Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт

Системы искусственного интеллекта
кафедра

ОТЧЕТ О НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ

Кафедра «Системы искусственного интеллекта» ИКИТ СФУ
Место прохождения практики

Обзор литературы на тему связанности в объектно-ориентированных системах
тема

Руководитель _____ А.В. Кушнаренко
подпись, дата *инициалы, фамилия*

Студент КИ17-11Б 031723019 _____ В.А. Рудт
номер группы, зачетной книжки *подпись, дата* *инициалы, фамилия*

Красноярск 2019

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

В данной работе рассматривается литература, представляющая с различных сторон тему измерения связанности (Coupling) в объектно-ориентированных системах.

Цель работы: с использованием имеющейся литературы рассмотреть, систематизировать и проанализировать текущие знания об измерении связанности в объектно-ориентированных системах.

Задачи:

- Изучить имеющеюся литературу на тему измерения связанности в объектно-ориентированных системах;
- Представить общий обзор имеющихся знаний на тему измерения связанности в объектно-ориентированных системах и проанализировать их;
- Определить перспективы развития исследований в области измерения связанности в объектно-ориентированных системах.

В последнее время, уровень качества программного обеспечения (ПО) является важным аспектом. Однако, как можно определить этот уровень? При оценке качества ПО нужно понять, с какой позиции мы его рассматриваем. С позиции пользователя, качество можно рассматривать как степень подверженности программы ошибкам, когда разработчик может воспринимать данное качество иначе. Для разработчика важной частью качественного ПО является его возможность и удобство расширяемости, а также его понятность. Если ПО не содержит данных качеств, то, даже если конкретная версия программы не содержит ошибок, новые ошибки неминуемо будут появляться.

Нетрудно догадаться, что от качества, рассматриваемого с точки зрения разработчика, зависит качество, как его представляет пользователь, поэтому его определение является важной задачей.

С развитием масштаба программных продуктов было выявлено несколько метрик качества ПО. Одной из самых важных метрик является связанность (coupling). Данная метрика определяет степень взаимодействия между программными модулями. Различным определениям, методам измерения данной метрики и её свойствам, а также их анализу, посвящена данная работа.

В работе в большей степени рассматривается связанность в объектно-ориентированных системах, так как данная парадигма наиболее перспективная в современном мире.

1 Понятие и определение связанности

Дать четкое определение связанности достаточно проблематично, так как его понятие расплывчато. Однако, большинство понимают данный термин как силу связи или степень взаимодействия программных модулей между собой. В объектно-ориентированных системах такими модулями являются классы и объекты.

Неясность данного понятия состоит в том, что понятие связанности сейчас чисто интуитивное и разные исследователи предлагают свои варианты, они схожи по смыслу, но приравнять их друг к другу нельзя.

Существует определения, которые были получены в результате совместной работы институтов стандартизации ISO, IEEE и IEC [1, с. 107].

Связанность: сила связей между модулями.

Связанность: характер и степень взаимозависимости между модулями.

Связанность: измерение того, насколько тесно связаны две подпрограммы или модули.

Связанность: мера взаимодействия между модулями в компьютерной программе.

Данные определения очень схожи между собой, однако, стоит дать собственное определение, которое будет использоваться в рамках данной работы. Также следует учитывать, что термин "coupling" пока не имеет четкого аналога на русском языке, так что в данной работе аналогом, как можно было заметить, является термин "связанность".

Связанность: степень взаимодействия между модулями объектно-ориентированной системы (классами и объектами).

Данное определение очень узкое и подходит только для данного исследования, так как отражает два аспекта: связанность является степенью взаимодействия, и эта метрика относится к объектно-ориентированным системам.

2 Общие положения о связанности и её измерении

Измерением связанности, в основном, интересовались инженеры во время девяностых и нулевых годов. Сейчас выпускается относительно мало литературы на данную тему, что является негативным фактом. Нет литературы, которая бы некоторым образом подводила итоги об измерении связанности и представляла бы общую картину в целом.

Однако работы велись и были достигнуты некоторые успехи в этом направлении.

Одно из первых упоминаний связанности встречается в работе Уэйна Стивенса [2, с. 232-237]. В данной книге связанность трактуется следующим образом:

Связанность: это мера прочности связи, которая устанавливается от одного модуля к другому.

Данное определение схоже с одним из тех, что мы упомянули выше. Имеется ввиду то определение, которое характеризует связанность как силу отношений между модулями.

Данная книга не уделяет времени вопросам на тему измерения связанности. К тому же, книга посвящена организации систем в процедурной парадигме, а не в объектно-ориентированной. Однако, хотя в книге и не уделяется внимания непосредственно измерению связанности в объектно-ориентированных системах, в ней описаны некоторые закономерности, связанные со связанностью, которые негативно влияют на качество ПО. А так

как объектно-ориентированное программирование в какой-то степени выросло из процедурного, значит, данные закономерности будут работать и для этой парадигмы. Разберем их подробнее, так как они дают общее представление о том, каким правилам должна подчиняться связанность, чтобы её можно было оценить, как положительную.

Первая закономерность такова, что уменьшение количества связей между программными модулями уменьшает и количество путей, по которым изменения или ошибки могут распространиться на другие части системы.

Закономерность крайне важна, так как ошибки и изменения, переданные по связи другому модулю, могут перейти и к следующим модулям тоже. Таким образом, изменения и ошибки нарастают волнообразно, как это проиллюстрировано на Рисунке 1.

Также, увеличение связей приводит к росту сложности системы, а значит, к сложности её поддержки и понимания.

Чтобы проиллюстрировать эту закономерность, автор приводит следующую ситуацию. Допустим, наша система имеет некую общую область данных, к которой обращаются модули, тем самым входя в некоторую общую среду. В таком случае, сами модули также взаимодействуют между собой, и включение нового модуля означает, что он будет включен в это общее окружение.

В таком случае, все имеют доступ к общей памяти, но не каждому следует её изменять. Изменение одним модулем общей памяти может вызвать ошибки в другом.

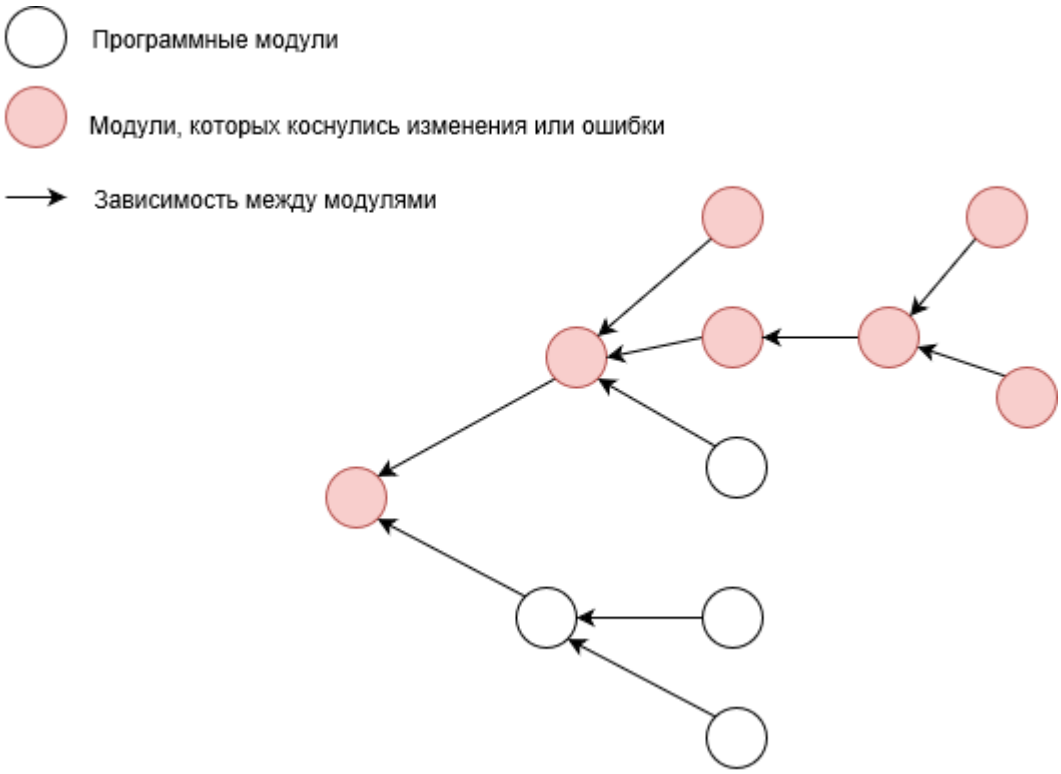

Рисунок 1 – Волнообразный эффект изменений и ошибок

Отойдем от абстракций и немного конкретизируем ситуацию. Допустим, в системе имеется три модуля: модуль ввода, модуль вычислений и модуль вывода. Изобразим это на Рисунке 2.

