DOI 10.37882/2223-2966.2023.9-2.17

КЛАССИФИКАЦИЯ, КАТЕГОРИИ И УРОВНИ СЛОЖНОСТИ ПРОГРАММНЫХ ОШИБОК: ОТ ПРОЕКТИРОВАНИЯ ДО ЭКСПЛУАТАЦИИ[1]

CLASSIFICATION, CATEGORIES, AND LEVELS OF COMPLEXITY OF SOFTWARE ERRORS: FROM DESIGN TO OPERATION

D. Makarov
A. Tsaregorodtsev

Summary. This article conducts research with the aim of systematizing and classifying software errors to simplify the process of their identification, analysis, and resolution. The main goal is to find answers to the following questions: what types and categories of software errors exist, what are their main characteristics and causes, and what are the methods for detecting and correcting various types of errors. To conduct the study, an analysis of existing sources of literature, articles, and documentation related to software development was carried out. The experience of developers and experts in this field was also analyzed [1–7].

During the research, various classifications and categories of software errors were identified and described, including errors by nature, impact, and primary causes; errors by development stages; errors by system impact; errors by type; errors by scope of influence; errors by cause; errors by visibility; errors by methods of detection and correction; and errors by levels of complexity in identification and resolution.

As a result of the study, it was determined that software errors can be divided into different categories and types depending on various aspects, such as the stage of development, impact on the system, type of error, and other parameters. This helps developers better understand and more quickly identify the causes of errors, as well as determine the most effective methods for their resolution.

Keywords: error systematization, error identification, code analysis, testing methods, development stages, types of errors, level of complexity, impact on the system, development processes.

Макаров Дмитрий Александрович
Московский государственный лингвистический университет
MakarovPostOffice@yandex.ru
Царегородцев Анатолий Валерьевич
Доктор технических наук, профессор, Финансовый университет при Правительстве Российской Федерации, Москва
AnVTsaregorodtsev@fa.ru

Аннотация. В данной статье проведено исследование с целью систематизации и классификации программных ошибок для упрощения процесса их идентификации, анализа и устранения. Основной целью было найти ответы на следующие вопросы: какие существуют виды и категории программных ошибок, каковы их основные характеристики и причины возникновения, а также каковы методы обнаружения и исправления различных типов ошибок. Для этого был проведен анализ существующих источников литературы, статей и документации, связанных с разработкой программного обеспечения. Также был проанализирован опыт разработчиков и экспертов в данной области [1–7].

В ходе исследования были выявлены и описаны различные классификации и категории программных ошибок, включая ошибки по характеру, воздействию и основным причинам возникновения, ошибки по стадиям разработки, по воздействию на систему, по типу ошибки, по области влияния, по причине возникновения, по видимости, по способу обнаружения и исправления, а также по уровню сложности выявления и устранения.

В результате исследования было определено, что программные ошибки могут быть разделены на различные категории и типы в зависимости от разных аспектов, таких как стадия разработки, влияние на систему, вид ошибки и других параметров. Это поможет разработчикам лучше понять и быстрее находить причины возникновения ошибок, а также определять наиболее эффективные методы их устранения.

Ключевые слова: систематизация ошибок, идентификация ошибок, анализ кода, методы тестирования, стадии разработки, типы ошибок, уровень сложности, влияние на систему, процессы разработки.

Введение

В современном цифровом мире программное обеспечение является неотъемлемой частью практически всех сфер человеческой деятельности. Однако, несмотря на значительные технологические достижения, проблема программных ошибок остается актуальной. Эти ошибки не только могут привести к сбоям в работе информационных систем, но и стать причиной значительных финансовых потерь и даже угрозы жизни в критически важных приложениях. Поэтому исследование категорий и уровней сложности программных ошибок имеет огромное значение для повышения надежности и безопасности сложных систем, особенно критически важных объектов.

Цель данного исследования заключается в систематизации и классификации программных ошибок с целью упрощения процесса их идентификации, анализа

[1] Статья подготовлена по результатам исследований, выполненных за счет бюджетных средств по государственному заданию Финуниверситета

и устранения. Для достижения этой цели были поставлены следующие задачи:

1. Определение основных типов и категорий программных ошибок.
2. Анализ влияния различных видов ошибок на работоспособность системы.
3. Изучение методов обнаружения и исправления ошибок на различных этапах разработки.

