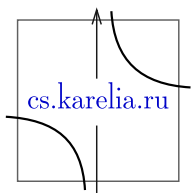
Петрозаводский государственный университет
Институт Математики и информационных технологий
Кафедра Информатики и математического обеспечения

Обеспечение качества программного продукта.

Лектор:

к.т.н., доцент Богоявленский Ю. А.
ybgv@cs.karelia.ru



Разделы

- Обеспечение качества, определение, цели, факторы
- Деятельность по обеспечению качества ПО: технические проверки и аудиты
- Деятельность по обеспечению качества: инспектирование, верификация и валидация
- Разработка требований к ответственным системам
- Анализ инструментов управления требованиями

Обеспечение качества, определение, цели, факторы.
Содержание

- Определение качества
  - Расширенное определение качества
  - Различия между контролем и обеспечением качества
- Цели обеспечения качества
- Факторы качества
  - Классическая модель факторов качества
  - Метрики качества
  - Группы метрик качества
  - Характеристики качества внутренних и внешних метрик
  - Атрибуты характеристик качества
  - Характеристики качества метрик «при использовании»
  - Что определяют характеристики качества

Определение качества программного обеспечения

В авторитетном словаре программной инженерии IEEE Std 610.12-90 «IEEE Standard Glossary of Software Engineering Terminology» записано:

- Качество ПО — это степень соответствия системы,компонента или процесса определенным требованиям.
- Качество ПО — это степень соответствия системы, компонента или процесса нуждам или ожиданиям заказчика, пользователя.

В 1979 году **Ф. Кросби** писал: «качество означает соответствие требованиям». **Дж. Джуран** в 1988 году ввел такое определение:

- Качество определяется теми характеристиками продукта, которые удовлетворяют потребности заказчика и таким образом обеспечивают удовлетворение продуктом.
- Качество означает отсутствие недостатков.

Определение качества программного обеспечения

Определение Кросби отсылает к степени соответствия написанного ПО тем спецификациям требований, которые были подготовлены заказчиком. Это означает, что погрешности спецификации не учитываются и не снижают программное качество, оцениваемое в пределах погрешностей подхода.

Определение Джурана нацелено на полное удовлетворение заказчика и объявляет истинной целью достижения качества «осуществление реальных нужд заказчика». В этом случае разработчик обязан приложить существенные усилия к исследованию и корректировке спецификации требований заказчика.

Определение термина «обеспечение качества ПО»

Чаще всего используют определение термина «обеспечения качества ПО» (SQA — software quality assurance) из словаря IEEE (1990 г.).

Обеспечение качества ПО — это:

1. планируемый и систематичный паттерн всех действий, которые обеспечивают полную уверенность в том, что элемент или продукт соответствует определенным техническим требованиям;
2. набор видов деятельности, проектируемый для оценки процесса, по которому продукты разрабатываются или производятся. Отличается от понятия «контроль качества».

Определение термина «обеспечение качества ПО»

Расширенное определение качества

Отметим, что это определение IEEE ограничивает область применения SQA исключая из нее вопросы сопровождения, а также ограничения по времени и бюджету. Также следует учесть следующие расширения:

- Распространить область действия SQA на весь жизненный цикл программной системы.
- Не ограничивать SQA техническими вопросами функциональных требований, а распространить его на вопросы планирования и учета бюджета.

Обеспечение качества ПО — систематический, планируемый набор действий, необходимых для формирования приемлемого уровня уверенности в том, что процесс разработки и сопровождения программной системы соответствует установленным функциональным техническим требованиям, а также организаторским требованиями соблюдения план-графика и бюджетных ограничений.

Определение термина «обеспечение качества ПО»

Различия между контролем и обеспечением качества

Контроль качества

- Определен как «набор видов деятельности, проектируемых для оценки качества разрабатываемого или производимого продукта» (IEEE, 1990).
- Применяется к разработке продукта до момента ее завершения.
- Типичная деятельность - тестирование.

Обеспечение качества

- Главная цель — минимизировать стоимость гарантированного качества за счет множества действий, выполняемых на различных этапах процесса разработки(производства).
- Действия по обеспечению качества существенно уменьшают процент продуктов, непригодных для отправки, сокращают стоимость гарантированного качества.

Цели обеспечения качества

- На этапе разработки ПО (ориентированы на процесс):
  1. Обеспечение приемлемого уровня уверенности в том, что ПО будет соответствовать функциональным техническим требованиям.
  2. Обеспечение приемлемого уровня уверенности в том, что ПО будет создаваться в соответствии с план-графиком и бюджетными требованиями.
  3. Инициализация и управление действиями по совершенствованию и повышению эффективности разработки ПО, а также действий SQA.

Цели обеспечения качества

- На этапе сопровождения ПО (ориентированы на продукт):
  4. Обеспечение приемлемого уровня уверенности в том, что действия по сопровождению ПО будут соответствовать функциональным техническим требованиям.
  5. Обеспечение приемлемого уровня уверенности в том, что действия по сопровождению ПО будут соответствовать план-графику и бюджетным требованиям.
  6. Инициализация и управление действиями по совершенствованию и повышению эффективности сопровождения ПО, а также действий SQA.

