Лабораторная работа № 1

Анализ требований к системе, планирование проекта.

Цели работы: 1. Рассмотреть этап анализа требований к системе. Познакомиться с актерами, прецедентами и пользовательскими историями.
2. Познакомиться с диаграммами UML, относящимися к данному виду деятельности.
3. Познакомиться с программным обеспечением для создания UML диаграмм.
4. Познакомиться с принципами экстремального планирования.

1. Некоторые теоретические сведения (понятия)

Прецеденты (Варианты использования - Use Cases) - это подробные процедурные описания вариантов использования системы всеми заинтересованными лицами, а также внешними системами, то есть всеми, кто (или что) может рассматриваться как **актеры** (actors). По сути, это своего рода алгоритмы работы с системой с точки зрения внешнего мира. Являются основой функциональных требований к системе, позволяют описывать границы проектируемой системы, ее интерфейс, а затем выступают как основа для тестирования системы заказчиком с помощью приемочных тестов.

Пользовательские истории (stories) – сокращенный вариант прецедентов, без описания побочных ветвей, всех расширений, пред- и пост- условий, менее формальные и более краткие. Используются в XP в основном для выполнения планирования разработки, оценки времени, сложности, распределения обязаностей в коллективе разработчиков, ну и как основа для тестирования тоже (?).

UML – унифицированный язык визуального моделирования, использующий нотацию диаграмм. В основном используется при создании приложений по объектной технологии, но не только. Используется вообще для моделирования ПО, задач, технических систем и пр. Основные типы диаграмм : 1) классов и пакетов, 2) взаимодействия (последовательностей и кооперации), 3) прецедентов, 4) активности (деятельности), 5) состояний, 6) компонентов, 7) размещения. На этапе анализа постановки задачи и требований к системе используют диаграммы прецедентов, диаграммы активности для расшифровки содержания прецедентов, диаграммы состояний для моделирования поведения объектов со сложным состоянием, диаграммы классов для выделения концептуальных сущностей предметной области задачи (по аналогии с ER-диаграммами ?!). В данной работе используются в основном диаграммы прецедентов, активности, может быть – состояний.

Диаграмма прецедентов – предназначена для отображения прецедентов, актеров и их взаимодействия и отношения друг к другу.

Диаграмма активности – предназначена для замещения схем алгоритмов, для отображения какой-либо процедуры (с расширениями для параллельной обработки, отоюражения состояний и пр.)

Диаграмма состояний – вариант нотации для отображении машины состояний или конечного автомата. (См. Теорию автоматов !!!)

Экстремальное планирование – вид деятельности в экстремальном программировании (XP), который предназначен для создания конкретного плана действий по созданию проекта, распределения обязанностей, выделения приоритетов задач, установки

очередности решения задач, измерения скорости разработки. Выполняется постоянно в ходе работы над проектом. Целью является составление плана реализации прецедентов и более мелких задач, составляющих прецеденты, а также технических задач, не видных заказчику, определение количества версий системы, количества итераций, разбиение задач по итерациям, оценки времени выполнения проекта. Первоначально составляется план версий, план итераций первой версии и подробный план на первую итерацию. **Версия** – это продукт, готовый к использованию, с частично реализованной требуемой функциональностью, **Итерация –** время, по истечении которого выдается программа, которую можно передать заказчику для получения от него откликов, замечаний и пр., а также – программа, готовая к концу этого временного интервала. Итерация также может содержать готовую функциональность, но она может быть недостаточной для внедрения данной итерации у заказчика. Вместе с тем она дает представление о направлении проекта, о его успешности и скорости работы команды разработчиков. В ходе планирования разработчик(и) рассматривают пользовательские истории, добавляют к ним обозримые технические задачи, оценивают время выполнения каждой задачи в идеальных неделях и днях и составляют план выпуска версий, примерный план итераций и подробный план итераций (см. примеры).

2. Пример(ы) для изучения

2.1. Пример(ы) описания прецедентов

2.1.1 В качестве такого примера рассмотрим описание прецедентов из книги К.Лармана. Там рассматривается задача создания POS – системы (торгового терминала или точки продаж – Point Of Sale) под названием **NextGen**.

Отрывок из книги Лармана :

"Сначала введем некоторые неформальные определения. *Исполнителем* (actor) будем называть сущность, обладающую поведением, например, человека (идентифицируемого по роли), компьютерную систему или Организацию, например кассира.

Сценарий (scenario) — это специальная последовательность действий или взаимодействий между исполнителями и системой. Его иногда также называют *экземпляром прецедента* (use case instance). Это один конкретный сценарий использования системы либо один проход прецедента, например, сценарий успешной покупки товаров за наличный расчет, либо сценарий неудачного завершения покупки из-за прерванной транзакции по обработке данных кредитной карточки.

Неформально, *прецедент* (use case) — это набор взаимосвязанных успешных и неудачных сценариев, описывающий использование системы исполнителем для решения одной из задач. Например, рассмотрим свободный формат прецедента, включающего некоторые альтернативные сценарии.

Возврат товара (Handle Returns)

Основной успешный сценарий. Покупатель подходит к кассе с товарами, подлежащими возврату. Кассир использует POS-систему для регистрации каждого возвращаемого товара...

Альтернативные сценарии. Если в авторизации кредитной карточки отказано, кассир информирует об этом покупателя и предлагает ему другой способ оплаты покупки.

Если идентификатор товара в системе не обнаружен, система уведомляет об этом кассира и предлагает ему вручную ввести идентификационный код (возможно, штрих-код поврежден и его сложно считать).

Если у системы возникают сложности при коммуникации с внешней системой вычисления налога,...

Несколько другое, но подобное определение прецедента приводится в RUP. **Прецедент** — это набор сценариев использования, в котором каждый экземпляр сценария

представляет собой последовательность действий, выполняемых системой для достижения ощутимого для конкретного исполнителя результата.

Фраза *"ощутимый результат"* несколько туманна, но очень важна, поскольку она указывает на то, что поведение системы должно быть ощутимо для пользователя.

Основное внимание при описании прецедента нужно сконцентрировать на вопросе: "Как использование системы обеспечивает ощутимый ддя? пользователя результат или решает его задачу?", а не на обдумывании системных требований в терминах свойств или функций.

На первый взгляд, требование обеспечения ощутимого результата может показаться очевидным. Однако в сфере программного обеспечения известно множество неудачных проектов, которые провалились из-за невыполнения реальных задач пользователей. К такому отрицательному результату может привести подход к описанию системных требований в виде списка свойств и функций системы, поскольку он не заставляет заинтересованных лиц рассматривать требования в более широком контексте достижения некоторого ощутимого результата или некоторой цели. В отличие от этого подхода, в контексте прецедентов свойства и функции системы ориентированы на достижение цели.

Прецеденты и функциональные требования

Прецеденты — это требования. В основном, это функциональные требования, указывающие на то, что должна делать система. В контексте типов требований, определяемых моделью FURPS+, основное внимание уделяется функциональным требованиям. Однако остальные типы требований тоже могут быть связаны с прецедентами. В рамках UP и большинства других современных методов прецеденты являются основным механизмом, рекомендуемым для **их** определения и исследования. Прецеденты определяют пожелания или соглашения относительно поведения системы.

