствовать целевая система, продукт или услуга. При разработке программных продуктов в рамках процесса инженерии требований осуществляется построение сложной системы требований на базе потребностей, ожиданий, ограничений и взаимодействий стейкхолдеров. Под стейкхолдерами в стандарте ISO/IEC/IEEE 29148-2011 в первую очередь понимаются пользователи и заказчики систем. Так же в круг стейкхолдеров включаются компании, вовлеченные в управление, настройку и сопровождение систем, и регулирующие органы, которые могут быть источником законодательных, отраслевых или других внешних требований, нуждающихся в тщательном анализе.

Инженерия требований играет ключевую роль в обеспечении успеха жизненного цикла разработки программного обеспечения. Выявление требований и управление ими – это чрезвычайно сложная задача для всех методологий управления проектами разработки программного обеспечения, которые обычно подразделяют на две группы: жёсткие (каскадные) и гибкие. Первым присуще детальное планирование. Инженерия требований здесь – это отдельная стадия работ, предшествующая всем остальным работам. В случае гибких методологий планирование выполняется только для текущей итерации. Инженерия требований здесь – это итеративный процесс, при котором требования постоянно эволюционируют. Изменчивость требований является серьезной проблемой для проектов разработки программного обеспечения. В этой связи жёсткие методологии значительно проигрывают гибким. При гибком подходе драгоценное время не тратиться на попытки предусмотреть все возможные требования и детально их задокументировать. Обычной формой фиксации требований высокого уровня в гибких методологиях являются запросы на реализацию функциональности и пользовательские истории, которые формулируются как одно или

несколько предложений, описывающих пожелания к системе. Детали выясняются при реализации требований в рамках очередной итерации в процессе активного взаимодействия со стейкхолдерами. В виду этого спецификациям требования в гибких методологиях присуща неполнота, которая компенсируется обширной неформальной связью со стейкхолдерами. Другой проблемной областью является обеспечение согласованности требований, поступающих от разных стейкхолдеров, а также обнаружение одних и тех же требований, предъявленных разными стейкхолдеров с применения различной терминологии.

На сегодняшний день SCRUM – одна из самых популярных и хорошо проработанных гибких методологий управления проектами разработки программных продуктов. Только в 2016 году, по данным интернет-опроса Agile Survey, данной методологией пользовались 58 % опрошенных специалистов [1]. Однако, несмотря на огромную популярность гибких методологий в целом, и методологии SCRUM в частности, подходы к определению качества спецификаций требований, состоящих из пользовательских историй и запросов на реализацию функциональности, разработаны все еще недостаточно. Многие существующие подходы используют модель INVEST, предложенную еще в 2003 году B. Wake [2]. В середине 2010-х начали появляться новые подходы к оценке качества спецификаций требований для гибких методологий. Среди них Quality User Story Framework [3] и Agile Requirements Quality Framework [4]. Трендом последних лет в области разработки систем поддержки процесса инженерии требований и оценки качества спецификации требований стали онтолого-ориентированные подходы. С точки зрения ученых, такие подходы являются многообещающими для преодоления некоторых недостатков практик инженерии требований [5]. Однако, существующие решения в большей степени ориентированы на оценку спецификаций требований в соответствии с международными стандартами, и не учитывают особенностей для гибких методологий.

В настоящей работе предлагается подход к построению онтологической модели для обеспечения интеллектуальной поддержки процесса инженерии требований при организации разработки программного продукта в рамках SCRUM-подхода. Объектом исследования является процесс поддержки принятия решений в области инженерии требований, предметом – методы и модели оценки качества требований, применяемые в гибких методологиях разработки программного обеспечения.

Статья организована следующим образом. В разделе 1 обосновывается актуальность темы исследования. В разделе 2 представлен обзор публикаций по теме исследования. В разделе 3 анализируются особенности инженерии требований в SCRUM. В разделе 4 рассматриваются вопросы оценки качества пользовательских историй. В разделе 5 предлагается подход к организации онтологической модели поддержки процесса инженерии требований в SCRUM. В разделе 6 делаются выводы о перспективах применения предложенного подхода.

