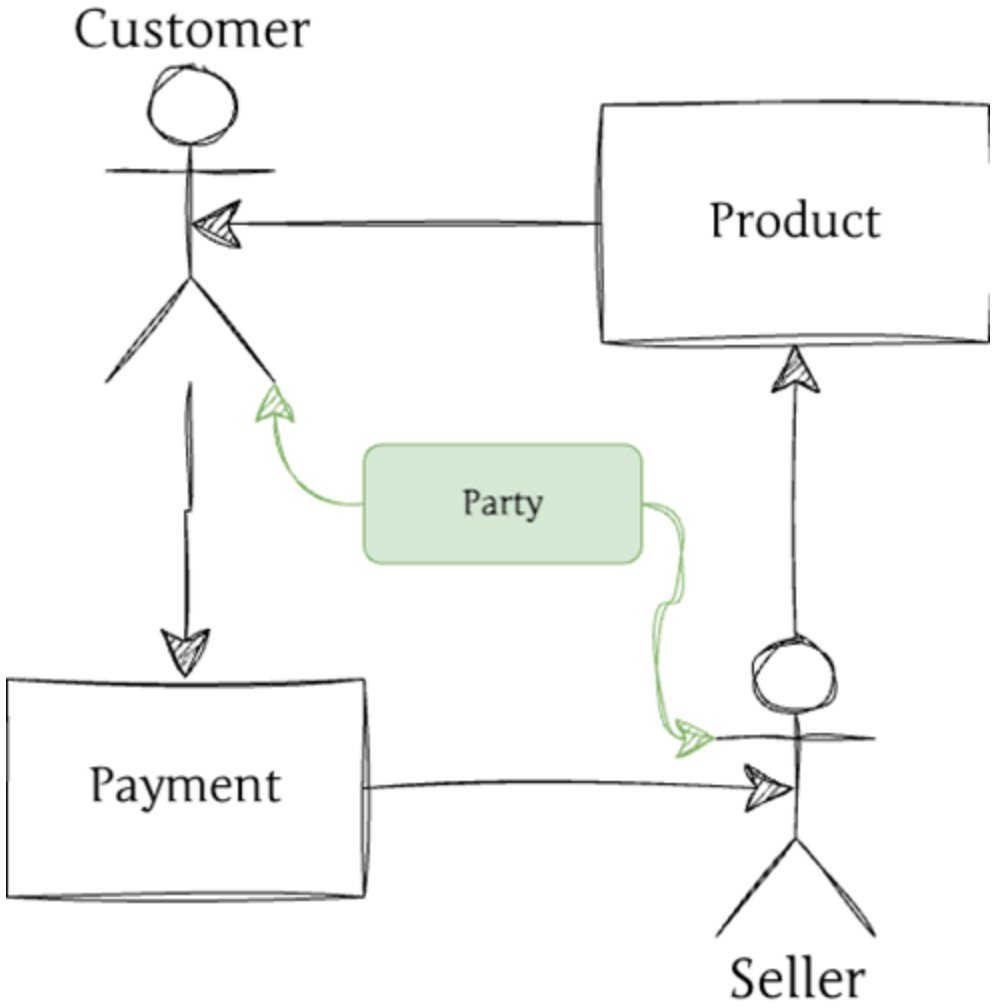

Customer
Product
Party
Payment
Seller

Jakub Ciszak

Senior Software Developer

BurdaMedia Polska

https://www.linkedin.com/in/jakub-ciszak/

developers ŁÓDŹ

Ktoś już to wymyślił

Modelowanie z użyciem archetypów biznesowych.

Plan

1. Co to jest archetyp?
2. Czym są archetypy biznesowe?
3. Kilka problemów do rozwiązania
    a. ceny uzależnione od różnych czynników
        - Money, Price
    b. rozbudowa systemu magazynowego
        - Inventory
    c. zarządzanie rolami
        - Party, PartyRelationship
4. Podsumowanie
5. QA

Co to jest archetyp?

Słownik Języka Polskiego

- pierwotny wzór postaci, obiektu lub zjawiska;
- schemat zachowania i postrzegania świata nieświadomie powielany przez ludność

Co to jest archetyp biznesowy?

Enterprise Patterns and MDA: Building Better Software with Archetype Patterns and UML
Jim Arlow, Ila Neustadt, December 22, 2003

"A business archetype is a primordial thing that occurs consistently and universally in business domains and business software systems."

...

„Archetyp biznesowy to pierwotny element, który występuje konsekwentnie i uniwersalnie w domenach biznesowych i systemach oprogramowania biznesowego."

Co to jest archetypowy wzorzec biznesowy?

Enterprise Patterns and MDA: Building Better Software with Archetype Patterns and UML
Jim Arlow, Ila Neustadt, December 22, 2003

"A business archetype pattern is a collaboration between business archetypes that occurs consistently and universally in business environments and software systems."

...

„Archetypowy wzorzec biznesowy to współpraca między archetypami biznesowymi, która występuje konsekwentnie i uniwersalnie w środowiskach biznesowych i systemach oprogramowania."

Kilka problemów

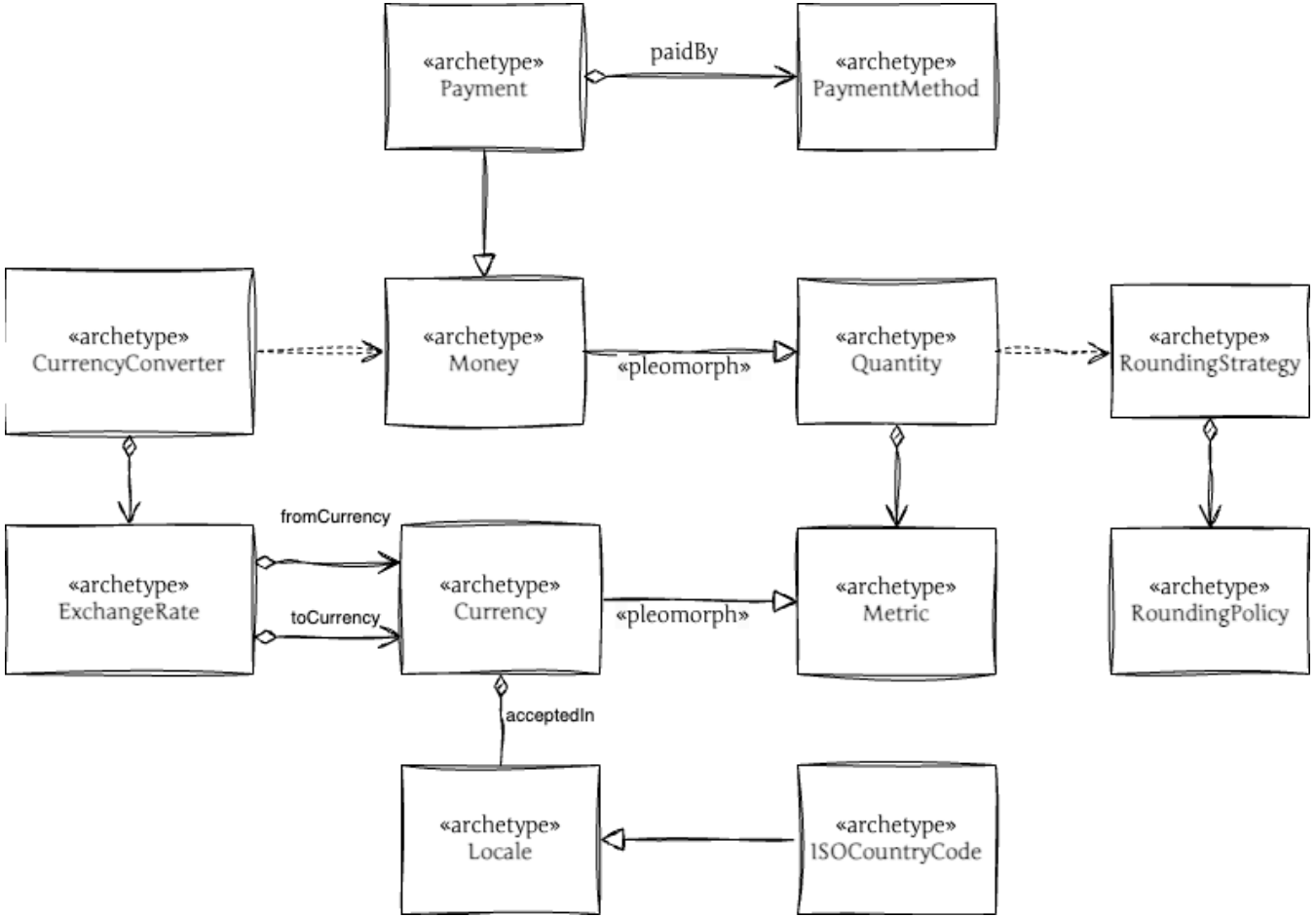
Ceny uzależnione od różnych czynników

Money Price

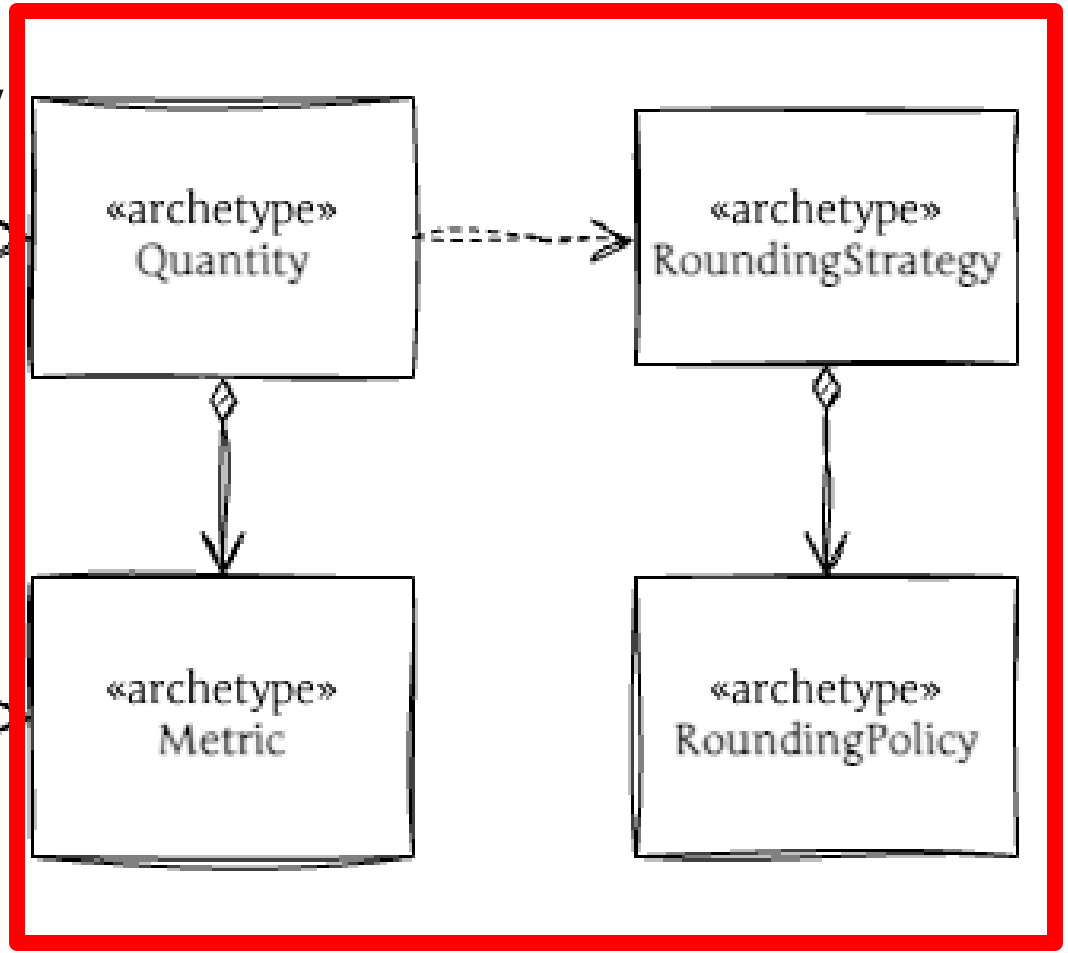
Ceny uzależnione od różnych czynników

Money Price

- Produkt może posiadać wiele cen
- Wybór odpowiedniej ceny jest uzależniony od różnych **zmiennych** czynników
- Możliwość implementacji przelicznika walut


«archetype»
Payment
paidBy
«archetype»
PaymentMethod
«archetype»
CurrencyConverter
«archetype»
Money
«pleomorph»
«archetype»
Quantity
«archetype»
RoundingStrategy
fromCurrency
«archetype»
ExchangeRate
toCurrency
«archetype»
Currency
«pleomorph»
«archetype»
Metric
«archetype»
RoundingPolicy
acceptedIn
«archetype»
Locale
«archetype»
ISOCountryCode

Quantity


«archetype»
Quantity
«archetype»
RoundingStrategy
«archetype»
Metric
«archetype»
RoundingPolicy

```php
readonly class Quantity
{
    public function __construct(
        protected float $amount,
        protected Metric $metric,
        protected RoundingPolicy $roundingPolicy
    ) {
    }

    public function amount(): float
    {
        return $this->amount;
    }

    public function add(Quantity $quantity): static{...}
    public function subtract(Quantity $quantity): static{...}
    public function multiply(float|Quantity $multiplier): static{...}
    public function divide(float|Quantity $divisor): static{...}
    public function isGreaterThan(Quantity $quantity): bool{...}
    public function isLessThan(Quantity $quantity): bool{...}
    public function isEqualTo(Quantity $quantity): bool{...}
    private function checkMetric(Quantity $quantity): void{...}
    private function fromNumber(float $number): static{...}
    public function value(): float
    {
        return $this->roundingPolicy->round($this->amount);
    }

    public function symbol(): string
    {
        return $this->metric->symbol();
    }
}
```

```php
interface Metric
{
    public function name(): string;
    public function symbol(): string;
    public function definition(): string;
    public function equals(Metric $other): bool;
}
```

```php
final readonly class RoundingPolicy
{

    public function __construct(
        public RoundingStrategy $strategy,
        public int $numberOfDigits = 2,
        public int $roundingDigit = 5
    ){
    }

    public function round(float $value): float
    {
        return match ($this->strategy) {
            RoundingStrategy::ROUND_UP => $this->roundUp($value),
            RoundingStrategy::ROUND_DOWN => $this->roundDown($value),
            default => $this->roundDefault($value),
        };

    }

    private function roundUp(float $value): float{...}

    private function roundDown(float $value): float{...}

    private function roundDefault(float $value): float{...}

}
```

```php
public function amount(): float
{
    return $this->amount;
}

public function add(Quantity $quantity): static{...}
public function subtract(Quantity $quantity): static{...}
public function multiply(float|Quantity $multiplier): static{...}
public function divide(float|Quantity $divisor): static{...}
public function isGreaterThan(Quantity $quantity): bool{...}
public function isLessThan(Quantity $quantity): bool{...}
public function isEqualTo(Quantity $quantity): bool{...}
private function checkMetric(Quantity $quantity): void{...}
private function fromNumber(float $number): static{...}
public function value(): float
{
    return $this->roundingPolicy->round($this->amount);
}

public function symbol(): string
{
    return $this->metric->symbol();
```

```php
readonly class Quantity
{
    public function __construct(
        protected float $amount,
        protected Metric $metric,
        protected RoundingPolicy $roundingPolicy
    ) {
    }

    public function
    {
        return $thi
    }

    public function
    public function
    public function
    public function
    public function
    public function
    public function
    private functi
    private functi
    public function
    {
        return $thi
    }

    public function symbol(): string
    {
        return $this->metric->symbol();
    }
}
```

```php
interface Metric
{
    public function name(): string;
    public function symbol(): string;
    public function definition(): string;
    public function equals(Metric $other): bool;
}
```

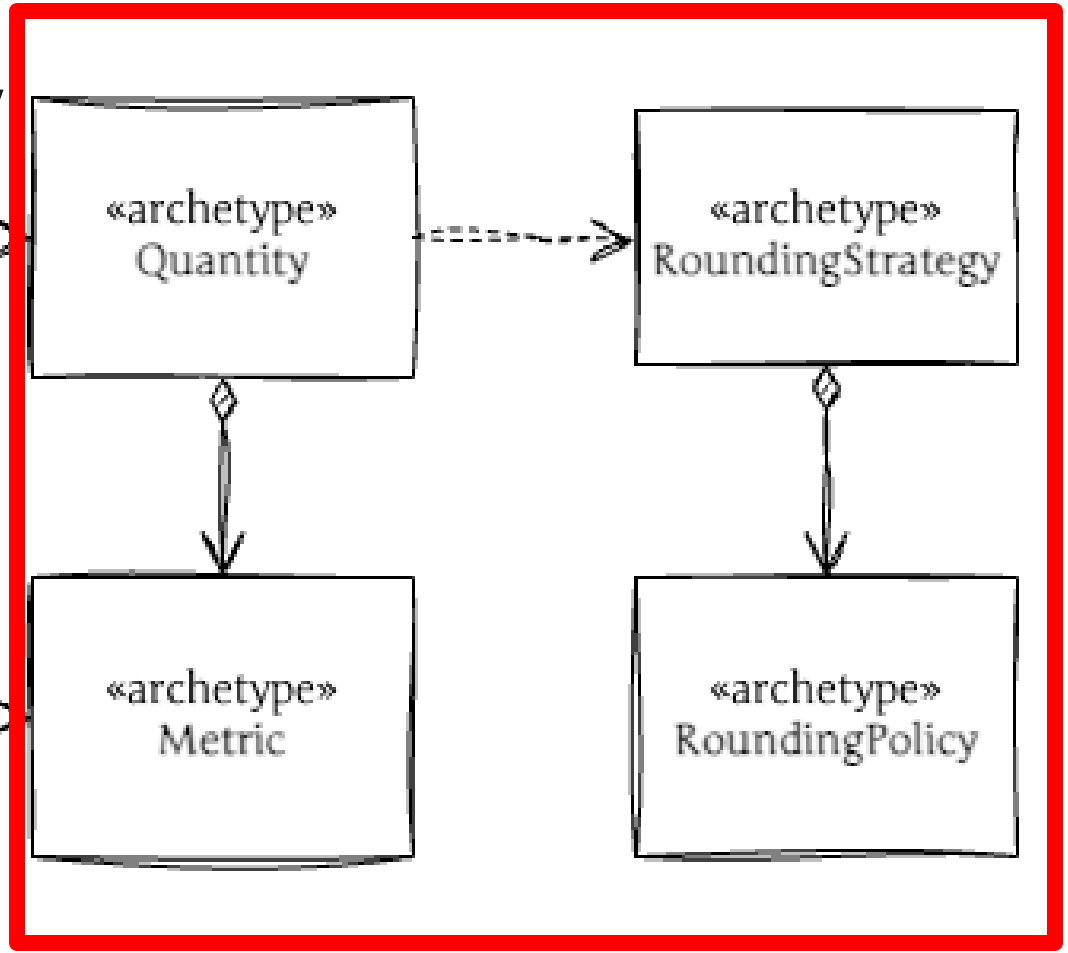
```php
    private function roundDefault(float $value): float{...}
}
```

```php
final readonly class RoundingPolicy
{

    public function __construct(
        public RoundingStrategy $strategy,
        public int $numberOfDigits = 2,
        public int $roundingDigit = 5
    ){
    }

    public function round(float $value): float
    {
        return match ($this->strategy) {
            RoundingStrategy::ROUND_UP => $this->roundUp($value),
            RoundingStrategy::ROUND_DOWN => $this->roundDown($value),
            default => $this->roundDefault($value),
        };

    }

    private function roundUp(float $value): float{...}

    private function roundDown(float $value): float{...}

    private function roundDefault(float $value): float{...}
```

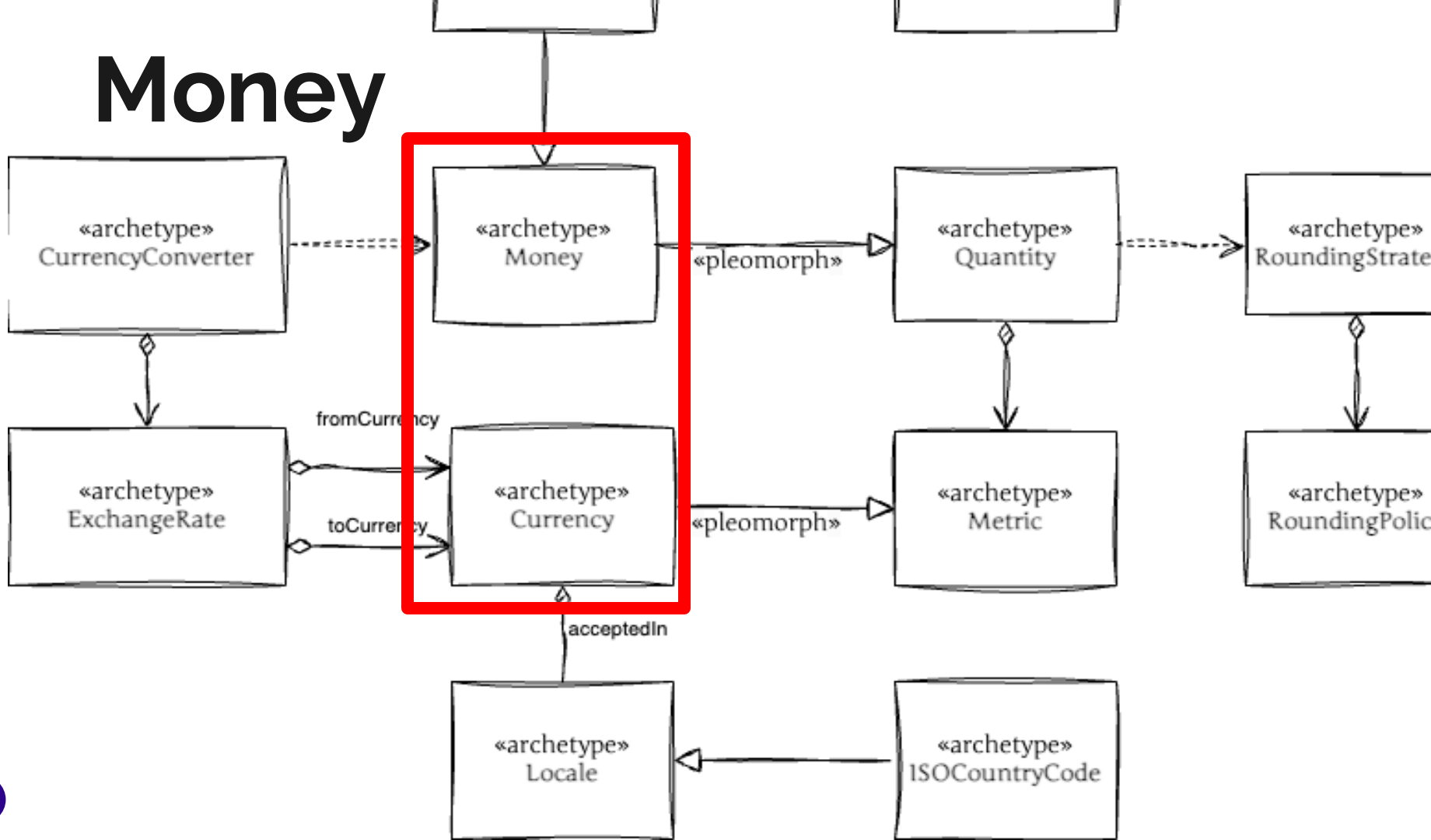
```php
enum RoundingStrategy
{
    case ROUND;
    case ROUND_UP;
    case ROUND_DOWN;
}
```

```php
class QuantityPresentationTest extends TestCase
{
    public function testPresentationWithPointer(): void
    {
        $quantity = QuantityFixtureFactory::get(
            number: 10.12345,
            roundingPolicy: RoundingPolicyFixtureFactory::get(numberOfDigits: 2),
            metric: MetricFixtureFactory::get( name: 'meter', symbol: 'm', ));
        $this->assertEquals( expected: 10.12, $quantity->value());
        $this->assertEquals( expected: 'm', $quantity->symbol());
    }

    public function testPresentationInt(): void
    {
        $quantity = QuantityFixtureFactory::get(
            number: 10.5653,
            roundingPolicy: RoundingPolicyFixtureFactory::get(numberOfDigits: 0),
            metric: MetricFixtureFactory::get( name: 'meter', symbol: 'm', ));
        $this->assertEquals( expected: 11.00, $quantity->value());
        $this->assertEquals( expected: 'm', $quantity->symbol());
    }
}
```

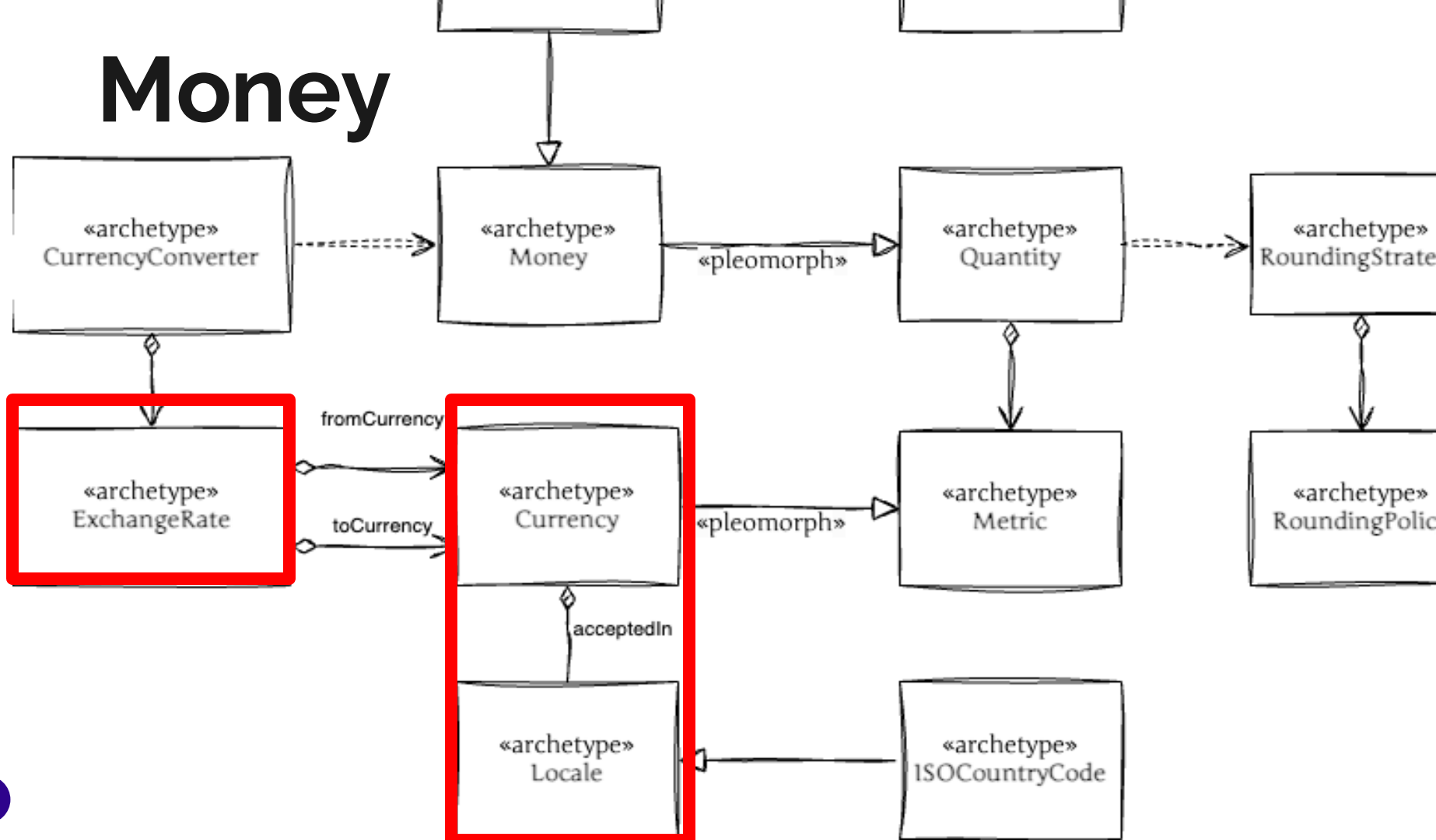
Quantity


«archetype»
Quantity
«archetype»
RoundingStrategy
«archetype»
Metric
«archetype»
RoundingPolicy

