Алгоритмы автоматизированного выявления связей между элементами проекта разработки программного обеспечения

**Человек:** Предметом исследования является определение связей между элементами проекта разработки программного обеспечения, реализуемое с помощью трассировки требований на основе данных из систем контроля версий исходного программного кода. Многие известные методы трассировки являются зависимыми от языка программирования, что ограничивает их использование в проектах, разрабатываемых с использованием нескольких языков программирования. Поэтому цель исследования заключалась в формировании набора алгоритмов, позволяющих выстраивать связи между сущностями процесса разработки программного обеспечения (артефактами) на основе исходного кода, и анализировать эти связи (при этом код должен быть независимым от языка программирования и простым в реализации). Методология исследования объединяет методы системного анализа, программной инженерии, разработки программного обеспечения, теории надежности, информатики и математической квалиметрии. Основными выводами проведенного исследования являются алгоритмы автоматизированного определения связей между элементами проекта разработки программного обеспечения, позволяющие решать поставленные задачи выполнения импакт-анализа. Высокую вычислительную сложность разработанных алгоритмов можно снизить путем постепенного формирования глобальной матрицы связности по мере развития проекта. Точность разработанных алгоритмов можно повысить, если в качестве элемента связи брать не файл, а функцию или метод класса.

**Key words:** программная инженерия, разработка программного обеспечения, требования к программам, трассировка требований, система контроля версий, программный код, версии программного кода, архитектура программного приложения, контроль целостности архитектуры, связи элементов проекта

=================================

**FastText\_KMeans\_Clean:** Аналитик, Архитектор. Требование - Задача. Ведущий разработчик. 2. Задача, дефект – один файл. St = min(sf1, sf2) (2). 2. В текущей ревизии для каждого файла отбираются задачи/дефекты ассоциированные с выбранным файлом средствами системы контроля версий, (как пример, команда hg annotate для CVS Mercurial) и вычисляется сила связи по формуле 1 для каждой пары дефект-файл или задача-файл. Таблица 8 – Сила связи между артефактами первого уровня.

**Key words part:** 0.5

=================================

**FastText\_KMeans\_Raw/:** Требование - Задача. Ведущий разработчик. - Если иерархическая связь содержит понижающий коэффициент, полученный на подготовительном этапе при формировании дерева, то он применяется к результатам предыдущих двух операций с использованием формулы 4. Выполнение алгоритма рассмотрим на структуре артефактов, представленной на рис. 4, где R – требование, Тс – тестовый сценарий, Т – задача, D – дефект.

**Key words part:** 0.3611111111111111

=================================

**FastText\_PageRank\_Clean/:** Требование - Требование. Аналитик, Архитектор. Ведущий разработчик. Ведущий разработчик. Ведущий тестировщик. Тестовый сценарий- Дефект. St = min(sf1, sf2) (2). F(x1,x2) = x1 + x2 - x1 x2 .

**Key words part:** 0.3333333333333333

=================================

**FastText\_PageRank\_Raw/:** Требование - Требование. Аналитик, Архитектор. Требование-Компонента. Ведущий разработчик. Ведущий разработчик. Ведущий тестировщик. Тестовый сценарий- Дефект. St = min(sf1, sf2) (2).

**Key words part:** 0.3333333333333333

=================================

**Mixed\_ML\_TR/:** Требование - Требование. В рассматриваемой модели непосредственно с участками кода соединяются задачи и дефекты, и в этом случае, возможны следующие комбинации связей через общий код:. В связи с тем, что с одной задачей или дефектом может быть ассоциировано большое количество файлов, тестовые сценарии могут быть сквозными и покрывать несколько и с учетом иерархической структуры требований и задач, необходимо разработать алгоритм вычисления силы связей между требованиями. Так как между одними и теми же артефактами может осуществляться связь через несколько файлов, определим требования к функции F, вычисляющей результирующее значение силы связи:. F(x1,F(x2,x3)) = F(F(x1,x2), x3) - ассоциативность, при выполнении алгоритма, должна быть возможность вычислять силу связи попарно, при этом не имеет значения порядок вычисления;. Подготовительная часть алгоритма, заключается в приведении графа иерархической связанности артефактов к виду леса деревьев и нормализации высоты деревьев. Для этого, если артефакты связаны через несколько файлов, сначала применяется формула 2, определяющая связь через каждый файл в отдельности, после чего применяется формула 3, последовательно к каждой паре значений, проводя свёртку и вычисление единого значения силы связи. 0. Таблица 7 – Сила связи между артефактами второго уровня. 0. На четвёртом уровне, табл.

