

Проектирование машиночитаемых форматов обмена данными: от теории к практике.

Машиночитаемый формат — это способ организации данных в электронном документе (обычно в форматах **XML** или **JSON**), который позволяет компьютерным программам автоматически распознавать, извлекать и обрабатывать информацию без участия человека. В этой статье мы рассмотрим классический жизненный цикл разработки таких форматов, методологию модельно-ориентированной системной инженерии (**MBSE / Arcadia**) и разберем сквозной практический пример для автоматизации склада.

Часть 1. Теоретические основы проектирования форматов

1. Традиционный жизненный цикл разработки.

Процесс создания машиночитаемого формата представляет собой стандартизированный цикл, превращающий бизнес-требования в строгую техническую структуру. Он состоит из 6 ключевых этапов:

Этап 1. Сбор требований и составление ТЗ (Анализ). Определяются бизнес-цели и нормативная база (законы, приказы). Формируется полный перечень реквизитов (например, ИНН, ФИО), типы полей (текст, число, дата) и логические связи (вложенность данных).

Этап 2. Проектирование логической модели (Архитектура). Бизнес-требования переводятся на язык ИТ-архитектуры. На этом этапе создаются объектные модели (UML-диаграммы) и задаются правила валидации (например, ИНН должен состоять из 10 цифр). Также выбирается базовый формат: **XML** (для строгих регламентированных документов с цифровой подписью в ЭДО) или **JSON** (для быстрой передачи данных через веб-сервисы и API).

Этап 3. Разработка схемы данных (Техническое описание). Создается «паспорт» формата, по которому компьютеры будут проверять файлы. Для XML создается **XSD-схема** (.xsd), а для JSON — **JSON Schema**. Здесь же подключаются справочники и классификаторы (НСИ).

Этап 4. Разработка структуры и шаблона. На основе схемы создаются эталонные образцы файлов для программистов.

Этап 5. Тестирование и валидация. Проводится синтаксический контроль (проверка целостности тегов и вложенности) и кросс-валидация парсерами с использованием созданных схем данных.

Этап 6. Публикация и интеграция. Формат официально утверждается. Разработчики внедряют библиотеки (сериализаторы) в код учетных систем (например, 1С) для автоматической выгрузки данных из баз данных SQL и наложения электронной подписи (ЭП).

2. Применение методологии MBSE и Arcadia

Традиционный подход (описание ТЗ текстом в Word) часто ведет к ошибкам и двоякому пониманию. Применение концепции **MBSE** (Модельно-ориентированная системная инженерия) и методологии **Arcadia** (в инструменте Capella) позволяет создать «цифровой двойник» документа, где схема XML/JSON автоматически генерируется из визуальной модели.

Методология делит проектирование на 4 уровня:

- 1. Операционный анализ (Operational Analysis):** Определение ролей (актеров) и их бизнес-целей (зачем нужен формат).
- 2. Системный анализ (System Analysis):** Описание требований к формату как к «черному ящику» (что входит и выходит, какие функции выполняет).
- 3. Логическая архитектура (Logical Architecture):** Проектирование объектной модели (диаграммы классов) с указанием блоков, связей, типов данных и кратности (0..*, 1..1) без привязки к технологиям.
- 4. Физическая архитектура (Physical Architecture):** Реализация модели в конкретном синтаксисе (XML/JSON) с использованием автоматической генерации схем (Model-to-Text).

Часть 2. Практический пример: ИС «Умный склад»

Рассмотрим сквозной пример проектирования формата обмена данными для компании ООО «Логист-Про».

Бизнес-контекст: Ручная сверка бумажных накладных вызывает задержки ввода данных в 1С. Требуется единый машиночитаемый формат **Документа складского грузооборота (складского ордера)** для обмена данными между ТСД (терминалами сбора данных) и ИС «Умный склад» в реальном времени.

Проектирование по методологии Arcadia

Уровень 1. Операционный анализ

Акторы: Кладовщик, Поставщик, Клиент.

Деятельность: Приемка товара от поставщика, размещение в ячейки, отбор товара под заказ.

