DDD dla architektów oprogramowania

Rozdziały III - IV

Agenda

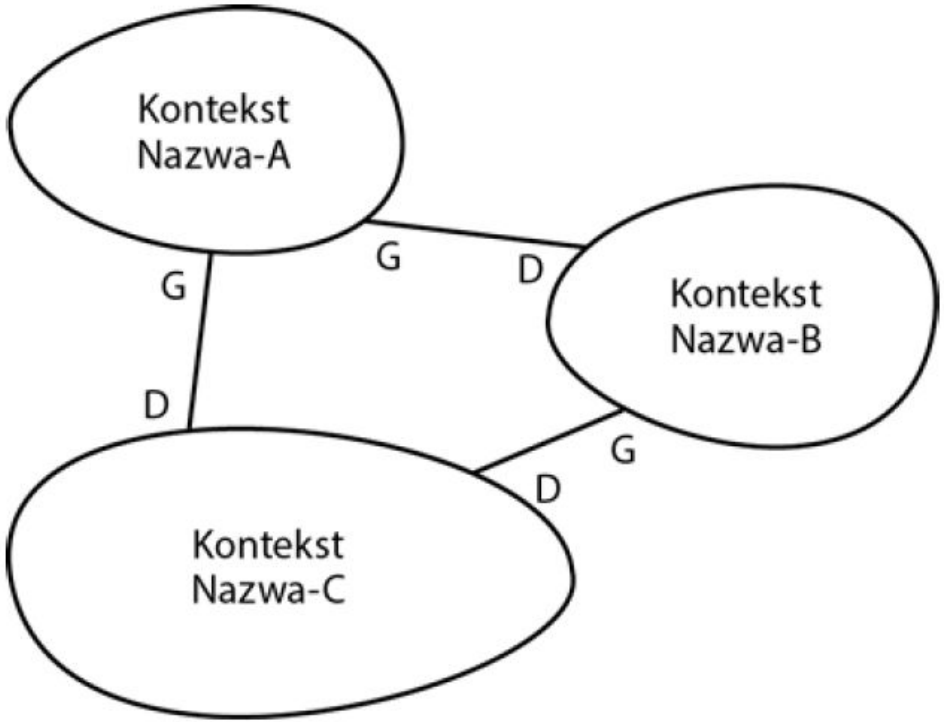
- Rozdział III - Mapy kontekstu
- Rozdiał IV - Architektura

Mapy Kontekstu

Dwa standardowe sposososoby opisu:

- Prosty diagram
- Kod źródłowy

Mapy kontekstu



Mapy kontekstu - wzorce integracji

- Partnerstwo
- Jądro współdzielone
- klient - dostawca
- Konformista
- Warstwa zapobiegająca uszkodzeniu
- Usługa otwartego hosta
- Język opublikowany
- Oddzielne drogi
- Wielka kula błota

Mapy kontekstu - wzorce integracji

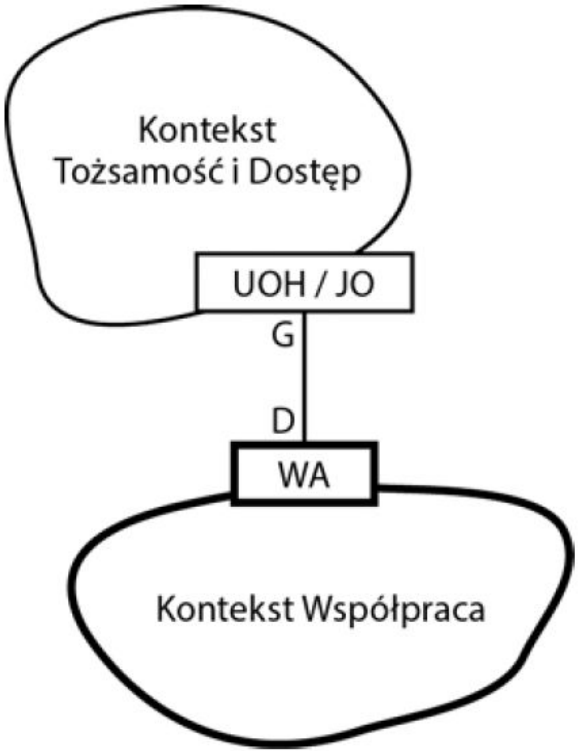
- Partnerstwo
- Jądro współdzielone
- klient - dostawca
- Konformista
- Warstwa zapobiegająca uszkodzeniu
- Usługa otwartego hosta
- Język opublikowany
- Oddzielne drogi
- Wielka kula błota

Co występuje w EB?