На общую память влияют все три модуля, однако, если мы хотим ввести данные, вычислить новый результат и вывести его с исходными данными программист может случайно или намеренно изменить исходные данные еще на этапе вычислений, что приведет к некорректной работе программы.

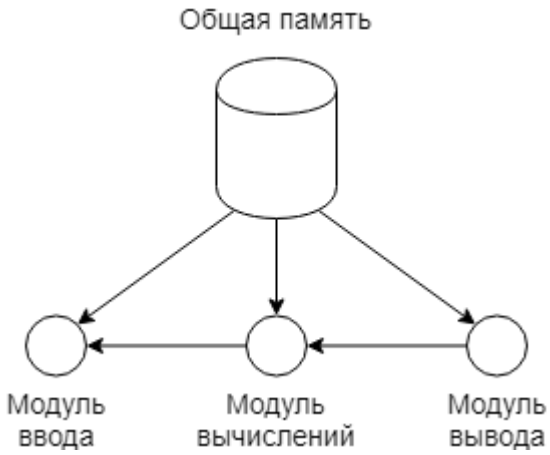

Рисунок 2—Пример общей среды

Ситуация может существенно улучшиться, если доверить работу с памятью и её распределение только одному модулю так, как это показано на Рисунке 3.

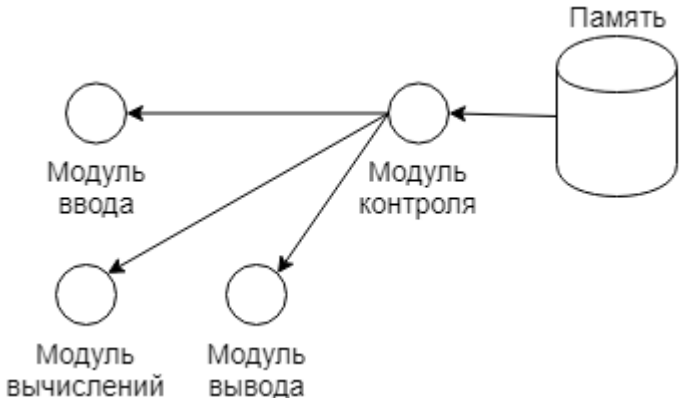

Рисунок 3— Система без общей среды

Как можно заметить, число связей уменьшилось, что сделало нашу систему более помехоустойчивой, так как на память напрямую влияет только модуль контроля.

После вышесказанного мы подходим к одному из главных принципов проектирования систем, однако для нас важна только одна его часть.

Принцы звучит так: “Low coupling, high cohesion”. Нас же интересует только первая его часть. После некоторых рассуждений, которые были описаны выше, становится понятно, что качество ПО в некоторой степени зависит от связанности. Чем ниже связанность, тем выше надежность системы. Это также относится и к объектно-ориентированным системам.

Теперь, после обзора некоторых закономерностей, которые наводят нас на принцип проектирование, гласящий, что связанность должна быть как можно меньше, мы понимаем важность данной метрики качества.

Определение метода, позволяющего четко определить связанность, становится важной задачей, решив которую можно частично определить уровень качества ПО (его расширяемость, поддержку и понятность) численно.

3 Способы измерения связанности

В процессе изучения различной литературы были выделены два типа методов измерения связанности. Условно определим эти типы как: точные и неточные.

Неточные методы, используемым в системах, появились на ранних стадиях изучения связанности. Такие методы были основаны на выделении некоторых классов на основе различных ситуаций. После определения, к какому классу принадлежит конкретный случай, можно было также определить уровень связанности по некоторому правилу.

Точное методы измерения появились позднее. К данному типу можно отнести методы, суть которых состоит в выделении некоторых базовых определений, через которые можно выразить метрику связанности, после чего элементы, удовлетворяющие определению метрики связанности, подсчитываются. Получившееся число является показателем уровня связанности.

Кроме этого, точные методы можно разбить еще на два типа: методы статического и динамического измерения связанности.

Статические методы измерения связанности характерны тем, что уровень связанности определяется одним числом, характеризующим систему в целом, и в основном может быть рассчитан без запуска программы.

Динамические методы измерения связанности характерны тем, что уровень связанности представляет собой случайную величину, которая зависит от конкретного пути выполнения программы, и в основном может быть рассчитан во время выполнения программы.

Все типы методов следует рассмотреть.

4 Неточные методы измерения

Стивенс был не единственным, кто занимался вопросом связанности. Есть и многие другие авторы, которые внесли свой вклад в развитие данного направления, и они тоже добились некоторых успехов.

4.1 Йоханн Эдер

Одним из способов приблизительного измерения связанности является составление классификации. Существует возможность рассмотреть различные ситуации, встречающиеся в объектно-ориентированных системах, обобщить их, разбить на классы и отсортировать в каком-либо порядке.

Йоханн Эдер вывел такую классификацию в своей работе [3, с. 6-19].

Автор выделяет три класса связанности:

– Связанность взаимодействия (interaction coupling);

– Связанность компонентов (component coupling);

– Связанность наследования (inheritance coupling).

Каждый из этих классов делится на подклассы, которые также требуют разбора. Некоторые подклассы одного класса могут пересекаться и данные классы следует скорее рассматривать как различные точки зрения на одну и ту же картину. Каждый класс в отдельности не может дать полную картину на связанность объектно-ориентированной системы.

Для каждого класса подклассы будут приводиться по правилу "от худшего к лучшему". Это означает, что сначала будут разбираться подклассы, которые соответствуют высокой связанности, и постепенно, от подкласса к подклассу, уровень связанности должен снижаться.

Составление данной классификации является одним из первых шагов к измерению связанности. Данная классификация не позволяет точно измерить связанность, но все же помогает найти приблизительный уровень связанности. Метод можно модифицировать и для численного измерения, однако он так же будет лишь приблизительным.

4.1.1 Связанность взаимодействия

Начнем со связанности взаимодействия. Данный класс связанности описывает взаимодействие между методами класса путем вызова одного метода другим и/или использования общей памяти.

Важным замечанием является то, что мы должны различать взаимодействие методов, находящиеся в одном классе, от взаимодействия методов из разных классов.

Перечислим подклассы данного класса, попутно разбирая их.

Связанность содержания (content coupling). Данный подкласс связанности образуется, когда один метод напрямую обращается к внутренней структуре (к реализации) другого метода. Данный вид связанности – наихудший, так как небольшое изменение одного метода сразу же влияет на реализацию другого и один метод должен точно знать внутренние компоненты другого, что очень сильно затрудняет сохранению инкапсуляции и сокрытию данных, принятых в объектно-ориентированном программировании.

Связанность в общем пространстве (common coupling). Данный подкласс связанности устанавливается между методами, которые коммуницируют через неструктурированное, глобальное, общее пространство данных. Такая связанность так же нарушает инкапсуляцию и сокрытие данных.

Однако объектно-ориентированных языков с общим неструктурированным пространством данных либо совсем нет, либо они мало малоизвестны.

Внешняя связанность (external coupling). подкласс описывается как связанность в общем пространстве, но в структурированном общем пространстве. Вместе с тем к данному подклассу переходят все те же недостатки, присущие связанности в общем пространстве. Разновидность такой связанности, которая осуществляется между классами, связанными наследованием, автор называет наследованная внешняя связанность.

Связанность управления (control coupling). Это подкласс, который характеризует связанность между методами, коммуницирующими только посредством передачи параметров. При этом один метод контролирует внутреннюю логику другого.

Штамповая связанность (stamp coupling). Подкласс, означающий связанность методов, при которой структура данных передается в качестве параметра, хотя требовалась только её часть. Суть подкласса такова: метод за-

висит от некоторых данных, переданных извне, и должен быть изменен, если структура данных изменилась.

Связанность данных (data coupling). При таком подклассе связанности так же передается структура данных и при этом она нужна целиком. Данный тип связанности является наилучшим и предоставляет высокую понятность взаимодействия.

Отсутствие связанности (no direct coupling). Данный подкласс связанности является теоретическим и обозначает, что два метода не зависят друг от друга напрямую.