Факторы качества

Классическая модель факторов качества

Дж. МакКолл сгруппировал 11 факторов в три категории:

- Операционные факторы
  1. Правильность
  2. Надежность
  3. Эффективность
  4. Целостность
  5. Практичность
- Факторы изменяемости
  6. Сопровождаемость
  7. Гибкость
  8. Тестируемость
- Факторы перемещаемости
  9. Мобильность
  10. Многократность использования
  11. Способность к взаимодействию

Факторы качества

Метрики качества

Формула для выражения каждого фактора:

$$F_j = \sum_{i=1}^{n} c_i * m_i$$

, где F_j — j-й фактор качества ПО, c_i — весовой коэффициент, m_i— метрика, оказывающая влияние на фактор качества ПО.

Для вычисления метрик создавались проверочные листы со схемой применения:

- Задаются несколько вопросов.
- Записываются ответы на вопросы.
- Ответы обрабатываются (взвешиваются, суммируются и т. д.).

Факторы качества

Группы метрик качества

Для определения факторов качества применяют серию из четырех международных стандартов:

1. ISO/IEC 9126-1:2001 «Quality model»
2. ISO/IEC TR 9126-2:2003 «External metrics»
3. ISO/IEC TR 9126-3:2003 «Internal metrics»
4. ISO/IEC TR 9126-4:2004 «Quality in use metrics»

Внутренние метрики фиксируют внутреннее качество, проявляющееся в ходе разработки ПО.

Внешние метрики определяют внешнее качество, заданное требованиями заказчика и демонстрируемое характеристиками конечной системы.

Метрики «при использовании» измеряют качество в ходе нормальной эксплуатации программной системы конечными пользователями.

Факторы качества

Характеристики качества внутренних и внешних метрик

1. *Функциональность* определяет способность продукта обеспечивать требуемые функции и операции.
2. *Надежность* показывает возможности продукта для обеспечения достаточного уровня работоспособности в реальных условиях.
3. *Практичность* означает понятность, легкость использования и изучения продукта.
4. *Эффективность* демонстрирует эффективность продукта при разумном задействовании ресурсов.
5. *Сопровождаемость* измеряет возможности продукта быть изменчивым и обновляемым, а также удобным в сопровождении.
6. *Переносимость* означает способность продукта быть системой, переносимой из одной аппаратно-программной среды в другую.

Факторы качества

Атрибуты характеристик качества

Каждая характеристика детализируется своим набором атрибутов.

Таблица 1 Детализация характеристики «функциональность»

Атрибут	**Описание**
Пригодность (suitability)	Пригодность к обеспечению требуемой функциональности
Точность (accuracy)	Обеспечение заданной точности результатов
Способность к взаимодействию (interoperability)	Способность к взаимодействию с внешними системами
Защищенность (security)	Защищенность внутренней информации от неавторизованного использования
Согласованность (compliance)	Отсутствие противоречий с ограничениями стандарта

Факторы качества

Характеристики качества метрик «при использовании»

Метрики «при использовании» генерируются для четырех характеристик качества, перечисленных в таблице ниже.

Характеристика	Описание
Эффективность (effectiveness)	Возможность для пользователя достичь цели с достаточной точностью и полнотой
Продуктивность (productivity)	Возможность для пользователя достичь цели при использовании объема ресурсов, обеспечивающих нужную производительность
Безопасность (safety)	Обеспечение приемлемого уровня рисков (относящихся к людским ресурсам, данным, среде и т. д.)
Удовлетворенность запросов (satisfaction)	Возможность полного решения задачи с помощью продукта

В 2005 году начат процесс создания линейки международных стандартов второго поколения по качеству ПО — стандартов серии 25 000. Данная серия вбирает в себя весь положительный опыт линейки 9126. В частности, в ней уточнен и расширен до восьми набор характеристик качества для внутренних и внешних метрик, существенно обновлен набор характеристик (и введено 13 атрибутов) для метрик «при использовании».

Факторы качества

Что определяют характеристики качества

Стандартизованные характеристики качества оказывают существенную помощь заинтересованным лицам при формировании требований, они могут служить шаблоном требований. Ведь они определяют:

- Что должно делать ПО
- Степень надежности ПО
- Степень удобства использования
- Степень эффективности
- Удобство сопровождения
- Степень переносимости

Деятельность по обеспечению качества ПО: технические проверки и аудиты. Содержание

- Введение.
- Способы оценки качества проектного решения.
- Стандарты и процедурные нормы.
- Потенциальный источник ошибок.
- Технические проверки.
- Проверяемая документация.
- Группа технической проверки.
- Аудиты.
- Польза технических проверок.

Введение

Обеспечение качества ПО (SQA) — защитная деятельность, применяемая на всем протяжении жизненного цикла системы и включающая следующие виды действий:

- применение технических методов и средств анализа, проектирования, кодирования и сопровождения;
- проведение технических проверок и аудитов на каждом шаге разработки;
- верификация и валидация продукта;
- внедрение в жизнь стандартов;
- контроль изменений;
- проведение измерений показателей качества и их оценка;
- сохранение записей и формирование отчетности.

Введение

- Технические методы и средства помогают аналитику получить высококачественную спецификацию, а проектировщику (программисту) — провести высококачественное проектирование (программирование).
- После того как спецификация и проектное решение (программный код) созданы, осуществляется оценка их качества. Для этого используют самые разные действия: технические проверки и аудиты, верификацию и валидацию.

