Управление требованиями к ответственным системам. Обзор решений

[1] *Н. К. Горелиц <nkgorelits@2100.gosniias.ru>*
[2] *Д. С. Кильдишев <kildishev@ispras.ru>*
[2,3,4,5] *А. В. Хорошилов <khoroshilov@ispras.ru>*

[1]*Государственный научно-исследовательский институт авиационных систем,*
Россия, 125319, г. Москва, ул. Викторенко, 7
[2]*Институт системного программирования им. В.П. Иванникова РАН,*
Россия, 109004, г. Москва, ул. А. Солженицына, д. 25
[3] *Московский физико-технический институт,*
141700, Московская область, г. Долгопрудный, Институтский пер., 9
[4] *Московский государственный университет имени М. В. Ломоносова*
Москва, 119991, ГСП-1, Ленинские горы, д. 1
[5] *НИУ "Высшая школа экономики",*
101000, Россия, г. Москва, ул. Мясницкая, д. 20

Аннотация. Требования являются неотъемлемой частью любого процесса разработки программных и аппаратных систем. Особенно тщательно относятся к требованиям при работе над ответственными системами, использование которых связано с риском для человеческой жизни. Разработка таких систем, как правило, контролируется сертифицирующими органами, требующими применения лучших практик с целью обеспечения безопасности разрабатываемого продукта. В статье рассматривается один из подходов к организации работы с требованиями, который сформировался на основе опыта разработки бортового оборудования гражданских воздушных судов и получил распространение в других отраслях. Приводится набор типовых задач, возникающих при таком подходе. Отталкиваясь от выделенного набора типовых задач формируется методика рассмотрения и оценки инструментов управления требованиями. Предложенная методика применяется для анализа ряда коммерческих и свободно распространяемых инструментов и в заключении формулируются выводы относительно их применения для управления требованиями в проектах по разработке ответственных систем.

Ключевые слова: управление требованиями; ответственные системы; инструменты для управления требованиями; управление изменениями; трассируемость

DOI: 10.15514/ISPRAS-2019-31(1)-2

Для цитирования: Горелиц Н. К., Кильдишев Д. С, Хорошилов А. В. Управление требованиями к ответственным системам. Обзор решений. Труды ИСП РАН, том 31, вып. 1, 2019 г., стр. 25-48. DOI: 10.15514/ISPRAS-2019-31(1)-2

1. Введение

Требования составляют фундамент любого проекта. С работы с требованиями начинается процесс реализации идеи, превращения потребностей и пожеланий в результат - законченный продукт или услугу. Требования и порожденные ими артефакты позволяют сформировать у команды разработчиков единое представление о целях проекта, о разрабатываемом продукте и используемых методах.

По определению международного стандарта по разработке требований ISO/IEC/IEEE 29148 требование - это утверждение, транслирующее или выражающее потребность и связанные с ней ограничения и условия [1].

Требования могут быть представлены в виде текстов на естественном языке, UML диаграмм или текстов на специализированных предметно-ориентированных языках. И даже если требования не зафиксированы документально, они все равно присутствуют в виде представлений о разрабатываемой системе и ее функциях.

Эффект от четкой формулировки требований и последующей аккуратной работы с ними особо заметен при разработке распределенной командой сложных систем с большим количеством связанных элементов. Чем масштабнее проект и больше количество разработчиков, тем сложнее удерживать разработку в рамках, преодолевать разнообразные риски и в итоге достичь удовлетворительного результата.

Связи требований с другими артефактами разработки, в том числе и с производными от требований, помогают осуществлять успешное взаимодействие всех членов и процессов проекта.

Тем не менее часто встречаются проекты со слабо формализованными требованиями, постановки задач с высокой степенью неопределенности. Недостаточная детализация требований может привести к усложнению разработки, нарушению взаимопонимания в коллективе, срыву сроков и реализации других рисков, провоцирующих дополнительные затраты.

В ежегодном выпуске Pulse of profession 2018 от PMI (Project Management Institute) приведены результатов опроса 4,5 тысяч специалистов по управлению проектами на предмет причин, по которым проекты, стартовавшие в организациях за последние 12 месяцев, потерпели неудачи. В 35% случаев респонденты указали в качестве основной причины неудач проектов проблемы с требованиями – с их сбором, анализом, управлением. Проблемы с требованиями занимают в результатах опроса третье место после изменения приоритетов организации и изменения целей проекта. Для сравнения, в Pulse of profession 2017 проблемы с требованиями указаны в качестве основной причины неудач проектов в 39% случаев (второе место), в 2016 – 37% (третье место), в 2015 – 38% (второе место).

Стоимость исправления ошибки, допущенной на этапе разработки требований заметно выше, чем стоимость ошибок на последующих этапах [2]. По данным Счетной палаты США [3][4] изменения в требованиях приводят к росту затрат на проекты более чем в 3 раза, к задержкам в сроках выполнения проектов более чем в 2 раза.

Для эффективной работы с требованиями согласно большинству источников [1][5][6] необходимо, чтобы требования обладали следующими характеристиками.

- Характеристики, касающиеся связи требований с предметной областью.
  - **Адекватность (adequacy)** – соответствие сформулированных требований всем аспектам потребностей и ожиданий пользователей, а также интересам всех остальных заинтересованных лиц.
  - **Выполнимость (feasibility)** – возможность реализовать требование в рамках заданных условий и ограничений.
- Характеристики представления требований самих по себе (внутренние).
  - **Однозначность (unambiguous)** – одинаковость понимания формулировок требований экспертами в соответствующей предметной области.
  - **Внутренняя полнота (completeness)** – охваченность в описании требований всех аспектов и ситуаций, возможных в рамках описанного контекста работы системы.
  - **Непротиворечивость (consistency)** – согласованность описаний требований друг с другом, отсутствие противоречий и расхождений между ними.
  - **Минимальность (minimality)** – невыводимость одних требований из других на основе формальной логики, однократная формулировка каждого нужного ограничения и отсутствие смысловых пересечений между разными требованиями.

- **Простота (singularity)** – отсутствие необходимости подразбиения требования на составные части.
- **Отсутствие деталей реализации (implementation freedom)** – формулировка требований, а не возможных способов их реализации.
- **Систематичность (systematicness)** – представление в виде системы с четко выделенными атрибутами и ясным описанием взаимосвязей и зависимостей между ними.

- Характеристики, касающиеся использования требований в процессе разработки (внешние).
  - **Проверяемость (verifiability)** – возможность для человека, обладающего определенными навыками, однозначно установить в каждой затрагиваемой требованием ситуации, выполнено оно или нарушено.
  - **Прослеживаемость (traceability)** – возможность установления связей между требованиями и их источникам, с одной стороны, а также с разделами и элементами возникающих при разработке текстов программ, документов и моделей, с другой стороны.
  - **Модифицируемость (modifiability)** – возможность удобного и эффективного внесения изменений в сформулированные требования в ходе процесса разработки, включая поддержку различных их версий и конфигураций, а также управление запросами на изменения.

Для того чтобы в результате предпроектной деятельности и мероприятий самого проекта требования получились качественными, управляемыми и позволяющими эффективно пройти все остальные стадии жизненного цикла, предусмотрены разные виды работы с требованиями, соответствующие разным стадиям и процессам проекта. Они включают в себя активности следующих видов:

- **Выделение требований (requirements elicitation)**, состоящее из определения источников требований, извлечения требований и их согласования.
- **Систематизация требований (requirements analysis)** с целью построения целостного набора требований и определения всех существенных взаимоотношений и связей между ними.
- **Описание требований (requirements specification)** в виде документов или моделей.
- **Валидация требований (requirements validation)**, направленная на проверку характеристик, касающихся связи требований с предметной областью (адекватности и выполнимости).
- **Верификация требований (requirements verification)**, нацеленная на проверку внутренних характеристик требований.
- **Управление требованиями (requirements management)**, включающее контролируемое внесение модификаций в различные представления требований и их атрибуты, управление взаимосвязями требований с другими артефактами, предоставление аналитической и иной информации о наборе требований.

В рамках настоящей работы основное внимание уделяется вопросам инструментальной поддержки управления требованиями, а также активностей, связанных с созданием, модификацией и анализом различных представлений требований и их атрибутов в контексте разработки ответственных систем.

Таким образом, из вышеприведенного списка активностей за рамками рассмотрения остаются только вопросы выделения требований.

2. Требования при разработке ответственных систем

Управление требованиями должно осуществляться с учетом целей и задач проекта. Наиболее аккуратное отношение к требованиям традиционно присутствует при разработке ответственных систем, таких как системы управления в гражданской авиации, железнодорожном транспорте, автомобилестроении, судостроении и атомной энергетике.