4.1.2 Связанность компонентов

Связанность компонентов является следующим классом связанности. В отличие от связанности взаимодействия, данный класс связанности относится только к объектам класса объектно-ориентированной системы. Данная взаимосвязь заключается в использовании одним классом экземпляра другого класса. Эдер также дает определение компонента.

Класс объекта C' является компонентом класса объекта C тогда и только тогда, когда C' фигурирует в C. C' фигурирует в C тогда и только тогда когда:

- C' является частью C (является полем);
- C' входит в состав входных или выходных параметров метода из класса объекта C;
- C' является локальной переменной какого-либо метода класса объекта C;

– C' входит в состав входных или выходных параметров метода, вызываемого в методе из класса объекта C.

Сразу за определением следует замечание, которое относится непосредственно к объектно-ориентированным системам. Так как при программировании в объектно-ориентированной парадигме появляется возможность использовать наследование, значит, любой суперкласс объектно-ориентированной системы C связан с любым подклассом класса C'. Такой класс связанности называют потенциальной связанностью компонентов (potential component coupling).

Как и в предыдущем классе, связанность компонентов делится на несколько подклассов.

Скрытая связанность. Класс объектно-ориентированной системы C связан с классом C' данным классом связанности, если C' не содержится в явном виде класса C. Данный класс связанности происходит, например, когда C' в своем методе в качестве возвращаемого значения вызывает метод объекта класса C при этом данные, которые могли использоваться в методе класса C, скрываются.

Рассеянная связанность (scattered coupling). Класс объектно-ориентированной системы C связан с классом C' данным классом связанности, если объект класса C' фигурирует в качестве реализации класса C. Под реализацией здесь имеются в виду поля класса и реализации методов. Локальные переменные так же включены в реализацию.

Отмеченная связанность (specified coupling). Класс объектно-ориентированной системы C связан с классом C' данным классом связанности, если C' фигурирует в сигнатуре методов класса C.

Отсутствие связанности (nil coupling). Теоретический оптимум, при котором два класса не связаны напрямую.

4.1.3 Связанность наследования

Данный класс связанности появляется, когда один класс объектно-ориентированной системы является подклассом другого.

Более точного определения автор не дает. Однако, есть некоторые замечания. Так же, как и в случае связанности компонентов, связанность наследования распространяется только на объекты класса.

Приводится конкретный пример для лучшего понимания влияния данного типа связанности на связанность в целом.

Допустим, у нас есть класс D и классы C' и C'', которые являются подклассами класса C. Существует метод m, принадлежащий классу C и не переопределенный в классах C' и C''. Тогда, если класс D содержит в себе экземпляры классов C' и C'' и вызывает их метод m, значит класс D связан с C' и C'' с помощью связанности компонентов. Однако, автор определяет, что благодаря наследованию класс D связан только с классом C, так как метод m не переопределен в классах-наследниках. Данный факт уменьшает количество связей.

Как и в предыдущих случаях, автор выделяет связанности наследования.

Связанность модификации (modification coupling). Подкласс связанности, при котором, помимо определения новой информации, она ещё и произвольно изменяется, а иногда даже удаляется. Автор различает модификацию сигнатуры (signature modification) и модификацию реализации (implementation modification).

Связанность модификации сигнатуры. Связанность между подклассом C' и суперклассом C может называться связанностью модификации сигнату-

ры тогда и только тогда, когда не только реализация, но и сигнатура наследуемого метода m изменяется без каких-либо ограничений или же вовсе удаляется.

Связанность модификации реализации. Связанность между подклассом C' и суперклассом C может называться связанностью модификации реализации тогда и только тогда, когда только реализация наследуемого метода m изменяется.

Связанность уточнения (refinement coupling). Подкласс связанности, при котором, помимо определения новой информации, она изменяется только в соответствии с заранее определенными правилами. Как и в предыдущем типе, автор различает связанность уточнения сигнатуры (signature refinement coupling) и связанность уточнения реализации (implementation refinement coupling).

Связанность уточнения сигнатуры. Связанность между подклассом C' и суперклассом C может называться связанностью уточнения сигнатуры тогда и только тогда, когда она не походит под определение связанности модификации и если сигнатура хотя бы одного унаследованного метода изменяется по некоторому правилу без изменения семантики данного метода. Например, можно изменять только список входящих или исходящих параметров в сигнатуре метода.

Связанность уточнения реализации. Связанность между подклассом C' и суперклассом C может называться связанностью уточнения реализации тогда и только тогда, когда сигнатура метода не изменяется и по крайней мере один метод изменяется по предопределенному правилу при сохранении семантики. Например, можно вызывать переопределяемый метод непосредственно в теле переопределения.

Связанность расширения (extension coupling). Данный класс связанности между подклассом и суперклассом объектно-ориентированной системы

устанавливается, когда подкласс не изменяет и не переопределяет методы и поля базового класса.

Отсутствие связанности. Данный вид связанности устанавливается, когда два класса не связаны наследованием.

4.1.4 Разбор пригодности для вычисления

Нужно сказать, что Йоханн Эдер внес немалый вклад в исследование связанности, перенеся данный термин с процедурных систем на объектно-ориентированные. Также он предложил классификацию связанности, описанную выше, что является важным этапом в исследовании любого явления.

Однако, следует оценить возможность выполнить измерения по данной классификации.

Прежде всего, стоит отметить, что автор не ставит перед собой цель четкого численного определения связанности. Именно поэтому с помощью данной классификации можно лишь примерно определить уровень связанности в объектно-ориентированной системе.

На первый взгляд, можно предложить в качестве численного измерения метод, когда определенному подклассу присваивается число, величина которого зависит от характера связанности. Все же такой метод является довольно грубым.

Классификация не строится на каких-либо базовых определениях и поэтому объяснение некоторых частей классификации можно трактовать поразному (но стоит отметить, что практически к каждому классу и подклассу данной классификации дано достаточное пояснение в первоисточнике).

Не стоит забывать и про то, что классификация в большей части строятся на описании различных ситуаций. Являются ли они универсальными для

всех языков программирования? Существуют ли ситуации, которые не подходят под данную классификацию? Эти вопросы покрытия всех ситуаций остаются открытыми.

Не лишним будет напомнить о том, что подклассы разных классов связанности могут пересекаться, а это создает новые вопросы, касающиеся выбора конкретного класса для измерения связанности.

Субъективно, можно выдвинуть еще один минус данной классификации – ее объем. При таком количестве подклассов очень легко упустить какой-либо подкласс и случайно отнести конкретную ситуацию к неверному подкласс.

Таким образом, данная классификация подходит только для приблизительного оценивания связанности. Но все же нельзя упускать из виду важность данной работы, так как она показывает множество ситуаций, которых стоит избегать или придерживаться, чтобы свести связанность к минимуму.

4.2 Пригодность неточных методов

Хотя был разобран всего один метод, не стоит подробно останавливаться на неточных измерениях, так как все они в итоге дают низкий численный результат.

Конечно, существуют и другие методы различных авторов [4; 5], которые тяжело причислить к точным методам измерения, однако подробно на них останавливаться не стоит.

5 Точные статические методы измерения

Повторимся, что точные методы измерения в рамках данной работы подразумевают выделение базовых понятий, на которых будут строиться метрики связанности, и дальнейший подсчет элементов, удовлетворяющих определению конкретной метрики. Статическими же являются методы, которые представляют уровень связанности одним числом и подразумевают расчет связанности без запуска программы.

Многие авторы прибегают именно к такому подходу.

5.1 Лионель Бриан

В работе данного автора [6] разбирается исследование существующих метрик объектно-ориентированного дизайна, среди которых встречается и связанность. Однако сначала автор вводит некоторые базовые понятия.

5.1.2 Базовые понятия и определения

Методы измерения связанности Бриан определяет на основе базовых определений, данных им в его работе, посвященной валидации метрик объектно-ориентированного дизайна [6, с. 4-6]

Перед началом работы автор определяет понятие модулей и высокоуровневого дизайна.

Согласно словам автора, модули в различной литературе определяются по-разному. В ранней литературе под понятием модуля понимали то же самое, что и под понятием подпрограмма. Однако в литературе, нацеленной на описание и исследование объектно-ориентированного дизайна модуль опи-

сывают как набор типов, данных и определений подпрограмм. В контексте работы Бриан называет модулем второй вариант, а подпрограммами – первый. Автор выводит еще один термин "часть ПО" (software part), однако, в контексте нашей работы оно эквивалентно модулю, и вместо этого термина будет использоваться термин "модуль".

