Федеральное государственное автономное

образовательное учреждение

высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

|  |
| --- |
| Институт космических и информационных технологий |
| *институт* |
| Системы искусственного интеллекта |
| *кафедра* |

ОТЧЕТ О ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ

|  |
| --- |
| Кафедра «Системы искусственного интеллекта» ИКИТ СФУ |
| *Место прохождения практики* |
| Обзор литературы на тему связанности в объектно-ориентированных системах |
| *тема* |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Руководитель от университета | | |  |  |  |  |
| Руководитель от предприятия | | |  |  |  |  |
|  | |  |  | *подпись, дата* |  | *инициалы, фамилия* |
| Студент | КИ17-11Б 031723019 | |  |  |  | В.А. Рудт |
|  | *номер группы, зачетной книжки* | |  | *подпись, дата* |  | *инициалы, фамилия* |

Красноярск 2019

Оглавление

[ВВЕДЕНИЕ 3](#_Toc17530980)

[1. Понятие и определение связанности 5](#_Toc17530981)

[2. Общие положения о связанности и её измерении 6](#_Toc17530982)

[3. Способы измерения связанности 10](#_Toc17530983)

[4. Нечеткие методы измерения 11](#_Toc17530984)

[4.1. Johann Eder 11](#_Toc17530985)

[4.1.1. Interaction coupling 12](#_Toc17530986)

[4.1.2. Component coupling 14](#_Toc17530987)

[4.1.3. Inheritance coupling 15](#_Toc17530988)

[4.1.4. Разбор пригодности для вычисления 17](#_Toc17530989)

[5. Четкие методы измерения 18](#_Toc17530990)

[5.1. Lionel C. Briand 19](#_Toc17530991)

[5.1.1. Базовые понятия 19](#_Toc17530992)

[5.2. Shyam R. Chidamber and Chris F. Kemerer 22](#_Toc17530993)

[5.2.1. Базовые понятия 22](#_Toc17530994)

[5.3. Yacoub S. 24](#_Toc17530995)

[5.3.1. Базовые понятия 25](#_Toc17530996)

[5.3.2. Export Object Coupling 26](#_Toc17530997)

[5.3.3. Import Object Coupling 27](#_Toc17530998)

[5.3.4. Связанность в рамках сценария 29](#_Toc17530999)

[5.4. Erik Arisholm и Audun Føyen 29](#_Toc17531000)

[5.4.1. Классификация связанности 30](#_Toc17531001)

[5.4.2. Базовые определения 31](#_Toc17531002)

[5.4.3. Метрики связанности 32](#_Toc17531003)

[5.4.4. Свойства связанности 35](#_Toc17531004)

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, качество программного обеспечения (далее ПО) является важным аспектом. Однако, как можно определить это качество? При рассмотрении 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 схоже по смыслу, но приравнять их друг к другу нельзя.

Хотя понятия как такового нет, как бы странно не звучало, есть несколько определений. Существует определения, которые были получены в результате совместной работы институтов стандартизации ISO и IEC [1].

**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 мера относится к объектно-ориентированным системам.

1. 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, которые негативно влияют на связанность. 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 хорошую.

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 [2].

Закономерность крайне важна, так как ошибки и изменения, переданные по связи другому модулю, могут перейти и к следующим модулям тоже и не факт, что следующий модуль будет один. 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.

Однако, 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 – 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. Однако программист может случайно или намеренно изменить исходные данные еще на этапе вычислений, что приведет к некорректной работе программы.



Рисунок —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 — 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 выесказанного 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.

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

1. 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 элемент нашей системы, 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 элементов, характеризующих связанность, которые впоследствии подсчитываются.

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

Статические методы измерения связанности характерны тем, что измерение связанности происходит до запуска программы, когда динамические методы связанности измеряют ее во время выполнения программы.

Все типы методов следует рассмотреть.

1. Нечеткие 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.

* 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.

Йохан Эдер вывел такую классификацию в своей работе [3, 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, хотя и 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. Да, данная классификация не позволяет точно измерить связанность, но все же помогает избегать некоторых методов и ситуаций, уменьшая при этом связанность.

Хотя метод можно модифицировать и для численного измерения.

* + 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 сохранение инкапсуляции и сокрытия 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 common coupling, но в структурированном общем пространстве. Вместе с тем к данному подклассу переходят все те же недостатки, присущие common coupling. Разновидность такой связанности, которая осуществляется между классами, связанными наследованием, автор называет inherited external coupling.

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 метода такова: метод зависит 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.

