Контрольная работа

Студента групи КС-32

Безрука Юрия Руслановича

Вариант 1

1. В модели Waterfall легко управлять проектом. Благодаря её жесткости, разработка проходит быстро, стоимость и срок заранее определены. Но это палка о двух концах. Каскадная модель будет давать отличный результат только в проектах с четко и заранее определенными требованиями и способами их реализации.

Каскадную методологию используют:

* Только тогда, когда требования известны, понятны и зафиксированы. Противоречивых требований не имеется.
* Нет проблем с доступностью программистов нужной квалификации.
* В относительно небольших проектах

V-образная модель применима к системам, которым особенно важно бесперебойное функционирование.

V-модель используют:

* Если требуется тщательное тестирование продукта, то V-модель оправдает заложенную в себя идею: validation and verification.
* Для малых и средних проектов, где требования четко определены и фиксированы.
* В условиях доступности инженеров необходимой квалификации, особенно тестировщиков.

В инкрементной модели полные требования к системе делятся на различные сборки.

Инкрементную модель используют:

* Когда основные требования к системе четко определены и понятны. В то же время некоторые детали могут дорабатываться с течением времени.
* Требуется ранний вывод продукта на рынок.
* Есть несколько рисковых фич или целей.

В «гибкой» методологии разработки после каждой итерации заказчик может наблюдать результат и понимать, удовлетворяет он его или нет. Это одно из преимуществ гибкой модели. К ее недостаткам относят то, что из-за отсутствия конкретных формулировок результатов сложно оценить трудозатраты и стоимость, требуемые на разработку.

Agile используют:

* Когда потребности пользователей постоянно меняются в динамическом бизнесе.
* Изменения на Agile реализуются за меньшую цену из-за частых инкрементов.
* В отличие от модели водопада, в гибкой модели для старта проекта достаточно лишь небольшого планирования.

Итерационная модель жизненного цикла не требует для начала полной спецификации требований. Вместо этого, создание начинается с реализации части функционала, становящейся базой для определения дальнейши требований. Этот процесс повторяется.

Когда оптимально использовать итеративную модель?

* Требования к конечной системе заранее четко определены и понятны.
* Проект большой или очень большой.
* Основная задача должна быть определена, но детали реализации могут эволюционировать с течением времени.

«Спиральная модель» похожа на инкрементную, но с акцентом на анализ рисков.

Эта модель не подойдет для малых проектов, она резонна для сложных и дорогих, например, таких, как разработка системы документооборота для банка, когда каждый следующий шаг требует большего анализа для оценки последствий, чем программирование.

2. Перечислите основные типы UML диаграм.  
1) Use Case Diagram (Диаграммы прецедентов). Под прецедентом имеется ввиду фиксированная последовательность действий между пользователями и ПС или между отдельными подсистемами, которая обеспечивает достижение поставленных целей.  
2) Class Diagram (Диаграмма классов). Она служит для представления статической структуры модели системы в терминологии классов объектно-ориентированного программирования. Она отражает различные взаимосвязи между отдельными сущностями предметной области, такими как объекты и подсистемы, и описывает их внутреннюю структуру и типы отношений.  
3) Composite structure diagram (Диаграмма композиционной структуры) это статическая структурная диаграмма, которая демонстрирует внутреннюю структуру классов и, по возможности, взаимодействие элементов (частей) внутренней структуры класса.  
4) Object diagram (Диаграмма объектов) демонстрирует полный или частичный снимок моделируемой системы в заданный момент времени. На ней отображаются экземпляры классов (объекты) системы с указанием текущих значений их атрибутов и связей между объектами.  
5) Sequence diagram (Диаграмма последовательности). Эта диаграмма определяет определенные сценарии обмена сообщениями (или вызовами соответствующих функций) между различными логическими (архитектурными) компонентами ПС.  
6) Communication diagram (Диаграмма коммуникации). На ней изображаются взаимодействия между частями композитной структуры или ролями кооперации.  
7) Activity diagram (Диаграмма активности). На ней показывается разложение некоторой деятельности на её составные части.  
8) State diagram (Диаграмма конечного автомата). Она описывает возможные последовательности состояний и переходов, которые в совокупности характеризуют поведение элемента модели в течение его жизненного цикла.  
9) Timing diagram (Диаграмма времени) это альтернативное представление диаграммы последовательности, явным образом показывающее изменения состояния на линии жизни с заданной шкалой времени.  
10) Interaction Overview diagram (Диаграмма образования взаимодействия)  
11) Component diagram (Диаграмма компонентов). Она описывает особенности физического представления системы. Диаграмма компонентов позволяет определить архитектуру разрабатываемой системы, установив зависимости между программными компонентами (исходный, бинарный и исполняемый код).  
12) Deployment diagram (Диаграмма развертывания) предназначена для визуализации элементов и компонентов программы, существующих лишь на этапе ее исполнения. При этом представляются только компоненты-экземпляры программы, являющиеся исполнимыми файлами или динамическими библиотеками.  
13) Package diagram (Диаграмма пакетов). Этот тип диаграммы позволяет агрегировать отдельные компоненты в отдельные группы (подсистемы), в зависимости от их логического назначения в системе, является понятие пакета.

3. Переплетение сложной бизнес-логики со служебным функционалом показывает, что для выделения некоторой общей функциональности, выше парадигмы объектно-ориентированного программирования недостаточно, т.к. множеству разных объектов приходится выполнять повторяющиеся, не связанные с их сущностью действия для служебных целей ПС.

Функциональность называют «сквозной» или «разбросанной», в виду того, что её реализация действительно разбросана по разным частям приложения. Примерами сквозной функциональности могут служить логирование, обработка транзакций, обработка ошибок, авторизация и проверка прав, кэширование и т.д.

Основной задачей аспектно-ориентированного программирования (АОП) является модуляризация сквозной функциональности, выделение её в аспекты.

4. Информационный эксперт используется когда в системе должна аккумулироваться, разсчитываться и т.п. различна информация. В этом случае нужно назначить обязанность действий с информацией некоторому классу (Информационному эксперту), который владеет необходимой информацией.

Например, необходимо рассчитать общую сумму продажи. Имеются классы проектирования "Продажа", "ТоварПродажа" (продажа отдельного вида товара в рамках продажи в целом), "ТоварСпецификация" (описание конкретного вида товара). Необходимо распределить обязанности по предоставлению информации и расчету между этими классами. Объект "Продажа" должен передать сообщение "Рассчитать промежуточную сумму" каждому экземпляру класса "ТоварПродажа" (которые, в свою очередь, передают сообщения "СообщитьЦену" объектам "ТоварСпецификация", с целью получения информации о цене экземпляра товара), и, затем, просуммировать полученные результаты. Промежуточную сумму рассчитывает объект "Товар Продажа". Таким образом, все три объекта являются информационными экспертами.