Автономная некоммерческая организация высшего образования
«Университет Иннополис»

АННОТАЦИЯ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
(БАКАЛАВРСКУЮ РАБОТУ)
ПО НАПРАВЛЕНИЮ ПОДГОТОВКИ
09.03.01 – «ИНФОРМАТИКА И ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА»

НАПРАВЛЕННОСТЬ (ПРОФИЛЬ) ОБРАЗОВАТЕЛЬНОЙ ПРОГРАММЫ
«ИНФОРМАТИКА И ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА»

Тема	**Анализ и визуализация кода для языка программирования Дарт**

Выполнил	**Нугаев Тимур Аладдинович**	подпись

Иннополис, Innopolis, 2023

Оглавление

Аннотация

Организация и структурирование кода крупного проекта является важной задачей, требующей постоянного анализа и рефакторинга кода. Существует множество методов для достижения этих целей, включая статический анализ кода, инструменты рефакторинга, визуализацию графов зависимостей и диаграмм UML. В данной работе предлагается новый подход к решению этой проблемы: создание веб-среды разработки (IDE), основанной на единичном выполнении статического анализа и других процедур. Разработанный в рамках исследования инструмент DartBoard представляет собой легкое, платформенно-независимое и компактное решение для анализа программного обеспечения.

Глава 1

Введение

1.1 Постановка проблемы

В настоящее время существует множество крупных масштабных проектов, таких как Windows или компилятор gcc. Все они требуют невероятных усилий для разработки и поддержания тысяч или миллионов строк кода. Это порождает ряд проблем, которые эта диссертация стремится решить. Три основные причины, по которым проекты становятся сложными:

- Масштаб. С увеличением размера программы управление ею становится все труднее. База кода может удвоиться за считанные недели. Это неизбежно. Поэтому нашей целью не является предотвращение этого, а скорее облегчение процесса понимания структуры кода в проекте.

- Командная работа. Множество программистов, работающих над одним и тем же проектом, может вносить дополнительную сложность в базу кода.

- Legacy. Система может стать настолько сложной, что со временем даже программист, который написал определенный фрагмент кода, может

не понимать, как он работает.

Поэтому программисты прибегли к визуализации своего кода.

1.2 Цель исследования

Целью этого исследования является создание полностью автономного приложения DartBoard, которое должно быть способно генерировать HTML-документ из проекта, написанного с использованием Dart/Flutter. Этот HTML-документ должен быть сгенерирован из существующего исходного кода и предоставлять необходимые для визуальной помощи инструменты.

1.3 Применяемость

Итоговое приложение будет полезно программистам для просмотра кода других программистов, предоставляя легкое и эффективное решение для анализа и визуализации кода. Предлагаемое решение требует меньше вычислений (следовательно, времени и стоимости) для генерации HTML, чем для запуска полноценной среды разработки.

1.3.1 Варианты использования

Ниже представлены некоторые потенциальные варианты использования приложения DartBoard:

- Ревью кода.
- Совместная работа над проектом.

- Отладка.
- Документация.

1.4 Контекст исследования

Следующие главы охватывают технику создания DartBoard, а также оценку его эффективности и удобства использования.

В этой вводной главе будет рассмотрена цель и значимость инструментов статического анализа в разработке программного обеспечения, особенно в свете языка программирования Dart. Мы также обсудим DartBoard, совершенно новый инструмент статического анализа, созданный специально для Dart, и как он отвечает на потребность в существующем инструментальном окружении. В главе "Обзор литературы"мы углубляемся в теоретические основы статического анализа, его различные подходы и их применение к языкам программирования. В главе "Требования"мы опишем функциональные и нефункциональные требования к DartBoard. В главе "Реализация"мы подробно обсудим архитектуру и дизайн DartBoard. В главе "Заключение"я подвожу итоги исследования, обобщая основные выводы и обсуждая последствия моей работы.

Глава 2

Обзор литературы

В данной главе рассматривается визуализация кода и процесс генерации кода, включая несколько инструментов и методов.

2.1 Визуализация

Инструменты визуализации кода, такие как "Отображение контура"и сворачивание блоков, помогают представить структуру исходного кода [1], [2].

2.2 Генерация

2.2.1 Генераторы документации

Генераторы документации автоматически создают документацию API из исходного кода приложения. Примерами служат Postman, Sphynx и Swagger.

2.2.2 Генераторы разметки

Языки разметки, например, Markdown или HTML, используются для форматирования текста и предоставления дополнительной информации. HTML был выбран в качестве основного языка разметки за его универсальность, гибкость, интерактивность, расширяемость и совместимость [3]—[7].

2.2.3 Генераторы кода

Генерация кода - распространенная практика в программной разработке, особенно в проектах на Flutter/Dart, которые используются в этом проекте. Генераторы кода автоматизируют создание шаблонного кода. Они обычно используют абстрактные синтаксические деревья (AST) или деревья элементов для анализа исходного кода.