Основной особенностью ответственных систем является то, что дефекты в конечном продукте могут повлечь за собой риск для жизни и здоровья человека. Также стоит обратить внимание на обычно весьма длительный срок полезного использования для практически всех типов сложных критически важных по безопасности систем, будь то самолет, судно или атомная станция. На протяжении десятков лет система должна выполнять свои функции с высоким уровнем доверия к безопасности и надежности. В связи с этим, эксплуатация таких систем возможна только после прохождения процедуры сертификации. Так, новая модель пассажирского авиалайнера может приступить к коммерческим рейсам только после получения сертификата типа воздушного судна, которое в России на момент написания данной статьи выдается Росавиацией, а, например, в США – Федеральным управлением гражданской авиации (FAA).

Большинство современных регламентов сертификации ответственных систем предъявляют требования не только к конечному продукту, но и к процессам его разработки [7]. Например, в гражданской авиации процесс разработки комплекса бортового оборудования воздушного судна регламентируется требованиями руководства по разработке воздушных судов гражданской авиации и систем Р-4754А [8], гармонизированного с международным аналогом ARP-4754A [9], а процессы разработки программного и аппаратного обеспечения – требованиями КТ-178С/DO-178C [10][11] и КТ-254/DO-254 [12][13], соответственно. Внедрение этих стандартов в гражданской авиации обеспечило достаточно низкий уровень происшествий, случившихся в результате проявления дефектов в бортовом оборудовании, и многие идеи, заложенные в них, в настоящее время распространились по регламентирующим документам в других отраслях.

2.1 Основные задачи разработки ответственных систем

Рассмотрим основные задачи, упоминающиеся в стандартах на разработку ответственных систем, которые относятся к работе с требованиями.

- **Регламенты.** В рамках проекта должны быть сформированы регламенты, в которых зафиксированы правила оформления требований каждого вида.

  Например, согласно КТ-178С регламент на оформление требований к программному обеспечению (ПО) высокого уровня, как правило, называется «стандартом на разработку требований к ПО», а регламент на оформление требований к ПО низкого уровня – «стандартом на проектирование ПО».
- **Оформление.** Все требования к разрабатываемой системе должны быть задокументированы согласно правилам соответствующего регламента.
- **Идентификация.** Каждое элементарное требование должно получить уникальный идентификатор.
- **Источник.** Для каждого требования должен быть указан источник или обоснование его появления (обычно источником являются требования более высокого уровня к системе, а обоснованием – принятые проектные решения).
- **Формальная инспекция.** Должен быть проведен анализ каждого требования на предмет правильности оформления, а также на предмет отсутствия неясностей, неопределенных условий, противоречий и возможности его проверки (верифицируемости). При этом, как

правило, требуется, чтобы анализ проводился группой инспекторов, независимых от авторов требований.

- **Базовая версия.** Для разделов требований, которые были утверждены (т. е. их проанализировали и анализ не выявил проблем), должна быть установлена базовая версия.

  Согласно КТ-178С, базовая версия (Baseline) – это утвержденная зарегистрированная конфигурация одной или нескольких утвержденных единиц конфигурации (например, элементарных требований или каталога требований), используемая в качестве базы для дальнейшей разработки и изменяемая только через процедуры управления изменениями.
- **Трассируемость.** Должны быть установлены и задокументированы связи между требованиями и артефактами, полученными на их основе, такими как исходный код ПО, тесты и прочие.
- **Управление изменениями.** Внесение изменений в требования, для которых была установлена базовая версия, должно выполняться по решению Совета по управлению изменениями, принимаемому по результатам рассмотрения явно сформулированного сообщения о проблеме, а вслед за внесением изменения должны быть проведены процедуры повторного анализа (формальной инспекции) модифицированных требований призванные определить последствия влияния внесенных изменений на существующий набор объектов.
- **Анализ последствий изменения.** Также по результатам изменения в требовании должна быть проведена оценка влияния этого изменения на источник требования, а также оценка его влияния на артефакты, полученные на его основе. Результатом оценки является указание изменений, которые необходимо внести в другие артефакты, а также список других действий, которые необходимо выполнить для приведения всего набора артефактов в согласованное состояние.
- **Данные трассировки.** Должны быть подготовлены данные трассировки (отчетные материалы), демонстрирующие двустороннюю связь между требованиями разных уровней, между требованиями и всеми артефактами, созданными на базе этих требований. Данные трассировки предназначены для:
  - обеспечения верификации полноты трансформации требований в требования более низкого уровня и другие артефакты;
  - наглядной демонстрации требований, которые не трассируются на требования более высокого уровня;
  - обеспечения верификации того, что нигде в исходном коде или оборудовании не реализованы недокументированные функции.
- **История изменений.** История изменений требований должна сохраняться как минимум на уровне базовых версий и быть доступна для анализа не только в ходе разработки системы, но и в течение всего срока ее эксплуатации [14].

Решить перечисленные задачи потенциально возможно и без применения специализированных систем, например, при помощи использования офисных пакетов [14][15]. Также на предприятиях часто можно столкнуться с непрофильным использованием имеющегося в активе программного обеспечения – по разным причинам (политическим, финансовым и др.) не приобретаются программные продукты, созданные для достижения преследуемых целей, зато создаются сложные «конструкторы» из ранее закупленных предприятием продуктов, изначально не предназначенных для автоматизации возлагаемых на них функций в явном виде. Подобные сложные комбинации инструменты обычно весьма тяжелы в поддержке и интеграции как внутри комбинации, так и при добавлении дополнительных продуктов в связку.

Специализированные системы управления требованиями обладают потенциалом для существенного снижения затрат на разработку и снижения риска возникновения ошибок.

2.2 Функциональные возможности систем управления требованиями

Рассмотрим, какие функциональные возможности систем управления требованиями могут потребоваться при разработке требований к ответственным системам.

I. Структурирование и хранение требований.

I.1. Элементы каталога требований и их свойства.

Как минимум, системы управления требованиями должны поддерживать хранение элементарных требований, но кроме того, при написании требований возникает необходимость в наличии других вспомогательных объектов, таких как:

- определения терминов, использующихся в требованиях;
- примечания, содержащие дополнительные комментарии;
- обоснования, поясняющие причины появления производных требований.

Второй вопрос – это номенклатура атрибутов у элемента каталога. Является ли она фиксированной? Может ли она быть дополнена пользователем? Поддерживается ли типизация атрибутов и ограничения на возможные значения атрибутов?

И наконец, основной вопрос относительно формирования содержательной части требований – какой формат используется для представления текста и других типов информации? Поддерживается ли в нем сложное форматирование, таблицы, изображения, диаграммы, математические формулы?

I.2. Идентификация элементарных объектов.

Для возможности ссылаться на отдельные объекты каталога требований, каждый из них должен иметь уникальный идентификатор. Так как с идентификатором приходится работать не только инструментам, но и людям (например, в ходе проведения формальных инспекций и обсуждения их результатов), то желательно, чтобы хотя бы один из способов идентификации был удобен для восприятия человеком.

Относительно способов формирования идентификаторов существует несколько часто встречающихся решений.

Первый из них – это автоматически генерируемые число-буквенные комбинации, такие как, универсальный уникальный идентификатор (universally unique identifier, UUID). Подобные идентификаторы удовлетворяют требованиям уникальности и используются на практике инструментами для внутренних целей. Но для работы с пользователем подобные идентификаторы не подходят, так как работа с ними затруднена – размер идентификатора сравнительно велик, его сложно запомнить, а значение не дает информации о соответствующем объекте.

Другой подход предполагает сопоставление каждому объекту уникального числа. Если такое число уникально в рамках отдельного одного каталога требований, то для однозначной идентификации в пределах базы требуется использовать составные идентификаторы, включающие идентификатор каталога. Числовые идентификаторы являются более компактными, их можно запомнить и достаточно удобно использовать в человеческом общении, но при этом они не содержат информации об объекте.

Третий вариант предусматривает использование идентификаторов, уникальных в пределах поддерева каталога требований. При таком подходе можно рассчитывать на присвоение объектам более-менее семантически значимых идентификаторов, но для глобальной идентификации потребуется использовать составные идентификаторы, включающие идентификаторы родительских элементов.

Например, «/Требования к ПО/Требования к компоненту MemoryManager/001» может являться составным идентификатором требования с номером 001 в некотором каталоге. При этом в идентификатор входит информация как о местоположении объекта в каталоге, так и о самом объекте. Полученные идентификаторы могут оказаться длиннее UUID, но при аккуратно организованном процессе присвоения идентификаторов они могут оказаться удобнее для восприятия человеком.