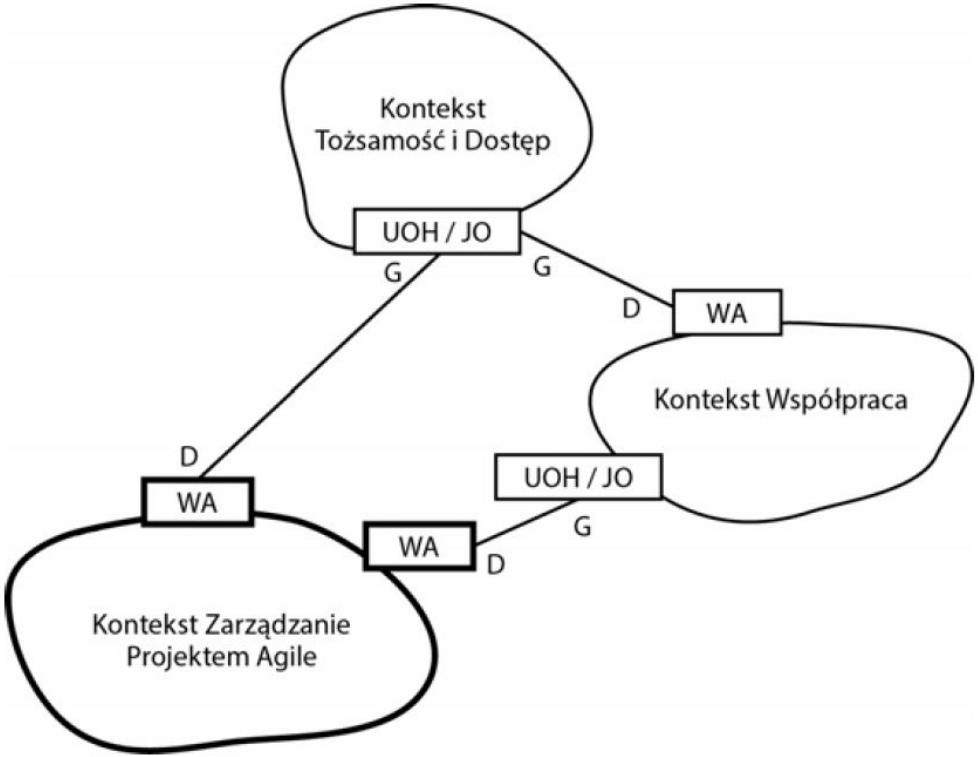
Mapy kontekstu - ProjectOvation

Mapy kontekstu - ProjectOvation

Mapy kontekstu - ProjectOvation



Mapy kontekstu - ProjectOvation



Mapy kontektu - ProjectOvation

Język Opublikowany

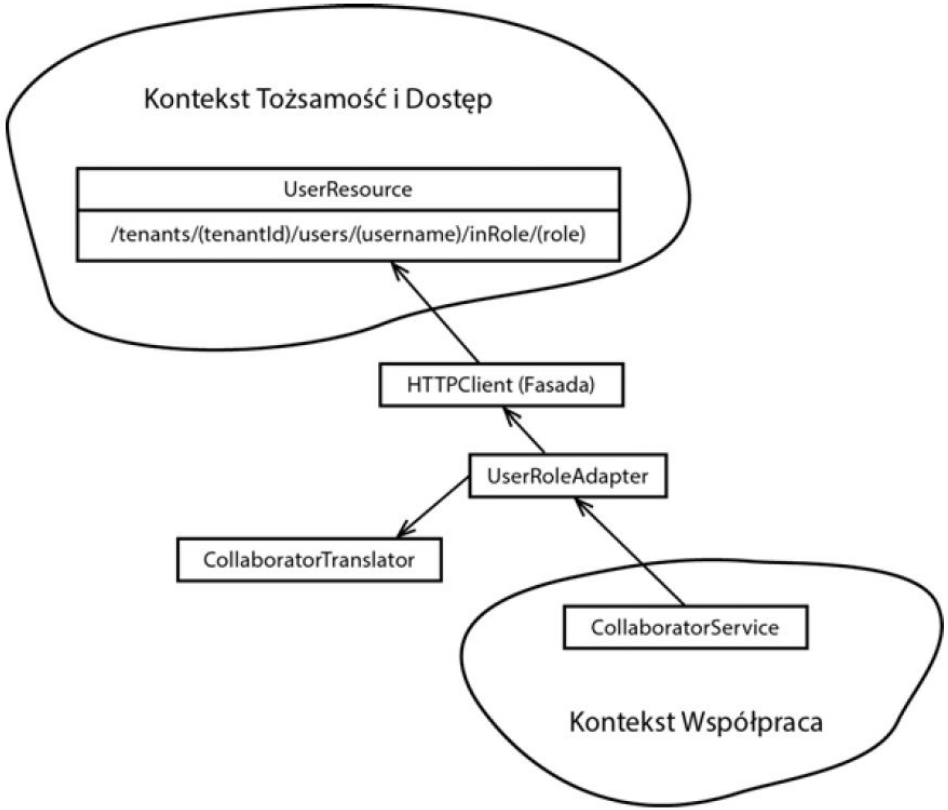
- Google Protocol Buffers
- SOAP
  - WSDL
- HTTP
  - WADL
  - HATEOAS