Итак, прецеденты — это требования (хотя и не все требования). Некоторые считают требованиями только список функций и свойств типа "система должна...". На самом деле это не так. Ключевая идея использования прецедентов как раз и состоит (обычно) в снижении роли списка требований в старом понимании этого слова. Теперь в качестве функциональных требований выступают сами прецеденты. Более подробно этот вопрос рассматривается ниже.

Описания прецедентов — это текстовые документы, а не диаграммы. Моделирование прецедентов — это процесс написания текста, а не рисования. Однако для иллюстрации имен прецедентов и исполнителей, а также их взаимоотношений в UML определены обозначения для диаграммы прецедентов.

Типы и форматы прецедентов

Прецеденты типа "черный ящик" и системные обязанности

Прецеденты типа "черный ящик" (black-box use cases) — это самый типичный и рекомендуемый тип прецедентов. Они не описывают внутреннюю работу системы, ее компоненты или дизайн. Наоборот, системе вменяются некоторые *обязанности* (responsibilities). Этот метафорический термин широко применяется в объектно-ориентированном проектировании: программные элементы имеют обязанности и взаимодействуют с другими элементами со своими обязанностями.

Определяя обязанности системы через прецеденты типа "черный ящик", можно указать, *что* должна делать система (функциональные требования), не расписывая, *как* это делать (не выполняя проектирование). Вообще, термины "анализ" и "проектирование" зачастую сводятся к вопросам "что" и "как". Это важные вопросы в хорошей программной разработке. В процессе анализа требований нужно избегать принятия решений "как", а описывать лишь внешнее поведение системы как черного ящика. Позднее, на этапе проектирования, создается решение, удовлетворяющее разработанной спецификации.

Хорошо : Система регистрирует покупку	**Не очень :** Система записывает сведения о покупке в базу данных. **Или, еще хуже:** система генерирует оператор SQL INSERT для данной продажи
Описание в стиле “черного ящика”	Другой стиль описания

Степень формализации

Прецеденты описываются в различных форматах, в зависимости от потребностей. Помимо типов "черного ящика" и "белого ящика", выделяют несколько степеней формализации описания прецедентов.

- *Сжатый* — аннотация в виде одного абзаца. Обычно она описывает только главный успешный сценарий. Пример такого описания для прецедента Оформление продажи (Process Sale) :

> **Обработка (оформление) продажи (process sale).** Покупатель подходит к кассе с выбранными товарами. Кассир с помощью POS-системы регистрирует каждый товар. Система отображает информацию о каждом наименовании товара и вычисляет общую сумму. Покупатель вводит требуемую информацию; система ее верифицирует и регистрирует. Система выполняет инвентаризацию. Покупатель получает товарный чек и покидает магазин с покупками.

- *Свободный* — неформальный стиль описания. Описание прецедента занимает несколько абзацев и охватывает различные сценарии. Примером такого описания является рассмотренный выше прецедент Возврат товара.
- *Развернутый* — наиболее подробный стиль описания. При таком подходе детально описываются все шаги и варианты развития сценария, а также предусловия и результаты.

Рассмотрим пример развернутого описания прецедента для системы NextGen.

Пример развернутого описания прецедента

Оформление продажи

Развернутые описания прецедентов структурированы и содержат большое количество деталей. Их полезно использовать для углубления понимания целей, задач и требований. Для примера POS-системы NextGen такие описания можно обсуждать на семинарах по определению требований на начальной стадии проекта вместе с системным аналитиком, экспертами предметной области и разработчиками.

Формат usecases.org

Для развернутого описания прецедентов существуют различные шаблоны форматирования. Однако чаще всего используется шаблон, приведенный на Web-узле www.usecases .org. Этот стиль проиллюстрирован в следующем примере.

Этот пример детального описания прецедента относится к рассматриваемой в данной книге системе NextGen и отражает множество типичных элементов и вопросов.

Прецедент П1. Оформление продажи

Основной исполнитель. Кассир.

Заинтересованные лица и их требования

- Кассир. Хочет точно и быстро ввести данные, не допуская ошибок в платеже, поскольку недостача вычитается из его зарплаты.
- Продавец. Хочет получить свои комиссионные от продажи.
- Покупатель. Хочет купить товары и быстро оформить покупку с минимальными усилиями. Хочет получить подтверждение факта покупки для возможного возврата товара.
- Компания. Хочет аккуратно записать транзакцию и удовлетворить интересы покупателя. Хочет удостовериться, что служба авторизации платежей зафиксировала все данные о платеже. Заинтересована в обеспечении устойчивости к сбоям; хочет продолжать регистрировать продажи, даже если серверные компоненты (например, служба удаленной проверки кредитоспособности) недоступны. Хочет автоматически обновлять бухгалтерскую документацию и вести складской учет.

- Государственные налоговые службы. Хотят получать налог от каждой продажи. Таких служб может быть несколько, в том числе национальная и местная.

Предусловия. Кассир идентифицирован и аутентифицирован.

Результаты (постусловия). Данные о продаже сохранены. Налоги корректно вычислены. Бухгалтерские и складские данные обновлены. Комиссионные начислены. Чек сгенерирован. Авторизация платежа выполнена.

Основной успешный сценарий (или основной процесс)

1. Покупатель подходит к кассовому аппарату POS-системы с выбранными товарами.
2. Кассир открывает новую продажу.
3. Кассир вводит идентификатор товара.
4. Система записывает наименование товара и выдает его описание, цену и общую стоимостью Цена вычисляется на основе набора правил.

Кассир повторяет действия, описанные в пп. 3-4, для каждого наименования товара.

5. Система вычисляет общую стоимость покупки с налогом.
6. Кассир сообщает покупателю общую стоимость и предлагает оплатить покупку.
7. Покупатель оплачивает покупку, система обрабатывает платеж.
8. Система регистрирует продажу и отправляет информацию о ней внешней бухгалтерской системе (для обновления бухгалтерских документов и начисления комиссионных) и системе складского учета (для обновления данных).
9. Система выдает товарный чек.
10. Покупатель покидает магазин с чеком и товарами (если он что-то купил).

Расширения (или альтернативные потоки)

*а. При каждом выходе системы из строя.
Для ввода системы в строй и корректной обработки платежа нужно обеспечить восстановление всех транзакций и событий с любого шага сценария.
1. Касссир перезапускает систему, регистрируется и предлагает восстановить предыдущее состояние.
2. Система восстанавливает предыдущее состояние.
2а. Система определяет аномалию, повлекшую сбой.
1. Система уведомляет об ошибке кассира, регистрирует ошибку и переходит в начальное состояние.
2. Кассир начинает новую продажу.

3а. Неправильный идентификатор.
1. Система уведомляет об ошибке и отменяет ввод данного наименования товара.

3б. В рамках одной категории существует несколько разных наименований товара и идентифицировать конкретное наименование не нужно (например, 5 пакетов леденцов).
1. Кассир может ввести идентификатор категории товара и количество единиц.

3-6а. Покупатель просит кассира отменить покупку одного из товаров.
1. Кассир вводит идентификатор товара для удаления из продажи.
2. Система отображает обновленную промежуточную стоимость.

3-6б. Покупатель просит кассира отменить продажу.
1. Кассир отменяет продажу.

