МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ

Ордена Трудового Красного Знамени федеральное государственное бюджетное образовательное учреждение высшего образования

**«Московский технический университет связи и информатики»**

****

Кафедра «Сетевые информационные технологии и сервисы»

**Учебно-методическое пособие для выполнения курсового проектирования**

**по дисциплине**

**Методы и средства проектирования информационных систем и технологий**

Москва 2024

План УМД на 2023/24 уч.г.

**Задание на курсовое проектирование**

В каждом из предложенных вариантов необходимо выполнить проектирование информационной системы на основе унифицированного процесса с распределением обязанностей с применением шаблонов проектирования GRASP. Для этого рекомендуется использовать CASE-средство Visual Paradigm (<https://www.visual-paradigm.com/download/>).

Процесс включает в себя:

* Выделение прецедентов (5-7 прецедентов);
* Описание нефункциональных требований;
* Моделирование предметной области;
* Создание системных диаграмм последовательностей;
* Описание операций и реализацию прецедентов.

В данном пособии приводится пример проектирования информационной системы поддержки проведения экзамена. Однако разобрано создание только одного артефакта каждого типа, встречающегося в проекте. В процессе выполнения курсового проектирования необходимо создавать и описывать в отчете такое количество артефактов каждого типа, которое требуется для адекватного представления всех требований, сформулированных в задании на проектирование. Номер варианта задания выбирается в соответствии с порядковым номером студента в списке группы.

**Критерии оценки курсового проекта**

**Оценка в процессе защиты:**

Для получения оценки «отлично» оценки студент должен:

1. Уметь аргументировать необходимость включения прецедентов в проект, опираясь на текстовое описание задания.
2. Уметь выявлять и классифицировать внешних исполнителей.
3. Обладать знаниями обозначений языка UML, требуемых для создания диаграмм классов, последовательностей, взаимодействия и прецедентов.
4. Знать основные артефакты унифицированного процесса проектирования и уметь объяснить их взаимосвязь.
5. Понимать и уметь применять основные шаблоны проектирования на основе распределения обязанностей.

Для получения оценки «хорошо» необходимо:

* Соответствовать пунктам а, b, c и d из раздела 1.
* Знать названия всех шаблонов проектирования на основе распределения обязанностей.

Для получения оценки «удовлетворительно» необходимо:

* Соответствовать пунктам а, b и с из раздела 1.
* Иметь знания основных принципов гибкого интерактивного проектирования.

**Оценка выполнения проектирования**

Для получения оценки «отлично», студенту необходимо:

1. Применить для реализации прецедентов первые пять шаблонов GRASP, такие как: Информационный эксперт (Information Expert), Создатель (Creator), Контроллер (Controller), Слабая связанность (Low Coupling) и Сильное Сцепление (High Cohesion). А также дать объяснение использования каждого шаблона в контексте конкретной задачи.
2. Построить модель предметной области и показать ее развитие на примерах прецедентов.
3. Построить модель прецедентов с учетом всех системных требований, изложенных в индивидуальном задании.
4. Применить минимум одну диаграмму прецедентов, одну диаграмму концептуальных классов предметной области, одну диаграмму классов проектирования, не менее трех описаний прецедентов, не менее трех системных диаграмм последовательностей, не менее трех описаний операций, не менее трех диаграмм взаимодействия объектов и словарь терминов.
5. Все прецеденты, выделенные в проектируемой системе, должны соответствовать задачам внешних основных исполнителей.

2. Для получения оценки «хорошо», студенту необходимо:

* Обязательное выполнить пункты b, c, d, e раздела 1.
* Обосновать применения хотя бы одного шаблона проектирования на основе распределения обязанностей.

3. Для получения оценки «удовлетворительно», студенту необходимо:

* Выполнить задания d и e из раздела 1.

**Итоговая оценка**

Оценка за курсовое проектирование определяется двумя компонентами, позволяющими оценить академическую деятельность в течение семестра и понимания предмета на момент защиты курсового проекта.

Неудовлетворительные результаты по любому из компонентов приводят к общей неудовлетворительной оценке за весь проект.

Однако отличная итоговая оценка возможна лишь при высоких результатах за оба компонента.

С учетом этого, наивысшей итоговой оценкой за курсовое проектирование будет считаться минимальная из оценок, полученных студентом за каждую из компонент.

**Пример: Проектирование системы поддержки проведения экзамена**

**1. Предварительное описание**

Система обеспечивает автоматическую выдачу билетов с использованием точки доступа и мобильных устройств, оснащенных модулем беспроводной связи. Система должна проверять доступ студентов к билетам только через одно мобильное устройство. Для привязки студента к устройству используются ФИО студента и MAC-адрес устройства. Преподаватель должен иметь информацию о вытянутых билетах студентами, а также время этого действия. Доступ к экзаменационным билетам должен быть предоставлен только студентам определенной группы, которые были допущены к экзамену. Преподаватель может разрешить студентам других групп сдать экзамен в исключительных случаях. После регистрации студента и его устройства система выдает случайный, еще не задействованный билет, и при последующих запросах с мобильного устройства выдается тот же самый билет.