Бриан описывает взаимодействия, которые строятся на двух понятиях: объявление данных (data declaration) и подпрограмма (subroutine).

Объявление данных автор определяет, как типы, переменные и константы. Подпрограмма же означает подпрограмму в привычном понимании.

На основе комбинации этих понятий и строятся 4 типа взаимосвязей:

- Объявление данных – объявление данных;
- Объявление данных – подпрограмма;
- Подпрограмма – подпрограмма;
- Подпрограмма – объявление данных.

Не все из этих взаимосвязей подходят для объектно-ориентированного дизайна. Подходят только первые два типа и их определения даны автором.

Объявление данных – объявление данных (DD) взаимодействие: объявление данных A связано с объявление данных B с помощью DD-взаимодействия, когда изменение или использование A становится причиной изменения или использования B соответственно.

Объявление данных – подпрограмма (DS) взаимодействие: объявление данных A связано с подпрограммой B, когда существует DD-взаимодействие между A и хотя бы одним объявлением данных из сигнатуры B.

Нельзя упустить, что понятие связанности в данной работе в некоторой степени отличается от данного нами. Связанность в контексте работы означает связь конкретной части ПО со всеми остальными частями.

5.1.3 Метрики связанности

Бриан разделяет связанность на два типа [6, с. 17-20]:

Импортная связанность (import coupling): степень, с которой одна часть ПО зависит от всех остальных.

Экспортная связанность (export coupling): степень, с которой остальные части ПО зависят от одной конкретной.

Это неформальные определения, не основанные на базовых понятиях. В представленной работе есть более формальное определение импортной и экспортной связанности.

Дан модуль m. Импортная связанность элемента m (IC(m)) это количество DD-взаимодействий между определениями данных внешними по отношению к m и определениями данных внутри m.

Дан модуль m. Экспортная связанность элемента m (EC(m)) это количество DD-взаимодействий между определениями данных внутри m и определениями данных внешними по отношению к m.

Импортную и экспортную связанность можно разделить на две подкатегории. Это транзитивная (transitive) и прямая (direct) связанность.

Прямая связанность обозначает прямое DD-взаимодействие между двумя модулями.

Транзитивная связанность обозначает транзитивное DD-взаимодействие через другие модули.

5.1.4 Свойства метрик

Бриан выделил несколько свойств, связанных с метриками [6, с. 18].

Неотрицательность (Nonnegativity). Дана программная часть sp. Тогда $IC(sp)$ $(EC(sp)) \geq 0$. $IC(sp)$ $(EC(sp)) = 0$ тогда и только тогда, когда sp не имеет импортных (экспортных) взаимодействий.

Монотонность (Monotonicity). Допустим, что m_1 является модулем, а $II(m_1)(EI(m_1))$ – множеством импортных (экспортных) взаимодействий. Если m_2 является модифицированной версией m_1 с таким же набором объявлений данных и подпрограмм, а также с еще одним импортным (экспортным) взаимодействием, таким что $II(m_2)(EI(m_2))$ включает в себя $II(m_1)(EI(m_1))$, тогда $IC(m_1)$ $(EC(m_1)) \leq IC(m_2)$ $(EC(m_2))$.

Слияние модулей (Merging of Modules). Сумма связанностей двух модулей больше или равна связанности модуля, который содержит все объявления данных слагаемых модулей

5.1.5 Исследование метрик

Бриан выдвигает гипотезы по отношению к импортной и экспортной связанности [6, с. 17-19].

Гипотеза по отношению к импортной связанности. Чем больше объявлений данных входит в модуль, тем больше информации нужно знать, чтобы совместить модуль с остальной частью программы. Другими словами, чем больше количество объявлений данных, тем более неполным является локальное описание интерфейса модуля, тем больше распространяется инфор-

мация, необходимая для интеграции модуля в систему. Значит, программная часть будет больше подвержена ошибкам.

Гипотеза по отношению к экспортной связанности. Чем чаще используется модуль, тем большее количество вычислительных и других сервисов она должна предоставлять, тем более гибкой она должна быть. Значит, программная часть будет больше подвержена ошибкам.

В процессе исследования автор проверил три системы на подверженность ошибкам [6, с. 22-27]. Первая система GOADA обеспечивает наземную поддержку ориентации для спутников, разработанная в Центре Космических Полетов Годдарда NASA. Вторая система GOESIM является динамическим имитатор для геостационарного спутника окружающей среды. Эти системы содержат 525 и 676 блоков Ада и имеют небольшой процент переиспользования (примерно пять процентов строк кода переиспользовано из других систем). Третья система TONS является бортовой системой навигации для спутников, которая имеет 180 блоков Ada и выделяется очень маленьким процентом переиспользования (около двух процентов исходного кода взято из других систем).

Выбор систем с таким малым процентом переиспользования обусловлен тем, что на такой системе выведенные автором метрики должны ярче выделяться, что облегчит анализ.

Во время разработки данных систем велась отчетность по их тестированию. В этой отчетности подробно излагалась информация об ошибках. Именно с помощью этой отчетности автор составлял анализ, а также с помощью специального инструмента для анализа существующего кода.

После составления логистической регрессии становится ясно, что экспортные метрики (экспортная транзитивная и прямая связанность) не являются значимыми во всех трех системах. Однако обе импортные связанности являются значимыми для систем GOADA и GOESIM.

Таким образом, и прямая импортная связанность, и транзитивная импортная связанность прошли первичную валидацию.

5.1.6 Вывод по работе

Подводя итог по работе можно сказать, что ей недостает формализма. Многие определения даны естественным языком, что априори исключает точность. Нет прочных базовых понятий, с помощью которых можно было бы четко определить связанность.

Однако Бриан является одним из первых, кто выдвинул свое определение, основанное на базовых понятиях.

Плюсом работы является то, что автор выдвигает гипотезы относительно своих метрик и пытается проверить их эмпирически.

5.2 Шьям Чидамбер и Крис Кемерер

Еще одним важным этапом в измерении связанности была работа "A Metrics Suite for Object Oriented Design", написанная Чидамбером и Кемерером [7]. Данные авторы критиковали существующие метрики связанности и подходы к их вычислению за недостаток формализма и математической четкости, а также предлагали свое описание.

5.2.1 Базовые понятия и определения

Руководствуясь книгой Робертса [7], авторы определяют объектно-ориентированный дизайн как реляционную систему, которая представляет

кортеж множества элементов, множества отношений и множества бинарных операций [8, с. 477-478]. Если говорить более конкретно в рамках объектно-ориентированного дизайна, то объектно-ориентированный дизайн – это реляционная система, включающая элементы-объекты (классы и объекты), эмпирические соотношения, относящиеся к сложности конкретного элемента, и бинарные операции, которые могут быть выполнены с элементами-объектами. Формально это выглядит так:

$$D \equiv (A, R_1 \cdots R_n, O_1 \cdots O_m) \quad (1)$$

Где A – это объекты-элементы;

R – Эмпирические соотношения (больше чем, меньше чем и т.д.);

O – Бинарные операции.

Автор приводит некоторое описание, которое способствует пониманию эмпирических соотношений. Если быть конкретным, то приводится описание, как именно сравнивать такие объекты-элементы. У дизайнеров объектно-ориентированных систем это понимание скорее интуитивное. К примеру, один класс, сложнее другого, если, при прочих равных условиях, у него больше методов.

Однако такое описание не пригодно, так как не представляет точных результатов. Но точка зрения может быть отправной точкой для изучения метрик. Таким образом, точка зрения является транзитивным бинарным отношением полного порядка между двумя возможными элементами-объектами.

Чтобы измерять что-либо с помощью метрик объектно-ориентированного дизайна, нужно перейти от эмпирической реляционной

системы с точкой зрения в качестве эмпирического отношения, описанной выше, к формальной реляционной системе [8, с. 478].

Допустим, что у нас есть формальная реляционная система F, которая представляет собой отношение, описанное ниже:

$$D \equiv (C, S_1 \cdots S_n, B_1 \cdots B_m) \quad (2)$$

Где C – это множество элементов (реальных чисел);

$S_1 \cdots S_n$ – формальные отношения элементов C (т.е. >,<,=);

$B_1 \cdots B_m$ – бинарные отношения элементов C (т.е. +, -, $*$).

