УДК 004.054

АНАЛИЗ ПРОГРАММНЫХ СИСТЕМ С ПОМОЩЬЮ СТРУКТУРНОЙ МАТРИЦЫ ЗАВИСИМОСТЕЙ

Антоненко Д.И., Балахонов Ю.В.
Севастопольский национальный технический университет
кафедра кибернетики и вычислительной техники
E-mail: AntonenkoDI@gmail.com

Аннотация

Антоненко Д.И., Балахонов Ю.В. Анализ программных систем с помощью структурной матрицы зависимостей(Dependency structure matrix). *Рассмотрена проблема выявления и анализа зависимостей между компонентами сложных программных проектов. Определены классы зависимостей и способы их идентификации. Разработаны средства построения и отображения DSM для двух вариантов представления: автономного приложения и встраиваемого приложения для Apache Maven.*

Общая постановка проблемы. Программное обеспечение, по своей природе, подвержено изменениям на протяжении своего жизненного цикла. В дополнение к постепенному росту сложности и увеличению времени локализации ошибок в тексте программ, есть риск того, что при реализации новых возможностей или изменении уже существующих функций, в систему будут введены дефекты. Зависимости между программными модулями играют важную роль в ремонтопригодности программного обеспечения. При изменении одного программного модуля, вполне возможно, что другие модули, зависящие от него, также должны быть изменены или протестированы.

Возможно два подхода для снижения негативного воздействия изменений в ПО. Первый характеризуется прогнозированием устойчивости каждого модуля на основе уже имеющих данных (изменения в различных версиях, история). Таким образом, разработчики могут сосредоточить свои усилия на выявленных проблемных областях.

Анализ воздействия является другим подходом: эффект от внедрения изменений оценивается до фактического осуществления чего-либо. Это дает возможность взвесить положительные последствия предлагаемых изменений в отношении каких-либо отрицательных побочных эффектов, и принять решение о наилучшем варианте действий в сложившейся ситуации. Анализ воздействия также может быть использован, чтобы выбрать наиболее выгодное решение из группы вариантов решений.

Одним из факторов, который влияет на изменяемость элемента проекта и который легко измерить – это стабильность. Одним из способов измерения стабильности является подсчет числа зависимостей, которые «входят» и «выходят» из этого пакета. Эти расчеты позволяют нам вычислить позиционную стабильность пакета.

Задачей исследования является разработка методики, на основании которой осуществляется проектирование приложений для анализа сложных программных систем.

Введем следующие обозначения. Пусть Ca – число афферентных соединений, то есть число классов вне этого пакета, которые зависят на классы в рамках этого пакета. Пусть CE - эфферентная соединений - число классов в этом пакете, которые зависят от классов вне этого пакета. Тогда под нестабильностью будем понимать отношение

$$I = \frac{Ce}{Ca + Ce}; \qquad (1)$$

Этот показатель лежит в диапазоне [0,1].

I = 0 указывает на максимально стабильный пакет. I = 1 указывает на максимально нестабильный пакет. Для расчета значений Ca и Ce может использоваться структурная матрица зависимостей.

Структурная матрица зависимостей (Dependency Structure Matrix или DSM) — это компактное матричное представление системы или проекта. DSM отображает зависимости в структурной матрице или конструкцию структурной матрицы. Это один из способов визуального представления системы.

На сегодняшний день DSM можно использовать для представления архитектуры программного обеспечения. Представление архитектуры программного обеспечения обычно состоит из простых матриц, которые отображают зависимости между подсистемами. Такой подход имеет 2 важных преимущества. Во-первых, так может быть представлен большой объем системных элементов и их связей компактным способом с выделением важных моделей в данных(таких как циклы и модули). Во-вторых, применимы техники анализа матриц, которые могут быть использованы для улучшения структуры системы.

