Примерный список ответов на примерный список вопросов I-го задания

Сами вопросы находятся здесь.

- *Укажите архитектурные представления в браузере модели. Какое представление из 5-ти стандартных отсутствует в модели?*

  1. UseCase View: представление функциональных возможностей ПО (представление вариантов использования, содержащее сценарии взаимодействия системы с внешней средой и роли, которые играют пользователи ПО и внешние системы).
  2. Logical View: представление логической организации ПО (логическое представление, элементами которого являются пакеты, подсистемы, классы, связи между ними).
  3. Component View: представление физической структуры программных компонент, входящих в состав ПО (представление реализации, элементами которого являются компоненты, связи между ними).
  4. Process View: представление структуры потоков управления и аспектов параллельной работы ПО (представление процессов, элементы которого: потоки управления, нити, параллелизм, синхронизация).
  5. Deployment View: описание физического размещения компонент ПО по узлам вычислительной системы (представление размещения, элементы которого: узлы вычислительной системы, устройства, линии связи, задачи).

  Очевидно, что в нашей модели отсутствует Process View.

- *Какой смысл вкладывается в ту или иную связь (коммуникацию, включение, расширение, ассоциацию, агрегацию, композицию, зависимость, обобщение, реализацию и т. д.)? Сравните связи разного типа между собой (композицию с агрегацией, ассоциацию с зависимостью и т. п.).*

  В диаграмме классов:

  1. Ассоциация [прямая линия] — связь между классами, описывающая группу однородных по структуре и семантике соединений между экземплярами классов. Соединения являются экземплярами ассоциации точно так же, как соединенные объекты являются экземплярами классов, связанных ассоциацией.
  2. Агрегация [ассоциация с белым ромбом] — более сильный тип ассоциативной связи между целым и его частями (пример: автомобиль и мотор).
  3. Композиция [ассоциация с черным ромбом] — усиленная агрегация, когда часть не может существовать без целого (пример: университет, факультет, кафедра). Композиция и агрегация транзитивны, в том смысле, что если B является частью A, и C является частью B, то C также является частью A (но на диаграмме связи, возникающие за счет транзитивности, явно не изображаются).
  4. Реализация [пунктирная линия с треугольным концом] — отношение между двумя элементами модели, в котором один элемент (клиент) реализует поведение, заданное другим (поставщиком). Например класс, реализующий интерфейс.
  5. Зависимость [пунктирная линия со стрелкой] — слабая форма отношения использования, при котором изменение в спецификации одного влечёт за собой изменение другого, причем обратное не обязательно. Возникает когда объект выступает например в форме параметра или локальной переменной.

  В диаграмме вариантов использования:

  1. Коммуникация [линия со стрелкой] — связь между вариантом использования и действующим лицом.
  2. Включения [пунктирная линия со стрелкой] — включение многократно используемой функциональности, представленной в виде абстрактного варианта использования.
  3. Расширение [пунктирная линия со стрелкой] — указывает на особый случай, описанный в абстрактном варианте использования.

  И там и там:

  1. Обобщение (наследование) [линия с треугольным концом] — показывает, что у нескольких действующих лиц имеются общие черты и различия.

- *Какие обязанности несут граничные классы (управляющие, классы-сущности)?*

  1. Граничные классы (boundary classes), являются посредниками при взаимодействии системы с действующими лицами и с аппаратной базой.
  2. Классы-сущности (entity classes), отвечают за хранение данных.

3. Управляющие классы (control classes), реализуют бизнес-логику и обеспечивают координацию поведения объектов в системе.

- *Какие элементы добавлены в модель после окончания анализа, во время проектирования?*

  Добавлена модель DesignModel, в которой классы анализа перешли в классы проектирования, тем самым поменялись диаграммы соответствующих классов, последовательностей. Класс CourseCatalogSystem перешел в подсистему. Для работы с СУБД используются механизмы JDBC. Уточняются детали всех элементов анализа (например определяются методы классов, атрибуты и их типы), etc.

- *Представим, что мы добавили в модель новый вариант использования, к каким последствиям это приведет? Какие диаграммы изменятся, какие добавятся?*

  Скорее всего изменится диаграмма KeyAbstractions, добавятся диаграммы деятельности, последовательностей или коммуникаций, которые будут реализовывать данный вариант использования.

- *Зачем в модель добавлен интерфейс?*

  Для того, чтобы изменение деталей реализации класса CourseCatalogSystem (и всей подсистемы) никак не затрагивало наш проект. Например, если мы захотим перенести систему на другую СУБД, то прокси-класс CourseCatalogSystem должен быть написан заново с учетом особенностей новой СУБД. Но благодаря интерфейсу, это повлечет за собой минимальные изменения в модели.

- *Что находится в реализации интерфейса?*

  Интерфейс реализует подсистема CourseCatalogSystem. Сама подсистема большая, состоит из одноименного прокси-класса и кооперации ICourseCatalogSystem (у нее есть стереотип ≪ interface realization ≫). Думаю что формально интерфейс реализует именно кооперация, но фактически вся подсистема.

- *Что дает использование проектного механизма?*

  Проектные механизмы являются переходным этапом от механизмов анализа к механизмам реализации. Механизм реализации – решение, имеющее конкретного поставщика, проектный механизм – каркас, максимально приближенный к реализации, имеющий конкретное наполнение, чем отличается от механизма анализа, являющегося лишь своеобразной меткой.

  В нашем случае проектный механизм это часть, которая отвечает за взаимодействие с базой данных. С одной стороны мы можем описывать таблицы на основе классов достаточно близко к тому, как это будет в реализации. С другой стороны мы не привязываемся к конкретному механизму, который будет осуществлять связь с базой данных.

- *В чем разница между деловым действующим лицом и действующим лицом?*

  Деловое лицо (business actor) — некоторая роль, выполняемая по отношению к бизнес-процессам организации. Кандидатами на эту роль являются: акционеры, заказчики, поставщики, партнеры, потенциальные клиенты, местные органы власти, коллеги из подразделений, не охваченных моделью, внешние бизнес-системы (предприятия или подразделения). Обнаружить действующих лиц бизнес-процессов можно, найдя ответы на вопросы:

  – Кто извлекает пользу из существования организации?
  – Кто помогает организации осуществлять свою деятельность?
  – Кому организация передает информацию и от кого получает?

  Действующее лицо (actor) — роль, обобщение элементов внешнего окружения системы, ведущих себя по отношению к системе одинаковым образом.

  Мне кажется что разница в том, что первое ориентировано на организацию, а второе на программный продукт. И еще они по разному отображаются на диаграмме.

- *Чем отличается внутренний переход состояния от перехода из состояние в само себя?*

  В состоянии есть возможность указывать действия, которые будут выполняться при входе в него и при выходе. Во время перехода из состояния в самого себя будут выполняться эти действия, а при внутреннем переходе — нет.

- *Зачем понадобился класс Classification?*

  При анализе мы определили, что бывают студенты дневного отделения и студенты вечернего отделения, создав иерархию наследования. Рассмотрим хороша ли такая модель, в случае перевода студента с одного отделения на другое. При переводе требуется создать студента нового типа, скопировать в него данные (значения атрибутов класса Student), а затем исходный объект удалить. Эффективнее было бы общую для дневных и вечерних студентов часть не трогать, а менять только то, что зависит от отделения. Выделим эту часть в абстрактный класс Classification, унаследуем от него FullTimeClassification и PartTimeClassification – переименованные бывшие классы FulltimeStudent и PartTimeStudent.

Ответы на остальные вопросы можно почитать по ссылке.