Praktycznie o Domain Driven Design

Marcin Lesiński

```
procedure sample(contractId: String);
begin
    // task: JIRA-7661
    if contractId.StartsWith('UM/') then
    begin
        sendContract(contractId.Substring(3));
    end;
end;
```

Marcin Lesiński

Praktycznie o Domain Driven Design

W poprzednim odcinku

Od zera do unit testera!

Esencja wiedzy której potrzebujesz, żeby zacząć używać unit testów w swojej firmie. Na wykładzie krok po kroku zostanie przedstawiony proces pisania testów dla jednostki kodu. Dowiesz się również co to: Mock, TDD. Używane frameworki: DSharp, DUnitX

Marcin Lesiński

Za młodu chciał zostać paleontologiem, dziś nie wyobraża sobie życia bez programowania. Uwielbia gromadzić wiedzę i ćwiczyć nowe umiejętności, żeby tworzyć soft jeszcze lepiej. Poza pracą gra w gry role play, czyta o kosmosie i hoduje kraby.

Spring4D + wstrzykiwanie zależności = klucz do architektury

Wstrzykiwanie zależności oraz wykorzystanie kontenera zależności z biblioteki Spring4D. Zastosowanie wstrzykiwania w kontekście architektury aplikacji oraz przy budowanie zestawu testów jednostkowych. Wymagana znajomość programowania obiektowego.

Marcin Lesiński

Za młodu chciał zostać paleontologiem, dziś nie wyobraża sobie życia bez programowania. Uwielbia gromadzić wiedzę i ćwiczyć nowe umiejętności, żeby tworzyć soft jeszcze lepiej. Poza pracą gra w gry role play, czyta o kosmosie i hoduje kraby.

Testowanie jest bardzo proste

```
procedure TMyTestObject.ShouldRefund100Points_WhenMandatePaidNotInWednesday;
begin
    // given
    _citizen := TCitizen.Create(ID, RANK, TPoints(100), TMandateRatio.of(2));

    // when
    _citizen.PayAMandate(MONDAY);

    // then
    Assert.AreEqual(200, _citizen.points);
end;
```

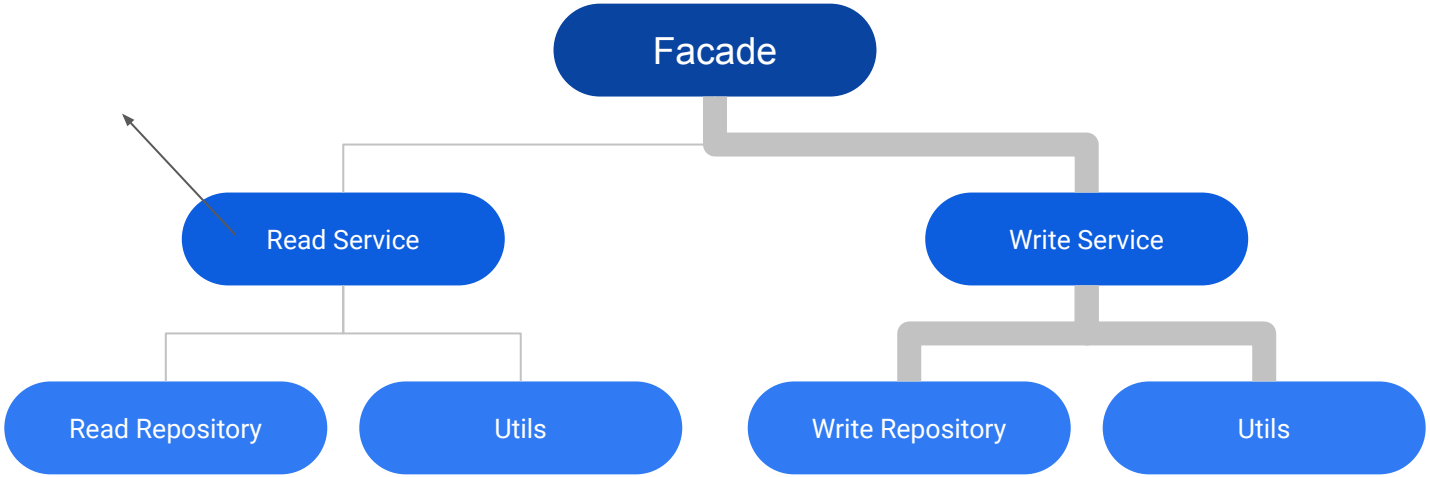
… gdy jednostki kodu są małe

```
function Mix(input: Integer):
Integer;
begin
    if input < 0 then
        Exit(0);

    if input > 100 then
        Exit(100);

    result := 100 - input;
end;
```

