ORES
Научно-практический электронный журнал
Оригинальные исследования (ОРИС), *ISSN 2222-0402*
www.ores.su
original-research.ru

УДК 004.415

ИССЛЕДОВАНИЕ МЕТОДОВ ПРЕДСКАЗАНИЯ ДЕФЕКТОВ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ НА ОСНОВЕ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

Чан Пэнсян,
Степень бакалавра
Национальный исследовательский Томский государственный университет
xiangxiang3451@yandex.ru

Аннотация

В данной работе исследуются современные методы предсказания дефектов программного обеспечения (ПО) с использованием искусственного интеллекта (ИИ). Рассматриваются алгоритмы машинного и глубокого обучения для анализа кода, выявления уязвимостей и прогнозирования ошибок. Цель исследования – сравнение методов предсказания дефектов ПО и выявление наиболее эффективных. Используются методы анализа литературы, сравнения алгоритмов и статистической оценки точности.

Задачи включают обзор традиционных и ИИ-методов, анализ их эффективности, сравнительную оценку точности и разработку рекомендаций. Представлены аналитические данные и примеры применения.

Ключевые слова: дефекты программного обеспечения, машинное обучение, искусственный интеллект, прогнозирование ошибок, анализ кода.

STUDY OF SOFTWARE DEFECT PREDICTION METHODS BASED ON ARTIFICIAL INTELLIGENCE

Chang Pengxiang,
Bachelor's degree
National Research Tomsk State University
xiangxiang3451@yandex.ru

ABSTRACT

This paper investigates modern methods of predicting software (software) defects using artificial intelligence (AI). Machine learning and deep learning algorithms for code analysis, vulnerability detection and bug prediction are considered. The purpose of the study is to compare methods of software defect prediction and identify the most effective ones. Methods used are literature analysis, algorithm comparison and statistical evaluation of accuracy.

Objectives include reviewing traditional and AI methods, analyzing their performance, comparing accuracy, and developing recommendations. Analytical data and application examples are presented.

Keywords: software defects, machine learning, artificial intelligence, error prediction, code analysis.

Дефекты программного обеспечения представляют собой ошибки, несоответствия или недостатки, которые могут привести к некорректному функционированию системы. Они могут быть вызваны человеческими ошибками, недостатками проектирования, проблемами с реализацией или внешними факторами. Предсказание дефектов в ПО играет ключевую роль в обеспечении качества, снижении затрат на исправление ошибок и повышении надежности программных продуктов.

Существует несколько подходов к классификации дефектов ПО, основанных на различных аспектах их возникновения и проявления. По происхождению дефекты могут быть связаны с ошибками проектирования, возникающими из-за некорректных архитектурных или алгоритмических решений, ошибками реализации, возникающими при написании кода, ошибками интеграции, возникающими при взаимодействии разных модулей системы, а также ошибками конфигурации, связанными с неправильными параметрами среды выполнения [1].

По уровню проявления дефекты могут быть синтаксическими, то есть связанными с нарушением правил языка программирования, семантическими, приводящими к логическим ошибкам, ошибками времени выполнения, возникающими только в процессе работы программы, а также ошибками производительности, вызывающими снижение эффективности работы системы.

По влиянию на систему дефекты могут быть критическими, приводящими к полной неработоспособности ПО, серьезными, влияющими на функциональность и вызывающими значительные сбои, средними, создающими неудобства для пользователей, но не мешающими выполнению основных задач, и незначительными, имеющими минимальное влияние на пользовательский опыт и работоспособность системы.

Таблица 1. Классификация дефектов ПО [2]

Класс дефекта	Описание	Пример
Синтаксические ошибки	Ошибки, связанные с некорректным синтаксисом кода	Отсутствующая скобка в коде
Логические ошибки	Ошибки в алгоритмах, приводящие к некорректным результатам	Неправильный расчет налогов в бухгалтерской программе
Ошибки производительности	Проблемы, связанные с медленной работой программы	Высокое потребление памяти при обработке данных

Дефекты программного обеспечения могут возникать по множеству причин, включая человеческие ошибки, недостатки методологий разработки и внешние факторы. Человеческие ошибки часто обусловлены сложностью кода, нехваткой опыта или недостаточной внимательностью разработчиков. Недостатки методологий разработки связаны с отсутствием четких процессов тестирования, неэффективным управлением требованиями или недостаточным контролем качества. Внешние факторы включают аппаратные сбои, изменения в окружении и влияние стороннего программного обеспечения.