Еще один важный аспект поддержки идентификации элементов каталога – это работа с идентификаторами удаленных объектов. Если идентификатор однозначно обозначает определенный объект, то при удалении этого объекта такой идентификатор не должен быть использован повторно, чтобы избежать риска путаницы со ссылками на старый и на новый объекты, в том числе, для случаев, когда эти ссылки хранятся вне системы управления требованиями.

I.3. Структура каталога требований.

Каталог требований, как правило, содержит требования некоторого уровня к определенной целевой системе. Также в каталоге часто находятся вспомогательные элементы: определения, примечания, обоснования. Наиболее распространенный случай организации этих элементов – это поддержка иерархической структуры. На эту структуру можно смотреть как на дерево папок, в которых находятся листовые элементы, или же как на иерархию разделов одного документа. Если иерархия поддерживается, то интересен вопрос, насколько нелистовые элементы каталога (условно папки или разделы) отличаются по свойствам и возможностям от листовых.

I.4. Средства редактирования текста требований.

Одна из ключевых задач разработчика требований – это собственно написание текста требований. Относительно этого основной вопрос к инструментам заключается в том, какие средства для редактирования текста они предоставляют. Наиболее распространенные варианты: поддержка HTML-редакторов в браузере, возможность редактирования требований в традиционных офисных пакетах или во встроенных текстовых редакторах.

Применение интеграции с текстовыми редакторами позволяет использовать опыт широкого круга пользователей по работе в привычной среде для разработки требований. При этом возникает несколько проблем. Первая из них связана с преобразованием требований в офисный формат и обратно, а также с использованием разных версий офисных программ и разных версий установленного у разработчиков требований ПО для управления требованиями, что по причине ограниченной совместимости версий может вызывать коллизии при преобразованиях данных между форматами. Вторая заключается в необходимости поддержки слияния различных версий каталога требований. Подобная ситуация может возникнуть, когда после выгрузки требований в документ в каталог требований были внесены изменения.

Использование браузера в качестве средства просмотра или редактирования требований позволяет обеспечить работу с требованиями на различных устройствах и операционных системах. При этом недостатком является необходимость в поддержке различных представлений требований, если формат описаний и значений свойств требований в инструменте отличается от HTML. Ряд инструментов представляют собой веб платформы и, соответственно, для них HTML редакторы можно считать встроенными.

Встроенные редакторы поддерживаются большинством инструментов и позволяют редактировать требования средствами самого инструментов. Из минусов можно отметить потенциальную необходимость в обучении пользователя работе с инструментом.

II. Поддержка связей.

Под связью понимается отношение элемента каталога требований либо с другим элементом каталога, либо с некоторой внешней сущностью. Связи могут быть различных видов, например, «тест проверяет требование», «требование к компоненту реализует требование к системе» или «требование использует термин, определенный в данном элементе каталога требований». Отдельно стоит выделить такой вид связи как отношение родитель – ребенок.

Принято выделять различные виды связей по количеству связанных узлов – один к одному, один ко многим, многие ко многим. Примером связи один ко многим может послужить отношение родитель – ребенок. Одному родителю соответствует множество детей, одному ребенку – один родитель.

Отдельные связи могут иметь вид многие ко многим – например, требование может уточняться множеством других требований. Требование может уточнять несколько других требований.

В общем случае, связи могут обладать произвольными атрибутами подобно элементам каталога требований.

Задание связей может быть явным, когда указываются идентификаторы обоих элементов отношения, или неявным, когда связь вычисляется, например, на основании значений атрибутов. Второй вариант может быть удобен для таких видов связей как «определение-использование», когда вместо явного указания элемента каталога, определяющего термин, можно перечислить список использованных терминов, а инструмент найдет объект с определением и установит связь автоматически.

Наличие хорошо проработанных связей между данными проекта – один из элементов процесса управления конфигурацией, который должен выполняться на протяжении всего жизненного цикла разработки и тесно связан с активностями остальных процессов жизненного цикла. Наличие связей позволяет проводить различные виды анализа данных и качества процессов и получать информативные результаты о состоянии работы в текущий момент. Это особенно важно в контексте управления проектами [16] – связи обеспечивают поддержание информационной базы для управления проектом в актуальном, полном, целостном состоянии, что в свою очередь помогает руководителю видеть текущую картину разработки и принимать эффективные, своевременные и обоснованные решения на всех этапах проекта.

Таким образом, для каждого класса связей, перечисленного ниже, интересны ответы на следующие вопросы:

- Как задаются и хранятся связи?
- Поддерживаются ли различные виды связей?
- Поддерживаются ли атрибуты связей?
- Какие способы визуализации связей поддерживаются?

II.1 Поддержка связей между элементами каталога требований.

II.2 Поддержка связей между требованиями и их внешним источником.

II.3 Поддержка связей между требованиями и внешними артефактами разработки, например, тестами или исходным кодом.

Если поддержка такого вида связей отсутствует, то на практике применяется обходной маневр, в рамках которого для каждого внешнего артефакта заводится «прокси» элемент каталога требований и используется механизм внутренних связей (п. 2.1). Это позволяет решить задачи организации трассируемости, но при этом приходится решать проблему синхронизации между «прокси» элементами и внешними артефактами.

II.4 Поддержка генерации данных трассировки.

Хотя данные трассировки являются частным случаем визуализации связей, ввиду их важности с точки зрения выполнения требований КТ-178С и других нормативных

документов, авторами статьи было принято решение вынести их в отдельный пункт.
В соответствии с целями использования данных трассировки могут поддерживаться специализированные представления для анализа покрытия некоторого множества сущностей связями определенного вида, например, покрытия требований исходным кодом или тестами.
Существует несколько используемых на практике форм представления данных трассировки. Среди них матрица связанности, список связанности и граф связанности.
Рассмотрим два множества элементов каталога требований N и M. Распределение элементов по множествам зависит от задач анализа. В качестве примера можно привести построение матрицы связанности объектов различных типов, например, требований и тестов (с целью решения задачи анализа покрытия требований тестами). Матрица связанности множеств будет представлять собой матрицу размера |N| на |M|. При этом в ячейке (i, j) будет содержаться информация о наличии связи между i-м элементом N и j-м элементом M. В списке связанности для каждого элемента в N будет приведен список всех связанных с ним элементов из M. Граф связанности будет представлять собой направленный граф. В качестве вершин будут включены все элементы N и M, для которых есть входящая или исходящая связь, а сами связи будут отображаться в виде ребер.

III. История изменений и управление изменениями.

III.1 История изменений каталога требований.

Поддержка регистрации истории изменений каталога требований является необходимым условием применимости инструмента для разработки ответственных систем. Работа с историей изменений основывается на понятии версионирования, рассматривая которое, следует выделить следующие моменты.

III.1.1 Базовый объект версионирования.

В качестве таких объектов могут выступать:

- каталог с требованиями в целом;
- произвольное поддерево каталога;
- отдельный объект.

Версионирования только на уровне объектов недостаточно для решения всех задач, так как по версиям отдельных объектов нельзя получить согласованное состояние всего каталога и для этого потребуется создание отдельного указателя конфигурации. И наоборот, версионирования на уровне всего каталога вполне достаточно, так как по версии каталога и идентификатору объекта можно однозначно восстановить состояние отдельного объекта в нужный момент времени.

III.1.2 Идентификатор версии:

- автоматически генерируемый
  - читабельный,
  - нечитабельный;
- определяемый пользователем.

III.1.3 Поддерживаемый набор атрибутов версии:

- дата и время изменения;
- автор изменения;
- комментарий, связанный с изменением.

III.1.4 Подход к сохранению изменений:

- сохранения выполняются пользователем:
  - сессия сохранения может быть прервана и продолжена после перезапуска

инструмента;
  - сохранение должно быть произведено в пределах сессии работы с инструментом;
  - сохранение должно быть произведено при редактировании отдельного объекта;
- Сохранения выполняются автоматически:
  - сохранение происходит при редактировании отдельного объекта;
  - сохранение происходит по завершению сессии редактирования.

III.1.5 Поддержка операций над версиями:

- визуализация истории изменений;
- визуализация отличий между двумя выбранными версиями;
- возможность восстановления состояния объекта до определенной версии

III.2 Поддержка статуса утвержденности.

Поскольку разработка требований ведется поэтапно, то часть разделов каталога требований может быть уже утверждена, тогда как другая часть может еще находиться в процессе разработки. Инструмент управления требованиями может предоставлять средства для хранения статуса утвержденности и его визуализации.