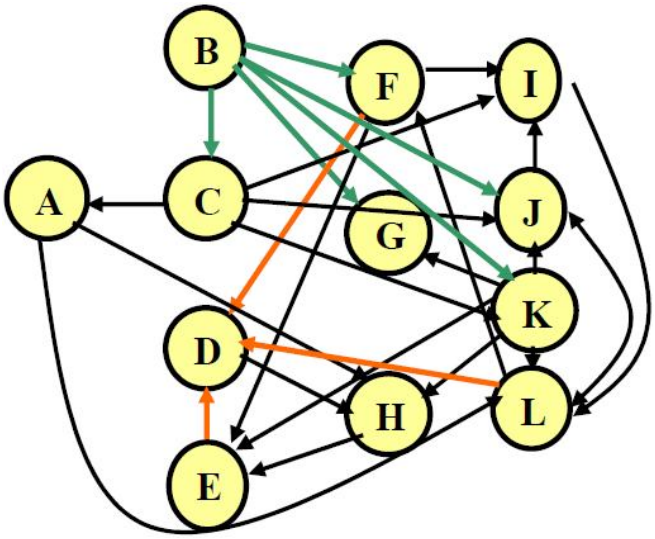

Рис. 1. Пример диаграммы зависимостей классов проекта.

Как видно из рис.1 представление межмодульных зависимостей в проектах с большим числом связей при помощи графической диаграммой не является интуитивно понятным. Более наглядное представление в виде DSM представлено на рисунке 2.

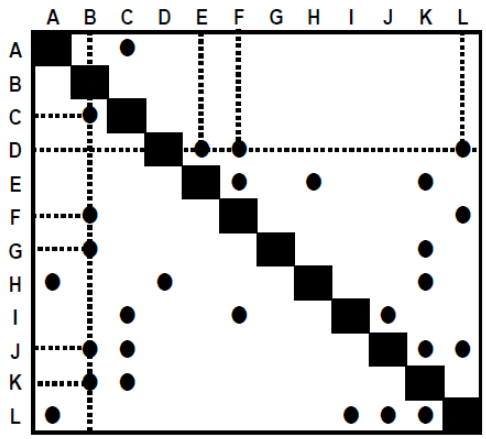

Рис.2. Структурная матрица зависимостей для графа представленного на рис.1.

Анализ DSM дает представление о том, как управлять сложными системами или проектами, выделяя степени зависимостей модулей Процесс анализа программной системы состоит в выполнении иерархической декомпозиции системы и получение информации о взаимодействии и связях между подсистемами. Например, если спецификация для компонента будет изменена, можно быстро идентифицировать все модули, которые зависимы от этой спецификации, уменьшая риск что работа продолжится на основе не верных данных.

Зависимости могут быть прямыми(direct), переходными(transitive), или циклическими(cyclic). Прямая зависимость существует между двумя элементами: зависимым, и того от которого зависит. Транзитивная зависимость возникает, когда два элемента связаны с помощью одного или нескольких узлов посредников. Транзитивные зависимости не столь существенны, как прямые. Если программная система определена правильно, с соблюдением основных принципов объектно-ориентированного программирования, таких как инкапсуляции и модульность, вполне вероятно, что большинство изменений не будет распространяться через транзитивные зависимости(. Джексон, 2004, стр. 199)

Циклические зависимости считаются наиболее вредным типом зависимостей. Их использование не рекомендуется в большинстве случаев, особенно на более высоких уровнях абстракции, например, на уровне пакета Java. Циклическая зависимость возникает, когда зависимые пункты создают ориентированный граф, который начинается и заканчивается в одном узле. Простейшая циклическая зависимость состоит из двух пунктов, которые зависят друг от друга.

Меньшее число зависимости означает лучшую возможность многократного использования, лучшие условия для тестирования и внесения изменений. Элемент должен зависеть только от элементов, которые являются более стабильными, чем он сам. Циклические зависимости являются основанием для большинства проблем. Необходимо минимизировать их число. В определенных сферах обычно также добавляют критерии определения хороших/плохих зависимостей. Например, в разработке программного обеспечения, серверный код не должен зависеть от кода клиента.

Целью данной работы является разработка специального вспомогательного программного обеспечения для анализа программных систем, которое позволяет проанализировать различные виды зависимостей внутри Java-проектов на основе построенной структурной матрицы зависимостей.

Для решения поставленной задачи был разработан обобщенный алгоритм, который представлен на рис.3.