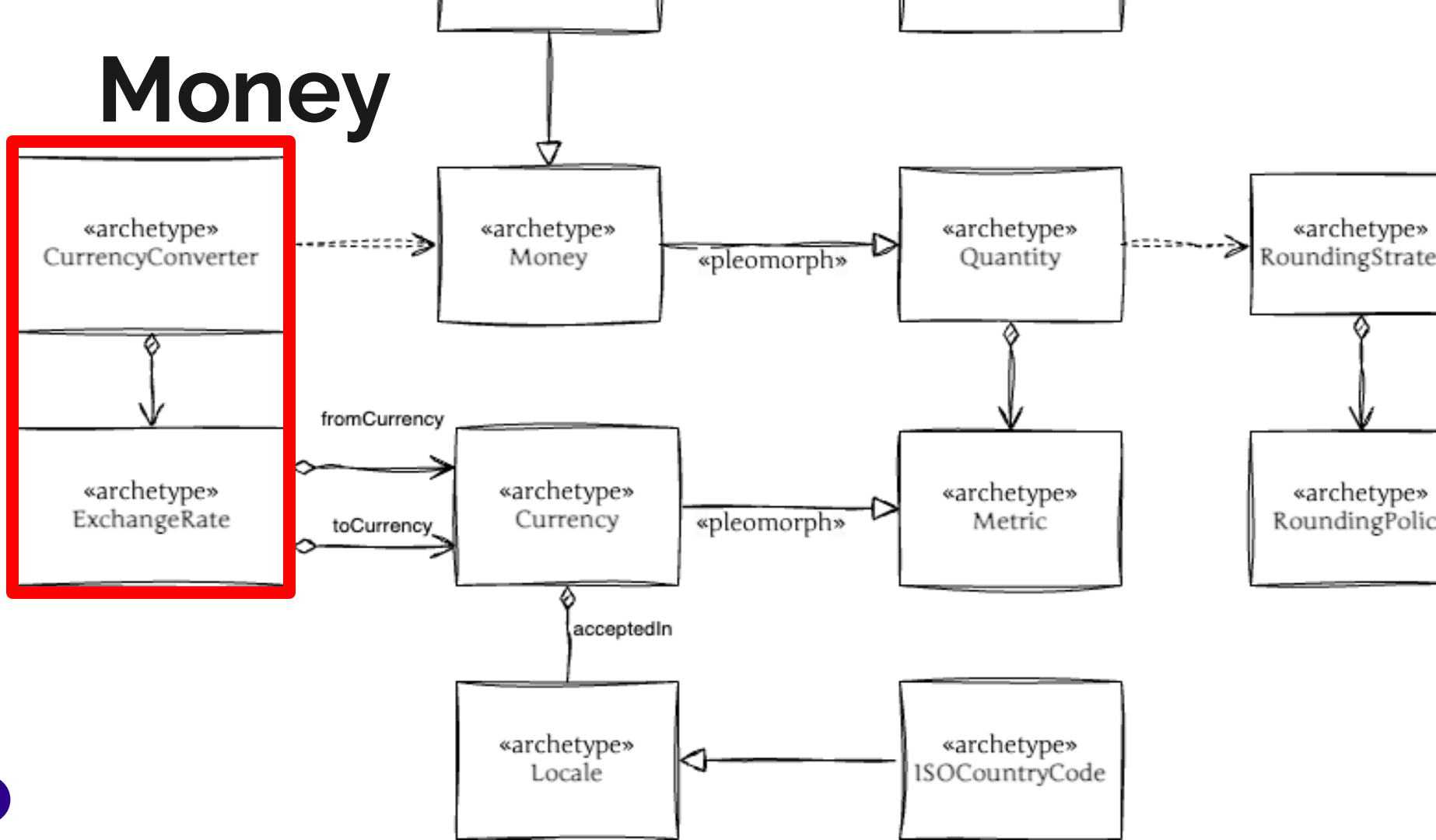
Money


«archetype»
CurrencyConverter
«archetype»
Money
«pleomorph»
«archetype»
Quantity
«archetype»
RoundingStrate
fromCurrency
«archetype»
ExchangeRate
toCurrency
«archetype»
Currency
«pleomorph»
«archetype»
Metric
«archetype»
RoundingPolic
acceptedIn
«archetype»
Locale
«archetype»
ISOCountryCode

Money


«archetype»
CurrencyConverter
«archetype»
Money
«pleomorph»
«archetype»
Quantity
«archetype»
RoundingStrate
«archetype»
ExchangeRate
fromCurrency
toCurrency
«archetype»
Currency
«pleomorph»
«archetype»
Metric
«archetype»
RoundingPolic
acceptedIn
«archetype»
Locale
«archetype»
ISOCountryCode

Money


«archetype»
CurrencyConverter
«archetype»
ExchangeRate
«archetype»
Money
«pleomorph»
«archetype»
Quantity
«archetype»
RoundingStrate
fromCurrency
toCurrency
«archetype»
Currency
«pleomorph»
«archetype»
Metric
«archetype»
RoundingPolic
acceptedIn
«archetype»
Locale
«archetype»
ISOCountryCode

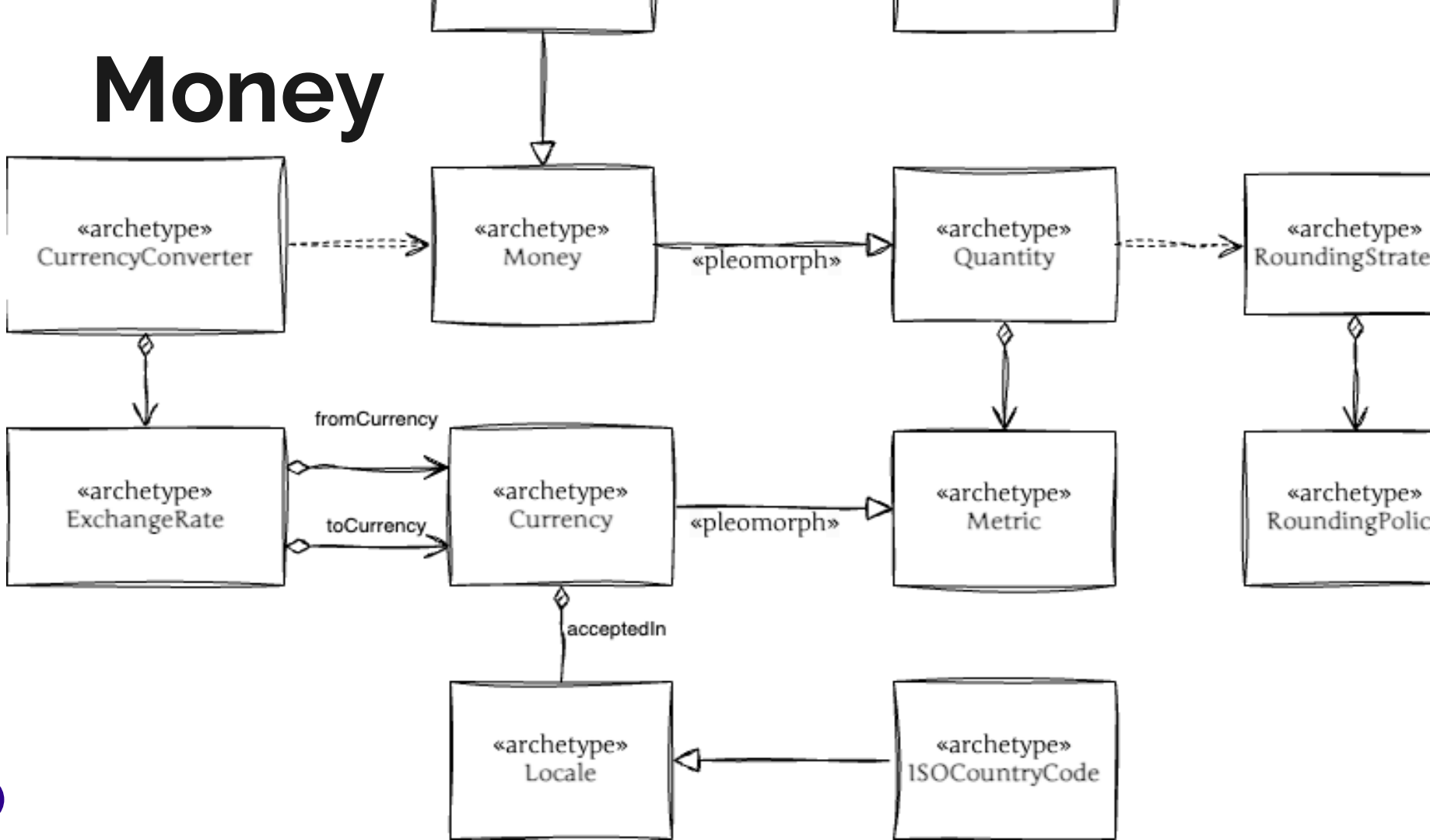
```php
public function testExchange(): void
{
    $converter = new CurrencyConverter();
    $exchangeRate1 = ExchangeRateFixtureFactory::getPLNtoEUR(
        rate: 0.3,
        validFrom: '2024-08-01',
        validTo: '2024-08-02'
    );

    $exchangeRate2 = ExchangeRateFixtureFactory::getPLNtoEUR(
        rate: 0.5,
        validFrom: '2024-08-02',
        validTo: '2024-08-03'
    );
    $converter->addExchangeRate($exchangeRate1)->addExchangeRate($exchangeRate2);
    $rate = $converter->getExchangeRateForDate(
        CurrencyFixtureFactory::getPLN(),
        CurrencyFixtureFactory::getEUR(),
        new DateTimeImmutable( datetime: '2024-08-02'),
    );

    $money = MoneyFixtureFactory::get( amount: 50, CurrencyFixtureFactory::getPLN());

    $convertedMoney = $converter->exchange($money, CurrencyFixtureFactory::getEUR(), $rate);


    $this->assertEquals( expected: 25, $convertedMoney->amount());
    $this->assertEquals( expected: 'EUR', $convertedMoney->currency()->alphaCode());
}
```

```php
public function testExchange(): void
{
    $converter = new CurrencyConverter();
    $exchangeRate1 = ExchangeRateFixtureFactory::getPLNtoEUR(
        rate: 0.3,
        validFrom: '2024-08-01',
        validTo: '2024-08-02'
    );

    $exchangeRate2 = ExchangeRateFixtureFactory::getPLNtoEUR(
        rate: 0.5,
        validFrom: '2024-08-02',
        validTo: '2024-08-03'
    );
    $this->assertEquals( expected: 'EUR', $convertedMoney->currency()->alphaCode());
}
```

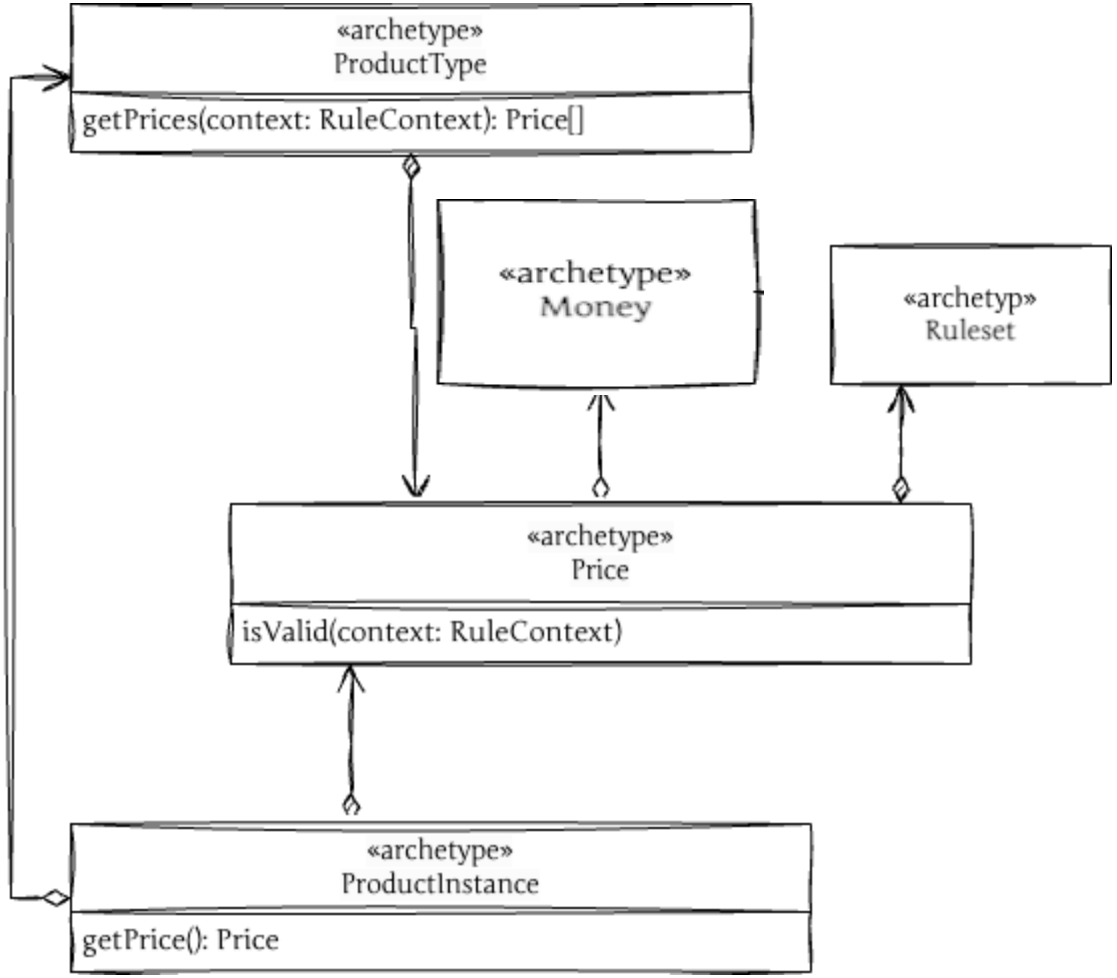
```
public function testExchange(): void
{
    $converter = new CurrencyConverter();

    $converter->addExchangeRate($exchangeRate1)->addExchangeRate($exchangeRate2);
    $rate = $converter->getExchangeRateForDate(
        CurrencyFixtureFactory::getPLN(),
        CurrencyFixtureFactory::getEUR(),
        new DateTimeImmutable( datetime: '2024-08-02'),
    );

    $money = MoneyFixtureFactory::get( amount: 50, CurrencyFixtureFactory::getPLN());

    $convertedMoney = $converter->exchange($money, CurrencyFixtureFactory::getEUR(), $rate);


    $this->assertEquals( expected: 25, $convertedMoney->amount());
    $this->assertEquals( expected: 'EUR', $convertedMoney->currency()->alphaCode());
}
```

```php
$converter->addExchangeRate($exchangeRate1)->addExchangeRate($exchangeRate2);
$rate = $converter->getExchangeRateForDate(
    CurrencyFixtureFactory::getPLN(),
    CurrencyFixtureFactory::getEUR(),
    new DateTimeImmutable( datetime: '2024-08-02'),
);

$money = MoneyFixtureFactory::get( amount: 50, CurrencyFixtureFactory::getPLN());

$convertedMoney = $converter->exchange($money, CurrencyFixtureFactory::getEUR(), $rate);


$this->assertEquals( expected: 25, $convertedMoney->amount());
$this->assertEquals( expected: 'EUR', $convertedMoney->currency()->alphaCode());
}
```

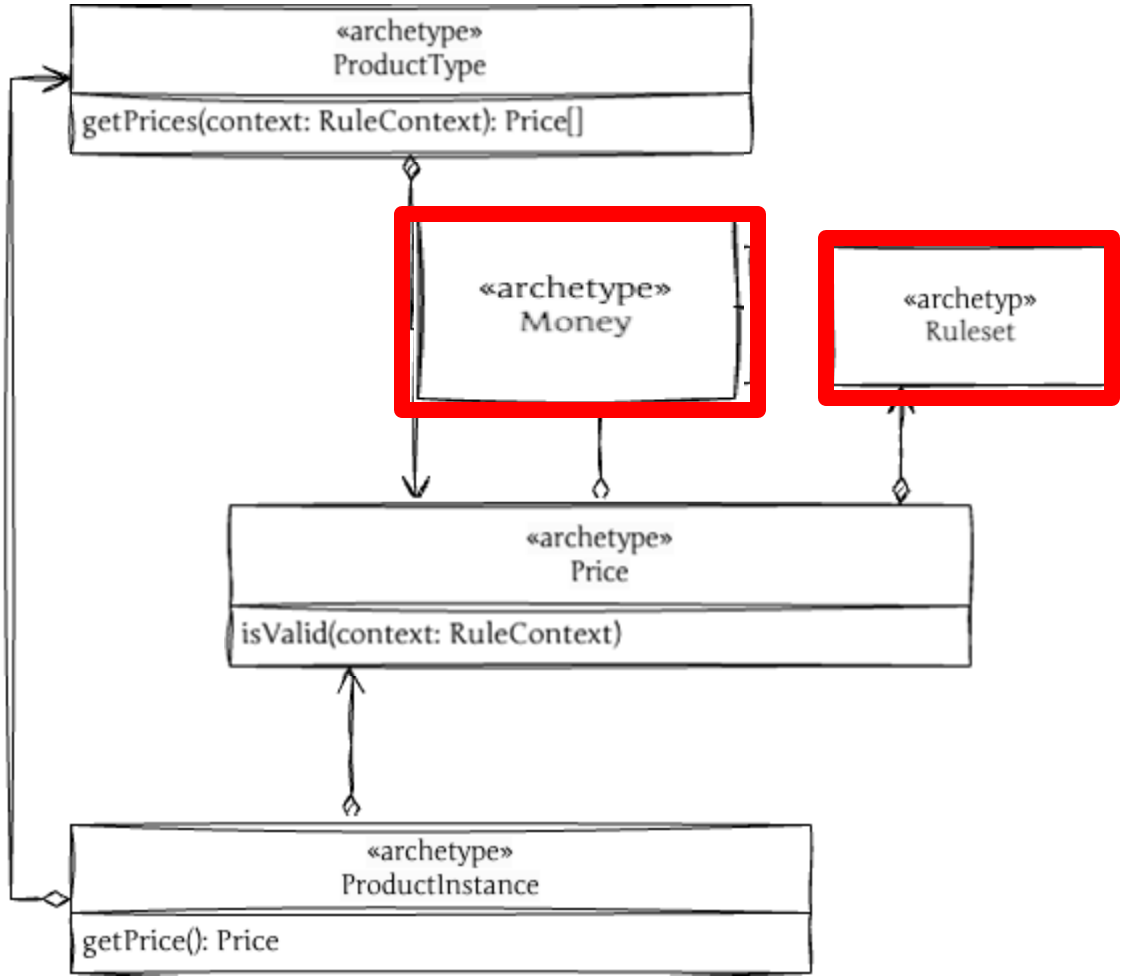
Money


«archetype»
CurrencyConverter
«archetype»
Money
«pleomorph»
«archetype»
Quantity
«archetype»
RoundingStrate
fromCurrency
«archetype»
ExchangeRate
toCurrency
«archetype»
Currency
«pleomorph»
«archetype»
Metric
«archetype»
RoundingPolic
acceptedIn
«archetype»
Locale
«archetype»
ISOCountryCode

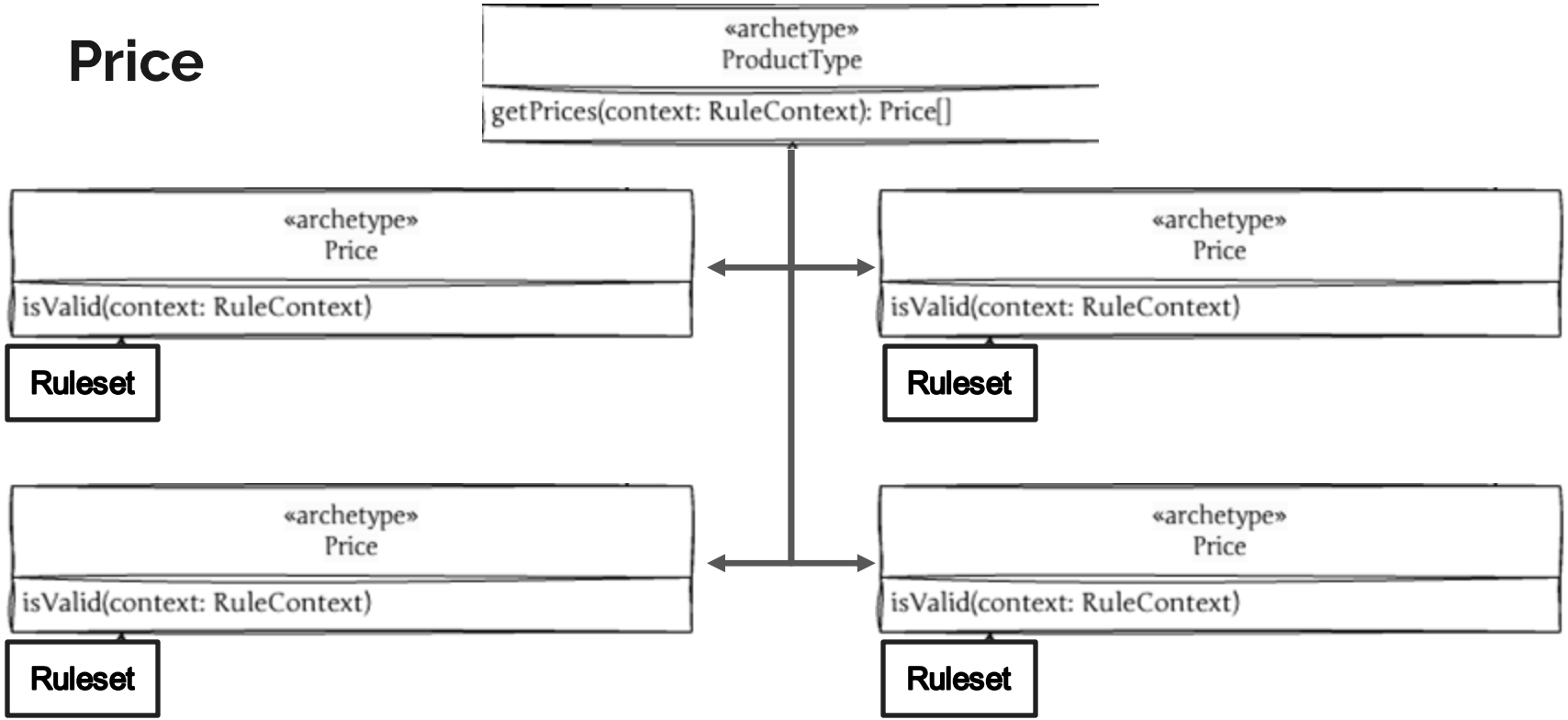
Price


«archetype»
ProductType
getPrices(context: RuleContext): Price[]
«archetype»
Money
«archetyp»
Ruleset
«archetype»
Price
isValid(context: RuleContext)
«archetype»
ProductInstance
getPrice(): Price

Price


«archetype»
ProductType
getPrices(context: RuleContext): Price[]
«archetype»
Money
«archetyp»
Ruleset
«archetype»
Price
isValid(context: RuleContext)
«archetype»
ProductInstance
getPrice(): Price

Price


«archetype»
ProductType
getPrices(context: RuleContext): Price[]
«archetype»
Price
isValid(context: RuleContext)
Ruleset
«archetype»
Price
isValid(context: RuleContext)
Ruleset
«archetype»
Price
isValid(context: RuleContext)
Ruleset
«archetype»
Price
isValid(context: RuleContext)
Ruleset

```php
readonly class Price
{
    1 usage
    private Ruleset $ruleset;

    no usages   new *
    public function isApplicable(RuleContext $ruleContext): bool
    {
        return $this->ruleset->apply($ruleContext);
    }
}
```

```php
readonly class Price
{
    1 usage
    private Ruleset $ruleset;

    no usages   new *
    public function isApplicable(RuleContext $ruleContext): bool
    {
        return $this->ruleset->apply($ruleContext);
    }
}
```

```php
class Product
{
    /** @var GenericList<Price> */
    private GenericList $prices;

    public function getPrice(RuleContext $context): Price
    {
        return $this->prices->filter(
            fn (Price $price) => $price->isApplicable($context)
        )->findFirst()->getOrElseThrow(
            new PriceNotFoundException( string: 'Can not find valid price.')
        );
    }
}
```

```php
class PurchaseOrder
{
    no usages   new *
    public function addItem(
        Product $product,
        UnsignedQuantity $quantity,
        \DateTimeImmutable $requestedAt
    ): void
    {
        $this->items->add(
            new OrderLineItem($product, $product->getPrice(
                Pricing::createRuleContext($product, $quantity, $this->location)), $quantity
            )
        );
    }
}
```

```php
class PurchaseOrder
{
    no usages   new *
    public function addItem(
        Product $product,
        UnsignedQuantity $quantity,
        \DateTimeImmutable $requestedAt
    ): void
    {
        $this->items->add(
            new OrderLineItem($product, $product->getPrice(
                Pricing::createRuleContext($product, $quantity, $this->location)), $quantity
            )
        );
    }
}
```

```php
class PurchaseOrder
{
    no usages   new *
    public function addItem(
        Product $product,
        UnsignedQuantity $quantity,
        \DateTimeImmutable $requestedAt
    ): void
    {
        $this->items->add(
            new OrderLineItem($product, $product->getPrice(
                Pricing::createRuleContext($product, $quantity, $this->location), $quantity
            )
        );
    }
}
```

```php
class PurchaseOrder
{
    no usages   new *
    public function addItem(
        Product $product,
        UnsignedQuantity $quantity,
        \DateTimeImmutable $requestedAt
    ): void
    {
        $this->items->add(
            new OrderLineItem($product, $product->getPrice(
                Pricing::createRuleContext($product, $quantity, $this->location)), $quantity
            )
        );
    }
}
```

```php
class Pricing
{
    2 usages  new *
    public static function createRuleContext(
        Product $product,
        UnsignedQuantity $quantity,
        CountryCode $location
    ): RuleContext {
        $elements = [
            new Variable('productCategory', $product->getCategory()),
            new Variable('quantity', $quantity->getValue()),
            new Variable('location', $location),
            new Proposition('seasonalPromotion', fn() => self::isSeasonalPromotionActive())
        ];

        return new RuleContext($elements);
    }
}
```

Ceny uzależnione od różnych czynników

Money & Price

Produkt może posiadać wiele cen

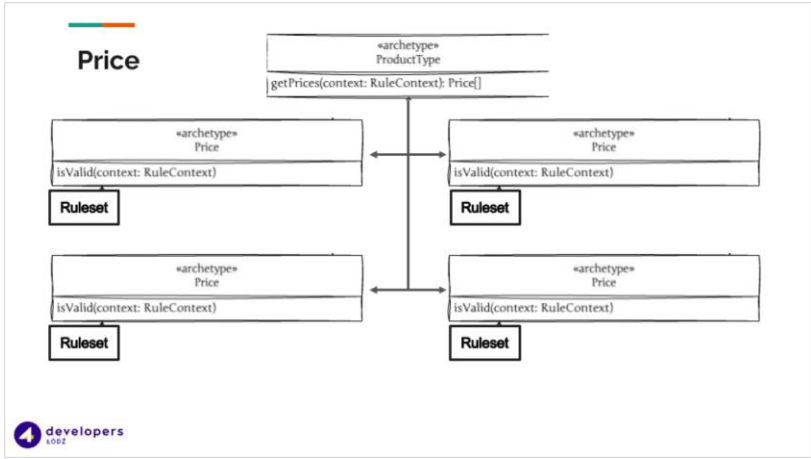
Wybór odpowiedniej ceny jest uzależniony od różnych **zmiennych** czynników