Mapy kontekstu - ProjectOvation

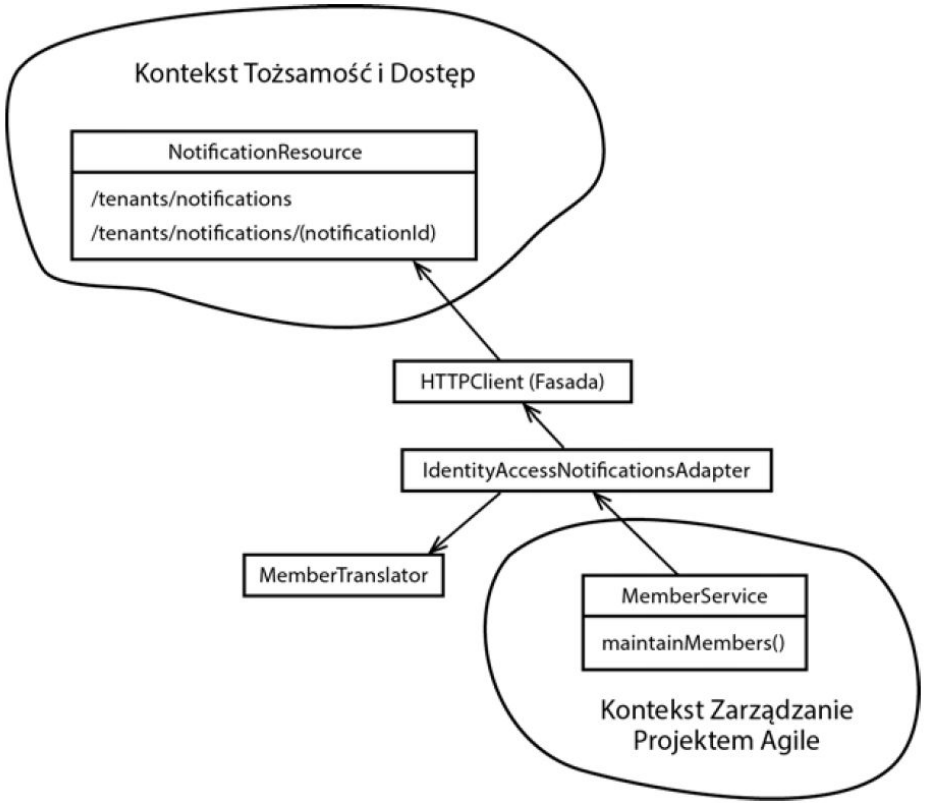
Warstwa Zapobiegająca Uszkodzeniu

- Translacja i walidacja
  - Np. Użytkownik <-> Moderator

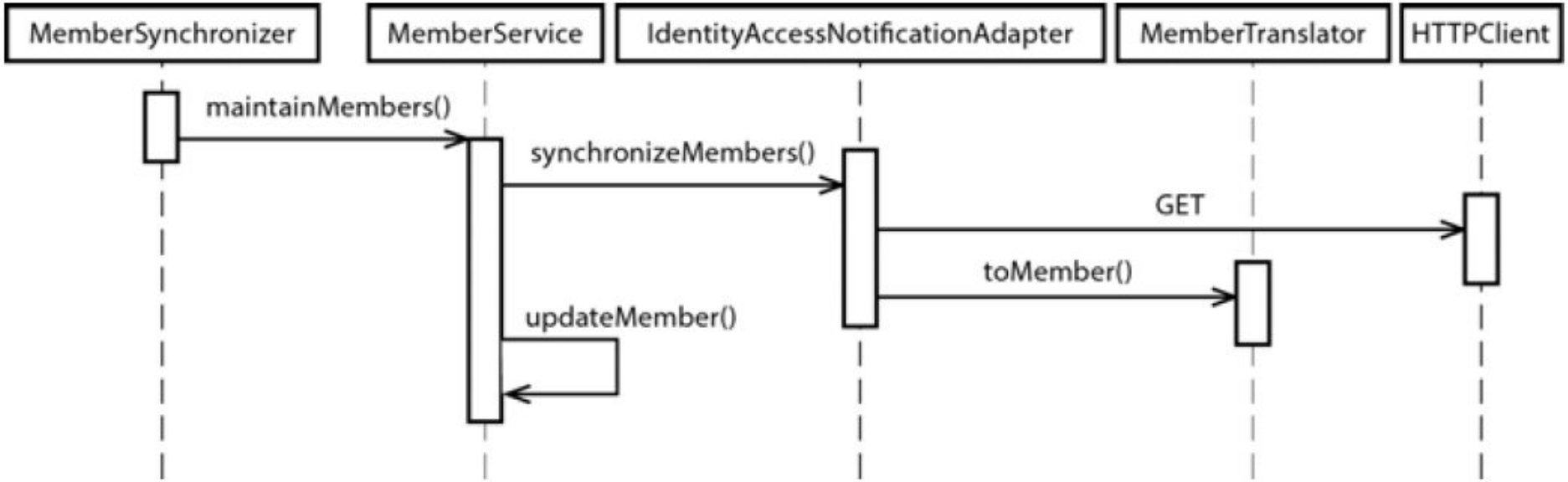
Mapy kontekstu - ProjectOvation



Mapy kontekstu - ProjectOvation



Mapy kontekstu - ProjectOvation

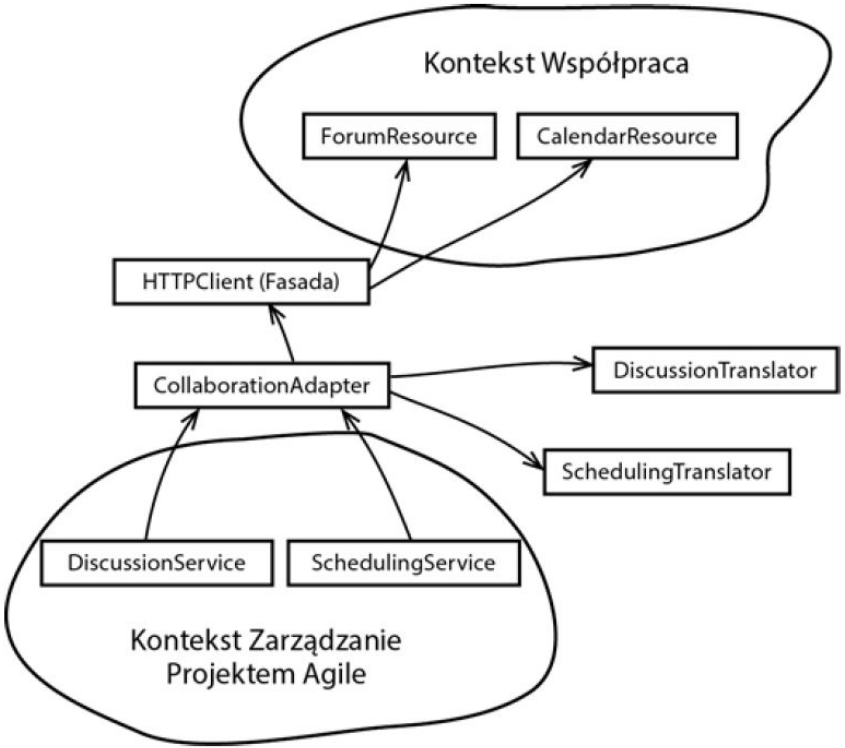


Mapy kontekstu - ProjectOvation

Obsługa niedostępności

```
public enum DiscussionAvailability {
    ADD_ON_NOT_ENABLED, NOT_REQUESTED, REQUESTED, READY;
}

public final class Discussion implements Serializable {
    private DiscussionAvailability availability;
    private DiscussionDescriptor descriptor;
    ...
}
public class Product extends Entity {
    ...
    private Discussion discussion;
    ...
}
```