Заменив объекты-элементы на реальные числа, мы переопределили точку зрения, как эмпирическое отношение, на более строгое отношение. Теперь же остается лишь выразить объект-элемент через число.

Базовая вещь, требующая определения – это понятие элемента в эмпирической реляционной системе. Чидамбер и Кемерер опираются на работы, которые адаптируют онтологию автора Банджи из книги “Treatise on Basic Philosophy” под область объектно-ориентированного дизайна [8, с. 478-479]. Согласно данной онтологии наш мир состоит из вещей, называемых реальными личностями (substantial individuals).

Основное, что здесь нужно понимать, это наличие у реальной личности конечного количества определенных свойств. Однако данные свойства могут быть явлены нам только через атрибуты. Реальная личность в совокупности с её свойствами образует объект.

Исследователи, занимающиеся исследованием данной онтологии, дали более формальное определение объекту [8, с. 479]:

$$X = \langle x, p(x) \rangle \quad (3)$$

Где x – реальная личность;

$p(x)$ – конечная коллекция его свойств.

Основываясь на этих базовых понятиях, автор выводит метрики объектно-ориентированного дизайна, включая связанность [8, с. 479]:

Пусть даны объекты $X = \langle x, p(x) \rangle$ и $Y = \langle y, p(y) \rangle$

$$p(x) = \{M_X\} \cup \{I_X\} \quad (4)$$

$$p(y) = \{M_Y\} \cup \{I_Y\} \quad (5)$$

Где $\{M_i\}$ – множество методов объекта i;

$\{I_i\}$ – множество переменных объекта i.

Тогда любое воздействие $\{M_X\}$ на $\{M_Y\}$ или $\{I_Y\}$ представляет собой связанность, так же как и $\{M_Y\}$ на $\{M_X\}$ или $\{I_X\}$. Когда M_X вызывает M_Y в своем коде, M_X изменяет историю использования M_Y.

5.2.2 Оценка метрики

Некоторые исследователи рекомендуют свойства, которыми должна обладать метрика, чтобы быть полезной. Один из таких исследователей является Вейкер. Данный исследователь предложил свой список таких свойств [9]. Позднее Чемавский и Смит подстроили этот список под объектно-ориентированный дизайн, убрав неактуальные свойства [10]. В итоге полу-

чился список из шести свойств, которыми и руководствовались Чидамбер и Кемерер при оценке своих метрик, в список которых входила и связанность.

Полный список можно найти в работе Чемавского и Смита [10], здесь же приведем только те свойства, которыми обладает связанность.

Для удобства будем использовать следующее обозначение: Если P является классом, то под $\mu(P)$ будем понимать связанность класса P. Тогда связанность обладает следующими свойствами:

Несогласованность (Noncoarseness). Для любого класса P найдется класс Q такой, что $\mu(P) \neq \mu(Q)$.

Неединственность (Nonuniqueness). Существуют различные классы P и Q такие, что $\mu(P) = \mu(Q)$.

Важность деталей дизайна (Design Details are Important). Если даны классы P и Q с одинаковой функциональностью, это не означает, что $\mu(P) = \mu(Q)$.

Монотонность. Для всех классов P и Q должны выполняться условия: $\mu(P) \leq \mu(P + Q)$ и $\mu(Q) \leq \mu(P + Q)$, где операция сложения обозначает слияние двух классов (результатом является класс, содержащий все поля и методы слагаемых классов).

Неэквивалентность взаимодействия (Nonequivalence of Interaction). Существуют классы P, Q, R такие что $\mu(P) = \mu(Q)$, но из этого не следует, что $\mu(P + R) = \mu(Q + R)$.

5.2.3 Вывод по работе

В отличие от предыдущей работы, данная работа обладает большим формализмом. Отличительной чертой является описание объекта, опирающе-

еся на философские труды. Однако, данный формализм все равно далек от привычного в классической математике, что до сих пор остается проблемой.

Минусом является отсутствие ясной валидации, какая была представлена в предыдущей работе.

Все же нельзя недооценивать вклад этой работы в общее развитие связанности, так как он является одной из первых попыток внести ясный формализм в изучение метрик качества объектно-ориентированных систем.

6 Точные динамические методы измерения

С развитием методов измерения связанности стало понятно, что статическое измерение не учитывает такие аспекты объектно-ориентированных систем, как полиморфизм и динамическое связывание.

Самый прямой путь – измерять связанность прямо во время выполнения программы, что и делают исследователи в своих работах. Динамическая связанность, в отличие от статической, является случайной величиной, значение которой зависит от пути выполнения программы.

6.1 Шериф Якуб

Еще одним шагом вперед стало развитие изучения динамического измерения связанности. Якуб был одним из первых, кто предложил метод измерения, основанный на динамическом измерении связанности. Данный класс методов предлагает наиболее точный результат измерения связанности, так как учитывает такие свойства объектно-ориентированных систем, как полиморфизм, динамическое связывание и т. д.

6.1.1 Базовые понятия и определения

Перед тем, как определить базовые понятия, введем обозначения, которыми пользуется автор [11, с. 4]:

- o_i: множество экземпляров класса;
- O: множество объектов, взаимодействующих во время выполнения сценария.

Теперь определим базовые понятия:

Сценарий (x): сценарий x из множества сценариев X это последовательность взаимодействий между объектами, вызванная входными данными или событиями.

Вероятность сценария (PS_x): частота выполнения одного сценария по отношению ко всем остальным.

Конфигурация сценариев: множество вероятностей, каждый элемент которого является вероятностью исполнения сценария, т.е. PS_x.

Множество сообщений ($M_x(o_i, o_j)$): Множество сообщений из объекта o_i в объект o_j в рамках сценария x. Здесь сообщение определено как запрос, посылаемый из одного объекта другому для предоставления сервиса. Любое взаимодействие между двумя объектами является сообщением.

Общее число сообщений в сценарии (MT_x): общее число сообщений, переданных между объектами, в рамках сценария x.

В своей работе Якуб определил два типа связанности: импортная и экспортная связанность. Оба типа описаны по определенному правилу.

Сначала идет описание контекста, в котором может употребляться конкретная метрика.

Далее идет текстовое, неформальное описание метрики, объясняющее её смысл.

Следом идет аналитическое описание в виде формулы, использующее обозначения, описанные в начале пункта, и элементы теории множеств.

И все это завершается описанием влияния данной метрики на атрибуты качества ПО: поддерживаемость, понятность, переиспользование, сопротивление ошибкам и распространение ошибок.

Далее Якуб описывает экспортную [11, с. 4-5] и импортную [11, с. 5]. связанность на основе этих базовых определений

6.1.2 Экспортная связанность объектов

Контекст. Данный тип связанности проявляется во время создания нескольких объектов и их совместном функционировании в рамках одного сценария.

Текстовое описание. $EOC_x(o_i, o_j)$: экспортная связанность для объекта o_i по отношению к объекту o_j это процент количества сообщений переданных от o_i к o_j по отношению ко всем сообщениям переданным в течении выполнения сценария x.

$$EOC_x(o_i, o_j) = \frac{|\{M_x(o_i, o_j) | o_i, o_j \in O \wedge o_i \neq o_j|}{MT_x} \times 100 \quad (6)$$

Влияние метрики. $EOC_x(o_i, o_j)$ Является мерой взаимной связанности между двумя объектами и играет значимую роль. Она может показать источники возможных ошибок. EOC_x Оказывает влияние на следующие атрибуты:

Поддерживаемость. Объект класса с высокой экспортной связанностью с другим конкретным объектом должен быть более критичным к изменениям и с большей вероятностью передаст изменения в этот конкретный объект.

Понятность. Объект, отправляющий много сообщений к другим объектам тяжелее понять, потому что его динамическое поведение крепко связано с другим объектом.

Переиспользуемость. Как и в предыдущем случае, данный объект сложно переиспользовать, так как он крепко связан с другим конкретным объектом.

Распространение ошибок. Объект, отправляющий много сообщений другим классом, является потенциальным поставщиком большого количества ошибок.

Замечание. Экспортную связанность можно расширить до измерения процента общего числа сообщений, отправленных объектом o_i ко всем остальным объектам в сценарии. Эту метрика имеет название Object Request for Service (OQFS) и может быть получена следующим образом:

$$OQFS_x(o_i) = \frac{\left|\left\{\bigcup_{j=1}^{K} M_x(o_i, o_j) \middle| o_i, o_j \in O \wedge o_i \neq o_j\right|\right.}{MT_x} \times 100 = \\ = \sum_{j=1}^{K} EOC_x(o_i, o_j) \quad (7)$$

6.1.3 Импортная связанность объектов

Контекст. Данный тип связанности проявляется во время создания нескольких объектов и их совместном функционировании в рамках одного сценария.