* + 1. 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. Eder также дает более четкое определение компонента.

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

* является частью (является полем)
* входит в состав входных или выходных параметров метода из класса объекта
* является локальной переменной какого-либо метода класса объекта
* входит в состав входных или выходных параметров метода, вызываемого в методе из класса объекта

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 связан с любым подклассом класса . 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 связан с классом данным классом связанности, если не содержится в явном виде класса . Данный класс связанности происходит, например, когда в своем методе в качестве возвращаемого значения вызывает метод объекта класса при этом данные, которые могли использоваться в методе класса , скрываются.

Рассеянная связанность (scattered coupling). Класс объектно-ориентированной системы связан с классом данным классом связанности, если объект класса фигурирует в качестве реализации класса . 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 связан с классом данным классом связанности, если фигурирует в сигнатуре методов класса .

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

* + 1. 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 и классы и , которые являются подклассами класса . Существует метод , принадлежащий классу и не переопределенный в классах и . Тогда, если класс содержит в себе экземпляры классов и и вызывает их метод , значит класс связан с и с помощью связанности компонентов. Однако, автор определяет, что благодаря наследованию класс связан только с классом , так как метод не переопределен в классах-наследниках. Данный факт уменьшает количество связей.

Как и в предыдущих случаях, автор выделяет подтипы inheritance coupling.

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 и суперклассом может называться связанностью модификации сигнатуры тогда и только тогда, когда не только реализация, но и сигнатура наследуемого метода изменяется без каких-либо ограничений или же вовсе удаляется.

Связанность модификации реализации. Связанность между подклассом и суперклассом может называться связанностью модификации реализации тогда и только тогда, когда только реализация наследуемого метода 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 и суперклассом может называться связанностью уточнения сигнатуры тогда и только тогда, когда она не походит под определение modification coupling 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 и суперклассом 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.

* + 1. 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 целю четкое численное определение связанности. Возможно 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 по-разному (но стоит отметить, что практически к каждому классу и подклассу данной классификации дано достаточное пояснение).

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 подтипы разных типов могут пересекаться, а это создает новые вопросы, касающиеся выбора конкретного типа 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.

* 1. Пригодность нечетких методов

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

Конечно, существуют и другие методы различных авторов, которые тяжело причислить к четким методам измерения, однако подробно на них останавливаться не стоит.

1. Четкие статические методы измерения

Повторимся, что четкие методы измерения в рамках данной работы подразумевают подсчет некоторых элементов, которые, по мнения автора, характеризуют уровень связанности между модулями.

Многие авторы прибегают именно к такому подходу.

* 1. Лионель Бриан

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

* + 1. Базовые понятия

Методы измерения связанности Бриан определяет на основе базовых определений, данных им в его работе, посвященной валидации метрик объектно-ориентированного дизайна [6, 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 подпрограмма. Однако в литературе, нацеленной на описание и исследование объектно-ориентированного дизайна модуль описывают 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".

Briand 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.

* + 1. Метрики связанности

Бриан разделяет связанность на два типа [6, 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 связанности. Определение экспортной связанности дано в одной из предыдущих работ автора.

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.

* + 1. Валидация метрик

Бриан выдвигает гипотезы по отношению к импортной и экспортной связанности [6, 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 информация, 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 валидации автор проверил три системы на подверженность ошибкам [6, 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 переиспользованно 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 так же 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.

* + 1. 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, и существуют другие работы, которые сыграли большую роль в определении связанности. Перед их рассмотрением, следует описать ряд других работ.

* 1. 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 и подходы к их вычислению за недостаток формализма и математической четкости, а также предлагали свое описание.

* + 1. Базовые понятия

Руководствуясь книгой Робертса [7], авторы определяют объектно-ориентированный дизайн как реляционную систему, которая представляет кортеж множества элементов, множества отношений и множества бинарных операций [8, 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:

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 это 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, описанной выше, к формальной реляционной системе [8, 478].

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

Где – это множество элементов (реальных чисел). – формальные отношения элементов (т.е. >,<,=). ­– бинарные отношения элементов (т.е. +, -,).

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, 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, 479]:

Где – реальная личность, а – конечная коллекция его свойств.

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

Пусть даны объекты и

Где – множество методов объекта , – множество переменных объекта .

* + 1. 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 получился 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 является классом, то под будем понимать связанность класса . Тогда связанность обладает следующими свойствами:

Несогласованность (Noncoarseness). Для любого класса найдется класс такой, что .