Mapy kontekstu - ProjectOvation

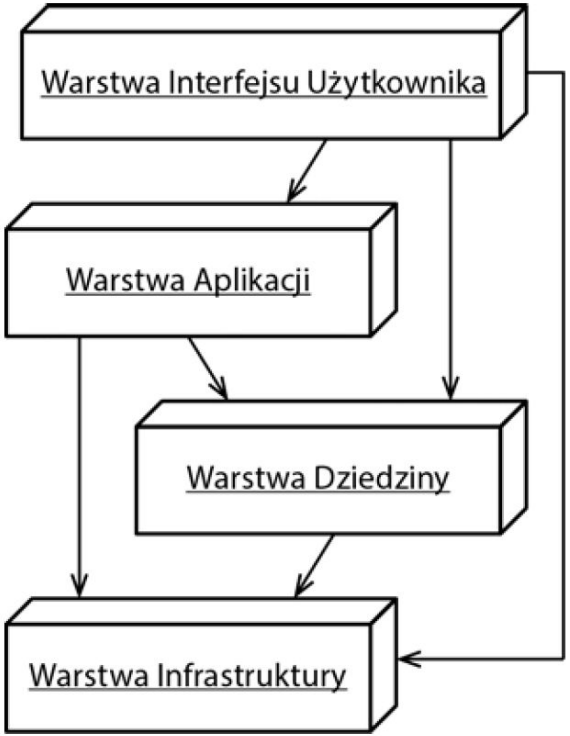


Architektura

Architektura powinna opowiadać o swoim czasie i miejscu,
ale tęsknić za ponadczasowością.

— Frank Gehry

Architektura - warstwy



Architektura - warstwy

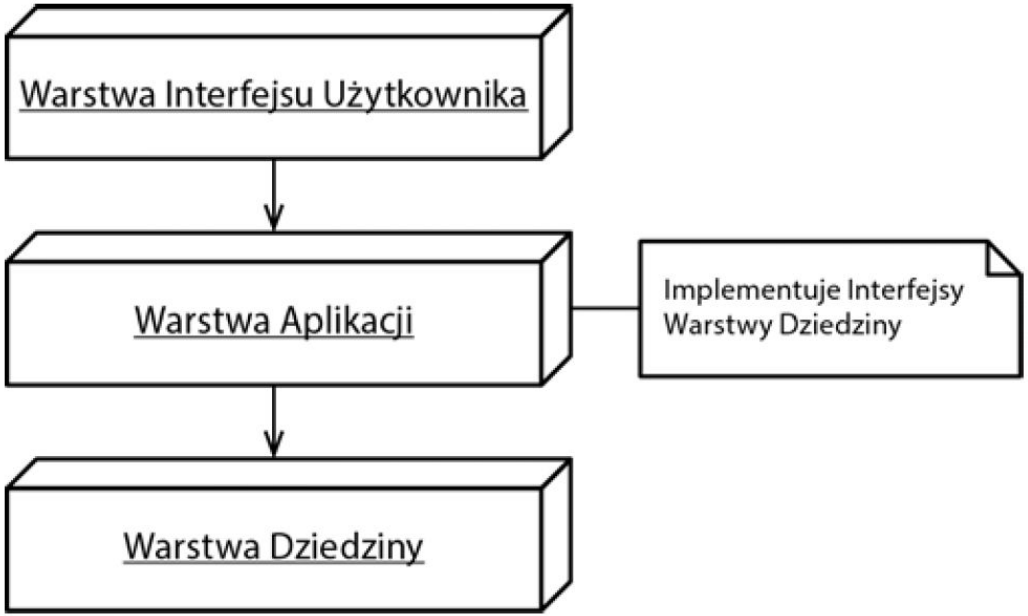
- Ścisła architektura warstw
- Złagodzona architektura warstw

Architektura - warstwy

Warstwa aplikacji

```
@Transactional
public void commitBacklogItemToSprint(
    String aTenantId, String aBacklogItemId, String aSprintId) {
    TenantId tenantId = new TenantId(aTenantId);

    BacklogItem backlogItem =
        backlogItemRepository.backlogItemOfId(
                tenantId, new BacklogItemId(aBacklogItemId));

    Sprint sprint = sprintRepository.sprintOfId(
                tenantId, new SprintId(aSprintId));

    backlogItem.commitTo(sprint);
}
```

Architektura - warstwy - problemy

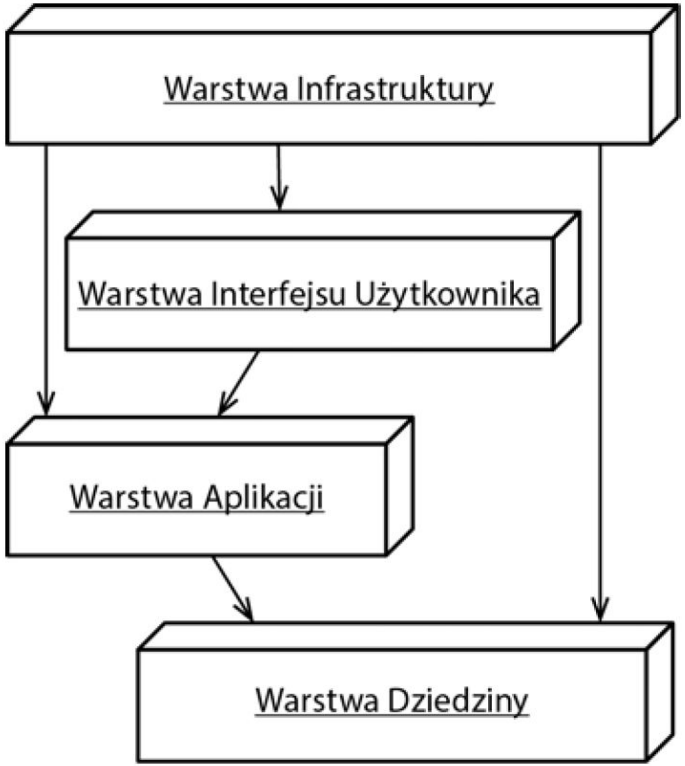


Architektura - warstwy - DIP

Moduły wysokiego poziomu nie powinny zależeć od modułów niskiego poziomu. Obie grupy powinny zależeć od abstrakcji.

