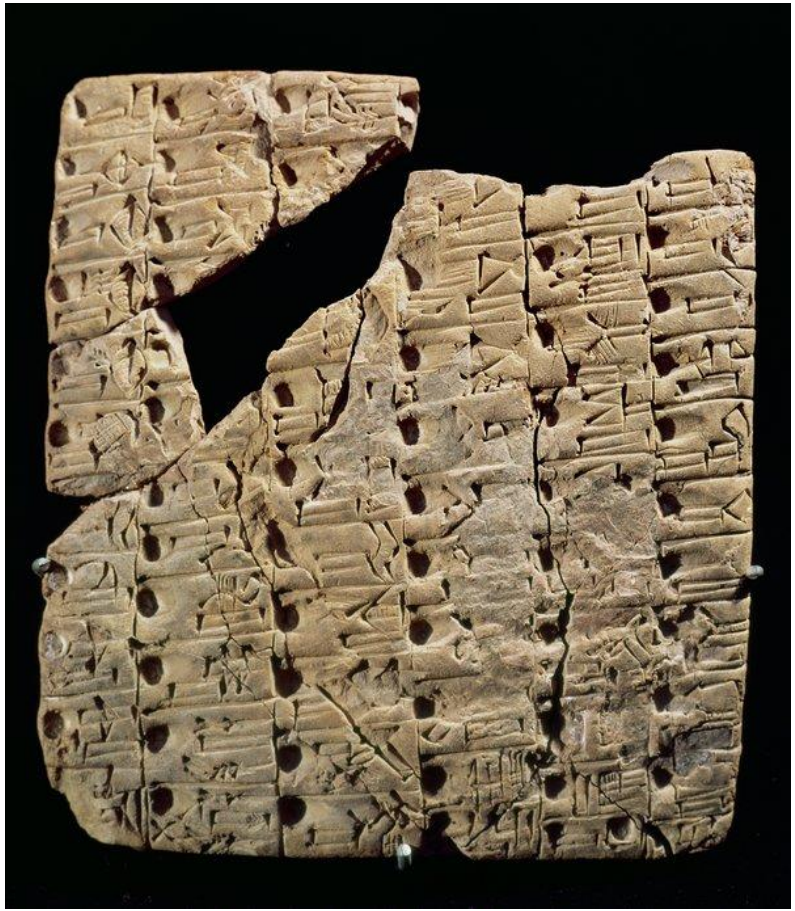
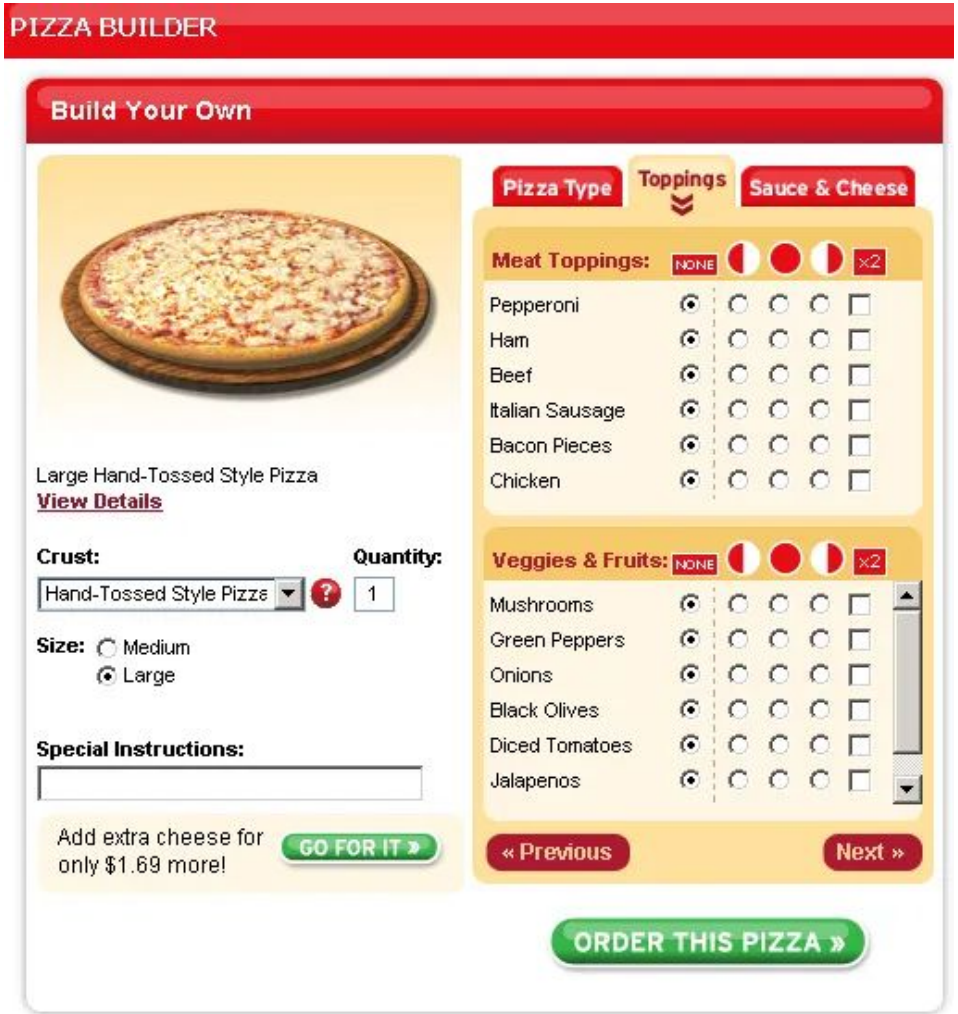