В ходе экзамена студент может вытянуть билет, позволяющий получить оценку автоматически, без ответа на вопросы. Для этого в течение семестра используется система промежуточной оценки знаний, результаты которой представлены в виде четырех пар значений: тема - оценка. Курс разбит на 10 тем таким образом, чтобы, ответив на любой вопрос из темы в течение семестра, можно было получить оценку за всю тему. Для вычисления автоматической оценки на основе выбранного билета система должна проверить, в какие темы попадают вопросы выбранного билета, и сопоставить их с оценками, полученными студентом, вытянувшим билет.

**2. Выделение прецедентов**

**2.1. Определение рамок системы**

Чтобы точнее установить рамки проектируемой системы, определим, за что система не должна отвечать:

1. Система не отвечает за процессы подключения и аутентификации мобильных устройств студентов, за это отвечают протоколы безопасности беспроводных сетей.
2. Система не отвечает за сопоставление конкретных МАС-адресов и адресов мобильных устройств в сети IP, за это отвечают протоколы DHCP и ARP, реализованные в рамках операционной системы или роутера (точки доступа).
3. Система не отвечает за визуализацию содержимого экзаменационного билета, за это отвечает браузер на мобильном устройстве.

Мы определили, за что система не отвечает, иначе говоря, – внешние вспомогательные исполнители.

Исполнитель (actor) – это сущность, которая обладает поведением, компьютерная система или организация.

К исполнителям может относиться и сама система, но только в том случае, если она вызывает службы других систем. Существуют специальные исполнители, например, «время», которое вводится в том случае, когда какие-либо действия должны осуществляться по расписанию или через заданный промежуток времени. В остальных случаях различают три типа внешних исполнителей по отношению к разрабатываемой системе:

1. Основной исполнитель (primary) – его задача выполняется с использованием системы. Используется для определения целей пользователя, на основе которых формулируются прецеденты.
2. Вспомогательный исполнитель (supporting) – обслуживает систему, например, предоставляет информацию. Используется для определения внешних интерфейсов и протоколов.
3. Закулисный исполнитель (offstage) – заинтересован в реализации прецедента, но не является основным или вспомогательным исполнителем.

В данном случае внешними вспомогательными исполнителями будут являться операционная система, беспроводной роутер, а также браузер мобильного устройства.

**2.2. Определение основных исполнителей и задач**

Для того, чтобы не пропустить важные аспекты при определении ключевых участников и задач, стоит рассмотреть следующие вопросы:

Кто включает и выключает систему?

Кто является системным администратором?

Кто управляет пользователями и безопасностью?

Относится ли время к числу исполнителей, другими словами, должна ли система выполнять какие-либо действия в ответ на события времени?

Есть ли мониторинг для автоматического перезапуска системы при сбое?

Кто отвечает за контроль деятельности и производительности системы?

Как происходит обновление программного обеспечения?

Кто анализирует журналы регистрации и есть ли удаленный доступ к ним?

Могут ли в качестве исполнителей выступать внешние программы или автоматические системы?

Кому сообщать об ошибках или сбоях в системе?

После определения основных исполнителей необходимо провести анализ каждого из них для определения требований, которые они могут предъявить. Для решения этой задачи требуется задать для каждого исполнителя эти 10 вопросов (данные вопросы носят общий характер и должны быть пересмотрены в рамках предметной области проектируемой системы). На основе анализа данных вопросов для каждого исполнителя можно выявить задачи.

Составим перечень исполнителей и задач в виде таблицы.

Таблица 2.2.1

Основные исполнители и задачи

|  |  |
| --- | --- |
| Исполнители | Задачи |
| Студент | Регистрируется на экзамене  Получает билет |
| Преподаватель | Включает и выключает систему  Уточняет участие студента в сдаче экзамена  Анализирует информацию о вытянутых билетах  Анализирует информацию о времени получения билетов |
| Ассистент (деканат) | Формирует списки студентов |
| Система промежуточной оценки знаний | Предоставляет информацию для выставления автоматической оценки за экзамен |

В системе, которую мы разрабатываем, каждая задача пользователя имеет свой прецедент, название которого начинается с существительного, описывающего действие. Из анализа таблицы следует, что в системе присутствуют два основных исполнителя: студент и преподаватель. Следовательно, мы определяем прецеденты, связанные с задачами этих основных участников.

Составим перечень задач и прецедентов в виде таблицы.

Таблица 2.2.2

Основные задачи и прецеденты

|  |  |
| --- | --- |
| Задачи | Прецеденты |
| Регистрируется на экзамен | Регистрация на экзамен |
| Получает билет | Получение билета |
| Уточняет участие студента на экзамене | Допуск на экзамен |
| Вызывает для сдачи экзамена | Вызов на собеседование |
| Анализирует ответ на экзамене | Собеседование на экзамене |

В таблице “Ранжирования прецедентов” мы должны определить, какой прецедент имеет более высокий ранг. Ранжирование прецедентов позволяет определить приоритеты и последовательности выполнения различных сценариев или задач. Это позволит нам оптимизировать подход к созданию системы, а также поможет повысить качество работы системы, сосредоточив усилия на наиболее важных прецедентах. Для ранжирования прецедентов, сравним их по 3 параметрам: важность, объем и сложность. Сложность оценивается на основе количества шагов, необходимых для выполнения прецедента, и степени детализации, которую требуется добавить в систему для поддержки этого прецедента. Способ ранжирования может быть разный (например, прецеденты ранжируются по 5-ти бальной шкале сверху вниз от наиболее до наименее значимого).