Исследование программных ошибок имеет долгую историю, начиная с первых дней разработки программного обеспечения. Однако, несмотря на множество проведенных исследований, до сих пор не существует универсальной системы классификации ошибок, которая бы удовлетворяла потребностям всех разработчиков. В данной статье представлены результаты анализа существующей литературы и опыта разработчиков, что позволяет сделать выводы о наиболее эффективных методах обнаружения и устранения ошибок.

Данное исследование является важным шагом на пути к созданию универсальной системы классификации программных ошибок, что, в свою очередь, будет способствовать повышению качества разрабатываемого программного обеспечения и уменьшению времени и ресурсов, затрачиваемых на исправление ошибок.

Классификация программных ошибок по характеру, воздействию и основным причинам возникновения

Программные ошибки можно разделить на разные категории в зависимости от их характера, влияния и причин возникновения. Основные категории программных ошибок включают:

1. *Уязвимости*: Ошибки, которые могут быть эксплуатированы злоумышленниками для получения несанкционированного доступа, нарушения конфиденциальности, целостности или доступности системы. Уязвимости могут быть результатом слабостей в коде, конфигурации или архитектуре программного обеспечения [2–4].
2. *Недекларируемые возможности*: Непреднамеренные функции или поведение программы, которые не были предусмотрены разработчиками или указаны в документации. Некоторые недекларируемые возможности могут быть эксплуатированы злоумышленниками, другие могут приводить к некорректной работе программы или улучшению ее функционала [2–3].
3. *Ошибки логики и вычислений*: Ошибки, возникающие из-за неправильной реализации алгоритмов, условий, циклов или других элементов логики программы. Эти ошибки могут привести к неправильным результатам, сбоям или непредсказуемому поведению программы [8–9].
4. *Ошибки ввода/вывода и обработки данных:* Ошибки, связанные с неправильным получением, обработкой, хранением или передачей данных между компонентами программы, файлами или внешними системами. Эти ошибки могут привести к потере, искажению или утечке данных [4, 9].
5. *Ошибки в работе с памятью*: Ошибки, связанные с неправильным выделением, освобождением, использованием или защитой памяти. Эти ошибки могут привести к утечкам памяти, повреждению данных или сбоям программы [8–9].
6. *Ошибки производительности*: Ошибки, которые приводят к снижению скорости, масштабируемости или отказоустойчивости программы. Эти ошибки могут возникнуть из-за неправильного использования ресурсов, алгоритмов или структур данных [2, 9].
7. *Ошибки кодирования и стандартов*: Ошибки, связанные с нарушением стандартов кодирования, стиля или соглашений. Эти ошибки могут затруднить понимание, сопровождение и модификацию программного кода, а также привести к возникновению ошибок в работе программы. Например, это может быть использование неправильных имен переменных, форматирование кода, отступов, комментариев и т.д. [1, 8–9]
8. *Ошибки тестирования*: Ошибки, связанные с неполным, некорректным или отсутствующим тестированием программы. Эти ошибки могут привести к упущению дефектов, не выявлению проблем и непредсказуемому поведению программы [2, 9].
9. *Ошибки развертывания и эксплуатации*: Ошибки, которые могут возникнуть при развертывании, установке или эксплуатации программы. Эти ошибки могут привести к неполадкам, сбоям или отказу программы, а также к уязвимостям безопасности [2–4].

Категории программных ошибок могут пересекаться, так как одна ошибка может принадлежать нескольким категориям. Например, ошибка, связанная с недостаточной проверкой входных данных, может привести как к ошибке входных данных, так и к ошибке логики.

Категории и типы ошибок программного обеспечения

Здесь представлено краткое изложение исследованной таксономии ошибок безопасности программного обеспечения, которая включает в себя девять категорий, включая ошибки входных данных, ошибки логики, ошибки в работе с памятью, ошибки производительности, ошибки безопасности, ошибки проектирования, ошибки кодирования, ошибки тестирования и ошибки развертывания и эксплуатации. Каждая из этих категорий включает множество типов ошибок, которые могут

привести к уязвимостям и компрометации безопасности программного обеспечения. Наша цель заключалась в том, чтобы помочь разработчикам программного обеспечения понимать различные категории ошибок и уязвимостей, чтобы они могли предотвращать их возникновение.

1. Ошибки входных данных:

Валидация и санитизация данных:
- — Неправильная проверка ввода.
- — Недостаточная проверка синтаксиса или типа данных.
- — Недостаточная проверка формы.
- — Неправильная нейтрализация специальных символов.