3-6в. Кассир приостанавливает продажу.
1. Система записывает сведения о продаже таким образом, чтобы они были доступны с любого терминала POS-системы.

4а. Сгенерированная системой цена товара не устраивает покупателя (например, у него есть дисконтная карта и он рассчитывает на более низкую цену товара).
1. Кассир вводит команду об изменении цены.
2. Система вычисляет новую цену.

5а. Система выявляет сбой при коммуникации с внешней службой вычисления налога.
1. Система перезапускает службу с данного узла POS – системы и продолжает работу.
1а. Система определяет, что служба не перезапускается.
1. Система сигнализирует об ошибке.
2. Кассир сожет вручную вычислить и ввести сумму налога либо отменить продажу.

5б. Покупатель сообщает о положенной ему скидке (например, для сотрудников данного предприятия или постоянных покупателей).
1. Кассир отправляет запрос на скидку.
2. Кассир вводит идентификационные данные покупателя.
3. Система предоставляет сумму скидки, вычисленную на основе дисконтных правил.

…

Специальные требования

- Сенсорный экран с интерфейсом пользователя для большого плоского монитора. Текст должен быть виден с расстояния один метр.
- Отклик службы авторизации в 90% случаев приходит в течении 30 секунд.
- Каким-то образом нужно обеспечить робастное восстановление информации в случае сбоя при доступе к удаленным службам, таким как система складского учета.
- Возможность локализации (представления на различных языках) отображаемого текста.
- Возможность добавления новых бизнес-правил на шагах 3 и 7 в процессе функционирования системы.

Список технологий и типов данных
3а. Идентификатор товара считывается со штрих-кода (при наличии последнего) лазерным сканером или вводится с клавиатуры.
36. Идентификатор товара может определяться по схемам кодирования UPC, EAN, JAN или SKU. 7а. Информация об открытом кредите вводится с помощью считывающего устройства или с клавиатуры. 76. Подпись при оплате чеком ставится на бумажном документе. Однако ожидается, что в течение двух лет большинство покупателей будут требовать цифровые устройства считывания подписи.

Частота использования: почти постоянно.

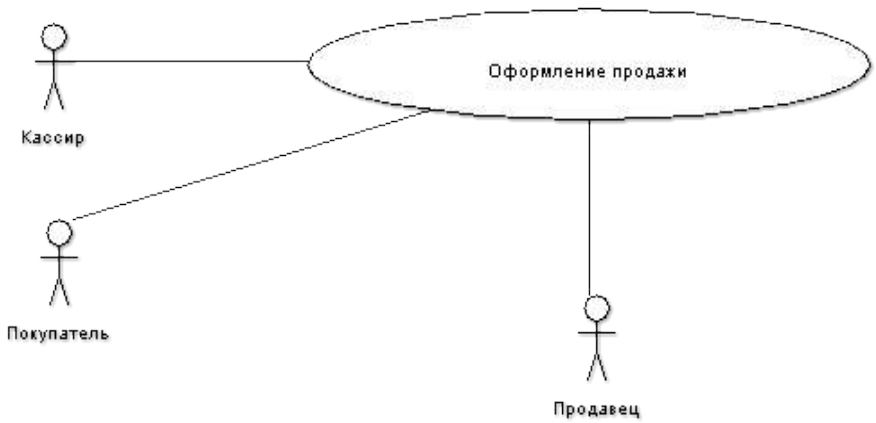
Открытые вопросы
- Изучить законодательство по налогообложению.
- Исследовать вопрос восстановления удаленных служб.
- Какая настройка потребуется для различных типов магазинов.
- Должен ли кассир снимать кассу при выходе из системы.
- Может ли пользователь сам использовать устройство считывания данных с карточки или это должен делать кассир.

... “

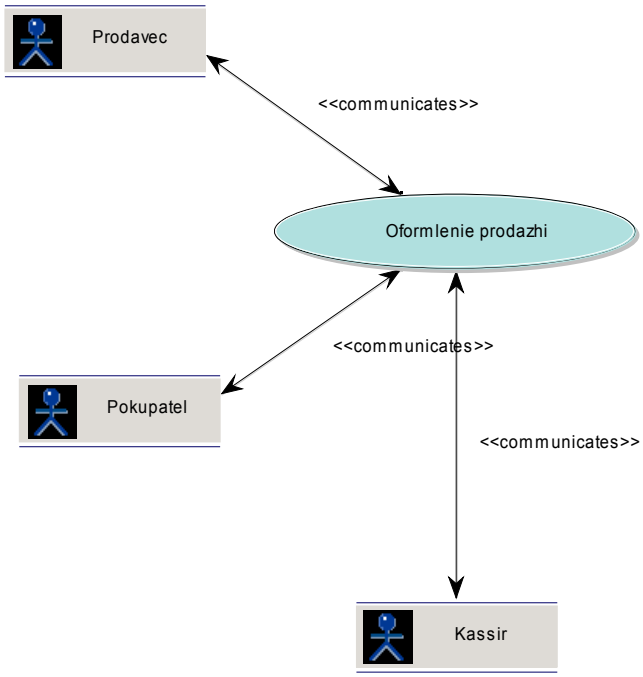
Конец отрывка из книги Лармана.

Тот же прецедент в виде **пользовательской истории** можно трактовать как сжатое описание прецедента, даже еще более сжатое, чтобы оно умещалось на одной небольшой бумажной карточке (типа перфокарты или листа из записной книжки).

Данный прецедент и актер(ы) на диаграмме прецедентов представляются очень просто (использовано ПО ArgoUML) :



Аналогичная диаграмма в ПО Objectif :



2.1.2 Пример начала анализа функциональных требований для программы тестирования (модуль обучающей системы)

Анализ функциональных требований и пользователей системы тестирования

Система тестирования прежде всего требуется следующим заинтересованным лицам :

- обучаемый (студент) ;
- преподаватель (составитель тестов) ;
- преподаватель, осуществляющий контроль за обучением ;
- лицо, осуществляющее контроль за успеваемостью, помимо преподавателя;
- администратор сети и баз данных учебного учреждеения.

Согласно теории прецедентного анализа, помимо собственно физических лиц и их ролей в качестве заинтересованных (действующих) лиц (actors) рассматривают также внешние технические (прежде всего – программные) системы, которые должны взаимодействовать с разрбатываемой системой. С этой точки зрения, конечно, в качестве такой технической системы в данном случае можно рассматривать внешнюю систему управления обучением (LMS, или AOC). В перспективе таким действующим лицом может стать и система управления учебным заведением (например, ее компоненты “Учебный отдел” и “Деканат”).

На начальном этапе создания системы мы можем ограничиться только двумя важными для нас **ролями** действующих лиц :

- студент (тестируемый) и
- администратор (он же преподаватель, он же администратор, он же составитель тестов).

Для взаимодействия с внешней системой обучения воспользуемся обменом через файлы XML в форматах, предусмотренных IMS, поэтому не будем рассматривать отдельное действующее лицо – внешнюю AOC.