Составим ранжировку прецедентов в виде таблицы.

Таблица 2.2.3

Ранжирование прецедентов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Прецеденты | Важность | Объем | Сложность | Ранг |
| Получение билета | 5 | 5 | 4 | 4,7 |
| Собеседование на экзамене | 3 | 4 | 5 | 4 |
| Допуск на экзамен | 4 | 3 | 3 | 3,3 |
| Регистрация на экзамен | 4 | 2 | 1 | 2,3 |
| Вызов на собеседование | 2 | 1 | 2 | 1,7 |

Исходя из параметров, можно сказать, что прецедент “Получение билета” имеет наиболее высокий ранг из-за его высокой важности, объема и сложности.

**2.3. Описание прецедентов**

Прецеденты — это описания того, как система использовалась для решения задач. Важно подчеркнуть, что прецеденты не представляют собой диаграммы, а текстовые описания процессов использования системы. Основной сложностью в описании прецедентов является выбор нужного уровня детализации.

Таким образом, 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.

Пояснения к полям:

**Рамки (границы системы):** отделяют систему от всего остального мира.

**Уровень:** «Задача, определенная пользователем» или «Нефункциональное требование» (возникает только тогда, когда есть сложное нефункциональное требование, которое надо описывать отдельным прецедентом, встречается редко).

**Основной исполнитель (главный актер):** исполнитель, инициирующий

прецедент. В зависимости от сложности прецедента он может иметь или не иметь альтернативные потоки (расширения) и специальные требования.

**Частота использования:** необходимо определить произвольные

категории (например, всегда часто иногда редко) и описать их в словаре.

В качестве примера рассмотрим развернутое описание прецедента Получение билета.

**Прецедент П1. Получение билета**

**Рамки.** Система поддержки проведения экзамена.

**Уровень.** Задача, определенная пользователем.

**Основной исполнитель**. Студент.

**Заинтересованные лица и их требования.**

* Студент. Хочет получить билет и узнать о возможности выставления автоматической оценки. Все это он хочет проделать без лишних волнений, и не отвлекая остальных участников экзамена.
* Преподаватель. Стремится быстро определить, кто из студентов и каким образом может быть автоматически оценен.
* Деканат. Хочет получить аккуратно заполненные ведомости о проведении экзамена.

**Предусловия.** Студент зарегистрировался на экзамене и имеет допуск.

**Результаты (Постусловия).** Студенту предоставлен случайный и еще не

занятый билет. Зафиксировано время получения билета. Определены

автоматические оценки за каждый вопрос в полученном билете.

**Основной успешный сценарий (или основной процесс)**

1. Студент запрашивает у системы выдачу билета.
2. Система проверяет историю выдачи билетов студенту.
3. Система случайным образом выбирает билет, который до этого ни разу не был выбран, и делает пометку о том, что билет занят конкретным студентом.
4. Система записывает время начала подготовки студента.
5. Система определяет номер темы, к которой относится вопрос выбранного билета.
6. Система вычисляет автоматическую оценку для ответа на вопрос на основе данных от системы промежуточного оценивания.

*Система повторяет пункты 5 и б для всех вопросов выбранного билета*

1. Система формирует билет в виде, возможном для отображения, передает его на мобильное устройство.
2. Студент получает на экране мобильного устройства все вопросы и автоматические оценки и начинает готовиться к ответу.

**Расширения (или альтернативные потоки)**

(Альтернативные потоки используются, когда действия пользователя или состояния системы не соответствуют основному сценарию. Например, если студент уже получил билет, система предлагает ему повторное использование текущего билета.)

2-4а. При повторном обращении студента к системе для получения билета:

1. Система определяет, какой билет был выдан студенту при его первом обращении.

4а. В случае, если нет доступных неиспользованных билетов:

1. Система информирует студента о необходимости подождать до появления свободных билетов.
2. Система сообщает преподавателю о том, что свободных билетов нет и конкретный студент не может начать подготовку к ответу.
3. Система завершает обслуживание студента.

**Специальные требования**

На третьем шаге основного сценария необходимо обеспечить, чтобы у разных студентов, выполнивших одновременное обращение к системе, были разные билеты.

**Список технологий и типов данных**

Для возможности работы с более широким кругом различных устройств содержание билетов должно формироваться в виде html разметки.

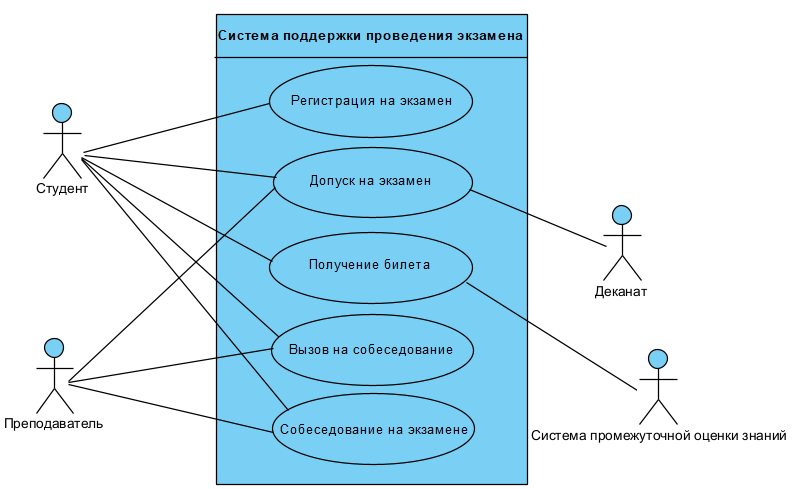
**Частота использования:** постоянно.