źródło: https://alternatywne.pl/top-10-najdrozsze-monety-swiata-w-2019-r/

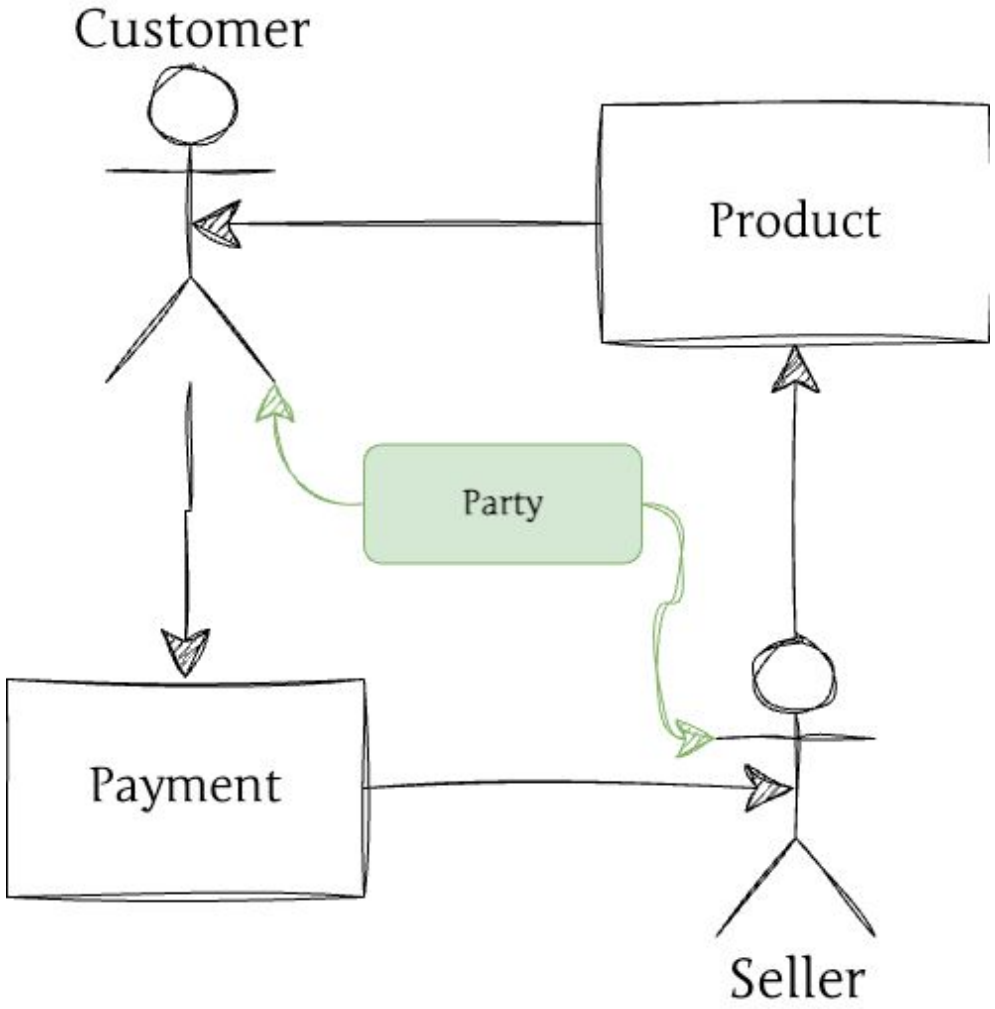
źródło https://kingkullen.com/about-us/

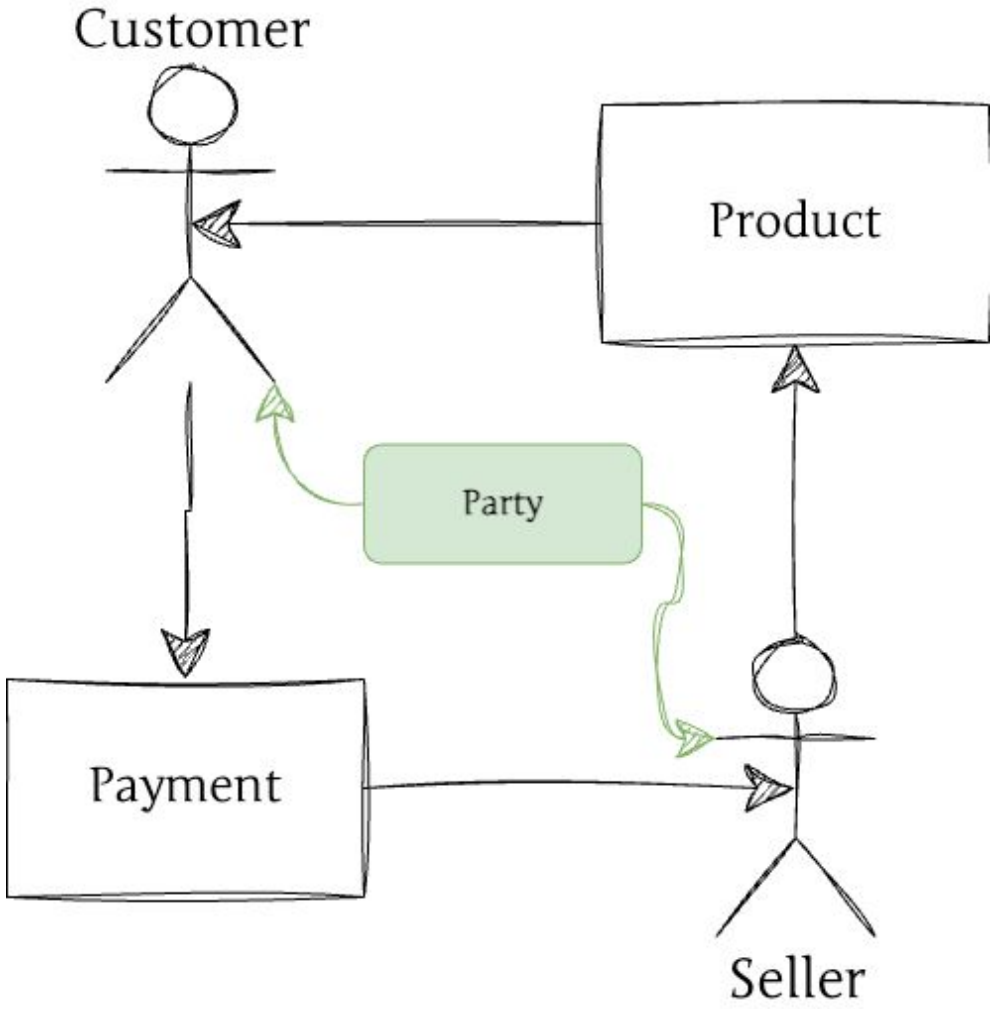


źródło https://thehistoryoftheweb.com/postscript/pizzanet/

źródło https://web.archive.org/web/19990421070114/http://www.netmarket.com/

źródło https://businessinsider.com.pl/firmy/historia-allegro-firmy-ktora-wygnala-shopee-z-polski/zjg3lcw


Customer
Product
Party
Payment
Seller


Customer
Product
Party
Payment
Seller

Jakub Ciszak

Senior Software Developer

Burda Media Polska

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

Ktoś już to wymyślił

Modelowanie z użyciem archetypów biznesowych.

Plan

1. Co to jest archetyp?
2. Czym są archetypy biznesowe?
3. Historia pewnej firmy
    a. ceny uzależnione od różnych czynników
    b. rozbudowa systemu magazynowego
    c. rozszerzenie modelu sprzedażowego
4. Wzorce
    a. **Quantity**, **Money** oraz **Price**
    b. **Inventory**
    c. **Party i PartyRelationship**
5. Implementacja
6. Podsumowanie
7. QA

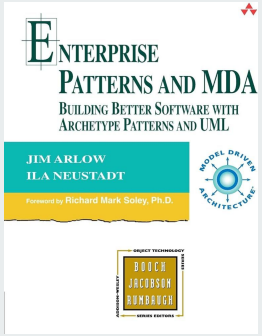
Co to jest archetyp?

Słownik Języka Polskiego

- pierwotny wzór postaci, obiektu lub zjawiska;
- postać, obiekt lub zjawisko odzwierciedlające taki wzór;
- 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.”

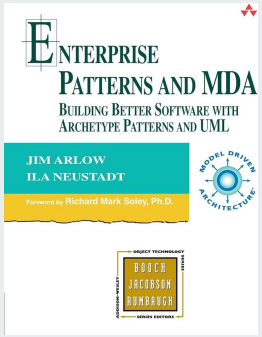
…

„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.”

Charakterystyka archetypów

- **Uniwersalne**: Muszą pojawiać się konsekwentnie w domenach biznesowych i systemach.
- **Wszechobecne**: Występują zarówno w domenie biznesowej, jak i w domenie oprogramowania.
- **Głęboka historia**: Na przykład archetyp Produktu istnieje od momentu, gdy ludzie zaczęli handlować.
- **Oczywiste dla ekspertów domenowych**: archetyp powinien być oczywisty dla eksperta domenowego.

TechShop

Planowana sprzedaż międzynarodowa

Ceny uzależnione od różnych czynników


class Product
{
/**
* @var array<string, int>
*/
2 usages
private array $prices;
1 usage
private readonly CurrencyCode $masterCurrency;
no usages new *
public function getPriceByCurrency(CountryCode $countryCode): int
{
return $this->prices[$countryCode->value] ?? $this->prices[$this->masterCurrency->value];
}
}

```php
class Order
{
    /**
     * @var ArrayCollection<OrderLineItem>
     */
    2 usages
    private ArrayCollection $items;

    no usages   new *
    public function __construct(private readonly CountryCode $currency) {
        $this->items = new ArrayCollection();
    }

    no usages   new *
    public function addItem(Product $product, int $quantity): void
    {
        $this->items->add(
            new OrderLineItem($product, $product->getPriceByCurrency($this->currency), $quantity)
        );
    }
//...
}
```

Rozwój systemu magazynowego

Brak wsparcia dla wielu magazynów

```php
class Product
{
    //…
    2 usages
    private Stock $stock;

    no usages    new *
    public function isAvailable(): bool
    {
        return $this->stock->isAvailable();
    }

    no usages    new *
    public function getStock(): Stock
    {
        return $this->stock;
    }
}
```

```php
readonly class StockEntry
{
    no usages    new *
    public function __construct(
        public Quantity $quantity,
        public \DateTimeImmutable $dateTime
    ) {
    }
}
```

```php
class Stock
{
    /**
     * @var ArrayCollection<StockEntry>
     */
    1 usage
    private ArrayCollection $entries;

    no usages    new *
    public function __construct()
    {
        $this->entries = new ArrayCollection();
    }

    no usages    new *
    public function addEntry(StockEntry $entry): void
    {
    }

    no usages    new *
    public function availableQuantity(): Quantity
    {
    }

    no usages    new *
    public function isAvailable(): bool
    {
    }
}
```

Wdrożenie modelu

B2B

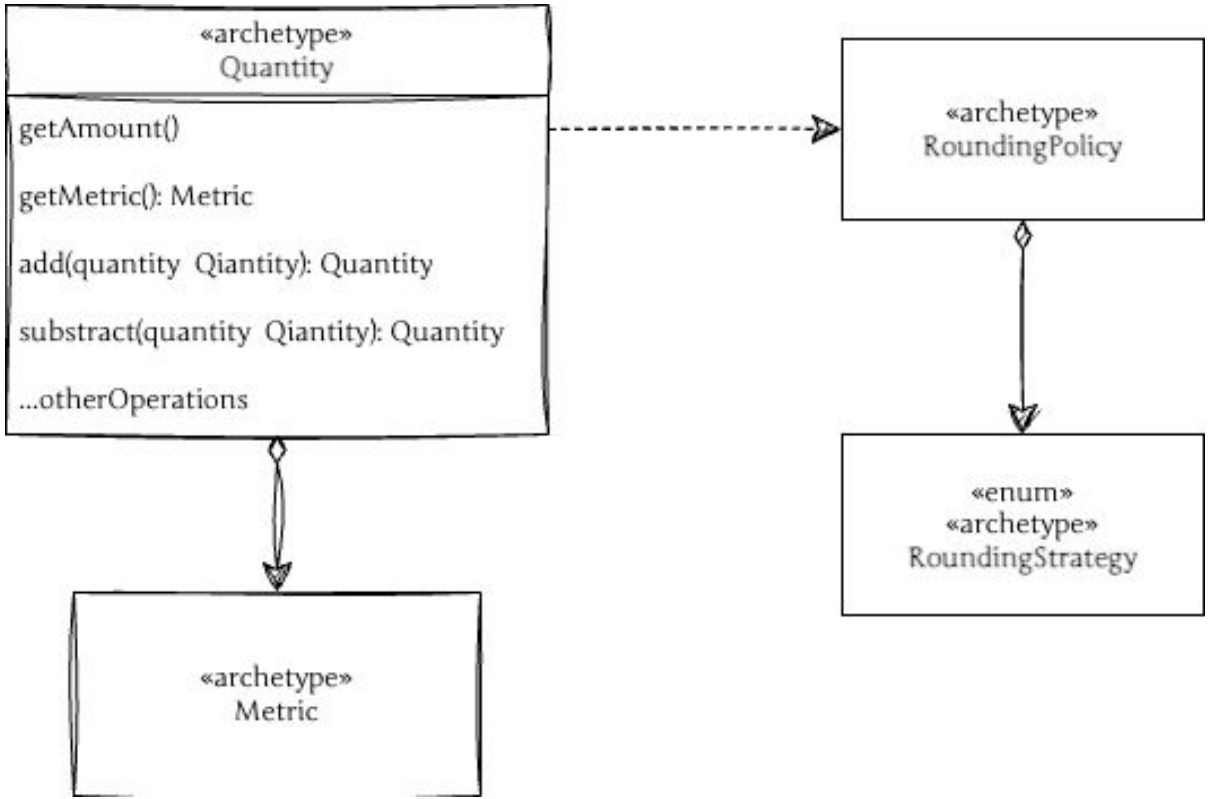
Różne struktury danych klientów.

