Разрешение циклических зависимостей графовой математической модели триссируемости требований к ПО на файлы исходного кода

Содержание

1 Аннотация

Некоторый текст анотации

2 Введение

Получение сведений о трассируемости требований к программному обеспечению (ПО) на файлы исходного кода приложения является сложной прикладной проблемой. Зачастую такие сведения необходимы для анализа эффективности проведенных работ по проектированию приложения, оценке стоимости его тестирования, а так же управления его конфигурацией. В данной работе описывается 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, получением данных о загруженности ресурсов вычислительной системы, иерархическим описанием зависимостей компонентов, получением данных о вызове процедур широкое применение нашли графовые математические модели. Это обусловлено простотой описания актуальных для программирования проблем в терминах графов, а значит они могут быть решены при помощи идентичных хорошо изученных и математически обоснованных механизмов из теории графов.

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 узлом, является одновременно источником и потребителем. Кроме того, он может быть источником для нескольких потребителей и потребителем от нескольких источников. В ряде задач один узел 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 узла к числу источников или приемников.

Для решения задач связанных с оптимизацией многопотоного 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 узлов графа, к которым применяются действия по преобразованию, результрующий граф всегда формировался бы одинаково. Описанный в статье [3] способ не гарантирует этого.

Изложенный в работе [20] подход к формированию графа с применением алгоритмов нейронных сетей не учитывает ограничение, что узлы результирующего графа должны быть 2х 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 [16] 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 узлов графа. Такие приемы требуют доработки и адаптации для решения задачи построения графа трассируемости треований 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 при выполнении операций над исходным графом. Объем издержек возрастает при наличии в исходном графе значительного числа требований к ПО и файлов исходного кода.

3 Описание модели

Это достигается благодаря графу трассируемости требований на исходный код с количественной оценкой циклически зависимых друг на друга файлов исходного кода.

В случае описания зависимостей по направлению звеньев из узла и в узел можно судить о том, на какие какие файлы исхожного кода существует зависимость у текущего, а так же для каких файлов он является зависимостью.

В случае описания трассируемости начальными вершинами будут являться файлы исходного кода, а конечными вершинами - требования, реализация которых описана в соответствующих файлах исходного кода.

(рисунок)

Построенный граф отображает, как реализуются требования в файлах исходного кода программы. Заметим, что в звенья данного графа удобно разделить на две группы: группа требований и группа файлов исходного кода. Звенья графа из группы требований не имеют звеньев между друг другом. Кроме того, одно требование может быть реализовано в нескольких файлах исходного кода. Звенья графа из группы файлов исходного кода имеют звенья между друг другом, так же связи могут образовывать цикл. Однако нет необходимости указывать связь файла на самого себя, так как для цели отображения файлов исходного кода, которые должны войти в сборку такая связь не имеет смысл.

Объединение звеньев, имеющих друг на друга циклические зависимости, следует производить итерационно до тех пор пока в результирующем графе не будут отсутствовать циклы. После объединения узлов в одну группу граф видоизменяется. При этом для звеньев, которые объединяются выполняется:

- звенья между друг другом удаляются
- узлы объединяются в группу
- образованная группа является новым звеном графа, при этом:

– образованная группа является начальным узлом для звеньев между конечными узлами для звеньев, связывающих объединенные узлы
– образованная группа является конечным узлом для звеньев между начальными узлами для звеньев, связывающих объединенные узлы

При итерационном поиске циклов в графе вышеобозначенные группы учитываются как обыычные узлы и таким образом могут учавствовать в образовании новых групп по вышеуказанным правилам.

4 Программная реализация

При достаточно большом количестве элементов изначального графа трассируемости требований на файлы исходного кода не представляется возможным его преобразование и последующая оценка эффективности без применения инструментов предоставляемых вычислительной техникой и возможностей языков программирования. Не существует универсального способа описания графа на языке программирования с целью применения полученной реализации для решения произвольной задачи. В каждом отдельном случае, для каждой отдельно взятой задачи существует необходимость описания графовой модели и алгоритмов действий над ней. Однако существуют распространенные практики, позволяющие эффективно описывать графовые математические модели на тех или иных языках программирования.

Так, в работах ... для описания состава узлов и связей между ними применяются массивы целочисленных данных. В работах ... продемонстрировано использование динамических списков. Широкое применение для решения такого класса задач нашли структуры, их возможности описаны в ... .

5 Эксперименты и обсуждение

6 Заключение