**Открытые вопросы**

* Требуется ли от студента каких-либо дополнительных действий для завершения взаимодействия с системой?

**2.4. Построение диаграммы прецедентов**

Средством CASE, используемым в данном контексте, является Visual Paradigm версии 17.1. Предполагается создание диаграммы прецедентов, схожей с приведенной на рисунке 1.



**Рисунок 1 - Диаграмма прецедентов**

На этой диаграмме основные исполнители находятся слева, а вспомогательные справа. Связь взаимодействия может устанавливаться между двумя участниками или двумя прецедентами, включая специальные виды связей: обобщение, расширение и включение. Однако использование этих типов связей в начальной фазе анализа не рекомендуется. За более детальной информацией о типах взаимодействий можно обратиться к Приложению 1.

**3. Описание нефункциональных требований**

Для полного описания всех требований к проектируемой системе недостаточно просто выделить прецеденты. Важными элементами также являются требования к отчетам, документированию, поддержке и лицензированию. В рамках унифицированного процесса используются артефакты, такие как «Дополнительная спецификация», «Видение» и «Словарь терминов» (эти артефакты не обязательны и используются лишь при необходимости).

Артефакт — представляет собой не только документ или диаграмму, но и процесс осмысления, анализа и разработки с последующей записью результатов во избежание повторения или забывания. Для итеративной и эволюционной разработки все артефакты анализа и проектирования рассматриваются как неполные и незавершенные, поскольку они должны эволюционировать в процессе разработки системы.

В данном курсовом проектировании сделаем акцент только на артефакт «Словарь терминов». Словарь терминов обычно делается в виде приложения.

**Словарь терминов**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Версия** | **Создан** | **Изменен** | **Описание** | **Автор** | **Изменил** |
| Черновой начальный вариант | 11 сентября, 2020 | 11 декабря, 2020 | В процессе выполнения учебного курсового проектирования формируется словарь терминов. Начальная итерация включает добавление в словарь терминов, относящихся к успешному сценарию прецедента «Получение билета». | Блок  А.А. | Ремарк  Э. М. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Термин** | **Определение** | **Формат** | **Правило верификации** | **Синоним** |
| Студент | Человек, который сдает экзамен |  |  | Student |
| Система | Система, помогающая в проведении экзамена |  |  | СП (система поддержки) |
| Экзамен | Совокупность вопросов, которые студент должен пройти во время сдачи экзамена |  |  |  |
| Билет | Структурированный формат теста, обычно включающий задачу и теоретические вопросы |  |  | Card |
| Получение билета | Действие выбора и отображения билета на мобильном устройстве студента |  |  | Обращение студента к СП; обращение; повторное обращение |
| Мобильное устройство | Гаджет с беспроводным модулем, принадлежащий студенту |  |  |  |
| Подготовка | Процесс студента, где он составляет ответы на вопросы и решает задачи в строго отведенное время |  |  |  |
| Тема | Это информация по определенному предмету, объединенная общей характеристикой |  |  | Topic |
| Задание | Вопрос, включенный в билет, касающийся либо теории, либо практики |  |  | Task, вопрос |
| Система промежуточной оценки знаний | Комплекс действий для оценки уровня знаний студентов на протяжении учебного семестра |  |  | MStorage, СОЗ |
| Автоматическая промежуточная оценка | Оценка знаний студента по конкретному вопросу и соответствующие рекомендации по самостоятельной подготовке | В целом, формат может изменяться в зависимости от системы оценок, выбранной преподавателем. | Должна быть неотрицательной и не превышать максимально допустимое значение. | Mark, оценка |
| MAC-адрес | Уникальный идентификатор, который назначается каждому активному устройству в компьютерных сетях. | 48-разрядное двоичное число, записываемое как совокупность 6 октетов |  | Media Access Control |
| Попытка | Интерактивное взаимодействие студента с системой в рамках определенной задачи. |  |  | Attempt |

Словарь терминов должен постоянно пополняться и развиваться во время всего процесса проектирования. Эти термины будут использоваться для создания множества других артефактов процесса UP. Кроме того, правильный подбор синонимов способен упростить последующее проектирование.

**4. Моделирование предметной области**

Концептуальная модель предметной области создается посредством выполнения следующих действий.

1. Выделения концептуальных классов.
2. Отображения их в модели предметной области в виде классов на диаграмме UML.
3. Добавления необходимых ассоциаций и атрибутов.

Результатом выполнения этих действий для предметной области системы поддержки проведения экзамена будет следующая диаграмма классов (рис.2).