Управление памятью и буферами:
- — Неправильное ограничение операций в пределах буфера памяти.
- — Неправильное завершение строки нулем.
- — Недостаточная проверка размера буфера перед копированием.

Обработка строк и форматирование:
- — Неправильное использование шаблонов форматирования строк.
- — Неправильное использование параметров функций.

Обработка символов и кодировки:
- — Неправильная обработка символов в кодировке, отличной от ASCII.
- — Неправильное использование функций для обработки символов Unicode.
- — Неправильная обработка кодировки Unicode.
- — Неправильная обработка многоязычных символов.

Проверка и обработка внешних данных:
- — Неправильное ограничение ссылок на внешние сущности.
- — Недостаточная проверка данных, полученных от API или других внешних систем.
- — Неправильная обработка ошибок при получении данных от сторонних источников.

Обработка пользовательского ввода:
- — Использование неправильного типа пользовательского ввода.
- — Недостаточная проверка типа передаваемых данных.
- — Недостаточная проверка вводимых данных в SQL-запросах.
- — Недостаточная проверка вводимых данных в командах системной оболочки.

Аутентификация и авторизация:
- — Пропуск проверки аутентификации или авторизации при обработке входных данных.
- — Неправильная обработка данных, связанных с правами доступа и ролями пользователей.

Обработка времени и даты:
- — Недостаточная проверка формата даты и времени в пользовательском вводе.
- — Неправильная обработка часовых поясов и смещений времени.

Обработка файлов и путей:
- — Недостаточная проверка формата и безопасности путей и имен файлов.
- — Неправильная обработка символов в путях и именах файлов.
- — Неограниченная загрузка файла с опасным типом [1, 10].

2. Ошибки логики:

Неправильное понимание требований к программе и/или неправильное проектирование программы:
- — Недостаточная проверка зависимостей и предусловий.
- — Неправильное использование алгоритмов и структур данных.
- — Неверное преобразование типов данных.

Ошибки в условиях и циклах.

Недостаточная обработка исключительных ситуаций.

Неправильное выполнение параллельных задач и синхронизация:
- — Неучтенные условия гонки при доступе к общим ресурсам
- — Неправильное использование примитивов синхронизации

Неправильная работа с функциями и методами.

Неправильное использование библиотечных функций.

Ошибки инициализации переменных и работы с указателями:
- — Использование неинициализированных переменных.
- — Неверная инициализация переменных.
- — Разыменование нулевых или некорректных указателей.
- — Неправильное использование ссылок на временные объекты.

Утечки ресурсов и неправильное управление памятью:
- — Неосвобождение выделенных ресурсов, таких как память или файловые дескрипторы.
- — Недостаточная обработка ошибок при выделении ресурсов.
- — Ошибки выделения и освобождения памяти.
- — Использование устаревших или небезопасных функций работы с памятью.

Ошибки в обработке ошибок и исключений:
- — Игнорирование возможных ошибок и исключений.
- — Неправильное восстановление системы после ошибки или исключения.

Ошибки в обработке ввода-вывода:
— Некорректная обработка данных, полученных от пользователя или из других источников.
— Неправильное форматирование и представление данных при выводе.

Ошибки в работе с временем и датами:
— Некорректная обработка часовых поясов и переходов на летнее/зимнее время.
— Использование устаревших или неточных алгоритмов работы с датами и временем.

Неправильное тестирование и отладка:
— Недостаточное покрытие тестами.
— Игнорирование результатов тестирования или неправильная интерпретация результатов [10-11].

3. Ошибки в работе с памятью:

Проблемы с памятью:
— Неверное выделение памяти.
— Неправильное освобождение памяти.
— Утечки памяти и ресурсов.
— Переполнение буфера или стека.
— Нарушение границ массива.

Проблемы с указателями и доступом к памяти:
— Нарушение прав доступа к памяти и защиты памяти виртуальной машины.
— Неправильная работа с указателями.
— Использование висячих указателей после освобождения памяти.

Проблемы с многопоточностью и синхронизацией:
— Состояние гонки при доступе к памяти.
— Неправильное использование атомарных операций и синхронизации.
— Неправильное использование разделяемой памяти в многопоточных приложениях.