… i gdy można je wyodrębnić

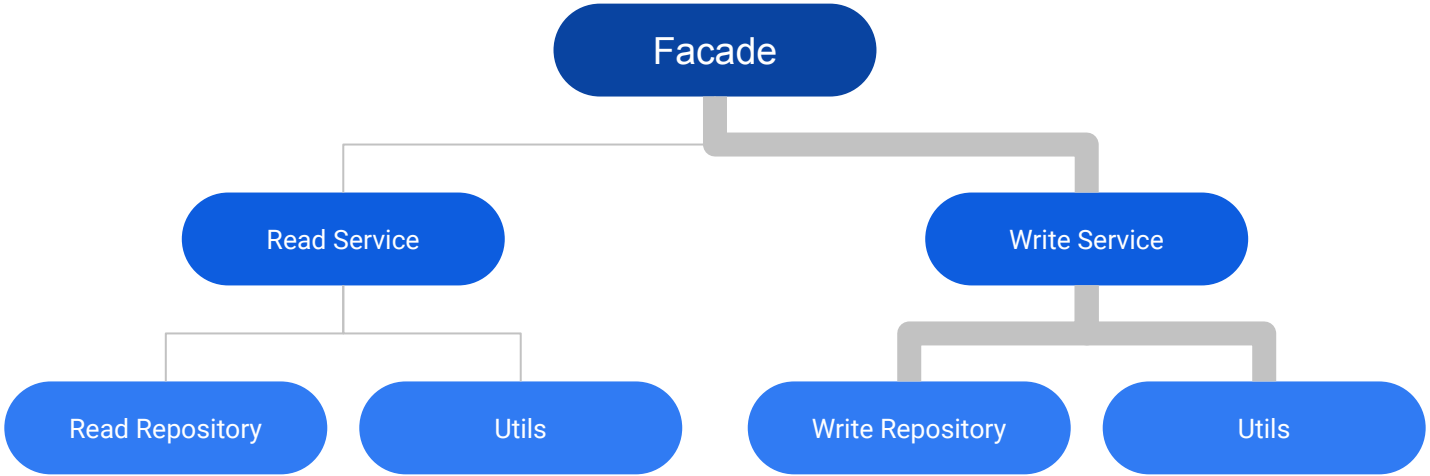


Dependency Injection - wstrzykiwanie zależności

```
constructor TCitizen.Create;
begin
    self._rank := TRank.Create();
    self._ratio := TRatio.Create();
end;

constructor TCitizen.Create(rank: TRank; ratio: TRatio);
begin
    self._rank := rank;
    self._ratio := ratio;
end;
```

```
CitizenForm.Inject(
    TCitizenService.Create(
        TCitizenRepository.Create()
    )
);
```



```
procedure sample(contractId: String);
begin
    // task: JIRA-7661
    if contractId.StartsWith('UM/') then
    begin
        sendContract(contractId.Substring(3));
    end;
end;
```

JIRA-7661

Kontrakty z prawidłowymi identyfikatorami należy wysłać używając skróconej nazwy.

```
procedure sample(ContractId: String);
begin
    var validContract := TContractId.Create(contractId);

    SendContract(validContract.shortForm);
end;
```

