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

TODO(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, качество программного обеспечения (ПО) является важным аспектом. Однако, как можно определить это качество? При рассмотрении качества ПО нужно понять, с какой позиции можно его можно рассмотреть. 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.

В современном мире большую роль играет объектно-ориентированное программирование, так как данная парадигма легко воспринимается человеком.

Еще в начале развития объектно-ориентированных систем было определенно несколько метрик для их исследования. Одной из самых важных метрик является связанность. Данная метрика определяет степень взаимодействия между программными модулями. Многие исследователи считают, что в зависимости от степени связности будет определяться качество ПО. Также, некоторые из исследователей предлагают методы для измерения связанности. Различным методам измерения данной метрики, а также их анализу, посвящена данная работа.

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 до конца не понятен весь спектр данного взаимодействия. Так же неясно, какую степень считать большой независимо от контекста. Последний вопрос является одним из ключевых в данной работе.

Теперь, когда было дано некоторое понятие о связанности, следует предоставить существующие попытки дать определение этому термину. Существует несколько определений, которые были получены в результате совместной работы институтов стандартизации ISO и IEC.

**Coupling**: the strength of the relationships between modules [1].

**Coupling**: manner and degree of interdependence between software modules [1].

**Coupling**: measure of how closely connected two routines or modules are [1].

**Coupling**: a measure of the interdependence among modules in a computer program [1].

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 компонентами 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 книге Structured Design [2], автором которой является W. Stevens. В данной книге связанность трактуется следующим образом:

Coupling is the measure of the strength of association established by a connection from one module to another [2].

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 [2].

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.



Рисунок — Волнообразный эффект изменений и ошибок

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.

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.

Таким образом, 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. В последующем авторы ссылались на труд “Structured design” так как он давал общее представление о связанности.

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 типы были присуще методам, используемым в системах, которые были написаны с использованием процедурной парадигмы. Несмотря на это, данные методы можно распространить и на объектно-ориентированные системы, чем и занимались некоторые авторы. 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.

Другим типом методов измерения являются точные измерения. Такое измерение часто используют в работах, посвященных измерению связанности именно в объектно-ориентированных системах. К данному типу можно отнести методы, суть которых состоит в выделении некоторых базовых элементов, отвечающих за уровень связанности, которые впоследствии подсчитываются.

И тот и другой тип должны быть рассмотрены.

1. Нечеткие методы измерения

Stevens 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. Johann Eder

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.

Johann Eder вывел такую классификацию. Данная классификация описана в книге “Coupling and Cohesion in Object-Oriented Systems” [3].

Автор выделяет три класса связанности:

* Interaction coupling
* Component coupling
* 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 качество 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.

* + 1. Interaction coupling

Начнем с первого класса “interaction coupling”. Данный класс связанности описывает взаимодействие между методами класса путем вызова одного метода другим и/или использования общей памяти.

Автор также отмечает, что объекты класса тоже могут быть связанны данным классом связанности, так как методы, по определению, должны находиться в классах. Еще одним 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.

Content coupling. Данный класс 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.

Сommon 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.

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.

Control coupling. Это подкласс, который характеризует связанность между методами, коммуницирующими только посредством передачи параметров. При этом один метод контролирует внутреннюю логику другого.

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.

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.

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

* + 1. Component coupling

Следующим классом является component coupling. В отличие от interaction coupling, 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 potential component coupling.

Как и в предыдущем классе, component coupling делится на несколько подклассов.

Hidden coupling. 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 метода объекта класса при этом данные, которые могли использоваться в методе класса , скрываются.

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.

Specified coupling. Класс объектно-ориентированной системы связан с классом данным классом связанности, если фигурирует в сигнатуре методов класса .

Nil coupling. Теоретический оптимум, при котором два класса не связаны напрямую.

* + 1. Inheritance coupling

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 component coupling, inheritance coupling 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 component coupling. 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.

Modification coupling. Подкласс связанности, при котором, помимо определения новой информации, она ещё и произвольно изменяется, а иногда даже удаляется. Автор различает signature modification и implementation modification.

Signature modification coupling. Связанность между подклассом и суперклассом может называться "signature modification coupling" тогда и только тогда, когда не только реализация, но и сигнатура наследуемого метода изменяется без каких-либо ограничений или же вовсе удаляется.

Implementation modification coupling. Связанность между подклассом и суперклассом может называться "implementation modification coupling" тогда и только тогда, когда только реализация наследуемого метода изменяется.

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 signature refinement coupling и implementation refinement coupling.