Проблемы с инициализацией и управлением объектами:
— Некорректное управление жизненным циклом объектов.
— Нарушение порядка инициализации переменных и ресурсов.
— Использование неинициализированных переменных.

Проблемы с типами данных и приведением типов:
— Некорректное приведение типов и использование неправильных типов данных.

Проблемы с оптимизацией и архитектурными особенностями:
— Проблемы с выравниванием памяти и кэш-когерентности.

4. Ошибки производительности:

Алгоритмы и структуры данных:
— Неверное использование алгоритмов и структур данных.
— Выбор алгоритмов с неоптимальной сложностью.

Управление ресурсами:
— Неверное управление памятью.
— Неправильное использование кеширования.

Сетевое взаимодействие:
— Неправильное использование сетевых запросов.
— Отсутствие сжатия данных и оптимизации передачи данных.

Многопоточность и многопроцессорность:
— Неправильное использование многопоточности и многопроцессорности.
— Излишняя синхронизация или неправильное использование мьютексов и блокировок.
— Неправильное планирование и распределение задач между потоками и процессами.

Оптимизация кода:
— Неиспользование оптимизаций компилятора.
— Отсутствие профилирования и оптимизации кода.
— Излишний код или дублирование кода.

Взаимодействие с библиотеками и API:
— Неправильное использование сторонних библиотек и API.
— Использование синхронного программирования, когда асинхронное подходит лучше.

Работа с базами данных:
— Отсутствие или неправильное использование индексов в базах данных.

Графика и медиа-ресурсы:
— Отсутствие оптимизации загрузки и отображения изображений и других медиа-ресурсов.
— Отсутствие оптимизации рендеринга графических элементов.

Обработка событий:
— Неправильное управление событиями и обработка событий.

Виртуализация:
— Неправильное использование виртуальных машин и контейнеров [11, 13].

5. Ошибки безопасности:

Управление доступом и аутентификация:
— Недостаточная проверка прав доступа.
— Неправильное управление доступом к данным и ресурсам.
— Недостаточное шифрование данных.
— Неправильная работа с сессиями.
— Ошибки в процессе аутентификации и авторизации.

Обработка данных и пользовательского ввода:
— Неправильная обработка пользовательского ввода.
— Недостаточная обработка внешних данных.
— Недостаточная проверка передаваемых данных.
— Неправильная валидация данных.

Коммуникация и сетевые протоколы:
— Использование незащищенных каналов связи.
— Неправильная обработка сетевых протоколов и запросов.
— Неправильная обработка ошибок в сетевых запросах.
— Неправильная обработка сетевых протоколов.

Компоненты и дополнительные функции:
— Уязвимости в компонентах и библиотеках.
— Уязвимости в дополнительных функциях (API, мобильные приложения, IoT).

Конфигурация и управление инцидентами:
— Ошибки в настройках безопасности сервера и приложения.
— Утечка информации.
— Ошибки в процессе управления инцидентами и инфраструктурой.

Организационный и человеческий факторы:
— Инсайдерские угрозы и ошибки сотрудников [13].

6. Ошибки проектирования

Архитектурные ошибки:
— Недостатки архитектуры.
— Неправильное разделение ответственности между компонентами.
— Недостаточная модульность и расширяемость системы

Ошибки в выборе алгоритмов и структур данных:
— Неправильный выбор алгоритмов и структур данных.

Ошибки в проверке требований и валидации:
— Недостаточная проверка требований и валидации.

Ошибки в документации:
— Недостатки в документации.

Ошибки, связанные с производительностью и ресурсами:
— Неучтенные ограничения производительности и ресурсов.
— Пренебрежение производительностью и оптимизацией.
— Недооценка требований к памяти, CPU и пропускной способности.

Ошибки в обработке ошибок и исключений:
— Недостатки в обработке ошибок и исключений.
— Отсутствие обработки исключений и ошибок.
— Неправильное восстановление после сбоев.

Ошибки в обеспечении безопасности:
— Проблемы с безопасностью.
— Неучтенные риски и уязвимости.
— Отсутствие шифрования и аутентификации, где это необходимо.

Ошибки, связанные с масштабированием и поддержкой:
— Неучтенные проблемы масштабирования и поддержки.
— Сложность поддержки и обновления системы [12].
— Отсутствие планирования на будущее масштабирование.