Możliwość implementacji przelicznika walut

Ceny uzależnione od różnych czynników

Money & Price

Produkt może posiadać wiele cen




```php
class Product
{
    /** @var GenericList<Price> */
    private GenericList $prices;

    public function getPrice(RuleContext $context): Price
    {
        return $this->prices->filter(
            fn (Price $price) => $price->isApplicable($context)
        )->findFirst()->getOrElseThrow(
            new PriceNotFoundException( string: 'Can not find valid price.')
        );
    }
}
```

developers ŁÓDŹ

Wybór odpowiedniej ceny jest uzależniony od różnych **zmiennych** czynników

Możliwość implementacji przelicznika walut

Ceny uzależnione od różnych czynników

Money & Price

Wybór odpowiedniej ceny jest uzależniony od różnych **zmiennych** czynników

```php
readonly class Price
{
    1 usage
    private Ruleset $ruleset;

    no usages  new *
    public function isApplicable(RuleContext $ruleContext): bool
    {
        return $this->ruleset->apply($ruleContext);
    }
}
```

developers ŁÓDŹ

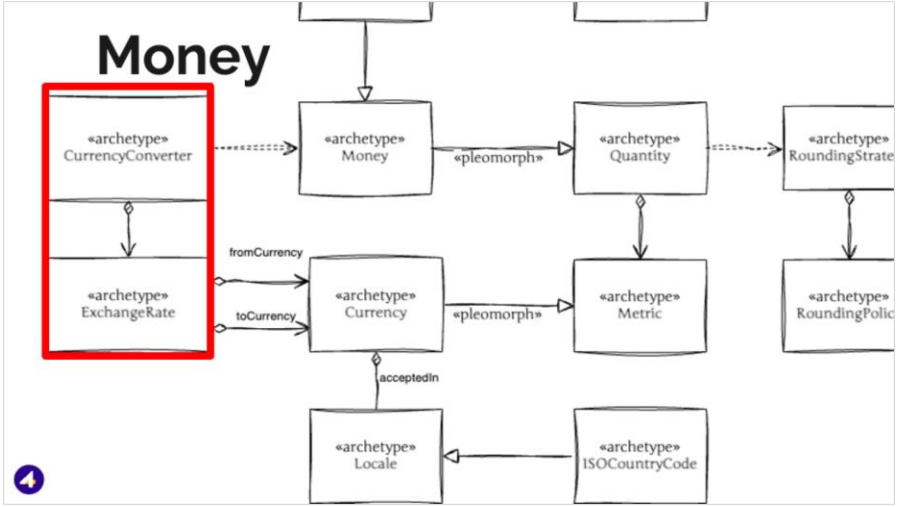
Produkt może posiadać wiele cen

Możliwość implementacji przelicznika walut

Ceny uzależnione od różnych czynników

Money & Price

Możliwość implementacji przelicznika walut



Produkt może posiadać wiele cen

Wybór odpowiedniej ceny jest uzależniony od różnych **zmiennych** czynników

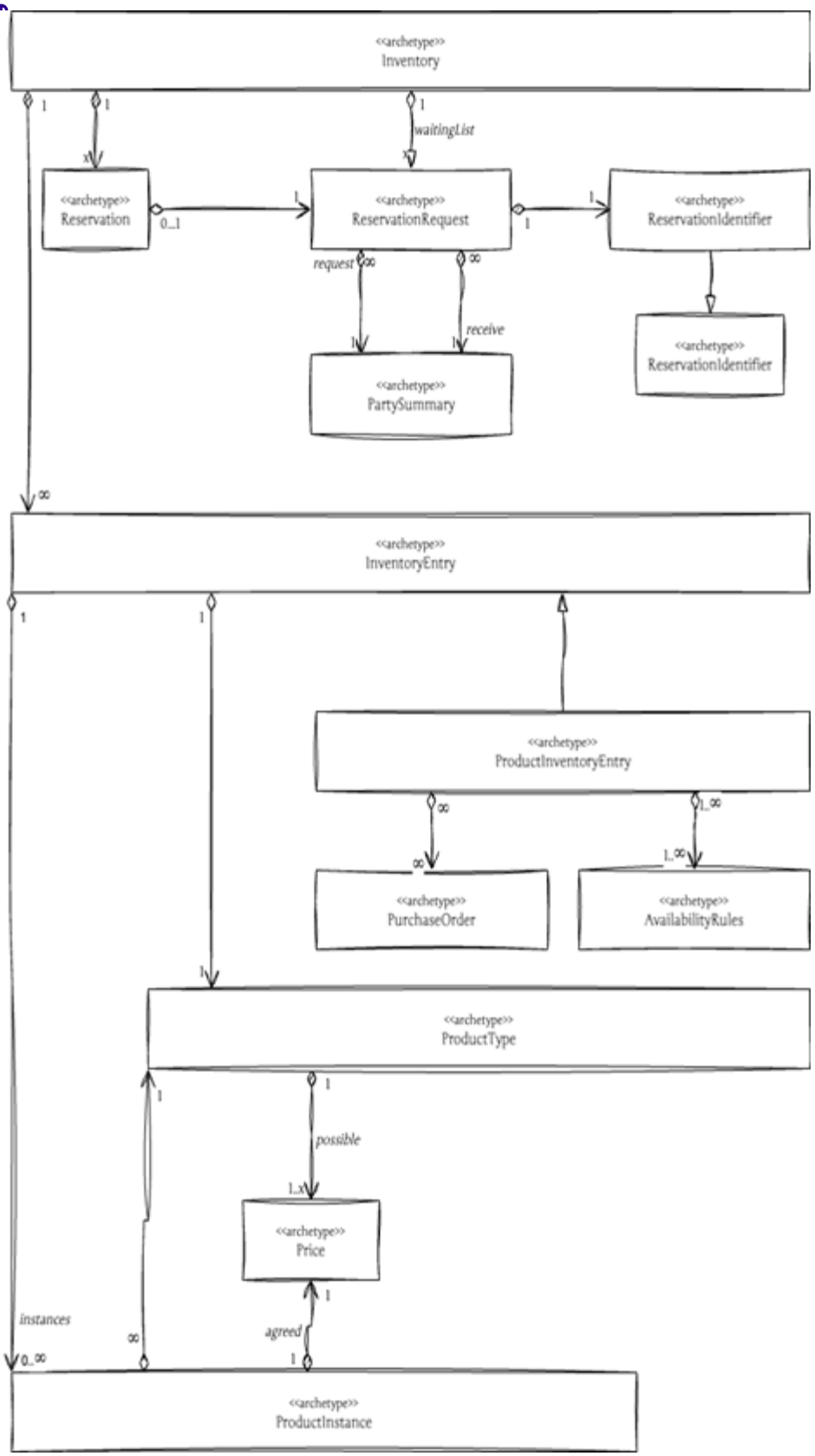
Rozbudowa systemu magazynowego

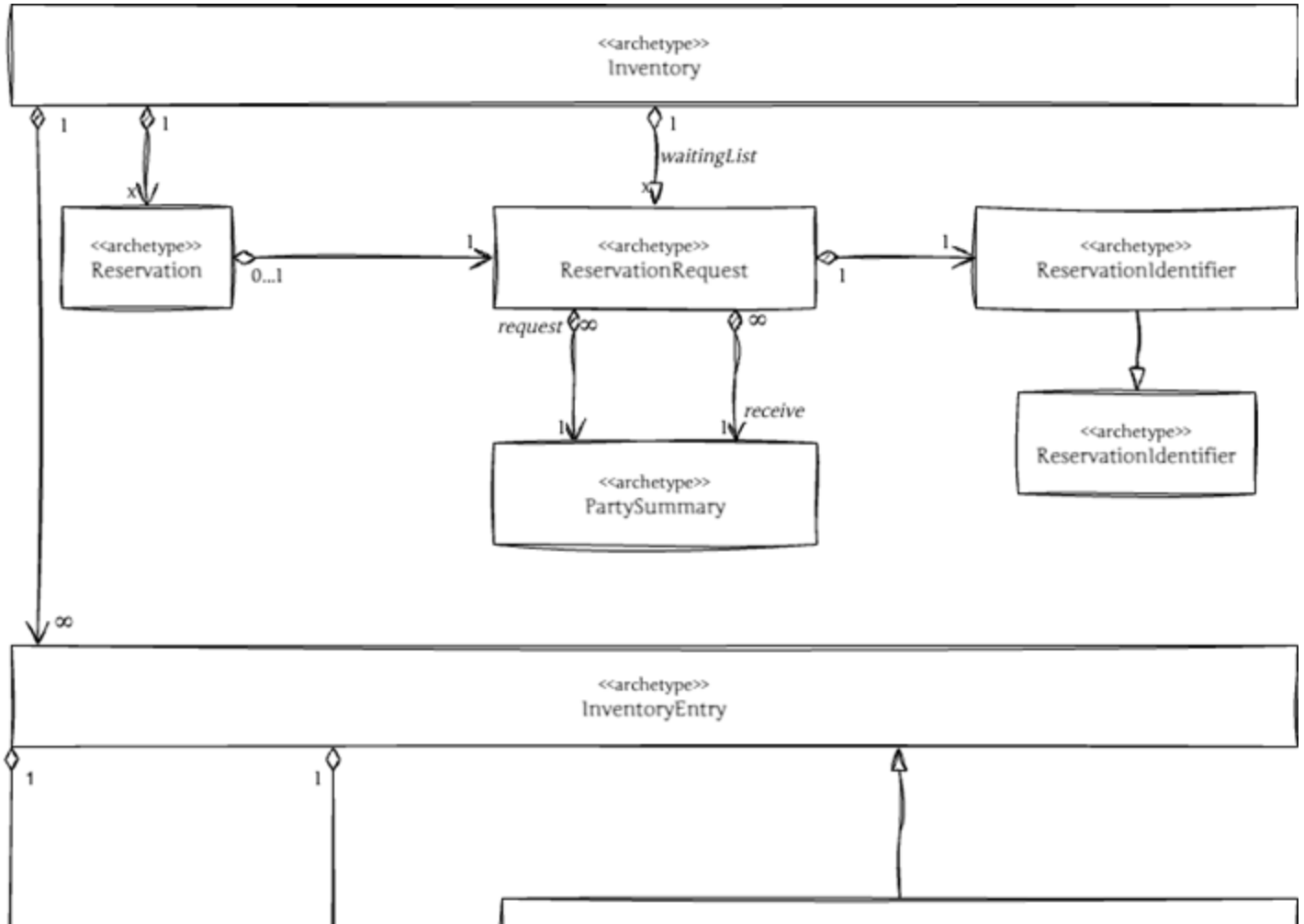
Inventory
Inventory*

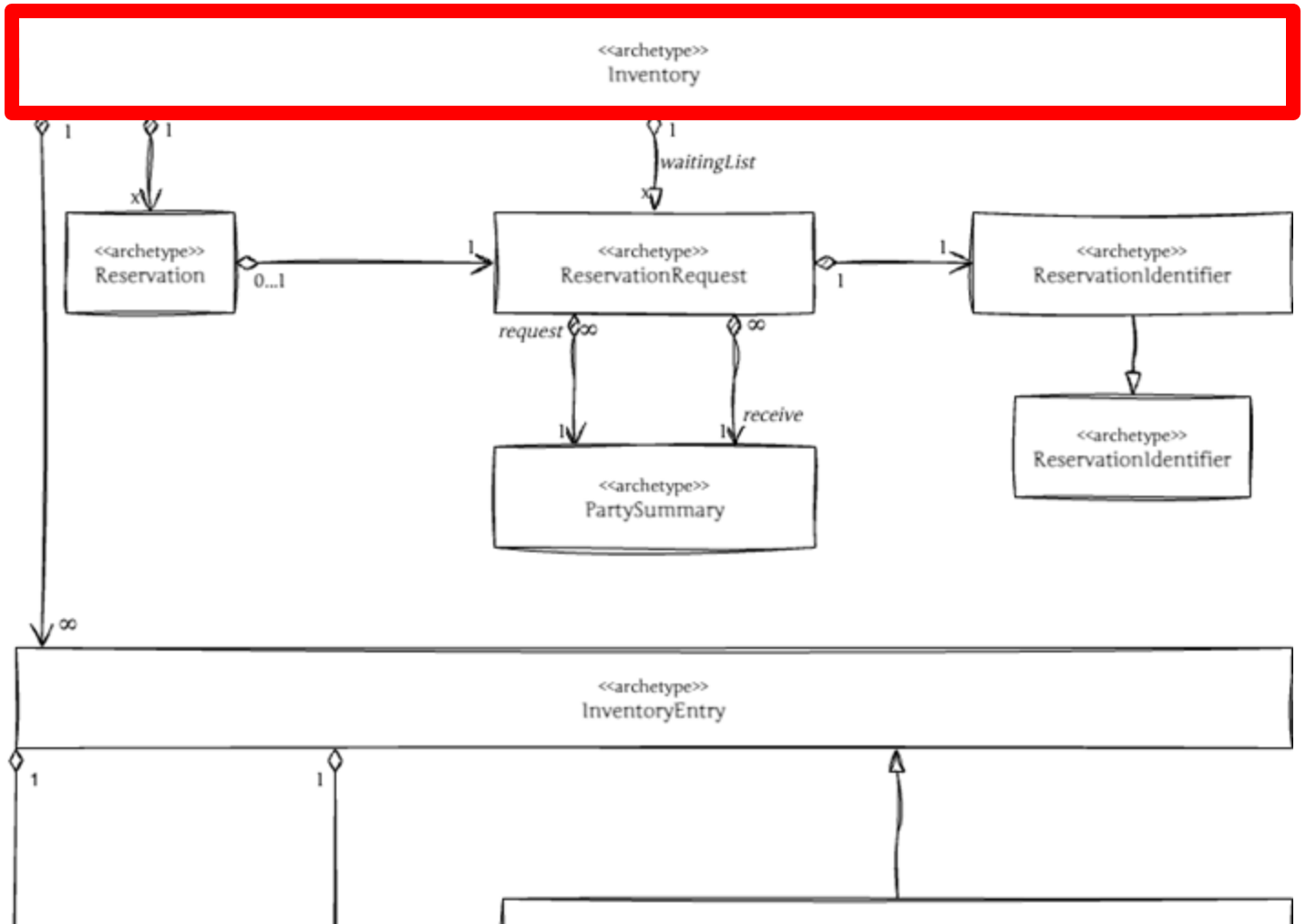
Rozbudowa systemu magazynowego

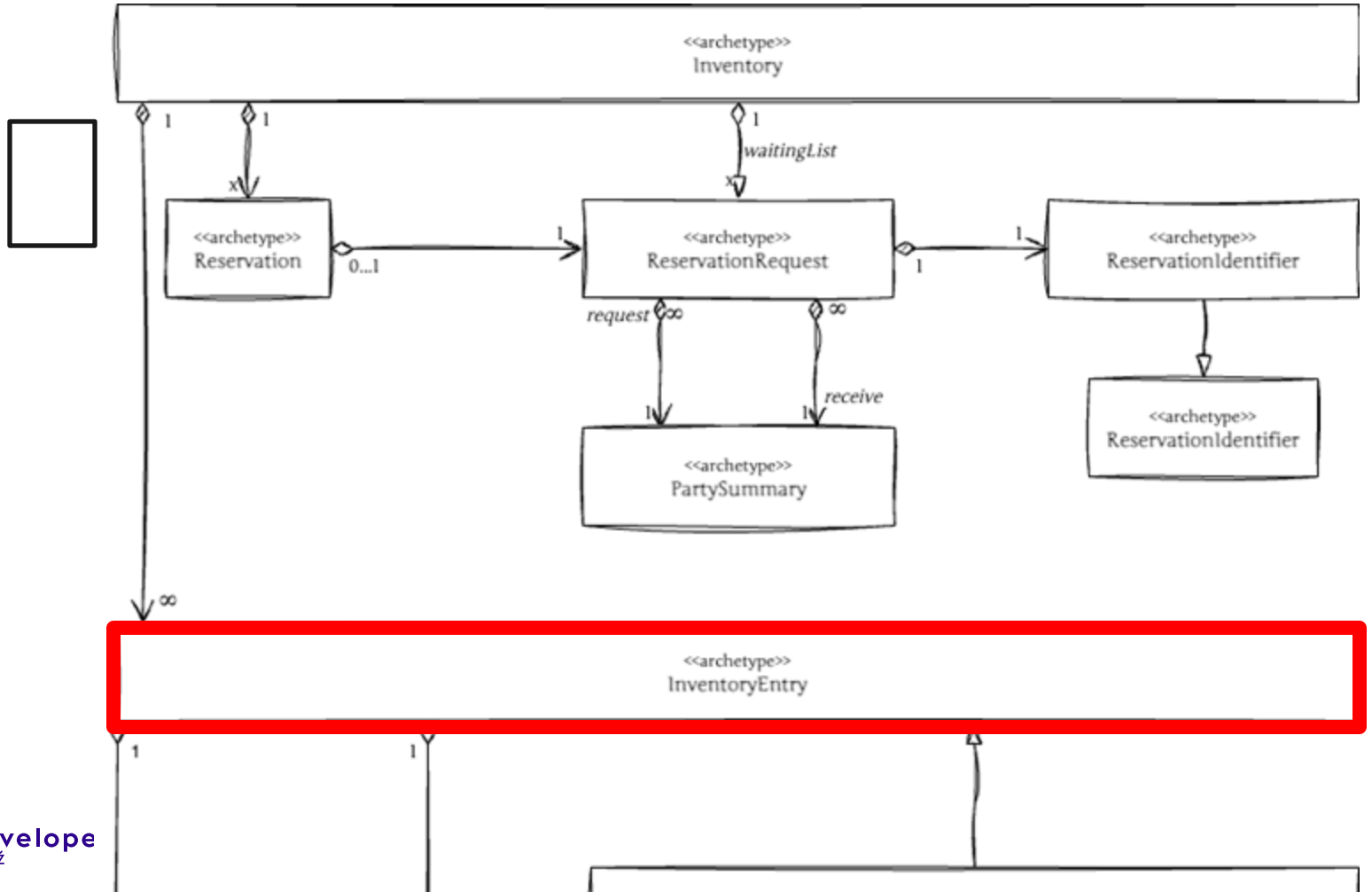
Inventory
Inventory*

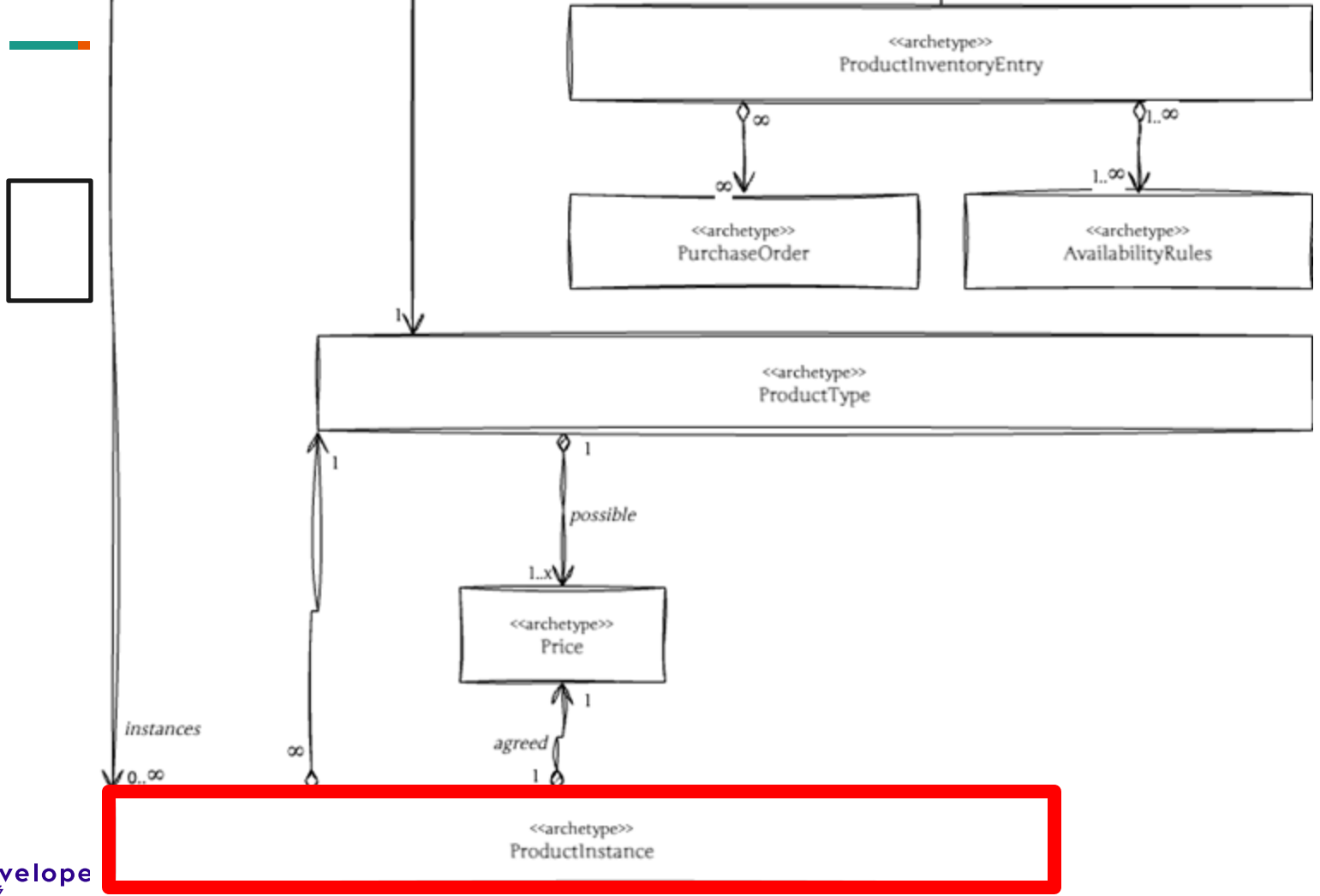
- Dostępność produktu zależy od różnych czynników
- Możliwość śledzenia historii inwentarzu
- Jeden typ produktu może być przechowywany w różnych magazynach.

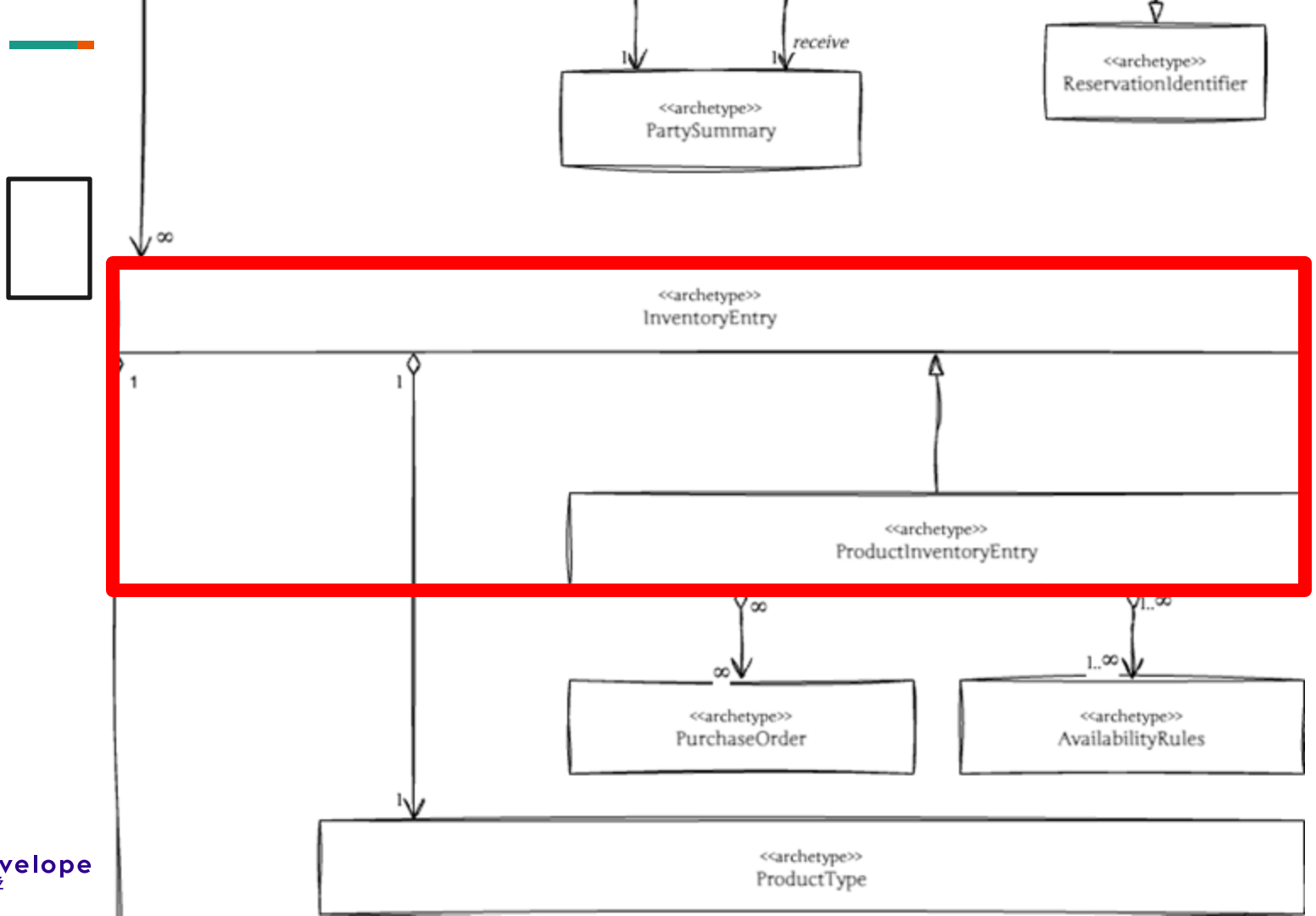

«archetype»
Inventory
1
1
1
waitingList
x
«archetype»
Reservation
0..1
1
«archetype»
ReservationRequest
1
1
«archetype»
ReservationIdentifier
request
∞
∞
receive
1
1
«archetype»
PartySummary
«archetype»
ReservationIdentifier
∞
«archetype»
InventoryEntry
1
1
«archetype»
ProductInventoryEntry
∞
∞
1..∞
1..∞
«archetype»
PurchaseOrder
«archetype»
AvailabilityRules
1
«archetype»
ProductType
1
1
possible
1..x
«archetype»
Price
1
instances
∞
agreed
0..∞
1
«archetype»
ProductInstance

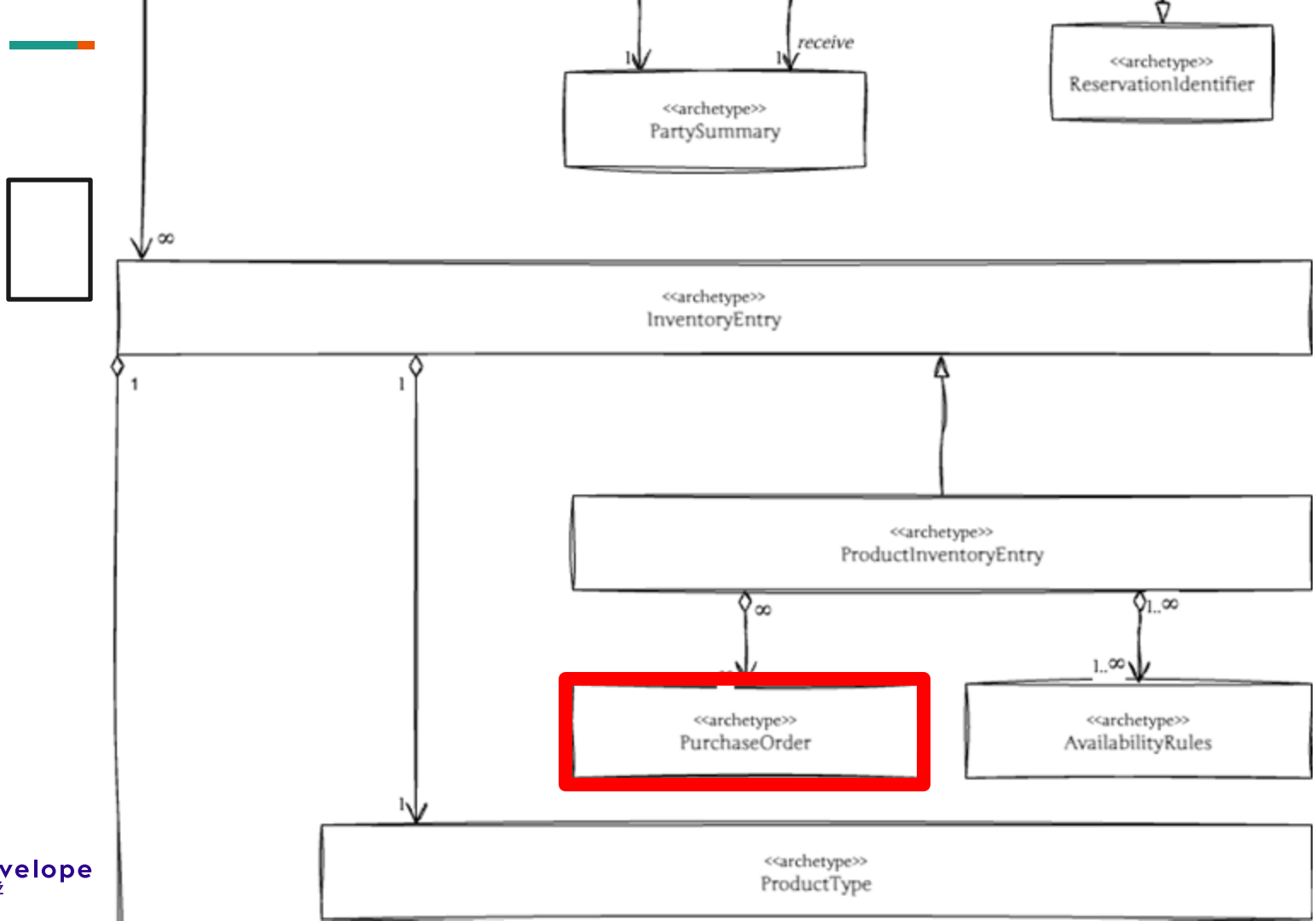


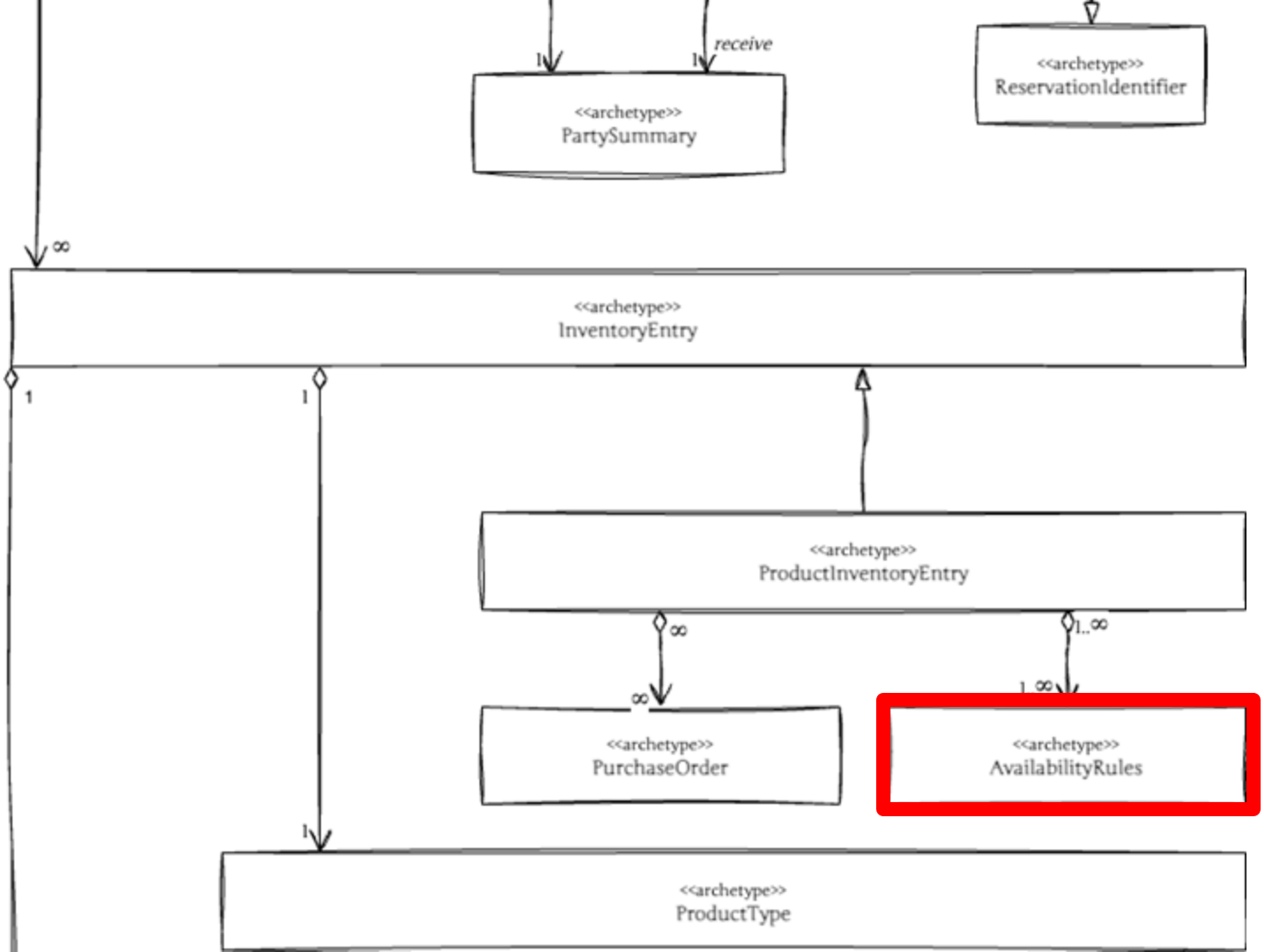

<<archetype>>
Inventory
1
1
1
waitingList
x
x
<<archetype>>
Reservation
0...1
1
<<archetype>>
ReservationRequest
1
1
<<archetype>>
ReservationIdentifier
request
∞
∞
receive
1
1
<<archetype>>
PartySummary
<<archetype>>
ReservationIdentifier
∞
<<archetype>>
InventoryEntry
1
1