III.3 Анализ последствий изменений.

При внесении изменений в требования возникает вопрос о том, как эти изменения повлияют на другие требования и артефакты разработки. Ответ на этот вопрос получают в рамках оценки влияния изменения.
Одно из инструментальных решений для поддержки подобного анализа - это поддержка специального статуса у связей, для которых необходимо провести анализ влияния произошедших изменений на другую сторону связи. Этот статус получил название «подозрительных связей» (suspect link). После проведения анализа влияния статус «подозрительности» снимается.

IV. Организация совместной работы.

IV.1 Разрешение конфликтов.

При совместной работе над общим каталогом требований возникает задача организации совместного редактирования требований. Существует два основных подхода к решению этой задачи. Первый их них предполагает блокировку доступа на время редактирования. При этом пользователь может отметить, что он редактирует определенный набор требований (раздел), и этот набор никто другой редактировать не сможет.
Второй подход заключается в апостериорном слиянии изменений в случае конфликта. При этом оба пользователя смогут редактировать некоторый набор требований локально, но при внесении изменений в общий репозиторий возникнет конфликт, который потребуется разрешить одному из этих пользователей. Обычно разрешением конфликта должен заниматься тот из пользователей, кто вносит изменения позднее. Он должен проанализировать оба набора изменений и вручную совместить их корректным образом. Инструменты могут предоставлять средства упрощения этих действий, но стоит отметить, что автоматическое слияние запрещается многими регламентирующими документами (например, КТ-178С) во избежание неконтролируемых изменений.

IV.2 Поддержка ролей и разграничения доступа.

В процесс работы с требованиями вовлечены люди, решающие разные задачи. Можно выделить разработчиков требований, экспертов, проводящих формальные инспекции, и разработчиков иных артефактов, использующих требования в качестве исходных данных. В зависимости от роли могут различаться права доступа

к каталогу требований, объектам и даже свойствам объектов. Инструменты управления требованиями могут поддерживать такие ограничения прав. Также инструменты могут поддерживать настройку специфических форм представления требований в зависимости от роли пользователя.

В дополнение к базовому набору функциональных возможностей мы также рассмотрим ряд дополнительных возможностей:

V. Обмен требованиями с другими инструментами.
В процесс работы с требованиями оказываются вовлечены разные люди и организации, при этом обладающие различным опытом и использующие различные инструменты. Дополнительные сложности возникают при отсутствии возможности обмена данными между инструментами напрямую в связи с ограничениями по безопасности.
Кроме того, от современных инструментов управления требованиями ожидается способность интегрироваться с другими инструментами, автоматизирующими смежные процессы жизненного цикла продукта, включая не только разработку, но и пред- и околопроектные активности, эксплуатацию, техническое обслуживание и вывод из эксплуатации. Если подобная интеграция отсутствует, появляется большой объем работы по связыванию и консолидации данных жизненного цикла из разрозненных инструментов, которую приходится выполнять вручную. В результате формирование целостного набора данных жизненного цикла оказывается затруднено и возникают дополнительные риски – как в вопросах ресурсов проекта (дополнительные трудозатраты и сдвиг сроков), так и в вопросах качества продукта (риски человеческого фактора).
Для обмена требованиями между различными инструментами применяются механизмы импорта-экспорта каталога требований в стандартизированные представления, такие как файлы специального формата, например ReqIF [17], или стандартизованные программные интерфейсы, такие как OSLC Requirements Management Specification [18].
Следует отметить, что стандарт ReqIF на данный момент имеет три стабильные версии, начиная с 1.0.1. В стандарте кроме подхода к описанию элементов каталога и их свойств также описываются дополнительные механизмы, такие как задание связей, добавление изображений и вложений. При этом многие инструменты на практике могут отходить от описанных в стандарте представлений. Отдельные инструменты также поддерживают только обмен иерархией и свойствами артефактов. Стандарт позволяет обмениваться форматированным текстом с возможностью добавления изображений и произвольных вложений за счет использования подмножества объектов языка XHTML 1.0. При этом накладывается запрет на использование свойства тегов class и ограничения на использование стилей в style.
OSLC представляет собой семейство стандартов, описывающих интерфейс веб-сервисов, предназначенных для интеграции различных инструментов поддержки разработки, включая инструменты управления требованиями и управления изменениями. Инструменты при этом могут выступать в качестве поставщиков соответствующих сервисов или их потребителей.
Применительно к системам управления требованиям OSLC Requirements Management Specification описывает интерфейс веб-сервисов, позволяющий запрашивать отдельные требования и их подмножества, создавать новые требования и редактировать свойства уже существующих элементов каталога.

VI. Поддержка шаблонов и повторное использование.
При разработке требований достаточно часто возникает ситуация, когда можно выделить группы схожих требований или схожих разделов каталога требований, отличающихся в небольшом количестве деталей. В таких ситуациях могут оказаться полезны механизмы шаблонизации и повторного использования.
Первый из них предполагает возможность задания шаблонов группы требований и

добавления в каталог требований несколько копий шаблона, раскрытых с разными параметрами. Если поддержка шаблонов существует только в редакторе требований, то при необходимости внесения изменений в шаблон придется вручную отредактировать все созданные на его основе копии требований. Если же шаблон сохраняется в каталоге и его раскрытие выполняется автоматически, то при необходимости внесения изменений в шаблон все его вариации будут обновлены автоматически. Второй подход получил название механизма повторного использования.

VII. Поддержка генерации отчетов.
При управлении требованиями могут возникнуть ситуации, когда необходимо вывести определенное представление требований или некоторую дополнительную статистическую информацию о каталоге. Примером может послужить вывод печатного документа, содержащего все требования в заданном формате или построение отчета о покрытии требований тестами на основе внешних связей.
Для решения этих задач многие инструменты поддерживают генерацию различных представлений каталога требований, а также механизм генерации документов по задаваемым пользователем шаблонам.

VIII. Средства проактивного информирования об изменениях.
В дополнение к средствам сравнения версий требований и анализа последствий изменения некоторые инструменты предоставляют возможность автоматического информирования о появлении изменений в интересующих пользователя разделах каталога требований. В качестве механизмов для уведомления применяются как внешние средства передачи информации (например, электронная почта или мессенджеры), так и средства, встроенные в инструменты работы с требованиями.
Проактивное информирование используется на практике для различных задач. Одно из встречаемых применений – отслеживание комментариев и дискуссий, связанных с каталогом требований. Это могут быть замечания автору элемента, обсуждения каталога или его фрагментов, в том числе в процессе формальной инспекции. Кроме этого встречается отслеживание структуры и содержимого каталога. При этом обычно формируется и распространяется список внесенных за определенный промежуток времени изменений.

2.3 Инструменты для управления требованиями

В рамках настоящего обзора рассматриваются возможности наиболее распространенных коммерческих программных решений, декларирующих поддержку управления требованиями в контексте разработки ответственных систем, а также ряд свободно распространяемых инструментов. В первую группу входят продукты семейства IBM Rational (RequisitePro, DOORS и DOORS Next Generation), а также инструменты ReqView, Jama и Polarion. Во вторую группу включены инструменты RMTOO, aNimble Platform, Eclipse ProR и разрабатываемый в ИСП РАН инструмент Requality. Последние два инструмента развиваются по модели открытого ядра («open core»), в соответствии с которой базовая функциональность распространяется под свободной лицензией, а дополнительные возможности доступны на коммерческой основе. Для Eclipse ProR коммерческий вариант называется ReqIF Studio, а для Requality – это дополнительные плагины, входящие в состав АРМ ПТ, разрабатываемого совместно с ГосНИИАС.

2.3.1 Методика анализа инструментов

Оценка инструментов будет проводиться с точки зрения поддержки функциональных возможностей, необходимых для работы с требованиями при разработке ответственных систем, которые были сформулированы в разделе .II.1. Для проведения оценки используется

следующая методика. Для каждого инструмента изучаются публично доступные актуальные версии документации и учебно-методических материалов и описываются обнаруженные возможности по каждому из пунктов. Для свободно распространяемых инструментов также анализируется их исходный код. На основе собранной информации проводится экспертная оценка полноты поддержки каждой возможности, в результате чего каждому инструменту присваивается числовая оценка по каждой из следующих групп возможностей:

- структурирование и хранение требований;
- поддержка связей;
- поддержка управления изменениями на уровне каталога;
- поддержка управления изменениями на уровне отдельного объекта;
- поддержка статуса утвержденности и оценки влияния последствий изменения;
- поддержка совместной работы;
- обмен требованиями с другими инструментами;
- другие возможности.

