Глава 5. Проектирование клиент-серверных приложений

Ранее мы рассматривали прикладные протоколы в основном в технологических аспектах, так сказать, изнутри. Теперь посмотрим на этот же предмет снаружи. Любой прикладной протокол является архитектурной компонентой проектируемой системы, поэтому его разработка должна быть органически встроена в процесс проектирования всей системы.

Вопросы жизненного цикла ПО, структуры технологического процесса разработки, их виды, методологии, модели, артефакты, документы и пр. являются предметной областью **программной инженерии.** Тема эта настолько обширна, что требует отдельного рассмотрения. Здесь же мы рассмотрим некоторые элементы жизненного цикла ПО, которые имеют отношение к прикладным протоколам и их проектированию.

Методологии разработки ПО могут быть весьма различными. Но даже если в конкретной методологии формально не присутствуют некоторые элементы, это не значит, что их там нет. Согласно методологии **унифицированного процесса (UP),** имеют место 4 этапа и 9 видов деятельности (дисциплин), составляющих сущность процесса разработки:

Этапы (фазы) – главные результаты:

- исследование – разработка видения проекта, оценка затрат и сроков, принятие решения о начале разработки;
- **разработка –** бизнес-анализ процессов, подлежащих автоматизации, модели бизнес-процессов и предметной области, проектирование функционала системы, графического интерфейса, архитектуры, прототипирование критических компонент - архитектурный прототип;
- реализация – кодирование всех компонент системы, тестирование – бета-версия
- развертывание – установка, обучение, сопровождение – продукт.

Дисциплины:

- **моделирование производства** - бизнес-анализ процессов, подлежащих автоматизации, модель предметной области бизнес-процессов;
- **управление требованиями –** системная аналитика, функционал системы: прецеденты и сценарии, требования, модель предметной области программной системы;
- **анализ и проектирование –** архитектура, модели проектирования – бизнес-объекты, коммуникации, структура программного кода, графический интерфейс;
- конструирование – разработка кода;
- тестирование;
- распространение ПО - установка, обучение, сервис;
- управление проектом - организация, менеджмент;
- управление конфигурацией – сопровождение, управление версиями и изменениями;
- управление средой разработки;

Если отбросить инфраструктурные компоненты (необходимое окружение процесса разработки и управление им, фазы исследования и развертывания), то в самом общем виде можно определить триаду основных элементов процесса разработки:

- **функционал –** описание того, **что** должны делать программная система и **над чем** (внутреннее представление внешних сущностей);
- **архитектура –** неизбыточное многостороннее (5 видов) описание наиболее значимых решений, структуры, поведения, стиля и компоновки программной системы;
- **реализация –** проектирование (на уровне моделей и прототипов) структуры программного кода, соответствующего функционалу и архитектуре, и его конструирование (кодирование и тестирование).

Протокол прикладного уровня – архитектурный компонент, поскольку соединяемые им компоненты (слои, подсистемы) являются элементами архитектуры. С другой стороны, содержательная часть протокола обычно имеет отношение к функционалу системы (например, передача бизнес-объектов в слое бизнес-логики).

Чтобы не привлекать большого объема материала из области программной инженерии, рассмотрим основные аспекты функционального, архитектурного и детального проектирования, имеющего отношение к прикладным протоколам, на примере **системы контроля рейтинга успеваемости** (см.4.4).

5.1. Функциональное проектирование

В разработке функционала имеется два различных вида деятельности, различающиеся, прежде всего, предметом, с которым они работают:

- **бизнес-аналитика** исследует все материальные процессы, к которым имеет отношение проектируемая система и создает их описание. Т.е. предметом бизнес-анализа является материальный мир до (и, возможно, после) включения в его состав программной системы. Бизнес-аналитика обязана дать адекватное представление о процессах (**описание бизнес-процессов**) и предметах (**модель предметной области**), с которым будет взаимодействовать программная система. При этом от бизнес-аналитика не требуется профессионализма в информационных технологий, т.к. он принципиально не занимается вопросами их применения;
- **системная аналитика** имеет дело с взаимодействием бизнес-процессов с программной системой в виде описания ее **функционала,** т.е. фактически с интерфейсом программной системы и определением того, что она должна делать в процессе этого взаимодействия. При этом системный аналитик должен обладать знаниями в области информационных технологий и информатики (точнее, computer science), чтобы оценить архитектурные и технологические возможности реализации того или иного функционала. В системной аналитике можно выделить несколько видов деятельности:
  - **моделирование предметной области,** точнее ее внутреннего представления в программной системе;
  - описание поведения программной системы в виде элементарных взаимодействий – **прецедентов** и подробной их расшифровки – **сценариев;**
  - выявление, систематизация и документирование фактов, касающихся различных свойств и аспектов функционирования системы – **разработка требований.**

