УДК 004.422

Волкова К.А.,
студент магистратуры
3 курс, факультет заочного обучения
Поволжский государственный университет телекоммуникаций и информатики
Россия, г. Самара
Научный руководитель: Захарова О.И.

АНАЛИЗ ПРИМЕНЕНИЯ КОНВЕЙЕРНОЙ АРХИТЕКТУРЫ ДЛЯ РЕСТРУКТУРИЗАЦИИ И ВАЛИДАЦИИ ДАННЫХ

Аннотация: *Статья содержит анализ применения конвейерной архитектуры для проектирования современного ПО. Рассматриваются достоинства и недостатки применения каналов и фильтров. Приводятся рекомендации по применению такой архитектуры.*

Ключевые слова: *архитектура ПО, конвейерная архитектура, архитектура фильтров и каналов, интеграция, реструктуризация данных.*

Annotation: *The article contains an analysis of the use of the pipeline architecture for the design of modern software. The advantages and disadvantages of using channels and filters are considered. Recommendations on the use of such an architecture are given.*

Key words: *software architecture, pipeline architecture, filter and channel architecture, integration, data restructuring.*

На сегодняшний день глобальная сеть интернет располагает таким количеством информации, которое практически невозможно охватить простому человеку. Десятки и сотни сервисов представляют информацию о товарах и услугах – их стоимость, характеристики, оценки пользователей и клиентов, мнение экспертов. Потенциальные пользователи в зависимости от своих

приоритетов хотят получить товар или услугу максимально дешево, максимально качественно, максимально быстро. Они не хотят тратить время на самостоятельный поиск наиболее оптимального варианта. В силу этого наблюдается тенденция к росту количества сервисов-агрегаторов.

Интеграция сразу с несколькими источниками влечёт за собой ряд сложностей. Различные источники хранят и представляют информацию по-своему. Например, информация из одного источника поступает в формате xml, а из другого – в виде json. Одна и та же информация могла храниться в источниках в разном виде, например, дата может быть представлена цифровым способом или словесно-цифровым, могут быть использованы разные разделители или другой порядок следования числа, месяца и года. Одни и те же сведения могут быть представлены различными типами данных. При попытке обработать разнородные данные одним способом наверняка будут получены ошибки. Для того чтобы работать с данными из нескольких источников может потребоваться инициация целой серии этапов обработки.

Можно создать модуль обработки входящих сообщений, выполняющий все необходимые преобразования. Однако подобный подход не удовлетворяет требованию гибкости и усложняет тестирование. Реализация всех функций в одном компоненте снижает возможность его повторного использования. Напротив, создание небольших, узкоспециализированных компонентов существенно повышает гибкость решения [12].

Для разделения сложной вычислительной задачи на последовательность простых, независимых этапов удобно использовать архитектуру каналов и фильтров (англ. Pipes and Filters). Данные преобразуются из одной формы в другую с использованием различных типов операций. Конвейерная архитектура включает в себя 2 типа независимых объектов – фильтры, которые выполняют преобразования входных данных, и каналы, которые соединяют потоки преобразовываемых данных.

Для аналогии можно представить воду, протекающую через трубы и фильтры. Трубы доставляют воду из одного места в другое, а фильтры каким-то

образом её преобразовывают. Например, вода может пройти через два фильтра, прежде чем попасть к пользователю: один фильтр очищает воду от любых загрязнений, а другой нагревает её. В итоге человек получает чистую и горячую воду.

Рассмотрим простую диаграмму, для демонстрирования функционирования архитектуры канала и фильтра (рис. 1).

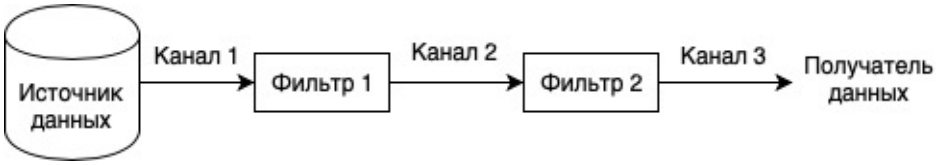

Рисунок 1. Схема функционирования архитектуры каналов и фильтров

Как видно на диаграмме, данные передаются в одном направлении. Он начинается с источника данных, затем перемещается сквозь цепочку каналов и фильтров, а затем, в конце концов, попадает к получателю данных. Изменения в данных выполняются последовательно от фильтра к фильтру. Каждый канал передает выходные данные одного фильтра в качестве входных данных для следующего фильтра. Каждый фильтр выполняет свое локальное преобразование входных данных, полученных от предыдущего канала, а затем отправляет вывод на следующий канал.

Следует отметить, что порядок, в котором фильтры преобразуют данные, может повлиять на конечный результат. В предыдущем примере с водой порядок не имел значения. Вода всегда будет выходить горячей и чистой. Однако, например, в математике порядок операций сложения или умножения в математическом выражении изменяет конечный результат. Это похоже на то, как порядок может повлиять на результат в архитектуре канала и фильтра.