Бизнес-цель: Исключить бумагу, сократить цикл обработки на 40%, поднять точность учета до 99.8%.

Уровень 2. Системный анализ

Функции системы: Импорт ожидаемых поставок из 1С, выдача заданий на ТСД, валидация по штрих-коду, экспорт результатов в 1С.

Сущность данных: Единый файл **WarehouseOrder** (Складской ордер), содержащий как задание, так и отчет.

Уровень 3. Логическая архитектура (Объектная модель)

Структура логических классов и типов данных:

Класс WarehouseOrder (Корневой элемент):

- OrderId (Строка/UUID, обязательное) — ID документа.
- OrderType (Перечисление: Inbound / Outbound, обязательное).
- Status (Строка, обязательное) — В работе / Завершен / Ошибка.
- WarehouseZone (Строка, обязательное) — Зона склада.

Класс ActorInfo (Участники, кратность 1..*):

- Role (Перечисление), Name (Строка), Id (Строка — ИНН или табельный номер).

Класс OrderLine (Товарные позиции, кратность 1..*):

- LineNumber (Целое число), Sku (Артикул/штрих-код), ProductName (Наименование), ExpectedQty (Ожидаемое количество), AcceptedQty (Фактическое количество), AddressCell (Адрес ячейки).

Уровень 4. Физическая архитектура (Реализация)

Ниже представлены два варианта реализации физической структуры одного и того же складского ордера.

Часть 3. Разработка XSD-схемы для валидации складского ордера

XSD (XML Schema Definition) — это специальный язык описания структуры XML-документа . Он выполняет роль «технического паспорта» или строгого контракта , который позволяет автоматическим парсерам мгновенно проверять файлы на синтаксическую корректность, типы данных и обязательность полей .

Шаг 1. Определение структуры и пространств имен (Корневой элемент)

Проектирование начинается с корневого тега `<xs:schema>`, где фиксируется целевое пространство имен (`targetNamespace`), соответствующее нашему проекту . Корневым элементом самого документа объявляется `WarehouseOrder` .

```
xml
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://w3.org"
    targetNamespace="https://logist-pro.ru"
    xmlns="https://logist-pro.ru"
    elementFormDefault="qualified">

    <!-- Корневой элемент документа -->
    <xs:element name="WarehouseOrder">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="Actors" type="ActorsType"/>
                <xs:element name="OrderLines" type="OrderLinesType"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>

    <!-- Определение атрибутов корневого элемента -->
    <xs:attribute name="OrderId" type="xs:string" use="required"/>
    <xs:attribute name="OrderType" type="OrderTypeEnum" use="required"/>
    <xs:attribute name="Status" type="xs:string" use="required"/>
    <xs:attribute name="WarehouseZone" type="xs:string" use="required"/>
</xs:schema>
```

Шаг 2. Описание сложных типов данных (Вложенные структуры)

Согласно нашей логической модели, документ содержит блоки участников (`Actors`) и товарных позиций (`OrderLines`) . Зададим правила их вложенности и кратности с помощью элементов `<xs:sequence>` и параметров `minOccurs` / `maxOccurs` .

```
xml
<!-- Тип данных для блока участников -->
<xs:complexType name="ActorsType">
    <xs:sequence>
        <!-- Документ должен содержать от 1 до бесконечности участников -->
        <xs:element name="Actor" maxOccurs="unbounded">
            <xs:complexType>
                <xs:attribute name="Role" type="xs:string" use="required"/>
                <xs:attribute name="Name" type="xs:string" use="required"/>
                <xs:attribute name="Id" type="xs:string" use="required"/>
            </xs:complexType>
        </xs:element>
    </xs:sequence>
</xs:complexType>

<!-- Тип данных для блока товарных позиций -->
```

```

<xs:complexType name="OrderLinesType">
  <xs:sequence>
    <!-- Документ должен содержать минимум 1 товарную позицию -->
    <xs:element name="OrderLine" maxOccurs="unbounded">
      <xs:complexType>
        <xs:attribute name="LineNumber" type="xs:integer" use="required"/>
        <xs:attribute name="Sku" type="SkuType" use="required"/>
        <xs:attribute name="ProductName" type="xs:string" use="required"/>
        <xs:attribute name="ExpectedQty" type="xs:decimal" use="required"/>
        <xs:attribute name="AcceptedQty" type="xs:decimal" use="optional"/>
        <xs:attribute name="AddressCell" type="xs:string" use="optional"/>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>