Неединственность (Nonuniqueness). Существуют различные классы и такие, что .

Важность деталей дизайна (Design Details are Important). Если даны классы и с одинаковой функциональностью, это не означает, что .

Монотонность. Для всех классов и должны выполняться условия: и , где операция сложения обозначает слияние двух классов (результатом является класс, содержащий все поля и методы слагаемых классов).

Неэквивалентность взаимодействия (Nonequivalence of Interaction). Существуют классы такие что , но из этого не следует, что .

* + 1. Вывод по работе

В отличие от предыдущей работы, данная работа обладает большим формализмом. Отличительной чертой является описание объекта, опирающееся 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 объектно-ориентированных систем.

* 1. Yacoub S.

Еще одним шагом вперед стало развитие изучения динамического измерения связанности. Yacoub 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.

* + 1. Базовые понятия

Перед тем, как определить базовые понятия, введем обозначения, которыми пользуется автор:

* : множество экземпляров класса.
* : множество объектов, взаимодействующих во время выполнения сценария.

Теперь определим базовые понятия:

Сценарий “”: сценарий из множества сценариев это последовательность взаимодействий между объектами, вызванная входными данными или событиями.

Вероятность сценария “”: частота выполнения одного сценария по отношению ко всем остальным.

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

Множество сообщений : Множество сообщений из объекта в объект в рамках сценария . 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 : общее число сообщений, переданных между объектами, в рамках сценария .

В своей работе Якоб 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 и распространение ошибок.

* + 1. Export Object Coupling

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

Текстовое описание. , экспортная связанность для объекта по отношению к объекту это процент количества сообщений переданных от к по отношению ко всем сообщениям переданным в течении выполнения сценария .

Формула.

Влияние метрики. Является мерой взаимной связанности между двумя объектами и играет значимую роль. Она может показать источники возможных ошибок. 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 связь можно расширить до измерения процента общего числа сообщений, отправленных объектом ко всем остальным объектам в сценарии. Эту метрика имеет название Object Request for Service (OQFS) и может быть получена следующим образом:

* + 1. Import Object Coupling

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

Текстовое описание. , Импортная связанность для объекта по отношению к объекту это процент количества сообщений запрошенных от и полученных от по отношению ко всем сообщениям переданным в течении выполнения сценария .

Формула.

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”.

Так как в объект, имеющий большое число поступающих в него сообщений, он 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:

* + 1. Связанность в рамках сценария

Метрики, описанные ранее, определены для конкретного исполнения сценария. Существует возможность расширить масштаб этих метрик за счет конфигурации сценария.

Где – множество всех сценариев.

//TODO(оценка и вывод)

* 1. Erik Arisholm и Audun Føyen

Другим методом измерения связанности является метод, основы которого заложили и создали Erik Arisholm и Audun Føyen. 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, после чего приводит неформальное и формальное определение, иллюстрирующее фундаментальные свойства. После чего, нам даются некоторые математические свойства.

* + 1. Классификация связанности

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

Измеряемая сущность. Так как измерения проводятся во время исполнения программы, то измеряемая сущность может быть не только классом, но и объектом.

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 |

* + 1. Базовые определения

Начнем с описания множеств, с которыми придется работать.

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

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

: Множество методов в системе.

N: Строки кода определены множеством натуральных чисел.

Теперь можно определить связи, основанные на данных множествах.

и связь на множестве классов , где – множество классов-потомков, а – множество классов-предков.

: множество всех возможных сообщений в системе.

: Множество всех возможных вызовов методов в системе.

Остальные бинарные отношения обозначаются как , где описывает некоторое отношение. Например, .

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

* + 1. 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 выражение, из которого он вызван в исходном классе, одинаковые.

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

Различные классы. Количество различных классов на стороне сервера (клиента), методы которых используют (используются) для данного объекта.

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

Ниже приводится описание всех двенадцати метрик связанности, для которых дано формальное определение.

|  |  |  |  |
| --- | --- | --- | --- |
| Направление | Измеряемая сущность | Сила связи | Формальное определение |
| Импортная связанность | Объект | Динамические сообщения |  |
| Различные методы |  |
| Различные классы |  |
| Класс | Динамические сообщения |  |
| Различные методы |  |
| Различные классы |  |
| Экспортная связанность | Объект | Динамические сообщения |  |
| Различные методы |  |
| Различные классы |  |
| Класс | Динамические сообщения |  |
| Различные методы |  |
| Различные классы |  |

* + 1. Свойства связанности

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

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

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

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.

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