Изображение выглядит как текст, диаграмма, снимок экрана, Прямоугольник

Автоматически созданное описание

**Рисунок 2 - Диаграмма классов предметной области**

Следует отметить, что классы на данной диаграмме изображают отдельный вид сущности, а связи между ними характеризуют порядок взаимодействия. Данная диаграмма НЕ ЯВЛЯЕТСЯ диаграммой классов проектирования и не должна содержать понятия из области программирования. В процессе выполнения курсового проектирования необходимо показать возможность выполнения основных сценариев для всех прецедентов на диаграмме классов предметной области.

**5. Составление системных диаграмм последовательностей**

Диаграмма последовательностей — наглядный и простой в создании артефакт, отображающий события, входящие и исходящие, связанные с данным программным продуктом.

Системная диаграмма последовательностей (СДП) — представляет собой схему, которая показывает внешние и внутренние события, связанные с конкретным сценарием использования системы. Она используется для отображения событий, которые проходят через границы системы между внешними исполнителями и самой системой. СДП изображает разрабатываемую систему как «черный ящик».

Для создания СДП обычно используется основной успешный сценарий прецедента, но также может быть составлена для сложных альтернативных сценариев. Хорошей практикой при построении СДП является использование глаголов в начале названия каждой операции.

Любые новые объекты или понятия, введенные на диаграмме, должны быть описаны в словаре терминов.

Далее приведен пример для построения СДП для прецедента «Получение билета» (рис.3).

Изображение выглядит как текст, снимок экрана, линия, Параллельный

Автоматически созданное описание

**Рисунок 3 - СДП «Получение билета»**

**6. Составление описаний операций**

Описания операций играют ключевую роль в описании того, как система реагирует на системные события и как изменяется состояние объектов в предметной области.

Системные события определяются в процессе создания СДП и обрабатываются системой через соответствующие системные операции. Совокупность всех таких системных операций, применяемых в рамках всех прецедентов системы, образует открытый системный интерфейс.

В процессе описания системных операций модель предметной области может меняться и совершенствоваться. Описания операций могут быть созданы частично на одной итерации и доработаны на последующих итерациях.

Далее представлен шаблон описания системной операции:

**Описание операции ОП№операции: имя операции**

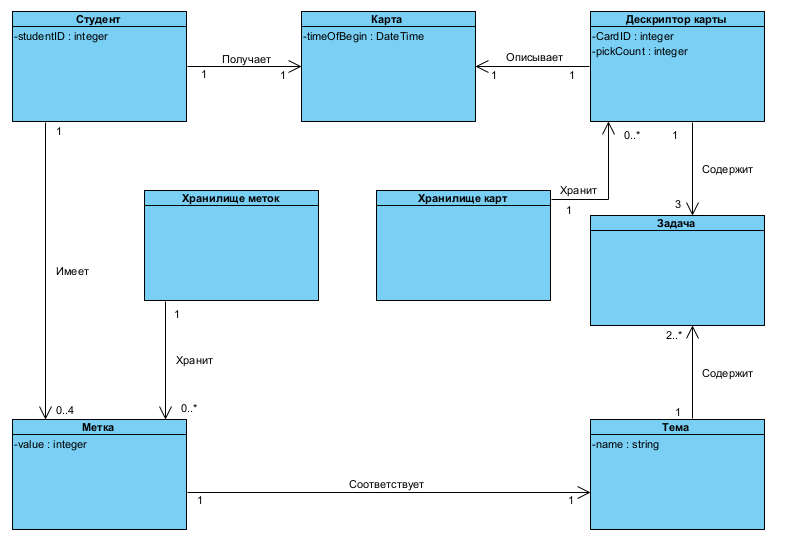
|  |  |
| --- | --- |
| **Операция** | Имя операции и ее параметры. |
| **Ссылки** | Прецеденты, в рамках которых может быть выполнена определенная операция. |
| **Предусловия** | Предположения о состоянии системы или объектов модели предметной области до выполнения операции. |
| **Постусловия** | Состояние объектов модели предметной области после завершения операции. |

Далее в качестве примера составим описание операции для выделенного в предыдущем параграфе системного события ***takeCard***.

**Описание операции ОП1: takeCard**

|  |  |
| --- | --- |
| **Операция** | takeCard(studentID:integer) |
| **Ссылки** | Прецеденты: Получение билета |
| **Предусловия** | Студент успешно зарегистрирован на экзамене и имеет идентификатор. |
| **Постусловия** | * Создан экземпляр **card** класса **Card** (создание экземпляра). * Атрибуту **card.timeOfBegin** присвоено значение **time** (модификация атрибута). * Экземпляр **card** связан с классом **CardDescriptor** на основе соответствия идентификатора (номера) билета (формирование ассоциации). * Атрибут **pickCount** экземпляра класса **CardDescriptor**, ассоциированного с экземпляром card, изменен (модификация атрибута). * Экземпляр **card** связан с классом **Student** на основе соответствия идентификатора студента (формирование ассоциации). |

В процессе разработки описания были интегрированы новые атрибуты в модель предметной области (рис. 4.)



**Рисунок 4 - Модель предметной области**

Описание системных операций может привести к значительным изменениям в модели предметной области, включая формирование новых связей и концептуальных классов.