2. Обзор публикаций по теме исследования

В начале 2000-х гг. K.K. Breitman и соавт. рассмотрели процесс разработки онтологий как подпроцесс процесса инженерии требований [6].

В 2005 г. Zhu X. было предложено использовать онтологию домена предметной области как инфраструктуру для уточнения требований к программному обеспечению. Автором предлагалось измерять несоответствие требований на основе древовидной структуры требований, и искать противоречивые отношения между конечными узлами на семантическом уровне [7].

В 2006 г. G. Dobson и P. Sawyer указали, что онтологии полезны для представления и взаимосвязи многих типов знаний. Природа требований предполагает привлечения знаний из многих источников, поэтому существует множество потенциальных применений онтологий в инженерии требований, включая:

- построение модели требований, обеспечивающей реализацию определенной парадигмы структурирования требований;
- формирование структуры сбора знаний о предметной области;

- предметная область;
- окружающая среда [8].

Одной из первых работ, в полной мере демонстрирующих возможности онтологий для инженерии требований, стала работа Kossmann M. и соавт. по созданию методология OntoREM (Ontology-driven Requirements Engineering Methodology) [9]. OntoREM – это полуавтоматическая методология, включающая процессы, методы и инструменты, разработанная для создания спецификаций требований к системам за меньшее время и при меньших затратах при одновременном повышении качества таких спецификаций. Данная методология была апробирована ее авторами в компании Airbus. Применение OntoREM привело к значимой экономии денежных средств и времени при разработке требований к эксплуатационной пригодности воздушных судов со значительным увеличением повторного использования требований. Kossmann M. и соавт. определили OntoREM-процесс, структура которого приведена на рисунке 1.

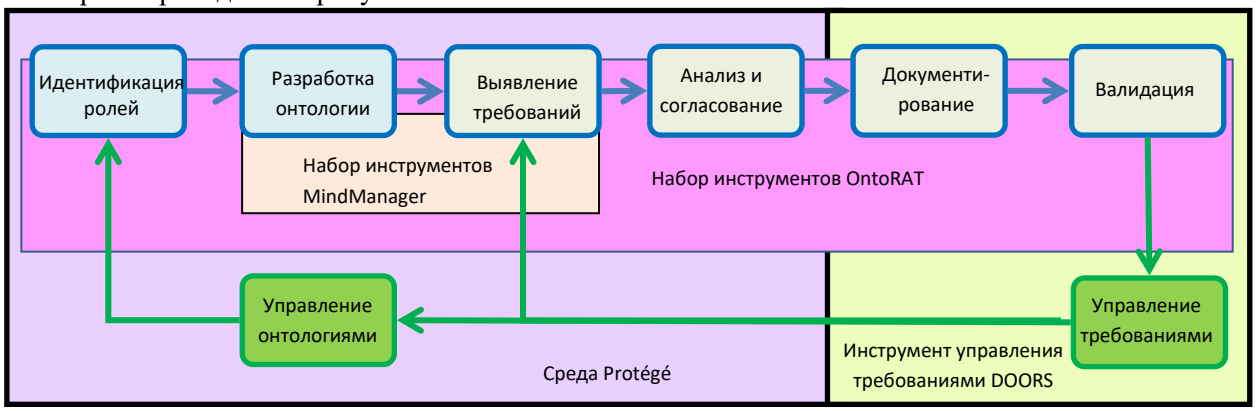

Рисунок 1. Структура OntoREM-процесса [10].

Для обеспечения OntoREM-процесса Kossmann M. и соавт. использовали следующие инструменты[10]:

- набор инструментов MindManager, который применяется для быстрой визуализации онтологии предметной области в виде майндкарты, что удобно при разработке и выявлении требований;
- набор инструментов OntoRAT (Ontology-driven Requirements Analysis Tool), который разрабатываемый в рамках исследовательского проекта OntoREM для анализа требований по статусу, цели, мягким целям (т.е. целям без четких критериев), а также трассируемости в рамках OntoREM-процесса;
- среда Protégé для управления онтологиями, полученными при помощи MindManager;
- программный пакет IBM Rational DOORS для управления требованиями.