7. Ошибки кодирования

Ошибки в логике, структуре и функциональности кода:
— Ошибки логики и вычислений.
— Неправильная обработка ошибок и исключительных ситуаций.
— Неправильное использование контрольных структур (циклы, условия, и т.д.).
— Несоответствие кода требованиям по функциональности и бизнес-логике.

Ошибки при работе с внешними ресурсами и инструментами:
— Некорректное использование API и сторонних библиотек.
— Неэффективное использование баз данных и операций с ними.
— Ошибки в работе с пользовательским вводом и валидацией данных.

Ошибки, связанные с проектированием и архитектурой кода:
— Неправильная работа с наследованием и полиморфизмом.
— Неправильное применение шаблонов проектирования.
— Недостаточное разделение ответственности между компонентами системы (нарушение принципов SOLID).
— Нарушение принципов модульности и инкапсуляции.

Ошибки, связанные с производительностью и оптимизацией:
— Недостаточная оптимизация кода и производительности.
— Утечки памяти и неправильное управление ресурсами.
— Ошибки при работе с многопоточностью, безопасностью и кодированием:
— Неправильное использование многопоточности и синхронизации.
— Небезопасное программирование, ведущее к уязвимостям и атакам.
— Ошибки, связанные с кодированием и обработкой символов различных языков (Unicode).

Ошибки, связанные с поддержкой, тестированием и оформлением кода:
— Отсутствие или недостаточное тестирование и проверка кода.
— Непонятный или сложный код, затрудняющий понимание и поддержку.

— Отсутствие или недостаточная документация кода.
— Нарушения стандартов кодирования.
— Неправильное использование глобальных переменных.
— Синтаксические ошибки [13].

8. Ошибки тестирования

Планирование и стратегия тестирования:
— Ошибки в планировании и проектировании тестов.
— Недостаточная автоматизация тестирования.
— Отсутствие четкой стратегии тестирования.
— Недостаточное время на тестирование.
— Ошибки в оценке приоритетов тестирования.

Взаимодействие и коммуникация:
— Неясные или неполные требования к продукту.
— Недостаточное взаимодействие с разработчиками.
— Игнорирование обратной связи от пользователей.

Технические аспекты тестирования:
— Проблемы с регрессионным тестированием.
— Проблемы с тестовыми данными.
— Некорректное использование инструментов тестирования.
— Проблемы с выполнением тестов.
— Ошибки в анализе и интерпретации результатов тестирования.
— Неправильная настройка среды тестирования.
— Несоблюдение правил построения и развертывания.

Обучение и документация:
— Пренебрежение документацией.
— Пренебрежение непрерывным обучением.

Специфические виды тестирования:
— Недооценка важности тестирования безопасности.
— Неадекватное тестирование краевых случаев.
— Игнорирование нагрузочного тестирования.
— Пренебрежение тестированием пользовательского опыта (UX).
— Отсутствие тестирования на различных платформах и устройствах.
— Игнорирование тестирования интеграции.
— Недостаточное покрытие тестами.
— Игнорирование тестирования внешних зависимостей.

Процессы и стандартизация:
— Отсутствие контроля качества и процессов управления ошибками.
— Отсутствие систематического ручного тестирования.
— Непостоянство в процессе тестирования.
— Отсутствие контроля над изменениями.
— Отсутствие мониторинга и анализа результатов тестирования [10-11].

9. Ошибки развертывания и эксплуатации

Настройка и конфигурация:
— Неправильная настройка серверов и сетевых настроек.
— Недостатки в управлении конфигурацией.
— Проблемы с установкой и настройкой ПО.

Безопасность и соответствие:
— Недостаточная безопасность и настройка мер безопасности.
— Несоблюдение стандартов и регулятивных требований.

Масштабирование и производительность:
— Проблемы с масштабированием и отказоустойчивостью системы.
— Недостаточное планирование мощности и ресурсов.

Обновления и управление изменениями:
— Проблемы с обновлениями и патчами.
— Недостаточное управление изменениями.

Мониторинг и восстановление:
— Проблемы с мониторингом и резервным копированием.
— Отсутствие стратегии восстановления после сбоев.

Организационные аспекты:
— Отсутствие автоматизации процессов.
— Неэффективная документация и обучение персонала.
— Слабая коммуникация между командами [6, 10, 14].

