УДК 004.053

МЕТРИКИ ДЛЯ ОЦЕНКИ СЛОЖНОСТИ ОБЪЕКТНО-ОРИЕНТИРОВАННЫХ МОДЕЛЕЙ

Жукова Д.К., Фонотов А.М.
Донецкий национальный технический университет

Учитывая центральную роль, которую играет программное обеспечение в области информационных технологий, менеджеры уделяют все больше внимания процессу улучшения ситуации в области программных разработок. Новые требования стимулировали появление целого ряда новых более совершенных подходов к развитию программного обеспечения. Наиболее важным из них является объектно-ориентированный подход [3]. Кроме того, акцент на улучшение программного обеспечения породил потребность в измерении самого процесса его создания. Каким же образом можно проанализировать объектно-ориентированную модель системы?

Отдельные параметры разрабатываемой системы измеряются метриками. Метрики – это количественные критерии оценки, которые должны использовать понятия объектно-ориентированных языков программирования. Метрики являются набором стандартов, в отношении которых можно оценить эффективность методов объектно-ориентированного анализа в проектирование системы [4], [5].

С точки зрения метрик выделяют пять характеристик объектно-ориентированных систем: локализацию, инкапсуляцию, информационную закрытость, наследование и способы абстрагирования объектов. Эти характеристики оказывают максимальное влияние на объектно-ориентированные метрики [1].

Нами были изучены и проанализированы все существующие метрические наборы для оценки сложности объектно-ориентированной модели. Исходя из полученных результатов, для качественной оценки сложности объектно-ориентированной модели можно выделить следующие пункты, на которые следует обратить внимание:

1. Есть необходимость в определении формальной модели объектно-ориентированной системы. Формальная модель должна определять, какие именно метрики берутся во внимание. Метрики должны быть выражены в терминах формальной модели точно и однозначно.
2. Формальная модель должна охватывать элементы Unified Modeling Language (UML) и пакета диаграммы классов. Разработка моделей с помощью UML – это наиболее широко используемый способ моделирования систем. Все без исключения CASE средства, появившиеся на рынке позже 2005-2007 годов, так или иначе поддерживают UML [6].
3. Формальная модель метрической системы должна также быть основана на UML диаграммах. UML модель обеспечивает работу с учетом принципов объектно-ориентированного проектирования. A XML Metadata Interchange (XMI) – это стандарт обозначения для модели UML, основанный на нем же. С использованием XMI возможна будет автоматизация процесса вычисления метрик [6].
4. Основной характеристикой формальной модели должно быть проектирование высокого уровня. Тогда измерения разработки не будут зависеть от детализированной модели, а могут быть проведены на ранних этапах построения модели [2].

Унифицированный язык моделирования (UML) — это язык визуального моделирования для решения задач общего характера, который используется при определении, визуализации, конструировании и документировании артефактов программной системы. С помощью языка UML можно фиксировать решения, принятые при создании различных систем. Он используется для того, чтобы лучше понимать, проектировать, поддерживать и контролировать эти системы. UML можно использовать со всеми методами разработки, во всех предметных областях и на всех этапах жизненного цикла программы [6].

UML позволяет разработчикам программного обеспечения достигнуть соглашения в графических обозначениях для представления общих понятий (таких как класс, компонент, обобщение (generalization), объединение (aggregation) и поведение) и больше сконцентрироваться на проектировании и архитектуре.

Диаграмма классов UML основана на широко распространенной модели "сущность-связь" (ERD - Entity Relationship Diagram), введенной П.Ченом. Служит для представления статичной структуры модели в терминологии классов ООП. Она представляет собой некоторый граф, вершинами которого являются элементы типа, классификатор, связанные друг с другом различными типами структурных отношений. Отдельные группы диаграммы Классов могут образовывать пакеты для представления более общей модели системы. По мере проработки модели можно дополнять классы атрибутами и операциями.

Обычно для описания системы создают несколько диаграмм Классов. На одних показывают некоторое подмножество классов и отношения между классами подмножества. На других отображают то же подмножество, но вместе с атрибутами и операциями классов. Третьи соответствуют только пакетам классов и отношениям между ними.

Нас интересуют такие диаграммы Классов, на которых есть возможность отследить их параметры и связи между собой.

Различные типы связи на диаграмме Классов говорят о разной степени сцепления между экземплярами классов, их взаимодействии и взаимозависимости. Например, Ассоциация описывает семантические связи между отдельными объектами определенных классов. Различают однонаправленные и двунаправленные Ассоциации. Однонаправленные ассоциации легче создавать, ими легче управлять, они позволяют определить классы, которые допускают повторное использование.

Класс с большим количеством исходящих однонаправленных связей повторно использовать трудно, а класс, все связи которого входящие – легко. Связь ассоциаций может быть рефлексивной, то есть один экземпляр класса может взаимодействовать с другим экземпляром этого же класса.