JIRA-7661

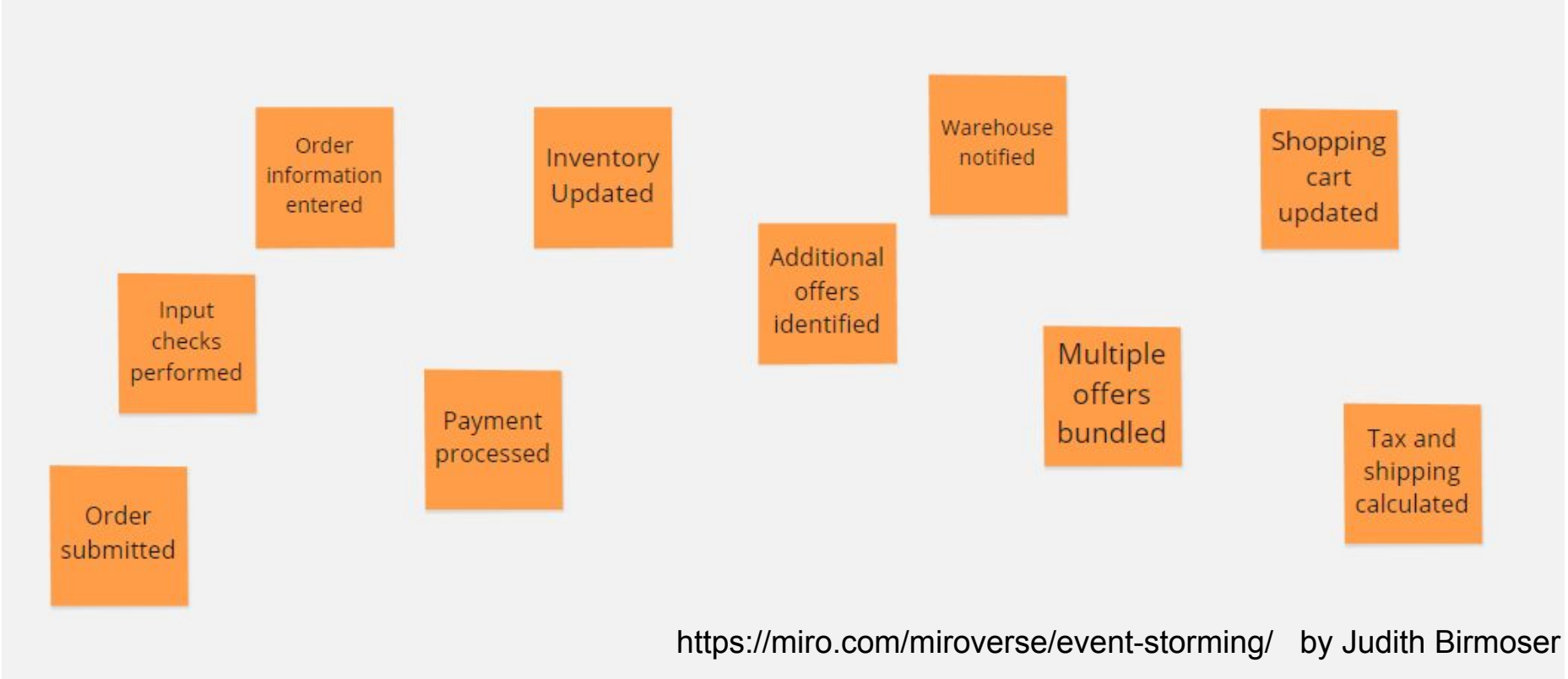
Kontrakty z prawidłowymi identyfikatorami należy wysłać używając skróconej nazwy.