Классификации ошибок в программном обеспечении

По стадиям разработки:
— Ошибки проектирования: возникают на этапе проектирования архитектуры или планирования программного продукта.
— Ошибки кодирования: возникают во время написания исходного кода.
— Ошибки тестирования: возникают при выполнении тестов и анализе результатов.
— Ошибки развертывания: возникают при установке и настройке программного обеспечения.
— Ошибки эксплуатации: возникают в процессе работы программы на этапе поддержки и обслуживания.

По воздействию на систему:
— Критические ошибки: приводят к сбоям, потере данных или нарушению безопасности.

— Серьезные ошибки: ограничивают функциональность системы или вызывают некорректное поведение.
— Мелкие ошибки: влияют на отдельные функции или элементы пользовательского интерфейса, но не мешают работе системы в целом.

По типу ошибки:
— Синтаксические ошибки: нарушения правил языка программирования [8, 15].
— Логические ошибки: некорректная логика программы или алгоритма [16].
— Ошибки времени выполнения: возникают при запуске программы из-за неправильной работы с памятью, файлами, сетью и другими ресурсами [8, 15].

По области влияния:
— Ошибки безопасности: приводят к уязвимостям и возможности атак злоумышленников [16].
— Ошибки производительности: снижают скорость работы программы или вызывают чрезмерное потребление ресурсов [17].
— Ошибки совместимости: вызывают проблемы при работе с другими программами, операционными системами или устройствами.
— Ошибки пользовательского интерфейса: приводят к проблемам в восприятии и взаимодействии с программой со стороны пользователей [15].

По причине возникновения:
— Ошибки человеческого фактора: вызваны недостатками знаний, опыта или внимания разработчика.
— Ошибки процессов и методологий: возникают из-за недостаточно четко определенных или неэффективных процессов разработки [14, 18].
— Ошибки инструментальных средств: связаны с некорректной работой инструментов разработки, тестирования или сборки программного обеспечения.

По видимости:
— Явные ошибки: проявляются непосредственно при работе программы, заметны пользователям или разработчикам.
— Скрытые ошибки: остаются незамеченными в обычных условиях работы программы, проявляются только при определенных комбинациях входных данных или состояний системы.

По способу обнаружения и исправления:
— Ошибки, обнаруженные в процессе разработки: находятся разработчиками на этапе написания кода, тестирования или код-ревью.
— Ошибки, обнаруженные пользователями: выявляются пользователями программы после ее выпуска и требуют выпуска обновлений или патчей.
— Классификация ошибок может быть полезна для анализа и улучшения процессов разработки, а также для определения приоритетов при исправлении обнаруженных проблем. Однако стоит учитывать, что в реальных проектах многие ошибки могут сочетать в себе несколько характеристик из разных классификаций.

Классификация ошибок программного обеспечения по уровню сложности их выявления и устранения

1. Низкий уровень сложности:
— Синтаксические ошибки: Ошибки, связанные с нарушением правил языка программирования, которые обычно легко обнаруживаются компиляторами или интерпретаторами.
— Ошибки стиля кодирования: Нарушения принятых стандартов и рекомендаций по написанию кода, которые можно обнаружить с помощью инструментов статического анализа кода.

2. Средний уровень сложности:
— Логические ошибки: Ошибки в алгоритмах и структурах данных, которые могут привести к неправильной работе программы. Выявление таких ошибок обычно требует тестирования, отладки или код-ревью [20–22].
— Ошибки обработки ошибок и исключений: Неправильная обработка ошибочных ситуаций, которая может привести к непредсказуемому поведению программы или потере данных.

3. Высокий уровень сложности:
— Проблемы производительности: Ошибки, связанные с медленной работой программы или излишним использованием ресурсов. Они могут быть результатом неправильного выбора алгоритмов, структур данных или неправильной оптимизации кода.
— Уязвимости безопасности: Ошибки, которые могут быть эксплуатированы злоумышленниками для получения несанкционированного доступа или нарушения конфиденциальности, целостности и доступности системы. Выявление и устранение таких ошибок требует специальных знаний и инструментов.

4. Очень высокий уровень сложности:
— Архитектурные и дизайнерские ошибки: Ошибки, связанные с проектированием системы, такие как неправильное разделение ответственности между компонентами, недостаточная модульность и расширяемость. Обнаружение и исправление таких ошибок может потребовать значительных изменений в коде и архитектуре системы [23].