Abstrakcje nie powinny zależeć od szczegółów. To szczegóły powinny zależeć od abstrakcji.

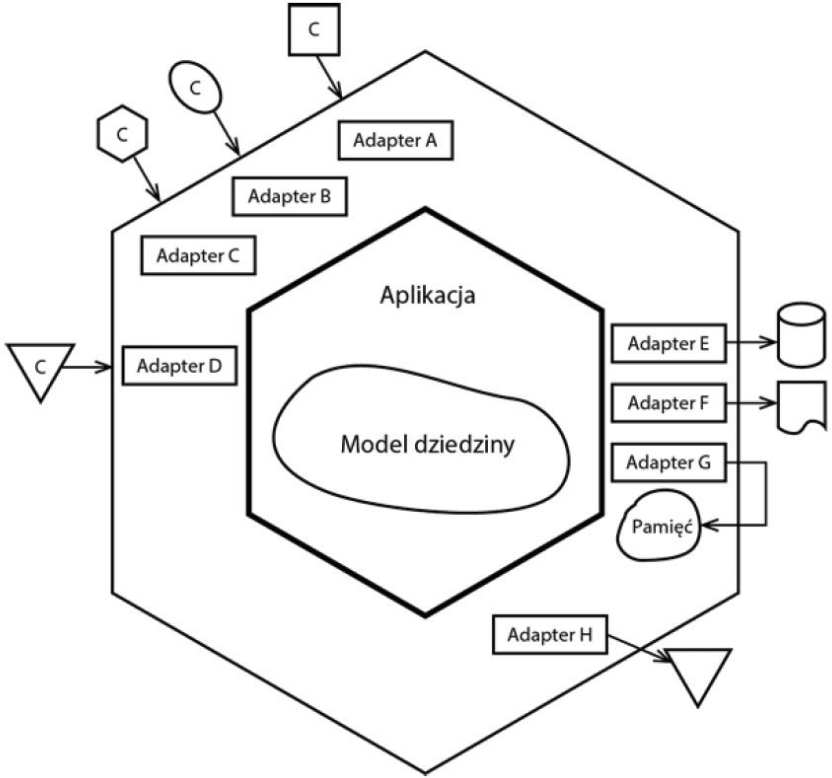
Architektura - warstwy - DIP



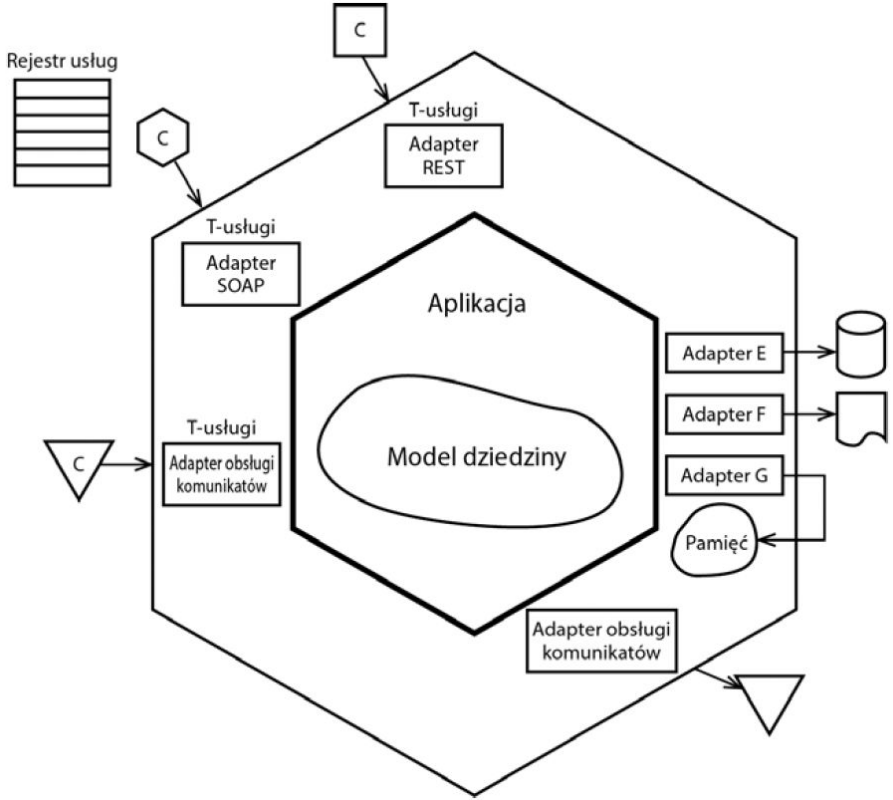
Architektura - warstwy - DIP

```
package com.saasovation.agilepm.infrastructure.persistence;

import com.saasovation.agilepm.domain.model.product.*;

public class HibernateBacklogItemRepository
    implements BacklogItemRepository {
    ...
    @Override
    @SuppressWarnings("unchecked")
    public Collection<BacklogItem> allBacklogItemsComittedTo(
        Tenant aTenant, SprintId aSprintId) {
        Query query =
```