Signature refinement coupling. Связанность между подклассом и суперклассом может называться "signature refinement coupling" 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.

Implementation refinement coupling. Связанность между подклассом и суперклассом может называться " implementation refinement coupling" 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.

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.

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

* + 1. Разбор пригодности для вычисления

Нужно сказать, что Johann Eder внес немалый вклад в исследование связанности, перенеся данный термин с процедурных система 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.

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. Четкие 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. Lionel C. Briand

В работе данного автора под названием “Defining and Validating Measures for Object-Based High-Level Design” разбирается валидация существующих метрик объектно-ориентированного дизайна, среди которых встречается и связанность. Однако сначала автор вводит некоторые базовые понятия.

* + 1. Базовые понятия

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 Briand называет модулем второй вариант, а подпрограммами – первый. Автор выводит еще один термин "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 понтиях: data declaration и subroutine.

Термин data declaration автор определяет, как типы переменные и константы. А subroutine – как подпрограмму.

На основе комбинации этих понятий и строятся 4 типа взаимосвязей:

* Data declaration to data declaration.
* Data declaration to subroutine.
* Subroutine to subroutine.
* Subroutine to data declaration.

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

Data declaration to data declaration (DD) взаимодействие – элемент "data declaration" A связан с элементом "data declaration" B с помощью DD-взаимодействия, когда изменение или использование A становится причиной изменения или использования B соответственно.

Data declaration to subroutine (DS) взаимодействие – элемент "data declaration" A связан с элементом "subroutine" B, когда существует DD-взаимодействие между A и хотя бы одним элементом "data declaration" 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.

Brian разделяет связанность на два типа:

Import coupling – степень, с которой одна часть ПО зависит от всех остальных.

Export coupling – степень, с которой остальные части ПО зависят от одной конкретной.

Это неформальный определения, не основанные на базовых понятиях. В представленной работе есть чуть более формальное определение только термина "import coupling". Определение термина "export coupling" дано в одной из предыдущих работ автора.

Дан модуль m. Import coupling от элемента m (IC(m)) это количество DD-взаимодействий между элементами data declaration внешними по отношению к m и элементами data declaration внутри m.

Дан модуль m. Export coupling от элемента m (EC(m)) это количество DD-взаимодействий между элементами data declaration внутри m и элементами data declaration внешними по отношению к m.

И import coupling и export coupling можно разделить на две подкатегории. Это transitive coupling и direct coupling.

Direct coupling обозначает прямое DD-взаимодействие между двумя модулями.

Transitive coupling обозначает транзитивное DD-взаимодействие через другие модули.

В двух книгах автор выдвигает гипотезы по отношению к терминам "import coupling" и "export coupling".

Гипотеза по отношению к import coupling. Чем больше элементов "data declaration" входит в модуль, тем больше информации нужно знать, чтобы совместить модуль с остальной частью программы. Другими словами, чем больше количество элементов "data declaration", тем более неполным является локальное описание интерфейса модуля, тем больше распространяется информация, необходимая для интеграции модуля в систему.

Гипотеза по отношению к export coupling. Чем чаще используется модуль, тем большее количество вычислительных и других сервисов она должна предоставлять, тем более гибкой она должна быть. Это может привести к ошибкам.

//TODO (заключение и оценка).

* 1. Shyam R. Chidamber and Chris F. Kemerer

Еще одним важным этапом в измерении связанности была работа "A Metrics Suite for Object Oriented Design", написанная авторами Chidamber и Kemerer. Данные авторы критиковали существующие метрики подходы к их вычислению за недостаток формализма и математической четкости и предлагали свое описание.

* + 1. Базовые понятия

Руководствуясь книгой Робертса, авторы определяют объектно-ориентированный дизайн как реляционную систему, которая представляет кортеж множества элементов, множества отношений и множества бинарных операций. 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 никуда не годится, 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.

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

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

Базовая вещь, требующая определения – это понятие элемента в эмпирической реляционной системе. Chidamber и Kemerer опираются на работы, которые адаптируют онтологию автора Bunge из книги “Treatise on Basic Philosophy” под область объектно-ориентированного дизайна. Согласно данной онтологии наш мир состоит из вещей, называемых реальными личностями.

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:

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

Основываясь на этих базовых понятиях, автор выводит метрики объектно-ориентированного дизайна, включая связанность.

Пусть даны объекты и

Где – множество методов, – множество переменных объекта

//TODO (вывод нужен и оценка измерений)

* 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.

Строки кода определены множеством натуральных чисел.

Теперь можно определить связи, основанные на данных множествах.

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