Текстовое описание. $IOC_x(o_i, o_j)$: Импортная связанность для объекта o_i по отношению к объекту o_j это процент количества сообщений запрошенных от o_i и полученных от o_j по отношению ко всем сообщениям переданным в течении выполнения сценария x.

$$IOC_x(o_i, o_j) = \frac{|\{M_x(o_j, o_i) | o_i, o_j \in O \wedge o_i \neq o_j|}{MT_x} \times 100 \quad (8)$$

Влияние метрики. Как и экспортная связанность, так и импортная связанность является мерой взаимной связанности между двумя объектами. Однако они отличаются направлением связи, а значит, определением, какой объект на какой влияет.

Атрибуты, на которые влияет импортная связанность, аналогичны тем, что описаны для экспортной связанности, однако атрибут “Распространение ошибок” нужно изменить на “Подверженность ошибкам”.

Объект, имеющий большое число поступающих в него сообщений, больше подвержен ошибкам, так как существует большая вероятность получения ошибки от какого-либо сервиса.

Замечание. Импортную связанность так же можно измерить между конкретным объектом и всеми остальными объектами системы. Такая связанность имеет название Object Response for Service (OPFS) и вычисляется следующим образом:

$$OPFS_x(o_i) = \frac{\left|\left\{\bigcup_{j=1}^{K} M_x(o_j, o_i)\right| o_i, o_j \in O \wedge o_i \neq o_j\right|}{MT_x} \times 100 =$$

$$= \sum_{j=1}^{K} IOC_x(o_i, o_j) \quad (9)$$

6.1.4 Связанность в рамках сценария

Метрики, описанные ранее, определены для конкретного исполнения сценария [11, с. 5-6]. Существует возможность расширить масштаб этих метрик за счет конфигурации сценария.

$$IOC(o_i, o_j) = \sum_{x=1}^{|X|} PS_x \times IOC_x(o_i, o_j) \quad (10)$$

$$OPFS(o_i) = \sum_{x=1}^{|X|} PS_x \times OPFS_x(o_i) \quad (11)$$

$$EOC(o_i, o_j) = \sum_{x=1}^{|X|} PS_x \times EOC_x(o_i, o_j) \quad (12)$$

$$OQFS(o_i) = \sum_{x=1}^{|X|} PS_x \times OQFS_x(o_i) \quad (13)$$

Где X– множество всех сценариев.

6.1.5 Исследование метрики

Автор делает важную работу, которая наглядно показывает сравнение динамических и статических мер связанности [11, с. 8-10]. При этом не рассматривается очень большая система, всё исследование проводится на небольшой системе, написанной для кардиостимулятора. В этой системе берется только два сценария и, исходя из этих сценариев, высчитывается $OPFS(o_i)$ и $OQFS(o_i)$.

В качестве статических мер связанности берутся различные метрики, одна из которых принадлежит Чидамберу и Кемереру, а именно, Связанность между объектами (CBO). Чтобы наглядно рассмотреть сравнение CBO с $OPFS(o_i)$ и $OQFS(o_i)$, берется относительная величина CBO, а именно отношение величины CBO одного модуля к сумме величин CBO всех модулей сценария.

Хотя вычисление динамических мер связанности и предполагает измерение в результате выполнения программы, Якуб предлагает измерять связанность с помощью динамических моделей выполнения программы, таких как ROOM (Real-Time Object Oriented Modeling). Метрики связанности в исследовании Якуба вычислены именно с помощью ROOM.

В результате были получены результаты, которые сильно разнятся между собой. На основе полученных данных автор заключает, что статические и динамические метрики в своей сущности являются разными.

При этом результаты можно назвать очевидными и интуитивно понятными. Коротко их разберем.

Статические метрики показывали большие значения для тех модулей, которые предоставляли наибольшее количество различных сервисов, когда динамические меры показывали большие значения для наиболее активных классов, которые посылали наибольшее количество сообщений в систему.

6.1.6 Вывод по работе

Данная работа представляет собой некоторую ценность. Во-первых, это одна из первых работ посвященных динамическим мерам связанности, в которой описаны некоторые подходы для её измерения с учетом того, что данная метрика является случайной величиной.

Во-вторых, данная работа на основе некоторого исследования четко показывает, что метрики статической и динамической связанности являются различными по своей смысловой нагрузке. Динамические меры отвечают за активность объектов некоторого класса, когда статические меры отвечают за сложность объектно-ориентированного дизайна. Однако нужно учитывать, что это относится только к тем мерам, которые определены в работе. Нельзя с полной вероятностью сказать, что другие динамические меры не могут оценивать сложность объектно-ориентированного дизайна.

В-третьих, данная работа является одной из немногих, в которой меры динамической связи измеряются путем составления динамических моделей выполнения объектно-ориентированных систем, что избавляет нас от необходимости иметь конечный продукт.

В противоположность плюсам имеются и минусы. Хотя в данной работе и представлен наглядный пример измерения связанности на простой системе, эти результаты должны подвергнуться валидации, так как показатели одной системы не являются достоверными, хотя и обладают интуитивной понятностью.

6.2 Эрик Арисхольм и Адан Фоен

Другим методом измерения связанности является метод, основы которого заложили и создали Эрик Арисхольм и Адан Фоен. Данный подход похож на предыдущий, однако он в некоторой степени более формализован. Подход тоже акцентируется именно на динамической связанности, а не на статической.

Сначала авторы описывают классификацию [12, с. 492-493], после чего приводит неформальное и формальное определение [12, с. 493-496], иллюстрирующее фундаментальные свойства. После этого, нам даются некоторые математические свойства [12, с. 498-499] и исследование метрик [12, с. 500-503].

6.2.1 Классификация связанности

Существует три критерия, по которым можно определить и классифицировать связанность: измеряемая сущность, детализация и масштаб.

Измеряемая сущность. Измеряемой сущностью может быть не только класс, но и объект.

Детализация. Под уровнем детализации понимается уровень агрегации. Если рассматривать в качестве измеряемой сущности объект, то детализация представлена следующими уровнями: уровень объекта, уровень класса, уровень множества сценариев, уровень множества вариантов использования и уровень системы. Однако если мы берем в качестве измеряемой сущности класс, то уровни будут следующими: уровень класса, уровень иерархии наследования, уровень подсистемы

Масштаб. Данный критерий отвечает за объем измерений, который должен проводиться. Нужно ли включать в измерения различные программные библиотеки, конкретные варианты использования или фреймворки? За это и отвечает масштаб.

В таблице ниже представлена вся классификация.

Сущность измерения	**Детализация**	**Масштаб**
Объект	Объект Класс Множество сценариев Множество вариантов использования Система	Объекты библиотек Объекты фреймворков Исключительные варианты использования
Класс	Класс Иерархия классов Множество подсистем Система	Классы библиотеки Классы фреймворка

6.2.2 Базовые понятия и определения

Начнем с описания множеств, с которыми придется работать.

C: Множество классов системы. Данное множество может быть разбито на несколько подмножеств: классы приложения (AC), классы библиотек (LC), классы фреймворков (FC). Причем выполняются следующие равенства:

$$C = AC \cup LC \cup FC \tag{14}$$

$$AC \cap LC \cap FC = \emptyset \tag{15}$$

O: Множество объектов класса, созданных в течении выполнения всех сценариев и вариантов использования.

M: Множество методов в системе.

N: Строки кода определены множеством натуральных чисел.

Теперь можно определить связи, основанные на данных множествах.

D и A связь на множестве классов $C \times C$, где D – множество классов-потомков, а A – множество классов-предков.

$ME \subseteq O \times M \times N \times O \times M$: множество всех возможных сообщений в системе.

$IV \subseteq M \times C \times M \times C$: Множество всех возможных вызовов методов в системе.

Остальные бинарные отношения обозначаются как R_{Domain}, где $Domain$ описывает некоторое отношение. Например, $R_{MC} \subseteq M \times C$.

