Основы

Информационная система

# Распределённые системы

Распределенная система - это система, в которой:

* Компоненты размещены на разных физических или виртуальных узлах (например, серверах, контейнерах, облаках),
* компоненты взаимодействуют через сеть (обычно TCP/IP, HTTP, gRPC и т.д.),
* Но выглядят для пользователя (или даже программиста) как единое целое.

Когда говорят, что service-oriented architecture (SOA) — это распределённая архитектура, это означает, что система состоит из нескольких компонентов (сервисов), работающих на разных машинах или в разных процессах, которые обмениваются сообщениями по сети.

|  |  |
| --- | --- |
| **Пример** | **Что распределено** |
| Веб-приложение | UI на фронтенде, API-сервер, база данных — на разных машинах |
| SOA / микросервисы | Каждый сервис — отдельный процесс / контейнер, общение по сети |
| Облачные системы | Компоненты живут в разных дата-центрах или регионах |
| Hadoop / Ceph / Kafka | Данные и вычисления распределены между множеством узлов |

Признаки распределённой системы:

* Сетевое взаимодействие — компоненты обмениваются сообщениями через сеть.
* Независимые сбои — один компонент может выйти из строя, другие — продолжат работать.
* Распределённое состояние — данные и логика могут быть распределены по множеству узлов.
* Нет глобальных часов — нет общего точного времени, как в одной машине.
* Трудности синхронизации и отказоустойчивости — возникают проблемы, которые не существуют в монолитных системах (например, частичные сбои, задержки, потеря сообщений).

Виды архитектур

Enterprise Architecture (например, TOGAF)

Системной инженерии (например, ISO/IEC 42010)

У системы существуют несколько родов архитектур:

логическая

физическая

# Логические архитектуры системы

Architectural pattern

Паттерны в отличие от архитектур не зависят от моды и других факторов и регулярно появляются при проектировании ИС.

# Big Ball of Mud

# Client/Server

Structural Design Pattern

# Приспособленец (Flyweight)

### 📌 Назначение (Intent):

Использовать разделение общего состояния (sharing), чтобы эффективно поддерживать большое количество мелких объектов.

### 🎯 Мотивация:

Некоторые приложения выигрывают от использования объектно-ориентированного подхода на всех уровнях, но наивная реализация может оказаться слишком затратной по памяти и производительности.

Пример — текстовый редактор:

* Объекты могут использоваться для представления встроенных элементов (таблиц, изображений и т.п.).
* Однако обычно не создаётся отдельный объект для каждого символа, так как это слишком тяжело.
* Если бы каждый символ был объектом, это позволило бы:  
  + обрабатывать символы и элементы единообразно (например, при отрисовке и форматировании),
  + гибко расширять систему, добавляя новые языки или шрифты, не затрагивая остальной код,
  + сделать структуру объектов аналогичной структуре документа.

### 📌 Суть подхода:

* Вместо создания миллиона одинаковых объектов (например, каждой буквы "а"), создаём один общий (разделяемый) объект и много "контекстов", которые определяют, где он используется.
* Это особенно полезно в ситуациях с огромным числом однотипных элементов, где можно отделить:  
  + внутреннее (разделяемое) состояние — общее для всех,
  + внешнее (уникальное) состояние — задаётся извне и не хранится внутри объекта.

### 🧠 Пример:

* В документе 100 000 символов, но только 100 уникальных глифов (букв, знаков).
* Вместо 100 000 объектов — храним 100 общих (разделяемых) объектов + координаты каждого символа.

Если хочешь, могу включить также UML-диаграмму и список участников паттерна.

Архитектуры ИС

В книге Fundamentals of Software Architecture рассмотрены следующие архитектурные стили:

# Layered Architecture Style

или “Multitier architecture”

[Multitier architecture - Wikipedia](https://en.wikipedia.org/wiki/Multitier_architecture)

In a logical multilayer architecture for an information system with an object-oriented design, the following four are the most common:

1. Presentation layer (a.k.a. UI layer, view layer, presentation tier in multitier architecture)

UI который видит пользователь.