Способы оценки качества проектного решения

- Техническая проверка и аудит организуются как встреча персонала с единственной целью — обнаружить проблемы качества.
- Верификация и валидация — это не единичный акт, а процесс, проверяющий соответствие ПО своей спецификации и требованиям заказчиков. Верификация и валидация охватывают полный жизненный цикл ПО, начинаются на этапе анализа требований и завершаются проверкой готовой программной системы. Основными средствами верификации и валидации являются инспектирование и тестирование. Многие разработчики полагают, что полное тестирование обнаружит большинство ошибок, вследствие чего ослабляется необходимость в других действиях по обеспечению качества. К сожалению, тестирование эффективно не для всех классов ошибок.

Стандарты и процедурные нормы

Диапазон стандартов и процедурных норм, применяемых к процессу разработки ПО разными компаниями, существенно различен. Во многих случаях стандарты задаются заказчиком или контрактом. В других случаях они выбираются самими разработчиками. Хорошо организованный процесс SQA создает в компании атмосферу культивирования качества, стимулирующий команду к качественной работе и поиску путей повышения качества. При этом стандарты и процедурные нормы считаются «камертоном качества».

Если стандарты приняты, то действия по обеспечению качества должны гарантировать их соблюдение. Оценка соответствия стандартам может выполняться разработчиками как часть технической проверки или в ситуациях, где требуется независимая верификация, специальная группа качества может проводить свою собственную ревизию.

Потенциальный источник ошибок

Каждое изменение в ПО рассматривается как потенциальный источник ошибок или как предпосылка для распространения ошибок. Процесс контроля изменений способствует качеству ПО:

1. формализацией запросов на изменения;
2. оценкой сущности изменения;
3. контролем влияния изменения.

Контроль изменений осуществляется как на стадии разработки, так и на стадии сопровождения.

Важной целью обеспечения качества считается отслеживание качества и оценка влияния методологических и процедурных изменений на улучшение качества ПО. Для достижения этой цели должны собираться результаты измерений, проводимых по метрикам качества ПО.

Технические проверки

Проверки ПО — фильтр для процесса разработки. Они позволяют очистить от дефектов такие виды действий, как анализ, проектирование, кодирование. Существует много типов проверок, которые могут проводиться как часть процесса SQA (неформальная встреча, формальное представление программного проекта аудитории из заказчиков, руководства, технического персонала и т. д.). У каждого типа свое место.

Техническая проверка проводится программными инженерами для программных инженеров.

Технические проверки

Назначение технической проверки:

1. обнаружение ошибок и противоречий в функциях, логике или реализации для любой формы представления ПО, включая документацию;
2. проверка соответствия ПО требованиям;
3. обеспечение того, чтобы ПО представлялось согласно определенным стандартам;
4. получение универсальной формы представления ПО;
5. формирование более управляемого проекта.

Проверяемая документация

- Спецификация,
- Диаграммы,
- Программный код,
- План-график разработки,
- План тестирования,
- Механизм управления конфигурацией,
- Стандарты,
- Документация для пользователя.

Группа технической проверки

В группу проверки включаются такие специалисты, которые могут принести максимальную пользу. Например, при проверке артефактов проектирования следует привлечь проектировщиков из схожих проектов.

Группу образуют 3-4 человека, один из них назначается старшим. Кроме этого, по частным вопросам могут привлекаться дополнительные сотрудники. Проверяемые документы раздаются до начала проверки, для предварительного ознакомления. Сама проверка длится не более двух часов. По окончании проверки все замечания заносятся в отчет. Отчет подписывается всеми проверяющими.

Аудиты

Отличие аудита от технической проверки:

- нет ограничения по времени;
- в состав группы входят лица из внешних организаций;
- разработка рассматривается с точки зрения контрактных обязательств;
- выполненная работа привязывается к план-графику проекта.

Польза технических проверок

- Очевидная польза технических проверок — раннее обнаружение программных дефектов. Обнаруженные дефекты исправляются перед следующим шагом разработки.
- Статистика показывает, что 50-60% всех ошибок разработки ПО приходится на действия по проектированию. Технические проверки обеспечивают обнаружение до 75% недостатков проектирования. Таким образом, обнаруживая и устраняя большой процент ошибок, процесс проверки существенно сокращает стоимость последующих шагов разработки и сопровождения.

Вывод

- Технические проверки и аудиты выполняются на каждом шаге разработки ПО для обеспечения его качества.
- Целью проведения технической проверки и аудита является обнаружение проблем качества.
- Важную роль играет формирование стандартов и процедурных норм, применяемых к процессу разработки ПО, так как действия по обеспечению качества направлены на их соблюдение.

Заключение

Анализ большого числа программных проектов привел к следующим цифрам. Если принять цену устранения ошибки, обнаруженной в ходе проектирования, за единицу, тогда для ошибки:

- обнаруженной перед тестированием, цена составит 6,5 единицы;
- обнаруженной в течение тестирования, цена составит 15 единиц;
- обнаруженной после завершения проекта, цена составит 60-100 единиц.

Деятельность по обеспечению качества: инспектирование, верификация и валидация. Содержание

- Инспектирование.
- Роли группы инспектирования.
- Этапы процесса инспектирования.
- Верификация и валидация.

Инспектирование

В отличие от тестирования инспектирование является статическим способом поиска ошибок. Инспектирование заключается в просмотре и проверке артефактов проекта на наличие ошибок. Такими артефактами могут быть:

- требования,
- результаты проектирования,
- программный код.

Главная цель инспектирования — обнаружение дефектов, а не исследование общих проблем проекта.