**7. Реализация прецедентов**

Реализация прецедентов включает создание модели проектирования, которая отображает как динамические, так и статические аспекты взаимодействия. Для этого используются два типа UML-диаграмм: диаграммы взаимодействий и диаграммы классов. Каждый прецедент и соответствующие ему операции описываются параллельно двум видам диаграмм. Начинают строить модель с диаграмм взаимодействий. Прецедент "Запуск системы" планируется реализовать в последнюю очередь, так как в рамках итерационной разработки создается простая версия прецедента запуска, которую затем уточняют после реализации каждого прецедента.

В объектно-ориентированном проектировании рекомендуется применять метод распределения обязанностей - (responsibility-driven design - RDD). Для распределения обязанностей между взаимодействующими объектами используют метод GRASP (General Responsibility Assignment Software Patterns - Общие шаблоны распределения обязанностей в программных системах). GRASP насчитывают девять принципов распределения обязанностей:

1. Information Expert (Информационный эксперт).
2. Creator (Создатель).
3. Controller (Контроллер).
4. Low Coupling (Слабая связанность).
5. High Cohesion (Сильное зацепление).
6. Polymorphism (Полиморфизм).
7. Pure Fabrication (Чистая выдумка).
8. Indirection (Посредник).
9. Protected Variations (Сокрытие реализации).

В этой работе необходимо соблюдать первые пять принципов, а оставшиеся четыре следует применять только при условии их обоснования на основе индивидуального задания.

**Краткое описание первых пяти шаблонов распределения обязанностей**

**Information Expert (Информационный эксперт)**

Шаблон Information Expert определяет базовый принцип назначения обязанностей. Он утверждает, что обязанности должны быть назначены объекту, который владеет максимумом необходимой информации для выполнения обязанности. Такой объект называется информационным экспертом. Возможно, этот шаблон является самым очевидным из девяти, но вместе с тем и самым важным.

Если дизайн не соответствует этому принципу, то при написании программного кода возникает так называемый "спагетти-код", который сложно понять и разобрать. Локализация обязанностей повышает уровень инкапсуляции и уменьшает уровень связанности. Помимо улучшения читаемости кода, уровень готовности компонента к повторному использованию также повышается.

**Creator (Создатель)**

Шаблон Creator решает, кто должен создавать объект. По сути, это применение шаблона Information Expert к проблеме создания объектов. Более подробно, его задача - назначить классу B обязанность создания экземпляров класса A, если одновременно выполняются следующие условия:

* Класс В содержит или агрегирует объекты А.
* Класс B выполняет запись экземпляров объектов A.
* Класс B активно использует объекты класса A.
* Класс B обладает необходимыми данными для инициализации объектов класса A.

**Controller (Контроллер)**

Контроллер берет на себя функцию обработки запросов, поступающих от пользователя, и, часто, управляет сценариями использования, связанными с определенными действиями (например, один контроллер может управлять созданием и удалением пользователя). Обычно контроллер не выполняет все операции самостоятельно, а делегирует обязанности компетентным объектам. Иногда класс-контроллер представляет всю систему в целом, корневой объект, устройство или важную подсистему (внешний контроллер).

**Low Coupling (Слабая связанность)**

Low Coupling - это принцип, который способствует разделению задач между объектами таким образом, чтобы уровень взаимосвязи между системными компонентами был минимален. Степень связанности (coupling) определяет, насколько тесно элемент связан с другими элементами или сколько информации об этих элементах он знает. Элемент с низкой степенью связанности (или слабой связью) зависит от небольшого числа других элементов и присущи следующие характеристики:

* Низкое количество взаимосвязей между классами (или подсистемами);
* Слабая зависимость одного класса (или подсистемы) от изменений в другом классе (или подсистеме);
* Возможность повторного использования компонентов системы в большей степени.

**High Cohesion (Сильное сцепление)**

High Cohesion (Сильное сцепление) – это принцип, который обеспечивает высокую степень внутренней целостности в подсистеме. Классы (подсистемы) таким образом получаются сфокусированными, легкоуправляемыми и понятными. Cohesion (зацепление) (точнее, функциональное зацепление) – это параметр, определяющий связь и сфокусированность обязанностей класса. Высокую степень сцепления имеет объект (или подсистема), когда его задачи хорошо соответствуют друг другу, и он не перегружен избыточной работой.

Классы с низкой сцепленностью выполняют разноплановые или несвязанные задачи, что может привести к следующим проблемам:

* сложности в понимании структуры;
* затруднения при повторном использовании;
* сложность поддержки;
* нестабильность, частые изменения требований.

Классы со слабым сцеплением зачастую слишком абстрактны или выполняют функции, которые легко можно распределить между другими объектами.

**Реализация прецедента "Получение билета"**

Для реализации прецедента “Получение билета” создаются проектные решения для каждой операции системы на основе сценария использования и описания операции.

**Проектное решение: takeCard**

Первым шагом является выбор класса-контроллера для обработки операции takeCard. В соответствии с шаблоном Controller, контроллером может выступать класс, удовлетворяющий одному

из следующих условий:

* Класс представляет всю систему в целом, "корневой объект", специализированное устройство или подсистему.
* Класс является получателем или обработчиком всех системных событий для некоторого сценария прецедента.