Недостатком данного подхода является субъективность экспертных оценок и возможность получения недостоверных оценок ввиду неполноты документации или недостаточной аккуратности ее изучения. Тем не менее, методика позволит получить первичную оценку, которая может быть уточнена в дальнейшем по мере получения дополнительной информации.

2.3.2 Объекты анализа

Для анализа использовались следующие версии инструментов и материалы.

- IBM Rational RequisitePro 7.1.5. По причине окончания поддержки были использованы данные последней версии документации [21], рекламных материалов [22] и встроенной документации.
- IBM Rational DOORS 9.4. Официальная документация [23].
- IBM Rational DOORS Next Generation 6.0.6. Официальная документация [25].
- ReqView 2.4.0. Официальная документация [26].
- Jama Connect версии 8.30. Официальная документация [27].
- Polarion REQUIREMENTS 3.18. Официальная документация [28].
- rmToo 24.0. Официальная документация [29].
- aNimble Platform 0.4. Официальная документация [30].
- Eclipse ProR 0.13. Официальная документация [31].
- Requality 1.0. Официальная документация [32].

2.3.3 Краткое описание результатов анализа

В настоящем разделе представлено краткое описание результатов проведенного анализа. Более подробная информация, включающая детали рассмотрения по каждому инструменту, представлена в [34].

Базовая функциональность по работе с объектами каталога требований, их идентификации и иерархической организации, а также средства редактирования текста присутствуют практически у всех рассмотренных инструментов (табл. 1). Исключением стал RMTOO, у которого отсутствует возможность не допустить переиспользования идентификаторов после удаления объекта. Также RMTOO оказался единственным инструментом, не

поддерживающим добавления к объектам каталога атрибутов, определяемых пользователем, и не поддерживающим отношения родитель-ребенок между объектами каталога. Следует отметить отсутствие поддержки читабельных идентификаторов, не подлежащих повторному использованию у ProR и Requality, хотя для Requality такая возможность доступна при помощи дополнительного плагина.

Относительно возможностей по оформлению текста требований наиболее ограниченным оказался RequisitePro, у которого описание требования может содержать только текст без какого-либо форматирования. В ReqView и ProR отсутствует возможность использовать в тексте требований таблицы. В DOORS существует поддержка таблиц специального вида, у которых каждая ячейка является отдельным требованием. Также в DOORS таблицы можно вставить в текст требования как OLE-объекты. Но в этом случае их можно редактировать только с помощью внешних редакторов в настольном клиенте, а в веб-интерфейсе такие таблицы представлены как картинки. Это не позволяет считать поддержку таблиц в тексте требований в DOORS полноценной. Также следует отметить неожиданное отсутствие в DOORS поддержки выделения фрагментов текста моноширинным шрифтом.

Табл. 1. Структурирование и хранение требований
Table 1. Structuring and storage of requirements

	1.1 Типы	*1.2 ID*	*1.3 Иерархия*	*1.4 Редакторы*	
				ВР	*ТП*
RequisitePro	П	ЧРД	Р	П	+
DOORS	П	ЧОД	РМ	ПРК	
		ИД			
DOORS NG	П	ЧИД	Р	ПРКТ	
ReqView	П	ЧОД	РМ	ПРК	
Jama	П	ЧИД	Р	ПРКТФ	
		ЧРД			
Polarion	П	ЧРД	Р	ПРКТ	
RMTOO	Ф	ЧР	М	ПРКТФ	
aNimble	П	ЧИД	Р	ПРКТ	
ProR	П	РД	Р	ПРК	
Requality	П	ЧР	Р	ПРКТ	+
		РД			

1.1 Типы – элементы каталога требований и их свойства
П/Ф – расширяемый/фиксированный набор типов.
1.2. ID – Идентификация элементарных объектов
Ч – идентификатор читабелен;
И/Р/О – идентификатор уникален в пределах инструмента/проекта/поддерева;
Д – идентификатор удаленного узла не будет повторно использован.
1.3. Иерархия – Структура каталога требований
Р – отношение родитель-ребенок;
М – средства группировки объектов в папки/модули.
1.4. Редакторы – Средства редактирования текста требований.
ВР – возможности встроенного редактора:
П/Р/К/Т/Ф – поддержка обычного текста/форматирования изображений/таблиц/формул;
ТП – интеграция с текстовыми процессорами.

Поддержка связей между объектами каталога требований и генерация данных трассировки присутствует у всех инструментов (табл. 2). При этом практически все инструменты поддерживают задание связей, настраиваемых пользователем. Исключением оказались RequisitePro, RMTOO и aNimble. Поддержка связей с внешними объектами не столь распространена. Возможность, по крайней мере, генерации отчетов о покрытии требований элементами, получающимися на их основе, такими как тесты или исходный код, доступна в RequisitePro, DOORS, DOORS NG, Jama, Polarion и Requality. А возможность установить связь с источником требований во внешних документах в явном виде присутствует только в RequisitePro и Requality.

Табл. 2. Поддержка связей между объектами и с внешними сущностями
Table 2. Support of relationships between objects and with external entities

	2.1. Внутр.	*2.2. Источник*	*2.3. Потребитель*	*2.4 Трассировка*
RequisitePro	1	+	+	1
DOORS	3		+	2
DOORS NG	3		+	2
ReqView	3			1
Jama	3		+	1
Polarion	3		+	1
RMTOO	2			1
aNimble	1			1
ProR	3			1
Requality	3	+	+	1

2.1 Внутр. – поддержка связей между объектами каталога:
1 – только один тип связей;
2 – поддержка множества типов связей;
3 – поддержка типов связей, задаваемых пользователем.
2.2 Источник – поддержка связей между требованиями и их источником во внешних системах.
2.3 Потребитель – поддержка связей с внешними артефактами разработки, разрабатываемыми на основе требований.
2.4 Трассировка – поддержка генерации данных трассировки:
1 – матрицы и/или списки связанности
2 – матрицы и/или списки связанности+графические представления

Управление историей изменений на уровне всего каталога (табл. 3) отсутствует только в ReqView и aNimble, что существенно сокращают возможность их применения для разработки ответственных систем, поскольку для решения задач конфигурационного управления требованиями потребуется привлекать дополнительный инструментарий. ProR в обязательном порядке формирует очередную версию при завершении редактирования отдельного объекта, а RequisitePro – при завершении очередного сеанса работы с инструментом. Оба варианта являются не самым лучшим решением с точки зрения удобства использования. Кроме того, ProR вместе с RMTOO и Requality формируют нечитабельный идентификатор версии. Для ProR ситуация усугубляется отсутствием возможности указать комментарий к версии. Все инструменты поддерживают базовые операции по визуализации истории изменений и сравнению версий. Также все кроме Polarion позволяют восстановить состояние каталога, соответствующее выбранной версии.

Табл. 3. История изменений на уровне каталога
Table 3. History of changes at the directory level

	3.1 ID	3.2 Атрибуты	3.3 Сохранение	3.4 Операции
RequisitePro	АЧ	ДВЗ	П2	ИРВ
DOORS	П	ДВЗ	П3	ИДВ
DOORS NG	П	ДВЗ	П3	ИДВ
ReqView	-	-	-	-
Jama	П	ДВЗ	П3	ИДВ
Polarion	АЧ	ДВ	П2	ИМ
	П	ДВЗ	П3	ИД
RMTOO	АН	ДВЗ	П3	ИДВ
aNimble	-	-	-	-
ProR	АН	ДВ	А1	ИДВ
Requality	АН	ДВЗ	П3	ИДВ

3.1 ID – Идентификатор версии:
АЧ – автоматический, читаемый;
АН – автоматический, нечитаемый;
П – задаваемый пользователем.
3.2 Атрибуты – поддерживаемый набор атрибутов версии:
Д – дата и время изменения;
В – автор изменения;
З – комментарий.
3.3 Сохранение – подход к сохранению изменений:
П/А – сохранение выполняется пользователем/автоматически:
сохранение при редактировании отдельного объекта;
сохранение в пределах сессии работы с инструментом;
работа может быть прервана и продолжена после перезагрузки.
3.4 Операции – поддержка операций над версиями:
И – поддержка визуализации истории изменений.
Поддержка сравнения двух версий:
Р – рабочей и выбранной;
Д – дух выбранных.
М – выбранной и состояния до изменения.
В – восстановление состояния объекта выбранной версии.

