**Контрольные вопросы**

1.

Основные этапы разработки программного продукта:

1) **Анализ требований**

В рамках этой стадии происходит максимально эффективное взаимодействие нуждающегося в программном решении клиента и сотрудников компании-разработчика, в ходе обсуждения деталей проекта помогающих более четко сформулировать предъявляемые к ПО требования.

2) **Проектирование**

В рамках данного этапа стороны должны осуществить:

-оценку результатов проведенного первоначально анализа и выявленных ограничений;

-поиск критических участков проекта;

-формирование окончательной архитектуры создаваемой системы;

-анализ необходимости использования программных модулей или готовых решений сторонних разработчиков;

-проектирование основных элементов продукта — модели базы данных, процессов и кода;

-выбор среды программирование и инструментов разработки, утверждение интерфейса программы, включая элементы графического отображения данных;

-определение основных требований к безопасности разрабатываемого ПО.

3) **Кодирование**

От грамотного подхода к этапу кодирования зависит эффективность работы компании, заказавшей разработку.

4) **Тестирование и отладка**

Данные этапы позволяют ликвидировать огрехи программирования и добиться конечной цели — полнофункциональной работы разработанной программы.

5) **Внедрение**

Процедура внедрения программного обеспечения в эксплуатацию является завершающей стадией разработки и нередко происходит совместно с отладкой системы.

2.

Артефакты программных продуктов — результаты процесса разработки программного продукта, которые напрямую не участвуют в работе созданного приложения, но являются результатами разработки, например — файлы исходного кода и файлы данных, из которых создаются артефакты размещения.

3.

В настоящее время наиболее широко используются три базовые стратегии разработки программного обеспечения, предназначенные для устранения или сокращения вышеназванных недостатков:

-каскадная;

-инкрементная;

-эволюционная.

***Каскадная стратегия*** представляет собой однократный проход этапов разработки.

Данная стратегия основана на полном определении всех требований к разрабатываемому программному средству в начале процесса разработки. Возврат к уже выполненным этапам разработки не предусматривается. Промежуточные результаты в качестве версии программного средства не распространяются.

***Инкрементная стратегия***представляет собой многократный проход этапов разработки с запланированным улучшением результата.

Данная стратегия основана на *полном*определении всех требований к разрабатываемому ПС в начале процесса разработки. Однако полный набор требований реализуется постепенно в соответствии с планом на последовательных циклах разработки. Результат каждого цикла называется ***инкрементом***.

***Эволюционная стратегия*** представляет собой многократный проход

этапов разработки. Данная стратегия основана на частичном определении

требований к разрабатываемому ПС в начале процесса разработки. Требования постепенно уточняются в последовательных циклах разработки. Результат каждого цикла разработки обычно представляет собой очередную поставляемую версию ПС.

4.

**Индивидуальный процесс разработки** (англ. Personal software process, **PSP**) — процесс разработки ПО, помогающий разработчикам понимать и улучшать собственную производительность. Создан для применения принципов модели зрелости процессов к практике одного разработчика.

**Командный процесс разработки** (англ. Team software process, **TSP**) - определенная структура операционных процессов, которая предназначена для помощи командам менеджеров и инженеров в организации проектов и производстве программного обеспечения для продуктов, размер которых варьируется от небольших проектов в несколько тысяч строк кода (KLOC) до очень больших проектов.

**Capability Maturity Model** — модель зрелости возможностей (модель полноты потенциала) создания ПО: эволюционная модель развития способности компании разрабатывать программное обеспечение.

5.

Team Leader, Senior Manager, Project Manager, Business Analyst, Developers, Tester

6.

Метрики программного обеспечения можно разделить на три категории —

-***Метрики продукта*** — описывает характеристики продукта, такие как размер, сложность, особенности дизайна, производительность и уровень качества.

-***Метрики процесса*** — эти характеристики могут использоваться для улучшения деятельности по разработке и сопровождению программного обеспечения.

-***Метрики проекта*** — эти метрики описывают характеристики и исполнение проекта. Примеры включают число разработчиков программного обеспечения, штатное расписание в течение жизненного цикла программного обеспечения, стоимость, график и производительность.

7.

**Инспектирование программ** – это просмотр и проверка программ с целью обнаружения в них ошибок.

Процесс инспектирования – формализованный. В нем принимает участие небольшая группа людей. У каждого в группе есть своя роль. Обязательно должны присутствовать: автор, рецензент, инспектор, координатор. Рецензент «озвучивает» программный код, инспектор проверяет его, координатор отвечает за организацию процесса. По мере накопления опыта инспектирования в организациях могут появляться другие предложения по распределению ролей в группе.

В процессе инспектирования организация накапливает определенный опыт, поэтому результаты инспектирования можно использовать для улучшения всего процесса разработки ПО. В ходе инспектирования выполняется анализ обнаруженных дефектов.

8.

**Функциональное требование** описывает, что должна делать программная система, в то время как **нефункциональные требования** накладывают ограничения на то, как система будет это делать.

Примером функционального требования может служить:

Система должна отправлять email всякий раз, когда выполняется определенное условие (например, заказ размещен, клиент зарегистрировался и т. Д.).

Связанные с этим нефункциональные требования к системе могут быть:

Электронные письма должны отправляться с задержкой не более 12 часов с момента такого действия.

9.

**Варианты использования программного продукта** - в разработке программного обеспечения и системном проектировании это описание поведения системы, когда она взаимодействует с кем-то (или чем-то) из внешней среды.

10.



11.