Понимание классификации и причин возникновения дефектов является ключевым шагом в разработке эффективных методов их предсказания, что позволяет повысить качество программного обеспечения и минимизировать издержки на исправление ошибок.

Статический анализ кода заключается в исследовании исходного кода без его выполнения. Он позволяет обнаружить синтаксические ошибки, потенциальные уязвимости и нарушения кодирования еще на ранних этапах разработки. Основными инструментами статического анализа являются линтеры (например, Pylint, ESLint), анализаторы кода (например, SonarQube) и инструменты для обнаружения уязвимостей (например, Checkmarx). Преимущества статического анализа включают возможность раннего выявления дефектов, автоматизированность процесса и сравнительно низкую стоимость. Однако данный метод не способен выявлять ошибки, которые проявляются только во время выполнения программы, а также не всегда эффективно справляется с анализом сложных логических зависимостей в коде[3].

Динамический анализ кода предполагает выполнение программы с целью выявления дефектов, которые невозможно обнаружить статическим анализом. Он включает тестирование на основе входных данных, профилирование производительности, отладку и анализ выполнения программы. Динамический анализ может выявлять ошибки времени выполнения, утечки памяти, проблемы с многопоточностью и другие дефекты, возникающие в процессе работы ПО. К его недостаткам относятся высокая стоимость, сложность реализации и необходимость наличия работающего прототипа системы.

Оба подхода широко применяются в индустрии и дополняют друг друга. Однако они имеют ограничения, связанные с высокой трудоемкостью, зависимостью от качества тестовых сценариев и невозможностью эффективного анализа больших объемов кода без автоматизации.

С развитием машинного обучения и ИИ появились интеллектуальные методы предсказания дефектов, которые анализируют данные о программном коде, выявляют закономерности и прогнозируют вероятность возникновения ошибок. К основным направлениям интеллектуального анализа относятся методы машинного обучения, глубинного обучения и обработка естественного языка (NLP) для анализа кода. Глубинное обучение использует многослойные нейронные сети для анализа кода. Такие модели способны выявлять сложные зависимости и более точно прогнозировать дефекты. Например, рекуррентные нейронные сети (RNN) и трансформеры могут анализировать последовательности кода, выявляя потенциальные ошибки на уровне структуры программы. Однако глубокие нейросети требуют больших объемов данных и значительных вычислительных ресурсов.

Обработка естественного языка (NLP) применяется для анализа текстовых данных в коде, включая комментарии, документацию и названия переменных. Использование NLP позволяет учитывать контекст программирования и улучшать точность предсказания дефектов.

Традиционные методы предсказания дефектов обладают высокой интерпретируемостью, так как позволяют разработчикам напрямую анализировать код и выявлять ошибки. Они хорошо подходят для статического и динамического анализа, но требуют значительных временных затрат и человеческого участия. В то же время интеллектуальные методы обеспечивают автоматизированное предсказание дефектов, анализируя большие объемы данных и выявляя скрытые закономерности.

Интеллектуальные методы способны обрабатывать сложные зависимости и прогнозировать ошибки с высокой точностью, но они требуют наличия качественных данных для обучения моделей. Кроме того, машинное обучение и ИИ могут сталкиваться с

проблемами интерпретируемости результатов, что усложняет их внедрение в процесс разработки ПО.

Таблица 2. Сравнение методов предсказания дефектов [4]

Метод	Достоинства	Недостатки
Статический анализ	Высокая скорость обнаружения ошибок	Низкая точность при сложных ошибках
Динамический анализ	Возможность тестирования в реальном времени	Высокие вычислительные затраты
Машинное обучение	Самообучаемость, высокая точность	Требует больших данных для обучения

Использование искусственного интеллекта для предсказания дефектов программного обеспечения открывает новые возможности в обеспечении качества кода, позволяя автоматизировать процесс выявления ошибок, анализировать большие объемы данных и предсказывать потенциальные уязвимости. В последние годы машинное обучение и глубинные нейросетевые модели продемонстрировали высокую эффективность в анализе программного кода, что делает их перспективными инструментами для автоматического обнаружения дефектов. Однако, несмотря на явные преимущества, внедрение ИИ в предсказание дефектов сопровождается рядом сложностей, включая необходимость наличия качественных данных, интерпретируемость моделей, вычислительные затраты и адаптацию к различным языкам программирования.