**Key words part:** 0.4166666666666667

=================================

**MultiLingual\_KMeans/:** Требование - Требование. В рассматриваемой модели непосредственно с участками кода соединяются задачи и дефекты, и в этом случае, возможны следующие комбинации связей через общий код:. F(x1,F(x2,x3)) = F(F(x1,x2), x3) - ассоциативность, при выполнении алгоритма, должна быть возможность вычислять силу связи попарно, при этом не имеет значения порядок вычисления;. Подготовительная часть алгоритма, заключается в приведении графа иерархической связанности артефактов к виду леса деревьев и нормализации высоты деревьев. Для этого, если артефакты связаны через несколько файлов, сначала применяется формула 2, определяющая связь через каждый файл в отдельности, после чего применяется формула 3, последовательно к каждой паре значений, проводя свёртку и вычисление единого значения силы связи. 0. На четвёртом уровне, табл.

**Key words part:** 0.4166666666666667

=================================

**Multilingual\_PageRank/:** St = min(sf1, sf2) (2). F(x1,x2) = x1 + x2 - x1 x2 . Приведение к виду леса деревьев осуществляется следующим образом:. Рисунок 3 – Приведение деревьев к максимальной высоте. Остальные явно проставленные ассоциативные и прочие связи не учитываются. Итоговая величина изменения R рассчитывается по формуле 5. Тестовые сценарии непосредственно при формировании матриц связности принимать участие не будут. 0. На четвёртом уровне, табл.

**Key words part:** 0.3055555555555556

=================================

**RuBERT\_KMeans\_Without\_ST/:** Подобная трассировка позволяет достаточно эффективно решать задачу 1, посредством связей отвечая на вопросы какие требования реализованы и какие требования протестированы, но не всегда эффективно позволяет решать задачу 2, в связи с тем, что между двумя требованиями может существовать связь на уровне программного кода, которая не проставлена на уровне требований, задач или тестовых сценариев. Ведущий разработчик. 2. Задача, дефект – один файл. F(x1,F(x2,x3)) = F(F(x1,x2), x3) - ассоциативность, при выполнении алгоритма, должна быть возможность вычислять силу связи попарно, при этом не имеет значения порядок вычисления;. 2. В текущей ревизии для каждого файла отбираются задачи/дефекты ассоциированные с выбранным файлом средствами системы контроля версий, (как пример, команда hg annotate для CVS Mercurial) и вычисляется сила связи по формуле 1 для каждой пары дефект-файл или задача-файл. С использованием полученной матрицы возможно выполнение следующих видов анализа:. Таблица 8 – Сила связи между артефактами первого уровня.

**Key words part:** 0.7222222222222222

=================================

**RuBERT\_KMeans\_With\_ST/:** 1. Задача, задача – один файл. В связи с тем, что с одной задачей или дефектом может быть ассоциировано большое количество файлов, тестовые сценарии могут быть сквозными и покрывать несколько и с учетом иерархической структуры требований и задач, необходимо разработать алгоритм вычисления силы связей между требованиями. В качестве основы для вычисления возьмём силу связи файл-задача или файл-дефект Sf, которая будет измеряться в отношении строк кода k ассоциированных с задачей или дефектом к общему числу строк кода N (формула 1). Фиксируются строки для двух элементов, для которых произойдёт свёртка на верхнем уровне.

**Key words part:** 0.4444444444444444

=================================

**RUBERT\_page\_rank\_Without\_ST/:** 2. Анализ влияния изменения;. Аналитик, Архитектор. Ведущий разработчик. Ведущий разработчик. Значение силы связи может принимать значения от 0 до 1.

**Key words part:** 0.3055555555555556

=================================

**RUBERT\_page\_rank\_With\_ST/:** Аналитик, Архитектор. Требование - Задача. Ведущий разработчик. Указанным условиям удовлетворяет следующая функция (3):. F(x1,x2) = x1 + x2 - x1 x2 .