Część praktyczna

Event storming

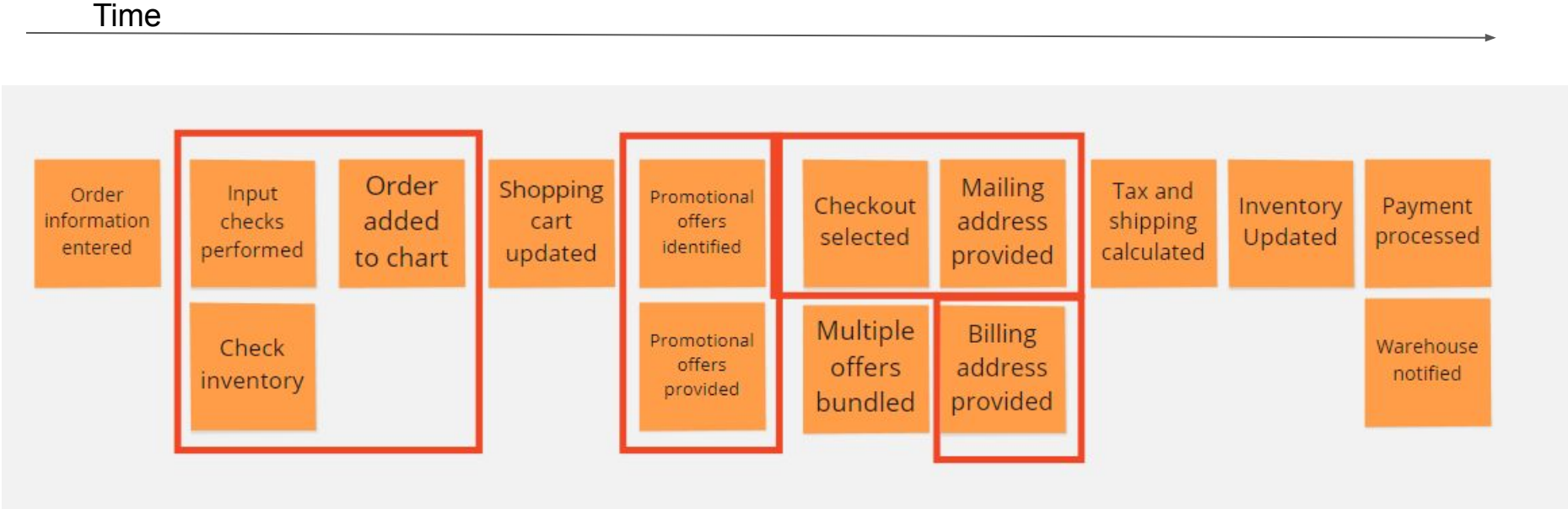
DDD - strategiczne

Event storming



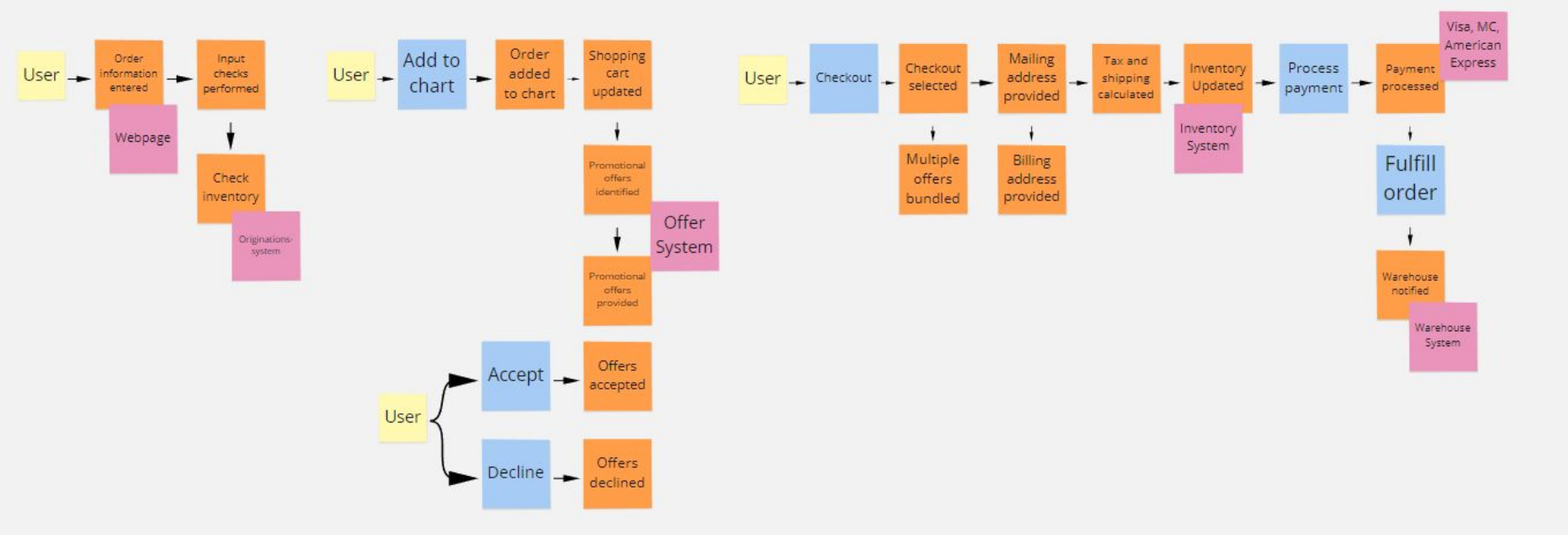
https://miro.com/miroverse/event-storming/ by Judith Birmoser