Дефектами являются:

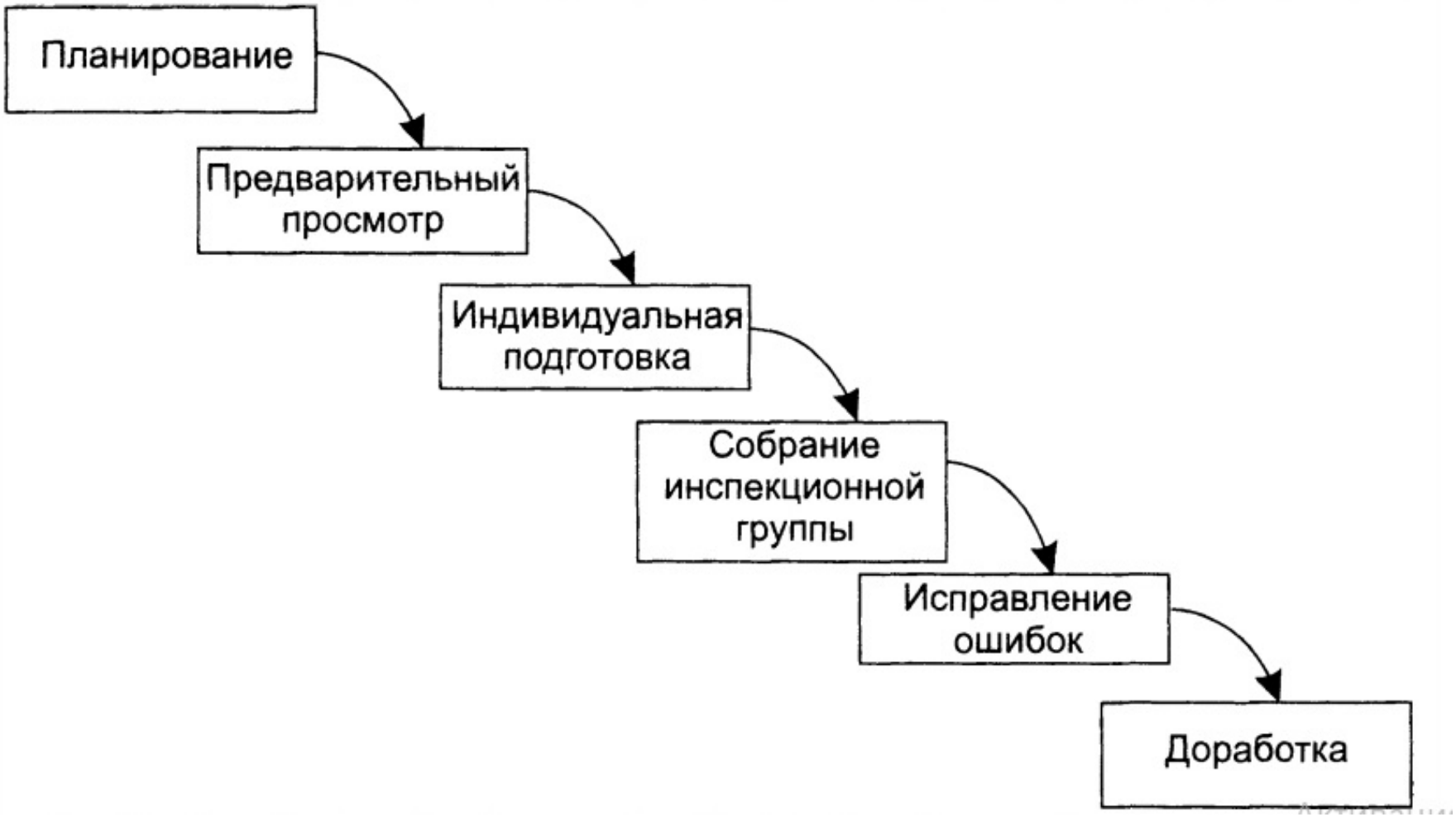
- ошибки во фрагменте системы,
- несоответствие фрагмента принятым стандартам.

Роли группы инспектирования

Процесс инспектирования формализован, выполняется группой разработчиков, состоящей из четырех человек. Автор отвечает за свою работу и за исправление в ней дефектов. Рецензент комментирует программный код, корректор проверяет код, регистратор записывает обнаруженные дефекты.

Роль	**Описание**
Рецензент	Комментирует (рецензирует) программный код на собрании группы инспектирования, отвечает за правильное проведение инспектирования
Автор	Несет ответственность за свою программу или документ, после завершения инспектирования самостоятельно исправляет все обнаруженные дефекты
Корректор	Ищет в программе (или документе) ошибки, противоречия и упущения
Регистратор	Отвечает за учет описания и классификацию обнаруженных дефектов, записывает результаты собрания группы инспектирования

Этапы процесса инспектирования



Пример проверочного списка для инспектирования программного кода

- Все ли переменные получили значения до начала их использования?
- Все ли константы именованы?
- Корректно ли заданы границы массивов?
- Выполняются ли условия для каждого условного оператора?
- Все ли циклы завершаются?
- Правильно ли расставлены скобки в составных операторах?
- Все ли варианты выбора реализуются в case-операторах?
- Используются ли в программе входные переменные?
- Все ли выходные переменные получают значения перед выводом на печать?
- Могут ли какие-нибудь входные данные привести к нарушению данных системы?