Соответственно, основные прецеденты (варианты использования) для нашей системы следующие :

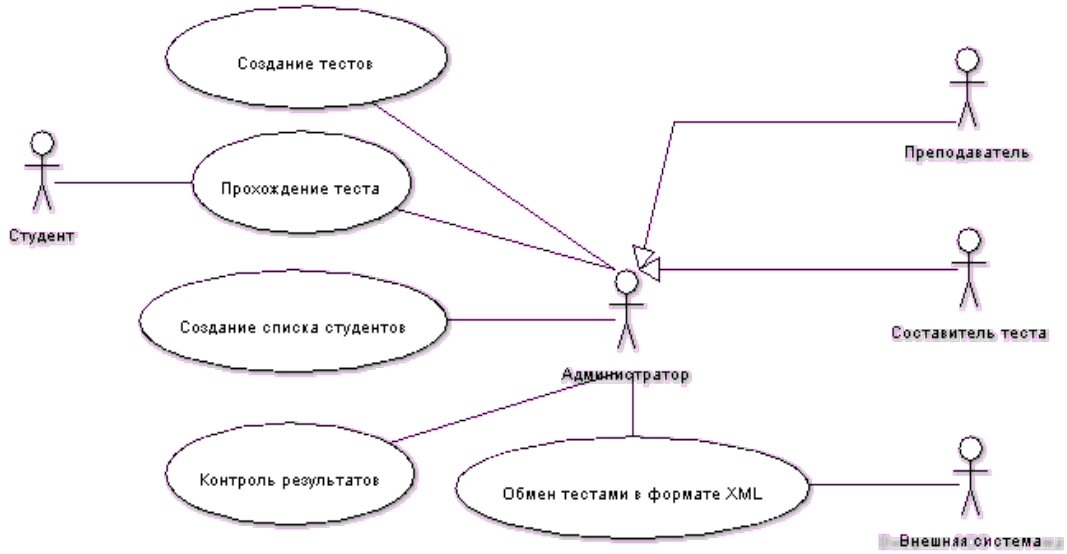
рецеденты для студента :

- П1 – Пройти тестирование;

прецеденты для администратора :

- П2 - Создать / изменить тест;
- П3 - Просмотреть результаты тестирования;
- П4 - Добавить /изменить пользователей и др.

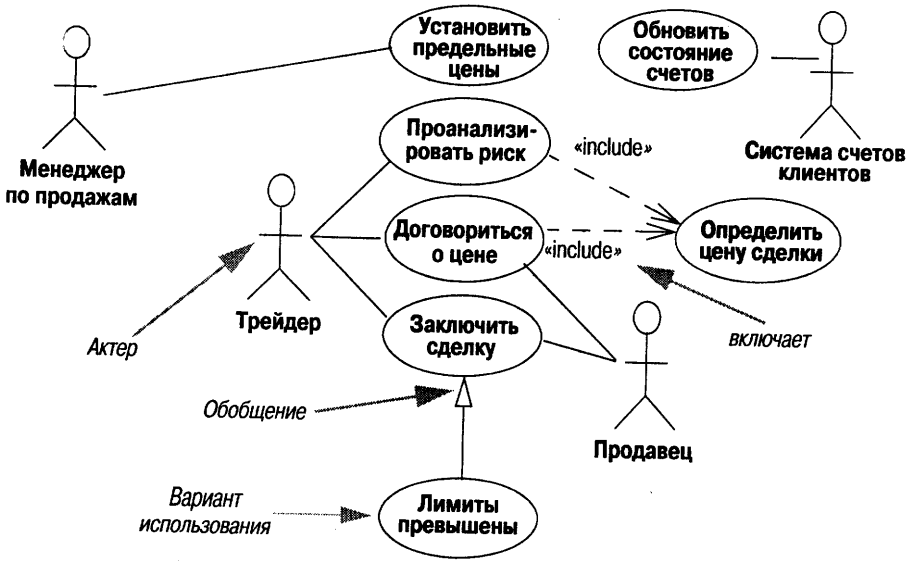
Для иллюстрации анализа прецедентов и заинтересованных лиц рассмотрим диаграмму прецедентов UML :



2.1.3 (Можно привести прецеденты из своих работ !)

2.2 Пример отношений между прецедентами

Здесь можно рассмотреть диаграмму, приводимую Фаулером в своей книге UML Distilled :



Здесь представлена диаграмма для финансовой торговой системы (торговля акциями, например). Актеры : Менеджер, Система счетов, Трейдер, Продавец, а также различные прецеденты и отношения между ними.

Также можно рассмотреть примеры диаграмм прецедентов из книги Леоненкова “Самоучитель UML”

2.3 **Описание нефункциональных требований** (по Ларману)

Перечислим вопросы, которые могут включаться в доп. Спецификацию :

Функциональность

{Имеющая отношение ко многим прецедентам)

Регистрация событий и обработка ошибок

Подключаемые бизнес-правила

Безопасность

Удобство использования

Человеческие факторы

Надежность

Возможность восстановления информации

Производительность

Возможности поддержки

Адаптация системы

Конфигурирование

Ограничения

Приобретаемые компоненты

Бесплатные компоненты на основе открытого кода

Интерфейсы

Важные интерфейсы и аппаратные средства

Программные интерфейсы

Бизнес-правила

Имя	Правило	Возможность изменения	Источник

Вопросы законодательства

Информация из предметной области

Ценовая политика

Обработка платежей по кредитной и дебитной карточке

Вычисление налогов

Идентификаторы товаров (UPC, EAN, SKU, штрих-коды и сканеры)

В дополнительной спецификации содержатся требования, ограничения и другая

информация, не вошедшая в описание прецедентов или словарь терминов, включая атрибуты качества и специальные требования. Заметим, что требования, связанные с прецедентами, должны быть представлены в описаниях прецедентов в разделе "Специальные требования", однако некоторые предпочитают также включать их в дополнительную спецификацию. В дополнительную спецификацию можно включать следующие элементы.

- Требования согласно модели FURPS+ — функциональные, требования к удобству использования, надежности, производительности и возможности поддержки.
- Отчеты.
- Ограничения на аппаратные и программные средства (операционные и сетевые системы и т.д.).
- Ограничения, накладываемые на процесс разработки (например, процесс или средства разработки).
- Другие ограничения проектирования или реализации.
- Международные соглашения (единицы измерения, языки и т.д.).
- Документация (пользовательская, руководство по установке и администрированию) и справочная информация.
- Соглашения о лицензировании или другие юридические соглашения.
- Разбиение на пакеты.
- Стандарты (технические, обеспечения качества и безопасности).
- Физические требования к окружению (например, температурный режим эксплуатации или ограничения на вибрацию).
- Операционные требования (например, способ обработки ошибок, частота архивации).
- Информация из предметной области (например, о полном цикле обработки платежа по кредитной карточке).

2.4. В*и*денье продукта (по Ларману)

“ ...

Введение

Позиционирование

Экономические предпосылки

Формулировка проблемы

Место системы

Заинтересованные лица

Необходимо определить, для кого предназначена система и каковы проблемы заинтересованных лиц.

Демографические особенности рынка...

Заинтересованные лица, не являющиеся пользователями системы...

Пользователи системы...

Основные задачи высокого уровня и проблемы заинтересованных лиц

Необходимо объединить информацию из списка исполнителей и задач, а также из раздела описания прецедентов, отражающего потребности заинтересованных лиц.

