

Архитектура как код на основе модели сущностей Catlair

Черкас Руслан Челединов Игорь

2025-04-19

Содержание

1	Введение	1
2	Определения	1
3	Теория	2
4	Практика	2
4.1	Первая сущность	3
4.2	Развитие модели	3
4.3	Контексты	4
4.4	Связи	4
4.5	Описание сущностей	6
5	Применимость	6
6	Резюме	6

Примечания

1. doi:10.5281/zenodo.14319493 https://doi.org/***
2. Постоянный адрес документа [ru]: <https://github.com/johnthesmith/catlair-archcode/blob/main/export/catlair-archcode-ru.pdf>
3. Постоянный адрес документа [en]: <https://github.com/johnthesmith/catlair-archcode/blob/main/export/catlair-archcode-en.pdf>
4. Статья опубликована под лицензией: <https://creativecommons.org/licenses/by-sa/4.0/> CC BY-SA 4.0

1 Введение

Статья предлагает концепцию описания архитектуры информационных систем в рамках минимального набора правил.

Концепция основана на использовании иерархии сущностей. Описание универсально и минималистично. Включает: индекс сущностей, контекстно-зависимые свойства и связи.

2 Определения

Концепция оперирует следующими специфическими понятиями:

1. Сущность — именованная абстракция, отображающая объект, понятие, роль или любое другое явление реального мира в информационной системе. Сущности образуют иерархии, могут иметь свойства и связываться друг с другом.

Далее по тексту сущность обозначается как **entity**, а множество сущностей — как **entities**.

2. Свойство сущности — описание сущности, определяющее его особенности, характеристики. Примером свойства сущности в архитектуре можно назвать наименование, описание, количественные показатели объема и тд.

Далее по тексту свойства обозначается как **prop**, а множество атрибутов — как **props**.

3. Связи сущностей — средство определения зависимости или взаимодействия между сущностями.

Далее по тексту связи как **link**, а множество связей — как **links**.

4. Контексты — специфические для различных участников и ситуаций взгляды на свойства и связи сущностей, в зависимости от окружения. Примером контекста может выступать язык описания, различия в восприятии сущности со стороны бизнес логики и разработчика, уровень детализации схем.

Далее по тексту контекст обозначается как **context**, а их множество как **contexts**.

3 Теория

В основе концепции лежат следующие принципы:

1. Любое понятие в информационной системе следует описывать как сущность.
2. Для каждой сущности необходимо минимально определить:
 - факт существования;
 - тип сущности.
3. Все иные описания и свойства сущности следует признать вторичными и зависящими от различных контекстов, а именно:
 - связь сущностей между собой;

- атрибуты или свойства сущности.
4. Концепция придерживается принципа разделения ответственности, а потому отделяет описание сущностей и их связей от реализации действий с ними и интерпретации в тех или иных целях.

4 Практика

Для практических примеров применяется синтаксис YAML.

4.1 Первая сущность

1. Для сущностей используется отношение кортежей в формате ключ-значение, где ключ — это идентификатор сущности, а значение — идентификатор её типа.
Это соответствует реляционному представлению с двумя атрибутами:
 - `id` - идентификатор сущности;
 - `type` - идентификатор типа сущности (который, в свою очередь является сущностью).
2. Поскольку тип сущности обязателен, а никакого типа еще не существует, первую сущность следует самотипизировать во множестве `entities`:

```
entities:
  entity: "entity"
```

Примечание: Самотипизирующийся кортеж создает новый домен. Их количество не ограничено, и каждый может развиваться независимо.

4.2 Развитие модели

1. Определим несколько сущностей, которые в дальнейшем будут использоваться при описании архитектуры.

```
entities:

  # Базовые сущности архитектуры

  # Компонент (сервисы, хранилища состояний и тд...)
  component: "entity"
  # Сервис как компонент
  service: "component"
  # База данных как компонент
```

```

db: "component"
# Агент обладающий возможностью действовать
agent: "entity"
# Определили пользователя, как агента
user: "agent"
# Определили клиента, как пользователя
client: "user"

# Добавим несколько специфичных для архитектуры сущностей:

# Определим сервис backend
my-backend: "service"
# Определим базу данных
my-db: "db"
# Определим внутреннего пользователя Алиса
alice: "user"
# Определим клиента Боб
bob: "client"

```

2. Таким образом последовательными декларациями описаны разнородные сущности, которые далее будут использованы.

4.3 Контексты

1. Описание сущности различается для участников с учетом окружения. Такие различия определяются контекстом.
2. Простейшим примером является описание сущности в человекочитаемом виде на различных языках.
3. Контекстом может быть не только язык, но любой специфический взгляд на сущности со стороны различных групп, отделов, подразделений, задач.
4. Добавим несколько контекстов.

```

entities:

# Контекст как сущность
context: "entity"
# Язык является контекстом
lang: "context"

# Русский язык
ru: "lang"

```

```
# Английский язык
en: "lang"
```

4.4 Связи

1. После добавления сущностей следует учитывать из связи между. Для этого применяется секция `links`.
2. Связи описываются как типизированное направления от одной сущности к другой, при этом тип так же является сущностью.
3. Так же для связей возможно указание множества контекстов, которые позволяют отобразить связи для различных ситуаций. Контекст является опциональным. В случае, если он не указан, возможно допущение применимости связи для любого контекста.
4. В общем виде связи могут быть представлены кортежем атрибутов:
 - `from` - сущность источник связи;
 - `to` - сущность направление связи;
 - `type` - сущность тип связи;
 - `context` - опциональный список контекстов, для которых связь актуальна;
 - `attr` - опциональный список специфичные атрибутов связи в формате ключ значение;
5. И так опишем некоторые связи сущностей:

```
entities:
  # Множество связей как сущность
  link: "entity"

# Определяем связи между сущностями в секции links
links:
  -
    # Определяем что mouse обладает правом select для БД
    from: "mouse"
    to: "service"
    link: "select"
    context:
      - "right"
  -
    # Кот может добавлять в базу данных
```

```

    from: "cat"
    link: "select"
    to: "my-db"
    context:
      - "right"
-
    from: "cat"
    link: "insert"
    to: "my-db"
    context:
      - "right"
-
    # Определяем что сервис подключается а БД
    from: "my-backend"
    to: "my-db"
    link: "connect"
-
    # Определяем что клиент подключается к сервису
    from: "client"
    to: "service"
    link: "connect"

```

6. Указанный способ описания может включать множество различных зависимостей включая техническую связь компонентов, иерархические структуры подчиненности, локацию размещений компонентов и прочее.

4.5 Описание сущностей

1. Далее для сущностей возможно описание специфичных свойств так же в разрезе контекстов.

5 Применимость

6 Резюме

Список литературы

- [1] EWD447 *On the role of scientific thought* <https://www.cs.utexas.edu/~EWD/transcriptions/EWD04xx/EWD447.html>
- [2] Simon Brown.
Software Architecture as Code.

<https://static.codingthearchitecture.com/presentations/saturn2015-software-architecture-as-code.pdf>

- [3] Cherkas R. Cheledinov I.
Архитектура сущностей Catlair.
<https://github.com/johnthesmith/scraps/blob/main/ru/entities.md>
- [4] Mark Richards, Neal Ford.
Is "Architecture as Code" the Future of Software Design?
<https://www.architectureandgovernance.com/applications-technology/is-architecture-as-code-the-future-of-software-design/>