2.2.4 Алгоритм Scanline

Алгоритм Scanline используется для упорядочивания генерации в процессе создания HTML-файла. Он позволяет эффективно обрабатывать блоки кода, учитывая их вложенность [8].

Глава 3

Системные требования

3.1 Функциональные требования

DartBoard предоставляет широкий набор функций для анализа и визуализации кода, включая:

3.1.1 Высокий приоритет

1. Отображение исходного кода на Dart.
2. Визуализация результатов анализа кода.
3. Подсветка синтаксиса, улучшающая читаемость кода.
4. Поиск и замена с использованием регулярных выражений.

3.1.2 Низкий приоритет

1. Поддержка внешних плагинов.
2. Возможность создавать собственные плагины.
3. Визуализация иерархии Flutter и дерева зависимостей.

4. Интеграция с CI/CD и автоматизация генерации HTML документации.

5. Переименование переменных/функций (рефакторинг).

6. Интеграция с pub.dev.

7. Подсветка неиспользуемых переменных и функций, а также недоступного кода.

3.2 Нефункциональные требования

DartBoard стремится удовлетворить следующие нефункциональные требования:

1. HTML, сгенерированный приложением, не должен иметь внешних зависимостей.

2. HTML должен работать на последних трех основных версиях Chrome, Edge, Firefox, Opera и Safari.

3. HTML должен работать на MacOS, Windows и Linux.

4. DartBoard должен обеспечивать всю необходимую функциональность современных редакторов кода, за исключением только чтения.

5. DartBoard должен иметь современный дизайн, напоминающий IDE.

6. DartBoard должен быть доступен в виде инструмента командной строки.

Глава 4

Реализация

Эта глава посвящена разработке и реализации приложения DartBoard. Структура следующая: Раздел 4.1 обсуждает технологический стек и причины выбора Dart и Flutter. Раздел 4.2 описывает компромисы (development trade-offs) в процессе разработки и решения, которые я принял для их разрешения. Раздел 4.3 описывает детали реализации, включая ключевой функционал приложения DartBoard.

4.1 Технологический стек

DartBoard создан на языке программирования Dart и фреймворке Flutter. Dart - современный объектно-ориентированный язык программирования, предназначенный для веб- и клиентской разработки. Flutter - это SDK для мобильных приложений, построенный на Dart и предлагающий богатый набор предварительно созданных виджетов для создания мобильных приложений.

Dart и Flutter были выбраны по нескольким причинам. Во-первых, Dart является основным языком для разработки приложений на Flutter,

что обеспечивает хорошую интеграцию этих технологий. Это позволяет приложению использовать все возможности обоих языков. DartBoard также использует внутренние свойства языка программирования Dart для анализа и представления программного проекта в целом, что позволяет эффективно анализировать и визуализировать код без необходимости во внешних инструментах или плагинах.

4.2 Компромиссы

В этом разделе обсуждаются различные компромиссы, с которыми пришлось столкнуться при разработке проекта.

4.2.1 Шаблонизация

Я сознательно решил не использовать библиотеку шаблонизации для генерации HTML и JS кода из Dart. Вместо этого я выбрал более прямой подход, работая напрямую с HTML, JS и CSS. Такое решение отражает стремление к большему контролю, более быстрой разработке и уменьшению зависимости от внешних инструментов.

4.2.2 JavaScript / Фреймворки

В рамках данного проекта было принято решение не использовать JavaScript-фреймворк для повышения уровня контроля над проектом, ускорения разработки без затрат времени на компиляцию и уменьшения зависимости от внешних библиотек.

4.2.3 Без сервера

Приложение не разворачивает сервер для сохранения портативности, быстродействия (в работе с файлами на диске), надежности и независимости от зависимостей и оверхедов, которые приходят с серверным подходом.

4.2.4 HTML

Было решено выбрать HTML как язык разметки, используемый для генерации выходного результата.

HTML обеспечивает прочную основу для создания веб-приложений, доступных, визуально привлекательных, интерактивных и совместимых на разных устройствах и платформах. Сами HTML-файлы разделены на несколько разных файлов и работают в совокупности, чтобы способствовать организованному рабочему процессу разработки.

4.3 Детали реализации

DartBoard предлагает множество функций, таких как дерево проекта, подсветка синтаксиса и переход к объявлениям переменных и функций.

4.3.1 Области видимости

В этом разделе основное внимание уделяется использованию Абстрактного Синтаксического Дерева (AST) и алгоритма scanline для эффективного анализа кода, рассказывая о том, как реализована область видимости в DartBoard. Было рассмотрено внедрение блочной и модульной области видимости.

DartBoard использует двухступенчатый процесс для достижения эффективной области видимости:

1. Просмотр AST: DartBoard перемещается по AST, останавливаясь на каждом узле, чтобы собрать релевантные данные о областях видимости, включая объявления переменных, параметры функций и конструкции, связанные с модулями. Он отслеживает текущую область видимости и связывает использования переменных с соответствующими объявлениями с помощью пользовательского визитора AST.

2. Организация данных: после обхода AST, DartBoard группирует данные в карты (map) и списки (list), для быстрого извлечения и обработки.