Веб-интерфейс на React, Angular или Vue

Мобильное приложение (iOS/Android)

1. Application layer (a.k.a. service layer[8][9] or GRASP Controller Layer [10])

Обрабатывает пользовательские запросы, координирует вызовы к бизнес-логике.

Является посредником между UI и бизнес-логикой.

Может реализовывать сценарии, т.е. последовательности действий (например, создать заказ → рассчитать цену → отправить подтверждение).

Контроллеры в Spring MVC, ASP.NET

Сервисные классы, которые вызываются из контроллеров

REST API, GraphQL endpoint

1. Business layer (a.k.a. business logic layer (BLL), domain logic layer)

Реализует предметную область (domain): бизнес-правила, алгоритмы, расчёты, проверки.

Независим от UI и хранения данных.

Основной смысл системы — именно здесь.

🔧 Примеры:

Классы вроде Order, Invoice, Customer с методами типа applyDiscount(), calculateTotal()

Правила валидации, формулы расчёта

Состояния объектов, бизнес-процессы

1. Data access layer (a.k.a. persistence layer, logging, networking, and other services which are required to support a particular business layer).

БД всей системы.

Обеспечивает чтение/запись данных (из БД, файлов, сетей и т.д.).

Абстрагирует работу с хранилищем.

Может включать кэширование, логирование, сетевые вызовы.

🔧 Примеры:

ORM (Hibernate, SQLAlchemy, Entity Framework)

Репозитории (UserRepository, OrderDAO)

Layered Architecture Style

Pipeline Architecture Style

Microkernel Architecture Style

Service-Based Architecture Style

Event-Driven Architecture Style

Space-Based Architecture Style

# (SOA) Orchestration-Driven Service-Oriented Architecture

Microservices Architecture

Анализ систем

# System analysis

1. Системный анализ - совокупность методов для представления некоторой системы в виде модели или схемы.
2. Анализ в свою очередь это разложение системы на такую модель.

## Structured analysis

1. Represents the system in terms of data and the processes that act upon that data. System development is organized into phases with durables and milestones to measure progress. The waterfall model typically consists of five phases: testing and maintenance & evolution. Iteration is possible among the phases.
2. structured analysis uses DFDs to model data and processes

### Structured analysis model

1. Конечным результатом структурного анализа является создание модели системы.
2. В структурном анализе моделью являются совокупность:

* источников данных
* data flows
* процессы

1. Эта модель выражается в DFD диаграмме, содержащей все эти элементы.

Но существуют и другие диаграммы для построения модели структурного анализа.

#### DFD

1. file:///D:/studwork/2%20%D0%BF%D0%BE%D1%81%D1%82%D1%83%D0%BF%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5/%D0%BE%D0%BB%D0%B8%D0%BC%D0%BF%D0%B8%D0%B4%D1%8B%20%D0%B8%20%D1%8D%D0%BA%D0%B7%D1%8B/%D0%98%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D1%8B%D0%B5%20%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D1%8B%20%D0%B8%20%D1%82%D0%B5%D1%85%D0%BD%D0%BE%D0%BB%D0%BE%D0%B3%D0%B8%D0%B8.%20%D0%A0%D0%B0%D1%81%D0%BF%D1%80%D0%B5%D0%B4%D0%B5%D0%BB%D0%B5%D0%BD%D0%BD%D1%8B%D0%B5%20%D0%B8%D0%BD%D1%82%D0%B5%D0%BB%D0%BB%D0%B5%D0%BA%D1%82%D1%83%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B5%20%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D1%8B%D0%B5%20%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D1%8B/%D1%80%D0%B0%D0%B7%D0%B4%D0%B5%D0%BB%209.%20%D0%90%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D1%8B%20%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D1%8B%D1%85%20%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC/systems-analysis-and-design\_compress.pdf
2. DFD диаграмма состоит из компонентов:

* Entity (источник данных)
* data flow
* process ( программы которые трансформируют данные)
* хранилища данных

1. Эти компоненты лишь абстракции созданные для упрощенной визуализации

НО те физические вещи, стоящими за этими компонентами описаны в **словаре данных**.