— Недекларируемые возможности: Непреднамеренные функции или поведение программы, которые не были предусмотрены разработчиками или указаны в документации. Они могут быть результатом сложного взаимодействия между компонентами системы, ошибками разработчиков или неправильным пониманием требований. Выявление и устранение недекларируемых возможностей может потребовать глубокого анализа кода, тестирования и ревью процессов разработки.

5. Критический уровень сложности:

— Ошибки совместимости: Ошибки, связанные с неправильным взаимодействием программного обеспечения с другими системами, библиотеками, стандартами или протоколами. Они могут привести к неправильной работе программы, потере данных или нарушению безопасности и могут потребовать значительных изменений в коде для их устранения.
— Ошибки в развертывании и эксплуатации: Ошибки, связанные с установкой, настройкой, мониторингом и обслуживанием программного обеспечения. Они могут привести к проблемам с доступностью, масштабируемостью и отказоустойчивостью системы, и их устранение может потребовать изменений в инфраструктуре, процессах и инструментах.

Стоит отметить, что границы между этими категориями ошибок могут быть размытыми, и некоторые ошибки могут относиться к нескольким категориям одновременно. Кроме того, сложность выявления и устранения ошибок может варьироваться в зависимости от контекста, опыта команды разработчиков и доступных инструментов.

Уровень сложности ошибок, который мы предложили, определен на основе нашего понимания и обобщения различных аспектов ошибок в программном обеспечении. Он основан на таких характеристиках и критериях, как:

1. Влияние на программное обеспечение: ошибки с большим влиянием на работу программы, безопасность или стабильность обычно считаются более сложными.
2. Сложность выявления: ошибки, которые трудно обнаружить и требуют значительных усилий, знаний или опыта для их выявления, считаются более сложными.
3. Сложность исправления: ошибки, которые затрудняют исправление, из-за сложных взаимосвязей в коде, архитектурных ограничений или недостатка знаний, считаются более сложными.
4. Зависимость от контекста: ошибки, которые зависят от специфических контекстов использования, среды или технологии, могут быть более сложными из-за необходимости разбираться во всех деталях и особенностях.
5. Неоднозначность: ошибки, которые имеют множество потенциальных причин, источников или сценариев взаимодействия, могут быть более сложными из-за необходимости учитывать все возможные аспекты.
6. Вероятность возникновения: ошибки, которые возникают редко или только при определенных наборах условий, могут считаться более сложными, поскольку их труднее воспроизвести и выявить.
7. Влияние на пользователя: ошибки, которые сильно сказываются на пользовательском опыте или могут привести к потере данных, могут считаться более сложными, так как они могут повлечь за собой серьезные последствия для пользователей и требуют оперативного решения.
8. Воспроизводимость: ошибки, которые сложно воспроизвести или проявляются непоследовательно, могут быть более сложными для анализа и исправления, так как это затрудняет точное определение причины и механизма возникновения.
9. Коммуникация и координация: ошибки, которые требуют координации между разными командами, отделами или сторонними разработчиками, могут быть более сложными из-за необходимости согласования действий и обмена информацией.
10. Требования к документации и процессам: ошибки, которые затрагивают существующую документацию или процессы, могут быть более сложными, так как их исправление может потребовать обновления документации или изменения процессов разработки.

Предложенный список критериев предназначен для оценки сложности ошибок в программном обеспечении и включает в себя разнообразные аспекты, которые могут влиять на сложность ошибок. Он предназначен для специалистов и команд разработчиков для лучшего оценивания и приоритизации ошибок, а также определения необходимых ресурсов и усилий для их исправления.

Заключение

В ходе проведенного исследования были систематизированы и классифицированы программные ошибки, что позволит упростить их идентификацию, анализ и устранение, и в конечном итоге повысить качество разрабатываемого программного обеспечения и уменьшить время, затрачиваемое на выявление и исправление ошибок. Авторы постарались ответить на вопросы о различных категориях и типах программных ошибок, их характеристиках и причинах возникновения, а также методах обнаружения и исправления.

Важность такой классификации заключается в том, что она является основой для создания эффективных методик тестирования, процессов разработки и профилактики возникновения ошибок. Таким образом, результаты данного исследования могут быть полезными для разработчиков, тестировщиков, а также специалистов в области управления проектами и процессами разработки программного обеспечения.

Программные ошибки могут быть классифицированы по различным аспектам, таким как характер, воздействие, причины возникновения, стадии разработки, воздействие на систему, тип ошибки, область влияния и другие параметры.