Architektura - porty i adaptery



Architektura - SOA



Architektura - REST

- Zorientowanie na zasoby
- Bezstanowość
- HATEOAS jako język opublikowany

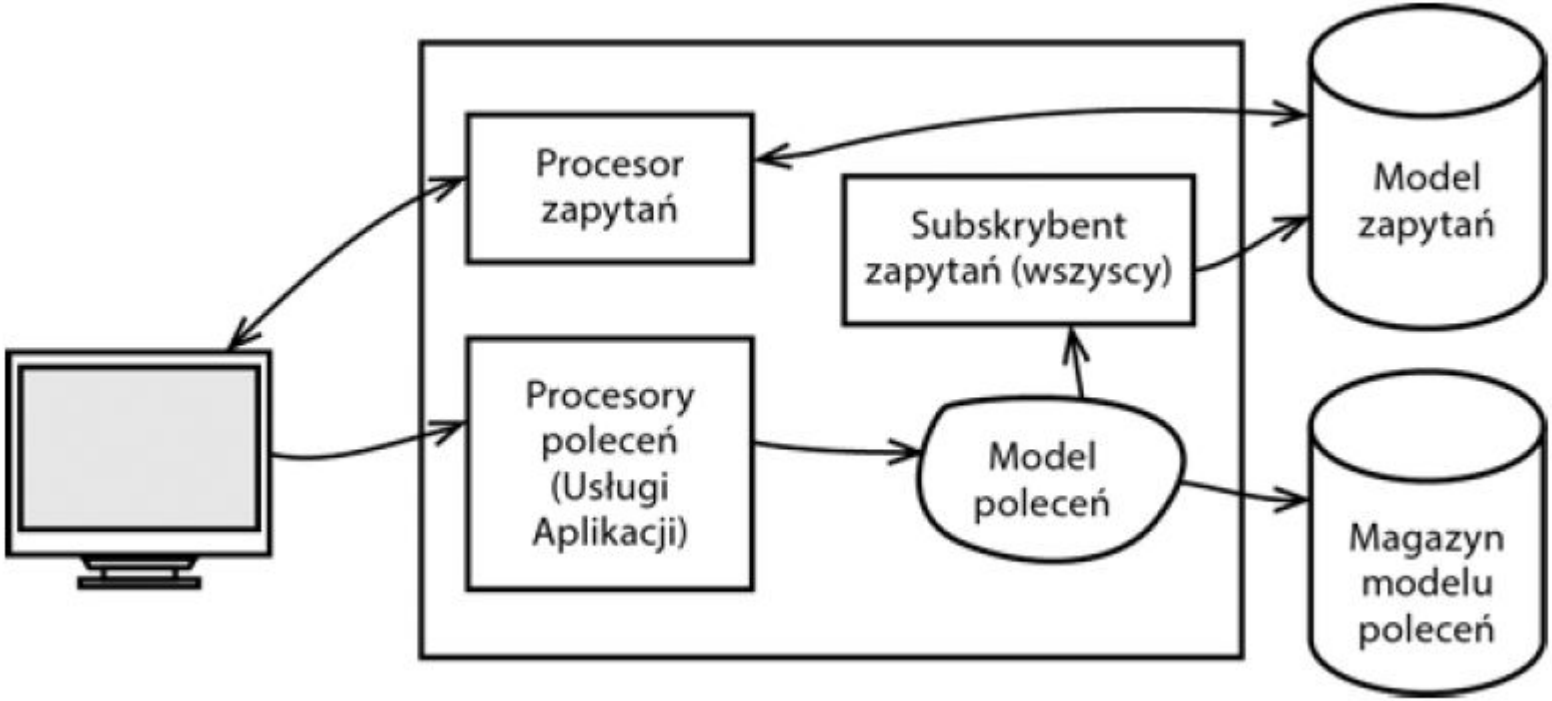
Dwa podejścia do zastosowania w DDD:

- Odseparowany kontekst ograniczony REST od modelu dziedziny (mapowanie)
- Model dziedziny projektowany pod interfejs, bazowanie na typie medium (?)

Architektura - CQRS

Każda metoda powinna być albo poleceniem, które wykonuje określone działanie, albo zapytaniem, które zwraca dane do wywołującego, ale nie może być i jednym, i drugim jednocześnie.

Architektura - CQRS



Architektura - CQRS

Model odczytu

- tabele/widoki z wybranymi danymi dla ról (wbudowane bezpieczeństwo)
- Baza dedykowana do odczytu

Architektura - CQRS

Model zapisu/poleceń:

- Procesory poleceń

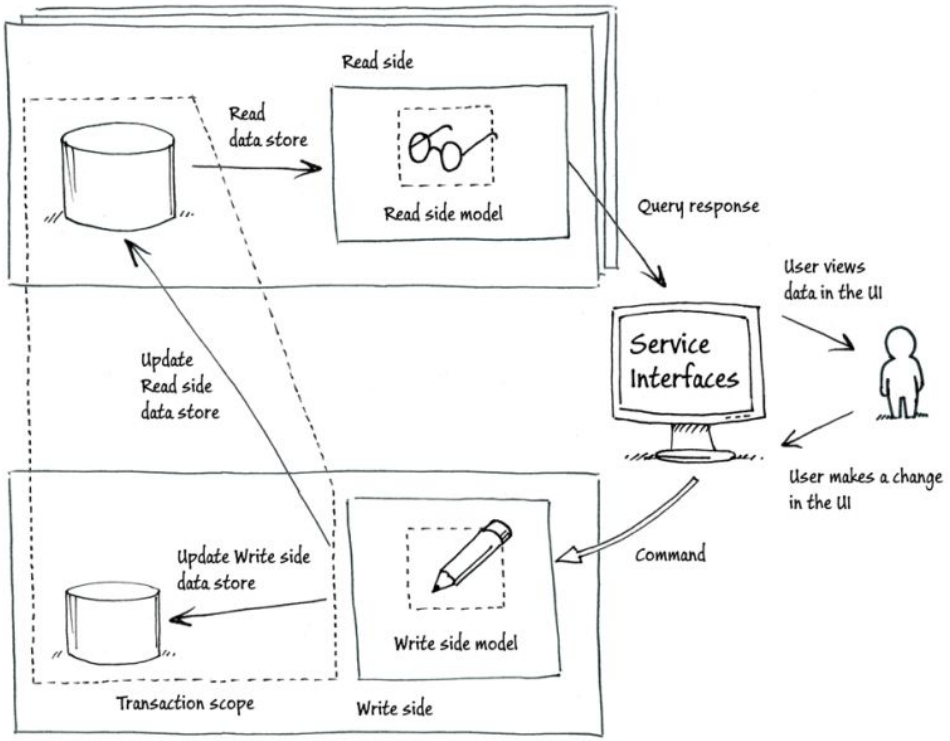
```
@Transactional
public void commitBacklogItemToSprint(
    String aTenantId, String aBacklogItemId, String aSprintId) {
    TenantId tenantId = new TenantId(aTenantId);

    BacklogItem backlogItem =
        backlogItemRepository.backlogItemOfId(
            tenantId, new BacklogItemId(aBacklogItemId));

    Sprint sprint = sprintRepository.sprintOfId(
            tenantId, new SprintId(aSprintId));

    backlogItem.commitTo(sprint);
}
```