Связь Обобщения говорит нам о наследовании классов, включая переход операций и атрибутов предков к потомкам, связь Реализация – о зависимости, при которой для правильного функционирования одного элемента требуется присутствие другого элемента и т.д. Таким образом, учет всех типов связей между классами поможет нам максимально объективно оценить сложность модели путем подсчета количества и качества взаимосвязей между ними.

Итак, используя диаграмму Классов, мы сможем оценить параметры системы и проанализировать ее свойства. Мы выделили такие метрики из изученных нами метрических наборов, которые будут доступны для оценки через диаграмму Классов и наиболее полно осветят интересные для нашей задачи параметры системы.

1. NLM (Number of Local Methods) – количество локальных методов. В процессе разработки своего метрического набора, мы пришли к выводу, что имеет смысл разделить эту метрику на две:
   - Общее число локальных методов в классе;

– Число методов в классе за исключением простых методов, таких как get() и set().

Такой подход связан с тем, что простые методы не оказывают влияние на сложность класса, и нам будет полезно в дальнейшем рассчитывать соотношение между общим числом методов в классе и числом сложных методов.

2. NDC (Number of Descendent Classes) – количество потомков для каждого класса. Эта метрика характеризует широту наследования, по которой класс может передать свои методы и атрибуты потомкам.
3. NAC (Number of ancestor Classes) – количество предков для каждого класса показывает глубину наследованных атрибутов и операций в каждом классе. Т.е. свойства скольких классов передались рассматриваемому, а также, при изменении какого количества классов, будет меняться и поведение рассматриваемого класса.
4. CMC (Class Method Complexity) – сложность методов в классе.

К сожалению, рассматривая только диаграмму Классов, мы не сможем определить, насколько сложными являются все методы заданного класса. Однако мы не можем опустить эту метрику, т.к. она является довольно важной и показательной в оценке сложности системы в целом. Поэтому применим искусственное разбиение методов класса на уровни сложности, и назовем эту альтернативную метрику CMC'.

Разбив все методы рассматриваемого класса на три группы, мы умножим количество методов каждой группы на соответствующий их сложности коэффициент и получим сложность методов классе с учетом их различий.

5. CTA (Coupling Through Abstract data type) – количество связей в классах через абстрактный тип данных.
6. MHF (Method Hiding Factor) – фактор сокрытия методов. Вычисляется, как отношение скрытых методов к общему числу методов системы.
7. AHF (Attribute Hiding Factor) – фактор сокрытия атрибутов. Вычисляется, как отношение количества скрытых атрибутов всех классов к общему числу атрибутов всех классов системы.
8. NC (Number of Class) – количество классов в модели.

Для того, чтобы принять решение, является ли объектно-ориентированная модель сложной для реализации или простой, необходимо использовать какую либо систему искусственного интеллекта. И наиболее удачно для этой цели подходит принцип нечеткой логики.

В нечеткой логике вводятся понятия лингвистических переменных, значениями которых являются не числа, а слова естественного языка, называемые термами. Точными физическими показателями, определяющими градуировку термов, станут количественные значения вычисляемых нами метрик.

Т.о. мы фаззифицируем множество значений метрик и переведем их в нечеткий формат. Благодаря такому подходу появится возможность учитывать степень влияния разных значений каждой из метрик на качество системы. Свой вес имеют как метрики для каждого класса отдельно (например, метрики NLM – количество локальных методов в классе, NDC – количество потомков для каждого класса и т.д.), так и для системы в целом (таких как количество классов в модели, факторы сокрытия методов и атрибутов модели).

Построив некоторое количество термов, и определив связанные с ними правила, мы придем к выходному значению нечеткого вида, а именно - в какой мере объектно-ориентированная модель соответствует той, которую следует реализовать.

Литература

[1] Иванов А.Г., Пятницкий А.А, Филинов Ю.Е. Объектно-ориентированный подход технологии программирования – СПб.: Питер, - 2003.

[2] Р.Ральф. Towards a Model for Object-Oriented Design Measurement - Institute of Computer Science, University of Stuttgart, Germany.

[3] Иванов А.Г., Пятницкий А.А, Филинов Ю.Е. Объектно-ориентированный подход технологии программирования – СПб.: Питер, - 2003.

[4] Review of Complexity Metrics for Object Oriented Software Products // International Journal of Computer Science and Network Security, No.11 - November 2008.

[5] Э. Гамма, Р. Хелм, Р. Джонсон, Дж. Влиссидес Приемы объектно-ориентированного проектирования. Паттерны проектирования. Издательство Питер, Санкт-Петербург, 2001.

[6] К. Ларман. Применение UML и шаблонов проектирования – Второе издание, изд.дом «Вильямс» Москава, С. Петербург, Киев, - 2004.