Złożone procesy zakupowe.

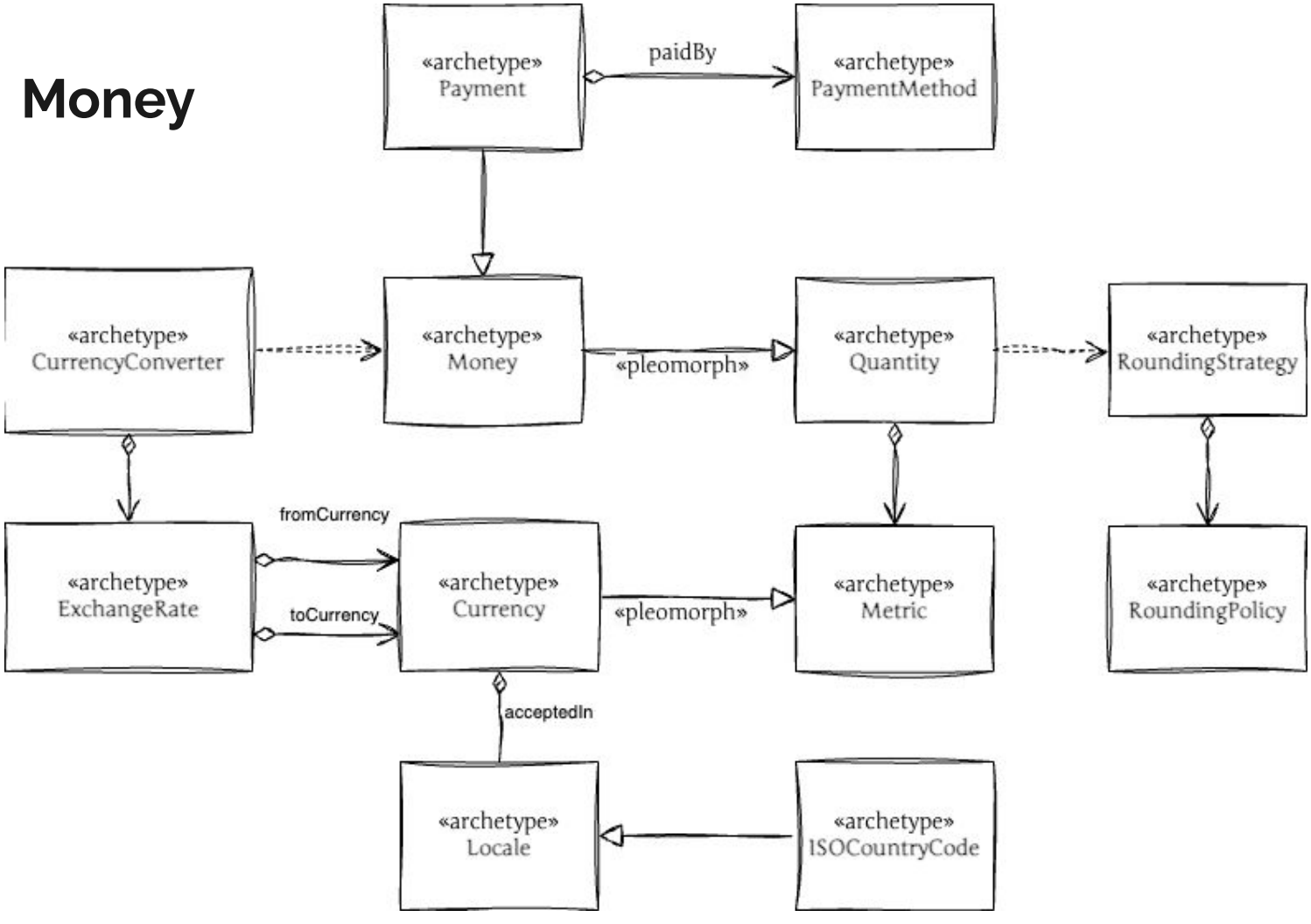
Wzorce

Quantity & Money & Price

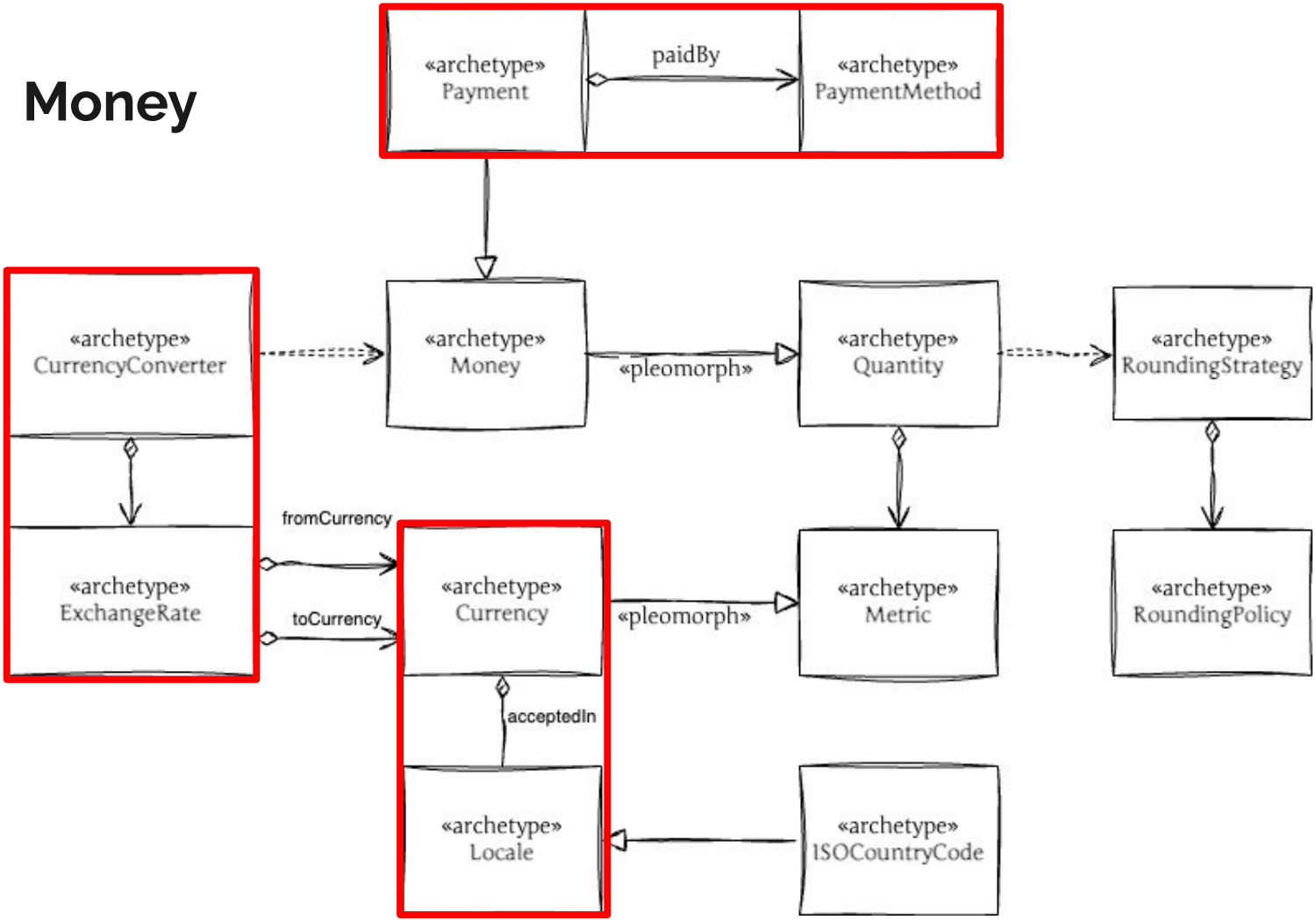
Quantity


«archetype»
Quantity
getAmount()
getMetric(): Metric
add(quantity Qiantity): Quantity
substract(quantity Qiantity): Quantity
...otherOperations
«archetype»
RoundingPolicy
«enum»
«archetype»
RoundingStrategy
«archetype»
Metric

