**Разрешение циклических зависимостей графовой модели взаимосвязи требований к программному обеспечению**

**Аннотация**

Получение сведений о трассируемости требований к программному обеспечению (ПО) на файлы исходного кода приложения является сложной прикладной проблемой. Особенно при установлении необходимого объема верификационных процедур, выполнение которых необходимо для подтверждения выполнимости требований к ПО при изменении одного или нескольких файлов исходного кода. Цель работы – это разработка модели, которая бы обеспечивала пользователя информацией о связности файлов исходного кода между собой и требованиями к ПО. Цель достигнута за счет разрешения циклических зависимостей и формирования графа трассируемости требований к ПО на файлы исходного кода, неразрывно связанных друг с другом. Предложена программная реализация модели, в состав которой включены опциональные модули для исследования ее оптимальной конфигурации. Исследована зависимость времени работы от реализации способа хранения данных в разработанном программном решении. Практическая значимость заключается в том, что разработанная модель позволяет получать полный перечень требований к ПО, корректность выполнения которых необходимо проверить после внесения изменений в файлы исходного кода разрабатываемого приложения. В дальнейшем планируется использовать разработанную модель и ее программную реализацию для решения задачи оптимальной декомпозиции файлов исходного кода по плагинам инструментального средства конфигурирования, выполненного как часть плагинной системы с целью исследования и разработки принципов построения таких программных решений.

Ключевые слова - требования, трассируемость, исходный код, графовые математические модели, ориентированный граф, циклы.

**Введение**

Трассируемость требований на файлы исходного кода ПО - это взаимосвязь между требованиями к ПО и файлами исходного кода, которая определяет происхождения, порождения или зависимости между ними. Получение сведений о такой трассируемости является сложной прикладной проблемой. Зачастую такие сведения необходимы для анализа эффективности проведенных работ по проектированию приложения, оценке стоимости его тестирования [1], а также для управления его конфигурацией. В данной работе предложен способ построения схемы, описывающей в каких файлах исходного кода реализуется заданный объем требований к ПО и адаптированной для получения сведений о зависимостях файлов исходного кода между собой.

Для решения задач, связанных с оптимизацией ПО [17], получением данных о загруженности ресурсов вычислительной системы [7], иерархическим описанием зависимостей компонентов [10] широкое применение нашли графовые математические модели. Это обусловлено простотой описания актуальных для программирования проблем с использованием графовых моделей [12].

При описании работы потоков управления, потоков данных, связи составных частей распределенной системы очень важно указывать направление потока или связи [5], [6]. При организации связей всегда есть источник и есть потребитель. Иногда один компонент системы, обозначаемый на графе вершиной, является одновременно источником и потребителем. Кроме того, он может быть источником для нескольких потребителей и потребителем от нескольких источников. В ряде задач одна вершина может быть источником и потребителем для самого себя, например в случае наличия обратных связей в описываемой модели. Для описания таких связей используются ориентированные графы, в которых по направлению дуг можно судить о принадлежности вершины к числу источников или приемников.

Для решения задач, связанных с оптимизацией многопоточного ПО [3], необходим механизм преобразования графа из исходного вида к целевому. Преобразование происходит по заранее сформулированным правилам и может осуществляться за несколько итераций. Условие окончания проведения итераций преобразования так же определено заранее. Для решения задачи построения графа трассируемости требований к ПО на файлы исходного необходимо, чтобы вне зависимости от очередности вершин графа, к которым применяются действия по преобразованию, результирующий граф всегда формировался бы одинаково. Описанный в статье [3] способ не гарантирует этого.

Изложенный в работе [19] подход к формированию графа с применением алгоритмов нейронных сетей [11] не учитывает ограничение, что вершины результирующего графа должны быть двух категорий. В то же время приведенный способ формирования результирующего графа может быть доработан для решения задачи построения оптимальной декомпозиции компонентов приложения с целью максимизации вариантов комплектаций его поставки при заданном трассировании требований к ПО на файлы исходного кода с учетом разрешенных циклических зависимостей и выполняя свою работу на уже предварительно преобразованном графе.

Описанные в [3] и [15] приемы преобразования ориентированного графа и видоизменения его в результате итерационно выполняемых действий нацелены на построение такого графа, который бы упрощал поиск цепочки задействованных в одном сценарии работы ПО вершин графа. Такие приемы требуют доработки и адаптации для решения задачи построения графа трассируемости требований к ПО на файлы исходного кода с учетом возможности наличия циклических зависимостей у файлов исходного кода между собой.

Кроме того, при описании способов преобразования исходного графа необходимо учитывать временные издержки, которые появляются в ходе выполнения операций над исходным графом. Объем издержек возрастает, если в исходном графе присутствует значительное число требований к ПО и файлов исходного кода.

**Описание модели**

Исходными данными для модели являются: информация о составе файлов исходного кода, сведения об их зависимостях между друг другом, а так же трассируемость требований на них.

Перед началом работы модели необходимо определить:

1. множество файлов исходного кода ;
2. множество требований к ПО ;
3. матрицу бинарных отношений трассируемости требований к ПО на файлы исходного кода размерности ;
4. матрицу бинарных отношений зависимостей файлов исходного кода размерности .

Значение элемента матрицы определяется следующим образом:

Значение элемента матрицы определяется следующим образом:

Примеры исходных матриц и приведены в таблицах и 1 и 2 соответственно.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  | … |  |
|  | 1 | 0 | … | 0 |
|  | 0 | 1 | … | 0 |
|  | 0 | 1 | … | 0 |
| … | … | … | … | … |
|  | 0 | 0 | … | 1 |