ИИ также способен анализировать изменения в коде и оценивать их влияние на работоспособность системы. Например, при внесении изменений в кодовую базу можно автоматически определить, насколько велика вероятность появления новых дефектов, и своевременно предупредить разработчиков. Такой подход позволяет минимизировать ошибки, возникающие при обновлениях программного обеспечения, и сократить время на исправление багов. Интересным направлением является применение обработки естественного языка для анализа документации, комментариев к коду и отчетов об ошибках. Использование NLP-моделей помогает выявлять несоответствия в требованиях, улучшать описание кода и автоматически анализировать сообщения об ошибках, что снижает вероятность дефектов, вызванных человеческим фактором.

С развитием технологий сбора и обработки данных компании получают возможность использовать большие объемы информации о предыдущих ошибках, что открывает перспективы для построения более точных моделей предсказания дефектов. Исторические данные о багах, исправлениях и метриках качества кода позволяют обучать модели, которые способны прогнозировать вероятные дефекты в новых проектах.

Дополнительной сложностью является необходимость адаптации моделей к разным языкам программирования. Каждый язык имеет свои особенности, и модели, обученные на одном языке, могут показывать низкую точность при анализе кода, написанного на другом языке. Это требует либо разработки универсальных алгоритмов, либо обучения отдельных моделей для каждого языка, что увеличивает затраты на внедрение ИИ.

В перспективе ИИ станет неотъемлемой частью процесса разработки ПО, помогая автоматизировать предсказание дефектов и минимизировать их влияние на конечный продукт. Совмещение традиционных методов анализа с интеллектуальными алгоритмами позволит создать более точные и надежные системы, способные эффективно выявлять дефекты и обеспечивать высокое качество программного кода.

В работе рассмотрены современные методы предсказания дефектов программного обеспечения с использованием искусственного интеллекта. Проанализированы ключевые подходы машинного обучения и глубокого обучения для обнаружения ошибок и

уязвимостей в коде. Представлена сравнительная характеристика традиционных и интеллектуальных методов предсказания дефектов. Исследование показало, что применение методов ИИ позволяет значительно повысить точность обнаружения ошибок, однако требует большого количества данных и вычислительных ресурсов. В дальнейшем перспективным направлением является разработка гибридных систем предсказания дефектов, объединяющих лучшие свойства различных методов.

Список литературы:

1. Бевзенко, С. А. Исследование методов автоматического программирования с применением искусственного интеллекта / С. А. Бевзенко. Текст : непосредственный // Молодой ученый. 2024. № 11 (510). С. 13-15. URL: https://moluch.ru/archive/510/112069/ (дата обращения: 08.02.2025).
2. Новицкая, И. В. Теория ошибки в свете различных подходов / И. В. Новицкая, А. Е. Вакалова. Текст: непосредственный // Молодой ученый. 2016. № 26 (130). С. 788-794. URL: https://moluch.ru/archive/130/36006/ (дата обращения: 08.02.2025).
3. Селиверстова А.В. Сравнительный анализ моделей и методов прогнозирования // Современные научные исследования и инновации. 2016. № 11 [Электронный ресурс]. URL: https://web.snauka.ru/issues/2016/11/74271 (дата обращения: 06.02.2025).
4. Часовских В.П., Аттокуров У.Т., Кох Е.В., Абдыракманова К.Т. Использование искусственного интеллекта для автоматизации тестирования // Управление образованием: теория и практика. 2024. № 6-1. С. 173-180.

References:

1. Bevzenko, S. A. Research of automatic programming methods using artificial intelligence / S. A. Bevzenko. Text: direct // Young scientist. 2024. No. 11 (510). P. 13-15. URL: https://moluch.ru/archive/510/112069/ (date of access: 02/08/2025).
2. Novitskaya, I. V. Error theory in light of various approaches / I. V. Novitskaya, A. E. Vakalova. Text: direct // Young scientist. 2016. No. 26 (130). P. 788-794. URL: https://moluch.ru/archive/130/36006/ (date of access: 02/08/2025).
3. Seliverstova A.V. Comparative analysis of forecasting models and methods // Modern scientific research and innovation. 2016. No. 11 [Electronic resource]. URL: https://web.snauka.ru/issues/2016/11/74271 (accessed: 06.02.2025).
4. Chasovskikh V.P., Attokurov U.T., Kokh E.V., Abdyrakmanova K.T. Using artificial intelligence to automate testing // Education management: theory and practice. 2024. No. 6-1. P. 173-180.