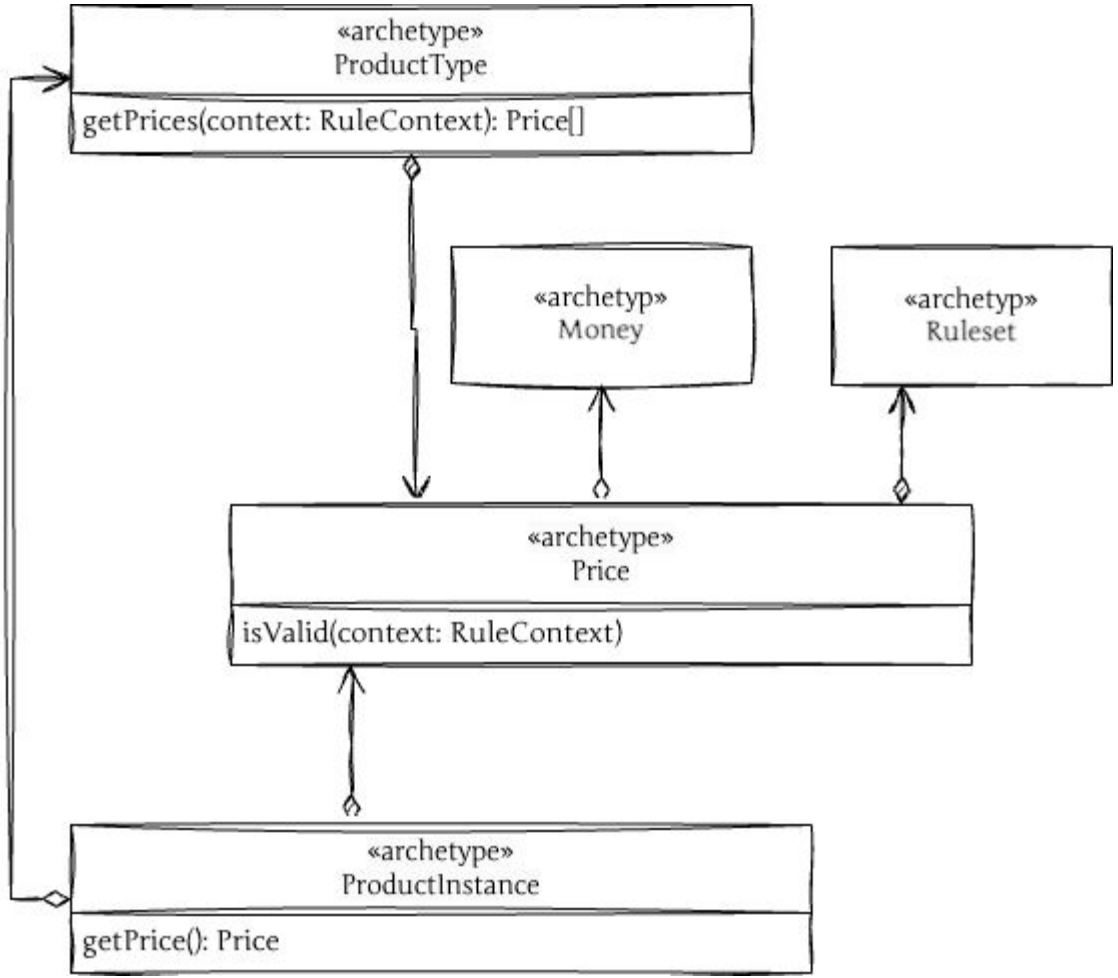
Money


«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

Money


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

Price

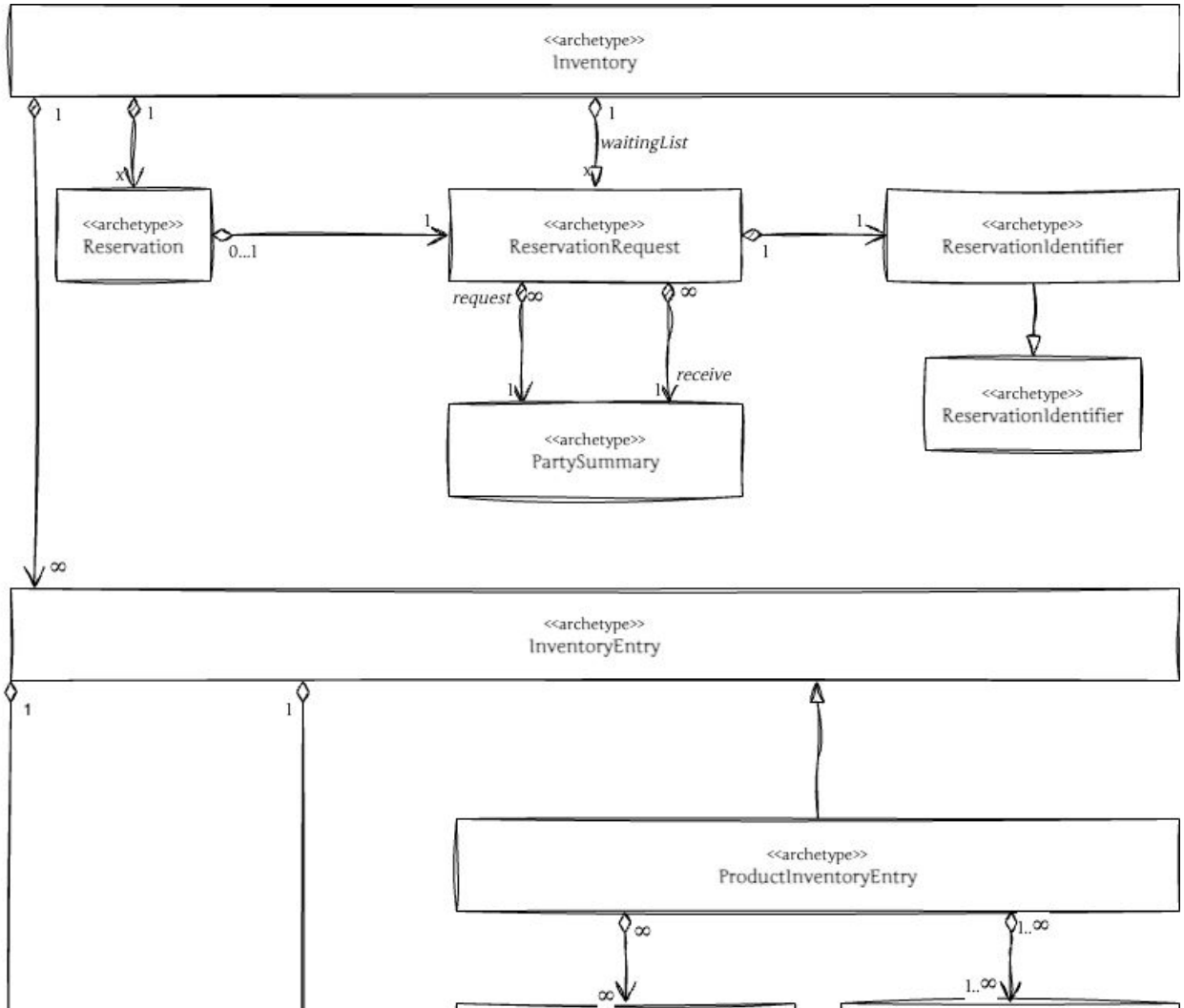

«archetype»
ProductType
getPrices(context: RuleContext): Price[]
«archetyp»
Money
«archetyp»
Ruleset
«archetype»
Price
isValid(context: RuleContext)
«archetype»
ProductInstance
getPrice(): Price

