

**МИНОБРНАУКИ РОССИИ**

**федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Московский государственный технологический университет «СТАНКИН»**

**(ФГБОУ ВО МГТУ «СТАНКИН»)**

|  |  |
| --- | --- |
| **Институт**  информационных технологий | **Кафедра**  информационных систем |

**Реферат № 1**

по дисциплине «**Основы программной инженерии**»

на тему: «Вводная. Введение в программную инженерию»

|  |  |  |
| --- | --- | --- |
| **Студент** группа ИДБ–22–06 |  | Мустафаева П.М. |
|  | подпись |  |
| **Руководитель**  к.э.н., доцент |  | **Ахмедов Э.Р.** |
|  | подпись |  |

Москва 2025 г.

**1.Что такое программная инженерия?**

**Программная инженерия –**это область компьютерных наук, ориентированная на проектирование, разработку, тестирование и сопровождение программных приложений. Она включает в себя применение инженерных принципов и опыта компьютерного программирования для разработки программных систем, отвечающих потребностям пользователей. Программная инженерия охватывает процесс создания, тестирования, внедрения и сопровождения цифровых продуктов. Основная задача — обучение специалистов, которые способны разрабатывать надёжные и эффективные системы, соответствующие требованиям пользователей и бизнеса.

Согласно SWEBOK программная инженерия включает в себя следующие десять областей знаний:

1) Требования (software requirements):

2) Проектирование (software design);

3) Конструирование (software construction);

4) Тестирование (software testing);

5) Поддержка и эксплуатация (software maintenance );

6) конфигурационное управление (software configuration management);

7) управление инженерной деятельностью ( software engineering management);

8) процессы инженерной деятельности (software engineering process);

9) инженерные инструменты и методы (software engineering tools and methods);

10) Качество (software quality).

**2. Чем отличается процесс промышленной разработки программного обеспечения от других производственных процессов?**

Процесс промышленной разработки программного обеспечения отличается от других производственных процессов следующими особенностями:

* Отличие кода программ для производственных целей от кода потребительского ПО;
* Ограниченный выбор языков программирования;
* Недопустимость использования программного обеспечения, которое будет с задержкой реагировать на запросы;
* Зависимость разрабатываемых программных продуктов от других систем и приложений, которые используются предприятием;
* Необходимость разрабатывать программное обеспечение совместно с аппаратной частью, чтобы в процессе эксплуатации не возникало сбоев.

**3.Каков состав документации, сопровождающий процесс разработки и сопровождения программного продукта согласно стандарту IEEE? Кратко охарактеризуйте каждый документ.**

* SVVP (План экспертизы проекта): Верификация и валидация.
* SQAP (План контроля качества программного обеспечения): Контроль качества.
* SCMP (План управления конфигурациями программного обеспечения): Управление конфигруациями.
* SPMP (План управления программным проектом)
* SRS (Спецификация требований к программному обеспечению): Ориентированная на заказчика (Реализация проекта), Ориентированная на разработчика (Работа с требованиями)
* SDD (Проектная документация программного обеспечения): Архитектура и Детальное проектирование – Проектирование.
* Исходный код: Написание кода.
* STD (Документация по тестированию программного обеспечения): Тестирование.
* Руководство пользования: Использование.

**4. Что такое артефакт?**

**Артефакт** — неотъемлемая часть результата и процесса выполнения программного проекта, реализованная в виде документации, программного кода (исходного или скомпилированного) или его части (например, модуля).

На начальных стадиях разработки создаются артефакты, описывающие требования к системе: технические задания, спецификации и пользовательские сценарии. В процессе проектирования разрабатываются архитектурные схемы, диаграммы и модели, отражающие структуру будущего программного обеспечения. На этапе программирования основными артефактами становятся исходный код и конфигурационные файлы.

Для обеспечения качества создаются тестовые сценарии, результаты тестирования и отчеты о найденных дефектах. В финальной стадии разработки артефакты включают в себя исполняемые файлы программы, документацию для пользователей и инструкции по установке.

Таким образом, артефакты служат ключевыми элементами процесса разработки, обеспечивая прозрачность, систематизацию и контроль над созданием программного продукта.

**5. Какие этапы определяют развитие процесса разработки программного продукта?**

Развитие процесса разработки программного продукта определяют следующие этапы:

* Планирование (Planning) На этом этапе разрабатывается детальный план проекта. Определяются сроки, бюджет, ресурсы и задачи. Этап включает в себя определение команды разработки, распределение обязанностей и создание графика работ.
* Анализ требований (Requirements Analysis). В рамках этого этапа осуществляется сбор и документирование требований к программному продукту. Он включает в себя общение с заказчиком и конечными пользователями для понимания их потребностей. Результатом этого этапа является создание документа с требованиями.
* Проектирование и дизайн (Design). В рамках этого этапа разрабатывается архитектура программного продукта. Проектируются структура данных, интерфейсы пользователя, компоненты системы и алгоритмы. Этот этап также включает в себя создание дизайна пользовательского интерфейса (UI) и определение технических спецификаций.
* Разработка (Development). На этом моменте начинается активное создание кода. Разработчики пишут программу в соответствии с требованиями и дизайном, определенными на предыдущих этапах. Разработка может включать в себя создание различных модулей, компонентов и функциональных частей программы.
* Тестирование и интеграция (Testing & Deployment). В рамках этого этапа происходит проверка качества программного продукта с помощью различных видов тестирования, включая модульное, интеграционное, функциональное и другие виды. После тестирования ПО выпускается в продакшн. Оно устанавливается на целевой сервер или распространяется конечным пользователям.
* Поддержка (Support). После выпуска ПО продолжает поддерживаться и обновляться. Этап может включать в себя внедрение обновлений, исправление возникающих ошибок, предоставление технической поддержки пользователям и реагирование на запросы на изменение или добавление необходимых функций.