Пример проверочного списка для инспектирования программного кода

- Все ли вызовы процедур и функций содержат правильное количество параметров?
- Согласованы ли типы формальных и фактических параметров?
- В правильном ли порядке расположены параметры?
- Если модули обращаются к разделяемым ресурсам, обеспечены ли взаимное исключение и условная синхронизация?
- Если связанная структура данных изменяется, правильно ли переопределяются ее связи?
- Если используется динамическая память, правильно ли она распределяется?
- Происходит ли сбор мусора и решена ли проблема повисших указателей?
- Все ли возможные ошибки рассмотрены в условиях, определяющих исключительные ситуации?

Верификация и валидация

Верификация и валидация являются составной частью процесса контроля качества.

- Определение понятия «верификация»

Вариант	Описание
А	Процесс оценки системы или компонента для определения — удовлетворяют ли продукты данного этапа разработки условиям, наложенным в начале этапа
Б	Процесс подтверждения того, что ПО и ассоциированные с ним продукты соответствуют требованиям (например, по правильности, законченности, совместимости, точности) ко всем видам деятельности жизненного цикла (приобретению, поставке, разработке, использованию и сопровождению); соответствие стандартам, практикам и соглашениям для всех процессов жизненного цикла; полное и успешное завершение каждой деятельности жизненного цикла и соответствие всем критериям для инициализации последующих видов деятельности жизненного цикла (например, корректной сборки ПО)

Верификация и валидация

- Определение понятия «валидация»

Вариант	Описание
А	Оценка системы или компонента во время или в конце процесса разработки для определения — удовлетворяет ли она (он) указанным требованиям
Б	Процесс подтверждения того, что ПО и ассоциированные с ним продукты удовлетворяют системным требованиям к программному продукту, определенным для завершения каждого вида деятельности жизненного цикла, правильно решают проблемы (например, учитывают закономерности предметной области, бизнес-правила, используют правильные системные допущения), а также соответствуют назначению и нуждам пользователей

Верификация отвечает на вопрос: правильно ли строится, создается наша система? Валидация же проверяет: правильно ли работает система, отвечает ли построенная система пожеланиям и нуждам заказчика?

Пример

Приведем простой пример, воспользовавшись известной задачей Гленфорда Майерса [10]. Заказчик предложил разработать систему, которая «решает квадратные уравнения вида ax2 + Ьx + c = 0 для коэффициентов, представляемых целыми числами».

Сформулируем это пожелание в виде детальных требований:

- Пользователь вводит целые числа a, b и c.
- Система отыскивает решение уравнения ax2 + Ьx + c = 0 с точностью 0,001.

Затем мы пишем систему, реализующую эти требования.

Верификация и валидация

Верификация заключается в том, что мы анализируем процесс построения системы и проверяем, что в этом процессе все сделано правильно.

Положим, что процесс разработки включает в себя следующие шаги:

- Получение требований заказчика.
- Подготовка детальных требований.
- Одобрение заказчиком детальных требований.
- Программирование системы.
- Тестирование системы.

Верификация и валидация

Верификация проверяет следующие вопросы:.

- Выражают ли требования реальные запросы и нужды заказчика? Здесь верификация может включать в себя следующие пункты: является ли уравнение ax2 + Ьx + c = 0 корректной желаемой формой? Верно ли задана точность? Может ли быть a = 0?
- Правильно ли реализован процесс одобрения заказчиком? Здесь возможны такие вопросы: Предоставили ли мы заказчику достаточно времени? Объяснили ли мы заказчику все используемые термины?
- Реализует ли программный код системы все требования? Здесь следует выполнить инспектирование кода, где он анализируется с точки зрения соответствия каждому требованию.

Верификация и валидация

- В какой степени предлагаемые тесты покрывают область применимости системы? Для реальной системы следует применить инспектирование плана тестирования, тестовых вариантов и тестовых процедур. Эти действия дополняют собственно процесс тестирования.

Валидация продукта заключается в получении одобрения заказчиком полных требований и в тестировании конечного кода. Вывод: верификация проверяет соответствие ПО функциональным и нефункциональным требованиям. Валидация является более общим процессом, в ходе которого надо убедиться, что программная система соответствует ожиданиям валидация проводится после верификации, для того чтобы определить, насколько система соответствует не только спецификации, но и ожиданиям заказчика

Верификация и валидация

В ходе верификации и валидации используют два основных действия:

- Инспектирование ПО. Инспектирование осуществляется на всех этапах разработки программной системы. Параллельно с инспектированием может выполняться автоматический анализ программного кода и других артефактов (например, различных документов). Инспектирование и автоматический анализ являются статическими методами верификации и валидации, поскольку им не требуется работающая система.
- Тестирование ПО. Требует запуска фрагментов исполняемого кода. Тестирование считается динамическим методом верификации и валидации, так как применяется к работающей системе.

Выводы: инспектирование можно выполнять на всех этапах разработки системы, а тестирование — лишь в тех случаях, когда создан исполняемый код.

Верификация и валидация

Верификация и валидация выявляют лишь степень приближения к идеалу заказчика, степень соответствия программной системы запланированным целям.

Эта степень определяется:

- назначением системы (для критических систем степень соответствия максимальна, например для системы управления ядерным реактором);
- ожиданиями пользователей (хотят заплатить мало, но терпеть недостатки, или платят много и недостатков не признают);
- условиями на рынке программных продуктов (есть конкуренция для необходимого семейства продуктов, или она отсутствует).