Rule Engine

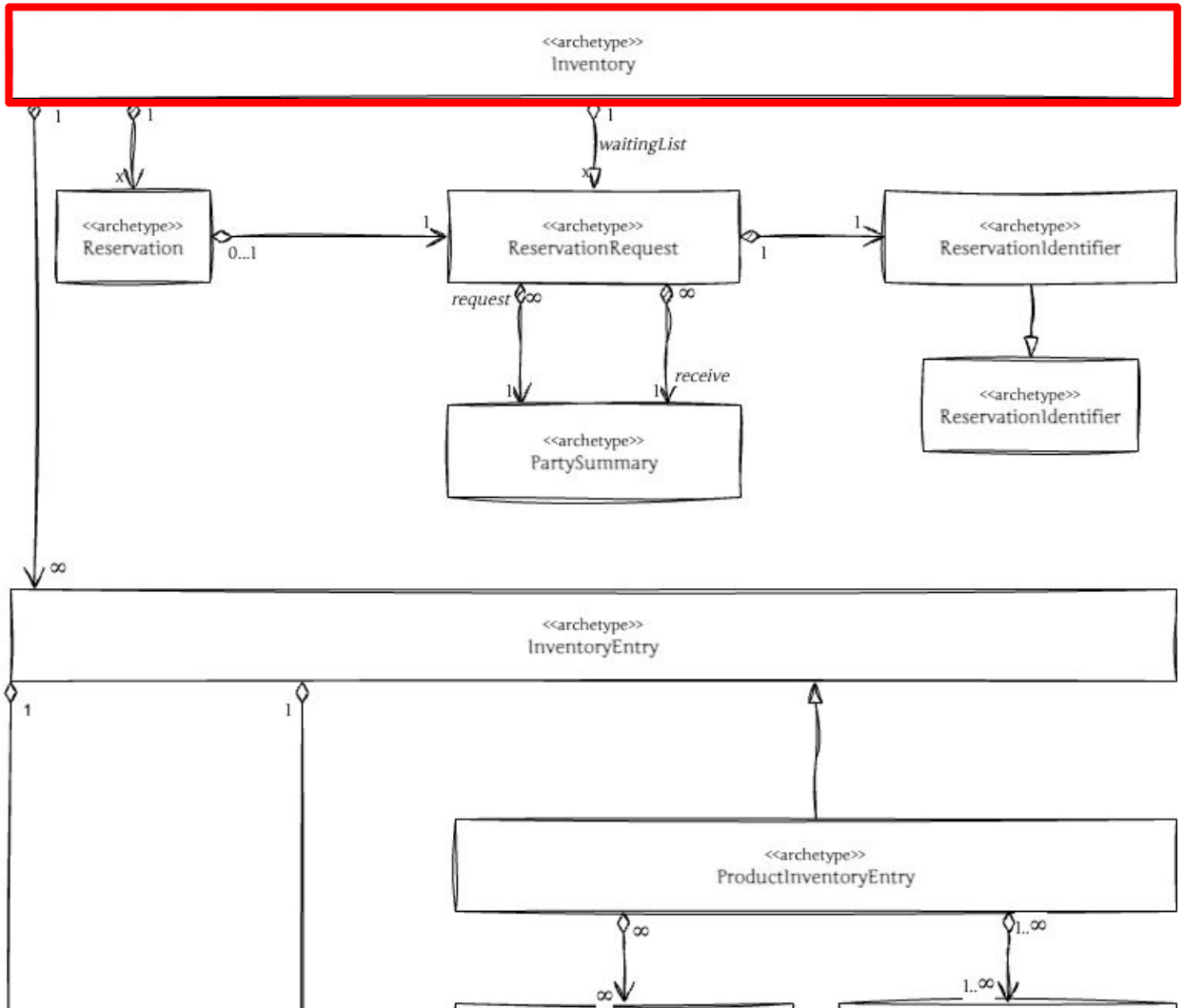
https://github.com/jakubciszak/rule-engine

Inventory

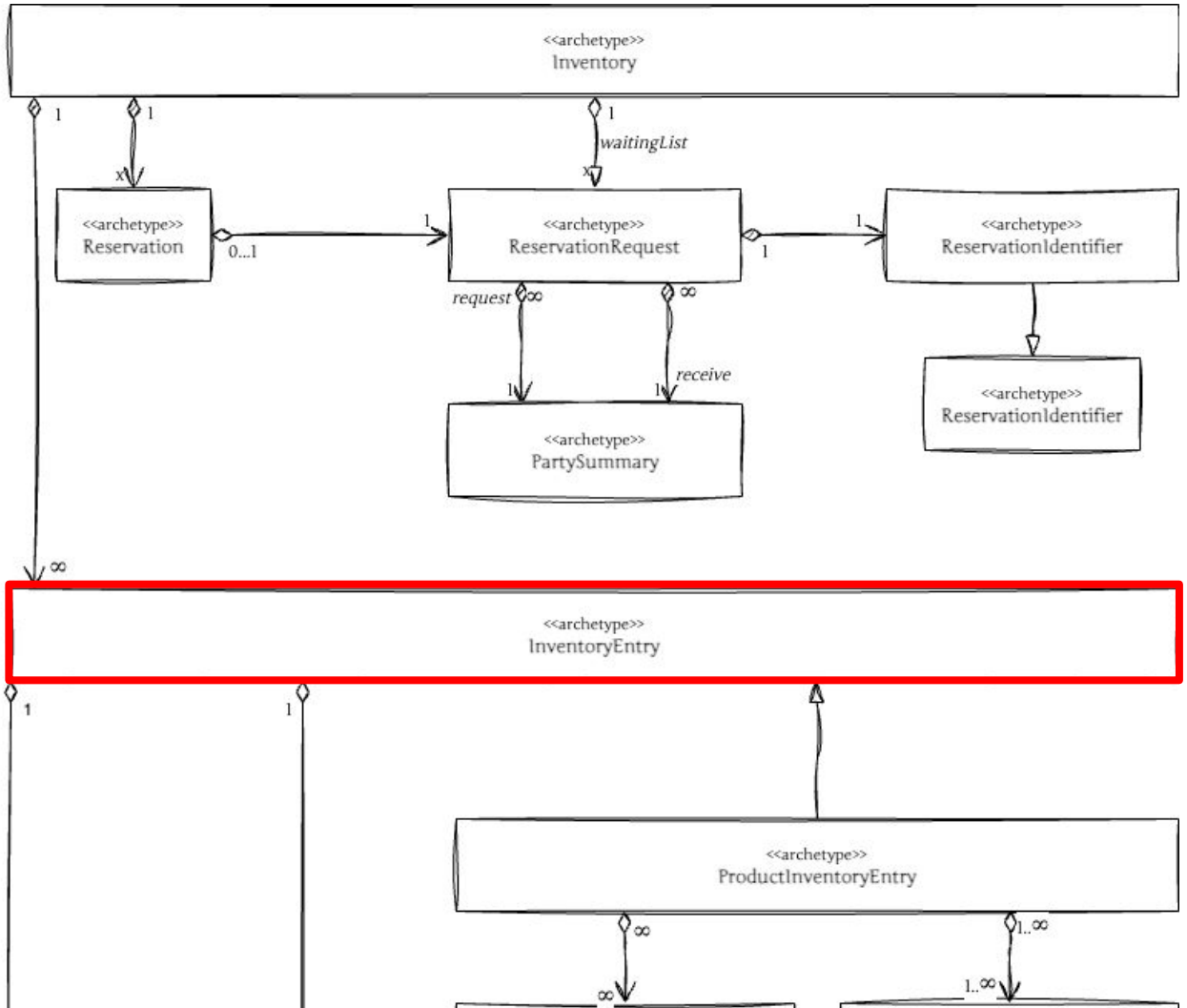
Inventory


<<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..∞

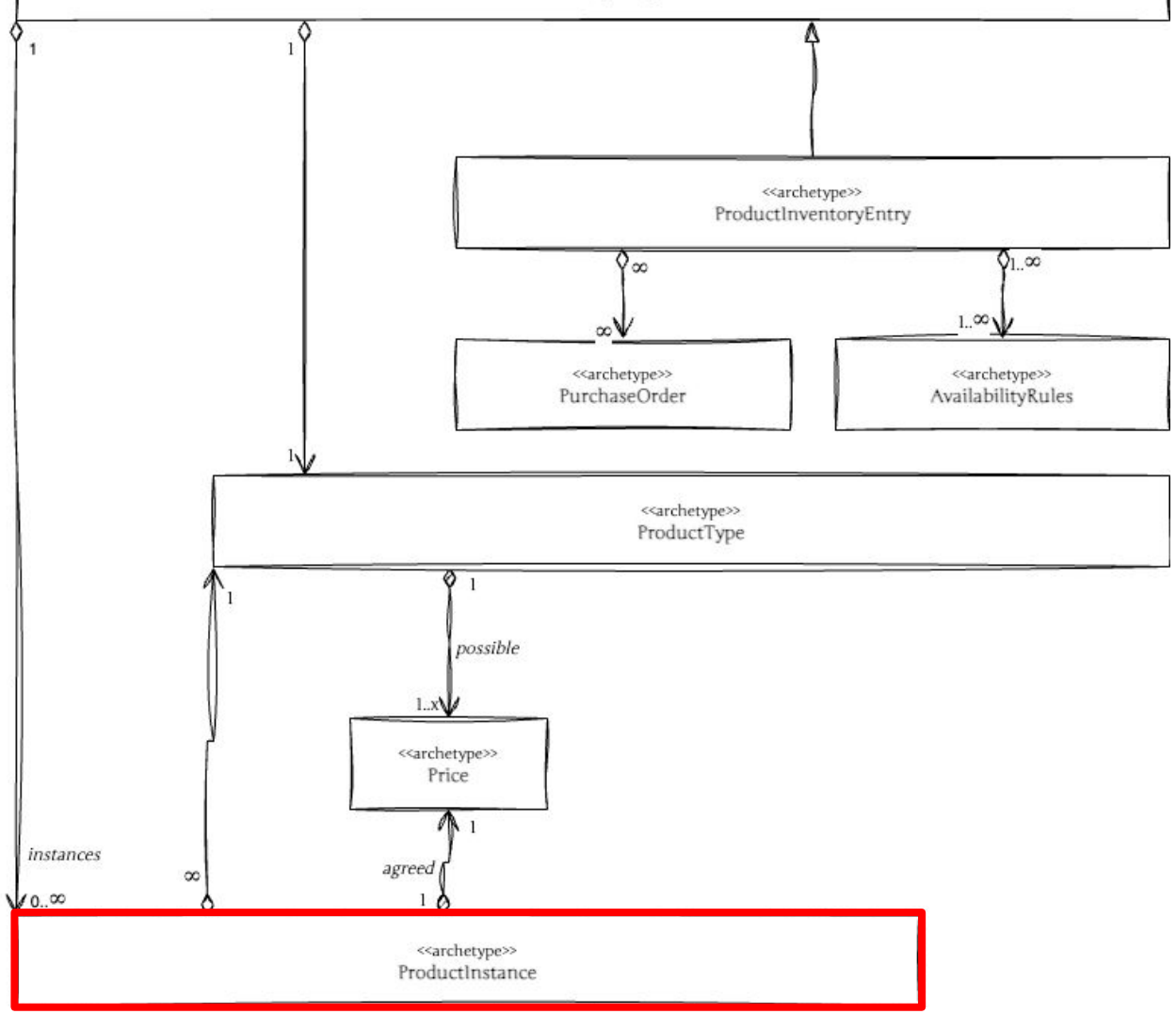
Inventory


<<archetype>>
Inventory
1
1
x
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..∞