Event storming



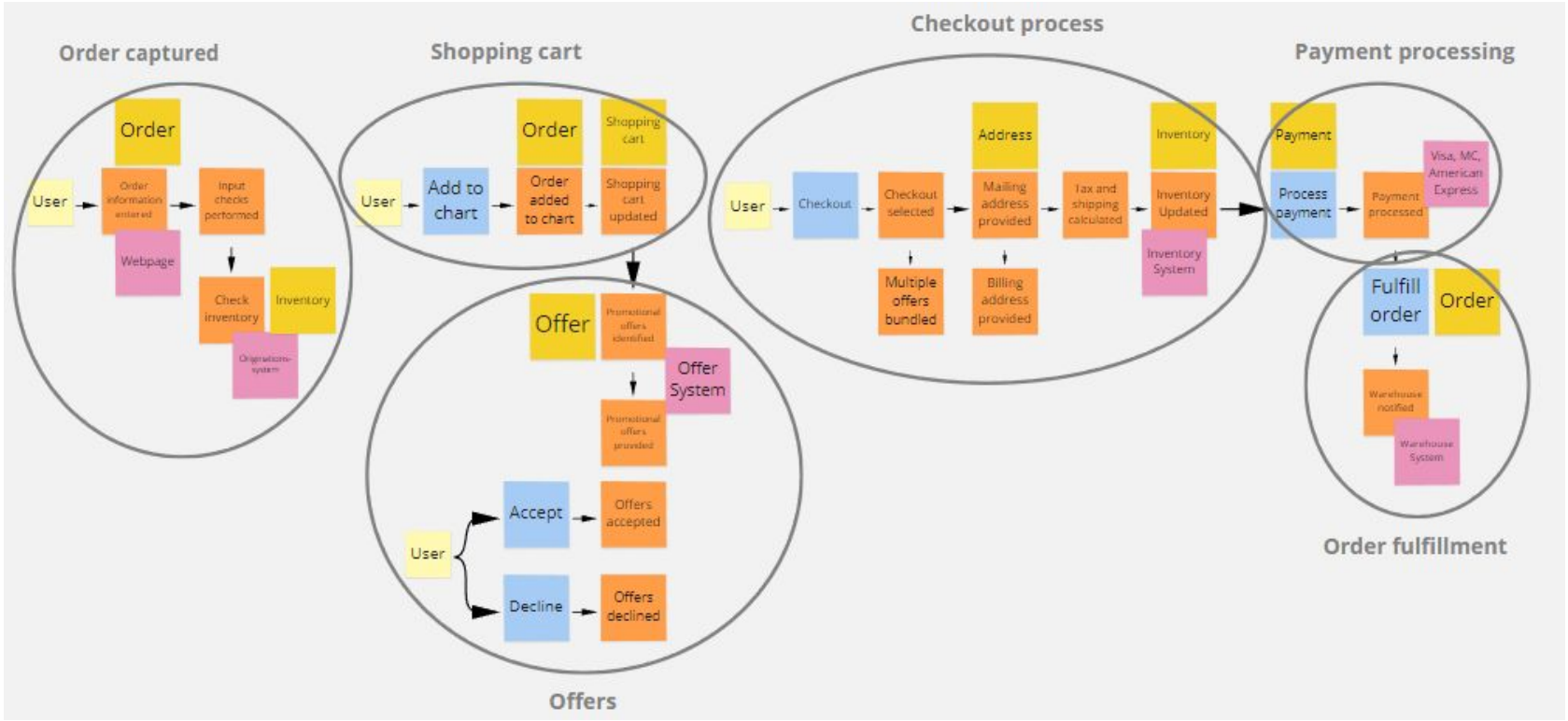
https://miro.com/miroverse/event-storming/ by Judith Birmoser

