**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «АСТРАХАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ   
ИМЕНИ В.Н. ТАТИЩЕВ»**

Факультет цифровых технологий и кибербезопасности

Кафедра цифровых технологий

Отчет по лабораторной работе № 1

выполнена в рамках изучения дисциплины

«Архитектура систем ии»

Направление подготовки: 09.04.02. Информационные системы и технологии

Направленность (профиль): Проектирование и разработка систем искусственного интеллекта" (ДИИ) очная форма обучения

Исполнитель: студент группы ДИИ-15

Глебов Виктор Валерьевич

Проверил: доцент кафедры прикладной   
математики и информатики

Гордеев Иван Иванович

СОДЕРЖАНИЕ

[Архитектурные шаблоны в разработке систем с использованием систем искусственного интеллекта 3](#_Toc131885008)

[1. Многоуровневая архитектура 4](#_Toc131885009)

[2. Каналы и фильтры 6](#_Toc131885010)

[3. Клиент-сервер 7](#_Toc131885011)

[4. Модель-представление-контроллер 8](#_Toc131885012)

[5. Событийно-ориентированная архитектура (EDA) 8](#_Toc131885013)

[6. Микросервисная архитектура 10](#_Toc131885014)

# Архитектурные шаблоны в разработке систем с использованием систем искусственного интеллекта

Времени, когда любая программа была произведением искусства, давно прошли, так как на текущий момент — это разновидность промышленного производства, это унификация и стандартизация. В разработке программного обеспечения такой стандартизацией является архитектурные шаблоны, которые аккумулируют в себе удачные решения и в дальнейшем распространяются на другие проекты. Архитектурный шаблон — это обобщенное часто используемое решение распространенной задачи в архитектуре ПО в заданном контексте. Использование архитектурных шаблонов обеспечивает ускорение разработки ПО и позволяет избегать типичных ошибок. Существуют разные архитектурные шаблоны для разработки ПО. Это:

* Многоуровневая архитектура.
* Каналы и фильтры.
* Клиент — сервер.
* Модель — представление — контроллер.
* Событийно-ориентированная архитектура.
* Архитектура на основе микросервисов.

Рассмотрим данные архитектуры подробнее.

# Многоуровневая архитектура

Многоуровневая архитектура – архитектурный шаблон, состоящий из нескольких отдельных горизонтальных уровней, которые функционируют вместе как единое целое программного обеспечения (рисунок 1).

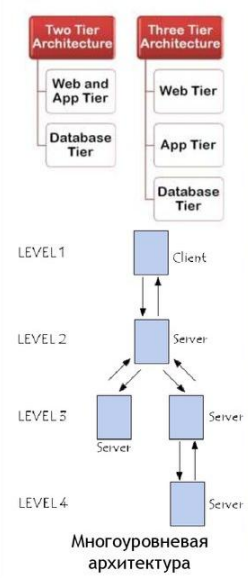


Рисунок 1 – многоуровневая архитектура

Слой - это логическое разделение компонентов или кода. Многоуровневая архитектура программного обеспечения в большинстве случаев содержит в себе четыре уровня, а именно:

* Представление данных – позволяет пользоваться данными для принятия решений;
* Бизнес-логика - обеспечивает манипулированием данными на уровне бизнес-процессов;
* Хранение данных (методы манипулирования данными);
* Базы данных.

Последние два метода принято считать не столько уровнями, сколько оболочками. В системах искусственного интеллекта, аналогично, архитектура представлена следующим образом:

* Интеллектуальный интерфейс;
* Поиск решений;
* Извлечение и обновление знаний до обучения (методы манипулирования знаниями);
* База данных и знаний.

Для реализации требуется обеспечивать контекст интеллектуальной системы. Это означает, что нужно чётко документировать разделения задач, поскольку задачи на различных уровнях решаются различными людьми, то необходимо обеспечить документирование разделения задач. Поскольку вертикальные связи – есть суть работы интеллектуальной системы, должна быть координация деятельности разработчиков. При этом иерархическиая организация должна реализовываться на основе модульности. Модульность означает независимые блоки, которые разрабатываются разными разработчиками, отдельно отлаживаются, отдельно документируются и для этого требуется обеспечить:

* Переносимость данных;
* Модифицируемость;
* Повторное использование данных.