**Key words part:** 0.3333333333333333

=================================

**RUSBERT\_KMeans\_Without\_ST/:** Ведущий разработчик. В рассматриваемой модели непосредственно с участками кода соединяются задачи и дефекты, и в этом случае, возможны следующие комбинации связей через общий код:. 3. Дефект, дефект – один файл. F(x1,F(x2,x3)) = F(F(x1,x2), x3) - ассоциативность, при выполнении алгоритма, должна быть возможность вычислять силу связи попарно, при этом не имеет значения порядок вычисления;. Подготовительная часть алгоритма, заключается в приведении графа иерархической связанности артефактов к виду леса деревьев и нормализации высоты деревьев. Таблица 8 – Сила связи между артефактами первого уровня.

**Key words part:** 0.3611111111111111

=================================

**RUSBERT\_KMeans\_With\_ST/:** Ведущий разработчик. В рассматриваемой модели непосредственно с участками кода соединяются задачи и дефекты, и в этом случае, возможны следующие комбинации связей через общий код:. F(x1,x2) = F(x2,x1) - коммутативность, порядок следования аргументов в операции не имеет значения. Таблица 8 – Сила связи между артефактами первого уровня.

**Key words part:** 0.3611111111111111

=================================

**RUSBERT\_page\_rank\_Without\_ST/:** 1. Получение актуального статуса системы по требованиям;. Требование - Задача. F(x1,x2) = x1 + x2 - x1 x2 . При этом выполняются следующие правила:. 0. На четвёртом уровне, табл.

**Key words part:** 0.3611111111111111

=================================

**RUSBERT\_page\_rank\_With\_ST/:** 1. Получение актуального статуса системы по требованиям;. Уровень трассировки. F(x1,x2) = x1 + x2 - x1 x2 . Приведение к виду леса деревьев осуществляется следующим образом:. Рисунок 3 – Приведение деревьев к максимальной высоте.

**Key words part:** 0.3888888888888889

=================================

**Simple\_PageRank/:** Более детально, в зависимости от методологии проектирования и обработки требований в организации, возможно деление требований на различные типы (как пример [2]: функциональные требования, не функциональные требования, атрибуты качества, ограничения, бизнес требования и бизнес правила), но в общем случае допускается, что зависимости могут возникать между требованиями разных типов. В связи с тем, что с одной задачей или дефектом может быть ассоциировано большое количество файлов, тестовые сценарии могут быть сквозными и покрывать несколько и с учетом иерархической структуры требований и задач, необходимо разработать алгоритм вычисления силы связей между требованиями. В связи с тем, что требования, как правило, формируют иерархическую структуру, выполним приведение деревьев в части только требований к максимальной высоте (рис. 3). В результате этих преобразований тестовый сценарий Тс2 и его дочерние элементы разбиваются на две подветви с силой связи с родительскими элементами R7 и R5 по 0,5. Пример: в рамках исправления нового дефекта D4 внесены изменения в файл F1, в этом случае нужно в первую очередь повторно протестировать требования R4 и R7, как иерархически ассоциированные с файлом F1, и требование R6, которое иерархически с кодом не связано, но имеет отличную от нуля связь с требованием R4. В случае, если требования R1 и К8 связаны с компонентами, которые не должны иметь логической связи друг с другом, то наличие связи через общий код должно послужить сигналом к расследованию корректности реализации спроектированной архитектуры.

**Key words part:** 0.5

=================================

**TextRank/:** Подобная трассировка позволяет достаточно эффективно решать задачу 1, посредством связей отвечая на вопросы какие требования реализованы и какие требования протестированы, но не всегда эффективно позволяет решать задачу 2, в связи с тем, что между двумя требованиями может существовать связь на уровне программного кода, которая не проставлена на уровне требований, задач или тестовых сценариев. В связи с тем, что с одной задачей или дефектом может быть ассоциировано большое количество файлов, тестовые сценарии могут быть сквозными и покрывать несколько и с учетом иерархической структуры требований и задач, необходимо разработать алгоритм вычисления силы связей между требованиями. В качестве основы для вычисления возьмём силу связи файл-задача или файл-дефект Sf, которая будет измеряться в отношении строк кода k ассоциированных с задачей или дефектом к общему числу строк кода N (формула 1). Сила связи St между разными задачами/дефектами ассоциированными с одним файлом будет определяться наименьшим значением силы связи между парой ассоциированных с файлом артефактов, рассчитанной по формуле 2. Так как между одними и теми же артефактами может осуществляться связь через несколько файлов, определим требования к функции F, вычисляющей результирующее значение силы связи:. Пример: в рамках исправления нового дефекта D4 внесены изменения в файл F1, в этом случае нужно в первую очередь повторно протестировать требования R4 и R7, как иерархически ассоциированные с файлом F1, и требование R6, которое иерархически с кодом не связано, но имеет отличную от нуля связь с требованием R4.

