УДК 004.9

РАЗРАБОТКА ИНСТРУМЕНТА СТАТИЧЕСКОГО АНАЛИЗА КРУПНЫХ ПРОГРАММНЫХ ПРОЕКТОВ ДЛЯ ПОСТРОЕНИЯ ГРАФОВ ВЫЗОВОВ И ПОТОКОВ ДАННЫХ

Соболев К.Г
Научный руководитель – преподаватель, Петухов П.Н.
Университет ИТМО
colorblind.work@yandex.ru

Введение

В большинстве крупных компаний с течением времени кодовая база сильно разрастается, что усложняет процесс отслеживания корректности перемещения данных между различными частями системы и понимание всех взаимодействий между программными модулями. Для визуализации графов вызовов и потоков данных на языке Golang существует встроенный пакет go/ast [1], предоставляющий доступ к типам и функциям для работы с абстрактным синтаксическим деревом (AST [2]) Go-программ. Наряду с встроенными пакетами есть и сторонние решения, такие как go-callvis, но они имеют ограниченный функционал и сложны в освоении. Целью работы является провести сравнительный анализ средств для визуализации графов вызовов и потоков данных и разработать собственное решение на основании пакета go/ast.

Основная часть

В ходе выполнения работы был проведён обзор существующих аналогов и подходов к построению графов вызовов, который показал, что имеющиеся решения покрывают не все выдвигаемые требования и не позволяют в полной мере достигнуть поставленной цели. В рамках практической реализации был разработан алгоритм построения графа вызовов программы на основе анализа данных из абстрактного синтаксического дерева (AST) исходного кода, сформированного средствами пакета go/ast. Граф вызовов представляет собой ориентированный граф, в котором узлы соответствуют функциям, а рёбра отражают их вызовы. Такой граф позволяет наглядно представить структуру взаимодействий внутри проекта, а также ускорить анализ зависимостей при сопровождении и рефакторинге. Разработанная консольная утилита принимает путь до проекта для анализа, а также наборы включаемых и исключаемых пакетов, что позволяет ограничивать область рассмотрения и снижать объём результирующего графа. На этапе загрузки исходного кода применяется пакет go/packages, далее для файлов выбранных пакетов строится AST с использованием go/ast. После этого выполняется обход дерева: выделяются объявления функций, определяются контексты их принадлежности и извлекаются выражения вызовов. На основе найденных данных формируется набор вершин и рёбер графа. Рёбра классифицируются по типу вызова в зависимости от конструкции языка: прямой вызов функции, отложенный вызов defer, запуск в отдельной горутине, а также вызовы через замыкания, что повышает информативность графа. Для итогового представления результатов граф поддерживает экспорт в формат .dot, что делает возможной

последующую визуализацию с использованием утилиты Graphviz [3], а также применение стандартных инструментов для анализа и обработки графов.

Выводы

Проведён анализ инструментов для построения графов вызовов и потоков данных на языке Golang, рассмотрены их преимущества и недостатки. Разработана консольная утилита, позволяющая строить графы вызовов в .dot формате, что позволяет визуализировать взаимодействие между частями системы и упрощает её анализ. Дальнейшее развитие проекта предполагает интеграцию с LLM для поиска циклических вызовов, неоптимальных зависимостей и прочих архитектурных проблем.

Литература

1. Документация пакета go/ast [Электронный ресурс]. – Режим доступа: https://pkg.go.dev/go/ast (Дата обращения 02.02.2026).
2. Reilles A. Canonical Abstract Syntax Trees // Proc. 6th International Workshop on Rewriting Logic and its Applications. Vienna, Austria, 2006. P. 165 – 179.
3. Документация Graphviz [Электронный ресурс]. Режим доступа: https://graphviz.org/ (Дата обращения 02.02.2026)