6.4. *Управление требованиями*

Функциональное описание системы – это не только сценарии поведения, отображение предметной области и графический интерфейс. Сюда входит множество фактов, каждый из которых может отражаться в различных компонентах функционала - своего рода база знаний о проекте. В узком смысле они и являются **требованиями** к системе. В широком смысле в технологический процесс **управления требованиями** входят все перечисленные выше артефакты и процессы работы с ними. Рис. 6-18 изображает компоненты функционального описания, связанные с требованиями.

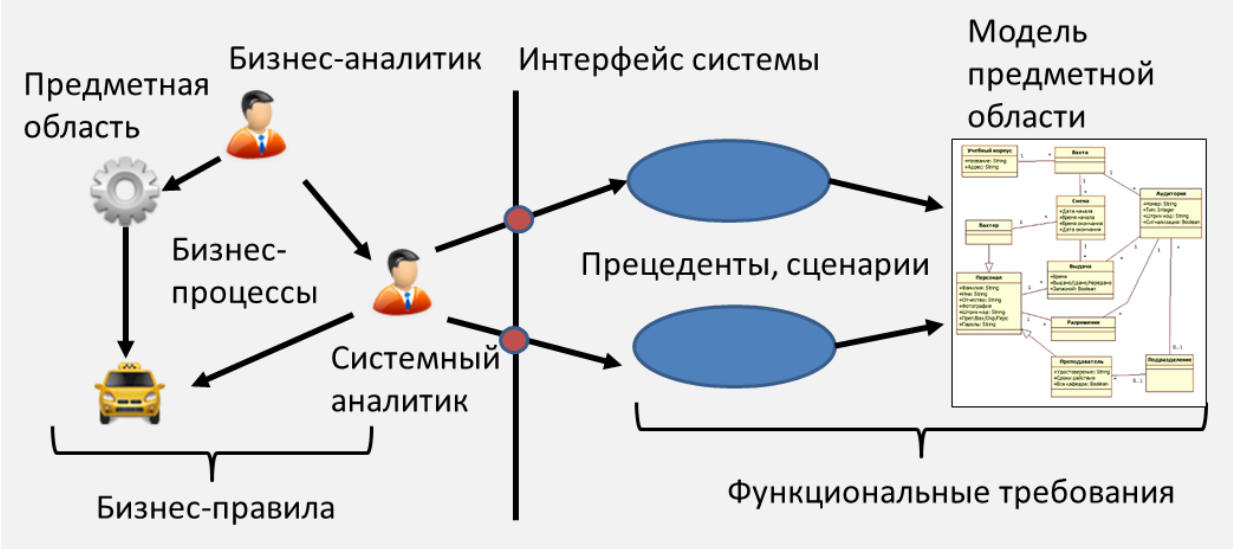

Рис.6-18. Виды и место требований в описании системы

Технологический процесс управления требованиями привязан, в основном, к первым двум фазам жизненного цикла: исследование и развитие. Классикой жанра в этой области является работа Вигерса [6-4], где приводится общепринятая классификация требований (рис.6-19):

- **бизнес-требования** – цели создания системы, получаемый эффект, образ и границы проекта, компоненты бизнес-анализа, создаваемые на этапе исследования проекта (см.6.1,6.2);
- **требования пользователей** – описания возможных действий пользователей и их поведения: варианты использования, сценарии, специфические факты потребительских свойств системы;
- **функциональные требования** – перечень реализуемого функционала с описанием основных параметров и характеристик, факты, касающиеся нескольких прецедентов и сценариев;
- **системные требования** – перечень требований, распространяющихся на всю систему в целом, требования к программной системе (ПС), ПО и аппаратным средствам;
- **бизнес-правила** – законы, постановления, юридические нормы, сложившаяся практика, любые формально выраженные ограничения, отношения и зависимости в предметной области;
- **атрибуты качества** – эффективность, надежность, простота использования и т.п., качественные характеристики и их количественная мера оценки;
- **ограничения,** следующие из бизнес-правил, условия, при которых невозможно выполнение сценариев.