Задачи уровня пользователя

Сюда можно включить список исполнителей и их задач, разработанный в процессе моделирования прецедентов, либо более сжатую информацию. Пользователи (и внешние системы) используют данную систему в таких целях.

- *Кассир.* Оформляет продажи, возврат товаров, регистрирует выручку.

• *Системный администратор.* Управляет пользователями, безопасностью и системными таблицами.

• *Менеджер,* Осуществляет запуск и завершает работу системы.

• *Система анализа торговой деятельности.* Анализирует данные о продажах.

• ...

Окружение...

Обзор

Перспективы продукта

Преимущества системы
Подобно перечню исполнителей и их задач, в этой таблице указаны задачи, их решения и преимущества, однако на более высоком уровне, чем при описании прецедентов.
Здесь описывается основное значение и отличительные свойства продукта.

Предположения и зависимости... Стоимость и

ценообразование... Лицензирование и установка...

Основные свойства системы

Как было упомянуто выше, свойства системы описываются сжато путем перечисления основных функций.

Оформление продаж.

Авторизация платежей (по кредитной или дебитной карточке, чеком), Системное администрирование и управление пользователями, безопасностью, таблицами констант и кодов и т.д.

Автоматический переход в автономный режим работы при выходе из строя внешних систем. Транзакции в реальном времени на основе промышленных стандартов с внешними системами, включая бухгалтерскую систему, систему складского учета, учета человеческих ресурсов, вычисления налогов, службы авторизации платежей.

Определение и выполнение настраиваемых бизнес-правил в фиксированных точках выполнения сценариев.

Другие требования и ограничения

Ограничения для процесса проектирования, удобства использования, надежности, производительности, перечень документации и т.д. описаны в дополнительной спецификации и модели прецедентов.

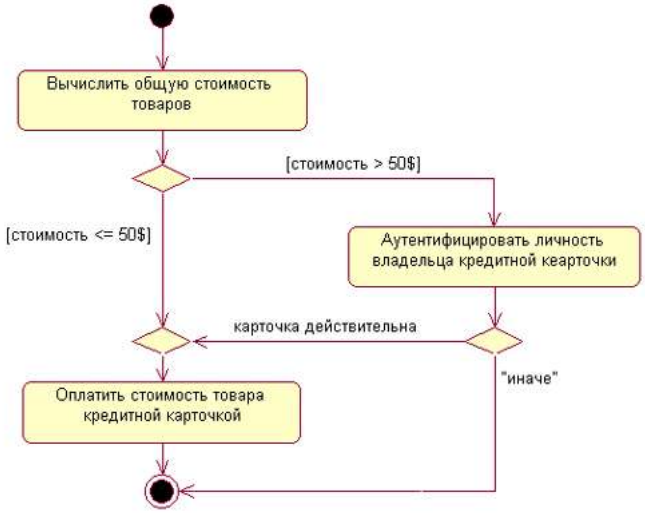
О порядке разработки артефактов говорить неуместно. Различные артефакты, относящиеся к требованиям, создаются параллельно. Тем не менее, можно порекомендовать такую последовательность разработки.

1. Создайте краткий черновой вариант документа "Видение".
2. Идентифицируйте задачи пользователей и соответствующие прецеденты.
3. Опишите некоторые прецеденты и приступите к разработке дополнительной спецификации.
4. На основе полученной информации уточните документ "Видение".

... “

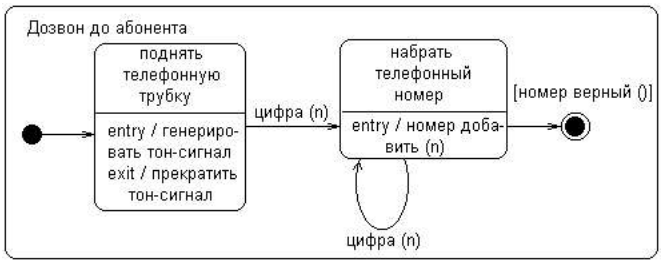
2.5 Пример(ы) диаграмм активности

(Из книги Леоненкова – часть алгоритма обработки покупки по кредитной карточке)



2.6 Пример(ы) диаграмм состояний

(Из книги Леоненкова – диаграммы, описывающие состояния почтовой программы и суперсостояние дозвона до абонента для объекта Телефон – просто для примера)



2.7 Пример планирования (из книги К. Бека)

В книге К. Бека приводится пример планирования для фантастического проекта системы бронирования билетов для космопорта. Для этой системы приводится список пользовательских историй :

Найти самый низкий тариф
Дать пользователю на выбор десять вариантов самых низких тарифов.
Показать информацию о свободных местах в космолетах
Показать информацию о наличии свободных мест в космолетах (в том числе для рейсов с пересадками).
Отсортировать имеющиеся варианты перелетов по комфортности
У пользователя системы должна быть возможность отсортировать имеющуюся информацию о рейсах по времени путешествия, количеству пересадок, местонахождению относительно места назначения и времени прибытия.
Купить билет
Купить билет по кредитной карте. Во время продажи проверить правильность кредитной карты. Помимо этого проверить общие иммиграционные правила (улавианам запрещается посещать Трааль и т. п.).
Создать личную карточку клиента
Сохранить личные данные клиента для возможных последующих обращений (в том числе информацию о кредитной карте, адресе, диетологических и гравитационных особенностях).
Вывести информацию о заказах
Показать все заказы, которые сделал в этой системе клиент.
Отменить заказ
Если клиент отменяет заказ, отменить все виды бронирования: полета, гостиницы и т. д.
Распечатать иммиграционные правила
Распечатать всю информацию, касающуюся въезда и выезда для посещаемых планет (например, «не для вогонов»).
Показать список гостиниц
Показать список гостиниц, расположенных вблизи места назначения.
Показать список гостиниц со свободными номерами
Показать список гостиниц, в которых на период путешествия клиента есть свободные номера.
Реализовать Сложный поиск по гостиницам
Дать пользователю возможность выбрать гостиницу не только по ее местонахождению и времени пребывания на планете. Дополнительные параметры: удобства, уровень обслуживания, цены и рекомендации.
Забронировать номер в гостинице
Забронировать номер в гостинице по кредитной карте. Во время бронирования проверить правильность кредитной карты.
Показать специальные скидки на гостиницы и рейсы
Показать список гостиниц, имеющих специальные соглашения с теми космоли-ниями, услугами которых пользуется клиент. Показать цену и скидки, но только для^тех космолиний, которые на текущий момент активно сотрудничают : нашей системой бронирования.
Предоставить возможность Аренды самолета
Дать клиенту возможность арендовать самолет для передвижения по планете. Связать с данными о прибытии на планету. Внести в личную карточку клиента дополнительные сведения о предпочитаемых им самолетах и пр.