Внешний контроллер выбирается в случае, когда в приложении небольшое количество системных операций и он не будет перегружен излишними обязанностями. Использование контроллеров прецедентов удобно для распределения обязанностей между различными классами и избежания перегрузки классов-контроллеров. В данном конкретном случае (учитывая небольшое количество прецедентов и системных операций, см. рис. 1) внешний контроллер GeneralController является подходящим выбором.

Согласно шаблону Creator, класс GeneralController хорошо подходит для создания объектов класса Card, так как он обладает необходимыми данными для инициализации объектов Card.

С одной стороны, можно было бы дать классу GeneralController полномочия напрямую работать с классами хранилища и передавать объекту Card всю необходимую информацию для генерации предоставления билета выбранным студентом. Однако, такой подход снизил бы степень сцепления для GeneralController, что противоречило бы принципу High Cohesion. Поэтому после создания объекта Card GeneralController должен передать сообщение другому объекту для начала процесса создания билета.

Для определения конкретного класса, реализующего генерацию

представления билета, будем использовать шаблон Information Expert. Для этого необходимо ответить на вопрос о необходимой информации для данной операции, которую можно найти в описании прецедента: требуется знание о номерах тем и оценках студента на коллоквиумах, а также номерах тем для выбранного билета. Информация о темах билета содержится в классе, описывающем билет, идентификатор которого присутствует в объекте Card с момента создания, а идентификатор объекта класса Student передается в конструктор. Следовательно, класс Card является информационным экспертом.

Классы CardStorage и MarkStorage являются частичными информационными экспертами в силу агрегации и соответственно, информации об объектах Mark и Topic. Диаграмма взаимодействия представлена на рис. 5.

Изображение выглядит как текст, диаграмма, Параллельный, снимок экрана

Автоматически созданное описание

**Рисунок 5 - Диаграмма взаимодействия для takeCard()**

Теперь рассмотрим пример построения диаграммы классов проектирования для данного проектного решения. Как диаграмма взаимодействия, так и диаграмма классов должны соответствовать описанию операции и быть построены с учетом принципов проектирования GRASP. Оба типа диаграмм должны быть взаимосвязаны и дополнять друг друга, обеспечивая выполнение поставленных условий.

Таким образом диаграмма классов проектирования содержит дополнительную информацию, которая дополняет анализ по составлению диаграммы взаимодействия. На следующем рисунке (рис. 6) представлена диаграмма классов проектирования для проектного решения takeCard.

Изображение выглядит как текст, снимок экрана, диаграмма, линия

Автоматически созданное описание

**Рисунок 6 - Диаграмма классов проектирования для takeCard**

На рис. 6 показано, что объект GeneralController отвечает за создание и удаление объектов Card, что также подчеркивается отношением композиции. Согласно принципу High Cohesion, класс Card содержит ссылки на объекты типов CardStorage и MarkStorage для делегирования им части своих обязанностей. Классы Card и Mark зависят от класса Student. Концептуальные классы Topic и Task не представлены как отдельные классы в диаграмме проектирования, но их объекты отображены как коллекции в классах хранилищ.

**Внесение изменений в проектное решение takeCard**

После проведения анализа полученных диаграмм проектирования становится очевидным одно упущение, ранее казавшееся незначительным.

Объекты хранилища сохраняют данные даже после перезапуска системы, что означает, что оценки, билеты и их содержимое остаются неизменными для каждого сеанса работы. В то же время, объекты Card создаются только на время, необходимое для генерации представления билета для конкретного студента. Это приводит к тому, что при повторном обращении студента система выдаст другой случайный билет, что не соответствует альтернативному сценарию прецедента П1. Для исправления этой ситуации необходимо ввести новый концептуальный класс Attempt в предметную область (рис. 7).

Изображение выглядит как текст, снимок экрана, диаграмма, Прямоугольник

Автоматически созданное описание

**Рисунок 7 - Исправленная модель предметной области**

Таким образом, требуется отразить внесенные изменения и в описании операции.

**Описание операции ОП1: takeCard**

|  |  |
| --- | --- |
| **Операция**  **Ссылки**  **Предусловия** | takeCard(studentID:integer).  Прецеденты: Получение билета.  Студент успешно зарегистрирован на экзамене и имеет идентификатор. |
| **Постусловия** | * Создан экземпляр **card** класса **Card** (создание экземпляра). * Атрибуту **card.timeOfBegin** присвоено значение **time** (модификация атрибута). * Экземпляр **card** связан с классом **CardDescriptor** на основе соответствия идентификатора (номера) билета (формирование ассоциации). * Атрибут **pickCount** экземпляра класса **CardDescriptor**, ассоциированного с экземпляром **card**, изменен (модификация атрибута). * Экземпляр **card** связан с классом **Student** на основе соответствия идентификатора студента (формирование ассоциации). * Создан экземпляр **a** класса **Attempt** (создание экземпляра). * Инициализированы все атрибуты объекта **a** соответствующими контексту значениями (модификация атрибута). * Объект a связан с классом **AttemptStorage** (формирование ассоциации). |

Следующим шагом изменим диаграмму взаимодействий (рис. 8).