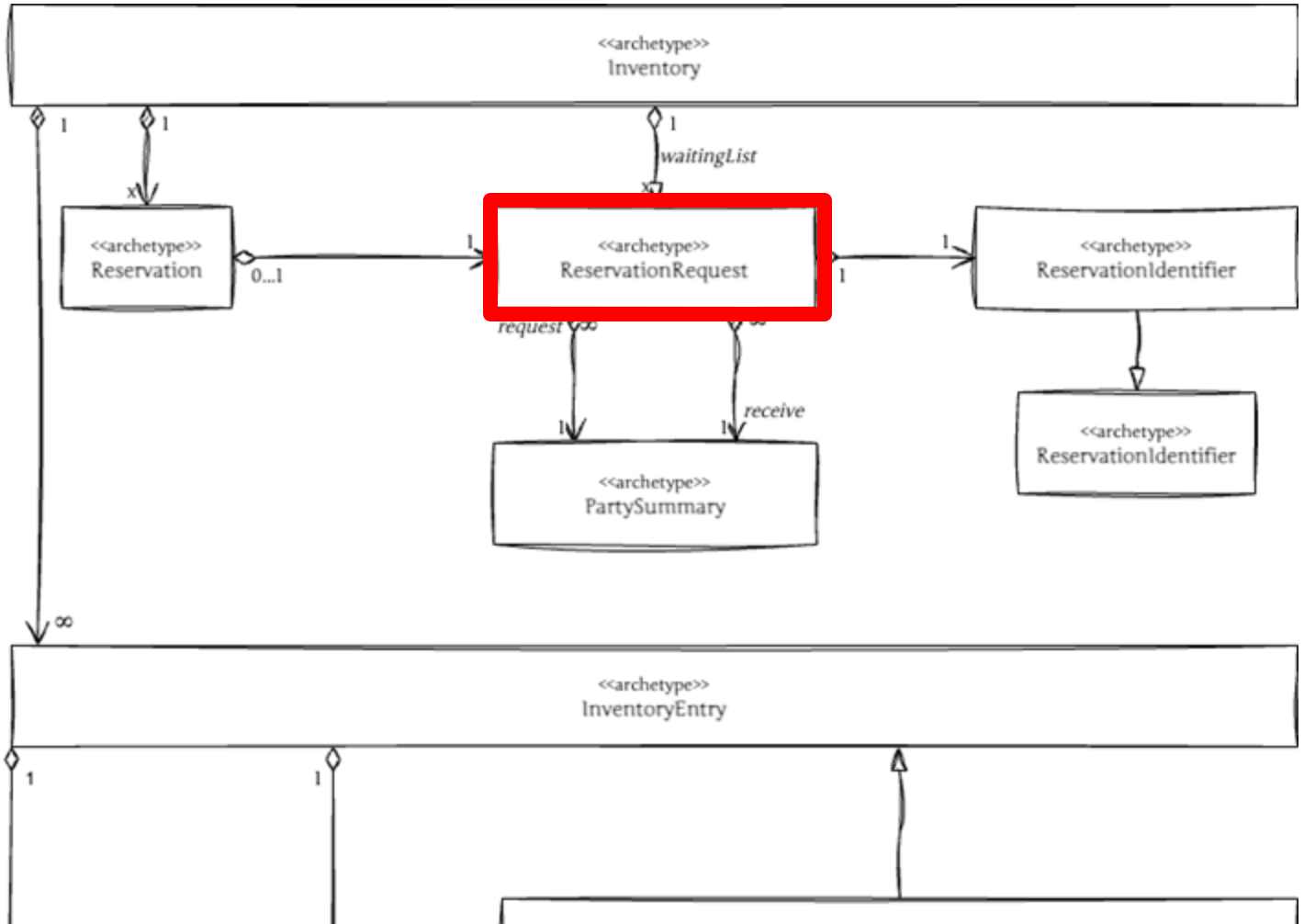

<<archetype>>
Inventory
1
1
1
waitingList
x
x
<<archetype>>
Reservation
0...1
1
<<archetype>>
ReservationRequest
1
<<archetype>>
ReservationIdentifier
request
∞
∞
receive
1
1
<<archetype>>
PartySummary
<<archetype>>
ReservationIdentifier
∞
<<archetype>>
InventoryEntry
1
1

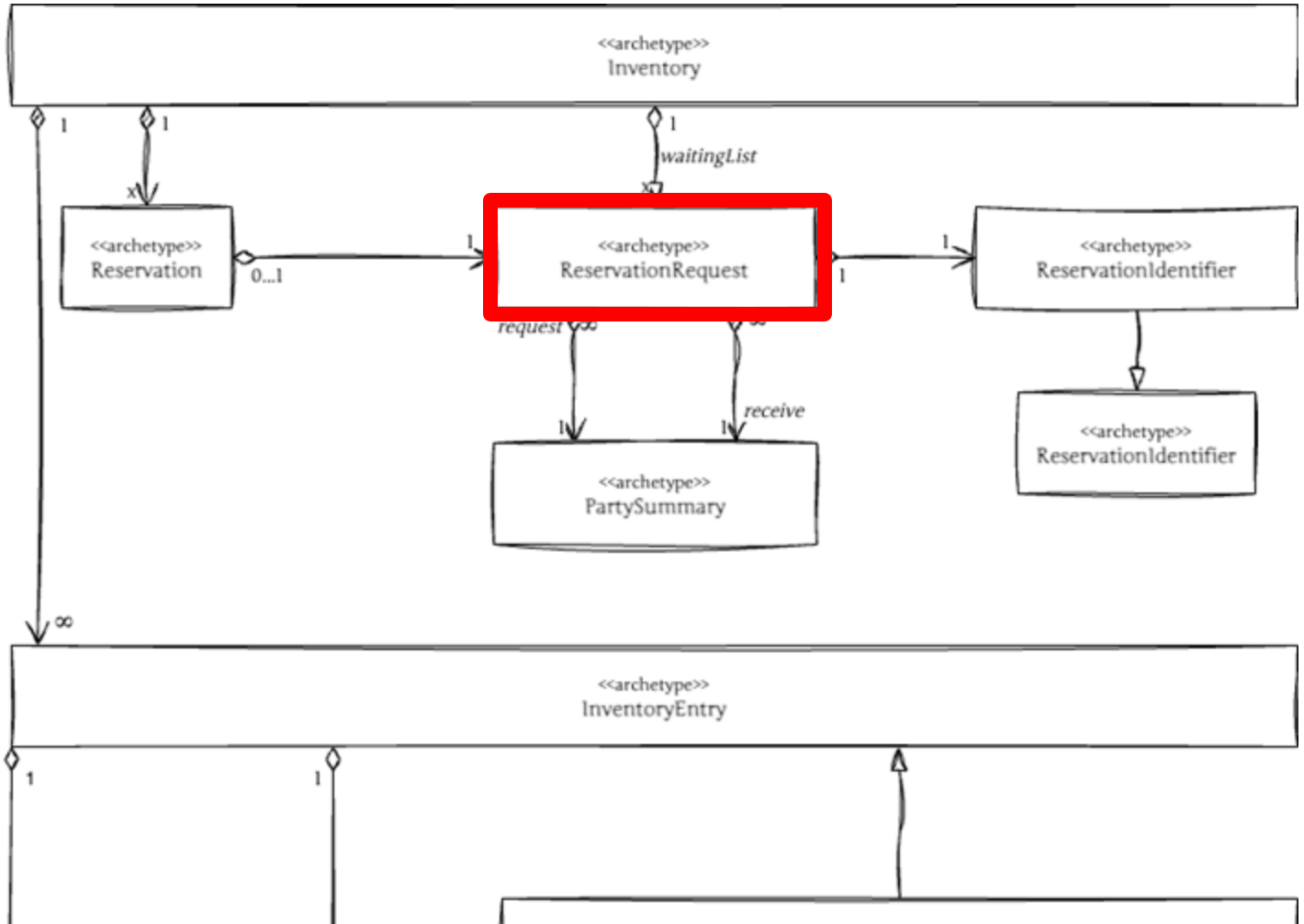

<<archetype>>
ProductInventoryEntry
∞
1..∞
∞
1..∞
<<archetype>>
PurchaseOrder
<<archetype>>
AvailabilityRules
1
<<archetype>>
ProductType
1
1
possible
1..x
<<archetype>>
Price
1
instances
∞
agreed
0..∞
1
<<archetype>>
ProductInstance

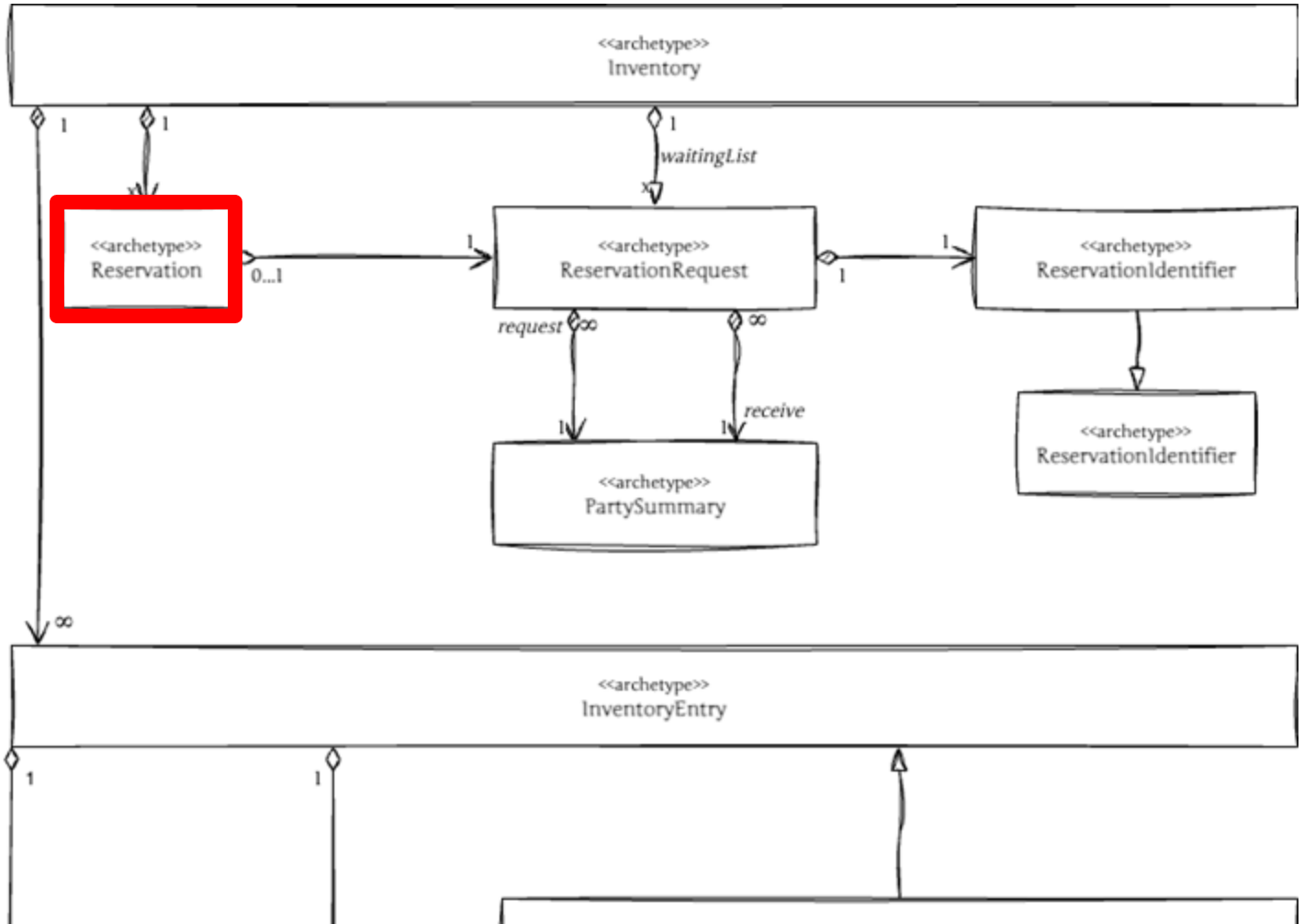







<<archetype>>
Inventory
1
1
1
waitingList
x
x
<<archetype>>
Reservation
0...1
1
<<archetype>>
ReservationRequest
1
<<archetype>>
ReservationIdentifier
request
receive
1
1
<<archetype>>
PartySummary
<<archetype>>
ReservationIdentifier
∞
<<archetype>>
InventoryEntry
1
1


<<archetype>>
Inventory
1
1
1
waitingList
x
x
<<archetype>>
Reservation
0...1
1
<<archetype>>
ReservationRequest
1
<<archetype>>
ReservationIdentifier
request
receive
1
1
<<archetype>>
PartySummary
<<archetype>>
ReservationIdentifier
∞
<<archetype>>
InventoryEntry
1
1



Rozbudowa systemu magazynowego

Inventory

Dostępność produktu zależy od różnych czynników.

Możliwość śledzenia historii inwentarzu

Jeden typ produktu może być przechowywany w różnych magazynach.

Rozbudowa systemu magazynowego

Inventory

Dostępność produktu zależy od różnych czynników.

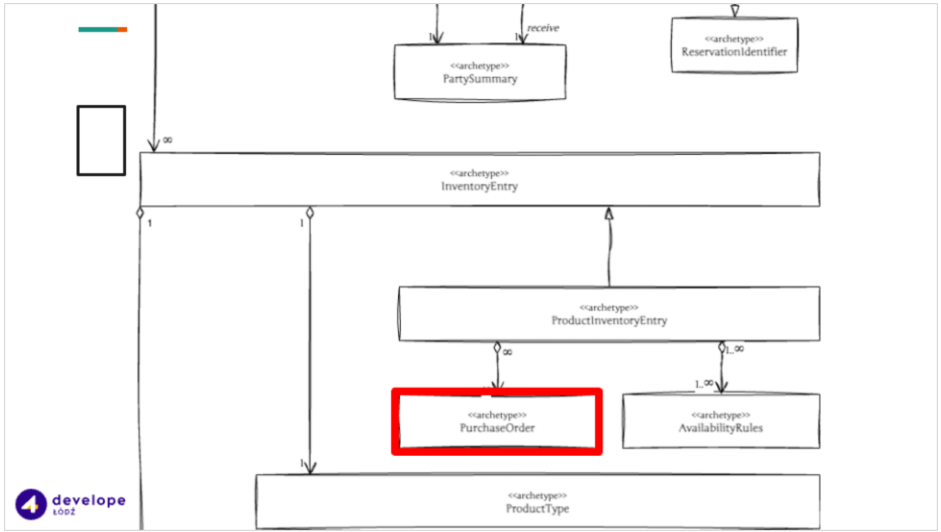
Możliwość śledzenia historii inwentarzu

Jeden typ produktu może być przechowywany w różnych magazynach.

Rozbudowa systemu magazynowego

Inventory

Możliwość śledzenia historii inwentarzu



Jeden typ produktu może być przechowywany w różnych magazynach.

Dostępność produktu zależy od różnych czynników.

developers
ŁÓDŹ

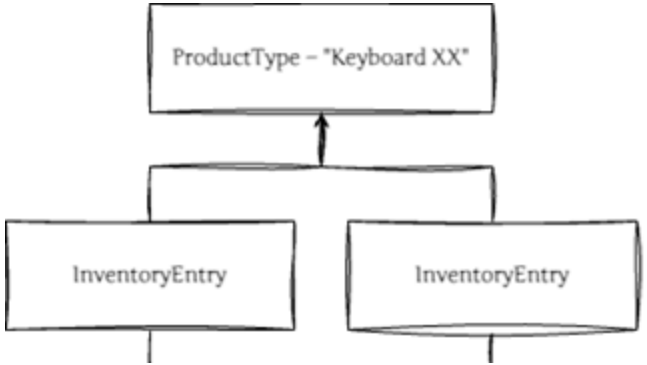
Rozbudowa systemu magazynowego

Inventory

Jeden typ produktu może być przechowywany w różnych magazynach.

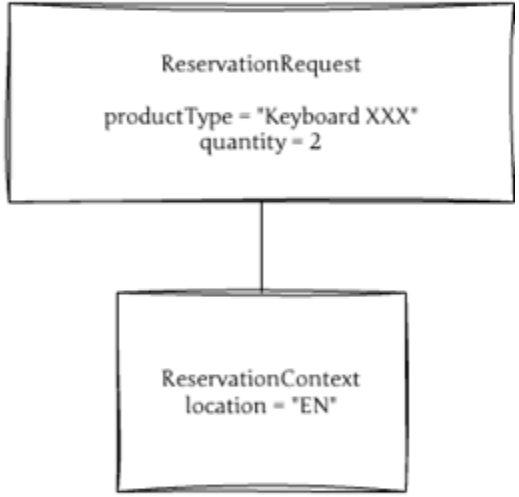
Dostępność produktu zależy od różnych czynników.

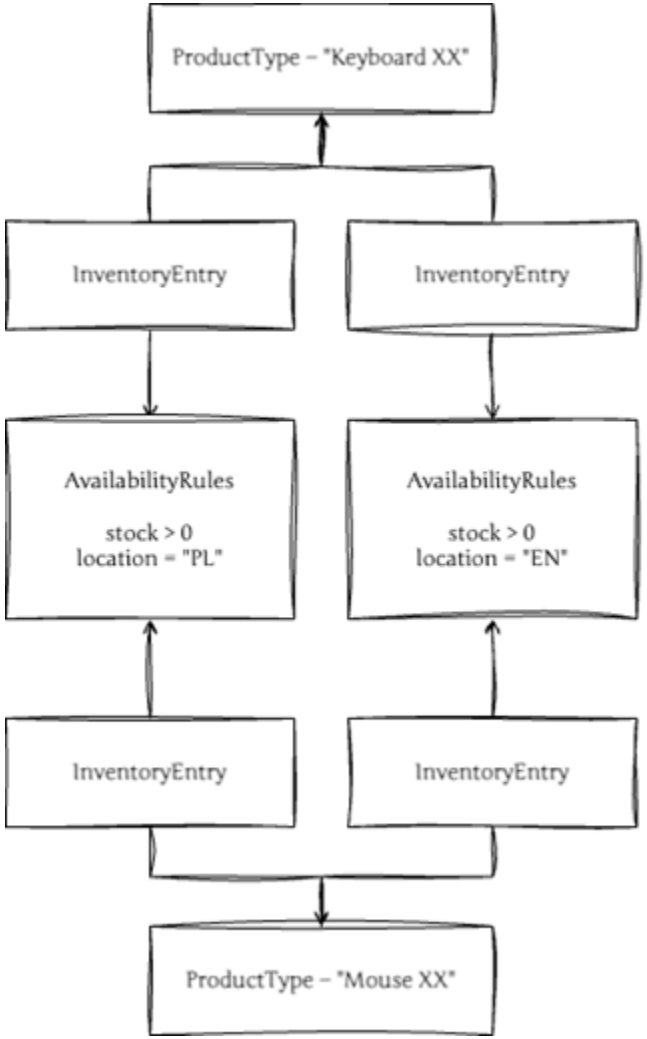
Możliwość śledzenia historii inwentarzu

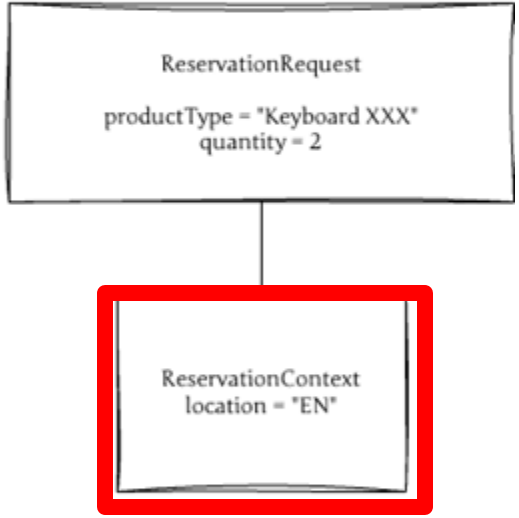

ProductType – "Keyboard XX"
InventoryEntry
InventoryEntry

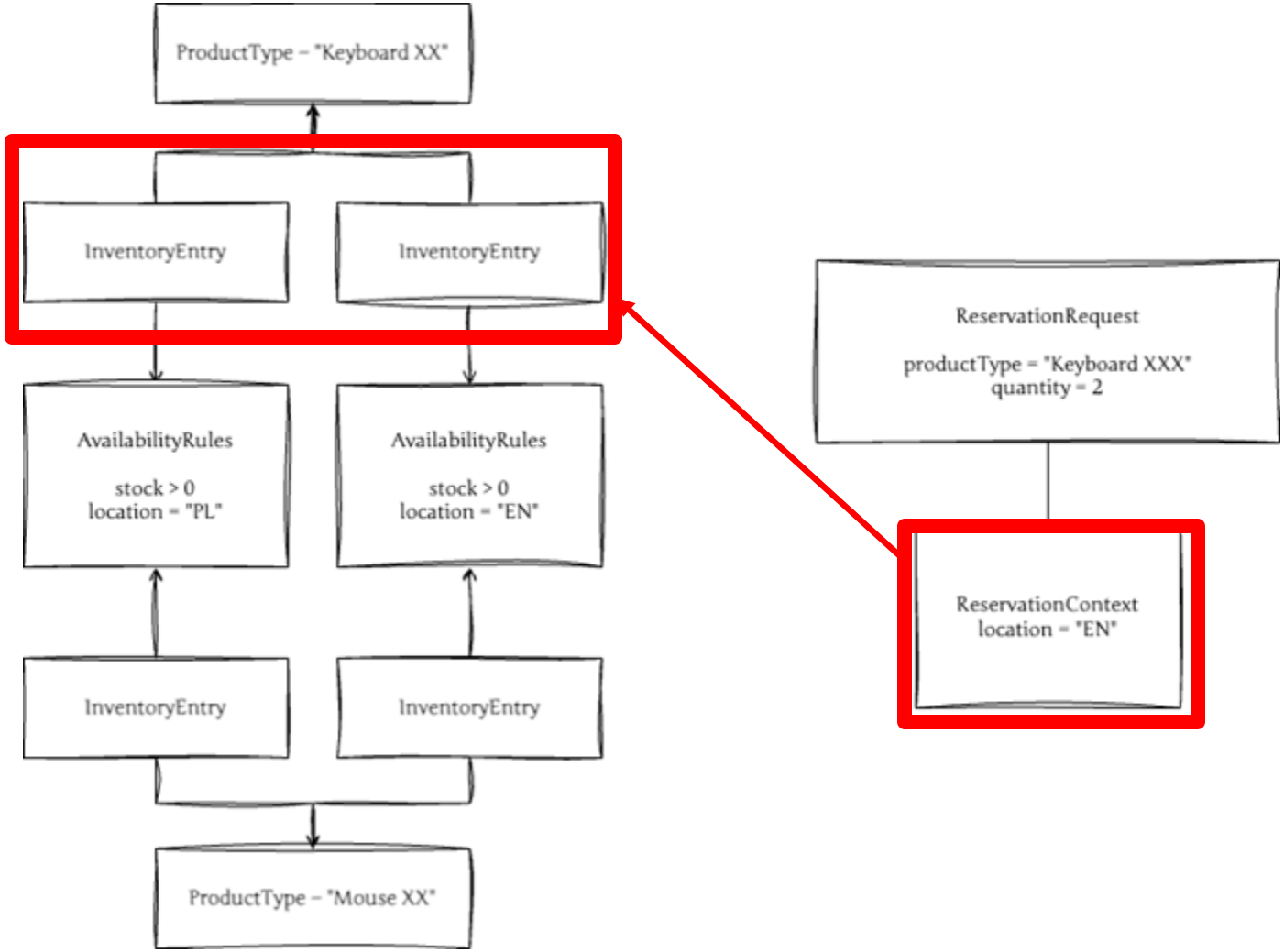

ReservationRequest

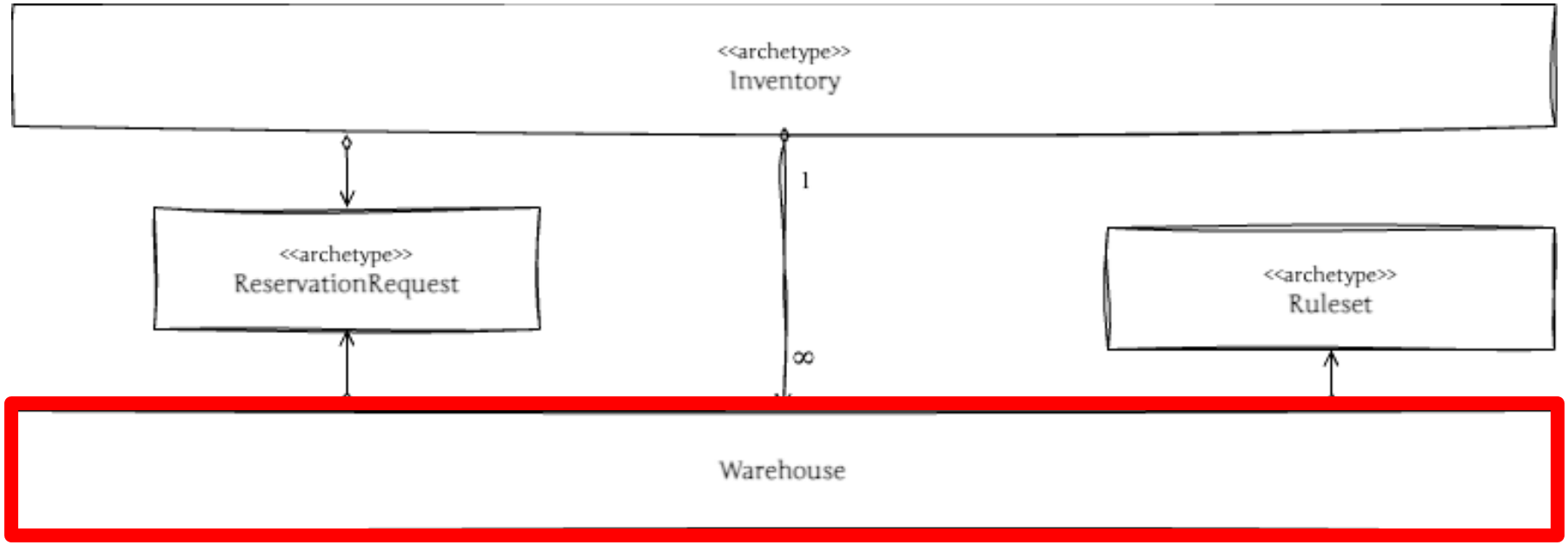

ProductType – "Keyboard XX"
InventoryEntry
InventoryEntry
AvailabilityRules
stock > 0
location = "PL"
AvailabilityRules
stock > 0
location = "EN"
InventoryEntry
InventoryEntry
ProductType – "Mouse XX"