DartBoard использует метод scanline – линейный метод обхода, для быстрой обработки блочной области видимости. Его основная функция - маркировать и обозначать отдельные участки кода, определяемыми "скоупами"в коде.

4.3.2 Генерация HTML

Процесс генерации HTML состоит из нескольких компонентов, включая анализ кода Dart, обработку кода через конвейер (pipeline) генерации HTML и создание окончательного HTML-вывода.

Конвейер генерации HTML - одна из самых важных и критических частей процесса, ответственная за преобразование кода Dart для включения функций, функционала. Одной из его ключевых особенностей является наложение слоев функционала умным и эффективным образом. Каждый шаг в конвейере фокусируется на конкретной фиче (функционал/feature), и порядок применения этих фичей в пайплайне в общем не имеет значения.

Этот модульный подход позволяет легко вносить корректировки и фичи, не затрагивая общий конвейер и/или конечный результат, обеспечивая хорошую поддерживаемость (maintainability), гибкость и расширяемость системы.

"Умное слоение"(layering) функций в конвейере позволяет применять каждую фичу функционала независимо, не влияя на другие. Этот подход к слоению гарантирует, что фичи применяются последовательно.

Этот конвейер выполняется для каждого файла в каждой вложенной папке входного проекта. На данный момент он состоит из таких шагов (и может дополняться без изменений кода по принципу "Open-closed"из SOLID): добавление блоков для сворачивания кода, добавление привязок объявлений для использования переменных, добавление всплывающих подсказок для комментариев и добавление подсветки синтаксиса. Каждый шаг меняет HTML после прохода по коду, поэтому я и использую "умное слоение"с использованием scanline, чтобы не было столкновений и HTML-документ генерировался корректно и эффективно в один проход один раз.

4.3.3 Номера строк

Эта функция добавляет номера строк в интерфейс DartBoard, улучшая навигацию, отладку и читаемость кода. Реализация включает добавление разметки для номеров строк в шаблоне HTML и динамическое заполнение этой разметки с помощью JavaScript при загрузке страницы.

4.3.4 Документация классов

Этот функционал включает анализ AST для поиска классов, извлечение метаданных о классе и их хранение в объекте ClassInfo, а также внедрение информации из узлов класса в теги, которые отображаются при наведении мыши в HTML-выводе. Информация используется для формирования HTML-тегов.

Глава 5

Заключение

5.1 Ценность и цель

Ценность данного проекта в его потенциале упрощать понимание сложных кодовых баз для разработчиков с помощью техник визуализации кода. Особенно это актуально для новых участников команды или непрограммистов, стремящихся понять структуру программного обеспечения.

5.2 Метрики

Успех проекта оценивается по удобству его использования и влиянию на эффективность процесса разработки.

5.2.1 Производительность

Система продемонстрировала высокую производительность в процессе визуализации кода для проектов малого и среднего размеров, в среднем достигая скорости генерации в 5-6 секунд, что является приемлемым в рам-

ках нефункциональных требований, объявленных ранее в соответствующей главе про системные требования.

Выходные .zip архивы для тестовых проектов Pets и Calculator занимают по 50 КБ каждый, что считается приемлемым для хранения на жестком диске пользователя.

5.3 Следующие шаги

В этом подразделе рассмотрены возможности дальнейшей разработки проекта. Следующим шагом будет улучшение системы тестирования для более полного покрытия возможных сценариев использования DartBoard.

5.3.1 Анализ кода

Дополнительные возможности анализа кода, такие как распознавание шаблонов проектирования или анализ потоков данных, могут быть полезными дополнениями к инструменту как средство визуализации.

Список использованной литературы

[1] T. A. Ball и S. G. Eick, *Software Visualization in the Large*, http://www.cs.kent.edu, Accessed: 13-Nov-2022. Available: http://www.cs.kentedu/Hmaletic/softvis/papers/BallEick1996.pdf, 1996.

[2] W. D. Pauw, D. Lorenz, J. Vlissides и M. Wegman, «Execution Patterns in Object-Oriented Visualization,» в *Conference on Object-Oriented Technologies and Systems (COOTS)*, 1998, с. 219—234.

[3] T. Berners-Lee, *HTML: A Representation of Textual Information and MetaInformation for Retrieval and Interchange*, World Wide Web Consortium (W3C), 1995.

[4] D. Raggett, A. L. Hors и I. Jacobs, *HTML 4.01 Specification*, World Wide Web Consortium (W3C), 1999.

[5] D. Flanagan, *JavaScript: The Definitive Guide*. O'Reilly Media, 2020.

[6] D. Crockford, *HTML5: Up and Running*. O'Reilly Media, 2010.

[7] I. Hickson, *HTML Living Standard*, Web Hypertext Application Technology Working Group (WHATWG), 2021.

[8] J. D. Foley, A. Van Dam, S. K. Feiner и J. F. Hughes, *Computer Graphics: Principles and Practice*, 2-е изд. Addison-Wesley, 1996, ISBN: 0-201-84840-6.