**6. Чем характеризуется стихийный подход к разработке ПО?**

Стихийный подход к разработке программного обеспечения характеризуется разработкой «снизу вверх». При нём вначале проектируют и реализуют сравнительно простые подпрограммы, из которых потом пытаются построить сложную программу.

Стихийный подход к разработке ПО характеризуется следующими особенностями:

- Отсутствие четкой структуры. Процессы разработки не имеют строгих планов и последовательности, чаще всего все осуществляется по мере необходимости.

- Минимальная документация. Документация либо отсутствует, либо ведется формально, что затрудняет понимание и поддержку проекта в будущем.

- Гибкость. Команда может быстро адаптироваться к изменениям требований, но это также может привести к хаосу и несогласованности в работе.

- Неявное определение требований. Требования часто возникают в процессе разработки и могут меняться, что затрудняет планирование и оценку времени.

- Низкая предсказуемость. Из-за отсутствия структуры и четких этапов, итоговые результаты могут варьироваться и быть непредсказуемыми.

**7. Охарактеризуйте структурный подход к разработке ПО.**

Структурный подход к разработке программного обеспечения — это методология, основанная на последовательном разбиении сложной системы на более простые и независимые части. Основная идея заключается в том, что программа создаётся как набор отдельных модулей, каждый из которых выполняет строго определённую функцию. Такой подход упрощает процесс разработки, тестирования и сопровождения программного продукта.

В основе структурного подхода лежит принцип декомпозиции, при котором сложная задача разбивается на более мелкие подзадачи. Каждая подзадача оформляется в виде отдельного модуля с чётко определёнными входными данными, выполняемыми функциями и выходными результатами. Это обеспечивает модульность системы, что позволяет разрабатывать, тестировать и изменять модули независимо друг от друга.

Программа в структурном подходе строится с использованием трёх основных управляющих конструкций: последовательности, разветвления и циклов. Последовательность определяет порядок выполнения операций, разветвление позволяет выбрать один из возможных вариантов выполнения в зависимости от условий, а циклы используются для многократного повторения операций. Такой подход делает программный код более читаемым, понятным и предсказуемым.

**8. Каковы основные положения объектного подхода к разработке ПО?**

Основные положения объектного подхода к разработке ПО включают следующие аспекты:

- Объекты и классы. Основная единица в этом подходе — объект, который объединяет данные и методы, работающие с этими данными. Классы служат шаблонами для создания объектов.

- Инкапсуляция. Скрытие внутренней реализации объектов и предоставление интерфейсов для взаимодействия с ними. Это повышает уровень абстракции и защищает данные от некорректного использования.

- Наследование. Возможность создавать новые классы на основе существующих, что позволяет повторно использовать код и уменьшать дублирование. Наследуемые классы могут расширять или изменять функциональность родительских классов.

- Полиморфизм. Способность объектов разных классов реагировать на одинаковые сообщения (вызовы методов) по-разному. Это увеличивает гибкость и расширяемость программ.

При этом класс объединяет как декларативные, так и процедурные знания: имеет поля (данные, атрибуты) и методы (функции, операции). Методы служат для работы с полями. Процесс разработки ПО стал более эффективным, так как уже на этапе проектирования осуществляется оперирование объектами, соответствующими реальной действительности, что облегчает взаимодействие заказчика и разработчика.

**9. Чем характеризуется компонентный подход к разработке ПО.**

Компонентный подход к разработке программного обеспечения — это методология, основанная на создании программных систем из отдельных, независимых компонентов, которые взаимодействуют друг с другом через чётко определённые интерфейсы. Такой подход позволяет разрабатывать сложные программные продукты с высокой степенью гибкости, повторного использования и масштабируемости.

Основная идея компонентного подхода заключается в том, что программа строится как совокупность автономных блоков, каждый из которых выполняет определённую функцию. Компоненты могут разрабатываться, тестироваться и модифицироваться независимо друг от друга, что значительно упрощает процесс сопровождения программного продукта. Каждый компонент обладает собственным интерфейсом, который определяет способы взаимодействия с другими компонентами, скрывая внутреннюю реализацию.

Важным свойством компонентного подхода является повторное использованиекомпонентов. Однажды созданный компонент может применяться в различных системах или проектах без необходимости его переписывания, что ускоряет процесс разработки и снижает затраты. Кроме того, использование стандартных интерфейсов позволяет легко заменять или обновлять отдельные компоненты без изменения остальной системы.

Компонентный подход способствует гибкости программного обеспечения, так как компоненты могут легко комбинироваться, настраиваться или дополняться. Это особенно важно при разработке сложных систем, где требуется возможность масштабирования или добавления новой функциональности.

**10. Перечислите основные моменты компонентного подхода.**

Основные моменты компонентного подхода к разработке ПО включают следующие аспекты:

- Компоненты как независимые единицы. Компоненты представляют собой самостоятельные модули с четко определенными интерфейсами, что позволяет им функционировать независимо друг от друга.

- Повторное использование. Компоненты можно повторно использовать в разных проектах и приложениях, что сокращает время разработки и повышает эффективность.

- Интерфейсы. Каждый компонент имеет четко определенные интерфейсы для взаимодействия с другими компонентами. Это облегчает интеграцию и замену компонентов без изменения всей системы.

- Составные системы. Компоненты могут комбинироваться для создания более сложных систем, что позволяет модульно наращивать функциональность приложения.