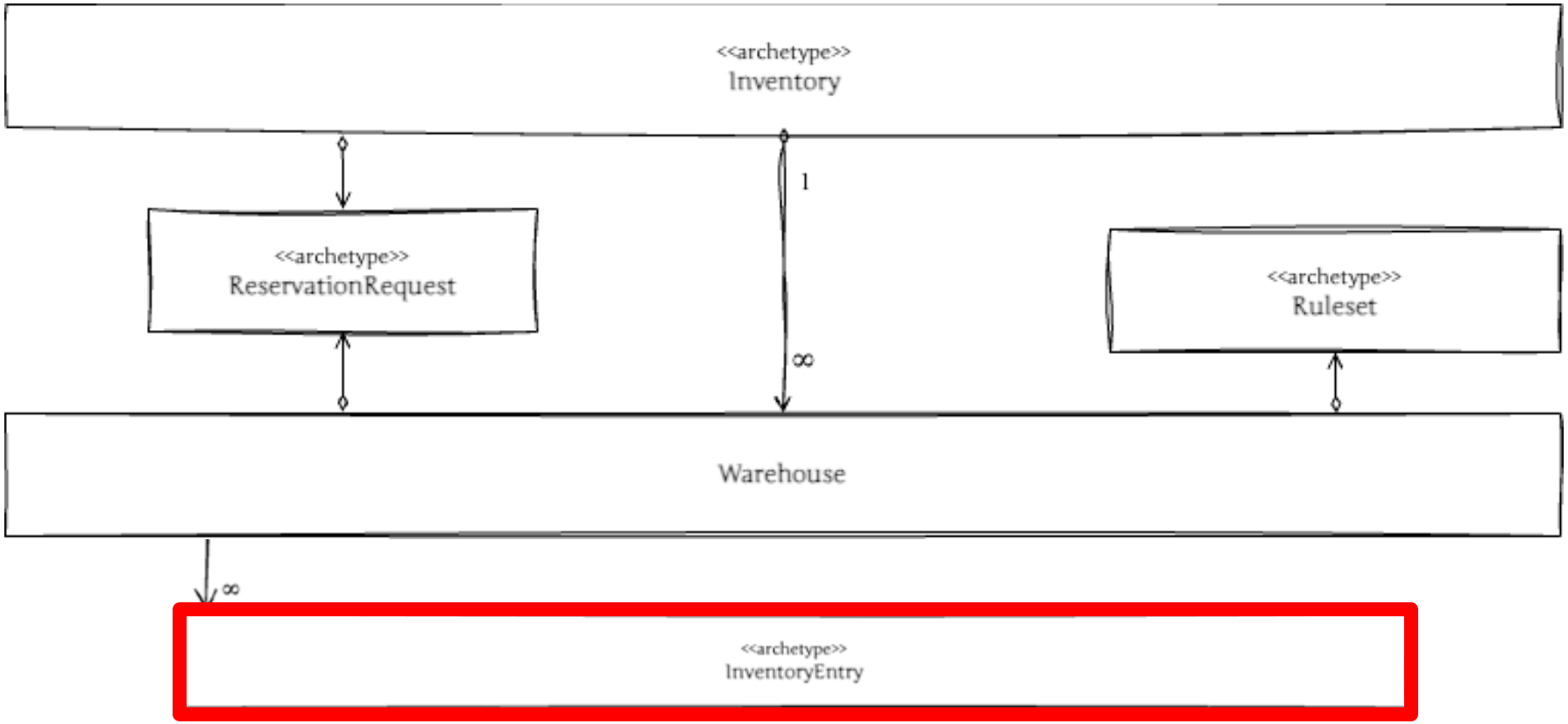

ReservationRequest
productType = "Keyboard XXX"
quantity = 2
ReservationContext
location = "EN"

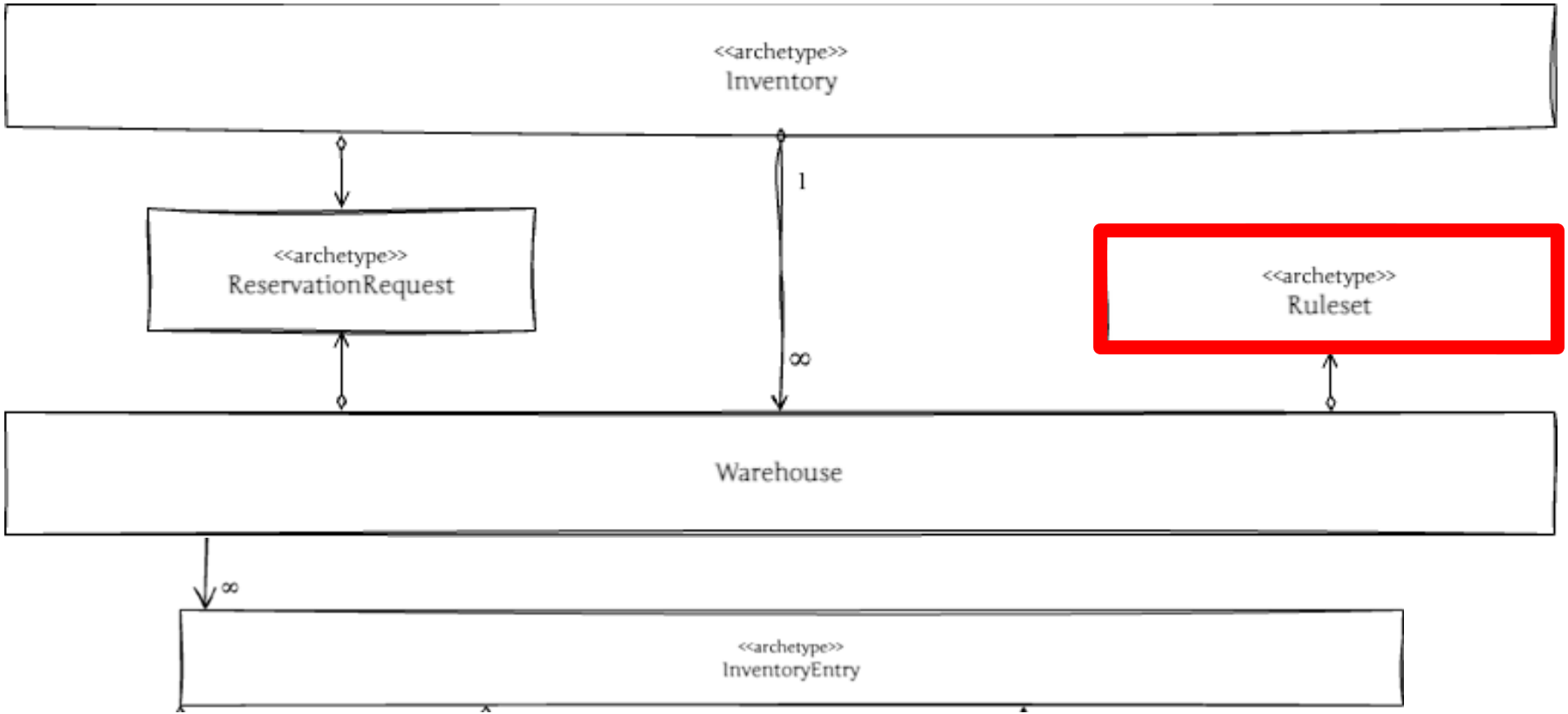

ProductType – "Keyboard XX"
InventoryEntry
InventoryEntry
AvailabilityRules
stock > 0
location = "PL"
AvailabilityRules
stock > 0
location = "EN"
InventoryEntry
InventoryEntry
ProductType – "Mouse XX"


ReservationRequest
productType = "Keyboard XXX"
quantity = 2
ReservationContext
location = "EN"


ProductType – "Keyboard XX"
InventoryEntry
InventoryEntry
AvailabilityRules
stock > 0
location = "PL"
AvailabilityRules
stock > 0
location = "EN"
InventoryEntry
InventoryEntry
ProductType – "Mouse XX"
ReservationRequest
productType = "Keyboard XXX"
quantity = 2
ReservationContext
location = "EN"


<<archetype>>
Inventory
1
<<archetype>>
ReservationRequest
∞
<<archetype>>
Ruleset
Warehouse


<<archetype>>
Inventory
1
<<archetype>>
ReservationRequest
∞
<<archetype>>
Ruleset
Warehouse
∞
<<archetype>>
InventoryEntry


<<archetype>>
Inventory
1
<<archetype>>
ReservationRequest
<<archetype>>
Ruleset
∞
Warehouse
∞
<<archetype>>
InventoryEntry

```php
readonly class Inventory
{
    new *
    public function __construct(private GenericList $warehouses)
    {
    }

    /**
     * @return Either<ReservationRejected, ReservationAccepted>
     */
    4 usages  new *
    public function makeReservation(ReservationRequest $request): Either
    {
        $warehouseOption = $this->warehouses->find(fn(Warehouse $warehouse) => $warehouse->canHandleRequest($request));

        return $warehouseOption->map(fn(Warehouse $warehouse) => $warehouse->makeReservation($request))
            ->getOrElse(Either::left(new ReservationRejected( reason: 'No warehouse can handle the request')));
    }
}
```

```php
readonly class Inventory
{
    new *
    public function __construct(private GenericList $warehouses)
    {
    }

    /**
     * @return Either<ReservationRejected, ReservationAccepted>
     */
    4 usages  new *
    public function makeReservation(ReservationRequest $request): Either
    {
        $warehouseOption = $this->warehouses->find(fn(Warehouse $warehouse) => $warehouse->canHandleRequest($request));

        return $warehouseOption->map(fn(Warehouse $warehouse) => $warehouse->makeReservation($request))
            ->getOrElse(Either::left(new ReservationRejected( reason: 'No warehouse can handle the request')));
    }
}
```

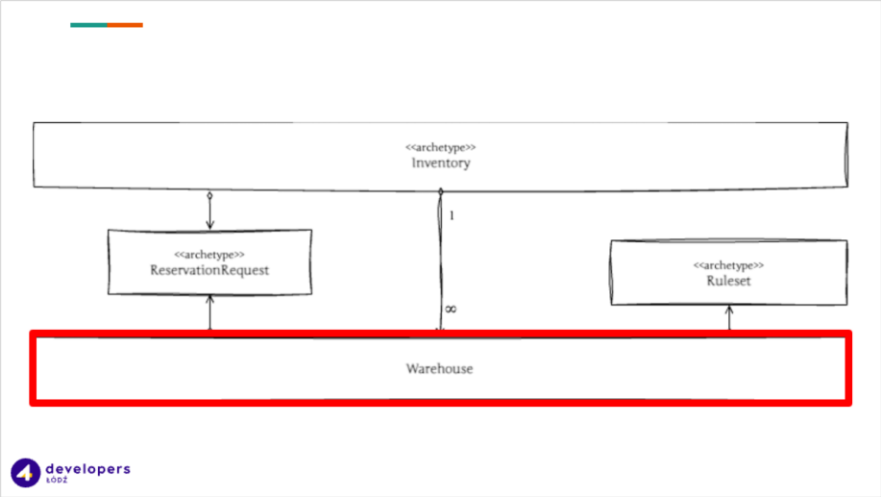
```php
class Warehouse
{
    /**
     * @param GenericList<InventoryEntry> $entries
     */
    5 usages  new *
    public function __construct(
        private readonly GenericList $entries,
        private Ruleset $handlingRules
    ) {}

    1 usage  new *
    public function canHandleRequest(ReservationRequest $request): bool
    {
        return $this->handlingRules->evaluate($request->reservationContext())->getValue();

    }

    2 usages  new *
    public function makeReservation(ReservationRequest $request): Either
    {
        return $this->entries->fold(
            Either::left(new ReservationRejected( reason: 'No products available')),
            fn(Either $either, InventoryEntry $entry) => $either->map(fn() => $either)->getOrElse($entry->makeReservation($request))

        );
    }
}
```

```php
class Warehouse
{
    /**
     * @param GenericList<InventoryEntry> $entries
     */
    5 usages  new *
    public function __construct(
        private readonly GenericList $entries,
        private Ruleset $handlingRules
    ) {}

    1 usage  new *
    public function canHandleRequest(ReservationRequest $request): bool
    {
        return $this->handlingRules->evaluate($request->reservationContext())->getValue();

    }

    2 usages  new *
    public function makeReservation(ReservationRequest $request): Either
    {
        return $this->entries->fold(
            Either::left(new ReservationRejected( reason: 'No products available')),
            fn(Either $either, InventoryEntry $entry) => $either->map(fn() => $either)->getOrElse($entry->makeReservation($request))

        );
    }
}
```

Rozbudowa systemu magazynowego

Inventory

Jeden typ produktu może być przechowywany w różnych magazynach.

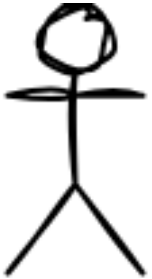


Dostępność produktu zależy od różnych czynników.

Możliwość śledzenia historii inwentarzu

Zarządzanie rolami

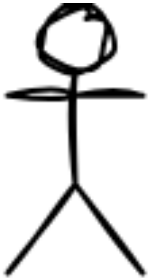
Party
PartyRelationship



Researcher

Reviever

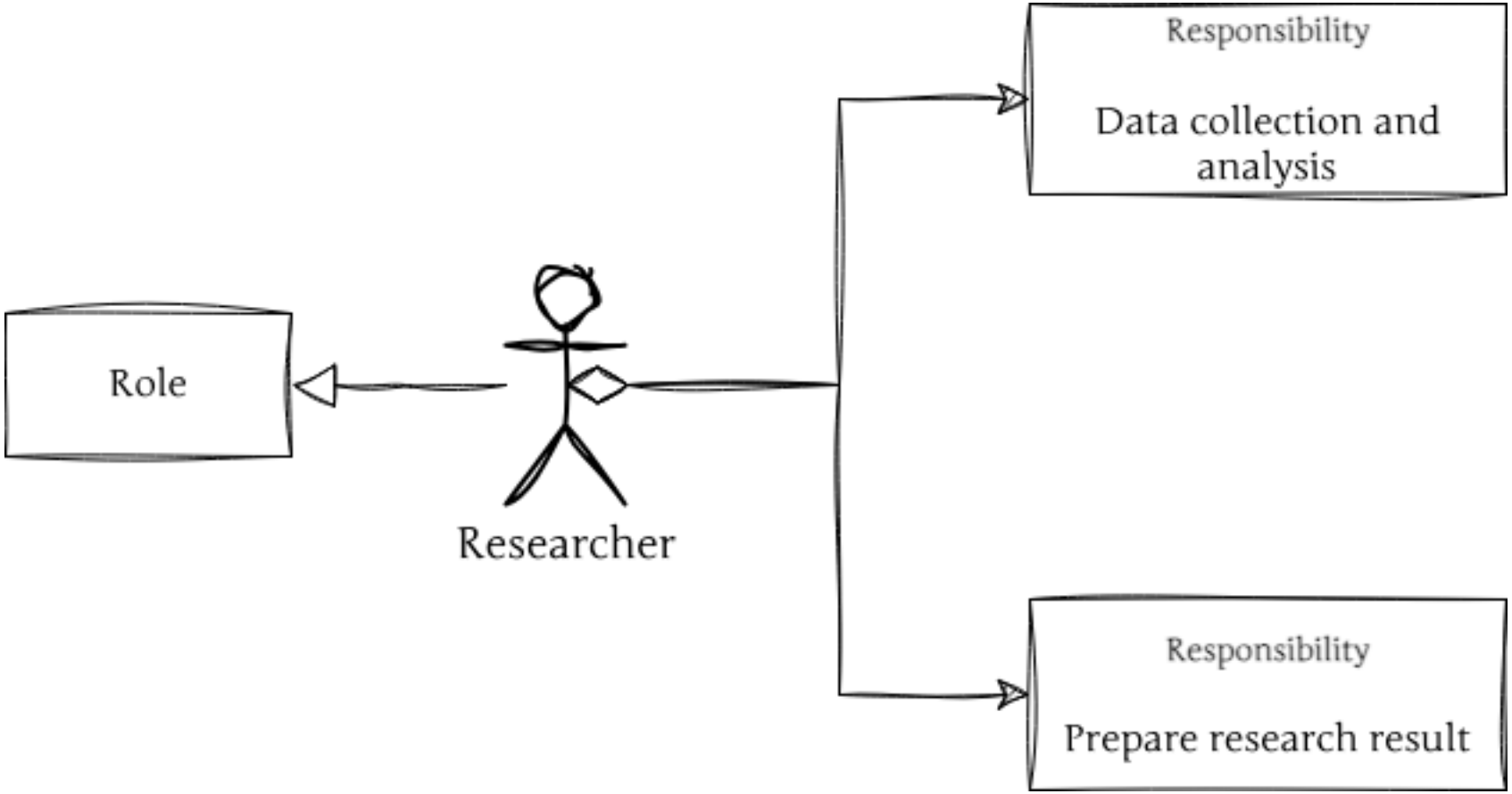
Badacz – osoba odpowiedzialna za prowadzenie badań naukowych zgodnie z planem projektu.

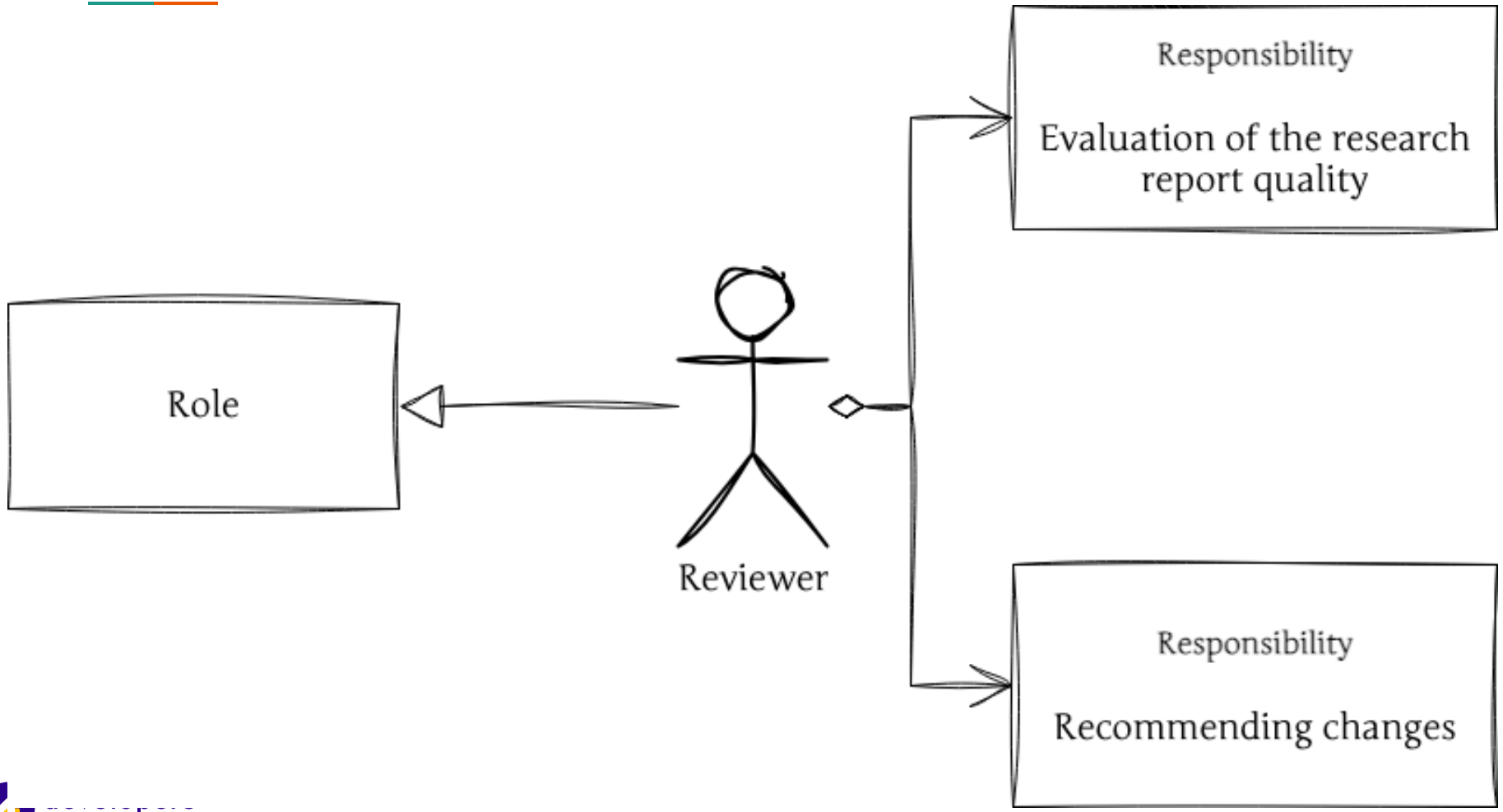


Researcher

Reviever

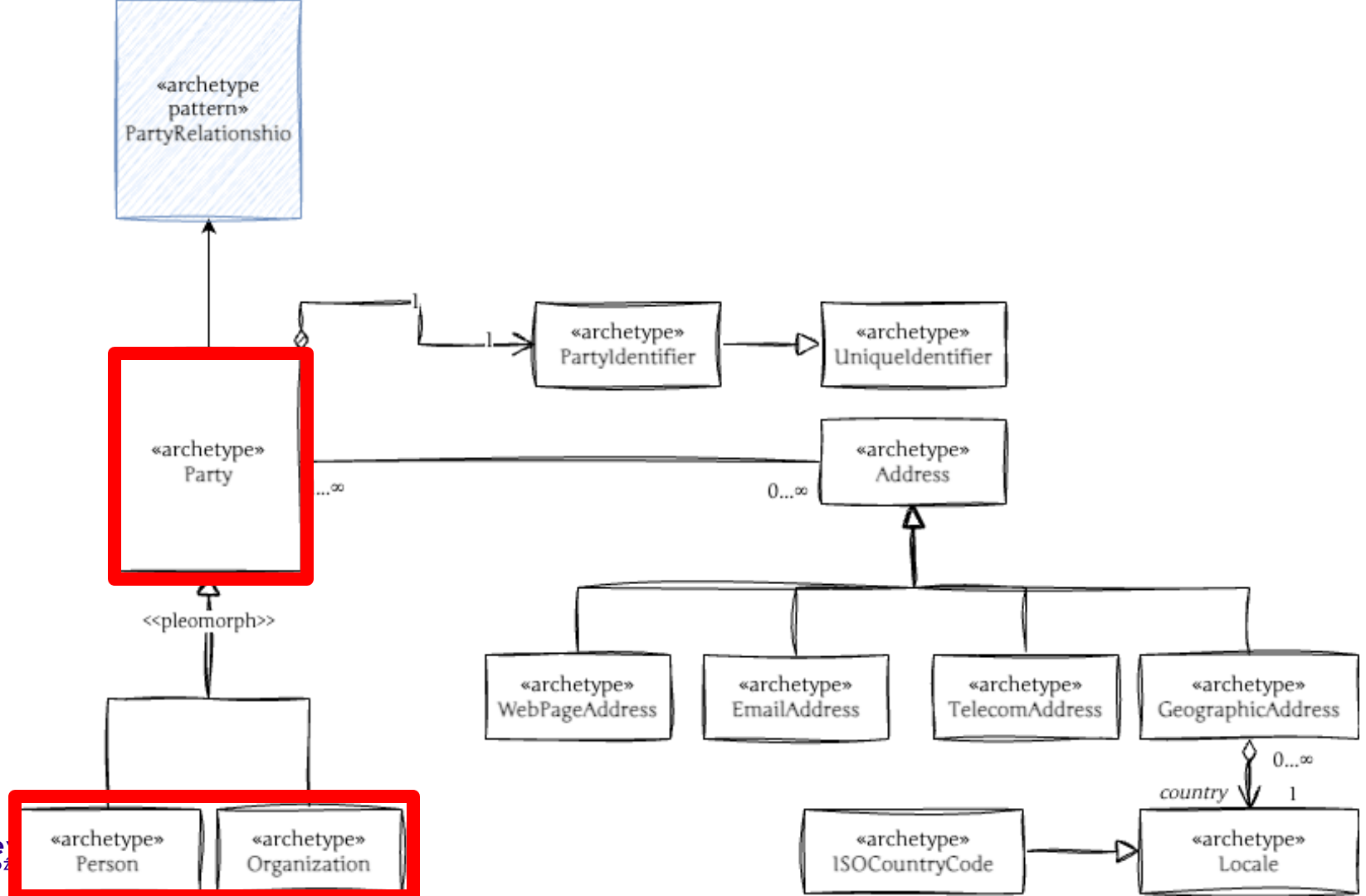
Recenzent – ekspert zewnętrzny odpowiedzialny za ocenę jakości pracy i zgodności wyników z założeniami projektu.

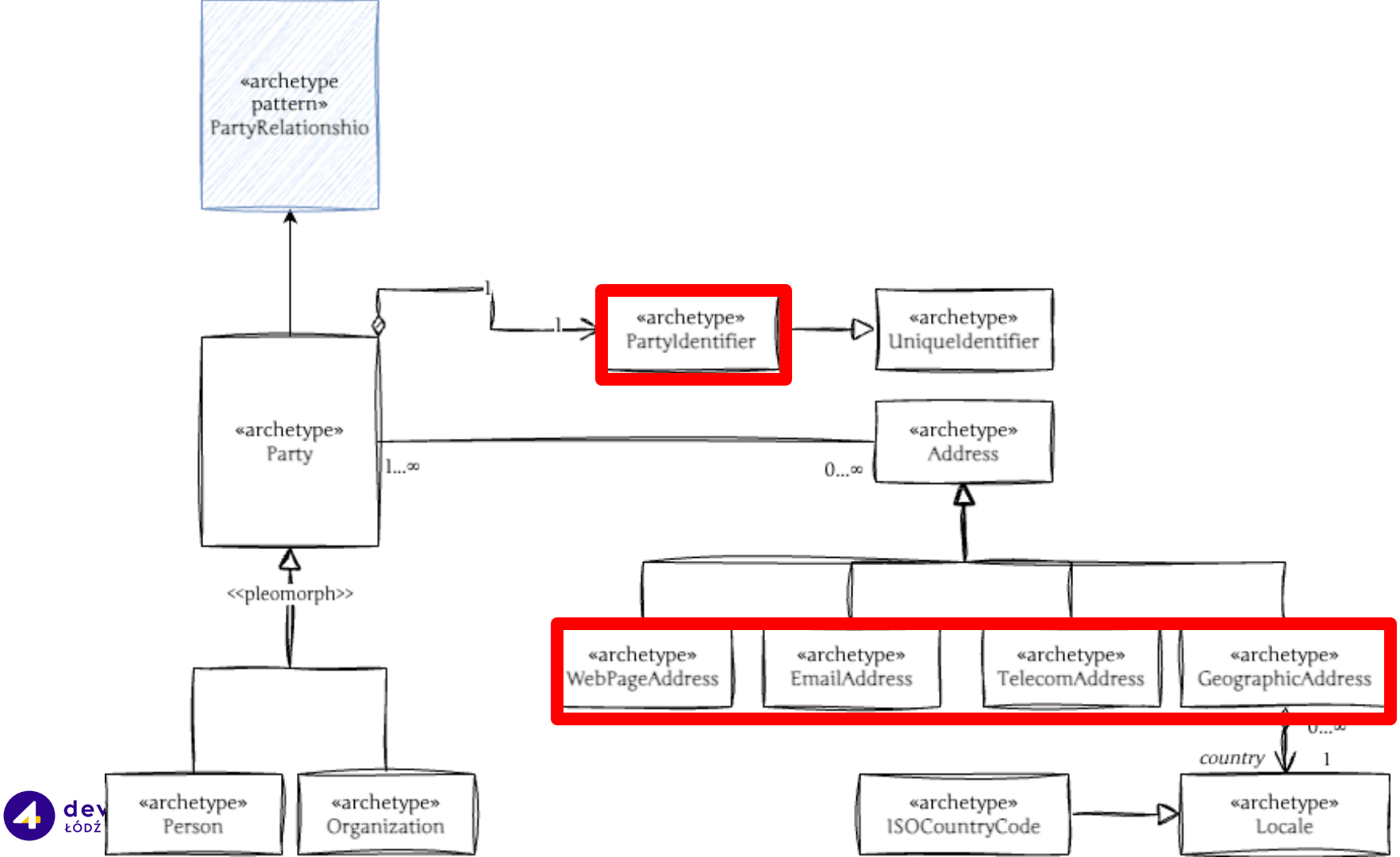

Responsibility
Data collection and analysis
Role
Researcher
Responsibility
Prepare research result