**Key words part:** 0.5833333333333334

=================================

**TF-IDF\_KMeans/:** 2. Анализ влияния изменения;. Цель этой статьи заключается в формировании набора алгоритмов, позволяющих выстраивать связи между сущностями процесса разработки программного обеспечения (артефактами) на основе исходного кода, и анализировать эти связи. Уровни трассировки и роли участников проекта за ведение связей представлены в табл. Требование - Задача. Ведущий разработчик. Требование – Тестовый сценарий. F(x1,x2) ≥ max(x1,x2) – в идеале, значение силы связи должно быть больше значения максимального элемента, при двух аргументах отличных от нуля. 0. Таблица 7 – Сила связи между артефактами второго уровня.

**Key words part:** 0.6944444444444444

=================================

**Текст:** 1. Получение актуального статуса системы по требованиям;. 2. Анализ влияния изменения;. Вигерс рассматривал полную трассировку от требования до исходного кода (функции) программы, но большинство систем управления требованиями поддерживает трассировку до уровня тест кейса или до уровня задачи разработчику на реализацию соответствующего требования. Подобная трассировка позволяет достаточно эффективно решать задачу 1, посредством связей отвечая на вопросы какие требования реализованы и какие требования протестированы, но не всегда эффективно позволяет решать задачу 2, в связи с тем, что между двумя требованиями может существовать связь на уровне программного кода, которая не проставлена на уровне требований, задач или тестовых сценариев. Подробный обзор работ, затрагивающих тему измерения силы влияния изменения проведён в работе [2], на основании которого выделены четыре класса анализа влияния изменений:. 1. Использование статического анализа программ основанного на структуре программы и отношении между элементами программы;. 2. Использование динамического анализа программ основанного на сборе данных во время выполнения программы;. 3. Использование анализа на основе обработки исторических данных, получаемых из систем контроля версий программного кода;. Большая часть рассмотренных методов является зависимой от языка программирования, что ограничивает их использование в проектах, разрабатываемых с использованием нескольких языков программирования [3-6]. Цель этой статьи заключается в формировании набора алгоритмов, позволяющих выстраивать связи между сущностями процесса разработки программного обеспечения (артефактами) на основе исходного кода, и анализировать эти связи. При этом он должен быть независимым от языка программирования и простым в реализации.. Рассмотрим модель, описывающую взаимосвязь элементов процесса разработки программного обеспечения, в дальнейшем будем называть их артефактами (рис. 1).. . Основу модели составляют требования, которые могут выстраиваться в иерархическую структуру с декомпозицией более общих требований более детальными требованиями. При этом ассоциативная связь может возникать между разными требованиями, не находящимися в иерархических отношениях друга другом. В работе под понятием требование будет использоваться общее определение требования, данное в своде знаний по программной инженерии третьей версии [7] как свойство, представленное чем-либо, для решения некоторой проблемы реального мира. Более детально, в зависимости от методологии проектирования и обработки требований в организации, возможно деление требований на различные типы (как пример [2]: функциональные требования, не функциональные требования, атрибуты качества, ограничения, бизнес требования и бизнес правила), но в общем случае допускается, что зависимости могут возникать между требованиями разных типов. Задачи так же могут выстраивать иерархию посредством декомпозиции более крупных задач на более мелкие [8-14]. При этом каждое требование должно быть ассоциировано как минимум с одной задачей, реализующей это требование. В процессе работы, каждому артефакту процесса разработки присваивается уникальный идентификатор. При написании кода в рамках задачи каждому блоку кода, добавляемому в репозиторий, разработчик проставляет комментарий с явным указанием номера