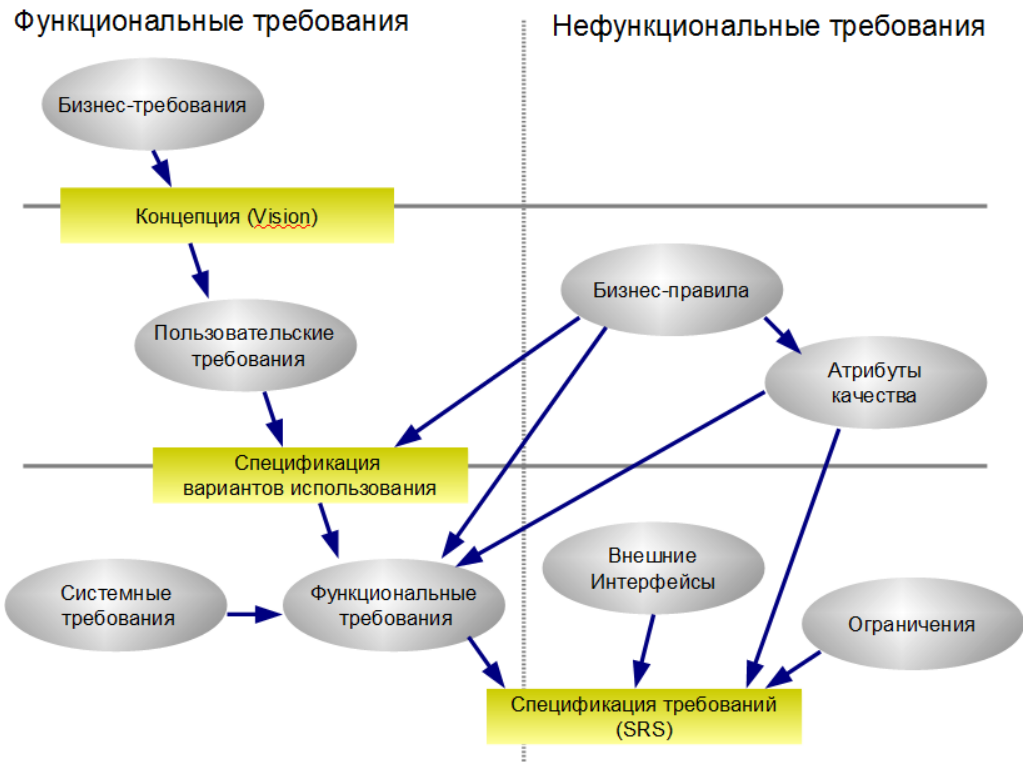

Рис.6-19. Классификация и документы требований по Вигерсу.

Некоторые виды требований с деятельностями в процессах бизнес-анализа и архитектурного проектирования:

- в какой «экологической нише» будет работать проект, в чем его привлекательность, социальная и экономическая значимость, способы монетизации и сроки окупаемости, перспективы - **бизнес-требования;**
- как видится процесс работы с системой, каковы ее потенциальные пользователи и их квалификация, основные сценарии работы – **требования пользователей;**
- параллельно с проработкой предметной области появляются **бизнес-правила,** определяющие законы, по которым живет предметная область;
- на этапе определения архитектуры, т.е. собственно начала проектирования возникают **системные требования** и **атрибуты качества –** потребительские и эксплуатационные свойства системы, обеспечиваемые этой архитектурой;
- на этапе проектирования возникают требования **внешних интерфейсов.**

Поскольку основной задачей системной аналитики является описание и сопровождение функционала системы, то основной группой требований являются **функциональные.**

Бизнес-правила

Бизнес-правила – это своего рода Бейсик предметной области. Все формализуемые положения или ограничения процессов предметной области формулируются в следующих категориях:

- **факты,** инварианты – соотношения, справедливые в течение всего времени функционирования системы;
- **ограничения** - условия, ограничивающие возможность исполнения определенных действий;

- **активаторы операций** – условия, приводящие к активизации или выполнению действий;
- **выводы** - аналог условных конструкций «если - то»;
- **вычисления** - функциональные зависимости, формулы.