Атрибуты качества. План обеспечения качества.

Содержание

- Атрибуты качества ПО.
- План обеспечения качества ПО.

Атрибуты качества ПО

К атрибутам качества можно отнести десятки характеристик продукта, хотя для большинства проектов хватает всего нескольких. Если разработчикам известно, какие характеристики наиболее важны для успеха проекта, они могут выбрать соответствующие приемы работы над архитектурой и дизайном, которые позволят достичь определенного качества. Для классификации атрибутов качества применяются различные схемы.

Классификация атрибутов качества:

- Внешнее качество.
- Внутреннее качество.

Атрибуты качества ПО

Потребность в конкретных атрибутах зависит от типа проекта:

- Встроенные системы.
- Интернет-приложения и корпоративные приложения.
- Настольные и мобильные системы.

Для различных частей системы необходимы различные сочетания атрибутов качества. Для одних крайне важна производительность, а для других — практичность. Например, в компании, занимающейся играми, могут требовать особой эмоциональной атмосферы в разрабатываемом ПО.

Атрибуты качества ПО

Если какие-то из требований к качеству относятся к определенным функциям, компонентам, функциональным требованиям или пользовательским историям, их нужно связать с соответствующими элементами в хранилище требований.

Внутреннее качество	Краткое описание
Эффективность	Насколько эффективно система использует ресурсы компьютера
Возможность модификации	Насколько легко обслуживать, модифицировать, улучшать и реструктурировать систему
Переносимость	Насколько легко заставить систему работать в другой операционной среде
Возможность повторного использования	В какой степени компоненты могут использоваться в других системах
Масштабируемость	Как хорошо система справляется с увеличением числа пользователей, транзакций, серверов и других расширений
Проверяемость и тестируемость	Как быстро разработчики и тестировщики могут подтвердить, что система реализована правильно

Атрибуты качества ПО

Внешнее качество	Краткое описание
Доступность	Насколько сервисы системы доступны, когда и где они нужны
Удобство установки	Насколько просто правильно установить, удалить и повторно установить приложение
Целостность	Насколько хорошо система защищает от неточности и потери данных
Совместимость	Насколько просто система может взаимодействовать и обмениваться данными с другими системами и компонентами
Производительность	Как быстро и предсказуемо система реагирует на ввод информации пользователем и на другие события
Надежность	Как долго система работает до первого сбоя
Устойчивость	Как хорошо система реагирует на неожиданные условия работы
Защита	Насколько хорошо система защищает от повреждений
Безопасность	Как хорошо система защищает от неправомерного доступа к приложению и его данным
Удобство использования	Как просто людям научиться использовать систему

План обеспечения качества ПО

План обеспечения качества ПО (SQA) отражает широкую панораму действий, ориентированных на формирование качества. Разрабатываемый группой SQA, план служит шаблоном действий, которые должны быть внедрены в каждый программный проект.

В стандарте IEEE Std 730-2002 предлагается следующая структура плана SQA:

- Цель.
- Упоминаемые документы.
- Руководство:
  - Организация.
  - Задачи.
  - Обязанности.
  - Оцениваемые ресурсы обеспечения качества.

План обеспечения качества ПО

- Документация:
  - Цель.
  - Минимальные требования к документации.
  - Прочее.
- Стандарты, практики, соглашения и метрики:
  - Цель.
  - Содержание.

План обеспечения качества ПО

- Проверки:
  - Цель.
  - Минимальные требования:
    - Проверка спецификаций ПО.
    - Проверка архитектурного проектирования.
    - Проверка детального проектирования.
    - Проверка плана верификации и валидации.
    - Аудит функциональности.
    - Аудит физических компонентов.
    - Внутренние аудиты процесса.
    - Проверка организации.
    - Проверка плана управления конфигурацией.
    - Проверка пост-реализации.
  - Прочие проверки и аудиты.

План обеспечения качества ПО

- Тестирование:
  - Может ссылаться на документацию по тестированию ПО.
- Отчеты о проблемах и корректирующие действия.
- Инструменты, технологии и методологии:
  - Может ссылаться на план управления программным проектом.
- Контроль носителей.
- Контроль поставщиков.
- Сбор, сопровождение и хранение протоколов.

План обеспечения качества ПО

- Обучение.
- Управление рисками:
  - Может ссылаться на план управления программным проектом.
- Словарь.
- Процедура изменения плана и версии.

При работе над планом SQA важно придерживаться предельно лаконичного стиля изложения. Если документ окажется слишком длинным, то вряд ли его дочитают до конца, что сведет на нет саму идею плана обеспечения качества.

Разработка требований к ответственным системам. Содержание

- Введение
- Характеристики требований
- Виды работы с требованиями
- Требования при разработке ответственных систем
- Основные задачи разработки ответственных систем
- Системы управления требованиями
- Структурирование и хранение требований
- Связи
- История изменений и управление изменениями
- Организация совместной работы
- Дополнительные возможности

Введение

ISO/IEC/IEEE 29148

Требование – это утверждение, транслирующее или выражающее потребность и связанные с ней ограничения и условия.

Представления требований:

- тексты на естественном языке,
- UML-диаграммы,
- тексты на специализированных предметно-ориентированных языках.

NB. Чем масштабнее проект и больше количество разработчиков, тем сложнее удерживать разработку в рамках, преодолевать разнообразные риски и в итоге достичь удовлетворительного результата, и тем заметнее эффект от четкой формулировки требований и последующей аккуратной работы с ними.