задачи или дефекта, в рамках которого было проведено изменение. Дефекты заводятся по итогам выполнения тестовых сценариев, проверяющих полноту реализации требования. Одно требование может быть проверено несколькими тестовыми сценариями, так и один сценарий может проверять несколько требований (сквозные сценарии). Уровни трассировки и роли участников проекта за ведение связей представлены в табл. 1.. Таблица 1 - Распределение ответственности за уровни трассировки по ролям участников проекта. Уровень трассировки. Роль участника проекта, ответственного за ведение трассировки. Требование - Требование. Аналитик, Архитектор. Требование-Компонента. Архитектор. Требование - Задача. Ведущий разработчик. Задача-Задача. Ведущий разработчик. Задача - Код. Разработчик. Требование – Тестовый сценарий. Ведущий тестировщик. Тестовый сценарий- Дефект. Тестировщик. Помимо явных связей между требованиями, формируемыми аналитиками, в рамках предложенного алгоритма формируются не явные связи, посредством трассировки по путям требование–задача–код и требование–тестовый сценарий–дефект–код. Основная идея метода заключается в том, что в современных языках программирования как правило отдельные классы и методы реализуются в отдельных файлах. И в случае, если две разные задачи имеют общий код в рамках одного файла, то велика вероятность, что между этими задачами есть логическая связь. В рассматриваемой модели непосредственно с участками кода соединяются задачи и дефекты, и в этом случае, возможны следующие комбинации связей через общий код:. 1. Задача, задача – один файл. 2. Задача, дефект – один файл. 3. Дефект, дефект – один файл. В связи с тем, что с одной задачей или дефектом может быть ассоциировано большое количество файлов, тестовые сценарии могут быть сквозными и покрывать несколько и с учетом иерархической структуры требований и задач, необходимо разработать алгоритм вычисления силы связей между требованиями. В качестве основы для вычисления возьмём силу связи файл-задача или файл-дефект Sf, которая будет измеряться в отношении строк кода k ассоциированных с задачей или дефектом к общему числу строк кода N (формула 1). В расчёте принимают участие только строки, содержащие операторы языка программирования.. Sf = k / N. (1). Сила связи St между разными задачами/дефектами ассоциированными с одним файлом будет определяться наименьшим значением силы связи между парой ассоциированных с файлом артефактов, рассчитанной по формуле 2.. St = min(sf1, sf2) (2). Значение силы связи может принимать значения от 0 до 1. В случае значения 0 – связь отсутствует, в случае 1 – связь максимальная. Так как между одними и теми же артефактами может осуществляться связь через несколько файлов, определим требования к функции F, вычисляющей результирующее значение силы связи:. В предельном случае, число аргументов функции равно двум и область определения входных аргументов принадлежит интервалу [0…1];. F(x1,x2) = F(x2,x1) - коммутативность, порядок следования аргументов в операции не имеет значения.. F(x,0) = x; - в случае, если связь между артефактами через некоторый файл отсутствует, то существующая связь недолжна измениться;. F(x,1) = 1 – если уже достигнута максимальная связь между артефактами, она не должна быть уменьшена;. F(x1,F(x2,x3)) = F(F(x1,x2), x3) - ассоциативность, при выполнении алгоритма, должна быть возможность вычислять силу связи попарно, при этом не имеет значения порядок вычисления;. F(x1,x2) ≥ max(x1,x2) – в идеале, значение силы связи должно быть больше значения максимального элемента, при двух аргументах отличных от нуля. В этом случае, при наличии связей через большое число файлов, результирующая сила связи будет стремиться к максимальному значению.. Указанным условиям удовлетворяет следующая функция (3):. F(x1,x2) = x1 + x2 - x1 x2 . (3). Подготовительная часть алгоритма, заключается в приведении графа иерархической связанности артефактов к виду леса деревьев и нормализации высоты деревьев. Приведение к виду леса деревьев осуществляется следующим образом:. s = 1/m (4). . Рисунок 2 – Графическая иллюстрация преобразования связей между артефактами. Вычисление