Модель предметной области в программной системе

Модель классов предметной области описывает представление в программной системе сущностей предметной области и их постоянных отношений. Тестирование модели состоит в проверке возможности хранения и извлечения данных для основных прецедентов. В дальнейшем модель преобразуется в другие модели ТП анализа и проектирования:

- модель БД серверного приложения;
- модели классов бизнес-объектов.

Важно, что модель классов предметной области описывает содержание (статику) ПС, а не ее поведение (динамику). Поэтому сущности типа **“оформление заказа”, “приобретение билета**” сюда не относятся.

В модель предметной области не включаются сущности и отношения:

- временная сущность, не сохраняемая в ПС. Пример: **отчет,** если он только формируется, и не хранится в архиве отчетов
- отношения между сущностями, если они не сохраняются в ПС, не используются в описании поведения ПС (прецеденты). Пример: зритель, покупающий билет в кассе, взаимодействует с ПС опосредованно через кассира, данные о нем не сохраняются, поэтому отношения “клиент - кассир (касса, смена)” в модели отсутствуют.

При наличии различных вариантов реализации отношения оно должно устанавливаться между сущностью и **интерфейсом** или **абстрактным классом.** Пример: билет может быть продан через кассу, забронирован через интернет, либо оплачен через платежные системы. Отношение устанавливается между сущностями **билет - способ продажи** (интерфейс или абстракция), от которой наследуются сущности **“через кассу”, “бронирование”, “через платежную систему”**

Модель классов предметной области в **системы контроля рейтинга успеваемости** приведена ниже. Она содержит все сущности, контролируемые системой и их устойчивые взаимосвязи. Комментарии к модели:

1. **группа-студент, Дисциплина-единица контроля** - обычная двухуровневая иерархия;
2. **рейтинг** - основная сущность модели, соответствующая паре группа - дисциплина, при создании рейтинга создаются необходимые отношения и ассоциированные с ними данные (классы) **студент ,рейтинг - бригада, единица контроля, рейтинг - срок сдачи, студент, единица контроля, рейтинг - оценка, студент, занятие, рейтинг – пропуск;**
3. с отношением **преподаватель-рейтинг** связан ассоциированный класс **разрешения;**
4. с рейтингом связан набор **параметров**, используемых для его вычисления;
5. загружаемые файлы отчетов и архивов хранятся в самой БД.