Введение

Тем не менее часто встречаются проекты со слабо формализованными требованиями, постановки задач с высокой степенью неопределенности.

Pulse of profession – опрос 4,5 тысяч специалистов:

Причины, по которым проекты, стартовавшие в организациях за последние 12 месяцев, потерпели неудачи?

Проблемы с требованиями (с их сбором, анализом, управлением):

- 2015 — 38% (второе место),
- 2016 — 37% (третье место),
- 2017 — 39% (второе место),
- 2018 — 35% (первое место).

Стоимость исправления ошибки, допущенной на этапе разработки требований заметно выше, чем стоимость ошибок на последующих этапах.

Характеристики требований

Для эффективной работы с требованиями необходимо, чтобы требования обладали следующими характеристиками.

1. Характеристики, касающиеся связи требований с предметной областью.
   - **Адекватность** – соответствие сформулированных требований всем аспектам потребностей и ожиданий пользователей, а также интересам всех остальных заинтересованных лиц.
   - **Выполнимость** – возможность реализовать требование в рамках заданных условий и ограничений.

Характеристики

2. Представления требований самих по себе (внутренние).
   - **Однозначность** – одинаковость понимания формулировок требований экспертами в соответствующей предметной области.
   - **Внутренняя полнота** – охваченность в описании требований всех аспектов и ситуаций, возможных в рамках описанного контекста работы системы.
   - **Непротиворечивость** – согласованность описаний требований друг с другом, отсутствие противоречий и расхождений между ними.
   - **Минимальность** – невыводимость одних требований из других на основе формальной логики.
   - **Простота** – отсутствие необходимости подразбиения требования на составные части.
   - **Отсутствие деталей реализации** – формулировка требований, а не возможных способов их реализации.
   - **Систематичность** – представление в виде системы с четко выделенными атрибутами и ясным описанием взаимосвязей и зависимостей между ними.

Характеристики

3 Характеристики, касающиеся использования требований в процессе разработки (внешние).

- **Проверяемость** – возможность для человека, обладающего определенными навыками, однозначно установить в каждой затрагиваемой требованием ситуации, выполнено оно или нарушено.
- **Прослеживаемость** – 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** – 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.

Виды работы с требованиями

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

Требования при разработке ответственных систем

I Наиболее аккуратное отношение к требованиям традиционно присутствует при разработке ответственных систем.

Основной особенностью ответственных систем является то, что дефекты в конечном продукте могут повлечь за собой риск для жизни и здоровья человека. В связи с этим, эксплуатация таких систем возможна только после прохождения процедуры сертификации.

Пример

Новая модель пассажирского авиалайнера может приступить к коммерческим рейсам только после получения сертификата типа воздушного судна, которое в России выдается Росавиацией.

Большинство современных регламентов сертификации ответственных систем предъявляют требования не только к конечному продукту, но и к процессам его разработки.

Основные задачи разработки ответственных систем

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

Основные задачи разработки ответственных систем

- **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.
- **Базовая версия.** Для разделов требований, которые были утверждены должна быть установлена базовая версия.
- **Трассируемость.** Должны быть установлены и задокументированы связи между требованиями и артефактами, полученными на их основе, такими как исходный код, тесты и т.д.

Основные задачи разработки ответственных систем

- **Управление изменениями.** Внесение изменений в требования должно выполняться по результатам рассмотрения явно сформулированного сообщения о проблеме, а вслед за внесением изменения должны быть проведены процедуры повторного анализа 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.

Основные задачи разработки ответственных систем

- **Данные трассировки.** Должны быть подготовлены данные трассировки, 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.

Системы управления требованиями

Специализированные системы управления требованиями обладают потенциалом для существенного снижения затрат на разработку, благодаря следующим возможностям:

1. структурирование и хранение требований;
2. поддержка связей;
3. история изменений и управление изменениями;
4. организация совместной работы и другие.

Структурирование и хранение требований

Как минимум, системы управления требованиями должны поддерживать хранение элементарных требований, но кроме того, при написании требований возникает необходимость в наличии вспомогательных объектов, таких как: определения терминов, примечания и комментарии и обоснования требований.

При организации хранения требований возникает ряд вопросов:

- Какой формат используется для представления текста и других типов информации?
- Как идентифицируются элементарные объекты?
- Какую структуру имеет каталог требований?
- Какие средства для редактирования текста система предоставляет?

Связи

Под связью понимается отношение элемента каталога требований либо с другим элементом каталога, либо с некоторой внешней сущностью.

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.

Связи обеспечивают поддержание информационной базы для управления проектом в актуальном, полном, целостном состоянии.

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

История изменений и управление изменениями

Поддержка регистрации истории изменений каталога требований является необходимым условием применимости инструмента для разработки ответственных систем.

Версионирование:

- базовый объект версионирования; В качестве таких объектов могут выступать: каталог с требованиями в целом; произвольное поддерево каталога; отдельный объект.
- идентификатор версии;
- поддерживаемый набор атрибутов версии (дата и время изменения, автор, комментарий);
- подход к сохранению изменений (выполняется пользователем или автоматически).

Организация совместной работы

Разрешение конфликтов:

- блокировку доступа на время редактирования,
- апостериорное слияние изменений в случае конфликта.