Изображение выглядит как текст, диаграмма, Параллельный, линия

Автоматически созданное описание

**Рисунок 8 - Исправленная диаграмма взаимодействия для takeCard**

На рис. 8, помимо всего прочего, был добавлен фрейм, демонстрирующий наличие альтернативного варианта развития прецедента. Помимо этого, хотя в описании операции этого не указано, объект Card теперь является временным.

И наконец, изменим диаграмму классов проектирования (рис. 9)

Изображение выглядит как текст, снимок экрана, диаграмма, линия

Автоматически созданное описание

**Рисунок 9 - Исправленная диаграмма классов проектирования для takeCard**

На диаграмме (рис. 9) отсутствует класс Attempt, так как он тривиален.

**Заключение**

Унифицированный процесс (UP) проектирования информационных систем является общепризнанной методологией проектирования и системного анализа. UP применим при проектировании программных систем, а также аппаратно-программных и только аппаратных. Кроме того, UP относится к классу гибких подходов к проектированию, то есть в процессе проектирования необязательно использование каких-либо артефактов. Более того, UP позволяет быстро вносить изменения в ходе проектирования. Благодаря этому UP является очень популярным методом проектирования.

Чтобы правильно понимать и применять основные принципы проектирования, заложенные в UP, конечно, требуется активно использовать его на практике, прочтения итерационных методов разработки и проектирования недостаточно, для того чтобы сформировать стабильный навык использования методологии UP. С другой стороны, положительные результаты в освоении UP удается получить посредством коллективной работы над одним проектом. Для этого в процессе выполнения курсового проектирования рекомендуется объединять студентов в группы, распределяя варианты задания между объединениями.

Данное пособие акцентируется на усвоении и дальнейшем практическом применении подхода к проектированию, основанного на распределении обязанностей. В этом контексте UP можно разделить не на модели, а на два этапа: этап формулирования обязанностей и этап их распределения.

На первом этапе строятся модели прецедентов и предметной области UP. Это необходимо для формализации "обязанностей" из совокупности требований к проектируемой системе. В UP "обязанности" — это СДП (графическое представление) и постусловия описаний операций (текстовое представление). Использование какого-либо артефакта в качестве "обязанности" обусловлено полнотой составления этого артефакта и простотой требований при создании обязанностей. Однако, чаще всего удобнее использовать список постусловий в описании операций.

На втором этапе проектировщик должен изучить возможность создания новых или использования существующих программных классов, опираясь на принципы распределения обязанностей, изложенные в виде шаблонов проектирования GRASP. В ходе этой работы создается модель проектирования, являющаяся совокупностью реализаций прецедентов (проектных решений). Самой ценной и содержательной частью курсового проекта является объяснение выбранных проектных решений с применением принципов распределения обязанностей. Следует отметить роль модели предметной области, а именно — промежуточное состояние между требованиями к системе и реальными программными классами, реализующие функционал и удовлетворяющие заданным требованиям. Благодаря модели предметной области любой проектировщик может эффективно распределять работу между другими участниками проекта и реагировать на любые новые требования к проектируемой системе.

**Содержание отчета**

1. Титульный лист.
2. Текст задания на курсовое проектирование.
3. Содержание.
4. Содержательная часть проектирования, выполненная в соответствии с критериями выполнения курсового проектирования.
5. Словарь терминов.
6. Список литературы.
7. Развернутые тезисы доклада на защите проекта с указанием авторов каждого тезиса.

**Приложение 1**

**Описание типов взаимодействия прецедентов**

На рисунке П1 показаны обозначения, принятые в языке UML, для различных типов отношений между прецедентами.

Точки расширения:

<<включает>>

<<расширяет>>

отношение обобщения

отношение расширения

отношение включения

**А**

**Рисунок П1 - Обозначение UML для различных отношений прецедентов**

Отношение включения, изображенное на левой стороне рисунка П1, показывает, что прецедент А содержит прецедент Б. Эта связь является основной и ключевой для предотвращения повторений текста при описании прецедентов. Например, если определенный сценарий часто встречается в описаниях прецедентов, его можно выделить в отдельное описание, на которое будут ссылаться другие зависимые прецеденты. Кроме того, это отношение позволяет разбивать слишком длинные сценарии на более мелкие прецеденты.

Отношение расширения, изображенное в центре рисунка П1, указывает на то, что прецедент Б дополняет прецедент А. Это отношение используется, когда необходимо добавить новый элемент к уже существующему прецеденту, но изменение самого сценария потребует множества последующих корректировок. Для использования этой связи необходимо указать точки расширения основного прецедента, которые имеют имена и соотносятся с конкретным шагом сценария, который будет расширен.

Отношение обобщения, изображенное справа на рисунке П1, показывает, что прецедент А является обобщением прецедента И. Смысл этого отношения заключается в демонстрации сценариев, которые являются частными случаями одного и того же прецедента. Использование этого отношения для связывания прецедентов не рекомендуется, и стоит предпочесть альтернативные сценарии или другие типы отношений.

Подписано в печать 18.06.2015г. Формат 60х90 1/16

Объем 1,8 усл. п. л. Тираж 50 экз. Изд.№ 44 Заказ 67.

ООО “Брис-М”. Москва, ул. Авиамоторная, д. 8