Управление историей изменений на уровне отдельных объектов (табл. 4) дополняет историю изменений на уровне каталога более удобными средствами работы в контексте одного объекта. Для ReqView и aNimble это единственная возможность отслеживать историю разработки требований, а Polarion, RMTOO и ProR, наоборот, работают с историей изменений только на уровне всего каталога. При этом они предоставляют возможность просмотра истории изменений выбранного объекта, которая автоматически извлекается из истории версий проекта. Все остальные инструменты формируют историю изменений объектов, в рамках которой версии объектов обладают собственным набором атрибутов и собственными

операциями по просмотру, сравнению и восстановлению. В подавляющем большинстве случаев формирование версии происходит независимо от формирования версии каталога в целом. Исключением является Requality, у которого новая версия объекта может быть сформирована только при сохранении всего каталога, но при этом версии объекта управляются по собственным правилам и обладают независимым идентификатором.

Табл. 4. История изменений на уровне объектов
Table 4. History of changes at the object level

	4.1 ID	4.2 Атрибуты	4.3 Сохранение	4.4 Операции
RequisitePro	АЧ	ДВ	НП2	ИР
DOORS	АН	ДВ	НП2	ИМВ
DOORS NG	АН	ДВ	НА1	И
ReqView	АН	ДВ	НА1	ИМ
Jama	П(АН)	ДВ	НП2 (НА1)	ИМ
Polarion	-	-	-	ИД
RMTOO	-	-	-	ИД
aNimble	АЧ	ДВ	НП1	ИРВ
ProR	-	-	-	ИДВ
Requality	АЧ	ДВ	П3	ИР

4.1 ID – идентификатор версии:
АЧ – автоматический, читаемый;
АН – автоматический, нечитаемый;
П – задаваемый пользователем.
4.2 Атрибуты – поддерживаемый набор атрибутов версии:
Д – дата и время изменения;
В – автор изменения.
4.3 Сохранение – подход к сохранению изменений:
Н – сохранение выполняется независимо от версии проекта.
П/А – сохранение выполняется пользователем/автоматически:
1 – сохранение при редактировании отдельного объекта.
2 – сохранение в пределах сессии работы с инструментом.
3 – работа может быть прервана и продолжена после перезагрузки.
4.4 Операции – поддержка операций над версиями:
И – Поддержка визуализации истории изменений;
Поддержка сравнения двух версий:
Р – рабочей и выбранной;
Д – двух выбранных;
М – выбранной и состояния до изменения;
В – восстановление состояния выбранной версии.

Все инструменты кроме aNimble и ProR поддерживают процесс анализа требований, отслеживая статус утвержденности требований (табл. 5). Эта пара вместе с Requality также не предоставляет специализированных возможностей по анализу влияние последствий изменений по связям между объектами.

Табл. 5. Процессные аспекты и организация совместной работы
Table 5. Process Aspects and Collaboration

	5.1 Статус	5.2 Изменения	5.3 Разрешение конфликтов			
			Слияние	Блокировки	Доступ	ПР
RequisitePro	+	+		К, П	Т	
DOORS	+	+		П, О	К, Т, С	+
DOORS NG	+	+		П, О	К, Т, С	+
ReqView	+	+	+			
Jama	+	+		О	К, Т	+
Polarion	+	+		П, О	К, Т	+
RMTOO	+	+	+*		К*	
aNimble						
ProR			+*		К*	
Requality	+		+*		К*	

** - с использованием системы управления версиями GIT*

5.1 Статус – поддержка статуса утвержденности.
5.2 Изменения – анализ последствий изменений.
5.3 Разрешение конфликтов:
Слияние – слияние изменений.
Блокировки:
К – каталога целиком;
П – поддерева каталога;
О – отдельных элементов или свойств.
Доступ – поддержка ограничений прав доступа:
К – ограничения уровня проекта;
Т – ограничения на типы объектов;
С – ограничения уровня свойств.
ПР – Поддержка представлений для различных ролей.

Относительно поддержки командной работы инструменты делятся на три группы. Первая группа инструментов, включающая RequisitePro, DOORS, DOORS NG, Jama и Polarion, предоставляют средства блокировки части требований на время редактирования. Вторая группа (ReqView, RMTOO, ProR и Requality) придерживается оптимистичного подхода к разрешению конфликтов, предоставляя возможность свободно редактировать любые части каталога и проверяя наличие несовместимых модификаций только в момент сохранения изменений. Если такие модификации обнаружены, то пользователю приходится проанализировать несовместимые изменения и сформировать новую версию, в которой должны быть учтены все конфликтующие модификации. В третьей группе находится только aNimble, который при одновременном редактировании одного объекта, просто записывает последнюю сохраняемую версию, тем самым теряя изменения, выполненные параллельно. В результате для предотвращения проблем, возникающих при одновременной работе над проектом aNimble необходимо применять исключительно организационные меры защиты.

Из приятных дополнений можно отметить возможность настройки специализированных представлений каталога требований, привязываемых к роли, которую выполняет пользователь, работая с данным каталогом. Такая возможность поддерживается в инструментах DOORS, DOORS NG, Jama и Polarion.

В табл. 6 представлена информация о поддержке инструментами дополнительных функциональных возможностей. Импорт-экспорт каталога в той или иной форме поддерживают все инструменты за исключением aNimble, причем наблюдается достаточно широкая поддержка формата ReqIF. К сожалению, недостаточная детальность спецификации формата не позволяет рассчитывать на беспроблемность обмена данными с его помощью. Также следует отметить растущую популярность предоставления доступа к содержимому каталога требований при помощи веб-сервисов. Хотя шаблоны проектов или отдельных требований встречаются в большинстве инструментов, более богатые возможности переиспользования требований, в рамках которых вместо принципа копирования-вставки поддерживается сохранение связи с шаблоном с возможностью автоматического распространения вносимых в него модификаций, доступны только в DOORS NG, Jama, Polarion и Requality. Генерация отчетов на основе каталога требований присутствует практически во всех инструментах, только в ProR для этого необходимо применять какие-то внешние инструменты. Проактивное информирование пользователей об изменениях в каталоге получило широкое распространение в коммерческих инструментах и отсутствует в свободно распространяемых.

Табл. 6. Дополнительные функциональные возможности
Table 6. Additional functionality

	6.1		*6.2*		*6.3*	*6.4*
	WebSvc	*ИЭ*	*Шабл.*	*ПИ*	*Отчеты*	*Информирование*
RequisitePro	A	H	+		+	+
DOORS	A	ReqIF2	+		+	+
DOORS NG	T	ReqIF2	+	+	+	+
ReqView		ReqIF1			+	
Jama	A	ReqIF2	+	+	+	+
Polarion	A	ReqIF2	+	+	+	+
RMTOO		ReqIF1	+		+	
aNimble			+		+	
ProR		ReqIF2				
Requality	C	ReqIF2	+	+	+	

6.1 Обмен требованиями с другими инструментами:
WebSvc - поддержка интеграции при помощи web-сервисов:
T – встроенная поддержка OSLC;
A – поддержка OSLC при помощи дополнений;
C – поддержка нестандартного интерфейса.
ИЭ – поддержка операций импорта/экспорта каталога:
ReqIF1 – только импорт ReqIF документов;
ReqIF2 – импорт/экспорт ReqIF документов;
H – нестандартный формат импорта и экспорта.
6.2 Поддержка шаблонов и повторное использование:
Шабл. – поддержка шаблонов;
ПИ – поддержка повторного использования.
6.3 Отчеты – Поддержка генерации отчетов.
6.4 Информирование – Проактивное информирование об изменениях.

В соответствии с методикой для каждого инструменты была проведена экспертная оценка полноты поддержки возможностей по каждой из восьми категорий, описанных в подразделе

3.1. Результаты этой оценки в графической форме представлены на рис. 1. Детальное описание алгоритма выставления оценок и сами оценки по каждой категории представлены в [34].

3.4 Выводы

Рассматривая пригодность инструментов для использования в разработке ответственных систем, следует отметить следующие значимые недостатки:

- В aNimble отсутствует управление историей изменений на уровне всего каталога, поддержка совместной работы и операции импорта-экспорта.
- В ReqView отсутствует управление историей изменений на уровне всего каталога.
- В RMTOO отсутствует возможность не допустить переиспользования идентификаторов после удаления объекта.
- ProR поддерживает историю изменений на уровне всего каталога, но только посредством автоматического сохранения объекта при завершении его редактирования, без читабельного идентификатора версии и без возможности добавить к версии комментарий. Также в ProR отсутствуют читабельные идентификаторы требований и встроенные средства генерации отчетов.
- RequisitePro содержит необходимую функциональность, но уже не поддерживается и устарел по многим параметрам.