Атрибуты качества

Атрибут качества - обобщенное качественное свойство системы. Атрибуты качества не связаны с отдельными элементами функционала, они характеризуют свойства системы в целом как готового продукта. Наиболее распространенные атрибуты качества:

- *доступность* - процент или время доступа с учетом сбоев, отказов, установки;
- *эффективность* - затраты на обеспечение производительности;
- *гибкость* - расширяемость, масштабируемость;
- *целостность* – гарантированная безопасность;
- *надежность* - гарантии работы без сбоев;
- *устойчивость к сбоям, способность к восстановлению*;
- *удобство использования* (useability);
- *удобство эксплуатации*;
- *мобильность, портируемость*;
- *повторное использование кода и данных.*

Атрибуты качества наиболее сложный для формулировки компонент. Всегда хочется, чтобы система была идеальной, как минимум «приятной во всех отношениях». В реальности же всегда приходится балансировать между качеством и затратами на его обеспечение. Поэтому системный аналитик решает здесь несколько проблем:

- оценка важности и актуальности каждого атрибута качества именно для данного варианта системы;
- определение **меры,** в которой это качество может быть описано, протестировано и предъявлено;
- оценка архитектурных затрат на реализацию атрибута качества в данном виде, поскольку многие показатели качества обеспечиваются не столько аппаратными средствами, сколько архитектурными решениями.

Основная ошибка при формулировании атрибутов качества – восприятие их именно как **характеристик качества** системы. Отсюда появляются требования типа «система должна быть проста в использовании», «система должна быть интуитивно понятной». Коренной порок таких требований – их нельзя проверить ввиду именно качественного характера формулируемых свойств. Отсюда главное умение системного аналитика - находить для атрибутов качества **количественную меру,** наиболее соответствующую этому атрибуту, а также оценивать возможные архитектурные затраты на его выполнение на указанном уровне.

Классификация и оценка требований

Прежде всего, требования необходимо оценить и классифицировать по разным категориям. В RUP приняты следующие параметры классификации:

- типы требований:
  - запросы заинтересованных сторон;
  - свойства – требования абстрактного вида;

- программные требования – системные требования к поведению отдельных компонент;
- прецеденты;
- функциональные требования;
- нефункциональные требования;

- категории нефункциональных требований:
  - практичность (Usability);
  - надежность (Reliability);
  - производительность (Performance);
  - поддержка и среда проектирования (Supportability);
  - безопасность;
- характеристики требований:
  - приоритеты: обязательное, желательное, возможное, перспективное;
  - полезность: критическое, важное, полезное;
  - решаемость: проблематичное, выполнимое, тривиальное;
  - трудоемкость: высокая, средняя, малая.

Как бы ни хороша была систематизация требований, она не гарантирует качество процесса. Некачественный процесс разработки требований может проявить себя следующими пороками:

- недостаточное вовлечение пользователей, пропуск классов пользователей и компонент функционала. Как правило, за бортом оказывается вспомогательный, но необходимый для работы функционал, например, отчетность, архивирование, администрирование;
- разрастание требований, неограниченный поток изменений. Необходимо жестко фильтровать требования с точки зрения их полезности, частоты использования, важности и трудоемкости;
- «золочение» продукта – стремление довести до совершенства функционал, сделать его универсальным там, где работают временные решения и заплатки;
- двусмысленность, недостаточная спецификация требований, небрежное планирование – типичные организационные моменты любого планирования.

Известно утверждение, что 80% пользователей используют 20% функционала. В то же время, функционально ограниченный продукт выглядит непривлекательно, даже если этот функционал является редко используемым. Анализ требований должен давать оценку **важности** требований. Оценивать необходимо не только отношение пользователей к факту присутствия функционала, но и к факту его отсутствия (рис.6-20).

<table>
<tr><td rowspan="2"></td><td rowspan="2"></td><td colspan="5">Отсутствие</td></tr>
<tr><td>нравится</td><td colspan="3">ожидаю | все равно | смирюсь</td><td>не нравится</td></tr>
<tr><td rowspan="5">Наличие</td><td>нравится</td><td>???</td><td colspan="3">привлекательный</td><td>линейный</td></tr>
<tr><td>ожидаю</td><td rowspan="4">лишнее</td><td colspan="3" rowspan="3">Безразличный</td><td rowspan="3">обязательный</td></tr>
<tr><td>все равно</td></tr>
<tr><td>смирюсь</td></tr>
<tr><td>не нравится</td><td colspan="3">лишнее</td><td>???</td></tr>
</table>

Рис.6-20. Оценка потребительских качеств требования

Сочетание оценки присутствия и отсутствия функционала дает качественную оценку важности:

- **обязательная** – отсутствие недопустимо, реализация выше некоторого предела не влияет на качество системы;
- **линейная** – пропорциональное увеличение оценки качества;
- **привлекательная** – дополнительная, расширяющая применение системы.