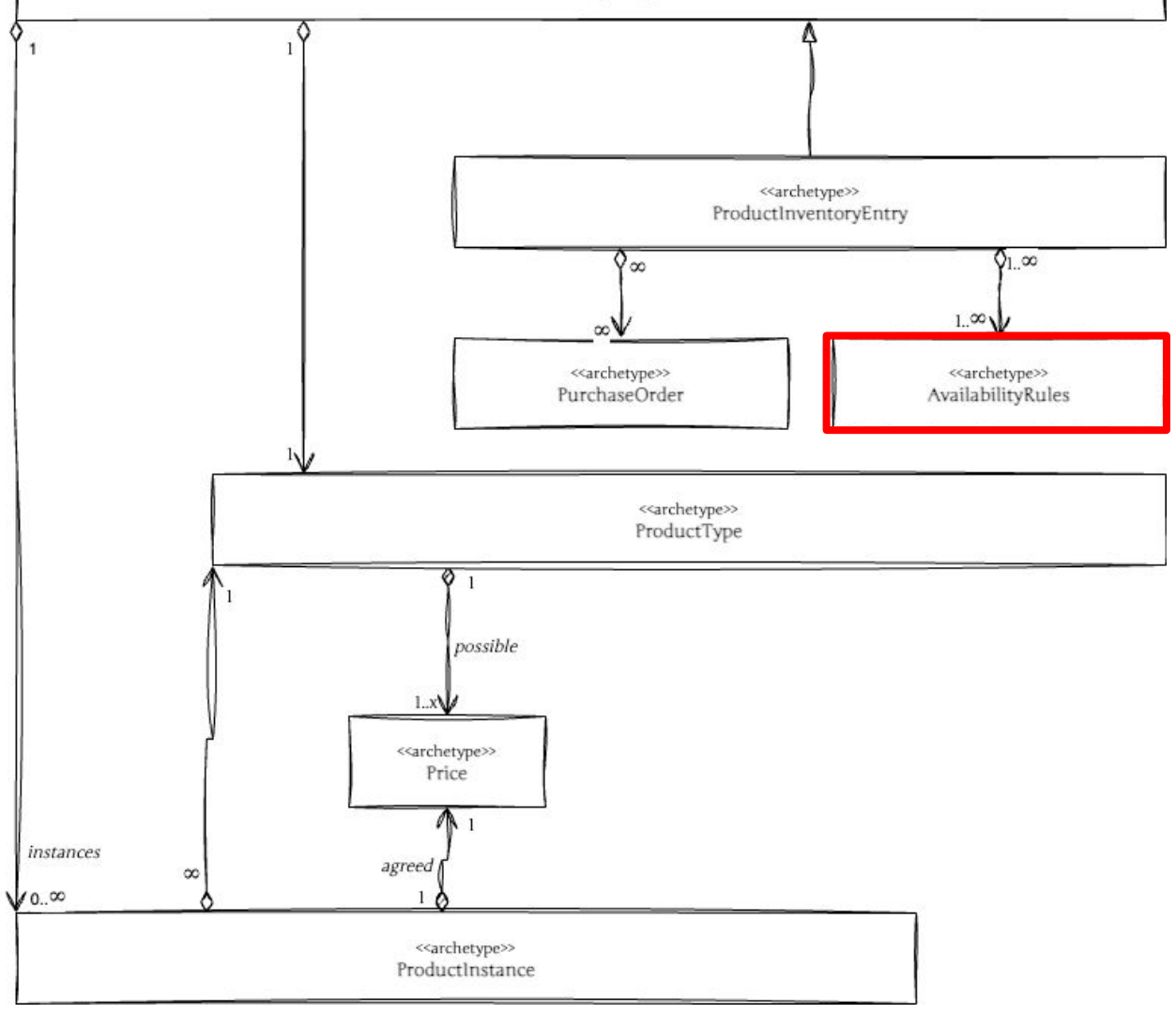
Inventory


<<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..∞

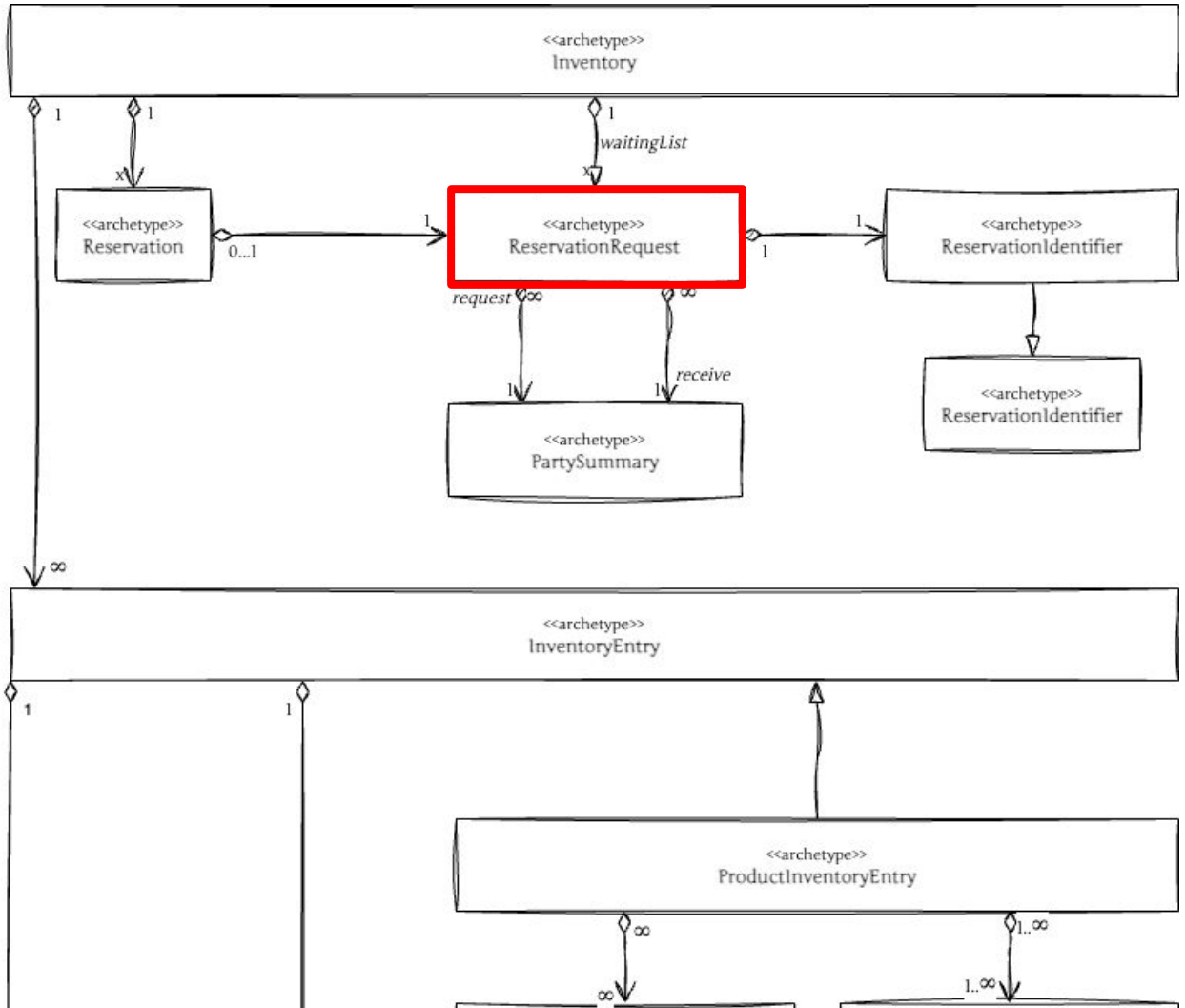
Inventory


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

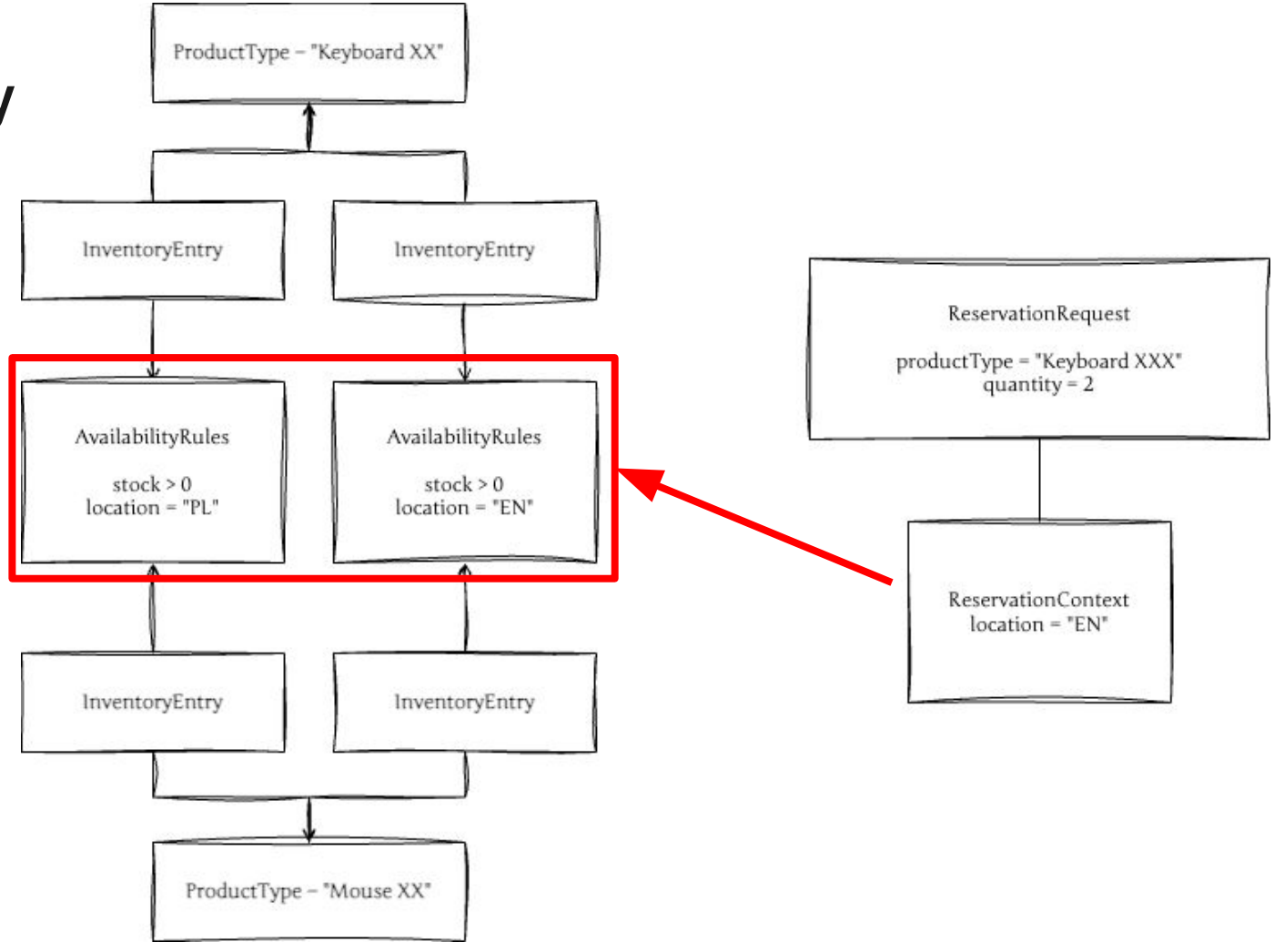
Inventory