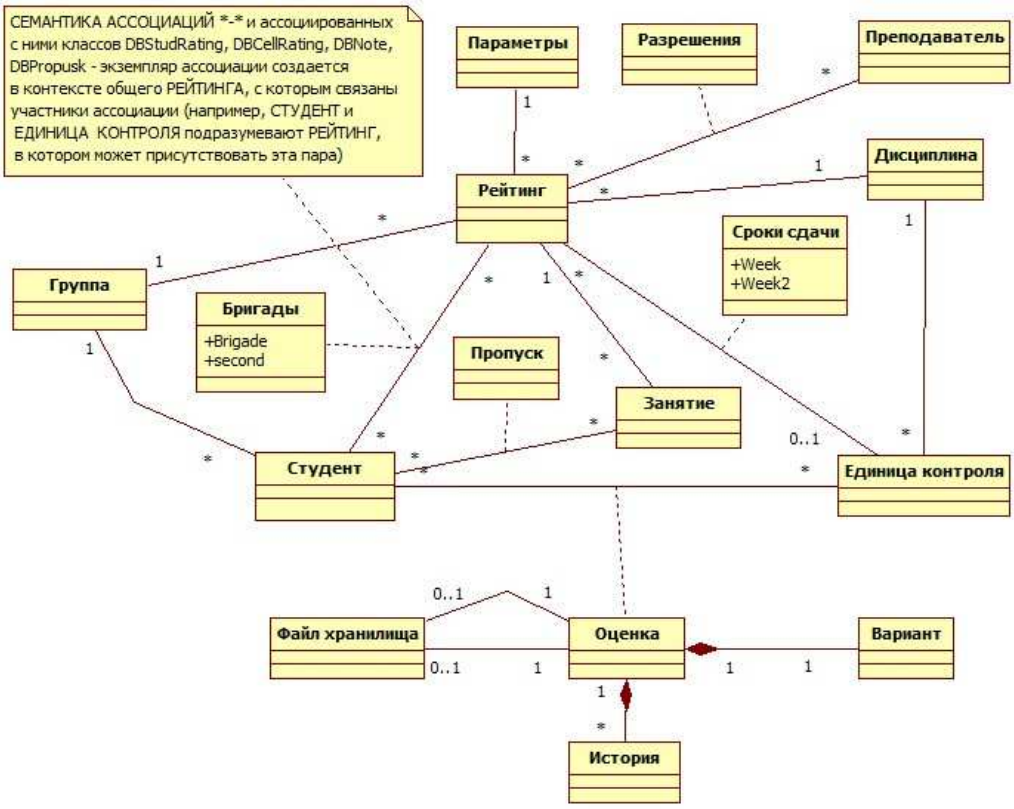

Рис. 5.1. Диаграмма классов предметной области

Прецеденты. Сценарии

Полный функционал системы описывается в виде диаграммы прецедентов, содержание которых расшифровывается в сценариях – описаниях последовательностей действий над объектами модели предметной области.

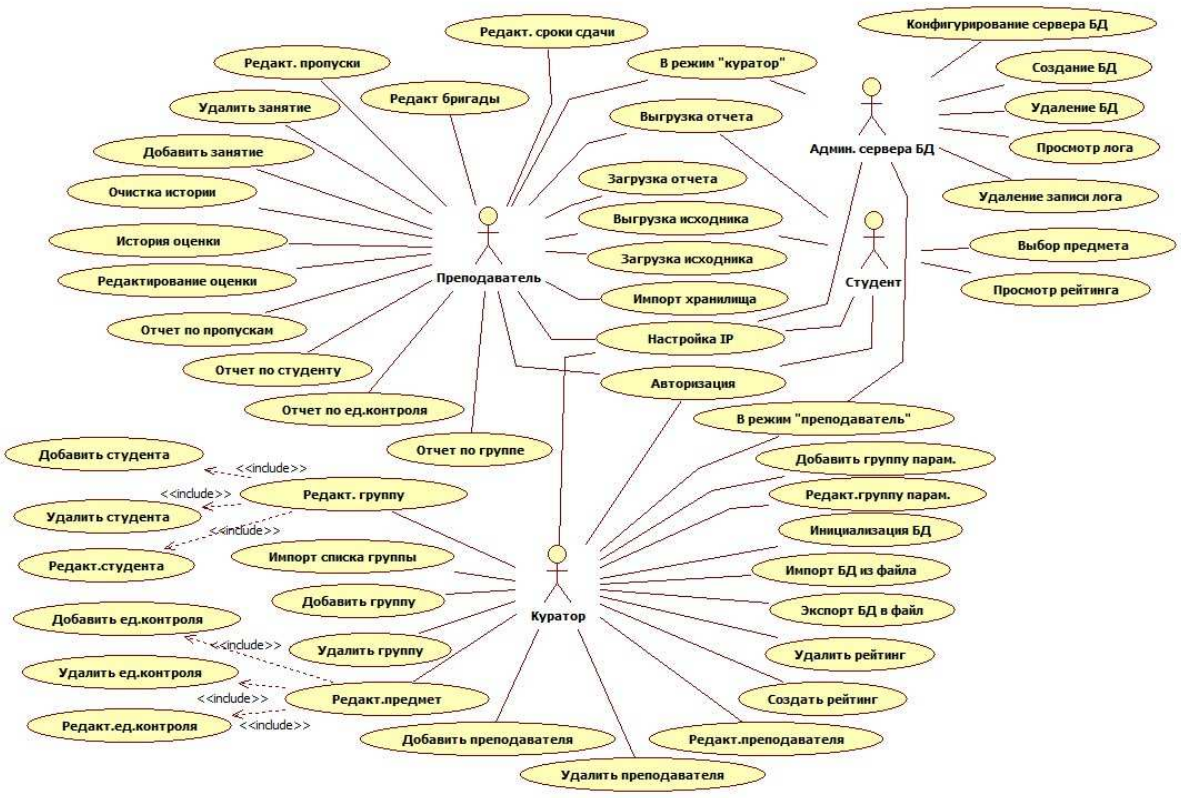

Рис. 5.2. Диаграмма прецедентов

Разработка требований

Требования – альтернативный прецедентам компонент описания функционала. Он ориентирован на систематизацию фактов, касающихся системы в целом, либо нескольких прецедентов. Классификация требований:

- **бизнес-требования** – цели создания системы (организация, заказчик), получаемый эффект, образ и границы проекта
- **требования пользователей** – описания возможных действий пользователей и их поведения (варианты использования (прецеденты), сценарии)
- **функциональные требования** – перечень реализуемого функционала с описанием основных параметров и характеристик (…должен…, может быть элементом нескольких прецедентов, не отраженным текущем уровне детализации сценария)
- **системные требования** – перечень требований, распространяющихся на всю систему в целом (ПС = ПО + оборудование + пользователи)
- **бизнес-правила** – законы, постановления, юридические нормы, сложившаяся практика
- **атрибуты качества** – эффективность, надежность, простота использования и т.п. Основное требование - должны быть сформулированы в виде **количественных оценок,** допускающих проверку и тестирование. Виды атрибутов качества:
    - доступность (% или время доступа с учетом сбоев, отказов, установки)
    - эффективность (затраты на обеспечение производительности)
    - гибкость (расширяемость, масштабируемость)
    - целостность (безопасность)
    - надежность (гарантии работы без сбоев)
    - устойчивость (к сбоям, способность к восстановлению)

- удобство использования (useability)
- удобство эксплуатации
- мобильность, портируемость
- повторное использование (кода, данных)

- **ограничения,** следующие из бизнес-правил, условия, при которых невозможно выполнение прецедентов

Форма документа описания требований обычно соответствует его дальнейшему использованию в жизненном цикле ПО. Возможные варианты:

- **документ** в "тяжеловесных" технологиях проектирования ПО. Выполняется в полном объеме (например, в соответствии с "Спецификация требований к ПО" на основе IEEE Standard 830-1998)

- **техническое задание** на внешнее исполнение (аутсорсинг). Включает в себя структуру БД, системные требования, бизнес-правила, требования к интерфейсам (в т.ч. графическим), развернутые сценарии.

- **иерархический справочник** в гибкой технологии разработки ПО. При наличии других документов нет необходимости расписывать сценарии (прецеденты). Необходима общая иерархическая БД фактов, касающихся ПС. Признаки классификации могут быть разными: особенности системы, варианты использования (use cases), режим работы, классы пользователей, стимулы, реакции, классы объектов или функциональной иерархии. Желательно использовать именованные теги вида **правила.рейтинг.сумма.** Один и тот же факт может присутствовать в разных ветках классификации (необходимы взаимные ссылки).

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.%за_показатель** - начисляемый или снимаемый % за *показатель качества* исполнения единицы контроля (dq)
- **правила.рейтинг.параметры.%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" (pi)
- **правила.рейтинг.единица_контроля.норматив** -каждой единице контроля назначается нормативное количество баллов
- **правила.рейтинг.единица_контроля.субъективная** - балл в единице контроля ставится преподавателем по собственным оценкам (неформально) в пределах

установленного значения (экзамен - 40 баллов), 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 представлений (экранов, View) и является независимым от приложения и может быть повторно использован. Контроллер приложения “преподаватель” полностью отделен от представления и используется 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** - должна хранить все записи об изменении оценки и обеспечивать просмотр
- **функция.оценка.история.очистка** - 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 кол-во файлов отчетов и исходников (в сумме) и их объем.
- **функция.хранилище.скачивание** - преподаватель скачивает все файлы отчетом и исходников для выбранного рейтинга “одним кликом” 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
- **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 режиме “преподаватель” 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 через clipBoard.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 (не хранится в БД)

Отчеты:

- **отчеты.формат** - необходимо представление всех отчетов в форматах html, pdf и интерактивная форма
- **отчеты.интерактивная_форма** - вывод отчета в виде экранной формы с возможностью прокрутки и позиционирования к основной форме через клик по ячейке таблицы с установкой параметров выбранной ячейки (например, студент - единица контроля)
- **отчеты.виды** - по группе (оценки), по единице контроля, по студенту, пропуски (по группе)

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 - ip сервера, имя БД, фамилию студента, название предмета