Естественно, что для разных групп пользователей этот показатель важности будет различен. Кроме того, для оценки эффективности реализации требования необходимо учитывать частоту его использования, трудоемкость, что условно математически можно изобразить так: **эффективность = важность * частота использования / трудоемкость.**

Известен также набор обязательных характеристик требований, выраженный аббревиатурой **SMART**:

- **S**pecific – точность и конкретность, подробность и внятность описания требования;
- **M**easurable – измеримость, возможность сопоставления с требованием количественной оценки (метрики);
- **A**chievable – степень достижимости, сложность реализации, трудности формализации;
- **R**elevant – релевантность, значимость для проекта и для исполнителя;
- **T**ime bound/framed – ограниченность во времени, временные рамки для реализации.

Разработка и управление требованиями

С требованиями связаны два рабочих потока – **разработка** и **управление.** Под разработкой понимается процесс извлечения требований, их классификация, наполнение содержанием. Управление – это процесс отслеживания продвижения и реализации требования во всех технологических процессах жизненного цикла.

Перечисленные выше свойства требований и способы их оценки относятся к рабочему потоку разработки требований.

Разработка требований, согласно SWEBOK, включает в себя следующие деятельности:

- извлечение – процесс получения первичных данных о требованиях. Включает в себя опросы пользователей, изучение документации по предметной области, наблюдение за работой пользователей, маркетинговые исследования, работу с прототипами, анализ сценариев работы пользователей, встречи заинтересованных лиц;
- анализ – трансформация требований от формулировки пользователей к виду, готовому к исполнению, включает в себя классификацию, определение атрибутов, сопоставление с моделями проектирования;
- документирование – разработка документов (спецификаций требований), объединяющих разрозненные требования в единое целое;
- валидация требований – проверка полученных требований на корректность и их утверждение. Включает в себя инспекцию (обзор) требований, прототипирование, приемочное тестирование требований.

Анализ и валидация требований не производится на пустом месте. Здесь необходима привязка к тем моделям системы и ее окружения, которые имеют место на этапе жизненного цикла, например, к *модели бизнес-процессов, модели предметной области, модели анализа, описанию архитектуры.* Иначе у системного аналитика отсутствуют основания для проведения анализа требований вообще: как оно соотносится

с реальными бизнес-процессами, как оно может быть архитектурно обосновано, как оно вписывается в текущие сценарии работы в модели анализа.

Управление жизненным циклом требований

Реализация требований происходит в нескольких технологических процессах: собственно управление требованиями, проектирование, конструирование, тестирование, управление конфигурациями. Поэтому основная задача в этом случае – координация и отслеживание деятельностей исполнителей. Сюда входят:

- взаимодействие с технологическим процессом **управления конфигурациями: управление изменениями** и **контроль версий;**
- контроль состояния требования - отслеживание его состояния в процессе реализации;
- отслеживание затрат на реализацию путем введения соответствующих **метрик**, чтобы иметь статистику влияния функциональной составляющей проекта на его стоимость;
- контроль связей с другими требованиями и дисциплинами разработки.

Процесс прохождения требования описывается простой диаграммой состояний (рис.6-21).

