|  |  |  |
| --- | --- | --- |
|  | **Федеральное государственное бюджетное образовательное**  **учреждение высшего образования**  **«Пензенский государственный университет»**  **(ФГБОУ ВО «Пензенский государственный университет»)**  **Нижнеломовский филиал ФГБОУВО**  **«Пензенский государственный университет»**  **(НлФ ФГБОУ ВО «ПГУ»)** |  |

**Практическая работа №1**

**по дисциплине:** «Поддержка и тестирование программных модулей»

на тему: «Выявление ошибок и причин их появления»

Выполнили:

студент группы 23НФПО

Кириллов А.О.

Веденяпин Д.А.

Мосман А.П.

Принял:

Преподаватель

Кайгородова В.О.

Нижний Ломов, 2025 г.

**Цель работы:** изучить проблематику создания сложной программной системы в отношении к разрабатываемой ИС.

**Контрольные вопросы:**

1. **Управление процессом разработки программного обеспечения** — это **организация, планирование, контроль и координация всех этапов** создания софта, начиная от идеи и до поддержки готового продукта. Похоже на искусство превращения хаоса в структурированную систему, где каждый элемент находится под контролем. По данным McKinsey за 2023 год, организации с отлаженными процессами управления разработкой демонстрируют на 42% большую вероятность завершения проектов в срок и в рамках бюджета.

Основные аспекты управления процессом разработки:

1. Планирование: определяются цели проекта (что именно нужно разработать, оцениваются сроки, риски и бюджет, выбирается подход (**Agile, Scrum, Waterfall, Kanban** и т.д.), составляется roadmap и спринты);
2. Постановка задач: все задачи разбиваются на маленькие подзадачи, каждой задаче присваивается приоритет, исполнитель и срок.
3. Управление командой: определить кто чем занимается (frontend, backend, тестировщики, дизайнеры), следить, чтобы не было конфликтов и простоев, обратная связь, мотивация, ревью кода;
4. Контроль качества: автоматическое и ручное тестирование, проверка кода другими разработчиками, CI/CD процессы (автоматическая сборка, проверка и деплой);
5. Оценка и управление рисками: оценить, где могут быть проблемы, создать запасной план на случай провала спринта, нехватки времени или отпуска разработчика.
6. Документация: документирование кода, API, архитектуры, бизнес-логики.

Ключевые метрики и инструменты мониторинга разработки:

1. Метрики производительности команды: Velocity (скорость команды) — объем работы, выполняемый за итерацию, Lead Time — время от начала работы над задачей до её завершения, CycIe Time — время активной работы над задачей, Throughput — количество завершенных задач за период времени;
2. Метрики качества: Defect Density — количество дефектов на единицу кода, Defect Escape Rate — процент дефектов, не обнаруженных до релиза, Test Coverage — процент кода, покрытый автоматическими тестами, Mean Time to Repair (MTTR) — среднее время исправления дефекта;
3. Метрики доставки ценности: Time to Market — время от идеи до выпуска продукта, Release Frequency — частота выпуска новых версий, Feature Usage — степень использования фунщиональности пользователями, Customer Satisfaction Score — уровень удовлетворенности клиентов.
4. Гибкость программного обеспечения — это **свойство программной системы адаптироваться к изменениям** без значительных затрат ресурсов, времени и качества. Это один из важнейших показателей **качества ПО**, особенно в условиях быстро меняющихся требований. По мнению Джейкобсона, основным фактором гибкости является частота изменений. Разработчики программного обеспечения должны быстро реагировать, чтобы адаптироваться к быстрым изменениям, которые описывает Джейкобсон. Но гибкость — это больше, чем эффективная реакция на изменения.

Основные характеристики гибкого ПО:

1. Модульность: программа состоит из независимых модулей, которые легко заменить или доработать;
2. Расширяемость: возможность добавления новых функций без изменения, уже работающих;
3. Переиспользуемость: Компоненты можно использовать в других проектах без переписывания;
4. Адаптируемость: легко адаптируется под разные платформы/устройства;
5. Конфигурируемость: Поведение системы можно изменить через конфигурации, не лазя в код;
6. Инкапсуляция: Детали реализации скрыты — другие части системы зависят только от интерфейса.

Гибкость применима к любому методу разработки программного обеспечения. Однако для этого важно, чтобы метод был разработан таким образом, чтобы команда проекта могла адаптировать задачи и корректировать их, проводить разработку в соответствии с гибким подходом к разработке, исключать наиболее важные рабочие продукты и поддерживать их в актуальном состоянии, а также использовать прогрессивную стратегию доставки, которая позволяет как можно быстрее предоставлять клиентам рабочие пакеты в соответствии с типом продукта и операционной средой.

1. Поведение программных систем описывается как **совокупность реакций системы на входные воздействия**, которые определяют, **что делает система при определённых условиях и как она это делает**. Такое описание может быть формальным (на уровне моделей, диаграмм) или неформальным (на уровне требований и документации).

Она используется:

1. На этапе проектирования – чтобы зафиксировать, как должна работать система;
2. Во время разработки – как ориентир для реализации логики;
3. При тестировании – чтобы проверить, соответствует ли фактическое поведение ожидаемому.

Диаграммы для описания поведения:

1. Диаграмма состояний: представляет жизненный цикл объекта в виде конечного автомата (показывает разрешенные состояния, переходы и события, которые влияют на эти переходы;
2. Диаграмма последовательности: моделирует взаимодействие объектов на основе временной последовательности, показывает, как одни объекты взаимодействуют с другими в конкретном сценарии;
3. Диаграмма деятельности: графическое представление рабочих процессов, включает шаги в выполнении процесса и переходы между ними.
4. Диаграмма прецедентов: отображает взаимодействие пользователей, подсистем или устройств с системой через выполнение определенных действий.

Поведение ПС описывается с помощью **диаграмм, моделей, сценариев, кода** и других методов, чтобы обеспечить правильную реализацию и предсказуемость работы ПО.

1. **Сложность программного обеспечения (ПО)** — это совокупность характеристик, которые определяют, насколько **трудно понять, разработать, протестировать, сопровождать или изменить** программную систему (**насколько трудно разобраться в программе**, внести изменения, не сломав её, и как легко её поддерживать и развивать в будущем).

Основные виды сложности ПО:

1. Цикломатическая сложность (логическая): показывает количество независимых логических путей в программе, чем больше условий (if, for, while, switch) – тем выше сложность, используется для оценки трудоемкости тестирования и чтения кода;
2. Структурная сложности: определяет, насколько легко понять код человек;
3. Алгоритмическая сложность: оценивает временные и пространственные затраты алгоритма, чем сложнее алгоритм, тем труднее его реализовать и отлаживать;
4. Коммуникационная и организационная: чем больше участников разработки, тем сложнее согласовывать изменения, вести документацию, следить за зависимостями между частями ПО.