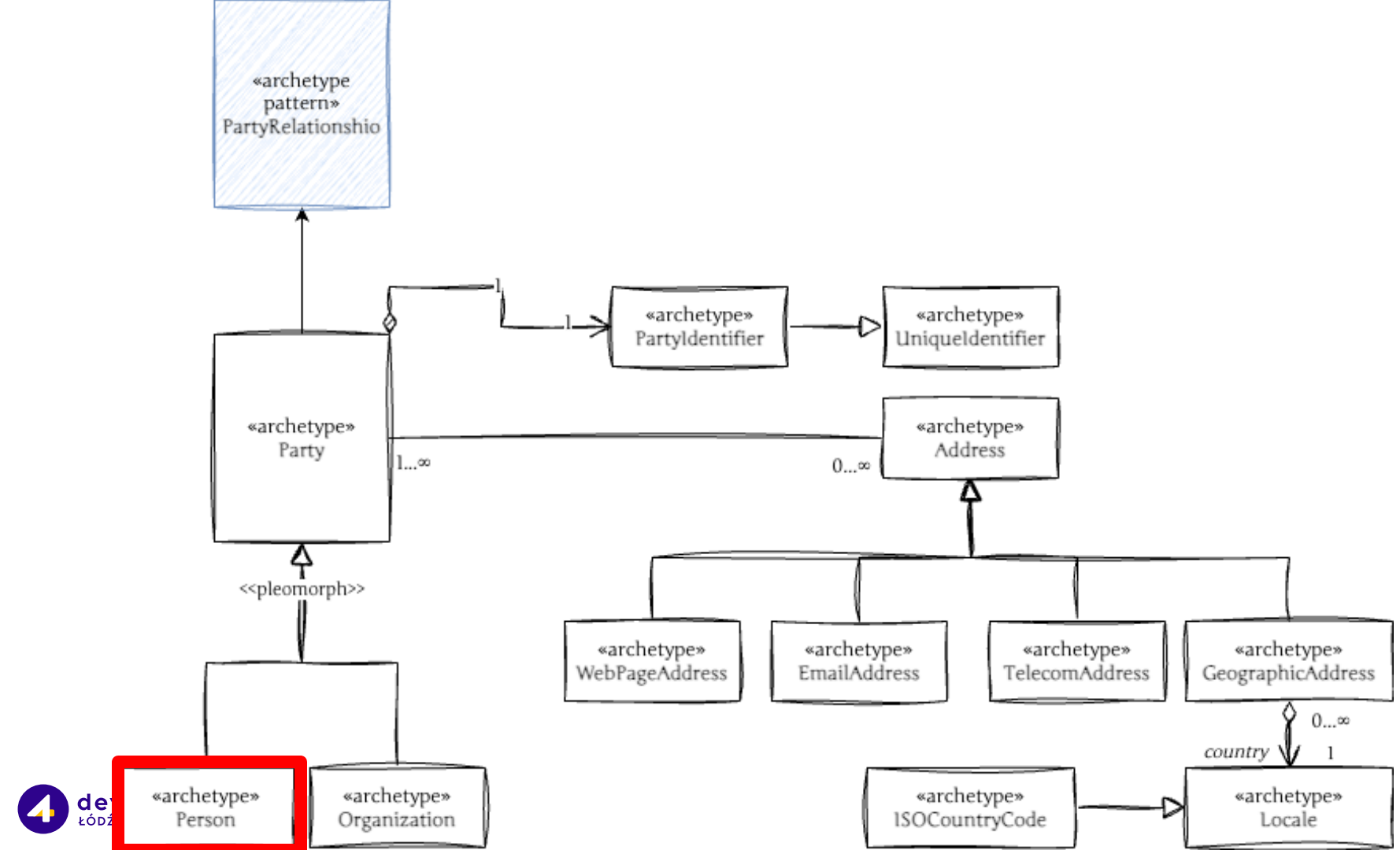

Responsibility
Evaluation of the research report quality
Role
Reviewer
Responsibility
Recommending changes

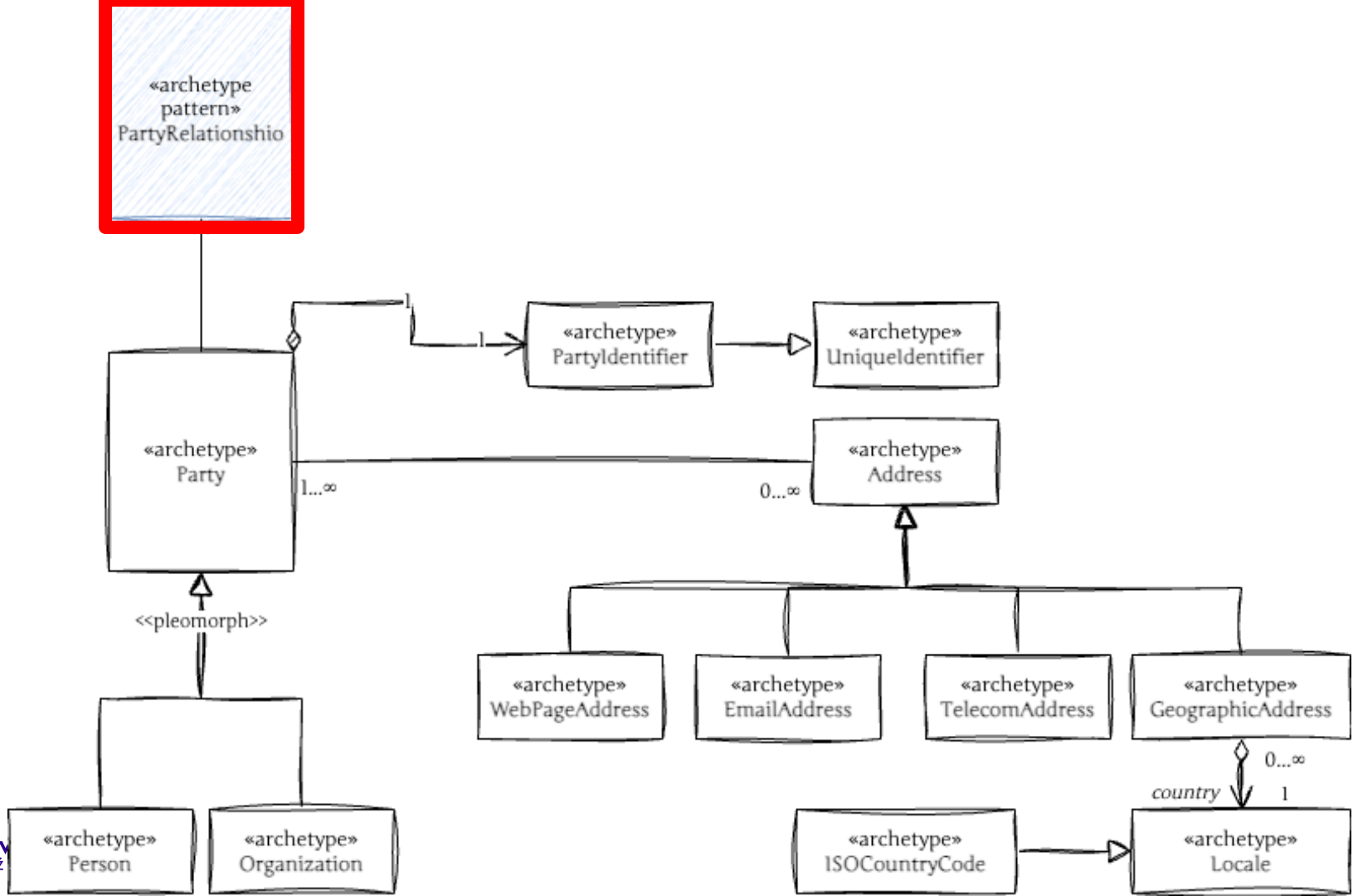
Zarządzanie rolami

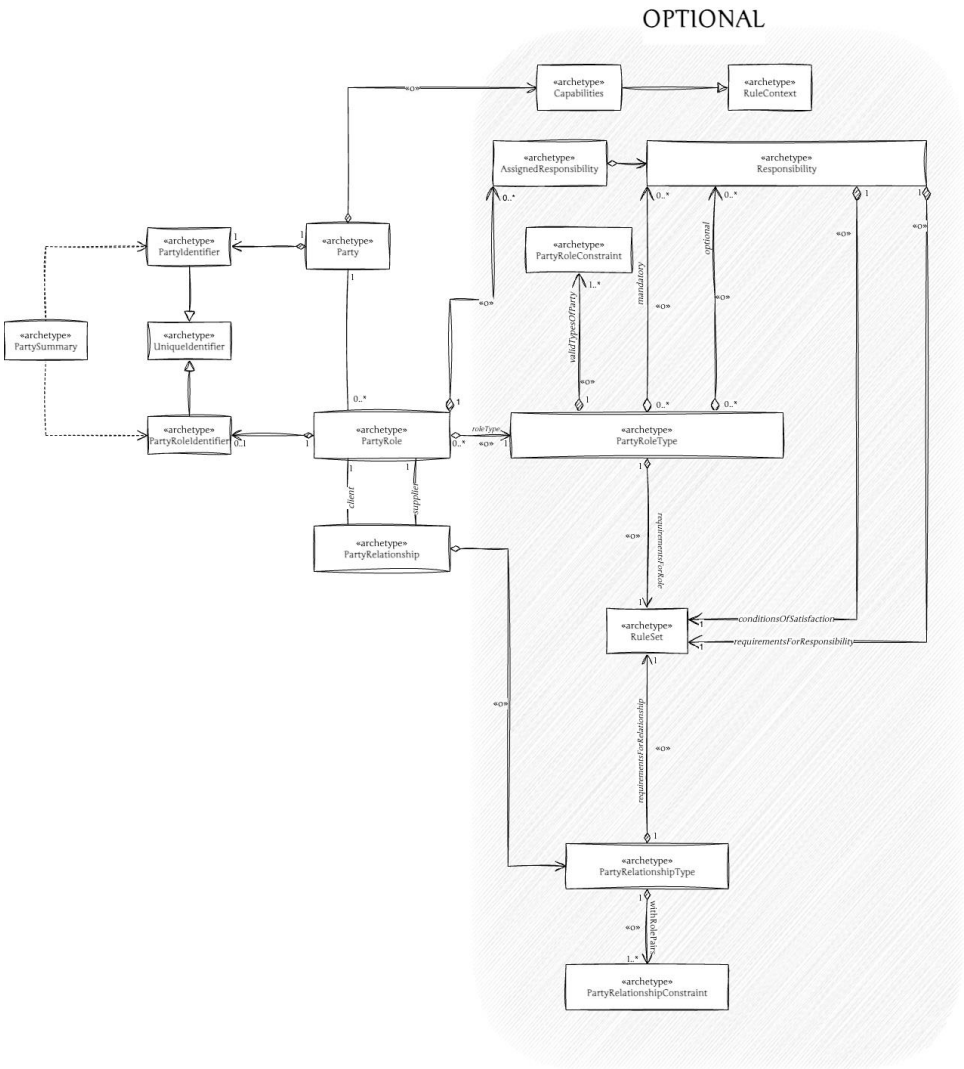
Party
PartyRelationship


«archetype pattern» PartyRelationshio
«archetype» Party
«archetype» PartyIdentifier
«archetype» UniqueIdentifier
«archetype» Address
1
1
...∞
0...∞
<<pleomorph>>
«archetype» Person
«archetype» Organization
«archetype» WebPageAddress
«archetype» EmailAddress
«archetype» TelecomAddress
«archetype» GeographicAddress
0...∞
country
1
«archetype» ISOCountryCode
«archetype» Locale

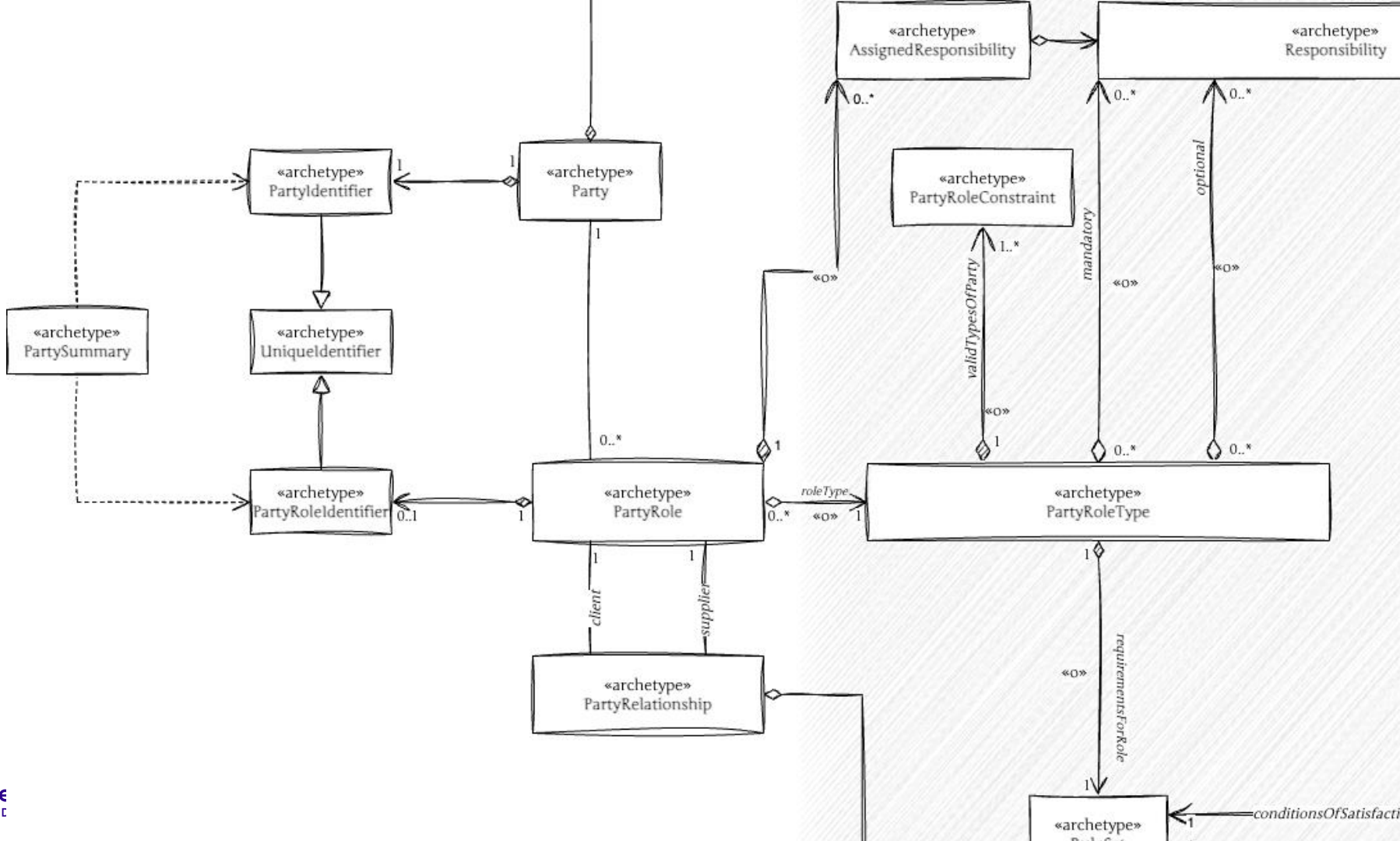

«archetype pattern» PartyRelationshio
1
1
«archetype» PartyIdentifier
«archetype» UniqueIdentifier
«archetype» Party
1...∞
0...∞
«archetype» Address
<<pleomorph>>
«archetype» WebPageAddress
«archetype» EmailAddress
«archetype» TelecomAddress
«archetype» GeographicAddress
country
1
«archetype» Person
«archetype» Organization
«archetype» ISOCountryCode
«archetype» Locale

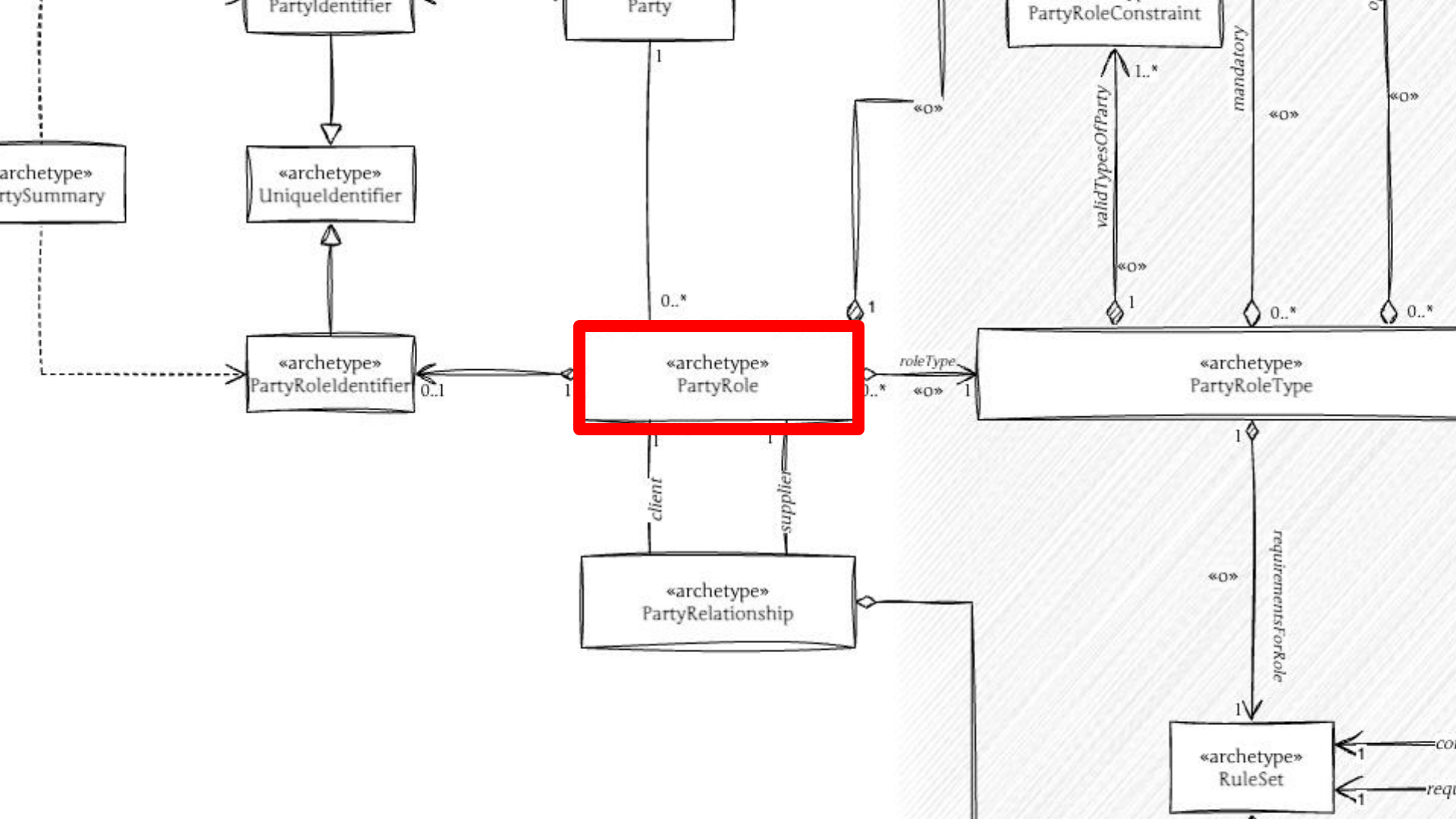

«archetype
pattern»
PartyRelationshio
1
1
«archetype»
PartyIdentifier
«archetype»
UniqueIdentifier
«archetype»
Party
1...∞
0...∞
«archetype»
Address
<<pleomorph>>
«archetype»
WebPageAddress
«archetype»
EmailAddress
«archetype»
TelecomAddress
«archetype»
GeographicAddress
0...∞
country
1
«archetype»
Person
«archetype»
Organization
«archetype»
ISOCountryCode
«archetype»
Locale

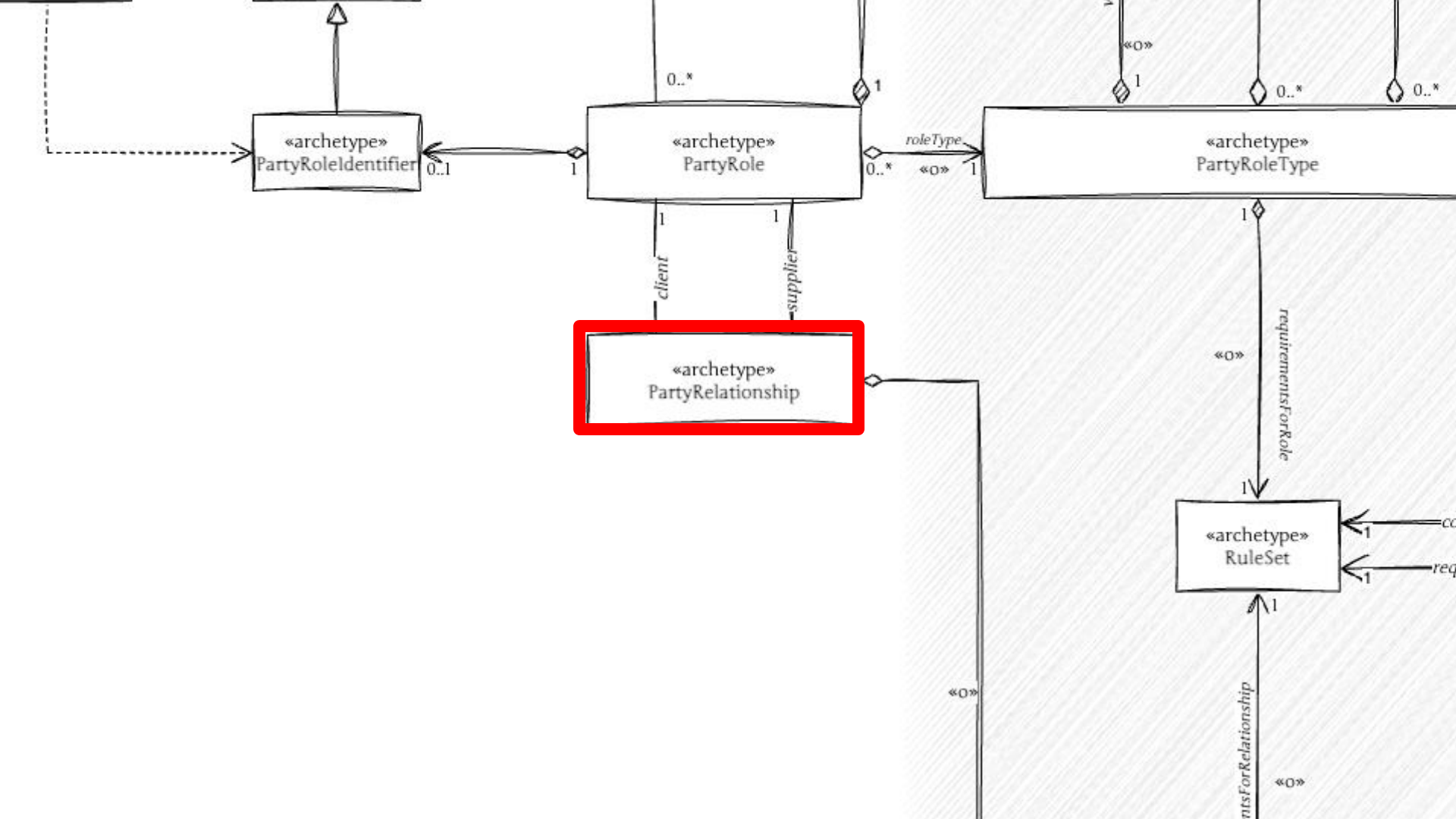

«archetype
pattern»
PartyRelationshio
1
1
«archetype»
PartyIdentifier
«archetype»
UniqueIdentifier
«archetype»
Party
1...∞
0...∞
«archetype»
Address
<<pleomorph>>
«archetype»
WebPageAddress
«archetype»
EmailAddress
«archetype»
TelecomAddress
«archetype»
GeographicAddress
0...∞
country
1
«archetype»
Person
«archetype»
Organization
«archetype»
ISOCountryCode
«archetype»
Locale

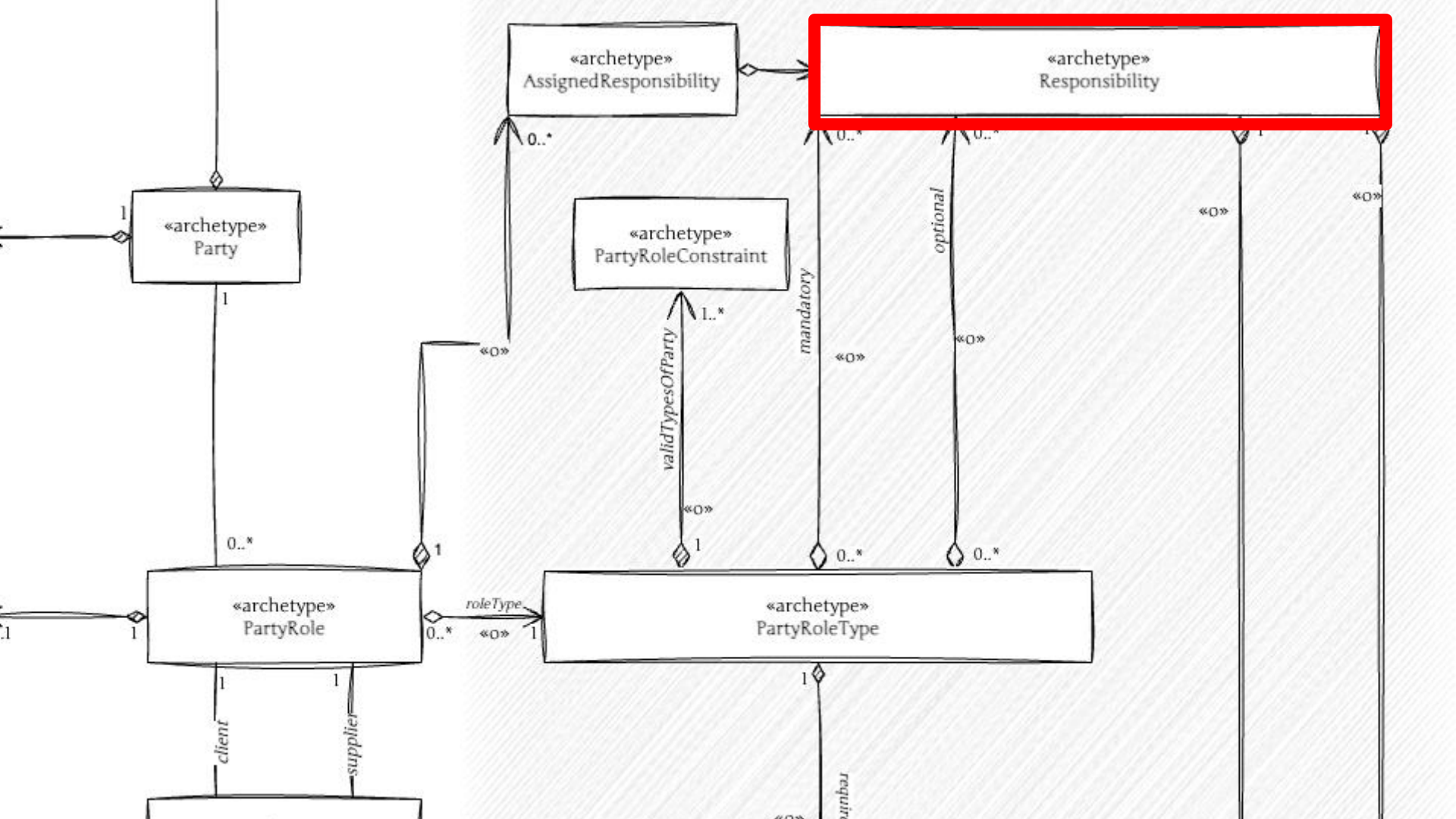


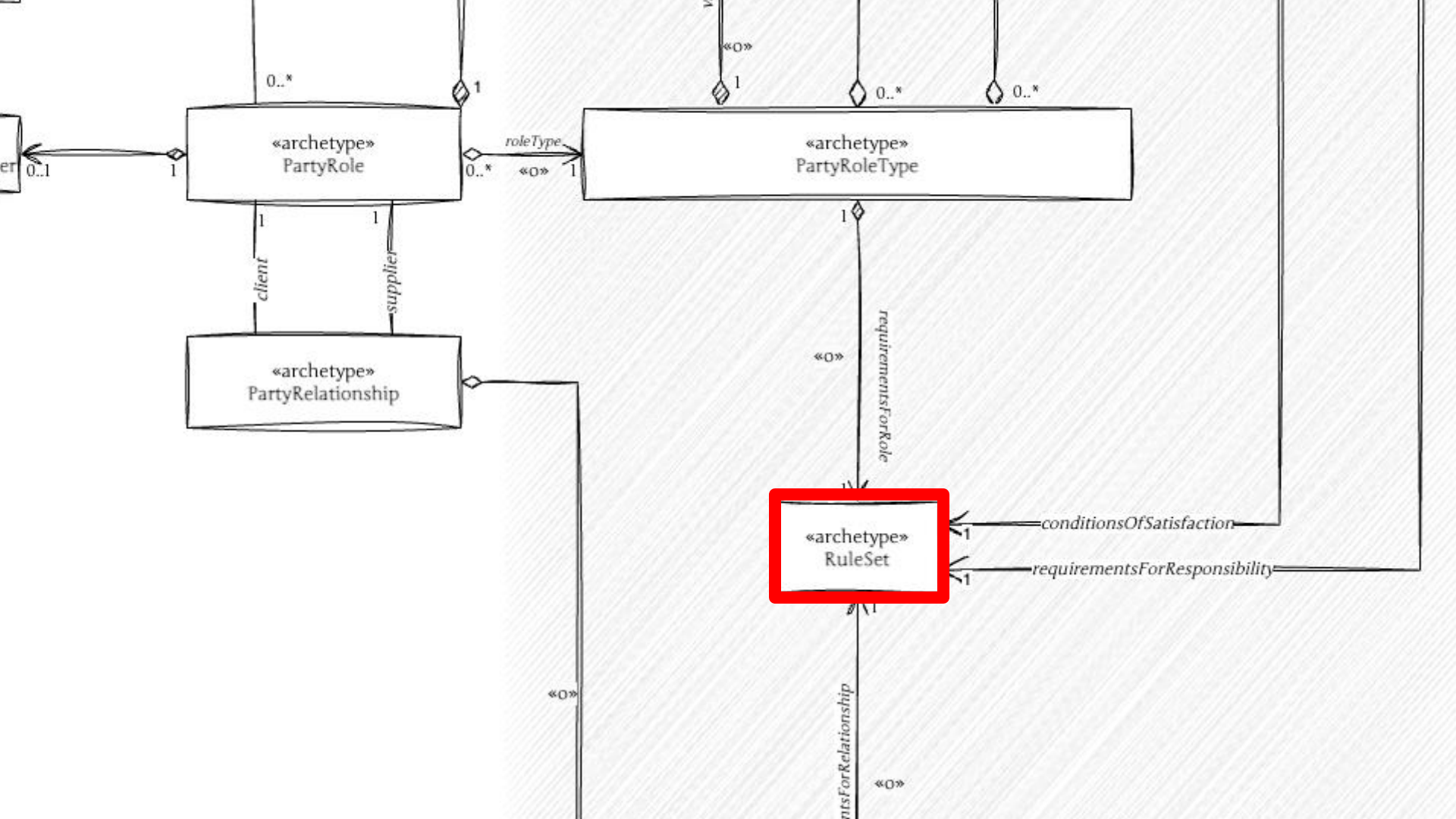
developers ŁÓDŹ


«archetype»
AssignedResponsibility
«archetype»
Responsibility
0..*
0..*
0..*
«archetype»
PartyIdentifier
1
1
«archetype»
Party
«archetype»
PartyRoleConstraint
optional
mandatory
validTypesOfParty
1..*
«o»
«o»
«o»
«archetype»
PartySummary
«archetype»
UniqueIdentifier
1
0..*
«archetype»
PartyRoleIdentifier
0..1
1
«archetype»
PartyRole
0..*
«o»
1
roleType
«archetype»
PartyRoleType
0..*
0..*
1
1
1
client
supplier
«archetype»
PartyRelationship
«o»
requirementsForRole
1
«archetype»
conditionsOfSatisfacti