Вот пример распределения задач (пользовательских историй) по итерациям (Iterations) и версиям (Releases) :

Таблица 13.1. План выпуска версий системы бронирования для космических путешествий

Пожелания клиента	Оценка трудозатрат (в идеальных неделях)	Реализовать в итерации #	Реализовать в версии #
Найти самый низкий тариф	3	2	1
Показать информацию о свободных местах в космолетах	2	1	1
Отсортировать имеющиеся варианты перелета с точки зрения комфорта	4		2
Купить билет	2	1	1
Создать личную карточку клиента	4		
Создать простую личную карточку клиента	2	1	1
Создать полную личную карточку клиента	3		
Вывести информацию о заказах	1	2	1
Отменить заказ	2		2
Распечатать иммиграционные правила	4		
Показать список гостиниц	3		2
Показать список гостиниц со свободными номерами	2*		2
Реализовать сложный поиск гостиниц	3		
Забронировать номер в гостинице	1	2	1
Показать специальные скидки на гостиницы и рейсы	3		
Предоставить возможность аренды самолета	3		

* 4, если пожелание «Показать список гостиниц» еще не реализовано.

Как видно из примера, наиболее важные задачи помещены в первую итерацию и версию, остальные отложены до второй версии, некоторые – еще дальше (исходя из сложности и важности). Первая версия расписана по итерациям и задачам. Пример плана для второй итерации (Здесь уже расписано по разработчикам, подзадачам и идеальным дням (можно тоже оформить в виде таблицы !) :

Найти самый низкий тариф

Объект для поиска альтернативного тарифа (задача) — КБ (исполнитель) 2 (идеальные дни)
Найти варианты тарифов по диапазону дат — МФ 1
Обновить данные по портам планеты, чтобы найти альтернативные варианты — КБ 1
Найти варианты тарифов для альтернативных портов — КБ 1
Специальные предложения — основные космолинии — МФ 2
Специальные предложения — недорогие космолинии — РД 3
Пользовательский интерфейс для низких тарифов — РД 1

Вывести информацию о заказах

Простой интерфейс для просмотра заказов — УК 2
Показать подробную информацию но конкретному заказу — РД 2

Забронировать номер в гостинице

Интерфейс для бронирования номера в гостинице — МФ 1
Интерфейс для гостиниц IHAB — МФ 2
Интерфейс для гостиниц HiHat: — МФ 1
Интерфейс для гостиниц Mary's Rote — МФ 1
Интерфейс для гостиниц HillTown — УК 1
Интерфейс для гостиниц BestSouthern — РД 1
Интерфейс для гостиниц WorldStar — УК 1

Показать список гостиниц запрос в IHAB по городам

Запрос в IHAB по гостиницам в указанном городе — УК 2
Пользовательский интерфейс для вывода информации по указанному городу — УК1

Другое

Доводка интерфейса — КА 2
Улучшение производительности сети — КБ 2
Проверка возможности использования IP v84 — КБ 1

2.8 Развернутый пример по основной (сквозной) задаче, которая рассматривается в ходе курса (расчет платежей по кредиту)

Приведем пример постановки задачи для небольшой информационной системы, предназначенной для кредитного отдела банка (или другой кредитной организации).

Краткая характеристика системы (первая постановка задачи от заказчика): программное обеспечение предназначено для расчета платежей по потребительскому кредиту (и другим видам кредитов тоже) и в перспективе – для заключения и ведения кредитных договоров.

Первоначально планируется просто выполнять расчет среднемесячного платежа по фиксированной процентной ставке. Затем, возможно, потребуется расчитывать помесячные выплаты (для тех клиентов, которые захотят платить неравными долями). В перспективе программа должна будет брать данные о процентах из какого-то внешнего источника. На первых порах программа будет использоваться только сотрудниками кредитного отдела, а потом, возможно, и клиентами удаленно (например, с сайта компании). Также, возможно, программа в будущем будет использоваться и при заключении договоров, то есть будет допускать ввод информации о клиенте, сохранять информацию о параметрах его кредита, и даже может быть – о его платежах и истории выплат, выполнять перерасчет его долга при отклонениях от графика погашения, начислять штрафы и прочее. Еще в более дальней перспективе программа может быть связана с межбанковской системой кредитных историй и дополнена алгоритмом проверки клиента. Альтернативный вариант развития системы – ее интеграция с АБС.

Что касается интерфейса программы, то он будет графическим, хотя на первых порах она должна просто считать ! Затем, возможно, потребуется Web интерфейс. Также и с сетевыми возможностями – пусть хотя бы программа будет работать на одном компьютере локально, а там, вполне вероятно, потребуется сделать ее сетевой, и даже, как упоминалось выше, разместить аналогичный модуль (или просто форму) в Интернет. Не исключен в дальней перспективе и мобильный клиент.

Предварительный анализ.

Пробуем выделить всех возможных заинтересованных лиц, а также прецеденты. Итак, прежде всего заинтересованными лицами для этой системы (назовем ее пока CreditLite) являются :

- собственно работник кредитного отдела, работающий с клиентом (Менеджер);
- клиент;
- в будущем – удаленный клиент;
- еще в более дальнем будущем – мобильный клиент;
- внешняя база данных, где хранятся проценты (или как далее – АБС);
- возможно – АБС (как источник или приемник данных, либо – как внешняя система);
- возможно – межбанковская система (в будущем);
- любой интересующийся (потенциальный клиент);
- клиент, заключивший договор и выплачивающий кредит и пр.

В принципе, если проанализировать этот список, то ряд заинтересованных лиц являются (по крайней мере на первых порах) просто более специфическими ролями базовых заинтересованных лиц :

- менеджера;
- клиента;
- внешней информационной системы.

Обратимся к прецедентам. Из неформального описания функций системы можно выделить следующие прецеденты :

П1 Расчет платежа по кредиту (за определенный период).
П1.1 Расчет среднемесячного платежа по кредиту
П1.2 Расчет помесячных платежей

П2 Заключение договора о кредите
П3 Перерасчет кредита (суммы основного долга, процентов и пр.) после погашения его части с учетом графика

П4 Начисление штрафов за просрочку
П5 Перерасчет кредита с учетом штрафов

П6 Загрузка кредитной истории
П7 Выгрузка кредитной истории
П8 Анализ анкеты клиента
П9 Анализ анкеты клиента с учетом кредитных историй
П10 Выгрузка информации о договоре в АБС и пр.

Остальное (например, превращение системы в АБС :)) пока находится даже за пределами фантазий самого заказчика.

Можно составить краткое (сжатое) описание каждого прецедента (это могут сделать сами студенты :)) :

П1 Расчет платежа по кредиту. Системе сообщается месячная (либо годовая) процентная ставка, сумма кредита и количество месяцев (срок), на который он берется. Система рассчитывает сумму начисляемых за все время пользования кредитом процентов с учетом ежемесячного погашения кредита равными долями и общую сумму выплат. (Для этого прецедента интересно составить диаграмму деятельности, то есть фактически – алгоритм !)

П1.1 Расчет среднемесячного платежа по кредиту – практически это то же самое, но система делит общую сумму выплат на количество месяцев (можно не выделять в качестве отдельного прецедента).