Ключевыми отношениями являются IV и ME. На практике, динамический анализ кода позволяет нам вывести только ME. Однако, существует возможность вывести IV, однако эта задача усложняется с учетом полиморфизма и динамического связывания. Сделать это можно с помощью правила последовательности:

$$
\begin{aligned}
&(\exists (o_1, c_1), (o_2, c_2) \in R_{OC})(\exists l \in \mathrm{N})(o_1, m_1, l, o_2, m_2) \in ME \Rightarrow \\
&\Rightarrow (\exists c_3 \in A(c_1) \cup \{c_1\}, c_4 \in A(c_2) \cup \{c_2\})((m_1, c_3) \in R_{MC} \wedge \\
&\wedge ((\forall c_5 \in A(c_1) - \{c_3\})(m_1, c_5) \in R_{MC} \Rightarrow\ c_5 \in A(c_3)))\ \wedge \\
&\wedge\ ((m_2, c_4) \in R_{MC} \wedge ((\forall c_6 \in A(c_2) - \{c_4\})(m_2, c_6) \in R_{MC} \Rightarrow \\
&\Rightarrow\ c_6 \in A(c_4)))\ \wedge (m_1, c_3, m_2, c_4) \in\ IV
\end{aligned}
\qquad (16)
$$

6.2.3 Метрики связанности

Все метрики определяются как мощности определенных множеств. Существует двенадцать различных множеств, которые определяются путем комбинирования по нескольким критериям. Первым критерием является измеряемая сущность. Измеряемой сущностью может быть объект или класс. Вторым критерием является направление связанности. Связанность может быть импортной или экспортной. И третьим критерием является сила связи.

Рассмотрим критерий силы связи на примере, когда измеряемая сущность является объектом, а уровень детализации – класс.

Динамические сообщения. Данный тип силы связи отвечает за количество различных сообщений отправленных (принятых) одним объектом другому (от другого). Два сообщения могут быть рассмотрены как одинаковые, если их исходные и целевые классы, метод, вызванный из целевого класса, и инструкция, из которой он вызван в исходном классе, одинаковые.

Различные вызовы методов. Количество различных методов, вызванных каждым методом в каждом объекте.

Различные классы. Количество различных серверных (клиентских) классов, которые используют методы в данном объекте.

Теперь приступим к формальному определению метрик, описанных выше. Но сначала нужно описать обозначения метрик. Метрики строятся по следующему шаблону: "XC_XX", где каждое значение X заменяется одним из приведенных выше критериев. Буква C в шаблоне означает связанность (coupling). Первая буква X заменяется на I или E для импортной (import) и экспортной (export) связанности соответственно. Вторая X заменяется на C или O для случаев, когда сущностью является класс (class) и объект (object) соответственно. И наконец, последняя буква X может заменяться на D, M и C для сил связи уровня динамических сообщений (dynamic messages), различных

методов (distinct methods) и различных классов (distinct classes) соответственно. Формальные определения даны в таблице 1.

Таблица 1 – Формальное определение метрик связанности

Направление	Измеряемая сущность	Сила связи	Формальное определение
Импортная связанность	Объект	Динамические сообщения	$IC\_OD(c_1) = \{(m_1, c_1, l, m_2, c_2) \mid (\forall (o_1, c_1) \in R_{OC}) \; c_1 \neq c_2 \wedge (o_1, m_1, l, o_2, m_2) \in ME\}$
		Различные методы	$IC\_OM(c_1) = \{(m_1, c_1, m_2, c_2) \mid (\forall (o_1, c_1) \in R_{OC}) \; (\exists (o_2, c_2) \in R_{OC}, I \in N) \; c_1 \neq c_2 \wedge \wedge (o_1, m_1, l, o_2, m_2) \in ME\}$
		Различные классы	$IC\_OC(c_1) = \{(m_1, c_1, c_2) \mid (\forall (o_1, c_1) \in R_{OC}) \; (\exists (o_2, c_2) \in R_{OC}, I \in N) \; c_1 \neq c_2 \wedge \wedge (o_1, m_1, l, o_2, m_2) \in ME\}$
	Класс	Динамические сообщения	$IC\_CD(c_1) = \{(m_1, c_1, l, m_2, c_2) \mid (\exists (o_3, c_3), (o_4, c_4) \in R_{OC}) (\exists l \in N) \; c_1 \neq c_2 \wedge \wedge (o_3, m_1, l, o_4, m_2) \in ME \wedge \wedge (\exists c_1 \in A(c_3) \cup \{c_3\}, c_2 \in A(c_4) \cup \{c_4\}) ((m_1, c_1) \in R_{MC} \wedge ((\forall c_5 \in A(c_1) - \{c_1\}) (m_1, c_5) \in R_{MC} \Rightarrow c_5 \in A(c_1))) \wedge ((m_2, c_2) \in R_{MC}) \wedge \wedge ((\forall c_6 \in A(c_4) - \{c_2\}) (m_2, c_6) \in R_{MC} \Rightarrow \Rightarrow c_6 \in A(c_2))) \wedge (m_1, c_1, m_2, c_2) \in IV\}$
		Различные методы	$IC\_CM(c_1) = \{(m_1, c_1, m_2, c_2) \mid (\exists (m_1, c_1), (m_2, c_2) \in \in R_{MC}) \; c_1 \neq c_2 \wedge (m_1, c_1, m_2, c_2) \in IV\}$
		Различные классы	$IC\_CC(c_1) = \{(m_1, c_1, c_2) \mid (\exists (m_1, c_1), (m_2, c_2) \in R_{MC}) \; c_1 \neq c_2 \wedge (m_1, c_1, m_2, c_2) \in IV\}$
Экспортная связанность	Объект	Динамические сообщения	$EC\_OD(c_1) = \{(m_2, c_2, l, m_1, c_1) \mid (\forall (o_1, c_1) \in R_{OC}) \; (\exists (o_2, c_2) \in R_{OC}, I \in N) \; c_1 \neq c_2 \wedge \wedge (o_2, m_2, l, o_1, m_1) \in ME\}$
		Различные методы	$EC\_OM(c_1) = \{(m_2, c_2, m_1, c_1) \mid (\forall (o_1, c_1) \in R_{OC}) \; (\exists (o_2, c_2) \in R_{OC}, I \in N) \; c_1 \neq c_2 \wedge \wedge (o_2, m_2, l, o_1, m_1) \in ME\}$
		Различные классы	$EC\_OC(c_1) = \{(m_2, c_2, c_1) \mid (\forall (o_1, c_1) \in R_{OC}) \; (\exists (o_2, c_2) \in R_{OC}, I \in N) \; c_1 \neq c_2 \wedge \wedge (o_2, m_2, l, o_1, m_1) \in ME\}$
	Класс	Динамические сообщения	$EC\_CD(c_1) = \{(m_2, c_2, l, m, c) \mid (\exists (o_3, c_3), (o_4, c_4) \in R_{OC}) (\exists l \in N) \; c_1 \neq c_2 \wedge \wedge (o_4, m_2, l, o_3, m_1) \in ME \wedge \wedge (\exists c_1 \in A(c_3) \cup \{c_3\}, c_2 \in A(c_4) \cup \{c_4\}) ((m_1, c_1) \in R_{MC} \wedge ((\forall c_5 \in A(c_3) - \{c\}) (m_1, c_5) \in R_{MC} \Rightarrow c_5 \in A(c_1))) \wedge \wedge ((m_2, c_2) \in R_{MC}) \wedge ((\forall c_6 \in A(c_4) - \{c_2\}) (m_2, c_6) \in R_{MC} \Rightarrow c_6 \in A(c_2))) \wedge \wedge (m_2, c_2, m_1, c_1) \in IV\}$
		Различные методы	$EC\_CM(c_1) = \{(m_2, c_2, m_1, c_1) \mid (\exists (m_1, c_1), (m_2, c_2) \in R_{MC}) \; c_1 \neq c_2 \wedge \wedge (m_2, c_2, m_1, c_1) \in IV\}$
		Различные классы	$EC\_CC(c_1) = \{(m_2, c_2, c_1) \mid (\exists (m_1, c_1), (m_2, c_2) \in R_{MC}) c_1 \neq c_2 \wedge \wedge (m_2, c_2, m_1, c_1) \in IV\}$

6.2.4 Свойства связанности

К сожалению, автор выборочно подходит к формальному описанию свойств связанности, поэтому я возьму на себя ответственность и сам попробую описать данные свойства

Неотрицательность. Данное свойство исходит из того, что каждая метрика является мощностью конкретного множества.