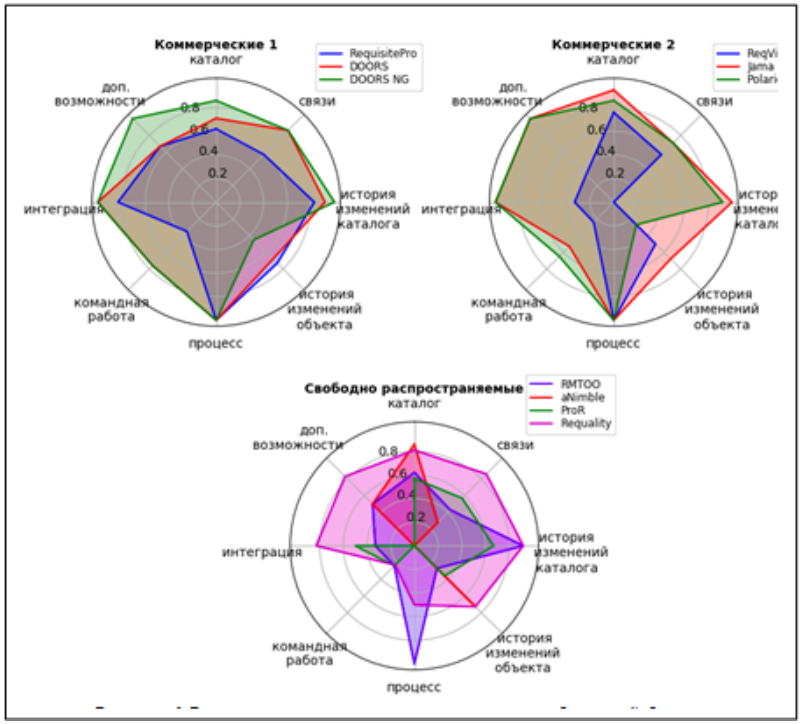

Рис. 1. Результаты оценки инструментов
Fig. 1 Results of tools' evaluation

Суммируя полученные результаты можно сделать следующие выводы. Коммерческие инструменты за исключение ReqView поддерживают все базовые функциональные возможности, необходимые для использования при разработке ответственных систем. Поэтому при сравнении между собой на первый план выходит стоимость владения этими инструментами, удобство использования, возможности по расширению их функциональности и адаптации к особенностям проекта.

Свободно распространяемые инструменты пока не предоставляют всей необходимой функциональности. Исключением является инструмент Requality, в котором присутствуют все наиболее важные функциональные возможности, хотя и наблюдается отставание относительно поддержки командной работы и автоматизации анализа влияния изменений. Это не препятствует использованию Requality в нескольких проектах разработки бортового программного обеспечения в соответствии с КТ-178С, более того его эксплуатации в этих проектах является основой для формирования требований к доработке инструмента. При этом следует отметить, что дополнительная функциональность, адаптирующая Requality для применения при разработке ответственных систем, доступна только в виде коммерческих плагинов.

4. Заключение

В рамках настоящей работы выделен набор функциональных возможностей инструментов управления требованиями, необходимый для их применения при разработке ответственных систем. Для проведения оценки доступности этих возможностей в существующих инструментах предложена методика, основанная на анализе публичной информации о рассматриваемых инструментах и экспертной оценки их возможностей. Данная особенность позволяет применить методику к любому инструменту. С другой стороны, она несет в себе ряд недостатков: субъективность экспертных оценок и возможность получения недостоверных оценок ввиду неполноты документации или недостаточной аккуратности ее изучения. Тем не менее, методика позволяет получить первичную оценку, которая может быть уточнена в дальнейшем по мере получения дополнительной информации.

В рамках настоящей работы методика применена для оценки наиболее распространенных коммерческих программных решений, декларирующих поддержку управления требованиями в контексте разработки ответственных систем (IBM Rational RequisitePro, IBM Rational DOORS, IBM Rational DOORS Next Generation, ReqView, Jama и Polarion), а также ряда свободно распространяемых инструментов (RMTOO, aNimble Platform, Eclipse ProR, Requality). Для каждого из инструментов проведен анализ документации, учебно-методических материалов и исходного кода и сформированы экспертные оценки их возможностей.

В результате анализа выделены значимые недостатки у ряда инструментов, отмечен паритет по реализации базовых функциональных возможностей в коммерческих инструментах и отставание по возможностям свободно распространяемых инструментов.

Список литературы

[1] ISO/IEC/IEEE 29148 Systems and software engineering – Life cycle processes – Requirements engineering.

[2] Dabney, J. B. Return on Investment of Independent Verification and Validation Study Preliminary. Phase 2B Report. NASA, 2003.

[3] GAO-06-391 Assessments of Selected Major Weapon Programs, Report to Congressional Committees, United States Government Accountability Office, 2006.

[4] GAO-09-326SP Assessments of Selected Major Weapon Programs, Report to Congressional Committees, United States Government Accountability Office, 2009.

[5] В.В. Кулямин, Н.В. Пакулин, О.Л. Петренко, А.А. Сортов, А.В. Хорошилов. Формализация требований на практике. Препринт No. 13 ИСП РАН, 2006, 70 стр.

[6] Requirements management: A practice guide, PMI, 2016, 82 p.

[7] Карл И. Вигерс. Разработка требований к программному обеспечению. Русская редакция, 2004, 576 стр.

[8] Руководство Р-4754А по разработке воздушных судов гражданской авиации и систем. М., АР МАК, 2016, 131 стр.

[9] SAE ARP4754A. Guidelines for Development of Civil Aircraft and Systems. 2010.

[10] Квалификационные требования КТ-178С. Требования к программному обеспечению бортовой аппаратуры и систем при сертификации авиационной техники. М., АР МАК, 2016, 131 стр.

[11] Software Considerations in Airborne Systems and Equipment Certification (RTCA DO-178C), 2011.

[12] Квалификационные требования КТ-254. Руководство по гарантии конструирования бортовой электронной аппаратуры. М., АР МАК, 2011, 89 стр.

[13] Design Assurance Guidance for Airborne Electronic Hardware (RTCA DO-254), 2000.

[14] Горелиц Н.К., Гукова А.С., Песков Е.В. Критерии, предъявляемые к программному обеспечению для разработки сложных сертифицируемых систем, критичных по безопасности. Труды ИСП РАН, том 30, вып. 4, 2018 г., стр. 63-78. DOI: 10.15514/ISPRAS-2018-30(4)-4

[15] Joy Beatty, Megan Jackson Stowe et al. Requirements Management Tool Evaluation Report. Seileve, 2016.

[16] Сабуров М.А., Синицын С.В. Роль учета состояния конфигурации программной продукции в управлении проектами. Авиакосмическое приборостроение, 2008, № 6, стр. 2-6.

[17] Requirements Interchange Format, The Object Management Group (OMG), 2016

[18] Open Services for Lifecycle Collaboration Requirements Management Specification Version 2.0. IBM, 2012

[19] Juan M. Carrillo de Gea, Joaquín Nicolás, José L. Fernández Alemán, Ambrosio Toval, Christof Ebert, Aurora Vizcaíno. Requirements engineering tools: Capabilities, survey and assessment. Information and Software Technology, vol. 54, no. 10, 2012, pp. 1142-1157

[20] Rational Unified Process Best Practices for Software Development Teams. Rational Software White Paper, 2001

[21] Rational RequisitePro. Version 2003.06.00, Rational Software Corporation, 2006

[22] IBM Rational RequisitePro. URL: ftp://ftp.software.ibm.com/software/rational/web/datasheets/reqpro.pdf, дата обращения 20.12.2019

[23] Getting started with Rational DOORS Next Generation. URL: https://jazz.net/help-dev/clm/index.jsp?re=1&topic=/com.ibm.rational.rrm.help.doc/topics/c_compose_reqs.html&scope=null, дата обращения 20.12.2019

[24] И.В. Ковернинский, А.В. Кан, В.Б. Волков, Ю.С. Попов, Н.К. Горелиц. Практический опыт реализации подходов программной и системной инженерии для управления требованиями при разработке программного обеспечения в авиационной отрасли. Труды ИСП РАН, том 28, вып. 2, 2016, стр. 173-180. DOI: 10.15514/ISPRAS-2016-28(2)-11

[25] Rational solution for Collaborative Lifecycle Management V6.0.6 documentation. URL: https://www.ibm.com/support/knowledgecenter/SSJJ9R_6.0.6/com.ibm.rational.clm.doc/helpindex_clm.html, дата обращения 20.12.2019

[26] ReqView Documentation Contents. URL: https://www.reqview.com/doc/welcome.html, дата обращения 20.12.2019

[27] Jama Software. URL: https://community.jamasoftware.com, дата обращения 20.12.2019

[28] Polarion ALM Platform Online Help System. URL: https://almdemo.polarion.com/polarion/help/index.jsp, дата обращения 20.12.2019

[29] rmToo – Requirements Management Tool. URL: http://rmtoo.florath.net/, дата обращения 20.12.2019

[30] aNimble Platform. URL: https://sourceforge.net/projects/nimble/http://rmtoo.florath.net/, дата обращения 20.12.2019

[31] ProR ProR Requirements Engineering Platform. URL: http://www.eclipse.org/rmf/pror/, дата обращения 20.12.2019

[32] Requality: руководство пользователя. URL: http://requality.org/ru/doc.ru.html, дата обращения 20.12.2019

[33] Кильдишев Д.С., Хорошилов А.В. Формализация метамодели системы управления требованиями. Труды ИСП РАН, том 30, вып. 5, 2018 г., стр. 163-176. DOI: 10.15514/ISPRAS-2018-30(5)-10.