Благодаря данному проекту, возможности применения онтологического подхода стали активно исследоваться в различных предметных областях, в том числе и в области разработки программного обеспечения.

В работе [5] K.Siegemund представила подход к автоматизации процесса валидации и измерения знаний о требованиях. В ее работе было выделено два общих представления по поддержке работ инженера по требованиям: (1) поддержка спецификации знаний о требованиях и (2) поддержка валидации и устранения ошибок. Разработанная K.Siegemund система, использует две онтологии, «Guidance Ontology» и «Requirements Ontology», для поиска противоречивых и неполных требований.

В работах [11; 12; 13] Пустовалова Н.В. и Авдеенко Т.В. предложили подход, основанный на фреймовой онтологии, позволяющий сформировать согласованную модель типов требований для конкретного проекта разработки программного продукта. Авторами спроектированы шаблоны структуры спецификации, призвано помочь аналитику учесть все аспекты описания требований. Для построения модели авторами была применена теории полевой структуры

частей речи и рекомендации руководства SWEBOK. Фреймовая онтология понятий инженерии требований была реализованная в среде Protégé.

Вопросы онтологического инжиниринга в гибком процессе разработки программного обеспечения были затронуты в Кошкидько А.В. [14]. Автором было отменено, что благодаря итеративности, свойственной гибким методология, преодолевается проблема получения «грубых» результатов (с большим количеством неточностей) ввиду однократного решения всех подзадач задачи онтологического инжиниринга с имеющимися знаниями о требованиях к продукту на этапе инициализации проекта. При гибком подходе на выходе из каждой итерации будет получена более полная и точная онтология программного продукта.

Рассмотренные выше онтологические модели в области инженерии требований ориентированы в большей степени на каскадный подход к управлению проектом, а проведенный анализ научных публикаций, показывает, что вопросы построения онтологической модели процесса инженерии требований в SCRUM практически не затронуты.

3. Инженерия требований в SCRUM

Согласно Руководству, SCRUM – это процессный фреймворк (набор базовых элементов и правил), предназначенный для разработки и поддержки сложных продуктов. Ядром SCRUM является *Спринт* – временной отрезок максимальной длительностью один месяц, в течение которого команда создает функционирующий и готовый к использованию и выпуску *инкремент продукта* (продукт с новой функциональностью). Общая схема организации работ по фреймворку SCRUM представлена на рисунке 2.

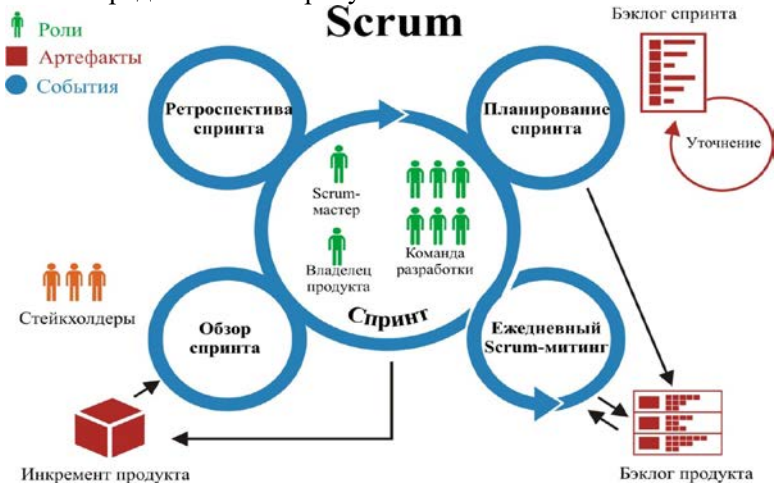

Рисунок 2. Основные элементы Scrum.

Инженерия требований в SCRUM является итеративным процессом, а требования эволюционируют в каждом спринте [15, с.27]. Выявление требований-потребностей происходит во время обзора спринта. Следующий этап – это анализ требований, принятых к исполнению в спринте. Требования, с точки зрения членов команды, должны быть как минимум недвусмысленными, полными и согласованными. Конфликты могут разрешаться путем определения приоритетов пользователя или переопределения неверно сформулированных требований.

