УДК 004.6

ДАТА-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ: ПОЛОЖИТЕЛЬНЫЕ И ОТРИЦАТЕЛЬНЫЕ АСПЕКТЫ

А.А. Файтельсон
Бакалавр второго года обучения по направлению подготовки «Математическое обеспечение и администрирование информационных систем»
Курский государственный университет
e-mail: z0tedd@gmail.com

Научный руководитель:
А.В.Кривоносов
Кандидат технических наук, доцент кафедры программного обеспечения и администрирования информационных систем.
Курский государственный университет

Статья посвящена рассмотрению дата-ориентированного программирования, положительных и отрицательных аспектов этой парадигмы. .Данная парадигма призвана помочь в проектировании информационных систем

Ключевые слова: *дата-ориентированное программирование, парадигмы программирования, проектирование информационных систем*

Введение. «Кризис программного обеспечения» был впервые выявлен в 1968 году [NR69, с.70] и в прошедшие десятилетия он скорее углубился, чем ослаб. Самая большая проблема в разработке и сопровождении крупномасштабных программных систем - сложность, большие системы трудно понять.

В своей классической статье «Серебряной пули нет» Брукс[Bro86] выделил четыре особенности программных систем, которые усложняют создание программного обеспечения: сложность, согласованность, изменчивость и невидимость. Из них считается, что сложность является более значимым признаком; остальные можно классифицировать как формы сложности или рассматриваться как проблематичные исключительно из-за сложности запутанность в системе.

Сложность является основной причиной подавляющего большинства проблем с программным обеспечением. Ненадежность, отсутствие безопасности, зачастую даже плохую производительность в крупномасштабных системах можно рассматривать как результат, в конечном итоге, неуправляемой сложности.

Актуальность сложности широко признана. Как сказал Дейкстра:

«...Мы должны делать системы четкими, распутанными и простотыми, если не хотим быть раздавленными сложностью, которую мы сами создали...»

Первичный статус сложности как основная причина этих других проблем заключается просто в том факте, что способность понять систему является предпосылкой для того, чтобы избежать всех проблем. Для решения проблемы понимания системы можно использовать парадигму дата-ориентированного программирования.

Дата-ориентированное программирование (Data-Oriented Programming, DOP) — подход к разработке информационных систем, в котором данные занимают центральное место.

Основным элементом процедурного программирования являются вызовы процедур, а ООП в основном имеет дело с объектами. В обоих случаях в центр ставится код: в одном случае это обычные процедуры (или функции), в другом — сгруппированный код, связанный с неким внутренним состоянием.

ДОП смещает фокус внимания с объектов на сами данные: тип данных, их расположение в памяти, способы их считывания и обработки в приложения. В отличие от традиционного объектно-ориентированного программирования (ООП), где данные скрыты внутри объектов и смешаны с логикой, в ДОП основное внимание уделяется отделению данных от логики.

Принципы дата-ориентированного программирования:

1. Отделение поведения от данных
2. Представление данных в общем виде
3. Отделение схемы от представления
4. Данные иммутабельны

Принцип №1 - это принцип проектирования, который рекомендует четкое разделение кода(поведение) и данных. Может показаться, что это принцип функционального программирования, но на самом деле придерживаться его или нет его можно как и в функциональном программировании, так и в объектно-ориентированном.

Следование этому принципу в ООП означает агрегирование кода как методов статического класса в ООП языках. Нарушение этого принципа в ФП означает сокрытие состояния программы в лексической области видимости функции.

При соблюдении принципа №1 код отделяется от данных. DOP не зациклен на структурах, которые следует использовать для организации кода, но в нем описано много о том, как должны быть представлены данные. В этом заключается принцип №2 – представление данных в общем виде.

В DOP данные представлены с помощью общих структур данных словарей или массивов вместо создания экземпляров данных с помощью определенных классов. Фактически, большинство сущностей можно пред-

ставить в качестве словарей и массивов, но можно использовать и другие общие структуры данных(деревья, очереди, множества).

Принцип 3 пропагандирует то, что данные должны быть неизменяемыми. DOP очень строг в этом вопросе. Мутация данных не допускается! В DOP изменения данных выполняются путем создания новых версий данных. Ссылка на переменную может быть изменена так, чтобы она ссылалась на новую версию данных, но само значение данных никогда не должно меняться.

С данными, отделенными от кода и представленными с помощью общих и неизменяемых структур данных, теперь возникает вопрос, как выразить форму данных. Принцип 4 отвечает на данный вопрос. В DOP ожидаемая форма выражается как схема данных, которая хранится отдельно от них. Главное преимущество этого принципа заключается в том, что он позволяет разработчикам решать, какие фрагменты данных должны иметь схему, а какие - нет.

Положительные аспекты Дата-ориентированного программирования. Кратко изложенные принципы данной парадигмы помогут понять плюсы от ее использования. Из первого принципа вытекает, то что тщательное разделение кода от данных приносит пользу нашим программам следующим образом:

1. Код можно повторно использовать в разных контекстах.
2. Код можно тестировать изолированно.
3. Системы, как правило, менее сложны.