Что представляют из себя источники данных (клиенты, банк итд), какая программа осуществляет тот или иной процесс - всё это в словаре данных.

#### ER-diagram

##### Конструктивные элементы

В ER-диаграммах (Entity-Relationship) основные конструктивные элементы описывают структуру данных и связи между ними.

**1️⃣ Сущности (Entities)**

* Представляют объекты реального мира, о которых нужно хранить информацию.
* Обозначаются **прямоугольниками**.
* Примеры: Student, Course, Order.

**2️⃣ Атрибуты (Attributes)**

* Свойства или характеристики сущностей.
* Обозначаются **эллипсами**, соединёнными с сущностью линией.
* Типы атрибутов:
  + **Простые (Simple)** — неделимые, например name.
  + **Составные (Composite)** — могут быть разбиты на части, например FullName = FirstName + LastName.
  + **Многозначные (Multivalued)** — могут иметь несколько значений, например PhoneNumbers.
  + **Ключевые (Key/Primary Key)** — уникально идентифицируют сущность, например StudentID.

**3️⃣ Связи (Relationships)**

* Показывают, как сущности связаны между собой.
* Обозначаются **ромбами**, соединяющими прямоугольники сущностей.
* Связи бывают:
  + **Один к одному (1:1)**
  + **Один ко многим (1:N)**
  + **Многие ко многим (M:N)**

**4️⃣ Атрибуты связей**

* Свойства самой связи.
* Например, связь Enrollment между Student и Course может иметь атрибут Grade.
* Обозначаются **эллипсом**, соединённым с ромбом связи.

**5️⃣ Общие элементы**

* **Подтип / Супертип (Generalization / Specialization)** — отношения наследования между сущностями.
* **Объединение (Aggregation)** — когда связь рассматривается как сущность для других связей.

## Object-oriented analysis

1. Рассматривает систему с точки зрения объектов, объединяющих данные и процессы. Объекты представляют реальных людей, вещи, транзакции и события. По сравнению со структурным анализом, фазы 0-0 как правило, более интерактивны. Можно использовать waterfall модель или модель, подчеркивающую большую итеративность.

### Object Modeling

1. The end product of 0-0 analysis is an **object model**, which represents the information system in terms of objects and 0-0 concepts.
2. systems analysts use UML to describe 0-0 systems (Object-oriented systems).

#### UML

1. UML Язык который представляет систему как набор объектов, которые принадлежат некоторым классам и у которых есть методы. А также бизнес логику всех существующих объектов. Механика такая же как у языков с ООП.   
     
   Объекты бывают двух типов

* актор (пациент)
* use case (пройти обследование).

# UML

## Ассоциация

В **UML (Unified Modeling Language)** **ассоциация** — это **связь между двумя (или более) классами**, которая показывает, что **один объект "знает" о другом и может с ним взаимодействовать**.

### **🧩 Определение:**

**Ассоциация (association)** — это **структурная связь**, указывающая на **отношения между объектами классов**, обычно через **ссылки (references)**.

### **📌 Пример:**

Если есть два класса:

Класс: Автомобиль

Класс: Двигатель

И в UML между ними нарисована **сплошная линия** — это **ассоциация**:

[Автомобиль] ────────── [Двигатель]

Это означает: **автомобиль имеет двигатель** (и знает о нём, может обращаться к нему).

### **🔢 Мультиплицирование (множественность):**

Ассоциация может сопровождаться **числовыми значениями** (мультиплицированием), например:

[Компания] 1 ───── \* [Сотрудник]

Означает:

* Компания имеет **много сотрудников**,
* Каждый сотрудник принадлежит **одной компании**.