Документирование требований помогает анализировать и проверять требования. Хотя моделирование и документация в гибких методологиях разработки программного обеспечения используются в меньших объемах, чем при каскадных, документирование требований является неотъемлемой частью работ.

Обычной формой фиксации требований в гибких методологиях являются:

- запрос на реализацию функциональности (Feature request), который представляет собой структурированный запрос (с заголовком, описанием и рядом атрибутов) для новых или расширенных функциональных возможностей системы;

- история (Story), представляющая собой высокоуровневое определение требования, сформулированное как одно или более предложений на повседневном или деловом языке пользователя так, чтобы разработчики могли дать разумную оценку усилий по ее реализации.

По форме записи выделяют следующие виды историй:

- пользовательская история (User Story);
- техническая история (Technical Story / Technical User Story);
- истории задачи (Job Story).

Самой популярной формой записи требований в SCRUM являются пользовательские истории. Они зарекомендовали себя как хороший способ фиксации требований к функциональности продукта, ценной с точки зрения пользователю программного обеспечения. Пользовательские истории в отличие от традиционных требований – это не результат соглашения о разработке определенной функциональности, достигнутое в результате работы бизнес-аналитика или менеджера с заказчиком, это результат обсуждения команды разработки и бизнес-пользователей актуальной для бизнеса функциональности программного продукт, т.е. несущей некую ценность для бизнеса. Для пользовательской истории характерно:

- создание письменного описания истории, которое используется для планирования работ;
- обсуждение истории для выявления ее деталей;
- определение критериев принятия истории (Definition of Done, DoD).

В SCRUM детальный анализ пользовательской истории происходит в момент реализации функциональности, к которой она относится. При этом командой определяются критерии принятия истории, которые могут быть описаны в форме привычных для каскадных методологий функциональных и нефункциональных требований. В качестве документации требований могут быть рассмотрены варианты использования, описывающие взаимодействие между пользователями и системой. Хорошим способом документирования требований также являются тест-кейсы. Таким образом, документирование требований может вестись по следующей схеме: Пользовательская история – Варианты использования – Тест-кейсы. Пользовательской истории обычно соответствует одна функциональность, этой функциональности – несколько вариантов использования, а каждому варианту использования – несколько тест-кейсов (хотя бы один позитивный и негативные).

Валидация требований происходит во время обзора спринта. Управление требованиями основано на обратной связи с клиентом во время обзора спринта. Следует отметить, что внимание сфокусировано на функциональных требованиях, поэтому многие нефункциональные требования могут быть не учтены.

4. Критерии оценки качества пользовательских историй

Существует множество подходов к оценке качества требований. Стандарт ISO/IEC/IEEE 29148:2011 *Системная и программная инженерия. Процессы жизненного цикла. Разработка требований*. Определяет критерии качеств отдельных требований и наборов требований. К характеристикам качества отдельных требований относятся:

- необходимость (Necessary): требование должно отражать существенную функцию или качество продукта, которые не могут быть перекрыты другими требованиями;
- свобода реализации (Implementation Free): определяется что требуется, а не как требование должно быть выполнено;
- однозначность (Unambiguous): требование интерпретируется только одним способом;
- консистентность (Consistent): требование не содержит конфликтов с другими требованиями;
- полнота (Complete): требование не нуждается в дальнейшем пояснении, поскольку оно измеримо и достаточно подробно описывает функциональные возможности и характеристики для удовлетворения потребности стейкхолдера;
- единичность (Singular.): формулировка требования включает только одно требование без использования союзов;

- реализуемость (Feasible): требование технически достижимо, не требует значительных технологических достижений и вписывается в системными ограничениями (например, стоимостью, графиком работ, нормативными ограничениями) с приемлемым риском;
- трассируемость (Traceable): требование трассируемо вверх (прослеживаются источники требования врехнего уровня) и вниз (ко всем порожденным им требования в спецификации);
- верифицируемость (Verifiable): требование можно проверить, чтобы доказать, что система удовлетворяет указанному требованию.

