Архитектура как код на основе модели сущностей Catlair

Черкас Руслан Челединов Игорь

2025-04-19

Содержание

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`

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

1. Сущность — именованная абстракция, которая отражает объект, понятие, роль или любое другое различимое явление в системе. Сущности образуют иерархии, могут иметь свойства и связываться друг с другом.

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

3 Теория

В основе концепции лежат следующие принципы:

1. Любое понятие в информационной системе следует описывать как сущность.
2. Для любой сущности неоходимо минимально определить:
   - факт существования;
   - тип сущности.
3. Все иные описания и свойства сущности следует признать вторичными и зависящими от различных контекстов и точек зрения, а именно:
   - связь сущностей между собой;
   - атрибуты или своуства сущности.
4. Концепция придерживается принципа разделения ответсвенности, а потому отделяет описание сущностей и их связей от реализации действий с ними и интерпритации в тех или иных целях.

4 Практика

Здесь и далее 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 огранично, и каждый может развиваться независимо.

4.2 Развитие модели

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

```
entities:

    # Определим базовые сущности архитектуры

    # Определили компонент
    component: "entity"
    # Определили контекст, как окружение или точка зрения
    context: "entity"
    # Определим сервис как компонент
    service: "component"
    # Определили хранилище состояний типа компонент
    state-storage: "component"
    # Определили очередь как хранилище состояний
    queue: "state-storage"
    # Определили базу данных как хранилище состояний
    db: "state-storage"
    # Определили агента, как сущность (Агент может действовать)
    agent: "entity"
    # Определили пользователя, как агента
    user: "agent"
    # Определили клиента, как агента
    client: "agent"
    # Определили клиента, как агента
    client: "agent"

    # Определили контексты, для примера различные языки
    lang: "contect"
    # Русский язык
    ru: "lang"
    # Английский язык
    en: "lang"
```

```
# Определили связи между сущностями
# right: entity
# select: right
# insert: right

# Добавим несколько специфичных для архитектуры сущностей:

# Определим сервис backend
my-backend: "service"
# Определим базу данных
my-db: "db"
# Определим внутреннего пользователя cat
cat: "user"
# Определим клиента mouse
mouse: "client"
```

4.3 Связи

1. После добавления сущностей мы можем описать разнородные связи между ними используя секцию 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 спекцифичные атрибутов связи в формате ключ значение;
5. И так опишем некоторые связи сущностей:

```
# Определяем связи между сущностями
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.

4.4 Описание сущностей

1. Далее для сущностей возможнос 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] Pavel Vlasov.
Architecture As Code.
https://medium.com/nasdanika/architecture-as-code-7c0eadfc0b2b

[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/