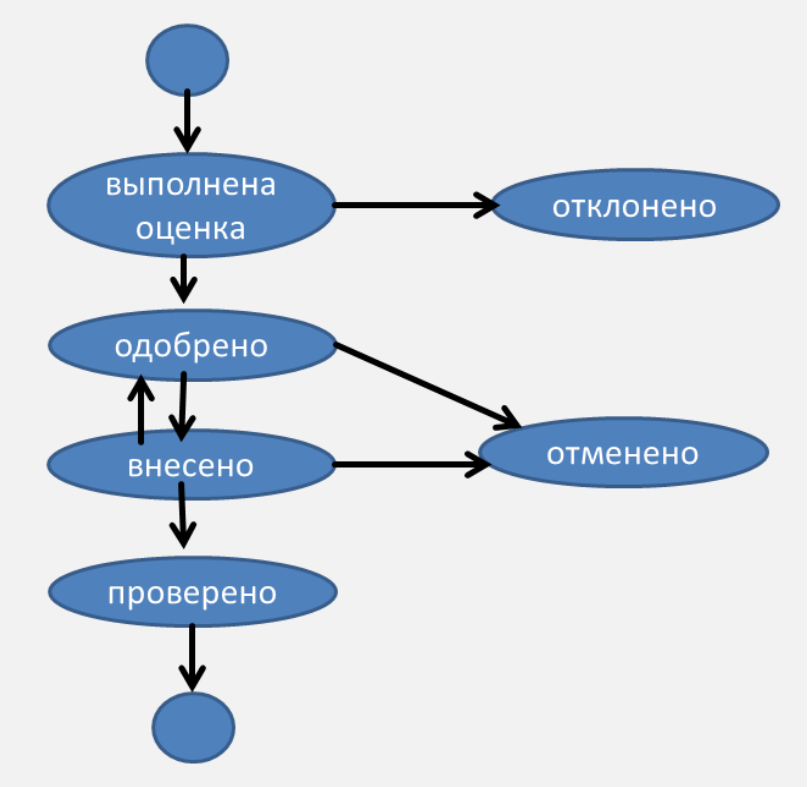

Рис.6-21. Состояния требования в процессе управления

В реализации требований задействованы исполнители (рис.6-22), деятельности которых должны быть скоординированы относительно этого требования. Этот процесс обозначается как **трассируемость требований.**

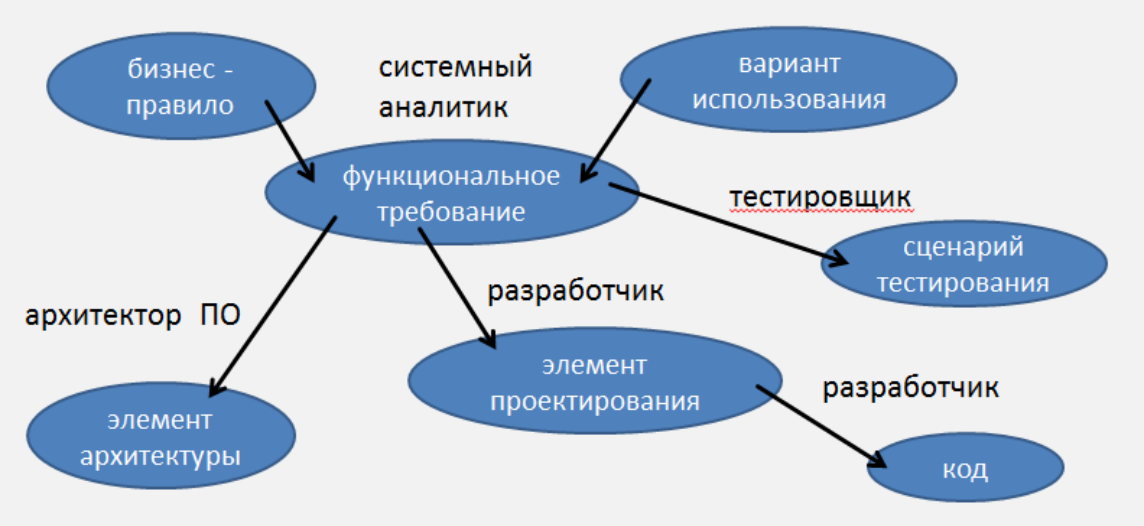

Рис.6-22. Связь требований с деятельностями, ролями и артефактами ЖЦ

Приведенная картинка является идеализированной, в действительности связи требования с другими элементами проекта могут быть вида 1:1, 1:N, N:N. Например, детализация атрибута качества «устойчивость к сбоям» в виде конкретного требования может потребовать внесения изменений в код различных компонент, ведомых разными исполнителями.