Event storming



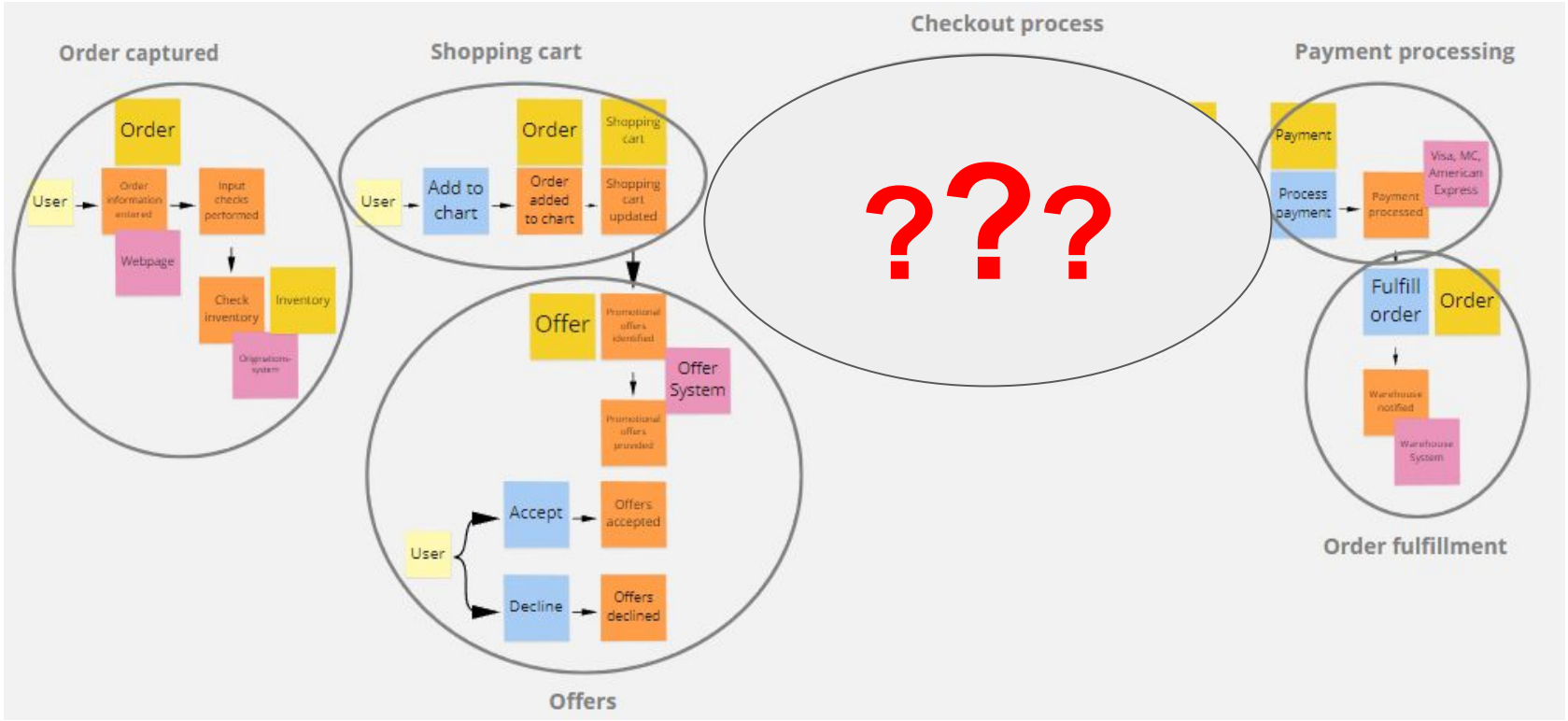
https://miro.com/miroverse/event-storming/ by Judith Birmoser