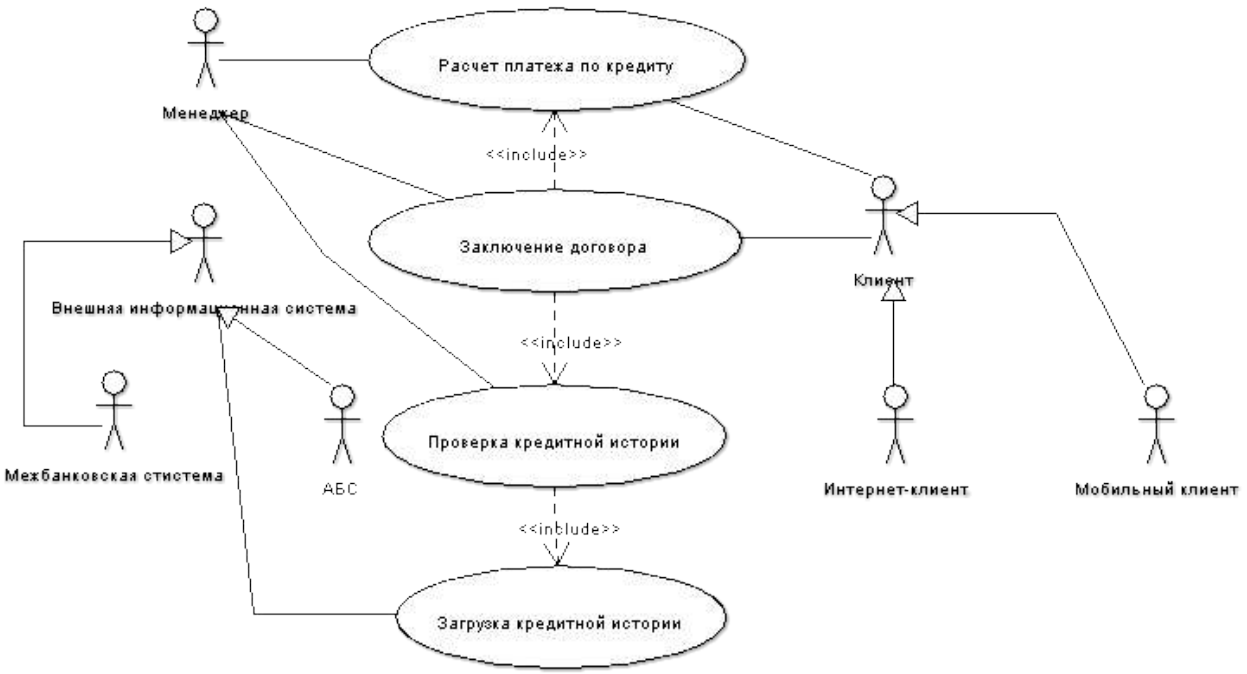
П1.2 Расчет помесячных платежей – выполняется почти такой же расчет, как и в предыдущем прецеденте, но – неравными долями (клиент выплачивает сумму кредита равными долями, а процент за кредит уменьшается от месяца к месяцу)

(Последние два прецедента не очень похожи на настоящие прецеденты, они почти не имеют собственного значения и являются подпрецедентами, либо же вообще их нужно объединить с первым).

П2 Заключение договора о кредите – Клиент обращается в кр. Отдел и сообщает сумму и срок кредита. Менеджер производит расчет, при этом задействуются прецеденты П1, П1.1, П1.2. Если Клиента устраивают условия кредита, он заполняет анкету, Менеджер анализирует ее (либо в будущем активируются прецеденты П6 и П8-П9), принимает решение о выдаче кредита (если да), затем вводит параметры договора в систему (возможно, при этом задействуется прецедент П10). - Для этого прецедента рекомендуется рассмотреть диаграмму активности, диаграмму взаимодействия, а сначала – рассмотреть подробное описание прецедента с альтернативными ветвями и пр. Это “настоящий” прецедент, который характеризуется явным значимым результатом и для Менеджера, и для Клиента, и для Кредитной организации.

и т.д. ...

На диаграмме прецедентов ниже представлены некоторые прецеденты, актеры и отношения между ними.



Теперь проанализируем (вместе с заказчиком) приоритеты прецедентов, риски и сроки для того, чтобы составить первоначальный план версий. Трудозатраты на каждое пожелание пользователя (прецедент, историю) оценим в **идеальных неделях**.

Идеальная неделя (день) - условная единица измерения времени, а вернее сказать – сложности задачи при планированиив XP. Под одним идеальным днем понимается 8-часовой рабочий день, в течение которого разработчик занимается только решением конкретной задачи (для которой оценивается сложность), не отвлекается на посторонние дела, он чувствует себя хорошо, работа “идет”, одним словом, разработчик работает со своей нормальной производительностью, близкой к максимальной. Идеальная неделя – это 5 рабочих (идеальных) дней. Одна итерация занимает от 1 до 4 недель (в зависимости от проекта, условий, команды, но в одном проекте продолжительность итерации фиксированная). Одна версия включает обычно минимум 2 итерации.

Список прецедентов дополним существенными пожеланиями пользователя по поводу свойств системы, такими как сетевые возможности, Web-интерфейс, доступ через Интернет, мобильный доступ, чтение условий кредитования из внешнего источника и пр.

Очередность реализации пожеланий определяется совместно заказчиком и разработчиком исходя из приоритетов задач, их объемов, рисков. Риск в данном случае – это некоторая угроза для проекта. К рискам можно отнести : неустойчивость работы системы (частые сбои, ошибки), некорректные реализации алгоритмов или некорректные алгоритмы, приводящие к ошибкам в расчетах, низкую производительность системы, технические трудности в реализации программ из-за отсутствия опыта работы с конкретным средством разработки, сторонней библиотекой, технологией, протоколом и пр.

В данном случае подобный вид первоначального плана версий объясняется прежде всего пожеланиями заказчика. Он определил, что его прежде всего интересует собственно расчет платежей, затем ему зачем-то нужно обязательно иметь сетевую версию, а уже потом - перерасчет платежей (допустим, ему кажется, что в основном все клиенты будут платить во-время). Также почему-то заказчик предпочел сначала перенести проект в Интернет, а потом уже расширять его функциональность (возможно, он еще не решил, стоит ли вообще это делать). Ну, тем лучше для нас – это позволит нам сосредоточиться на важных технических деталях. :) .

Пожелания клиента	Оценка трудозатрат, в ид. неделях	Реализовать в итерации #	Реализовать в версии #
Расчет платежа по кредиту (за определенный период) и среднемесячного платежа.	1	1	1
Расчет помесячных платежей	1	2	1
Заключение договора о кредите (просто фиксация данных о клиенте и параметрах договора, без анализа анкеты и кредитной истории)	1	5	2
Перерасчет кредита	2	4	2
Перерасчет кредита с учетом штрафов	1	5	2
Загрузка кредитной истории	2		
Выгрузка кредитной истории	2		
Анализ анкеты клиента	2		
Анализ анкеты клиента с учетом кредитных историй	2		
Заключение договора с анализом анкеты и кредитной истории	1		
Выгрузка информации о договоре в АБС	2		
Чтение условий кредитования из БД	1	2	1
Сетевая версия (для ЛВС)	2	3	2
Веб-интерфейс	2	6	3
Доступ через Интернет	2	7	3
Мобильный доступ	3		

Заметим, что прецеденты немного изменены, в частности, прецедент заключения договоров разбит на два, причем вторая часть отнесена на неопределенный срок. Некторые пожелания вообще не фигурируют в обозримом списке итераций. Это связано с теми оценками времени, которые сделал разработчик для каждой задачи. Поскольку итерация имеет фиксированный размер (2 идеальные недели в данном случае), часть задач просто не уместилась в обозримый план из 14 идеальных недель (это примерно 3 идеальных месяца). Заметим также, что реально на реализацию проекта даже в таком виде уйдет заведомо больше 3 месяцев, так как недели и дни не могут быть все идеальными. Так что можно смело умножить 3 месяца на 2 и получим полгода ! Это не очень устраивает заказчика, так как

через полгода он хотел бы уже иметь все задачи решенными, но ему приходится согласиться с оценками разработчика (хотя в душе он надеется на лучшее). Разработчик тоже надеется на некоторое ускорение работы по мере хода проекта (поэтому, например, он выделил на первую задачу трудоемкостью в 1 неделю целую итерацию, а затем стал планировать по две таких задачи в итерацию). Оценки трудоемкости выбираются разработчиком на основании его предыдущего опыта и представлений о задаче на момент составления плана. Однако в будущем его оценки изменятся в зависимости от той информации, которую он будет получать в ходе проекта (как от заказчика, так и от самого проекта, то есть как бы от себя) !