Architektura - CQRS

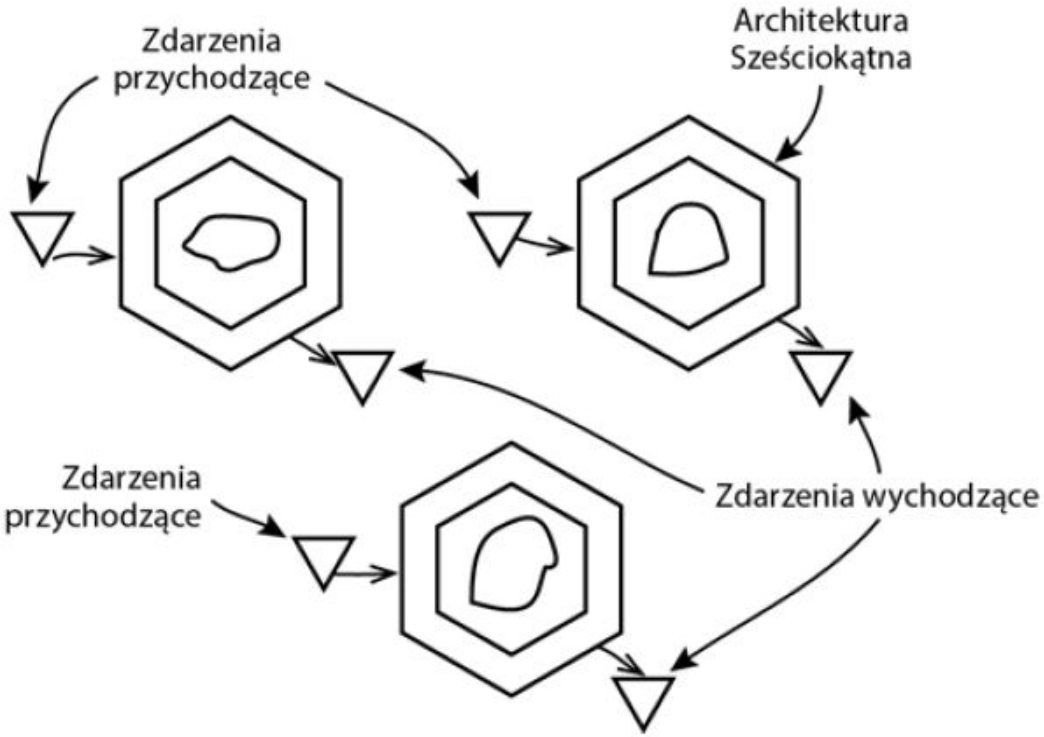
Model zapisu - publikowanie zdarzenia dziedziny

```
public class BacklogItem extends ConcurrencySafeEntity {
    ...
    public void commitTo(Sprint aSprint) {
        ...
        DomainEventPublisher
            .instance()
            .publish(new BacklogItemCommitted(
                    this.tenant(),
                    this.backlogItemId(),
                    this.sprintId()));
    }
    ...
}
```