Использование универсальных структур данных для представления данных имеет множество преимуществ, такие как возможность использования универсальных функций, которые не ограничиваются нашим конкретным вариантом использования. Также данный принцип позволяет строить информационные системы с более гибкой моделью данных

Когда программы ограничены от мутации данных, мы получаем выгоду во многих отношениях, а именно:

1. Доступ к данным без страха изменения
2. Предсказуемое поведение кода
3. Быстрые проверки равенства
4. Безопасность многопоточного доступа

Разделение схемы данных от представления дает свободу выбора данных для проверки, а также позволяет создавать необязательные поля, расширенные условия проверки. Присутствие схемы данных, позволяет переводить ее в готовые диаграммы, что может послужить более простому пониманию всей системы.

Отрицательные аспекты дата-ориентированного программирования. Цена, которую мы платим за получение выгоды от разделения кода и данных тройная:

1. Не существует контроля над тем, какой код к каким данным имеет доступ.
2. Нет упаковки данных и методов работы над ними в один класс.
3. Система состоит из большего количества сущностей.

Когда код и данные смешаны, легко понять, к каким фрагментам кода можно получить доступ. Например, в ООП данные инкапсулируются в объект, который гарантирует, что данные доступны только методами объекта. В ДОП данные стоят сами по себе. Они прозрачные для всех участков кода. Когда происходит изменение формы данных при рефакторинге, нужно будет изменять методы, работающими с ними.

Так как код находится отдельно от данных, то методы работы с ними могут лежать в разных директориях и пакетах в связи с чем возникает проблема того, что разработчик может не знать о существовании того или кода, из-за чего появляется не нужная дупликация функционала.

Проблемы, возникающие при представлении данных в общем виде таковы:

1. Снижение производительности.
2. Отсутствие структуры класса, а также проверок компилятора на соответствие аргументов в функциях работы с данными
3. Необходимость приведения типов в статически типизированных языках

Под отсутствием структуры класса подразумевается, то что у разработчика не будет некоторого класса со всеми полями и методами, а будет одни обобщенный объект в виде словаря, для работы с которым будут отдельно доступны методы. В случае представления в виде обобщенного объекта подсказки среды разработки также будут недоступны.

Использование иммутабельных данных накладывает вычислительные ограничения, а также в некоторых языках требует сторонних библиотек.

В зависимости от реализации разделения данных и их вида, может повлечь за собой более слабую связь между двумя этими понятиями, а также появление дополнительных затрат на валидацию данных в соответствии с их схемой.

Вывод. Несмотря на некоторые отрицательные аспекты, DOP упрощает проектирование и реализацию информационных систем, ставя в приоритет представление данных. Достигается ДОП следованием 4 основным принципам:

1. Отделение кода от данных.
2. Представление данных приложения с помощью общих структур данных.
3. Обработка данных как неизменяемых.
4. Отделение схемы данных от представления данных.

Ben Moseley and Peter Marks. 2006. Out of the tar pit. Software Practice Advance-
ment (SPA) 2006 (2006).

Frederick P. Brooks, Jr. No silver bullet: Essence and accidents of software engineering. Information Processing 1986, Proceedings of the Tenth World Computing Conference, H.-J. Kugler, ed.: 1069–76. Reprinted in IEEE Computer, 20(4):10-19, April 1987, and in Brooks, The Mythical Man-Month: Essays on Software Engineering, Anniversary Edition, Chapter 16, Addison-Wesley, 1995.

Dijkstra. The tide, not the waves. In Peter J. Denning and Robert M. Metcalfe, editors, Beyond Calculation: The Next Fifty Years of Computing, Copernicus, 1997. 1997.

Список используемой литературы

1. *Основы алгоритмов* Яндекс образование [сайт]. Официальный сайт. 2024 год URL:https://education.yandex.ru/handbook/algorithms/article/razdelyaj-i-vlastvuj (дата обращения: 12.03.2024).
2. *Адитья Бхаргава* Грокаем алгоритмы. Иллюстрированное пособие для программистов и любопытствующих: пер. с англ. / изд. «Питер», 2015. – 88 с.
3. *TProger* Алгоритмы и структуры данных для начинающих: двоичное дерево поиска. Официальный сайт. 2024 год URL:https://tproger.ru/translations/binary-search-tree-for-beginners(дата обращения: 12.03.2024).
4. *University of San Francisco* Data Structure Visualizations Официальный сайт. 2024 год URL:https://www.cs.usfca.edu/~galles/visualization/BST.html (дата обращения: 12.03.2024).
5. Algorithmica Официальный сайт. 2024 год URL: https://ru.algorithmica.org/cs/basic-structures/heap/(дата обращения: 12.03.2024).
6. BinaryTreeVisualiser Официальный сайт. 2024 год URL: http://btv.melezinek.cz/binary-heap.html(дата обращения: 12.03.2024).
7. WorkatTech Официальный сайт. 2024 год URL: https://workat.tech/problem-solving/tutorial/sorting-algorithms-quick-

sort-merge-sort-dsa-tutorials-6j3h98lk6j2w(дата обращения: 12.03.2024).
8. Essential Algorithms Essential Programming Books URL:https://www.programming-books.io/essential/algorithms/heap-sort-basic-information-3193e2927dbe4c03bcbc5645fa66cf21 (дата обращения: 12.03.2024).