К характеристикам качества набора требований относятся:

- полнота (Complete): набор требований не нуждается в дальнейшем уточнении, поскольку содержит все, что связано с определением системы или системного элемента.
- консистентность (Consistent): набор требований не имеет отдельных требований, которые противоречат друг другу. Требования не дублируются. Один и тот же термин используется для одного и того же элемента во всех требованиях;
- осуществимость (Affordable): полный набор требований может быть удовлетворен решением, которое можно получить с учетом ограничений жизненного цикла;
- ограниченность (Bounded): набор требований поддерживает идентифицированную область для предполагаемого решения, не выходя за пределы того, что необходимо для удовлетворения потребностей пользователей.

Для оценки качества пользовательских историй, может быть использован фреймворк QUS (Quality User Story), предложенный Lucassen и соавт. [3]. Ввиду того, что пользовательская история – это контролируемый язык (controlled language), структура критериев во фремймворке была основана на понимании качества в категориях O.I. Lindland. Это:

- синтаксическое качество, касающееся текстовой структуры пользовательская история без учета ее значения;
- семантическое качество, касающееся отношений и смысла (частей) текста сюжета пользователя;
- прагматическое качество, которое управляет субъективной интерпретацией текста истории пользователя, кроме синтаксиса и семантики.

Критерии оценки синтаксического качества отдельной пользовательской истории:

- хорошая формализованность (Well-formed): пользовательская история включает в себя, по крайней мере, роль и действие;
- атомарность (Atomic): пользовательская история выражает требование только для одной функции системы;
- минимальность (Minimal): пользовательская история не содержит ничего кроме, роли, действия и цели.

Критерии для оценки семантического качества отдельной пользовательской истории:

- концептуальное звучание (Conceptually sound) : действие выражает функцию, а цель ее логическое обоснование;
- ориентированность на проблему (Problem-oriented): история пользователя определяет проблему, а не решение для нее;
- однозначность (Unambiguous): в истории пользователя избегаются условия или абстракции, которые приводят к нескольким интерпретациям.

Критерии оценки прагматического качества отдельной пользовательской истории:

- полнота словесной формулировки (Full sentence): пользовательская история – это хорошо сформулированная фраза;
- готовность к оценке (Estimatable): в пользовательской истории не указывается крупное требование, которые трудно планировать и приоритизировать.

Для набора пользовательских историй производится оценка семантического качества по критерию бесконфликтности (Conflict-free): история пользователя не должна быть несовместимой с любой другой историей пользователя.

Для набора пользовательских историй производится оценка прагматического качества по критериям:

- уникальность (Unique): каждая пользовательская история уникальна, дубликаты не допускаются;
- унифицируемость (Uniform) : все истории пользователей в спецификации используют один и тот же шаблон;
- независимость (Independent): пользовательская история является автономной и не имеет неотъемлемых зависимостей от других историй;
- полнота (Complete): выполнение набора пользовательских историй создает полнофункциональное приложение, никакие шаги не пропущены (для критических пользовательских историй) .

Для оценки пользовательских историй по перечисленным критерия качества может быть использована онтологическая модель и синтаксический и морфологический анализаторы.

5. Описание разработанной модели

Для поддержки процесса инженерии требований на базе онтологического подхода подготовлена онтология, которая аккумулирует в себе знания об особенностях инженерии требований в SCRUM. На рисунке 3 представлены фрагмент онтологии.