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

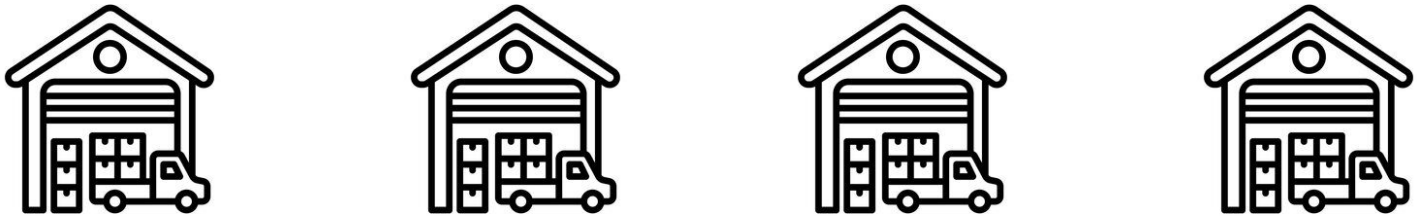
Inventory


<<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..∞

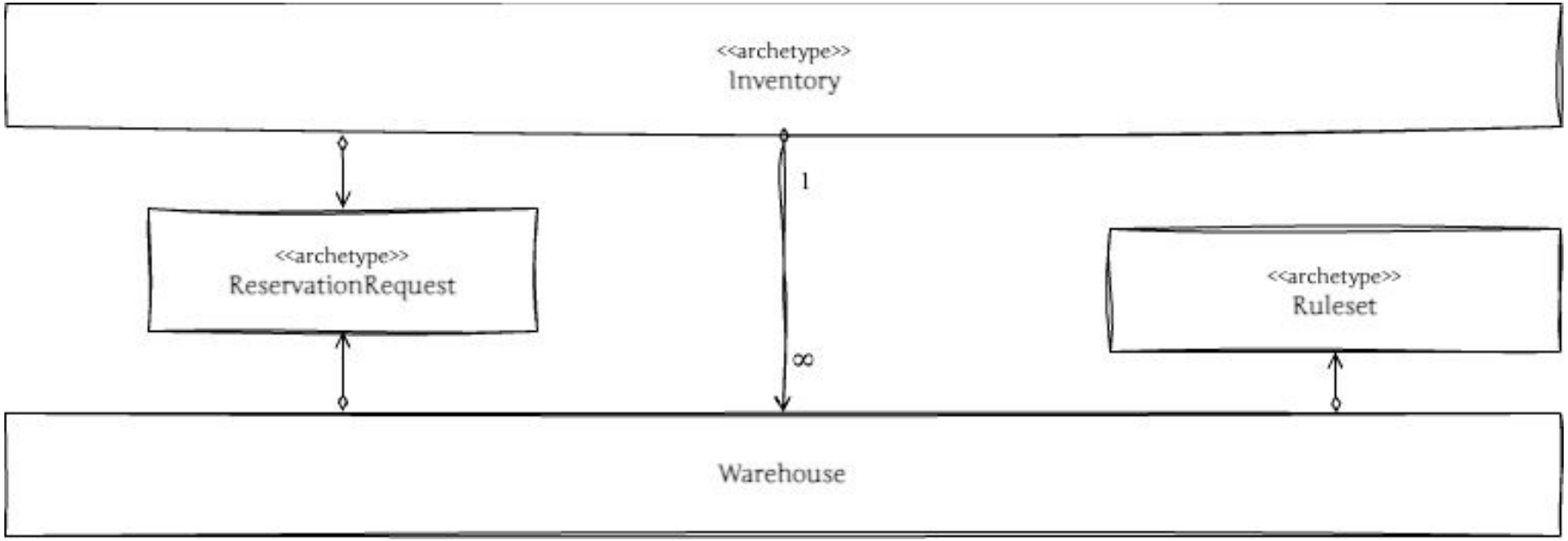
Inventory


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"