### **📘 Типы ассоциаций:**

|  |  |  |
| --- | --- | --- |
| **Тип** | **Обозначение** | **Значение** |
| **Ассоциация** | Сплошная линия | Объекты знают друг о друге |
| **Агрегация** | Линия с **пустым ромбом** | "Целое–часть", слабая связь (например, класс и ученики) |
| **Композиция** | Линия с **закрашенным ромбом** | "Жёсткое целое–часть", сильная связь, часть не может существовать без целого (например, комната и дом) |

### **🧠 В коде:**

Ассоциация в UML часто отражается в виде **поля в классе**:

class Car:

def \_\_init\_\_(self, engine):

self.engine = engine # ассоциация с объектом Engine

# Примеры ИС

## Виды ИС

ERP (Enterprise Resource Planning)

KMS (Knowledge Management System)

LMS (Learning Management System)

CMS (Content Management System)

SCADA (Supervisory Control and Data Acquisition)

DMS (Document Management System)

HRM / HCM (Human Resource Management / Capital Management)

BPM (Business Process Management)

BI (Business Intelligence)

Help Desk / Service Desk

### 📦 1. \*\*ERP (Enterprise Resource Planning)\*\*

\*\*Для управления ресурсами предприятия.\*\*

Функциональность: финансы, производство, закупки, склад, кадры, логистика и пр.

\*\*Примеры:\*\* SAP, Oracle E-Business Suite, 1С\:ERP.

---

### 👨‍🏫 2. \*\*LMS (Learning Management System)\*\*

\*\*Для организации дистанционного обучения.\*\*

Функциональность: курсы, тестирование, трекинг прогресса, отчёты.

\*\*Примеры:\*\* Moodle, Canvas, Google Classroom.

---

### 🛒 3. \*\*CMS (Content Management System)\*\*

\*\*Для управления веб-контентом.\*\*

Функциональность: создание и редактирование сайтов без программирования.

\*\*Примеры:\*\* WordPress, Joomla, Drupal.

---

### 🏭 4. \*\*SCADA (Supervisory Control and Data Acquisition)\*\*

\*\*Для мониторинга и управления промышленными процессами.\*\*

Используется в энергетике, водоснабжении, производстве.

\*\*Примеры:\*\* Wonderware, Siemens WinCC.

---

### 📑 5. \*\*DMS (Document Management System)\*\*

\*\*Для управления электронными документами.\*\*

Функциональность: хранение, версионирование, согласование.

\*\*Примеры:\*\* Directum, DocsVision, Alfresco.

---

### 👨‍⚖️ 6. \*\*HRM / HCM (Human Resource Management / Capital Management)\*\*

\*\*Для управления персоналом.\*\*

Функциональность: учёт сотрудников, расчёт зарплаты, рекрутинг.

\*\*Примеры:\*\* SAP HCM, BambooHR, 1С:Зарплата и кадры.

---

### 💼 7. \*\*BPM (Business Process Management)\*\*

\*\*Для моделирования и автоматизации бизнес-процессов.\*\*

Часто интегрируется с ERP и CRM.

\*\*Примеры:\*\* ELMA, Camunda, Bizagi.

---

### 📊 8. \*\*BI (Business Intelligence)\*\*

\*\*Для аналитики и визуализации данных.\*\*

Функциональность: отчёты, дашборды, прогнозирование.

\*\*Примеры:\*\* Power BI, Tableau, Qlik.

---

### 💬 9. \*\*Help Desk / Service Desk\*\*

\*\*Для поддержки пользователей и управления инцидентами.\*\*

Функциональность: тикеты, SLA, база знаний.

\*\*Примеры:\*\* Jira Service Management, Zendesk, OTRS.

---

### 🧠 10. \*\*KMS (Knowledge Management System)\*\*

\*\*Для сбора, хранения и передачи знаний.\*\*

Используется внутри компаний, особенно R\&D отделами.

\*\*Примеры:\*\* Confluence, Notion, Guru.