Медленная скорость работы разработчика при реализации первой версии объясняется еще и тем, что он :

- хочет сосредоточиться на создании корректного ключевого алгоритма и его реализации;
- хочет создать инфраструктуру проекта (папки, документы, диаграммы, проект для конкретного средства разработки и пр.);
- хочет освоить новые для него технологии разработки (XP, модульное тестирование, рефакторинг, шаблоны, диаграммы UML и пр.);
- вынужден осваивать не очень знакомые ему особенности бесплатной библиотеки GUI и особенности работы с бесплатным средством разработки, так как это одно из нефункциональных требований заказчика.

Несмотря на эти веские причины, конечно, трудно представить, то в реальности программисту на реализацию двух не очень сложных алгоритмов потребуется не два дня, а две недели (то есть 10 дней). Но не стоит забывать, что речь идет об учебном проекте ! И, кроме того, еще раз вспомним о понятии идеальных дней (в реальности их приходится умножать не на 2, а на 3 или даже 4; лишь бы не на 10 и не на бесконечность :)).

Остановимся на более подробном плане первой итерации.

Итак, на первую итерацию запланирована всего одна задача (прецедент) : Расчет платежа по кредиту (за определенный период) и среднемесячного платежа. Нужно расшифровать эту задачу и выяснить, что нужно сделать для ее реализации уже в масштабе идеальных дней. Допустим, что заказчик пока согласился, что результатом первой итерации будет программа, не имеющая графического интерфейса (то есть у нее интерфейс командной строки), она не будет использовать БД или другие источники данных, а будет использовать фиксированный процент, который сообщил заказчик. При этом такая программка уже будет что-то считать, а значит, будет полезна заказчику (то есть за нее можно уже и денег запросить :)). Что нужно сделать, чтобы создать такую программу ? Ну, во-первых, нужно разработать алгоритм. Во-вторых, поскольку программист решил работать в рамках ООП, то нужно создать класс, инкапсулирующий этот алгоритм (или алгоритмы). В третьих, поскольку разработчик решил работать по канонам XP, ему нужно создать тесты для тестирования своего класса, причем двух видов : приемочные тесты для оценки правильности расчета (их он составляет с помощью заказчика, или вообще получает от заказчика) и модульные тесты для реализации самого класса и его рефакторинга. Поскольку программист не очень знаком с этими видами деятельности, принятыми в XP (тестирование, рефакторинг, разработка через тестирование), то он пока решает запланировать каждый вид деятельности отдельно (хотя на самом деле эти виды деятельности выполняются последовательно – параллельно и составляют собственно процесс разработки классов в XP). Кроме того, разработчик хочет отвести некоторое время на проектирование всей разработки (не очень большое, как принято в XP), хотя бы на архитектурное. Ну и, как упоминалось выше, ему нужно время на создание начальной инфраструктуры проекта, необходимых документов, диаграмм, других артефактов, и время на изучение средств разработчи и библиотек.

Получаем следующий примерный план итерации (P – разработчик):

Задача	Трудозатраты в идеальных днях	Ответ-ственный
Расчет платежа по кредиту (за определенный период) и среднемесячного платежа	5 – 10 (5 в запасе)	P
1. Составление плана итерации	0,5	
2. Создание проекта и инфраструктуры	1	
3. Создание эскизного проекта архитектуры	0,5	
4. Изучение особенностей средства разработки	1	
5. Изучение оболочки для тестирования	1	
6. Создание приемочного теста	0,5	
7. Создание модульных тестов	1	
8. Разработка класса с алгоритмом расчета	2	
9. Разработка приложения	1	
10. Оценка скорости разработки, планирование (в том числе текущее)	0,5	
Итого :	**9**	

В общем случае, когда разработчик не один, последний столбец реально используется и позволяет примерно поровну распределить задачи.

В принципе, целесообразно также составлять текущие планы на каждый день. Там прописываются конкретные дела (создание артефактов, элементов интерфеса, таблиц и запросов к БД, классов и методов, или, как мы увидим в дальнейшем, необходимые модульные тесты). Это помогает следить за скоростью разработки и даже несколько повышать ее. В таком плане по итогам дня вычеркиваются выполненные пункты и можно оценить реальную скорость.

3. Порядок выполнения работы

I Анализ требований

1. Изучить представленный пример(ы) постановки задачи, прецедентов, требований, а также диаграммы UML
2. **Создать диаграмму прецедентов и другие диаграммы из примеров в программных средствах ArgoUML и ObjectiF.**
3. ***Создать собственный прецедент и собственные диаграммы по вариантам.***

II Планирование

1. Изучить примеры планирования.
2. ***Создать собственный план по вариантам.***

III Семестровая работа

1. Сформулировать краткую постановку задачи для своей системы (В***и***денье).
2. Разработать список пользовательских историй.
3. Разработать 1- 2 прецедента (подробно).
4. Создать диаграмму прецедентов UML.
5. Разработать нефункциональные требования к системе.
6. Составить список рисков и соотнести со списком прецедентов / историй.
7. Составить план версий, итераций (для последующей корректировки).

4. Варианты заданий (10 шт)

4.1. Задания для составления прецедентов, диаграмм, планов (типовые проекты)

Составить список прецедентов и актеров, расписать 1-2 прецедента подробнее, составить план версий и итераций. Например :

1. Интернет – магазин.
2. Информационная система для учета кадров на небольшом предприятии
3. Информационная система для расчета зарплаты на небольшом предприятии
4. Утилита мониторинга сети (для администратора сети)
5. Обучающая программа с функцией контроля знаний (это основная функция !) (фактически кусок этого варианта рассмотрен выше в п. 2.7)
6. Биллинговая система для учета интернет-трафика.
7. Веб - сервер
8. Почтовый сервер
9. Текстовый редактор (потом – текстовый процессор ?!)
10. Программа моделирования цифровых схем
11. Информационная система складского учета (учет прихода и ухода товаров, категории, поиск и пр.)
12. АРМ менеджера компьютерной фирмы (не бухгалтерия !). - Быстрая прикидка конфигураций, выписка счетов, отслеживание счетов их оплат и пр.
13. АРМ начальника отдела техподдержки (учет ремонтов).
14. Графический редактор (векторная графика).
15. Графический редактор (растровая графика).