Архитектурный код в моделе 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. Влючает: 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, определяещее 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 неоходимо 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 ответсвенности, а потому отделяет описание сущностей и их связей от реализации действий с ними и интерпритации 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 огранично, 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 последоватльными 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 конекстом
    lang: "contect"
    # Планы todo являются контекстом
    todo: "context"
```

```
    # Описание asis является контекстом
    asis: "context"

    # Русский язык
    ru: "lang"
    # Английский язык
    en: "lang"
```

4.4 Связи

1. После добавления сущностей следует учитывать из взаимосвязи. Для этого применяется секция links.

2. Связи описываются как типизированное направления от одной сущности к другой, при этом тип так же является сущностью.

3. Для связей возможно указание множества контекстов, которые позволяют отобразить связи для различных ситуаций. Контекст является опциональным. Отсутсвие указания контекста интерпритируется на уровне представления.

4. В общем виде связи могут быть определены кортежем атрибутов, каждый из которых может содержать одну или более сущностей:

   - `from` - сущность источник связи;
   - `to` - сущность направление связи;
   - `type` - сущность тип связи;
   - `context` - опциональный список контекстов, для которых связь актуальна;
   - `tuple` - опциональный список спекцифичных атрибутов связи в формате ключ значение;

5. Опишем некоторые связи сущностей:

```
entities:
    # Связь как сущность
    link: "entity"

# Определяем связи между сущностями в секции links
links:
    -
        # 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"
-
    # Алиса может читать и добавлять данные
    from: "alice"
    link:
        - "select"
        - "insert"
    to: "my-db"
    context: "right"
-

    # Боб обладает правом select для сервиса в контексте прав
    # для концепта asis
    from: "bob"
    to: "service"
    link: "select"
    context:
        - "right"
        - "asis"
    # Планируется что Боб будет обладать правами создания и
    # добавления согласно концепта todo
    from: "bob"
    to: "service"
    link:
        - "select"
        - "insert"
    context:
        - "right"
        - "todo"
```

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.

2. В общем виде свойства сущностей моугт быть определены кортежем атрибутов:

   - `entity` - сущность для которой выполняется описание, может содержать множество;
   - `tuple` - список свойств сущности в формате ключ значение;
   - `context` - опциональный список контекстов, для которых выполняется описание;

3. Описание сущностей производится в секции props:

```
props:
    # Человекочитаемое описание для сущности
    -
        entity: "entity"
        context: "ru"
        tuple:
            name:"Сущность"
    # Описание Алисы вне контекста
    -
        entity: "alice"
        tuple:
            age:21
            weight:71
    # Описание свойств для Алисы и Боба на разных языках
    -
        entity: "alice"
        context: "ru"
        tuple:
            name:"Алиса"
    -
        entity: "alice"
        context: "en"
        tuple:
            name:"Alice"
    -
        entity: "bob"
        context: "ru"
        tuple:
            name:"Боб"
    -
        entity: "bob"
        context: "en"
        tuple:
            name:"Bob"
```

4. Аналогичным образом возможно описание любых свойств сущностей.

5 Применимость

1. Приведенные примеры показывают возможность использования модели сущностей Catlair для описания моделей архитектуры.
2. Нотация Catlair позволяет единообрзано описать перечнень объектов, компонентов из взаимосвязи, при этом все перечисленное может быть представлено с различных точек зрения, включая хронологические.
3. Модель применима для описания организационных структур, прав доступа, технических компонентов.
4. Модель может использовтся для единообразного формирования ER, BPMN, C4 диаграмм на всех уровнях, при этом соблюдается единообразие нотации, а формат представления определяется контекстом.

6 Приложение

Приложение содержит компактный листинг выше описанных примеров для общего восприятия.

```
entities:
    entity: "entity"
    component: "entity"
    service: "component"
    db: "component"
    agent: "entity"
    user: "agent"
    client: "user"
    my-backend: "service"
    my-db: "db"
    alice: "user"
    bob: "client"
    link: "entity"
links:
    -
        from: "my-backend"
        to: "my-db"
        link: "connect"
    -
        from: "client"
        to: "service"
```

```
    link: "connect"
-
    from: "alice"
    link:
        - "select"
        - "insert"
    to: "my-db"
    context: "right"
-
    from: "bob"
    to: "service"
    link: "select"
    context:
        - "right"
        - "asis"
    from: "bob"
    to: "service"
    link:
        - "select"
        - "insert"
    context:
        - "right"
        - "todo"
props:
    -
        entity: "entity"
        context: "ru"
        tuple:
            name:"Сущность"
    -
        entity: "alice"
        tuple:
            age:21
            weight:71
    -
        entity: "alice"
        context: "ru"
        tuple:
            name:"Алиса"
    -
        entity: "alice"
        context: "en"
        tuple:
            name:"Alice"
    -
        entity: "bob"
```

```
    context: "ru"
    tuple:
        name:"Боб"
-
    entity: "bob"
    context: "en"
    tuple:
        name:"Bob"
```

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/