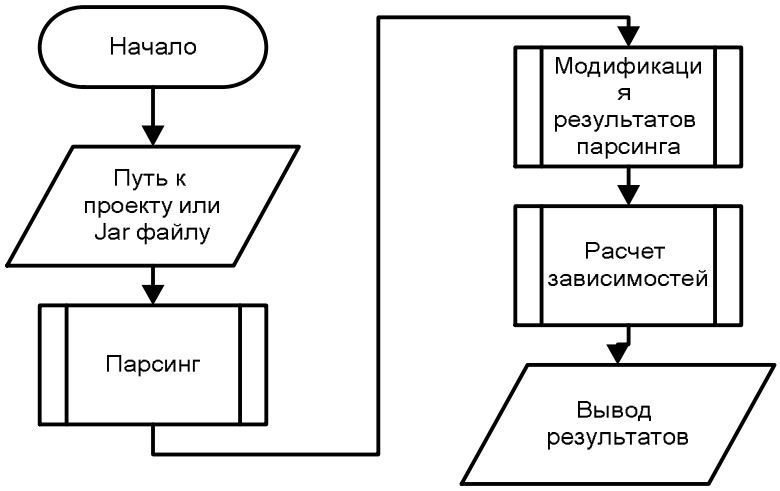

Рис.3.Обобщенный алгоритм выполнения программы

Для реализации этапа парсинга был использован пакет Dtangler, позволяющий на выходе получить всю необходимую информацию о проекте и специализированные структуры данных, информация в которых является основанием для расчета и визуализации значений Ca и Ce. В результате анализа полученной информации могут быть рассчитаны значения Ca и Ce и степень нестабильности I. Однако, разработчику программного обеспечения важно знать не само значение степени нестабильности, а минимизировать его с помощью операции рефакторинга программной системы. Процедура рефакторинга до сих пор слабо формализирована. Одним из подходов является визуальный анализ матрицы зависимостей программной системы с целью выделения наиболее сильно связанных участков графа, проблемных областей, которые будет целесообразно выделить в отдельный пакет. Для этого были разработаны средства построения и отображения DSM в двух направлениях: автономное приложение, а также плагин к фреймворку Apache Maven.

Apache Maven — фреймворк для автоматизации сборки проектов, специфицированных на XML-языке POM. Maven также собирает статистическую информацию по проекту для генерации отчетов, которые обеспечивают поддержку разработки проекта. Разработанный плагин позволяет просматривать зависимости между пакетами сложной программной системы, а также между классами в каждом пакете. Загрузка встроенного программного модуля происходит при генерации сайта для Maven проекта.

Автономное приложение обладает графическим интерфейсом пользователя, а также расширенными функциями навигации по DSM, такими как раскрытие или группировка элементов (классов и пакетов) и представление структурной матрицы зависимостей между различными уровнями проекта. Таким образом, пользователь может выделить наиболее зависимые пакеты, а затем и наиболее зависимые классы от другого пакета, не переходя на другой уровень представления. Графический интерфейс так же выделяет на структурной матрице зависимостей наиболее негативные, циклические зависимости.

Так же реализованные программные средства позволяют определить степень стабильности для каждого пакета в соответствии с формулой (1).

Язык разработки приложения – Java, среда разработки Eclipse IDE 2.3.0 Helios SR1 /NetBeanse IDE 6.7.1, платформа Windows/Linux.

Выводы

Разработанное программное обеспечение позволяет визуализировать DSM, представить структурную матрицу зависимостей в виде интуитивно понятной системному аналитику структуры связей. Это позволит сократить время рефакторинга системы с целью оптимизации нестабильности. С другой стороны, при внесении изменений в проект, системный аналитик может выделить наиболее зависимые элементы системы и произвести их тестирование.

Литература

1. Kaisa Kittila, Analysing and Managing Software Dependencies with a Dependency Structure Matrix Tool[Текст] – University of Oulu Department of Information Processing Science – 2008 – C.12.

2. Object mentor [Electronic resourse] - http://www.objectmentor.com/resources/articles/stability.pdf. - Stability.

3. Dependency Structure Matrix homepage [Electronic resourse] - http://www.dsmweb.org/

4. Apach Maven development center [Electronic resourse] - http://maven.apache.org/plugin-developers/index.html

5. Ali A. Yassine, An Introduction to Modeling and Analyzing Complex Product Development Processes Using the Design Structure Matrix (DSM) Method[Текст] – University of Illinois at Urbana-Champaign – 2004 – C.4.