$$|XC\_XX| \geq 0 \tag{17}$$

Свойство пустых множеств. На уровне системы, если принять за S множество, которое включает все объекты всех вариантов использования, то действует следующее свойство:

$$ME = \emptyset \Leftrightarrow XC\_XX(S) = \emptyset \tag{18}$$

Монотонность. Если к одному классу добавить новые методы, то уровень его связанности может только расти или оставаться неизменным.

$$\begin{aligned}&(\exists\, c_1)(\forall c_2)(|\{(c_1, m) \mid (c_1, m) \in R_{CM}\}| < \\ &< |\{(c_2, m) \mid (c_2, m) \in R_{CM}\}|) \Rightarrow |XC\_XX(c_1)| \leq |XC\_XX(c_2)|\end{aligned} \tag{19}$$

Слияние классов. При слиянии классов, когда из двух классов создается трети путем объединения в нем методов из первых двух. Связанность получившегося класса будет больше либо равной сумме связанностей двух исходных классов.

$$(\forall c_1, c_2, m_1, m_2)(\exists c_3, m_3)\ (m_3 = \{m_1 |\ (c_1, m_1) \in R_{MC}\} \cup$$
$$\cup \{m_2 |\ (c_2, m_2) \in R_{MC}\}) \wedge ((c_3, m_3) \in R_{MC}) \Rightarrow \quad (21)$$
$$\Rightarrow |XC\_XX(c_3)| \geq |XC\_XX(c_1)| + |XC\_XX(c_2)|$$

Симметричность импортной и экспортной связанностей. Объединение всех импортных связанностей равно объединению всех экспортных связанностей.

$$(\forall c \in C)\left(\bigcup IC\_CX(c) = \bigcup EC\_CX(c)\right) \quad (22)$$

$$(\forall o \in O)\left(\bigcup IC\_OX(o) = \bigcup EC\_OX(o)\right) \quad (23)$$

6.2.5 Исследование метрики

Главной целью исследования было определение того, есть ли корреляция между метриками динамической связанности и подверженностью системы изменениям, т.е. степенью изменения системы в различных версиях.

Во время исследования был выбран довольно известный проект с открытым исходным кодом, называющийся Velocity. Данная система является частью Apache Jakarta Project и предназначен для создания веб-страниц, SQL-запросов, PostScript и других документов на основе шаблонов. Система содержит 17 последовательных версий (от версии 1.0b1 до версии 1.3.1). В ана-

лизе использовались четыре версии, которые представляют собой четыре субрелиза одного из релизов.

Были собраны несколько типов данных. Во-первых, были собраны данные о количестве добавленных и удаленных строк внутри каждого класса по сравнению с предыдущей версией, а во-вторых, были рассчитаны все двенадцать метрик динамической связанности, представленные авторами, для каждого класса каждой версии.

Статистическая гипотеза о влиянии метрик динамической связанности на подверженность системы изменениям проверялась с помощью метода наименьших квадратов.

Кроме проверки гипотезы относительно динамических мер связанности, гипотеза проверялась и относительно статических мер связанности.

В результате анализа, значимыми оказались следующие метрики динамической связанности: EC_OC, EC_OM, EC_OD, IC_CC.

Как оказалось, статические меры связанности тоже влияют на подверженность изменениям. Полный их список показан в работе, среди таких значимых мер имеется уже разобранная метрика CBO.

6.2.6 Вывод по работе

Подводя итоги данной работы можно сказать, что данный подход имеет свои плюсы.

Во-первых, достаточный формализм данного метода является отличительной чертой от всех предыдущих работ. Оперируя теорией множеств, авторы четко определили свои метрики. Однако, некоторые базовые понятия, такие как методы, классы и объекты все же требуют дополнительных определений и уточнений.

Во-вторых, очень четко показан переход от одной метрики к другой, что позволяет сделать описанный ранее формализм теории множеств.

И наконец, существует проверка некоторой статистической гипотезы о влиянии статических и динамических метрик связанности на подверженность системы изменениям, которая, однако, требует дополнительных исследований, так как результаты одной системы нельзя распространить на остальные.

Больших минусов замечено не было.

ЗАКЛЮЧЕНИЕ

На основе проведенной работы можно сделать несколько выводов.

Для начала стоит отметить, что существует большое множество различных работ и много авторов пытается определить свои новые подходы и метрики связанности, которые идут в разрез с остальными. Можно предположить, что такая ситуация происходит из-за того, что нет четкого понятия, а тем более, определения связанности. В результате такого явления каждый понимает данный термин по-своему.

Много авторов вводят свои метрики связанности, но не все метрики основываются на четких базовых понятиях, что приводит к различному пониманию их определений.

Как итог, первая проблема в изучении связанности – это отсутствие формализма и математической строгости в базовых определениях.

Разрозненность в работе различных исследователей приводит также ко второй проблеме изучения связанности – малое число исследований метрик. Когда каждый исследователь предлагает свои базовые определения, каждый пытается угнаться за лучшей теорией, когда исследование самих метрик остается на втором плане. Много метрик не подвергаются должному исследованию, а значит, не открываются свойства систем, связанные с данными метриками.

В качестве общего вывода по состоянию имеющихся знаний на тему связанности можно сказать, что сейчас изучение связанности продолжает находиться в зачаточном состоянии. Многие работы были написаны более десяти лет назад, а новые работы являются повторением старых и не привносят ничего нового.

Теперь можно ясно определить, какие работы должны быть проведены для успешного освоения данной области.

Первоочередное, что заслуживает внимания, это базовые понятия. На данный момент нужно найти, создать или же усовершенствовать набор четких базовых понятий, с помощью которых можно было бы определить все имеющиеся метрики связанности, описанные выше, а также в других работах. При этом нужно учитывать четкость и математическую строгость данных понятий, а также отдавать предпочтения именно понятиям, лежащим в основе точных измерений связанности.

Вторым этапом нужно описать имеющиеся метрики связанности при помощи базовых понятий, а также выявить новые, если таковые существуют.

На третьем этапе предполагаются обширные измерения существующих систем разного масштаба, рода и сложности для выявления некоторых закономерностей, связанных с метриками. На основе данных закономерностей следует выдвигать и проверять статистические гипотезы.

Субъективно, именно таким должно быть развитие измерения связанности.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. ISO/IEC/IEEE International Standard. Systems and software engineering – Введ. 01.09.2017. – 2017. – 536 с.
2. Stevens, W.P. Structured design / W.P. Stevens, G. J. Myers, L. L. Constantine // IBM SYSTEMS JOURNAL. – 1999. – Т.38, №2. – С. 232-256.
3. Eder, J. Coupling and Cohesion in Object-Oriented Systems / J. Eder, M. Schrefl, K. Gerti – 1995. – 34 с.
4. Offutt, A. J. A Software Metric System for Module Coupling / A. J. Offutt, M. J. Harrold, P. Kolte // J. SYSTEMS SOFTWARE. –1993. №20. – С. 295-308.
5. Hitz, M. Measuring Coupling and Cohesion In Object-Oriented Systems / M. Hitz, B. Montazeri – Вена : Университет Вены, 1995. – 10 с.
6. Briand, L. Defining and Validating High-Level Design Metrics / L. Briand, S. Morasca, V. R. Basili – Мэриленд : Университет Мэриленда, 2005. – 32 с.
7. Roberts, F. Encyclopedia of Mathematics and its Applications / F. Roberts. – Addison-Wesley –1979.
8. Chidamber, S. R. A Metrics Suite for Object Oriented Design / S. R. Chidamber, C. F. Kemerer // IEEE TRANSACTIONS ON SOFTWARE ENGINEERING. – Т.20, №6. – 1994. – С. 476-493.
9. Weyuker, E. Evaluating software complexity measures / E. Weyuker // IEEE TRANSACTIONS ON SOFTWARE ENGINEERING. – Т. 14. – 1998. – С. 1357-1365.

10. Chemiavsky, J. C. Evaluating software complexity measures / J. C. Chemiavsky, C. H. Smith // IEEE TRANSACTIONS ON SOFTWARE ENGINEERING. – T. 17. – 1991. – C. 636-638.

11. Yacoub, S. M. Dynamic Metrics for Object Oriented Designs / S. M. Yacoub, H. H. Ammar, Tom Robinson – Моргантаун : Университет Западной Виргинии, 1999. – 12 с.

12. Arisholm, E. Dynamic Coupling Measurement for Object-Oriented Software / E. Arisholm, L. C. Briand, A. Føyen // IEEE TRANSACTIONS ON SOFTWARE ENGINEERING. – T. 30, №8. – 2004. – C. 491-506.