Требования**,** как и остальные артефакты проекта, постоянно изменяются в процессе разработки. Поэтому им свойственна версионность. Согласованная версия требований называется **базовой версией (BaseLine).** В процессе работы с требованиями системный аналитик формирует очередную базовую версию, после фиксации которой она становится доступной к использованию другими участниками проекта.

Документирование требований

В п.1.1 обсуждались вопросы самоценности и самодокументируемости кода. Каркас кода или код прототипа могут выступать в качестве спецификаций в фазах развития и построения. Требования в любом случае нуждаются в документировании, если только эта процедура не пущена на самотек. Начнем, как всегда, с недостатков спецификаций. К ним относятся:

- терминологическая неопределенность - отсутствие глоссария или привязки к нему;
- отсутствие представления о классификации требований, подмена категорий и смешение требований;
- фокусировка на деталях пользовательского интерфейса;
- излишнее акцентирование внимания на деталях реализации;
- слабая формализация бизнес-процессов.

Само требование обладает набором атрибутов, используемых при его классификации и в процессах управления, например:

- дата создания требования, автор, ответственный или список заинтересованных лиц, происхождение или источник;
- состояние требования, обоснование требования;
- затрагиваемые подсистемы, номер версии продукта;
- метод проверки или критерий тестирования приемлемости;
- характеристики: приоритет, важность, полезность, решаемость, трудоемкость;

- стабильность - вероятность изменения в будущем.

Форма документа описания требований должна соответствовать его дальнейшему использованию в принятой методологии. Возможные варианты:

- **стандартизованный документ** в тяжеловесных технологиях проектирования ПО. Образцом может служить «Спецификации требований к ПО» по стандарту IEEE 830-1998 [6-4];
- **техническое задание** на внешнее исполнение (аутсорсинг) для системы в целом или ее части. Включает в себя структуру БД, системные требования, бизнес-правила, требования к графическим и внешним программным интерфейсам и форматам, развернутые сценарии. Техническое задание в таком варианте включает в себя результаты фаз исследования и развития для той компоненты, которую оно описывает;
- **иерархический справочник** в гибкой технологии разработки ПО. При наличии модели предметной области, модели классов анализа, описания структуры БД, прототипа графического интерфейса нет особой необходимости расписывать сценарии при наличии взаимопонимания между системным аналитиком и программистами. В конце концов, программист в состоянии из всех перечисленных документов понять *в какой последовательности надо давить кнопки*. В дополнение к этому необходима общий иерархический справочник фактов, касающихся системы. Признаки классификации могут быть разными: особенности системы, варианты использования, режим работы, классы пользователей, стимулы, реакции, классы объектов или функциональной иерархии. Желательно использовать именованные теги вида **A.B.C**, например *правила.рейтинг.сумма***.** Один и тот же факт может присутствовать в разных ветках классификации, в этом случае необходимы взаимные ссылки.

Проект средней руки. Пример фрагмента иерархического справочника для системы контроля рейтинга успеваемости. Перечислены только иерархические теги и наименования требований.

Бизнес-правила

правила.рейтинг.параметры.начало семестра - текущая неделя отсчитывается от этого параметра.
правила.рейтинг.параметры.штраф_за_пропуск - балл, снимаемый за пропуск занятия **(kp).**
правила.рейтинг.параметры.процент_за_показатель - начисляемый или снимаемый % за *показатель качества* исполнения единицы контроля **(dq).**
правила.рейтинг.параметры.процент_за_неделю - процент, снимаемый за одну неделю просрочки. При досрочной сдаче симметрично добавляется **(dw).**
правила.рейтинг.параметры.недель_просрочки - интервал в неделях, в течение которого снимается процент, в дальнейшем - остается на достигнутом уровне.
правила.рейтинг.сумма - сумма баллов рейтинга по обязательным единицам контроля должна бы 100. Для учета индивидуальных внеплановых заданий и бонусов вводятся единицы контроля с весом 0.
правила.рейтинг.пропуск - за пропуск занятия из общей суммы снимается балл рейтинга, указанный в *параметры.рейтинг.параметры.штраф_за_пропуск*.
правила.рейтинг.показатель_качества_исполнения - устанавливается как логическое значение (наличие/отсутствие), предусматривает как увеличение, так и уменьшение балла на фиксированный процент, например, «+сложность», «-оформление».
правила.рейтинг.единица_контроля.норматив -каждой единице контроля назначается нормативное количество баллов **(pi).**

правила.рейтинг.единица_контроля.субъективная - балл в единице контроля ставится преподавателем неформально, по собственным оценкам в пределах установленного значения, либо по критериям, не включенным в систему.
правила.рейтинг.единица_контроля.формальная - балл в единице контроля выставляется системой по формуле расчета.
правила.рейтинг.единица_контроля.формула - N0 - (w - w0)*dw + dq*∑pi.