связи возможно между артефактами одного уровня. В связи с тем, что требования, как правило, формируют иерархическую структуру, выполним приведение деревьев в части только требований к максимальной высоте (рис. 3). Для этого:. . Рисунок 3 – Приведение деревьев к максимальной высоте. После выравнивания деревьев на уровне требований, по тому же алгоритму выполняется приведение для поддеревьев на уровне задач.. Приведём алгоритм определения силы связи между артефактами системы. В процессе вычисления принимают участия только связи через общий код и связи типа декомпозиция. Остальные явно проставленные ассоциативные и прочие связи не учитываются.. Алгоритм вычисления связей между требованиями для текущей ревизии (версии) и ветки репозитория:. 1. Отбирается список файлов в проекте. 2. В текущей ревизии для каждого файла отбираются задачи/дефекты ассоциированные с выбранным файлом средствами системы контроля версий, (как пример, команда hg annotate для CVS Mercurial) и вычисляется сила связи по формуле 1 для каждой пары дефект-файл или задача-файл.. 3. Формируется матрица связи между задачами/дефектами, непосредственно ассоциированными с исходным кодом. Для этого, если артефакты связаны через несколько файлов, сначала применяется формула 2, определяющая связь через каждый файл в отдельности, после чего применяется формула 3, последовательно к каждой паре значений, проводя свёртку и вычисление единого значения силы связи.. 3. На основании полученных на шаге 3 значений, формируется квадратная матрица связности между сущностями разработки – задачами и дефектами, непосредственно ассоциированными с исходным кодом, симметричная относительно главной диагонали.. 4. Для каждой отличной от нуля записи в таблице формируем связь между элементами, стоящими выше по уровню иерархии. При этом выполняются следующие правила:. - Если для каждого элемента верхнего уровня иерархии связь осуществляется только через один элемент нижнего уровня иерархии, то значение связи передаётся выше по уровню иерархии без изменений.. - Если для элемента верхнего уровня иерархии связь осуществляется через несколько элементов нижнего уровня, то для свёртки применяется формула 3, последовательно к каждой паре значений. При операции на матрице, алгоритм выглядит следующим образом. Фиксируются строки для двух элементов, для которых произойдёт свёртка на верхнем уровне. Создаётся матрица, в которой две свёртываемые строки и два свёртываемых соответственно столбца, заменены одной строкой и столбцом. Для столбцов, номера которых не равны выбранным строкам по исходной матрице, производится операция свёртки по формуле 3 результат копируется в новую матрицу. Операция производится попарно до тех пор, пока все дочерние элементы не будут свёрнуты.. - Если иерархическая связь содержит понижающий коэффициент, полученный на подготовительном этапе при формировании дерева, то он применяется к результатам предыдущих двух операций с использованием формулы 4.. Итогом выполнения алгоритма является набор по уровневой связности артефактов системы посредством программного кода. С использованием полученной матрицы возможно выполнение следующих видов анализа:. Определение требований, которые необходимо повторно протестировать при изменении кода в выбранном файле осуществляется следующим образом:. Для определения величины изменения системы, выбирается требование, которые необходимо изменить и отбираются требования непосредственно связанные с ним. Проводится трассировка до уровня исходного кода, после чего вычисляется число файлов, затронутых этими требованиями, и процент кода в этих файлах, ассоциированных с этими требованиями. Итоговая величина изменения R рассчитывается по формуле 5.. R = Na/N, (5). где Na – число строк кода, связанных с выбранными требованиями. N – общее число строк кода в проекте.. Выполнение алгоритма рассмотрим на структуре артефактов, представленной на рис. 4, где R – требование, Тс – тестовый сценарий, Т – задача, D – дефект. Исходные данные в виде числа строк в файлах и их ассоциация с артефактами приведена в таблице 2.. Таблица 2 – Число строк, ассоциированных с каждым артефактом. 