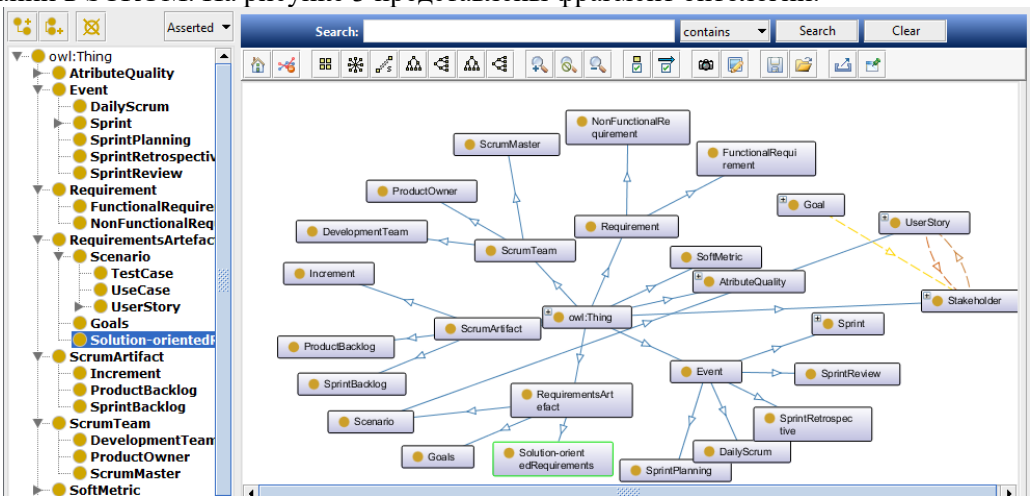

Рисунок 3. Фрагмент онтологии в редакторе Protégé.

Рассмотрим модель концепта «Пользовательская история» (см. рисунок 4). Пользовательская история является частью бэклог продукта и может быть включена в бэклог спринта. Для пользовательской истории, включенной в бэклог спринта, должны быть определены критерии принятия. У пользовательской истории должен быть автор – стейкхолдер и для нее должен быть указан приоритет. У пользовательской истории должна быть определенная структура.

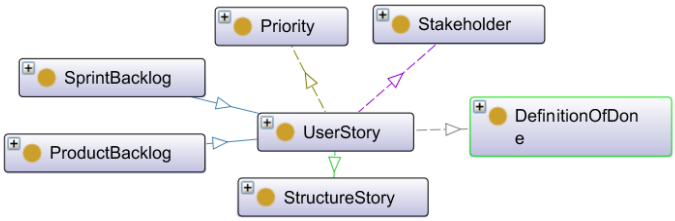

Рисунок 4. Класс «Пользовательская история».

Для пользовательской истории обычно используется следующая схема записи требования:
Как <пользователь>, я хочу <действие> [для того, чтобы <цель>].

Конечная цель задачи может не указываться, если она очевидна. Концептуальная модель структуры пользовательской истории приведена на рисунке 5.

Пользовательская история всегда определяет роль пользователя и действия, которое он хочет совершить в системе. Последнее в простейшем случае грамматически состоит из глагола действия и объекта, на который направлено действие. Например: «Как пользователь, я хочу искать фильмы». Однако действие может быть описано множеством слов. Например, «Как пользователь, я хочу искать фильмы по названию и году». В данных примерах фильм – это основной объект, а название и год – косвенные. Необязательной частью формулировки является цель пользователя, выражающая, какую выгоду он получает от данной истории. Цель может быть связана с другой функциональностью или выражать качественные требования к работе в системе, или и то и другое одновременно. Например, «Как пользователь, я хочу сортировать фотографии, чтобы легко просматривать их». В данном примере из цели следует нефункциональное требование – простота использования.

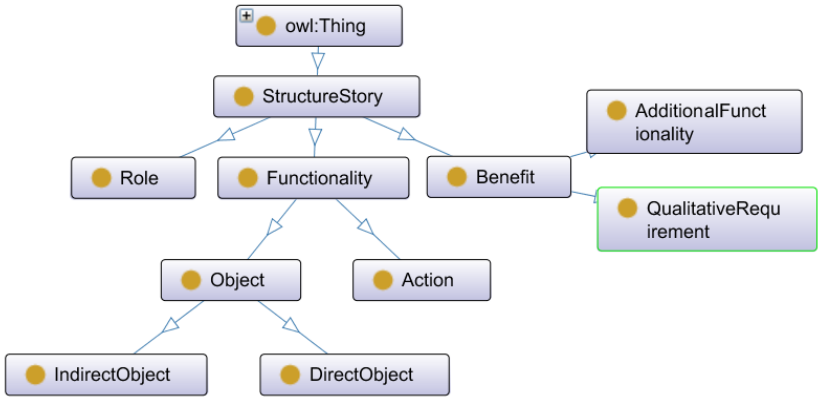

Рисунок 5. Концептуальная модель структуры пользовательской истории.