[34] Горелиц Н. К., Кильдишев Д. С, Хорошилов А. В. Методика анализа инструментов управления требованиями к ответственным системам, Препринт 32, ИСП РАН, 2019 г. (в печати)

Requirements management for safety-critical systems. Overview of solutions

[1]*N.K. Gorelits <nkgorelits@2100.gosniias.ru>*
[2] *D.S. Kildishev <kildishev@ispras.ru>*
[2,3,4,5] *A.V. Khoroshilov <khoroshilov@ispras.ru>*

[1] *State Research Institute of Aviation Systems, 125319, Russia, Moscow, Viktorenko Str, 7*
[2] *Ivannikov Institute for System Programming of the Russian Academy of Sciences, 25, Alexander Solzhenitsyn st., Moscow, 109004, Russia*
[3] *Moscow Institute of physics and technology (State University), 141700, Moscow region, Dolgoprudny, Institutskiy per., 9*
[4] *Moscow State University named after M. V. Lomonosov, Moscow, 119991, GSP-1, Lenin hills, d. 1*
[5] *National research University "Higher school of economics", 101000, Russia, Moscow, Myasnitskaya, d. 20*

Abstract. Requirements are an integral part of any software and hardware development process. The area where requirements become significantly important is the development of safety-critical systems which usage may cause risks on human lives. So the process of their development is often maintained by certification centers that requires from developers to meet the best practices supporting the safety of end product. This article reveals one possible approach to requirements management that was based on experience of embedded hardware development for civil avionics. This approach is now spread over different areas. Authors list the set of common tasks related to given approach. They also define the set of software features used to reduce the complexity of development and to mitigate risks. Authors review set of existing solutions in requirements management area using the listed features. In this article it is also defined on how given features can be applied within the given approach.

Keywords: requirements management; safety-critical systems; requirement management tools; change management; traceability

DOI: 10.15514/ISPRAS-2019-31(1)-2

For citation: Gorelits N.K., Kildishev D.S., Khoroshilov A.V. Requirement management for safety-critical systems. Overview of solutions. Trudy ISP RAN/Proc. ISP RAS, vol. 31, issue 1, 2019. pp. 25-48 (in Russian). DOI: 10.15514/ISPRAS-2019-31(1)-2

References

[1] ISO/IEC/IEEE 29148 Systems and software engineering – Life cycle processes – Requirements engineering.
[2] Dabney, J. B. Return on Investment of Independent Verification and Validation Study Preliminary. Phase 2B Report. NASA, 2003.
[3] GAO-06-391 Assessments of Selected Major Weapon Programs, Report to Congressional Committees, United States Government Accountability Office, 2006.
[4] GAO-09-326SP Assessments of Selected Major Weapon Programs, Report to Congressional Committees, United States Government Accountability Office, 2009.
[5] V.V. Kulyamin, N.V. Pakulin, O.L. Petrenko, A.A. Sortov, A.V. Khoroshilov. Formalization of requirements in practice. Preprint No. 13, ISP RAS, 2006, 70 стр. (in Russian).
[6] Requirements management: A practice guide, PMI, 2016, 82 p.
[7] Karl Wiegers. Software Requirements, 2nd ed. Microsoft Press, 2003, 544 p.
[8] Guideline R-4754A on the development of civil aircraft and systems. M., AR MAK, 2016, 131 p. (in Russian).
[9] SAE ARP4754A. Guidelines for Development of Civil Aircraft and Systems. 2010.

[10] Qualification requirements CT-178C. Software requirements for onboard equipment and systems for certification of aviation equipment. M., AR MAK, 2016, 131 p. (in Russian).
[11] Software Considerations in Airborne Systems and Equipment Certification (RTCA DO-178C), 2011.
[12] Qualification Requirements CT-254. Guidance on the warranty design of onboard electronics. M., AR MAK, 2011, 89 p.
[13] Design Assurance Guidance for Airborne Electronic Hardware (RTCA DO-254), 2000.
[14] Gorelits N.K., Gukova A.S., Peskov E.V. Criteria for software to safety-critical complex certifiable systems development. Trudy ISP RAN/Proc. ISP RAS, vol. 30, issue 4, 2018, pp. 63-78 (in Russian). DOI: 10.15514/ISPRAS-2018-30(4)-4
[15] Joy Beatty, Megan Jackson Stowe et al. Requirements Management Tool Evaluation Report. Seileve, 2016.
[16] M.A. Saburov, S.V. Sinitsin. The Role of Software Configuration Status According in Project Management. Aerospace Instrument-Making, No. 6, 2008, pp. 2-6 (in Russian).
[17] Requirements Interchange Format, The Object Management Group (OMG), 2016
[18] Open Services for Lifecycle Collaboration Requirements Management Specification Version 2.0. IBM, 2012
[19] Juan M. Carrillo de Gea, Joaquín Nicolás, José L. Fernández Alemán, Ambrosio Toval, Christof Ebert, Aurora Vizcaíno. Requirements engineering tools: Capabilities, survey and assessment. Information and Software Technology, vol. 54, no. 10, 2012, pp. 1142-1157
[20] Rational Unified Process Best Practices for Software Development Teams. Rational Software White Paper, 2001
[21] Rational RequisitePro. Version 2003.06.00, Rational Software Corporation, 2006
[22] IBM Rational RequisitePro. URL: ftp://ftp.software.ibm.com/software/rational/web/datasheets/reqpro.pdf, accessed20.12.2019
[23] Getting started with Rational DOORS Next Generation. URL: https://jazz.net/help-dev/clm/index.jsp?re=1&topic=/com.ibm.rational.rrm.help.doc/topics/c_compose_reqs.html&scope=null , accessed20.12.2019
[24] Koverninsky I.V., Kan A.V., Volkov V.B., Popov Yu.S., Gorelits N.K. Practical experience of software and system engineering approaches in requirement management for software development in aviation industry. Trudy ISP RAN/Proc. ISP RAS, vol. 28, issue 2, 2016, pp.173-179 (in Russian). DOI: 10.15514/ISPRAS-2016-28(2)-11
[25] Rational solution for Collaborative Lifecycle Management V6.0.6 documentation. URL: https://www.ibm.com/support/knowledgecenter/SSJJ9R_6.0.6/com.ibm.rational.clm.doc/helpindex_clm.html, accessed20.12.2019
[26] ReqView Documentation Contents. URL: https://www.reqview.com/doc/welcome.html, accessed20.12.2019
[27] Jama Software. URL: https://community.jamasoftware.com, accessed20.12.2019
[28] Polarion ALM Platform Online Help System. URL: https://almdemo.polarion.com/polarion/help/index.jsp, accessed20.12.2019
[29] rmToo – Requirements Management Tool. URL: http://rmtoo.florath.net/, accessed20.12.2019
[30] aNimble Platform. URL: https://sourceforge.net/projects/nimble/http://rmtoo.florath.net/, accessed20.12.2019
[31] ProR ProR Requirements Engineering Platform. URL: http://www.eclipse.org/rmf/pror/, accessed20.12.2019
[32] Requality: User Manual (in Russian). URL: http://requality.org/ru/doc.ru.html, accessed20.12.2019
[33] D.S. Kildishev, A.V. Khoroshilov. Formalizing Metamodel of Requirement Management System. Trudy ISP RAN/Proc. ISP RAS, vol. 30, issue. 5, 2018, pp. 163-176. DOI: 10.15514/ISPRAS-2018-30(5)-10
[34] Gorelits N.K., Kildishev D.S., Khoroshilov A.V. Methods to analyze tools for requirements management for critical systems. Preprint 32, ISP RAS, 2019 (in print) (in Russian)