```

Шаг 3. Внедрение бизнес-ограничений (Валидация типов)

Одно из главных преимуществ MBSE-подхода и XSD — возможность жестко закрепить правила валидации прямо в схеме. Ограничим поле OrderType только двумя значениями (перечисление), а артикул Sku (штрих-код) заставим проверять по регулярному выражению (ровно 13 цифр).

```

xml
<!-- Ограничение для типа ордера (Только Inbound или Outbound) -->
<xs:simpleType name="OrderTypeEnum">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Inbound"/>
    <xs:enumeration value="Outbound"/>
  </xs:restriction>
</xs:simpleType>

<!-- Ограничение для SKU (строка 13 цифр формата EAN-13) -->
<xs:simpleType name="SkuType">
  <xs:restriction base="xs:string">
    <xs:pattern value="\d{13}"/>
  </xs:restriction>
</xs:simpleType>

</xs:schema>

```

Как работает автоматическая проверка (Кросс-валидация)

Когда ТСД отправляет XML-файл в информационную систему, парсер на сервере выполняет автоматический контроль:

1. **Если в XML-файле** атрибут OrderType будет иметь значение "Draft", валидатор выдаст ошибку, так как значение не соответствует правилу xs:enumeration.
2. **Если в поле Sku** передадут штрих-код из 10 цифр или добавят буквы (например, "4601234567"), система отклонит документ на этапе xs:pattern, предотвратив запись некорректных данных в базу 1С.

Вариант А. Реализация в формате JSON (Для REST API между ТСД и ИС)

```

json
{
  "$schema": "https://logist-pro.ru",
  "orderId": "ORD-2026-1009-0042",
  "orderType": "Inbound",
  "status": "In_Progress",
  "warehouseZone": "Receiving_Zone",
  "actors": [
    {
      "role": "Supplier",
      "name": "ООО Чистый Дом",
      "id": "7701234567"
    },
    {
      "role": "Operator",
      "name": "Иванов П.С.",
      "id": "EMP-041"
    }
  ],
  "orderLines": [
    {
      "lineNumber": 1,
      "sku": "4601234567890",
      "productName": "Средство для мытья посуды 0.5л",
      "expectedQty": 150.0,
      "acceptedQty": 148.0,
      "addressCell": "A-08-02-01"
    },
    {
      "lineNumber": 2,
      "sku": "4609876543210",
      "productName": "Стиральный порошок Автомат 3кг",
      "expectedQty": 50.0,
      "acceptedQty": 50.0,
      "addressCell": "B-03-01-02"
    }
  ]
}

```

Вариант Б. Реализация в формате XML (Для пакетного обмена с 1С:УТ)

```
xml
<?xml version="1.0" encoding="UTF-8"?>
<WarehouseOrder xmlns="https://logist-pro.ru"
    OrderId="ORD-2026-1009-0042"
    OrderType="Inbound"
    Status="In_Progress"
    WarehouseZone="Receiving_Zone">
    <Actors>
        <Actor Role="Supplier" Name="000 Чистый Дом" Id="7701234567" />
        <Actor Role="Operator" Name="Иванов П.С." Id="EMP-041" />
    </Actors>
    <OrderLines>
        <OrderLine LineNumber="1"
            Sku="4601234567890"
            ProductName="Средство для мытья посуды 0.5л"
            ExpectedQty="150.0"
            AcceptedQty="148.0"
            AddressCell="A-08-02-01" />
        <OrderLine LineNumber="2"
            Sku="4609876543210"
            ProductName="Стиральный порошок Автомат 3кг"
            ExpectedQty="50.0"
            AcceptedQty="50.0"
            AddressCell="B-03-01-02" />
    </OrderLines>
</WarehouseOrder>
```