Исходя из вышесказанного, в онтологической модели можно определить правила оценки качества отдельных пользовательских историй и наборов пользовательских историй. Примеры правил для отдельных пользовательских историй:

Хорошая формализованность:

IF StructureStory has NO (Role or Functionality) THEN print error: "Пользовательская история должна включать в себя роль и действие"

IF StructureStory has NO Benefit THEN print warning: "Для пользовательской история не указана цель"

Атомарность:

IF StructureStory has NO exactly 1 DirectObject THEN print error: "Пользовательская история не атомарна"

Полнота описания отдельной истории:

IF UserStory belongs SprintBacklog and has NO DefinitionOfDone THEN print error: "Для пользовательской истории, включенной в бэклог спринта, должны быть заданы критерии принятия "

IF UserStory has NO Priority THEN print error: "Для пользовательской истории должен быть указан приоритет"

IF UserStory has NO Stakeholder THEN print error: "Для пользовательской истории должен быть указан автор".

На практике определение полноты и конфликтности набора пользовательских историй – это контекстно-зависимая задача, которую трудно обобщить. Тем не менее, можно говорить о неполноте набора пользовательских историй, если в пользовательских историях встречаются требования по манипуляции над элементами системы, которые не были до этого созданы. И о конфликтности набора пользовательских историй, если для одного объекта со стороны одного типа пользователей существует разрешение и запрет на одно и то же действие.

6. Заключение

В работе показано, что применение онтолого-ориентированного подхода к поддержке процесса инженерии требований в SCRUM является крайне актуальной задачей. Требования представляют собой исходные данные для разработки и должны удовлетворять определенным характеристикам качества. В настоящее время основные характеристики качества требований к программным продуктам определены стандартом ISO/IEC/IEEE 29148:2011 *Системная и программная инженерия. Процессы жизненного цикла. Разработка требований*. Проведенный анализ научных публикаций, позволяет говорить о недостаточности характеристик, применяемых при каскадной модели разработки и необходимости применения моделей оценки учитывающих специфику гибких требований.

В данной работе была представлена онтология, учитывающая особенности SCRUM-фреймворка и критериев качества, характерных для пользовательских историй. В ходе анализа особенностей инженерии требований в SCRUM установлено, что основные усилия сосредоточены на анализе функциональной составляющей, поэтому нефункциональные требования зачастую не документируются. С учетом частоты обновления требований при работе по методологии SCRUM применение онтологической модели требований с возможностью поиска по ней неполноты требований и конфликтов будет способствовать повышению производительности труда команды разработки. Введение записей в онтологической базе знаний позволит оперативно отслеживать соответствие вновь разрабатываемых требований уже реализованным и послужит хорошим инструментом документирования хода работ.

7. Благодарности

Работа поддержана грантом Министерства образования и науки РФ в рамках проектной части государственного задания, проект № 2.2327.2017/4.6 «Интеграция моделей представления знаний на основе интеллектуального анализа больших данных для поддержки принятия решений в области программной инженерии».

8. Литература

[1] The 11th annual State of Agile™ report [Electronic resource] // VersionOne Inc. – 2017. – Access mode: https://explore.versionone.com/state-of-agile/versionone-11th-annual-state-of-agile-report-2 (15.11.2017).

[2] Wake, B. INVEST in Good Stories, and SMART Tasks [Electronic resource] / B. Wake // XP123: Exploring Extreme Programming. – August 17, 2003. – Access mode: https://xp123.com/articles/invest-in-good-stories-and-smart-tasks/ (15.11.2017).

[3] Lucassen, G. Forging High-quality user stories: towards a discipline for agile requirements / G. Lucassen, F. Dalpiaz, J.M. van der Werf, S. Brinkkemper // Proceedings of the IEEE international conference on requirements engineering (RE). IEEE. – 2015. – P. 126-135.

[4] Heck, P. A quality framework for agile requirements: a practitioner's perspective [Electronic resource] / P. Heck, A. Zaidman // arXiv preprint arXiv:1406.4692. – 2014. – Access mode: https://arxiv.org/pdf/1406.4692.pdf (15.11.2017).