Разработчики и тестировщики могут использовать классификацию ошибок для ускорения процесса выявления и устранения проблем, определения наиболее эффективных методов их решения и снижения рисков возникновения новых ошибок.

Знание классификаций и типов ошибок поможет лучше понять их влияние на программное обеспечение, сложность выявления и исправления, зависимость от контекста и неоднозначность.

В перспективе, результаты этой статьи могут послужить основой для создания инструментов по автоматическому обнаружению и исправлению ошибок, а также совершенствованию существующих методик тестирования и процессов разработки.

ЛИТЕРАТУРА

1. Martin, R.C. Clean Code: A Handbook of Agile Software Craftsmanship. Prentice Hall, 2008. 464 p. ISBN: 978-0132350884.
2. Kim, G., Debois, P., Willis, J., Humble, J. The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations. IT Revolution Press, 2016. 480 p. ISBN: 978-1942788003.
3. Johnsson, D.B., Deogun, D., Sawano, D. Secure by Design. Manning Publications, 2018. 256 p. ISBN: 978-1617294358.
4. Adkins, H., Beyer, B., Blankinship, P. Building Secure and Reliable Systems: Best Practices for Designing, Implementing, and Maintaining Systems. Google, 2020. 228 p. ISBN: 978-1492083122.
5. Zalewski, M. The Tangled Web: A Guide to Securing Modern Web Applications. No Starch Press, 2011. 320 p. ISBN: 978-1593273880.
6. Fowler, M. Refactoring: Improving the Design of Existing Code. Addison-Wesley, 2018. 448 p. ISBN: 978-0134757599.
7. Блохина О., Царегородцев А.В. Классификация параметров, влияющих на критичность инцидента информационной безопасности // В книге: Collegium Linguisticum — 2019. Тезисы докладов ежегодного научного студенческого общества МГЛУ. 2019. s. 286.
8. McConnell, S. Code Complete: A Practical Handbook of Software Construction. Microsoft Press, 2004.
9. Sommerville, I. Software Engineering. Addison-Wesley, 2015.
10. Hunt, A., Thomas, D. The Pragmatic Programmer: Your Journey to Mastery. Addison-Wesley, 2019. 352 p. ISBN 978-0-13-595705-9.
11. Bloch, J. Effective Java. Addison-Wesley, 2018. 416 p. ISBN 978-0-13-468599-1.
12. Desikan, S., Ramesh, G. Software Testing: Principles and Practices. Pearson Education, 2006. 580 p. ISBN 978-81-7758-165-2.
13. Humble, J., Farley, D. Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation. Addison-Wesley, 2010. 512 p. ISBN 978-0-321-60191-9.
14. Gamma, E., Helm, R., Johnson, R., Vlissides, J. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, 1994. 395 p. ISBN 978-0-201-63361-0.
15. Myers, G.J., Sandler, C., Badgett, T. The Art of Software Testing. Wiley, 2011. 256 p. ISBN 978-1-118-20605-0.
16. Agans, D.J. Debugging: The 9 Indispensable Rules for Finding Even the Most Elusive Software and Hardware Problems. AMACOM, 2006. 192 p. ISBN 978-0-8144-0889-8.
17. Brooks, F.P. The Mythical Man-Month: Essays on Software Engineering. Addison-Wesley, 1995. 336 p. ISBN 978-0-201-83595-3.
18. Martin, R.C. Clean Architecture: A Craftsman's Guide to Software Structure and Design. Prentice Hall, 2017. 432 p. ISBN 978-0-13-449416-6.
19. Sipser, M. Introduction to the Theory of Computation. Cengage Learning, 2012. 482 p. ISBN 978-1-133-18779-0.
20. Zeller, A. Why Programs Fail: A Guide to Systematic Debugging. Elsevier, 2009. 400 p. ISBN 978-1-55860-866-0.
21. Kaner, C., Bach, J., Pettichord, B. Lessons Learned in Software Testing: A Context-Driven Approach. Wiley, 2001. 320 p. ISBN 978-0-471-08112-1.
22. Spolsky, J. Joel on Software. Apress, 2004. 384 p. ISBN 978-1-59059-389-9.
23. Rubin, K. Essential Scrum: A Practical Guide to the Most Popular Agile Process. Addison-Wesley, 2012. 500 p. ISBN 978-0-13-704329-3.

Журнал «Современная наука: актуальные проблемы теории и практики»