То есть многоуровневая организация обеспечивает разделение труда, повышение производительности за счёт того, что происходит привлечение групп разработчиков, но при этом накладываются определённые требования, увеличивающие трудоёмкость. Реализация многоуровневой архитектуре требует решения задач разделения уровней, а именно:

* Изоляция разработчиков, связь уровней только через публичный интерфейс;
* Закрепление за каждым уровнем ролей и ответственности. Разделение задач;
* Технические разделение на компоненты;
* Невозможность обхода уровней.

Основными недостатками данной архитектуры является:

* Низкая производительность из-за искусственной изоляции уровней;
* Высокая стоимость вследствие избыточных требований к модулям;
* Усложнение системы.

Таким образом, наилучшее применение данной архитектуры являются несложные системы при ограниченном бюджете и времени.

# Каналы и фильтры

Архитектура «Каналы и фильтры» может применяться в системах ИИ, задачи в которых являются сепарабельными. Например, в беспилотном автомобиле есть задачи навигации и задачи пилотирования, которые решаются отдельно. Задача пилотирования, в свою очередь, распадается на задачи контроля движения в потоке, маневрирования, распознавания дорожных знаков и светофоров, обнаружения препятствий, экстренного торможения и др.

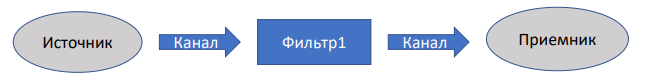


Рисунок 2 – Классификация систем ИИ по методам обработки данных (несвязанные задачи)

Задачи могут быть несвязанными, если они замыкаются на разные приемники данных, либо связанными. Примером связанных задач являются задача пилотирования самолета и задача предотвращения столкновений (traffic collision avoidance system), которые замыкаются на одни и те же органы управления и должны при этом не допускать выхода за пределы допустимых параметров, в частности закритических углов тангажа, которые могут привести к сваливанию в штопор и катастрофе.

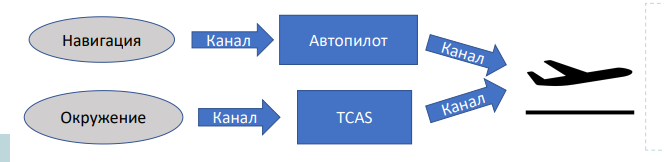


Рисунок 3 – Классификация систем ИИ по методам обработки данных (связанные задачи)

# Клиент-сервер

Архитектура клиент-сервер является предпочтительной в тех случаях, когда требуются мощные вычислительные ресурсы и/или доступ к центральной базе знаний. Естественно, задачи, порождаемые клиентом, недолжны загружать сервер на длительное время. Преимуществами архитектуры клиент-сервер также являются простота масштабирования, модифицируемость и повторное использование. Недостатками являются неприменимость к системам реального времени, поскольку серверное обслуживание не гарантирует мгновенный и непрерывный доступ к ресурсам, а также наличие единой точки отказа. Кроме того, необходимость контроля доступа может быть сдерживающим фактором.

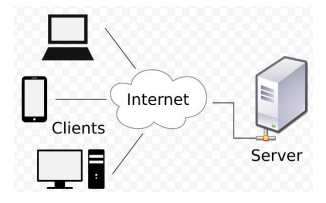


Рисунок 4 – архитектура клиент-сервер

# Модель-представление-контроллер

Шаблон Модель-представление-контроллер основан на разделении функционалов хранения знаний, их обработки и интерфейса пользователя.

Ключевая идея MVC состоит в том, что любое приложение с пользовательским интерфейсом в первом приближении можно разбить на 2 модуля: модуль, отвечающий за реализацию бизнес-логики приложения, и пользовательский интерфейс.



Рисунок 5 – шаблон MVC

Такое разделение позволяет легко персонифицировать интерфейс пользователя, а также гибко реагировать на изменения в базах данных знаний. К недостаткам следует отнести сложность реализации, которая может стать чрезмерной для несложных задач. Кроме того, данные абстракции могут быть нереализуемыми в некоторых средствах разработки.