[5] Siegemund, K. Contributions To Ontology-Driven Requirements Engineering : dissertation to obtain the academic degree Doctoral engineer (Dr.-Ing.) / K. Siegemund. – Dresden: Technischen Universität Dresden, 2014 – 236 p.

[6] Breitman, K.K. Ontology as a Requirements Engineering Product [Electronic resource] / K.K. Breitman, J.C.S. Prado Leite // International Requirements Engineering Conference. – 2003. – P. 309-319. – Access mode: http://eolo.cps.unizar.es/docencia/MasterUPV/Articulos/Ontology%20as%20a%20RE%20product.pdf

[7] Zhu, X. Inconsistency Measurement of Software Requirements Specifications: An Ontology-Based Approach / X. Zhu // Proceedings of the 10th IEEE International Conference on Engineering of Complex Computer Systems. – Washington, 2005. – P. 402-410.

[8] Dobson, G. Revisiting Ontology-Based Requirements Engineering in the age of the Semantic Web [Electronic resource] / G. Dobson, P. Sawyer // Dependable Requirements Engineering of

Computerised Systems at NPPs,Institute for Energy Technology (IFE), Halden. – 2006. – Access mode: http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=1742F7D3AA1345425A8CAEFC2E2E3668?doi=10.1.1.102.5362&rep=rep1&type=pdf.

[9] Kossmann, M. Ontology-driven Requirements Engineering: Building the OntoREM Meta Model / M. Kossmann, R. Wong, M. Odeh, A. Gillies // Information and Communication Technologies: From Theory to Applications, 2008. ICTTA 2008. 3rd International Conference on. – 2008. – P. 1-6.

[10] Kossmann, M. Ontology-driven Requirements Engineering with Reference to the Aerospace Industry / M. Kossmann, M. Odeh, A. Gillies, S. Watts // ICADIWT ’09, IEEE. – London, UK, 2009.

[11] Пустовалова, Н.В. Построение согласованной модели требований для процесса программной инженерии / Н.В. Пустовалова, Т.В. Авдеенко // Труды СПИИРАН. – 2016. – № 1(44). – С. 31-49.

[12] Avdeenko, T.V. The ontology-based approach to support the requirements engineering process / T.V. Avdeenko, N.V. Pustovalova // Actual problems of electronic instrument engineering (APEIE–2016): тр. 13 междунар. науч.-техн. конф., 3–6 окт. 2016 г.: в 12 т. – Новосибирск: Изд-во НГТУ, 2016. – Т.1, ч. 2. – С. 513-518.

[13] Avdeenko, T.V. The ontology-based approach to support the completeness and consistency of the requirements specification / T.V. Avdeenko, N.V. Pustovalova // International Siberian conference on control and communications (SIBCON–2015): proc., Omsk, 21–23 May, 2015. – Omsk: IEEE, 2015. – 4 p.

[14] Кошкидько, А.В. Метод онтологического инжиниринга в гибком процессе разработки программного обеспечения / А.В. Кошкидько // Объектные системы – 2016: материалы XII Международной научно-практической конференции. – Ростов-на-Дону: ШИ(ф) ЮРГПУ (НПИ) им. М.И. Платова, 2016. – С. 14-17.

[15] Darwish, N.D. Requirements Engineering in Scrum Framework / N.D. Darwish, S. Megahed // International Journal of Computer Applications. – 2016. – Vol. 149(8). – P. 24-29.

The Ontology-driven approach to Support the Requirements Engineering Process in Scrum Framework

M.Sh. Murtazina[1], T.V. Avdeenko[1]

[1]Novosibirsk State Technical University, Karl Marx Avenue 20, Novosibirsk, Russia, 630073

Abstract. The paper presents an approach to the Support of the Requirements Engineering Process in the field of software development process by applying ontological techniques. The work analyzes the distinctive features of requirements engineering related to the management of software development projects in SCRUM. The criteria for assessing the quality of user stories are explored. The ontology is implemented in the Protégé environment.

Keywords: software engineering, requirements engineering, ontology, user story, ontology engineering.