В процесс работы с требованиями вовлечены люди, решающие разные задачи. В зависимости от роли могут различаться права доступа к каталогу требований, объектам и даже свойствам объектов. Инструменты управления требованиями могут поддерживать такие ограничения прав.

Дополнительные возможности

В дополнение к базовому набору функциональных возможностей у систем управления требованиями имеется ряд дополнительных возможностей.

- Обмен требованиями с другими инструментами (например, при работе со сторонними организациями).
- Поддержка шаблонов требований и повторное использование.
- Поддержка генерации отчетов.
- Средства проактивного информирования об изменениях.

Анализ инструментов управления требованиями. Содержание

- Введение.
- Методика анализа инструментов.
- Объекты анализа.
- Краткое описание результатов анализа.
- Выводы.

Введение

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, а также инструменты ReqView, Jama и Polarion.
- Во вторую группу включены инструменты RMTOO, aNimble Platform, Eclipse ProR и разрабатываемый в ИСП РАН инструмент Requality.

Методика анализа инструментов

Оценка инструментов будет проводиться с точки зрения поддержки функциональных возможностей, необходимых для работы с требованиями при разработке ответственных систем. 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.

Методика анализа инструментов

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.

Методика анализа инструментов

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.

Объекты анализа

Для анализа использовались следующие версии инструментов и материалы:

- IBM Rational RequisitePro 7.1.5.
- IBM Rational DOORS 9.4.
- IBM Rational DOORS Next Generation 6.0.6.
- ReqView 2.4.0.
- Jama Connect версии 8.30.
- Polarion REQUIREMENTS 3.18.
- rmToo 24.0.
- aNimble Platform 0.4.
- Eclipse ProR 0.13.
- Requality 1.0.

Краткое описание результатов анализа

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.

Краткое описание результатов анализа

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

Краткое описание результатов анализа

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.

Краткое описание результатов анализа

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.

Краткое описание результатов анализа

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.

Краткое описание результатов анализа

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.

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.

Выводы

- 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.

Выводы

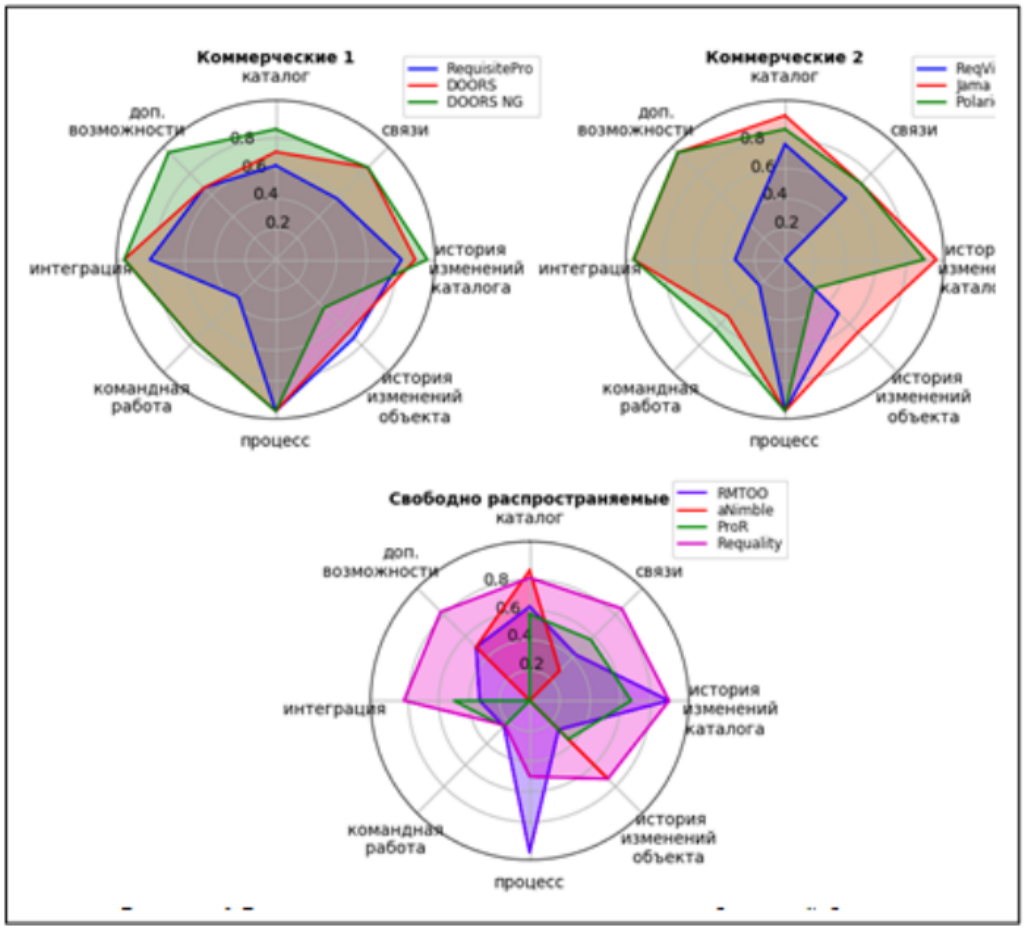


Рис. 1. Результаты оценки инструментов
Fig. 1 Results of tools' evaluation

Выводы

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.

Список литературы

- Горелиц Н. К., Кильдишев Д. С., Хорошилов А. В. Управление требованиями к ответственным системам. Обзор решений. с. 36-45.

Спасибо за внимание.