PartyIdentifier
Party
PartyRoleConstraint
«archetype»
rtySummary
«archetype»
UniqueIdentifier
validTypesOfParty
mandatory
«archetype»
PartyRoleIdentifier
«archetype»
PartyRole
roleType
«archetype»
PartyRoleType
client
supplier
«archetype»
PartyRelationship
requirementsForRole
«archetype»
RuleSet


«archetype»
PartyRoleIdentifier
«archetype»
PartyRole
«archetype»
PartyRoleType
«archetype»
PartyRelationship
«archetype»
RuleSet
roleType
client
supplier
requirementsForRole
0..1
0..*
1
«0»


«archetype»
AssignedResponsibility
«archetype»
Responsibility
«archetype»
Party
«archetype»
PartyRoleConstraint
«archetype»
PartyRole
«archetype»
PartyRoleType
validTypesOfParty
mandatory
optional
roleType
client
supplier
0..*
1..*
1
«O»


«archetype»
PartyRole
roleType
«archetype»
PartyRoleType
«archetype»
PartyRelationship
«archetype»
RuleSet
client
supplier
requirementsForRole
conditionsOfSatisfaction
requirementsForResponsibility
«o»
0..*
1
0..1

```php
abstract class Party
{
    /**
     * Lista ról przypisanych do strony.
     * @var GenericList<Role>
     */
    private GenericList $roles;

    public function __construct()
    {
        $this->roles = GenericList::empty();
    }

    public function getRoles(): GenericList
    {
        return $this->roles;
    }

    public function addRole(Role $role): void
    {
        $this->roles = $this->roles->append($role);
    }
}
```

```php
abstract class Party
{
    /**
     * Lista ról przypisanych do strony.
     * @var GenericList<Role>
     */
    private GenericList $roles;

    public function __construct()
    {
        $this->roles = GenericList::empty();
    }

    public function getRoles(): GenericList
    {
        return $this->roles;
    }

    public function addRole(Role $role): void
    {
        $this->roles = $this->roles->append($role);
    }
}
```

```php
abstract class Role
{
    /**
     * Lista odpowiedzialności przypisanych do roli.
     * @var GenericList<Responsibility>
     */
    private array $responsibilities;

    public function getResponsibilities(): array
    {
        return $this->responsibilities;
    }

    public function addResponsibility(Responsibility $responsibility): self
    {
        $this->responsibilities = $this->responsibilities->append($responsibility);
    }

    public function executeResponsibilities(RuleContext $context): void
    {
        $this->responsibilities
            ->filter(fn(Responsibility $responsibility) => $responsibility->canAssign($context))
            ->forEach(fn(Responsibility $responsibility) => $responsibility->execute());
    }
}
```

```php
abstract class Role
{
    /**
     * Lista odpowiedzialności przypisanych do roli
     * @var GenericList<Responsibility>
     */
    private array $responsibilities;

    public function getResponsibilities(): array
    {
        return $this->responsibilities;
    }

    public function addResponsibility(Responsibility $responsibility): self
    {
        $this->responsibilities = $this->responsibilities->append($responsibility);
    }

    public function executeResponsibilities(RuleContext $context): void
    {
        $this->responsibilities
            ->filter(fn(Responsibility $responsibility) => $responsibility->canAssign($context))
            ->forEach(fn(Responsibility $responsibility) => $responsibility->execute());
    }
}
```

```
class ScienceEmployee extends Party
{
}
```

```
class ScienceEmployee extends Party
{
}

class RoleOfScientist extends Role
{
}
```

```php
readonly class Responsibility
{
    public function __construct(
        public string $description,
        public bool $isMandatory,
        private Ruleset $conditionsOfSatisfaction,
        private Ruleset $conditionsForResponsibility,
        private ?\Closure $executionLogic = null
    ) {}

    /**
     * Sprawdza, czy warunki przypisania odpowiedzialności są spełnione.
     */
    public function canAssign(RuleContext $context): bool
    {
        return $this->conditionsForResponsibility === null
            || $this->conditionsForResponsibility->evaluate($context);
    }

    /**
     * Wykonuje logikę odpowiedzialności.
     */
    public function execute(): void
    {
        if ($this->executionLogic === null) {
            throw new Exception('No execution logic defined for this responsibility.');
        }

        $this->executionLogic($this->conditionsOfSatisfaction);
    }
}
```

```php
readonly class Responsibility
{
    public function __construct(
        public string $description,
        public bool $isMandatory,
        private Ruleset $conditionsOfSatisfaction,
        private Ruleset $conditionsForResponsibility,
        private ?\Closure $executionLogic = null
    ) {}

    /**
     * Sprawdza, czy warunki przypisania odpowiedzialności są spełnione.
     */
    public function canAssign(RuleContext $context): bool
    {
        return $this->conditionsForResponsibility === null
```

```php
readonly class Responsibility
{
    public function __construct(
        public string $description,
        public bool $isMandatory,
        private Ruleset $conditionsOfSatisfaction,
        private Ruleset $conditionsForResponsibility,
        private ?\Closure $executionLogic = null
    ) {}

    /**
     * Sprawdza, czy warunki przypisania odpowiedzialności są spełnione.
     */
    public function canAssign(RuleContext $context): bool
    {
        return $this->conditionsForResponsibility === null
```

```php
readonly class Responsibility
{
    public function __construct(
        public string $description,
        public bool $isMandatory,
        private Ruleset $conditionsOfSatisfaction,
        private Ruleset $conditionsForResponsibility,
        private ?\Closure $executionLogic = null
    ) {}

    /**
     * Sprawdza, czy warunki przypisania odpowiedzialności są spełnione.
     */
    public function canAssign(RuleContext $context): bool
    {
        return $this->conditionsForResponsibility === null
```

```php
readonly class Responsibility
{
    public function __construct(
        public string $description,
        public bool $isMandatory,
        private Ruleset $conditionsOfSatisfaction,
        private Ruleset $conditionsForResponsibility,
        private ?\Closure $executionLogic = null
    ) {}

    /**
     * Sprawdza, czy warunki przypisania odpowiedzialności są spełnione.
     */
    public function canAssign(RuleContext $context): bool
    {
        return $this->conditionsForResponsibility === null
```

```php
readonly class Responsibility
{
    public function __construct(
        public string $description,
        public bool $isMandatory,
        private Ruleset $conditionsOfSatisfaction,
        private Ruleset $conditionsForResponsibility,
        private ?\Closure $executionLogic = null
    ) {}

    /**
     * Sprawdza, czy warunki przypisania odpowiedzialności są spełnione.
     */
    public function canAssign(RuleContext $context): bool
    {
        return $this->conditionsForResponsibility === null
```

```php
$analyzeDataResponsibility = new Responsibility(
    description: 'Analyze research data',
    isMandatory: true,
    // Ruleset określający warunki satysfakcji odpowiedzialności
    conditionsOfSatisfaction: $analyzeDataConditionsOfSatisfaction,
    // Ruleset określający warunki przypisania odpowiedzialności
    conditionsForResponsibility: $analyzeDataConditionsForResponsibility,
    // Tutaj logika analizy danych
    executionLogic: fn(Ruleset $conditionsOfSatisfaction) => 1+1
);

$prepareReportResponsibility = new Responsibility(
    description: 'Prepare research report',
    isMandatory: true,
    conditionsOfSatisfaction: $prepareReportConditionsOfSatisfaction,
    conditionsForResponsibility: $prepareReportConditionsForResponsibility,
    // Tutaj logika przygotowania raportu
    executionLogic:  fn(Ruleset $conditionsOfSatisfaction) => 2+2
);
```

```php
$analyzeDataResponsibility = new Responsibility(
    description: 'Analyze research data',
    isMandatory: true,
    // Ruleset określający warunki satysfakcji odpowiedzialności
    conditionsOfSatisfaction: $analyzeDataConditionsOfSatisfaction,
    // Ruleset określający warunki przypisania odpowiedzialności
    conditionsForResponsibility: $analyzeDataConditionsForResponsibility,
    // Tutaj logika analizy danych
    executionLogic: fn(Ruleset $conditionsOfSatisfaction) => 1+1
);
```

```php
$prepareReportResponsibility = new Responsibility(
    description: 'Prepare research report',
    isMandatory: true,
    conditionsOfSatisfaction: $prepareReportConditionsOfSatisfaction,
    conditionsForResponsibility: $prepareReportConditionsForResponsibility,
    // Tutaj logika przygotowania raportu
    executionLogic:  fn(Ruleset $conditionsOfSatisfaction) => 2+2
);
```

```php
$researcherRole = new RoleOfScientist();
$researcherRole->addResponsibility($analyzeDataResponsibility);
$researcherRole->addResponsibility($prepareReportResponsibility);
$concreteResearcherEmployee = new ScienceEmployee();
$concreteResearcherEmployee->addRole($researcherRole);

// Tworzenie nowego badania, z czapy faktorką,
// ale nie zmieścimy się w slajdzie inaczej
$research = Research::create();

$researchResult = $concreteResearcherEmployee->getRoles()
    ->executeResponsibilities($research->getContext());
```

```php
$researcherRole = new RoleOfScientist();
$researcherRole->addResponsibility($analyzeDataResponsibility);
$researcherRole->addResponsibility($prepareReportResponsibility);
$concreteResearcherEmployee = new ScienceEmployee();
$concreteResearcherEmployee->addRole($researcherRole);

// Tworzenie nowego badania, z czapy faktorką,
// ale nie zmieścimy się w slajdzie inaczej
$research = Research::create();

$researchResult = $concreteResearcherEmployee->getRoles()
    ->executeResponsibilities($research->getContext());
```

```php
$researcherRole = new RoleOfScientist();
$researcherRole->addResponsibility($analyzeDataResponsibility);
$researcherRole->addResponsibility($prepareReportResponsibility);
$concreteResearcherEmployee = new ScienceEmployee();
$concreteResearcherEmployee->addRole($researcherRole);

// Tworzenie nowego badania, z czapy faktorką,
// ale nie zmieścimy się w slajdzie inaczej
$research = Research::create();

$researchResult = $concreteResearcherEmployee->getRoles()
    ->executeResponsibilities($research->getContext());
```

```php
$researcherRole = new RoleOfScientist();
$researcherRole->addResponsibility($analyzeDataResponsibility);
$researcherRole->addResponsibility($prepareReportResponsibility);
$concreteResearcherEmployee = new ScienceEmployee();
$concreteResearcherEmployee->addRole($researcherRole);

// Tworzenie nowego badania, z czapy faktorką,
// ale nie zmieścimy się w slajdzie inaczej
$research = Research::create();

$researchResult = $concreteResearcherEmployee->getRoles()
    ->executeResponsibilities($research->getContext());
```

```php
$validRole = new Rule( name: 'validResponsibilityRule');
$validRole
    ->variable( name: 'role')
    ->variable( name: 'expected role', value: RoleOfScientist::class)
    ->equalTo();

$ownershipRule = new Rule( name: 'ownershipRule');
$ownershipRule
    ->variable( name: 'owner')
    ->variable( name: 'employee')
    ->notEqualTo();

$conditionsForResponsibility = new Ruleset($validRole, $ownershipRule);
$analyzeDataResponsibility = new Responsibility(
    // reszta
    conditionsForResponsibility: $conditionsForResponsibility,
);
```

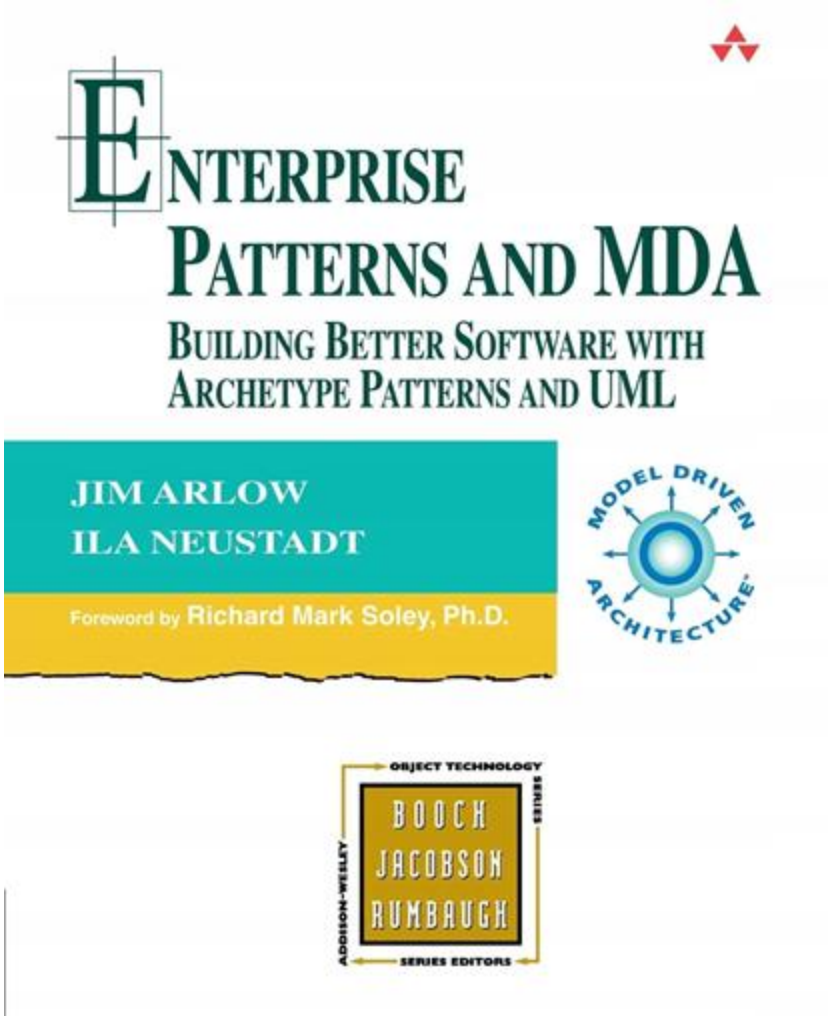
```php
$validRole = new Rule(name: 'validResponsibilityRule');
$validRole
    ->variable(name: 'role')
    ->variable(name: 'expected role', value: RoleOfScientist::class)
    ->equalTo();

$ownershipRule = new Rule(name: 'ownershipRule');
$ownershipRule
    ->variable(name: 'owner')
    ->variable(name: 'employee')
    ->notEqualTo();

$conditionsForResponsibility = new Ruleset($validRole, $ownershipRule);
$analyzeDataResponsibility = new Responsibility(
    // reszta
    conditionsForResponsibility: $conditionsForResponsibility,
);
```

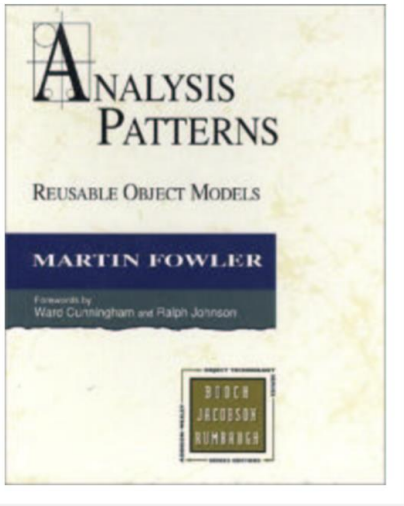
```php
$validRole = new Rule( name: 'validResponsibilityRule');
$validRole
    ->variable( name: 'role')
    ->variable( name: 'expected role', value: RoleOfScientist::class)
    ->equalTo();

$ownershipRule = new Rule( name: 'ownershipRule');
$ownershipRule
    ->variable( name: 'owner')
    ->variable( name: 'employee')
    ->notEqualTo();

$conditionsForResponsibility = new Ruleset($validRole, $ownershipRule);
$analyzeDataResponsibility = new Responsibility(
    // reszta
    conditionsForResponsibility: $conditionsForResponsibility,
);
```

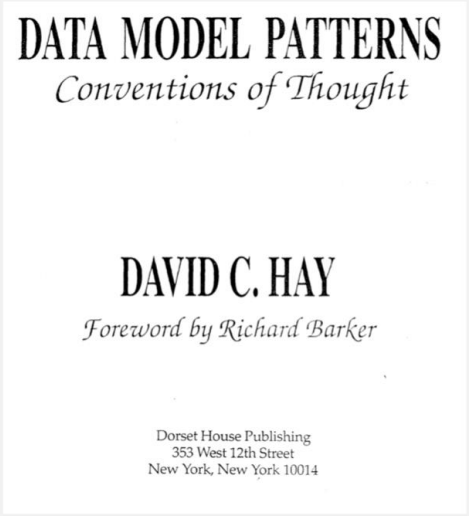
```php
$validRole = new Rule(name: 'validResponsibilityRule');
$validRole
    ->variable(name: 'role')
    ->variable(name: 'expected role', value: RoleOfScientist::class)
    ->equalTo();

$ownershipRule = new Rule(name: 'ownershipRule');
$ownershipRule
    ->variable(name: 'owner')
    ->variable(name: 'employee')
    ->notEqualTo();

$conditionsForResponsibility = new Ruleset($validRole, $ownershipRule);
$analyzeDataResponsibility = new Responsibility(
    // reszta
    conditionsForResponsibility: $conditionsForResponsibility,
);
```

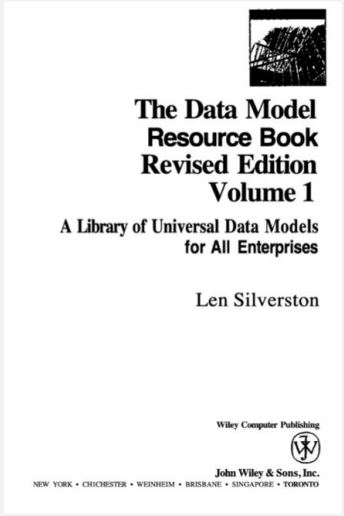
```php
$validRole = new Rule(name: 'validResponsibilityRule');
$validRole
    ->variable(name: 'role')
    ->variable(name: 'expected role', value: RoleOfScientist::class)
    ->equalTo();

$ownershipRule = new Rule(name: 'ownershipRule');
$ownershipRule
    ->variable(name: 'owner')
    ->variable(name: 'employee')
    ->notEqualTo();

$conditionsForResponsibility = new Ruleset($validRole, $ownershipRule);
$analyzeDataResponsibility = new Responsibility(
    // reszta
    conditionsForResponsibility: $conditionsForResponsibility,
);
```

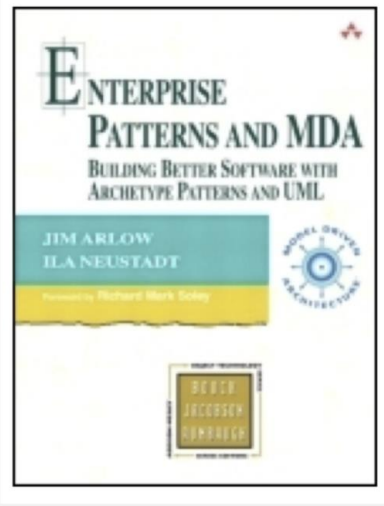
```php
readonly class Responsibility
{

    /**
     * Sprawdza, czy warunki przypisania odpowiedzialności są spełnione.
     */
    public function canAssign(RuleContext $context): bool
    {
        return $this->conditionsForResponsibility === null
            || $this->conditionsForResponsibility->evaluate($context);
    }
```


ENTERPRISE
PATTERNS AND MDA
BUILDING BETTER SOFTWARE WITH
ARCHETYPE PATTERNS AND UML
JIM ARLOW
ILA NEUSTADT
Foreword by Richard Mark Soley, Ph.D.
MODEL DRIVEN
ARCHITECTURE
OBJECT TECHNOLOGY SERIES
BOOCH
JACOBSON
RUMBAUGH
ADDISON-WESLEY
SERIES EDITORS


ANALYSIS PATTERNS
REUSABLE OBJECT MODELS
MARTIN FOWLER
Foreword by
Ward Cunningham and Ralph Johnson
BOOCH
JACOBSON
RUMBAUGH


DATA MODEL PATTERNS
Conventions of Thought
DAVID C. HAY
Foreword by Richard Barker
Dorset House Publishing
353 West 12th Street
New York, New York 10014


The Data Model
Resource Book
Revised Edition
Volume 1
A Library of Universal Data Models
for All Enterprises
Len Silverston
Wiley Computer Publishing
John Wiley & Sons, Inc.
NEW YORK • CHICHESTER • WEINHEIM • BRISBANE • SINGAPORE • TORONTO

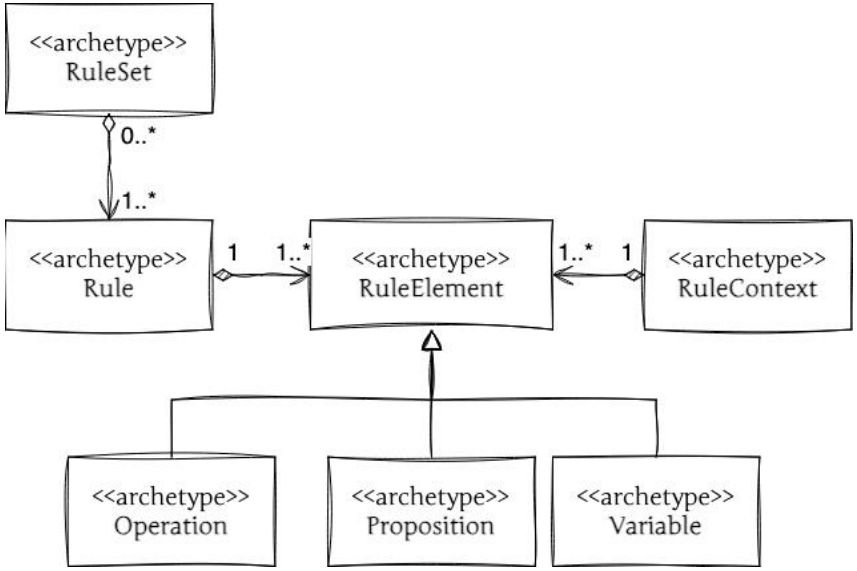

ENTERPRISE
PATTERNS AND MDA
BUILDING BETTER SOFTWARE WITH
ARCHETYPE PATTERNS AND UML
JIM ARLOW
ILA NEUSTADT
BOOCH
JACOBSON
RUMBAUGH