Event storming



https://miro.com/miroverse/event-storming/ by Judith Birmoser

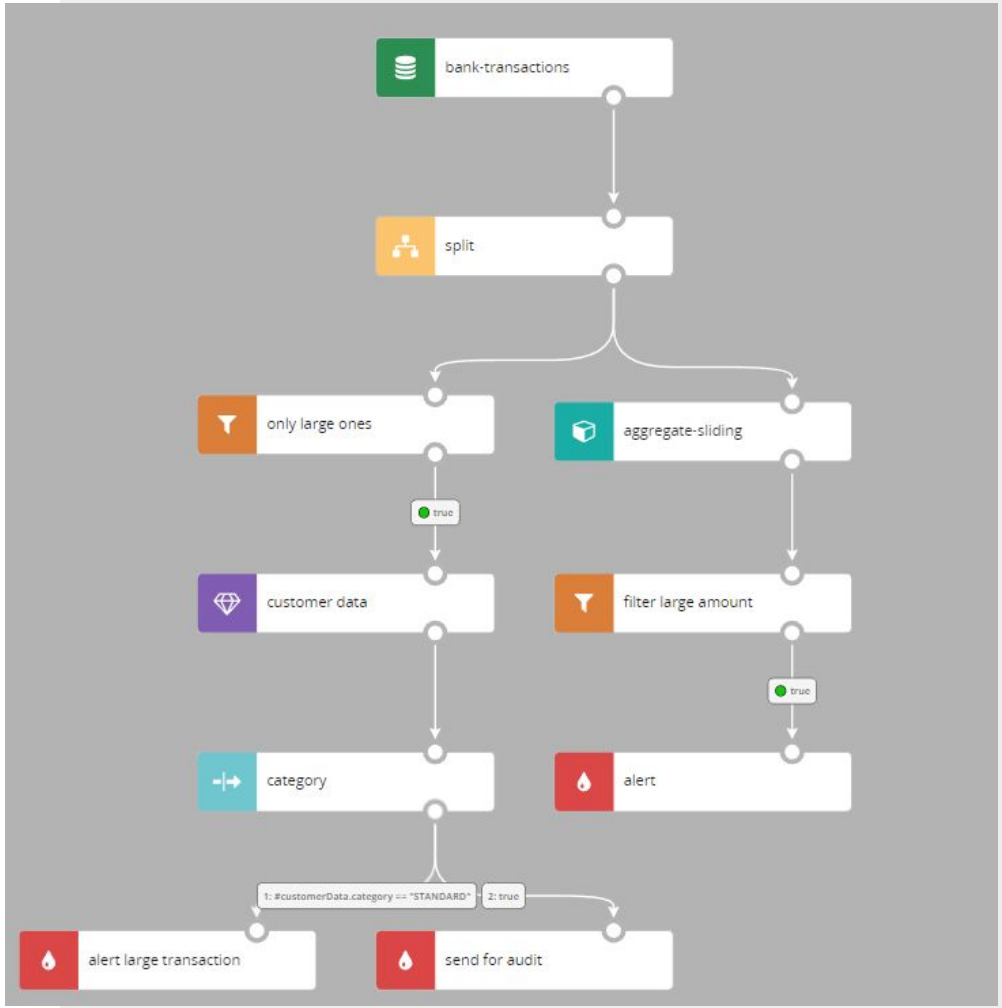
Event storming



https://miro.com/miroverse/event-storming/ by Judith Birmoser

Logika zarządzana przez biznes

demo.nussknacker.io



Nussknacker

nussknacker.io

- open source github.com/TouK/nussknacker
- tworzony przez software house
- wsparcie lub zlecenie wykonania integracji

Value Object

Wartość

```
TValueObject = record
private
    value: String;
public
    procedure Change();
    function Transform()
    : TOther;
end;
```

Entity

Wartość z tożsamością

```
TEntity = record
private
    id: Cardinal;
    value: String;
public
    procedure Change();
    function Transform()
    : TOther;
end;
```

Aggregate

- Generuje eventy
- Stanowi fasadę
- Granice transakcyjności

```
TAggregate = class
private
    value: TValue;
    entity: TEntity;
    //...
public
    function command(): TEvents;
end;
```

Service

Gdy logia nie pasuje do wartości

Nie zawiera stanu

```
TService = class
public
    function operation(value: TValue; entity: TEntity): TResult;
end;

TAggregate = class
private
    service: TService;
end;
```

Repository

Składuje i dostarcza

```
TRepository = class
private
    values: TList<TValue>;
public
    function findBests(): TList<TValue>;
    function findWorsts(): TList<TValue>;
end;
```

Factory

Wytwarza

```
TFactory = class
public
    function construct(foo: TFoo; bar: TBar): TValue;
end;
```

