***Шаблий А.Д.***

**Формализация процесса разрешения зависимостей файлов исходного кода**

**Введение**

Важным этапом при анализе структуры программного обеспечения (ПО) является разрешение фукциональных зависимостей между файлами исходного кода. Результаты такого анализа востребованы, например, при построении математических моделей и проектировании программных систем. Так, современное ПО допускает наличие нескольких точек входа (функций main), задействованная точка входа зависит от конфигурации сборки ПО. При этом в зависимости от точки входа задействуются разные объемы кодовой базы.

Текущие методы построения математических моделей предполагают использование готовых результатов анализа. Например, их можно получить при помощи инструментов статического анализа, таких как SonarQube или Clang Static Analyzer. При этом исходными данными становятся не файлы исходного кода и данные о их функциональных зависимостях, а результаты работы стороннего программного средства, что может приводить к ошибкам и трудным для отладки неточностям.

Целью настоящей работы является сформированный механизм разрешения функциональных зависимостей файлов исходного кода, что позволит автоматизировать процесс формирования математиеских моделей и невилировать обозначенные недостатки.

**Теоретическая основа и математическая модель**

Для формирования такого механизма необходимо, чтобы все файлы исходного кода были проиндексированиы, а так же ввести следующие обозначения:

1. - количество файлов исходного кода;
2. - вектор бинарных отношений, определяющий файл с задействованной точкой входа;
3. - матрица бинарных отношений, описывающая наличие функциональные зависимости между файлами исходного кода.

Элемент вектора принимает значение если в -ом файле описана задействованная точка входа и если не описана. Элемент матрицы принимает значение если ‑й файл зависит от -го и если не зависит.

Разрешение зависимостей файла исходного кода с задействованной точкой входа:

Разрешение их зависимостей:

Тогда очевидно, что:

Заметим, что глубина разрешения зависимостей не может превышать . Тогда файлы исходного кода, необходимые для разрешения зависимостей исходного обозначим за , значения элементов которого вычисляются как:

**Пример**

Рассмотрим пример, описывающий 3 файла исходного кода. Точка входа описана в файле № 1. Файл № 1 имеет зависимость на файл № 2. Файл № 2 имеет зависимость на файл № 3. Файл № 3 не имеет зависимостей. Тогда математически условие можно описать следующим образом:

Разрешение зависимостей :

Разрешение зависимостей :

Разрешение зависимостей :

Состав файлов с учетом разрешения зависимостей :

Вывод: для разрешения зависимостей необходимы все файлы исходного кода.

**Заключение**

В настоящей работе описан механизм разрешения функциональных зависимостей между файлами исходного кода и приведено описание результата в виде математического объекта, что позволяет его использовать при математическом моделировании проектируемой программной системы.

В дальнейших исследованиях предполагатся описание механизма формирования исходного перечня файлов исходя не из логики, что какой-то из них содержит главную функцию программы, а исходя из реализуемых этими файлами заданного объема функциональных требований. Это позволит нивелировать ограничение, что только один элемент вектора может принимать значение 1 и расширит спроектированный механизм задействованием таких понятий как функциональные требования и трассировка требований на файлы исходного кода.

**Список литературы**

1. Минаева Н.В., Беспалов М.Е. Средства автоматизированного проектирования информационных систем // Сборник научных трудов Международного научно-технического симпозиума и III Международного Косыгинского Форума. Москва, 2021. С. 75-77.
2. Мустаев Р.Р. Автоматизация процессов сборки и развертывания программного продукта // Студенческий. 2021. № 7-1 (135). С. 10-14.
3. Сорин Н.В., Котлярова Л.П., Ковалев А.Д. Визуализация графа этапов сборки программных продуктов с использованием библиотеки D3.js // Современные технологии в теории и практике программирования. Сборник материалов научно-практической конференции студентов, аспирантов и молодых ученых. Санкт-Петербург, 2023. С. 251-253.
4. Ефимов А.А. Непрерывный контроль качества кода с помощью SonarQube // Инновации. Наука. Образование. 2020. № 11. С. 286-297.
5. Бучатский Р.А., Чуркин Я.А., Чибисорв К.А., Пантилимонов М.В., Долгодворов Е.В., Вязовцев А.В., Волохов А.Г., Трунов В.В., Миракян Г.О., Китаев К.Н., Белеванцев А.А. Проверка программ на соответствие стандарту MISRA C с использованием инфраструктуры Clang // Труды Института системного программирования РАН. 2023. Т. 35. № 5. С. 169-192.