169. 79. 0. 82. 0. 0. 0. 0. 8. 0. 0. 138. 55. 83. 0. 0. 0. 0. 0. 0. 0. 0. 107. 0. 107. 0. 0. 0. 0. 0. 0. 0. 0. 146. 0. 0. 121. 0. 0. 0. 0. 0. 25. 0. 158. 0. 0. 0. 67. 91. 0. 0. 0. 0. 0. 161. 0. 0. 0. 95. 54. 0. 0. 0. 0. 12. 92. 0. 0. 0. 0. 0. 85. 0. 0. 0. 7. 193. 0. 0. 0. 0. 0. 0. 193. 0. 0. 0. . Для исходной структуры проведем предварительные преобразования. В результате этих преобразований тестовый сценарий Тс2 и его дочерние элементы разбиваются на две подветви с силой связи с родительскими элементами R7 и R5 по 0,5. Итоговое дерево артефактов (рис. 5) имеет высоту 9 уровней, из них 1 уровень файлов и один уровень тестовых сценариев. Тестовые сценарии непосредственно при формировании матриц связности принимать участие не будут. Нумерацию уровней будем вести от корней дерева – требования R1 и R9.. В таблице 3 представлены данные, полученные на основе связей между файлами и артефактами самого нижнего уровня. Из них, наибольшей связностью обладают задачи Т5 и Т6, для которых связь осуществляется через два файла.. . Таблица 3 – Сила связи между артефактами седьмого уровня. Артефакт. 0. 0,4. 0,47. 0. 0. 0. 0. 0,05. 0. 0. 0,4. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0,47. 0. 0. 0. 0. 0. 0. 0,05. 0,21. 0. 0. 0. 0. 0. 0,62. 0. 0. 0. 0. 0,07. 0. 0. 0. 0,62. 0. 0. 0. 0. 0. 0,07. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0,08. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0,05. 0. 0,05. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0,21. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0,07. 0,07. 0,08. 0. 0. 0. 0. Таблица 4 – Сила связи между артефактами шестого уровня. Артефакт. T1. T2. T3. T4. T7. T8. D1. D2. D3. T1. 0. 0,4. 0,47. 0. 0. 0. 0,05. 0. 0. T2. 0,4. 0. 0. 0. 0. 0. 0. 0. 0. T3. 0,47. 0. 0. 0. 0. 0. 0,05. 0,21. 0. T5. 0. 0. 0. 0. 0. 0. 0. 0. 0,07. T7. 0. 0. 0. 0. 0. 0. 0. 0. 0,08. T8. 0. 0. 0. 0. 0. 0. 0. 0. 0. D1. 0,05. 0. 0,05. 0. 0. 0. 0. 0. 0. D2. 0. 0. 0,21. 0. 0. 0. 0. 0. 0. D3. 0. 0. 0. 0,07. 0,08. 0. 0. 0. 0. Таблица 5 – Сила связи между артефактами четвёртого уровня. Артефакт. R4. R6. R7. R3. R9. R10. R4. 0. 0,4. 0,49. 0. 0. 0. R6. 0,4. 0. 0. 0. 0. 0. R7. 0,49. 0. 0. 0,1. 0. 0. R3. 0. 0. 0,1. 0. 0,07. 0. R9. 0. 0. 0. 0,07. 0. 0. R10. 0. 0. 0. 0. 0. 0. Таблица 6 – Сила связи между артефактами третьего уровня. R4. R5. R3. R9. R10. R4. 0. 0,69. 0. 0. 0. R5. 0,69. 0. 0,1. 0. 0. R3. 0. 0,1. 0. 0,07. 0. R9. 0. 0. 0,07. 0. 0. R10. 0. 0. 0. 0. 0. Таблица 7 – Сила связи между артефактами второго уровня. R2. R3. R9. R10. R2. 0. 0,1. 0. 0. R3. 0,1. 0. 0,07. 0. R9. 0. 0,07. 0. 0. R10. 0. 0. 0. 0. Таблица 8 – Сила связи между артефактами первого уровня. R1. R8. R1. 0. 0,07. R8. 0,07. 0. На четвёртом уровне, табл. 5, представлена связь между требованиями самого нижнего уровня. Она уже может использоваться для проведения анализа влияний при изменении кода в определённом файле.. Пример: в рамках исправления нового дефекта D4 внесены изменения в файл F1, в этом случае нужно в первую очередь повторно протестировать требования R4 и R7, как иерархически ассоциированные с файлом F1, и требование R6, которое иерархически с кодом не связано, но имеет отличную от нуля связь с требованием R4.. Итоговая таблица 8, представляет связь между требованиями первого уровня. В случае, если требования R1 и К8 связаны с компонентами, которые не должны иметь логической связи друг с другом, то наличие связи через общий код должно послужить сигналом к расследованию корректности реализации спроектированной архитектуры.. Предложенный алгоритм позволяет решать поставленные задачи выполнения импакт-анализа. Но его основным недостатком является высокая вычислительная сложность, с проработкой N2 элементов. В крупных программных проектах число артефактов процесса разработки ПО достигает нескольких десятков, если не сотен тысяч. Как средство решения этой проблемы возможно предложить постепенное формирование глобальной матрицы связности по мере развития проекта. Вторым основным недостатком метода является невысокая точность, которую можно повысить, если в качестве элемента связи брать не файл, а функцию или метод класса.