Policy

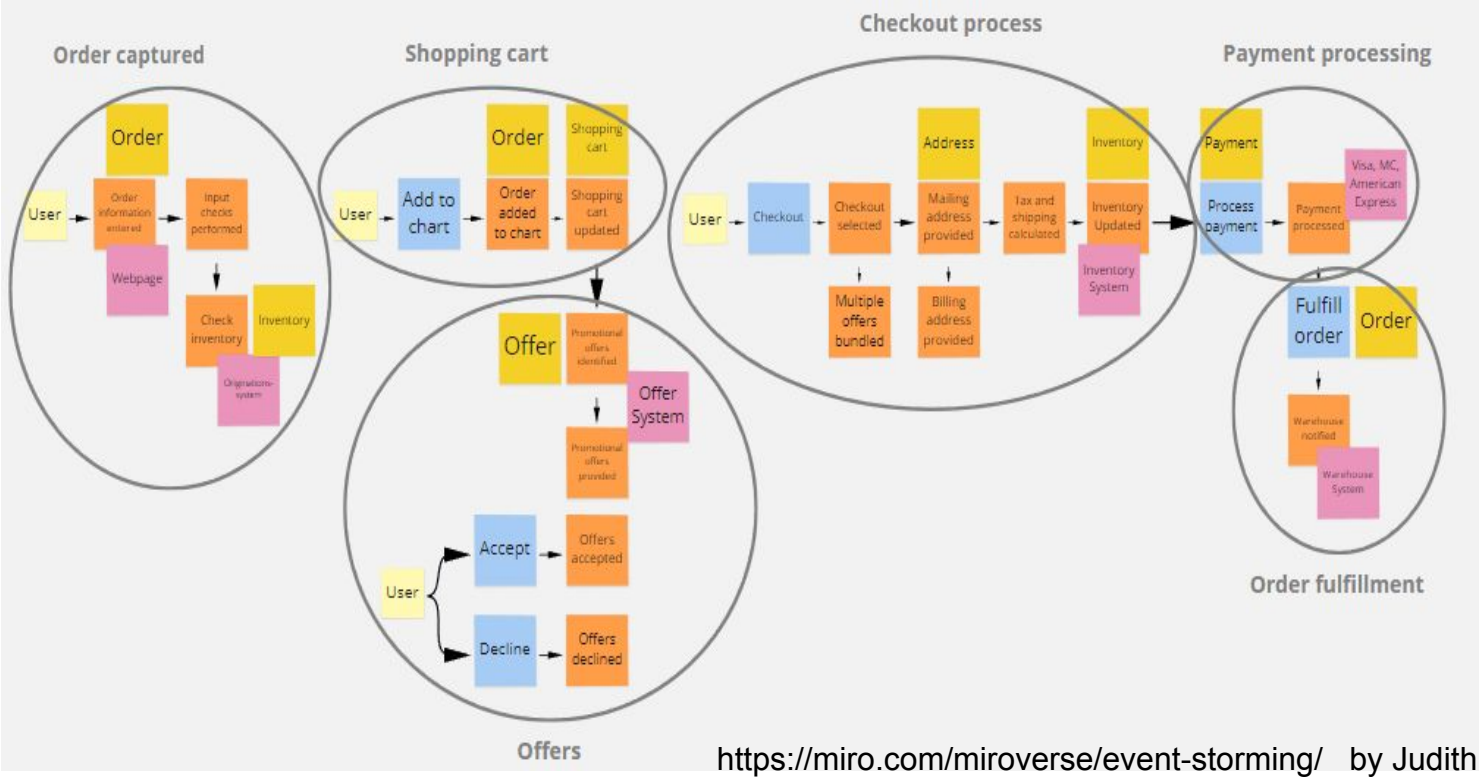
Strategia

```
IPolicy = interface
    procedure apply();
end;

TFooPolicy = class(TInterfacedObject, IPolicy)
    procedure apply();
end;

TBarPolicy = class(TInterfacedObject, IPolicy)
    procedure apply();
end;
```

Bounded Context



https://miro.com/miroverse/event-storming/ by Judith Birmoser

Praktycznie o DDD

Marcin Lesiński

- Testowanie
- Wstrzykiwanie zależności
- DSL
- Przykłady w kodzie
- Event storming
- Edytor graficzny logiki
- Elementy DDD