# Событийно-ориентированная архитектура (EDA)

База знаний представлена фактами типов: событие и состояние. Событийно-ориентированная архитектура хорошо приспособлена к темпоральным базам знаний. В таких системах база знаний представлена основными фактами двух видов: события и состояния, которые являются следствиями событий. События могут только пополнять базу знаний, но не могут удаляться. Однако, другие события могут отменять действие предыдущих.

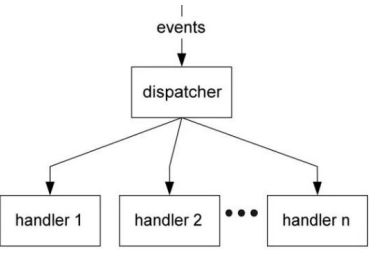


Рисунок 6 – EDA

Event driven architecture (EDA) может применяться при разработке и реализации приложений и систем, передающих события среди слабосвязанных программных компонентов и служб.



Рисунок 7 – пример EDA (состояние семейного положения человека)

Преимуществом EDA состоит в компактном представлении данных, возможности быстрого извлечения текущего состояния, а также доступа к ретроспективным данным. Недостатком является сложность восстановления после ошибок, которая может потребовать обработки событий за длительный период времени. Примером событийно-ориентированной архитектуры являются регистры бухгалтерского учета. Наличные деньги пересчитать легко, а с безналичными сложнее: их текущее количество (состояние) определяется историей прихода и расхода (события) и записывается на бухгалтерские счета (регистры). Это обеспечивает быстрый доступ к текущему состоянию счета, однако, в случае потери этого регистра потребуется обработка всех приходных и расходных операций с начала года или месяца.

# Микросервисная архитектура

Микросервисная архитектура является предпочтительной, если все задачи являются сепарабельными, причем, распадающимися на типовые процедуры. В таком случае новую задачу можно собирать как конструктор Лего из отдельных микромодулей. Примерами микросервисов в обработке данных могут быть фильтрация, нормализация, унификация данных. Микросервисная архитектура обеспечивает высокое повторное использование элементов архитектуры, однако, накладывает более повышенные требования к универсальности.

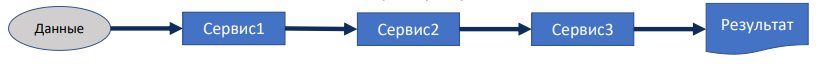


Рисунок 8 – конвейер микросервисов

Основными преимуществами являются:

* Гетерогенность сред разработки;
* Модифицируемость;
* Повторное использоване.

К недостаткам можно отнести:

* Накладные расходы на организацию;
* Сложность мониторинга.

Таким образом, не существует универсального архитектурного шаблона для программных систем в целом и для систем ИИ в частности. Каждый шаблон имеет свои преимущества и недостатки, и выбор конкретного шаблона должен выбираться с учетом специфики задач и данных.

СПИСОК ЛИТЕРАТУРЫ

1. Остроух А.В. Математическая модель связей в системе диагностики электрооборудования автомобилей / А.В. Остроух, А.А. Солнцев, О.Ф. Калухов, Г.Г. Ягудаев // Вестник МАДИ. – 2010. – Вып. 2(21). – С. 66-70;
2. Остроух А.В. Автоматизация управления автотранспортными предприятиями. Новый подход на основе интеллектуальных мультиагентных систем / А.В. Остроух, А.В. Воробьева, Н.Е. Суркова. – Saarbrucken, Germany: LAP LAMBERT Academic Publishing, 2015. – 117 p. – ISBN 978-3-659-47576-4.
3. Суркова Н.Е. Методология структурного проектирования информационных систем: монография / Н.Е. Суркова, А.В. Остроух. – Красноярск: Научно-инновационный центр, 2014. – 190 с. – ISBN 978-5-906314-16-1.
4. Суркова Н.Е. Методы проектирования информационных систем / Н.Е. Суркова, А.В. Остроух – М.: РосНОУ, 2004. – 144 с. – ISBN 5- 89789-021-8.
5. Остроух А.В. Основы построения систем искусственного интеллекта для промышленных и строительных предприятий: монография / А.В. Остроух. – М.: ООО «Техполиграфцентр», 2008. – 280 с. – ISBN 978- 5-94385-033-2.