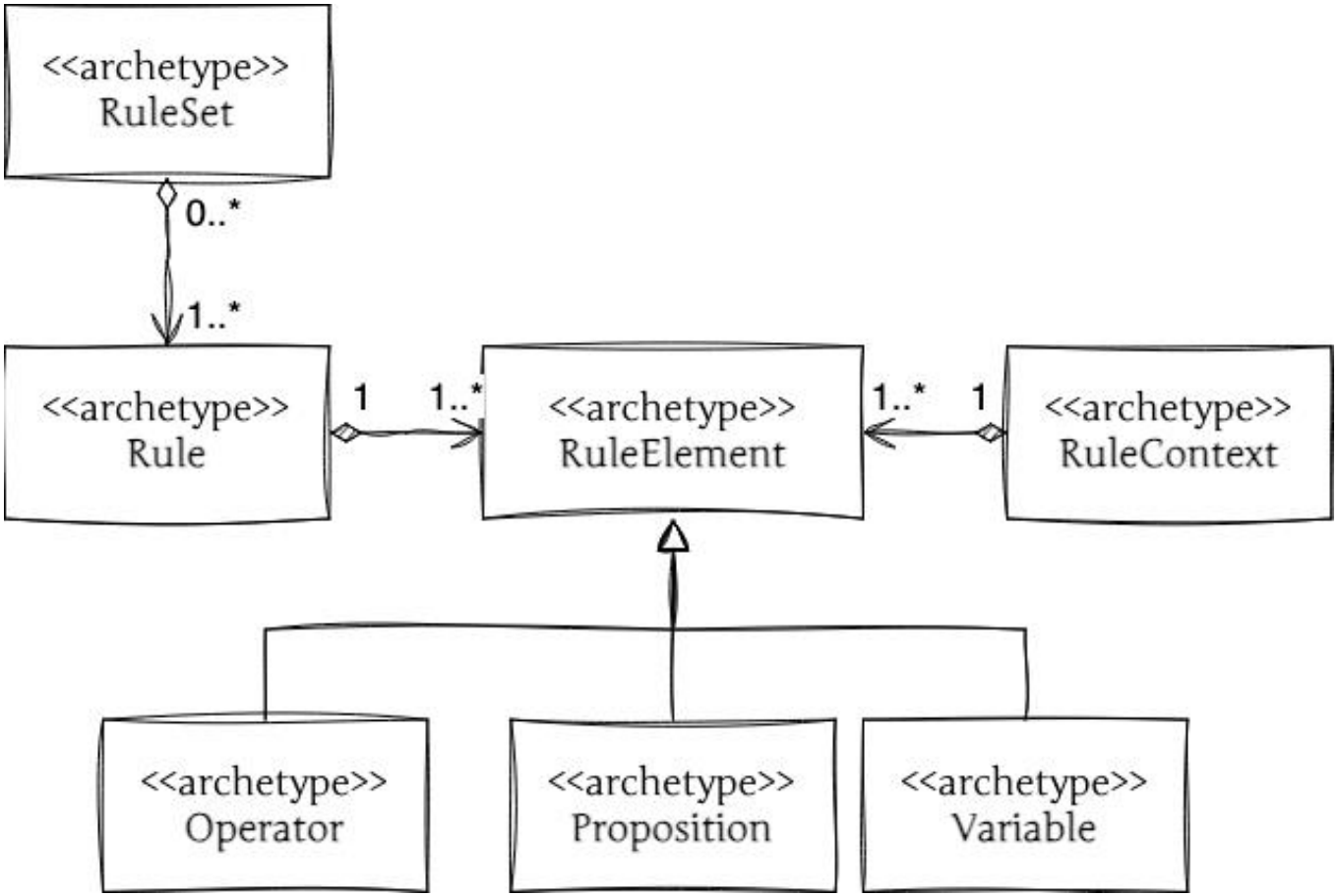
https://softwarearchetypes.com/

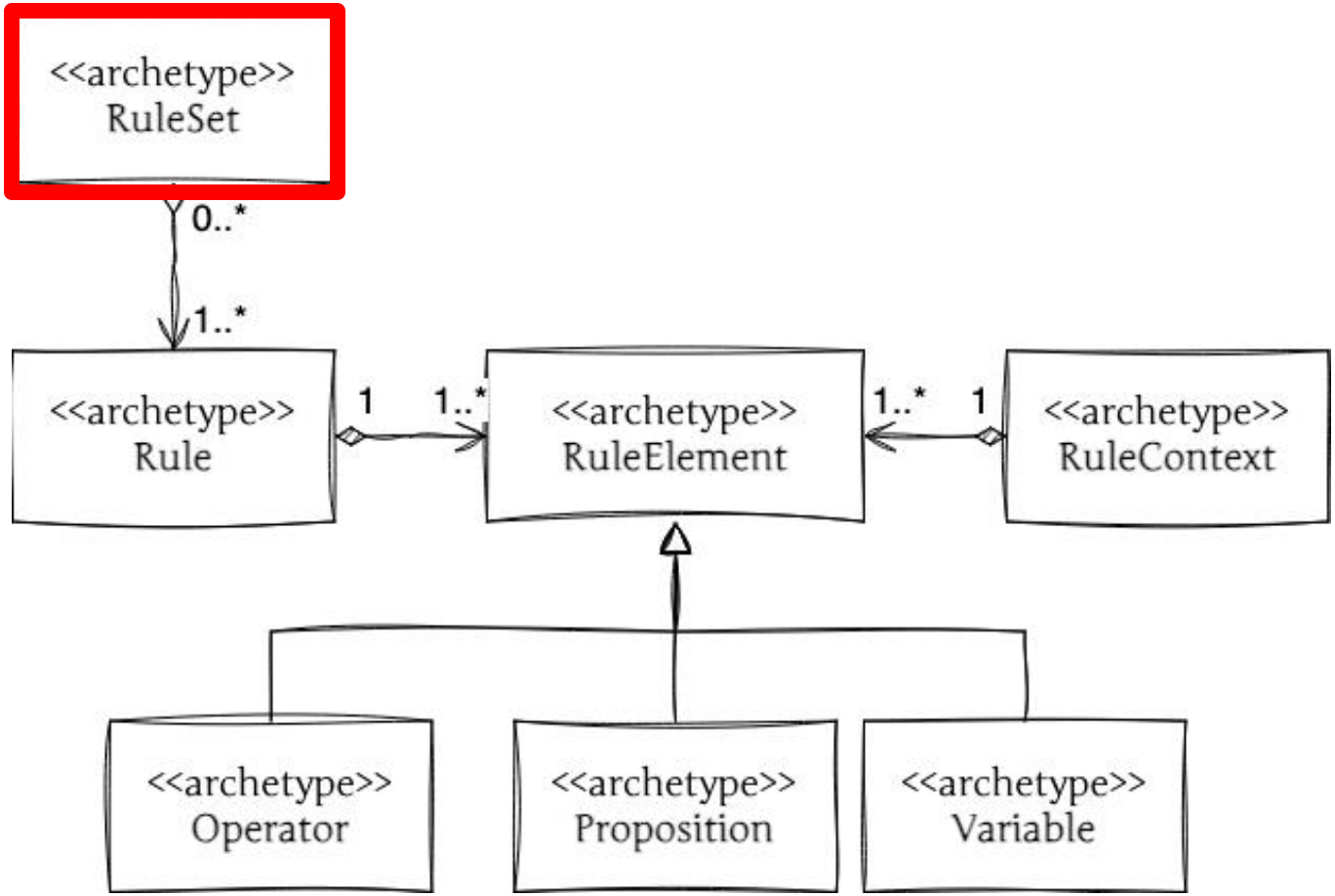
Zarządzanie regułami

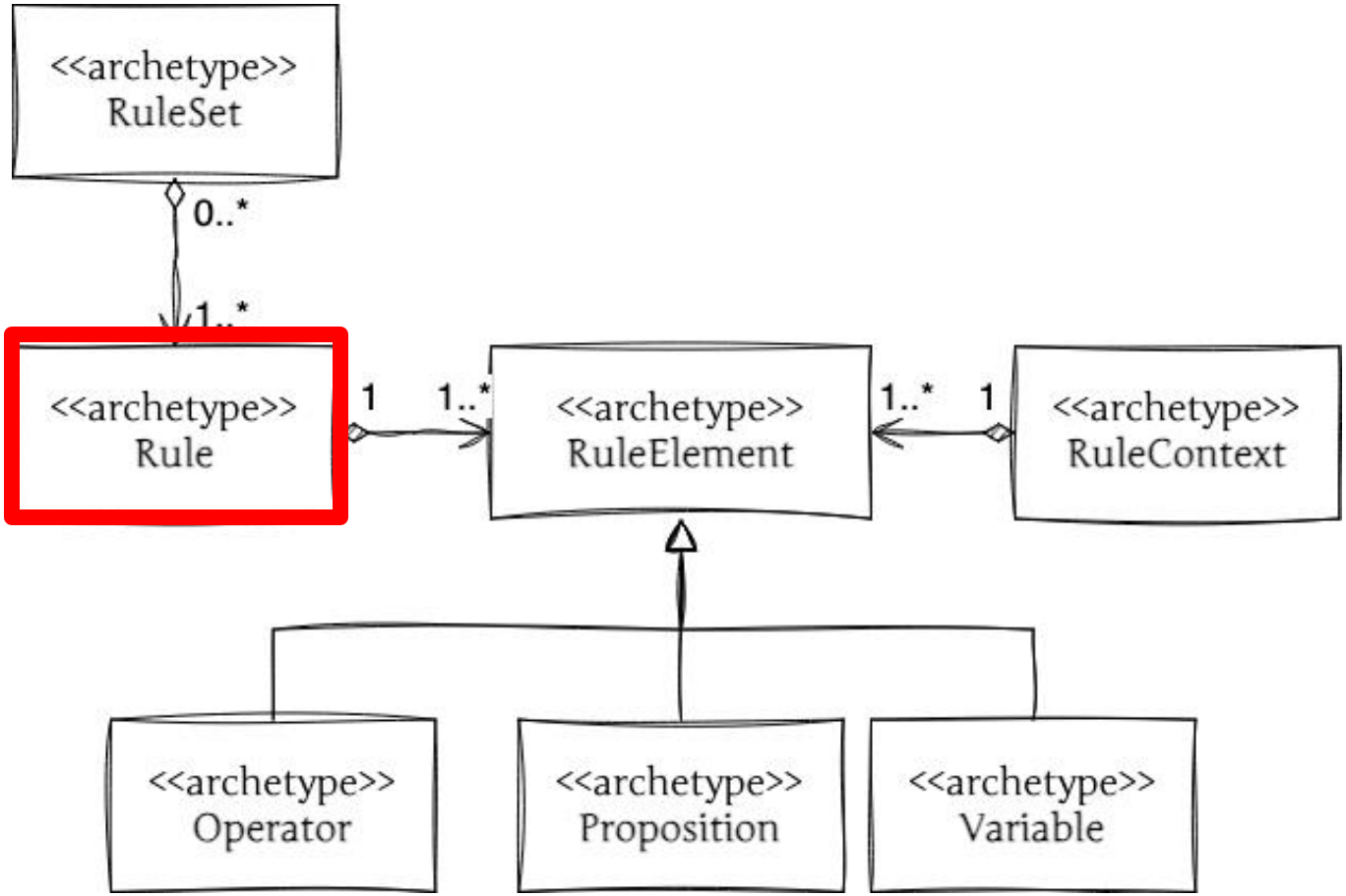
Rule

Zarządzanie regułami

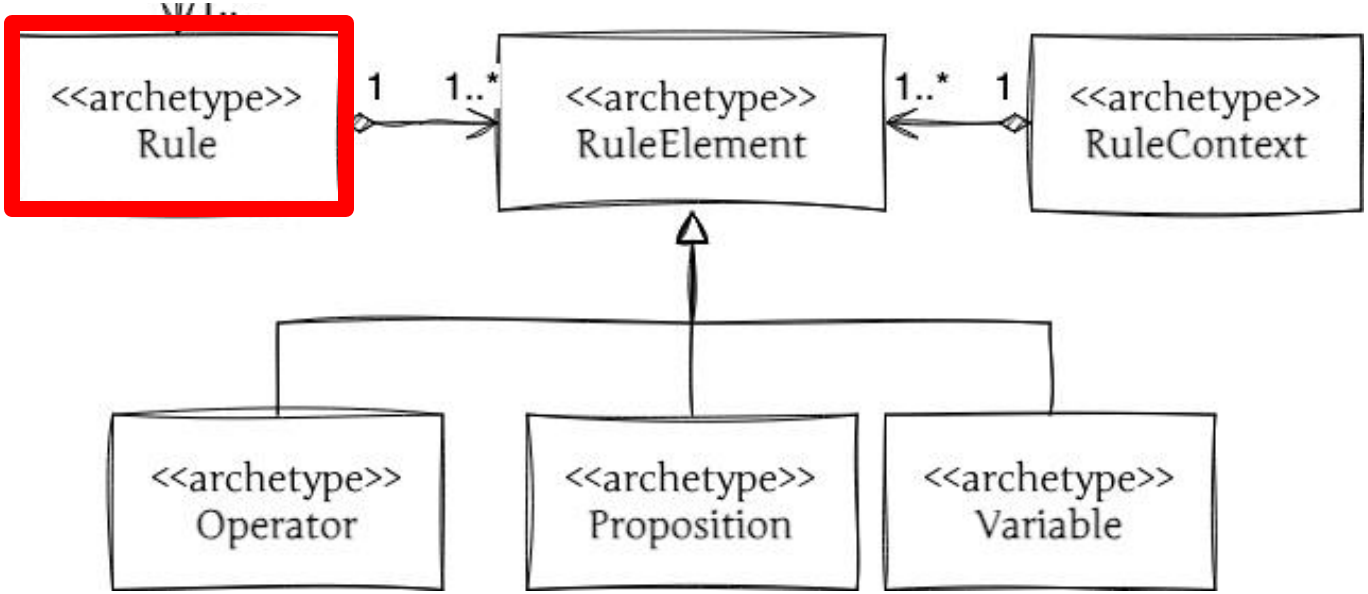


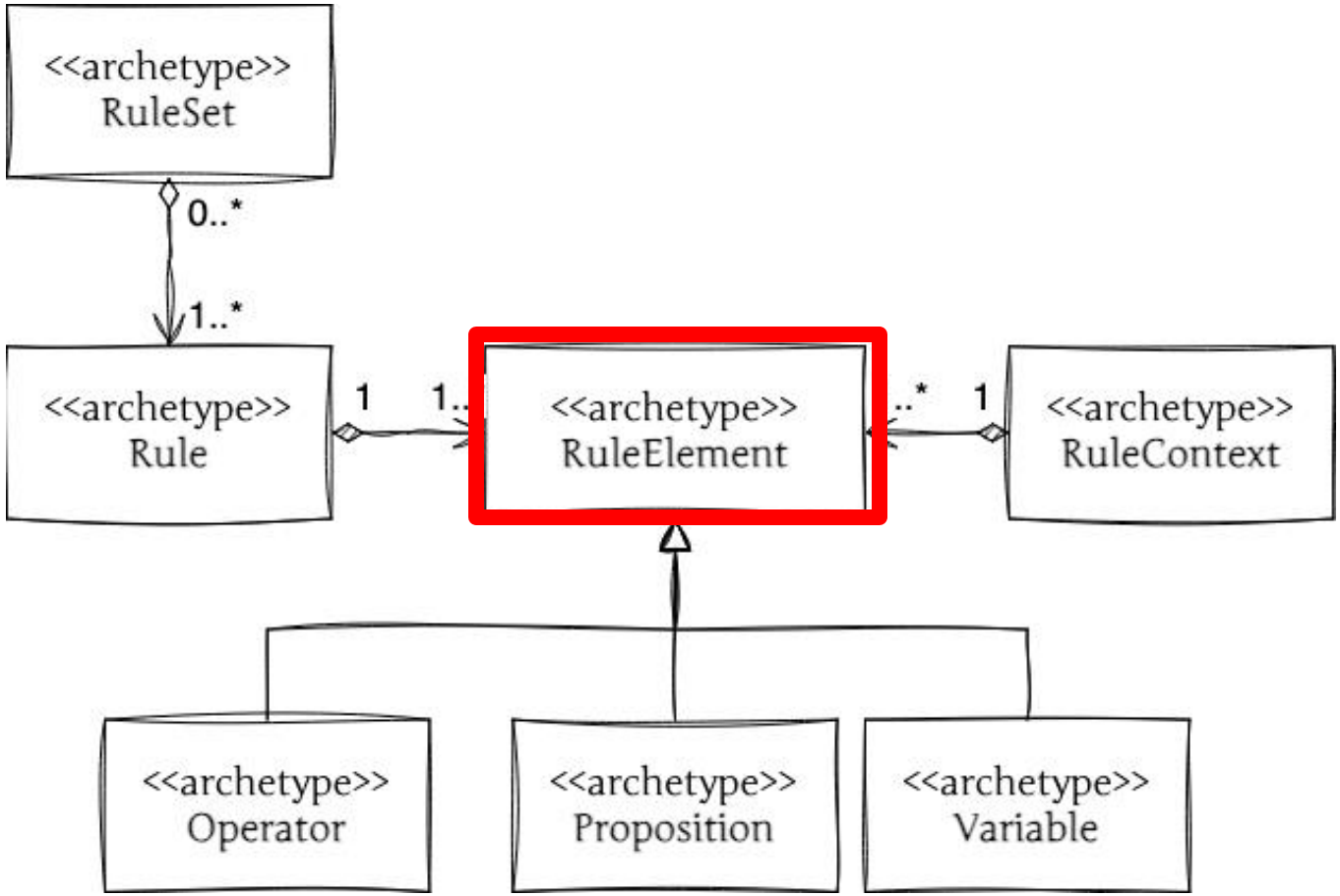

<<archetype>>
RuleSet
0..*
1..*
<<archetype>>
Rule
1
1..*
<<archetype>>
RuleElement
1..*
1
<<archetype>>
RuleContext
<<archetype>>
Operator
<<archetype>>
Proposition
<<archetype>>
Variable

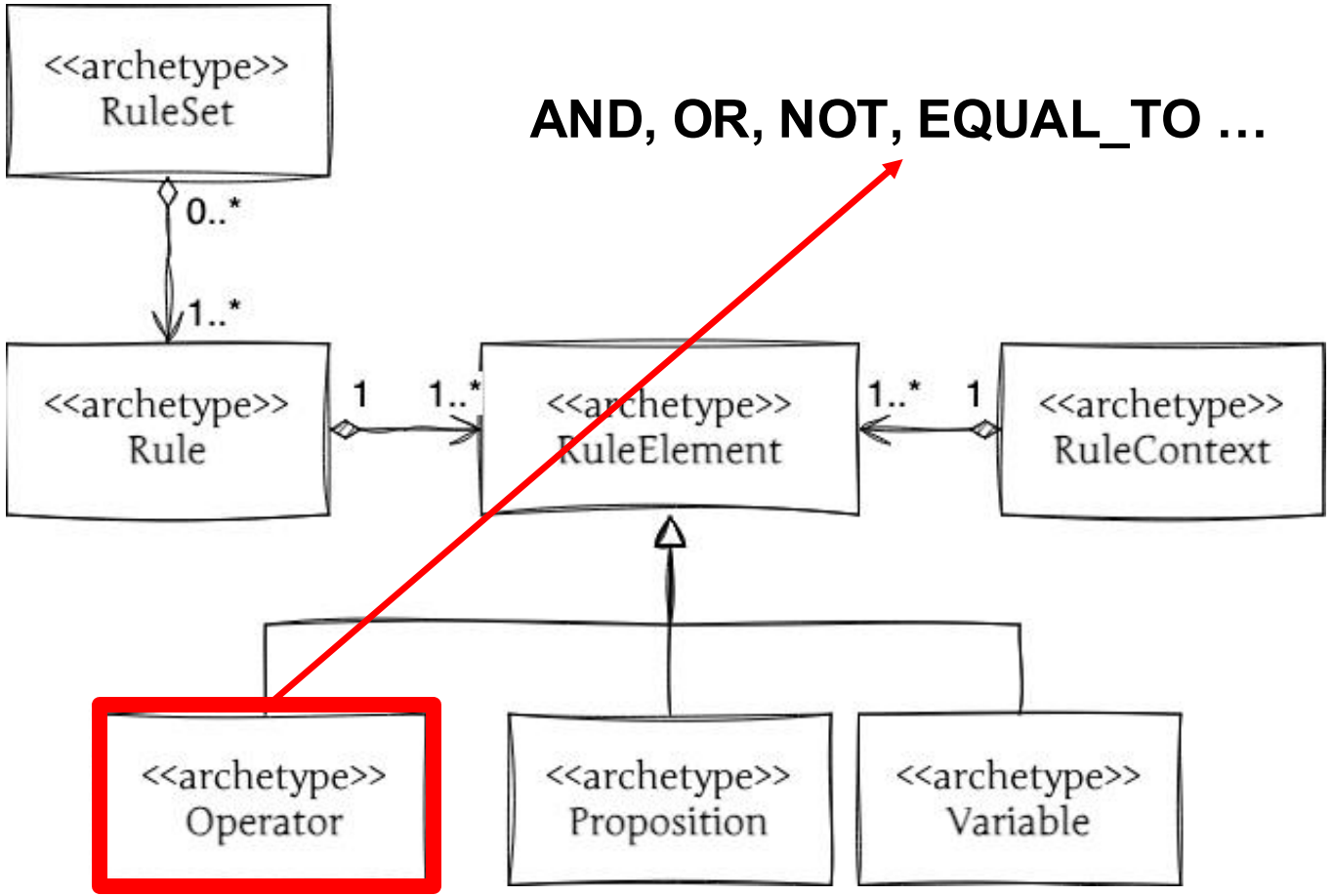

<<archetype>>
RuleSet
0..*
1..*
<<archetype>>
Rule
1
1..*
<<archetype>>
RuleElement
1..*
1
<<archetype>>
RuleContext
<<archetype>>
Operator
<<archetype>>
Proposition
<<archetype>>
Variable

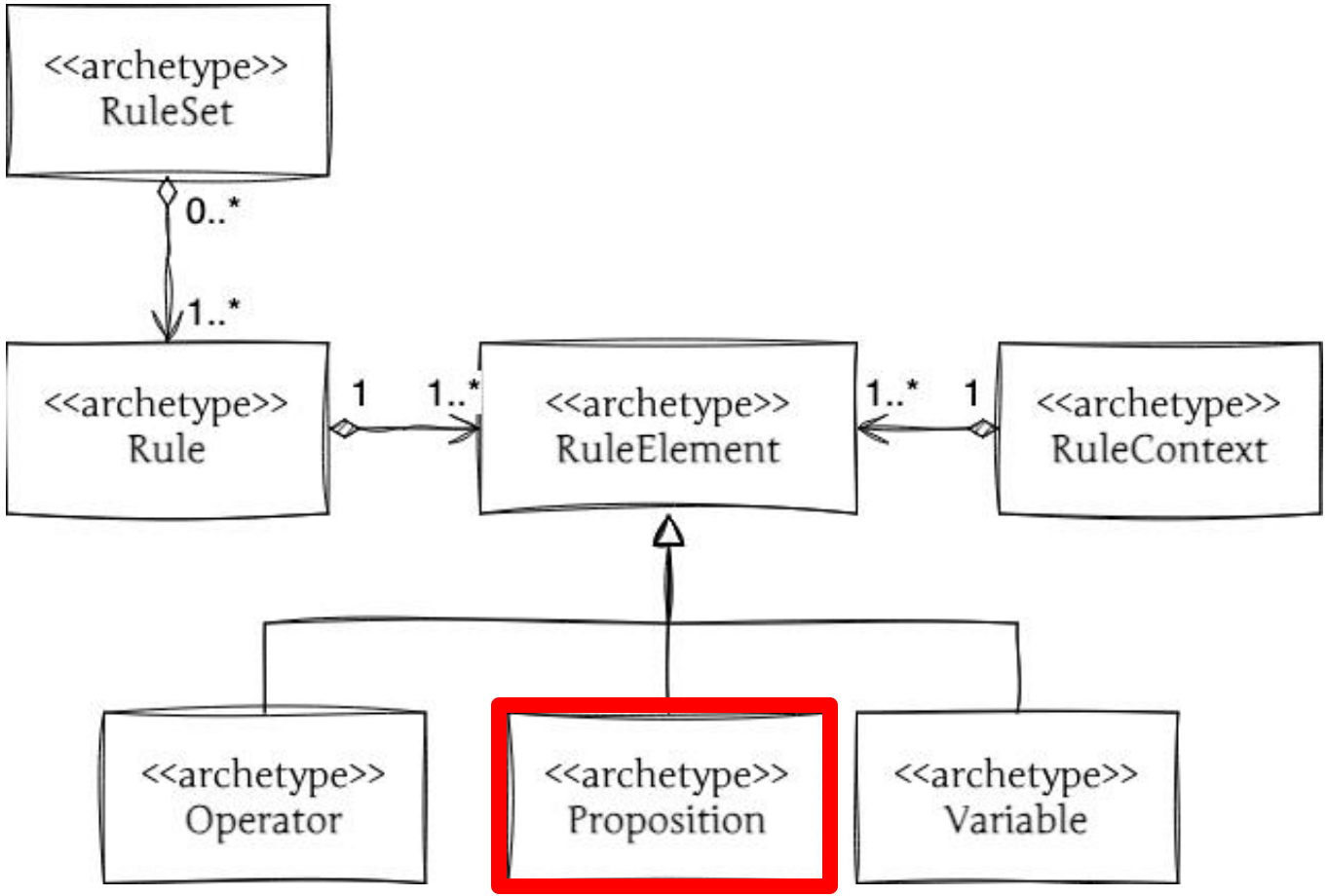

<<archetype>>
RuleSet
0..*
1..*
<<archetype>>
Rule
1
1..*
<<archetype>>
RuleElement
1..*
1
<<archetype>>
RuleContext
<<archetype>>
Operator
<<archetype>>
Proposition
<<archetype>>
Variable

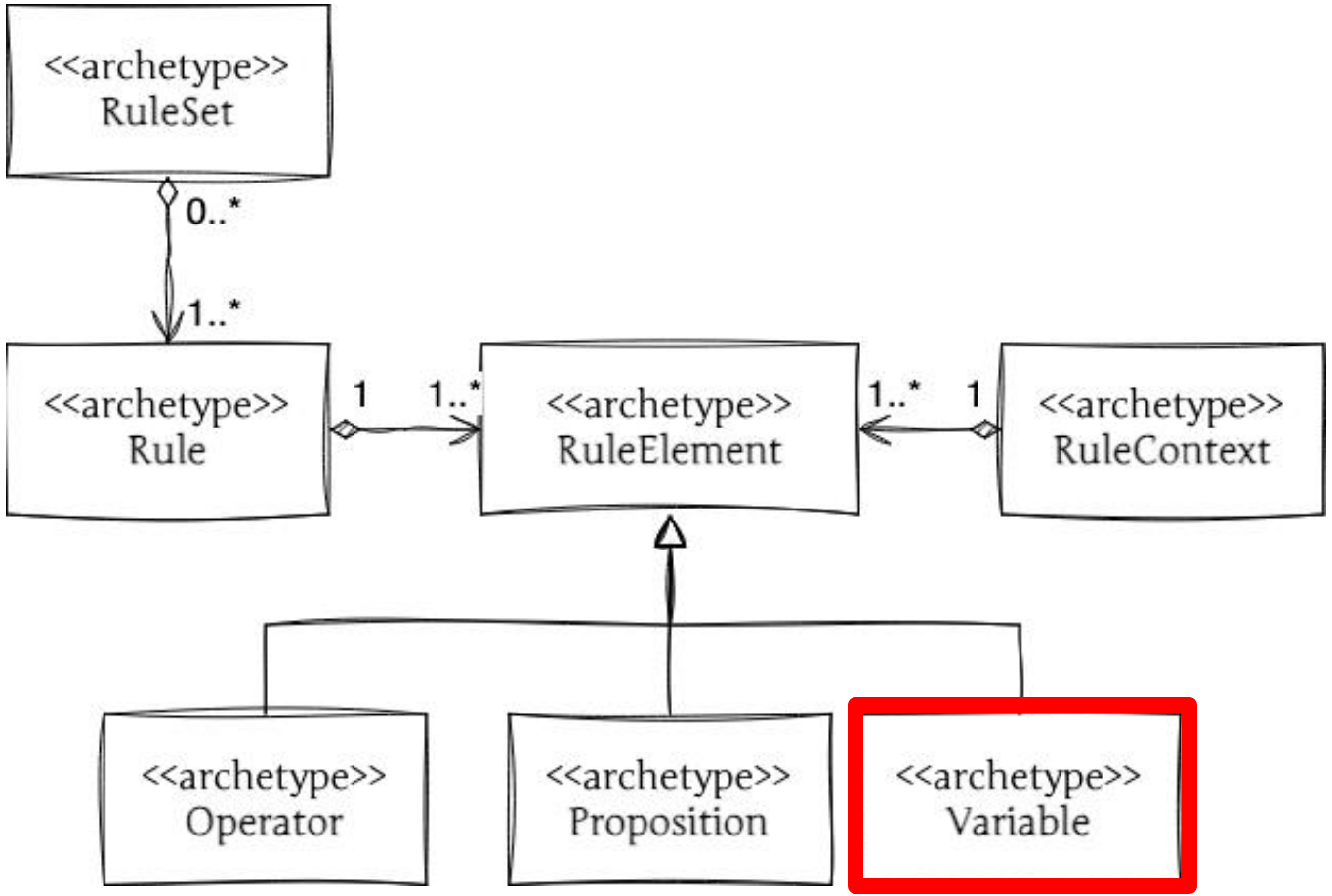
```php
$validRole = new Rule( name: 'validResponsibilityRule');
$validRole
    ->variable( name: 'role')
    ->variable( name: 'expected role',  value: RoleOfScientist::class)
    ->equalTo();
```

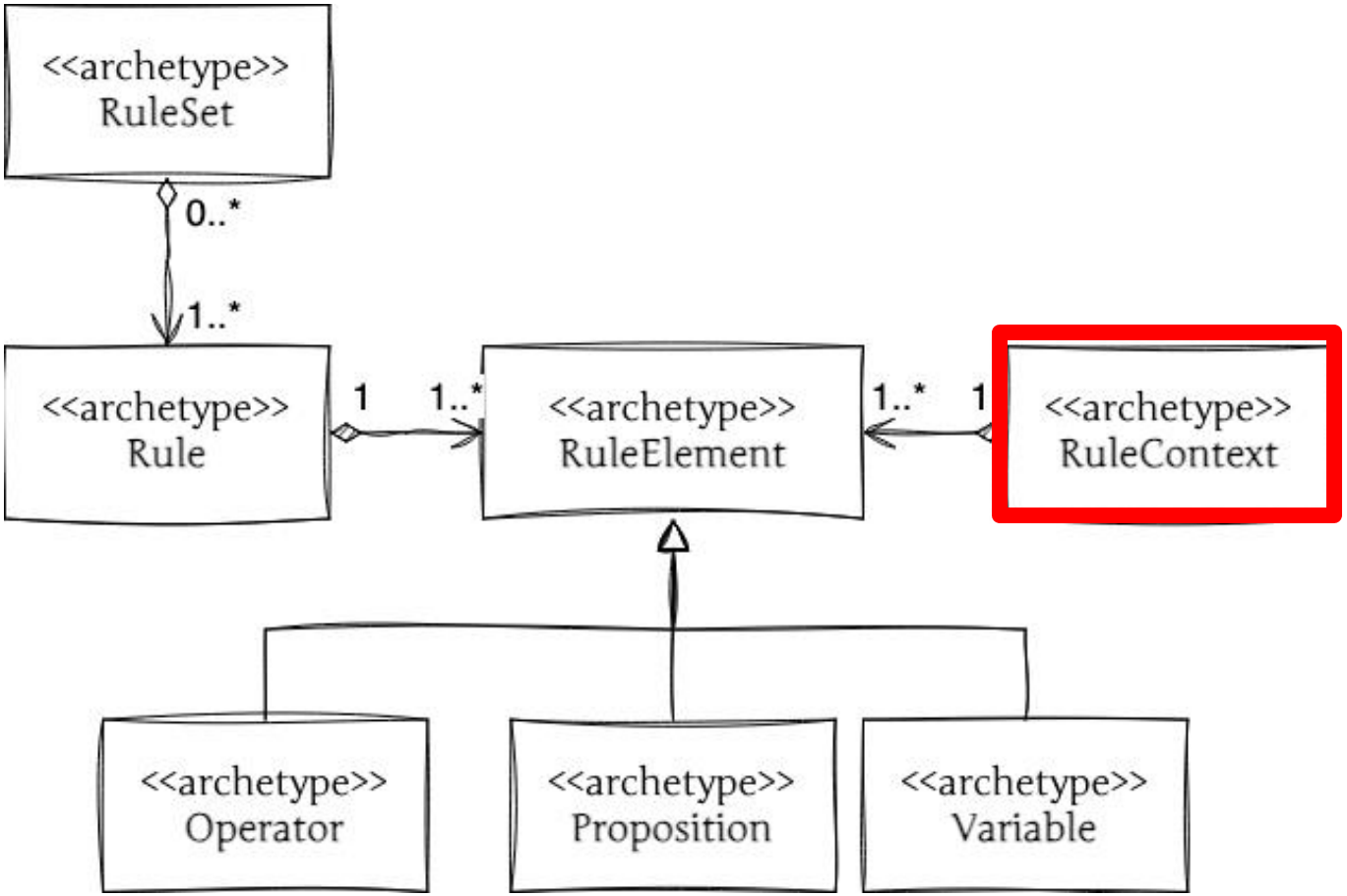



<<archetype>>
RuleSet
0..*
1..*
<<archetype>>
Rule
1
1..
<<archetype>>
RuleElement
..*
1
<<archetype>>
RuleContext
<<archetype>>
Operator
<<archetype>>
Proposition
<<archetype>>
Variable


AND, OR, NOT, EQUAL_TO …
<<archetype>>
RuleSet
0..*
1..*
<<archetype>>
Rule
1
1..*
<<archetype>>
RuleElement
1..*
1
<<archetype>>
RuleContext
<<archetype>>
Operator
<<archetype>>
Proposition
<<archetype>>
Variable


<<archetype>>
RuleSet
0..*
1..*
<<archetype>>
Rule
1
1..*
<<archetype>>
RuleElement
1..*
1
<<archetype>>
RuleContext
<<archetype>>
Operator
<<archetype>>
Proposition
<<archetype>>
Variable


<<archetype>>
RuleSet
0..*
1..*
<<archetype>>
Rule
1
1..*
<<archetype>>
RuleElement
1..*
1
<<archetype>>
RuleContext
<<archetype>>
Operator
<<archetype>>
Proposition
<<archetype>>
Variable


<<archetype>>
RuleSet
0..*
1..*
<<archetype>>
Rule
1
1..*
<<archetype>>
RuleElement
1..*
1
<<archetype>>
RuleContext
<<archetype>>
Operator
<<archetype>>
Proposition
<<archetype>>
Variable

```php
class Pricing
{
    2 usages  new *
    public static function createRuleContext(
        Product $product,
        UnsignedQuantity $quantity,
        CountryCode $location
    ): RuleContext {
        $elements = [
            new Variable('productCategory', $product->getCategory()),
            new Variable('quantity', $quantity->getValue()),
            new Variable('location', $location),
            new Proposition('seasonalPromotion', fn() => self::isSeasonalPromotionActive())
        ];

        return new RuleContext($elements);
    }
}
```

Operator | Proposition | Variable

Archetypes Rules JS

https://github.com/commonality/archetypes-rules

Archetypes Rules PHP

https://github.com/jakubciszak/rule-engine

Ktoś już to wymyślił

Modelowanie z użyciem archetypów biznesowych.

Jakub Ciszak