Атрибуты качества

качество.код.повторное использование - весь код, за исключением слоев управления представлением является независимым от приложения и может быть повторно использован. Контроллер приложения для преподавателя полностью отделен от представления и используется одновременно в desktop и android-приложениях.
качество.доступность.автономно - при отсутствии соединения с сервером приложение преподавателя может работать с локальными копиями рейтингов, которые открывались при работе с сетью. При повторном входе и наличии соединения с сервером внесенные изменения автоматически синхронизируются с сервером БД.
качество.доступность.масштабируемость.сервер_БД - приложение имеет настройки на сервер БД. Сервер БД содержит список независимых однотипных БД.

Функциональные требования

функция.оценка.история – БД должна хранить все записи об изменении оценки и обеспечивать просмотр.
функция.оценка.история.очистка – возможна очистка истории по всему рейтингу с оставлением последней оценки.
функция.рейтинг.имя - имя рейтинга = название предмета <пробел> название группы.
функция.хранилище.объем - для каждого рейтинга должно выводиться количество файлов отчетов и исходников и их объем.
функция.хранилище.скачивание - преподаватель скачивает все файлы отчетов и исходников для выбранного рейтинга одним действием в выбранный каталог, при скачивании файлы из хранилища удаляются.
функция.хранилище.загрузка.разрешение - загрузку файлов в хранилище могут выполнять преподаватель и студент.
функция.хранилище.скачивание.путь - в выбранном каталоге для скачивания создается каталог с именем рейтинга.
функция.куратор.БД.экспорт - куратор может сохранить содержимое БД рейтингов в файл .
функция.куратор.БД.импорт - куратор может загрузить содержимое БД рейтингов из файла.
функция.куратор.БД.инициализация - куратор инициализирует БД рейтингов, старая БД уничтожается, создается новая структура БД.
функция.преподаватель.разрешения - преподаватель имеет список разрешений на просмотр/редактирование рейтингов, устанавливаемый куратором.
функция.куратор.разрешения - куратор в роли преподавателя имеет доступ ко всему списку рейтингов.
функция.куратор.единица_контроля.порядок - порядок следования единиц контроля при выводе редактируется куратором.
функция.куратор.список_группы - список группы студентов может быть загружен из текстового файла, полученного копированием соответствующей формы ЦИУ черезбуфер обмена. Цифровые номера зачетных книжек при чтении файла пропускаются
функция.вывод.студент - при выводе списков и текстовых полей при просмотре и редактировании рейтинга отчество студента не выводится.
функция.единица_контроля.тип - набор типов единиц контроля зашит в приложения и не хранится в БД.

Отчеты

отчеты.формат - необходимо представление всех отчетов в форматах html, pdf и интерактивной форме.
отчеты.интерактивная_форма - вывод отчета в виде экранной формы с возможностью прокрутки и позиционирования к основной форме редактирования через клик по ячейке таблицы с установкой параметров выбранной ячейки.
отчеты.виды – отчеты должны формироваться по группе (только оценки), по единице контроля, по студенту, также необходим отчет по пропускам для всей группы.

Графический интерфейс

GUI.мобильный.отчет - должен быть реализован в интерактивной форме. При ограничении размеров экрана прокрутка должна сопровождаться **выделением цветом** выбранных строки и столбца.
GUI.desktop.отчет.интерактивная_форма - прокрутка таблицы отчета должна производиться с сохранением в панели просмотра заголовков строк и столбцов.
GUI.desktop.подсказки - клик по правой кнопкой мыши по любой кнопке формы сопровождается выводом подсказки о ее функции.
GUI.форма.title - каждая форма в заголовке содержит основные данные о позиционировании – адрес сервера, имя БД, фамилию студента, название предмета.