Конвейерная архитектура обладает следующими достоинствами:

1. Поддержка принципов «Низкая связанность» и «Высокое зацепление» (англ. Low Coupling & High Cohesion). Каждый фильтр работает независимо от других фильтров, концентрируясь только на своих входных и выходных данных, что позволяет легко добавлять новые фильтры или перемещать их в системе для

достижения разных результатов. Кроме того, слабая связь позволяет легко вносить изменения в отдельные фильтры, не затрагивая другие фильтры в системе. Это очень важно для разработки быстро меняющихся систем [13].

2. Фильтры можно рассматривать как черные ящики. Пользователям системы не нужно знать внутреннюю работу каждого фильтра. Таким образом, они могут просто использовать фильтр для преобразования различных наборов данных и не беспокоиться о знании логики этого преобразования.

3. Возможность повторного использования. Каждый фильтр можно вызывать и использовать снова и снова с разными входными данными. Фильтры могут многократно использоваться в разных приложениях для разных целей.

4. Составляющие конвейер фильтры могут выполняться на разных компьютерах, что позволяет масштабировать их независимо друг от друга и использовать возможности эластичности, которые предоставляются в большинстве облачных сред. Ресурсоемкие фильтры могут выполняться на оборудовании с высоким уровнем производительности, а другие, менее ресурсоемкие, могут размещаться на более дешевом стандартном оборудовании. Фильтры не должны находиться в одном центре обработки данных или географическом расположении. Это позволит каждому элементу в конвейере работать в среде, близкой к необходимым ресурсам [14].

5. Возможность выполнения обработки для каждого фильтра в параллельном режиме, если входные и выходные данные фильтра структурированы в виде потока. Первый фильтр в конвейере может начать свою работу и вернуть результаты, которые непосредственно передаются следующему фильтру в последовательности, прежде чем первый фильтр завершит свою работу.

6. Устойчивость. Если произойдет сбой фильтра или компьютер, на котором он выполняется, станет недоступен, конвейер позволит перепланировать операции, выполняемые фильтром и передать их другому экземпляру компонента. Сбой одного фильтра не обязательно приводит к сбою всего конвейера.

Тем не менее, у такой архитектуры есть несколько недостатков:

1. Снижение производительности из-за чрезмерных издержек в фильтрах. Каждый фильтр получает входные данные, анализирует их, выполняет преобразования и затем передаёт данные. Следующий фильтр будет делать то же самое – анализировать входные данные, преобразовывать и отправлять результат в другой фильтр, который также будет делать то же самое. Подобный синтаксический анализ приводит к росту накладных расходов. По мере добавления всё большего количества фильтров производительность системы будет быстро снижаться. Кроме того, эта архитектура может привести к перегрузке фильтров огромными объемами данных для обработки. В то время как один фильтр усердно работает над обработкой большого количества данных, другие фильтры могут голодать в ожидании своих входных данных. [13]

2. Использование большого числа каналов. Следует помнить, что реализация канала предполагает расходование ресурсов памяти и процессора.

3. Не может использоваться для интерактивных приложений. Передача и преобразование данных займет время в зависимости от объема передаваемых данных. Таким образом, этот тип архитектуры не подходит для приложений, которые требуют быстрых ответов [14].

Взаимодействие между компонентами рекомендуется реализовать с помощью асинхронного обмена сообщениями. В этом случае компонент сможет отправить сообщение другому компоненту, не дожидаясь от него ответа. Использование асинхронного обмена сообщениями позволит организовать одновременную обработку нескольких сообщений, по одному на каждый компонент. По сравнению с последовательной обработкой сообщений конвейерная обработка существенно повышает пропускную способность системы. [12]

Несмотря на вышеперечисленные недостатки, многие преимущества использования архитектуры конвейеров и фильтров делают ее очень популярной во многих системах. Простота и эффективность каналов и фильтров завоевала этой архитектуре огромную популярность, а несложная семантика позволила

описать ее формальными методами. Когда вам необходимо по-разному манипулировать различными наборами данных, вам следует рассмотреть возможность использования архитектуры. Разбиение системы на фильтры и каналы может обеспечить большее повторное использование, уменьшить связность и обеспечить большую гибкость в вашей системе.

ИСПОЛЬЗОВАННЫЕ ИСТОЧНИКИ

1. Хоп, Г. Шаблоны интеграции корпоративных приложений / Г. Хоп, Б. Вульф.— М.: ООО «И.Д. Вильямс», 2007.— 672 с.
2. Garlan, D. An Introduction to Software Architecture / D. Garlan, M. Shaw.— School of Computer Science Carnegie Mellon University, 1994.— 39 c.
3. Шаблон каналов и фильтров. [Электронный ресурс]. URL: https://docs.microsoft.com/ru-ru/azure/architecture/patterns/pipes-and-filters (дата обращения: 15.12.2018).