Architektura - CQRS

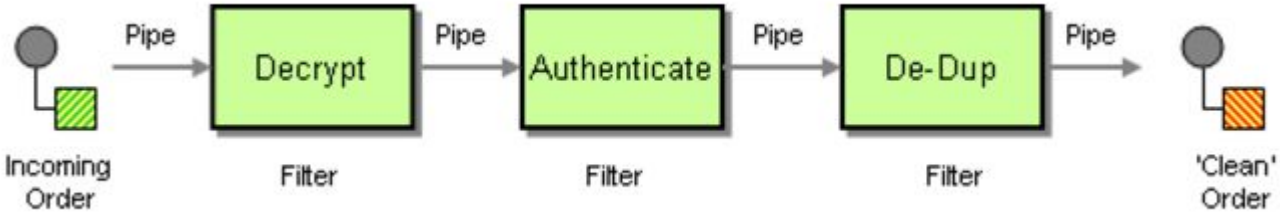


Architektura - EDA



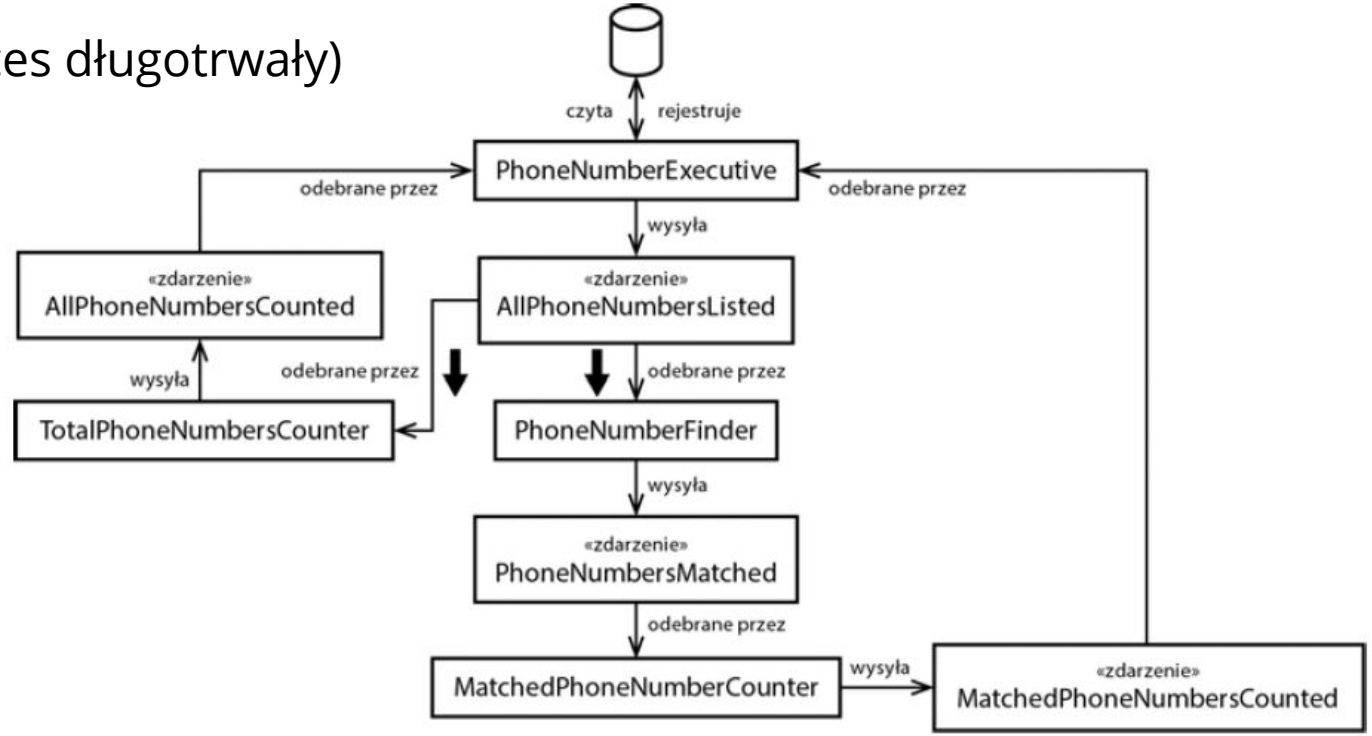
Architektura - EDA

Potoki i filtry



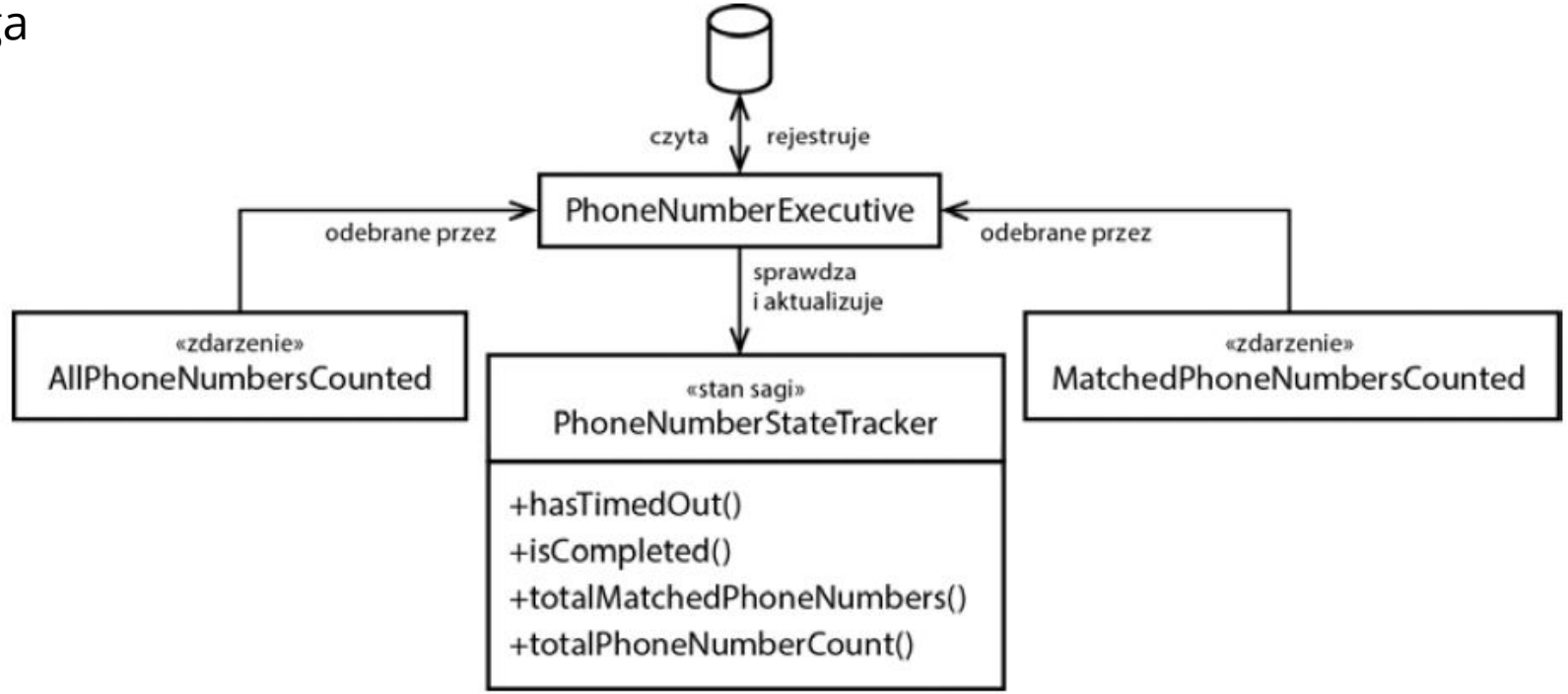
Architektura - EDA

Saga (Proces długotrwały)



Architektura EDA

Saga



Architektura - EDA - Event Sourcing

Architektura - EDA

Data Fabric i przetwarzanie rozproszone