Inventory*
multiple warehouses

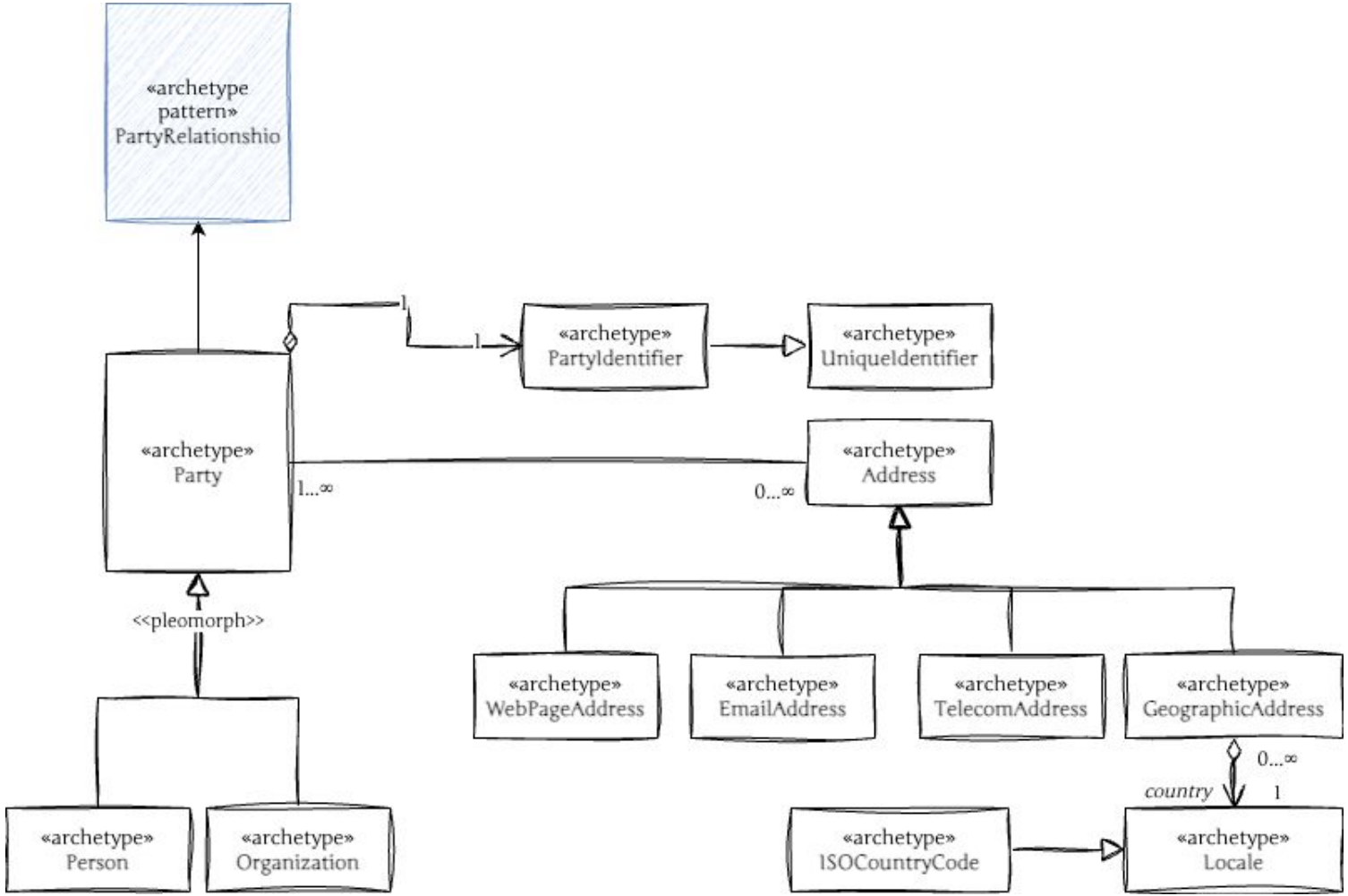


Inventory*

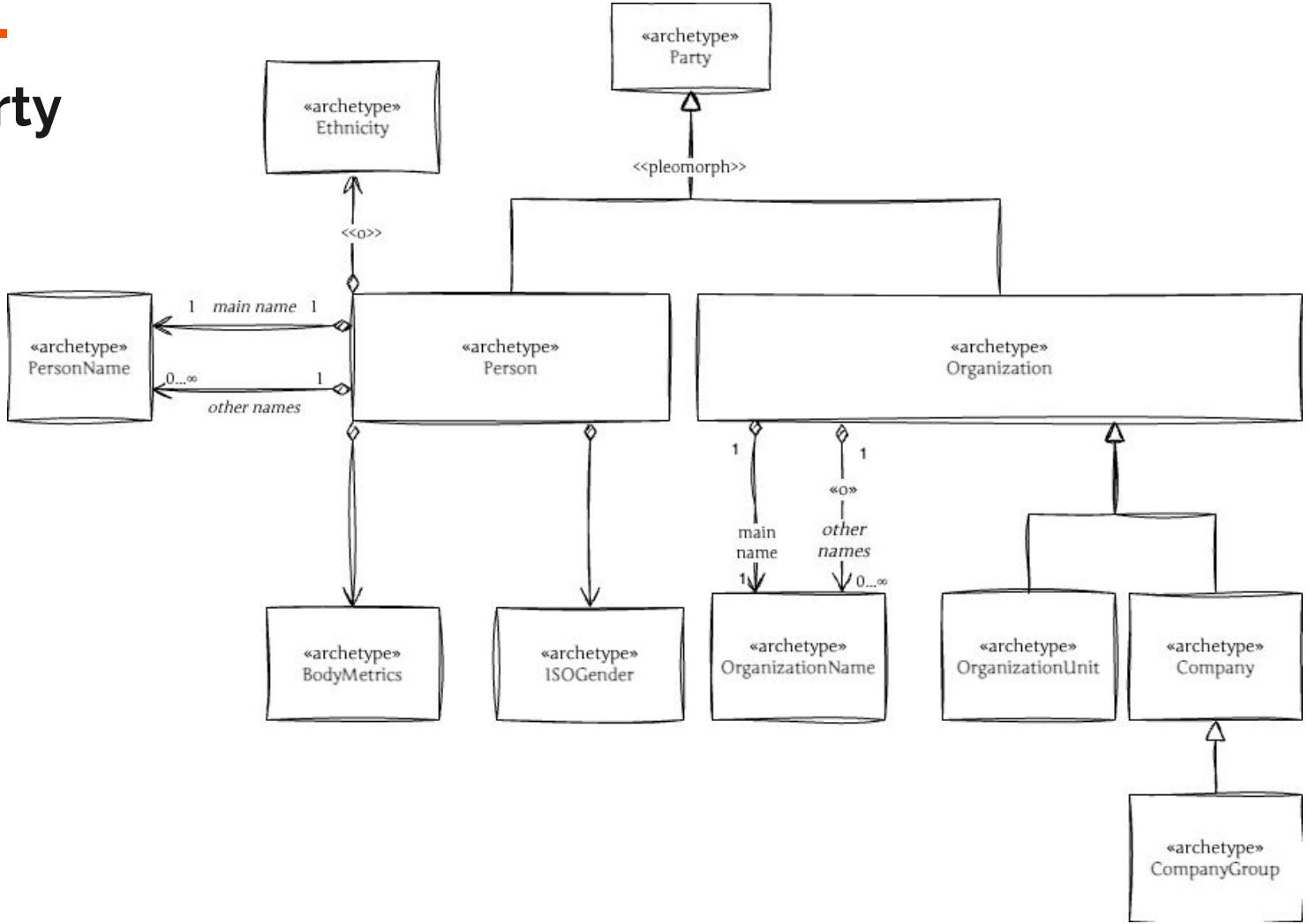

<<archetype>>
Inventory
1
∞
<<archetype>>
ReservationRequest
<<archetype>>
Ruleset
Warehouse

Party & Party Relationship

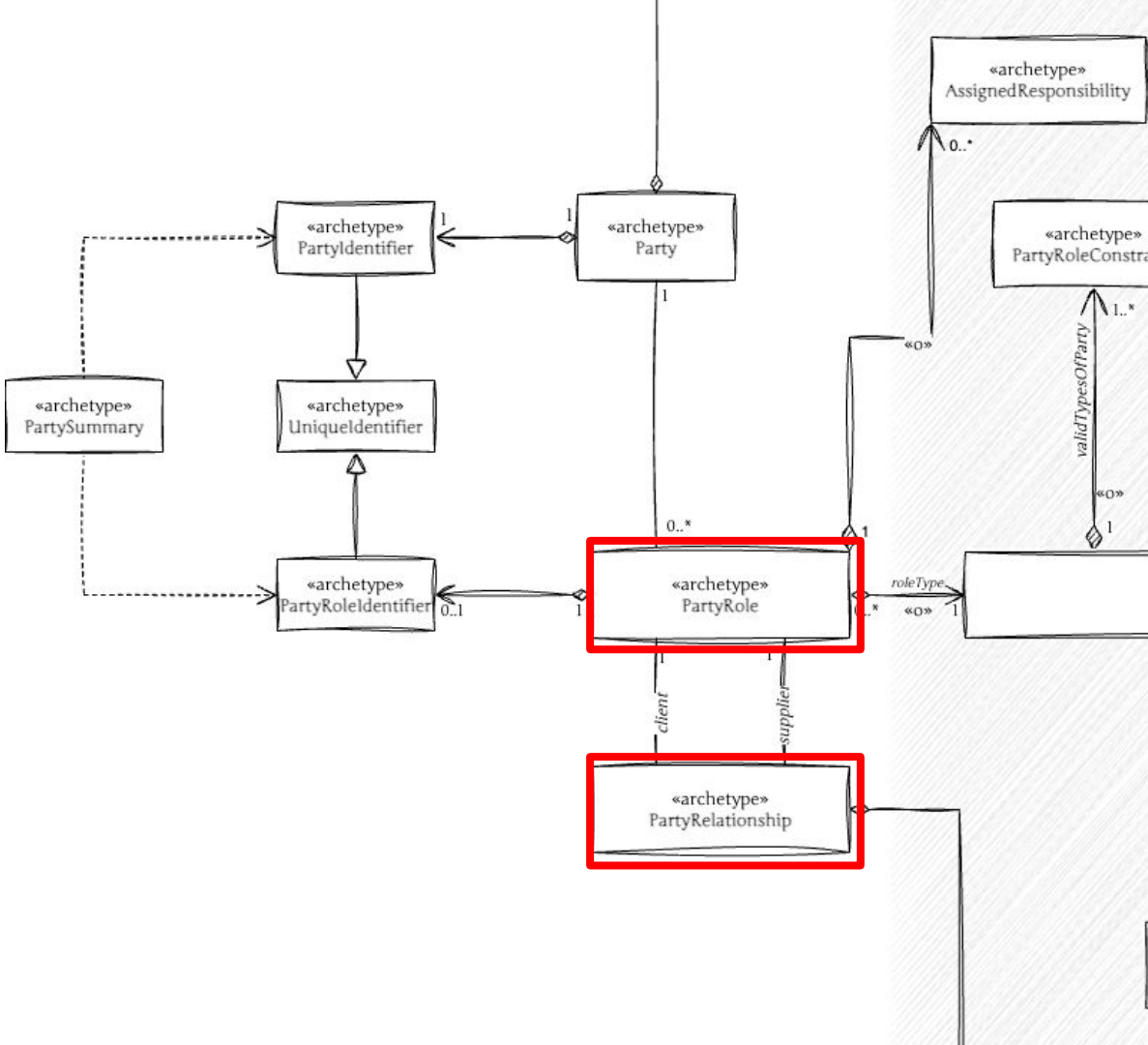
Party


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

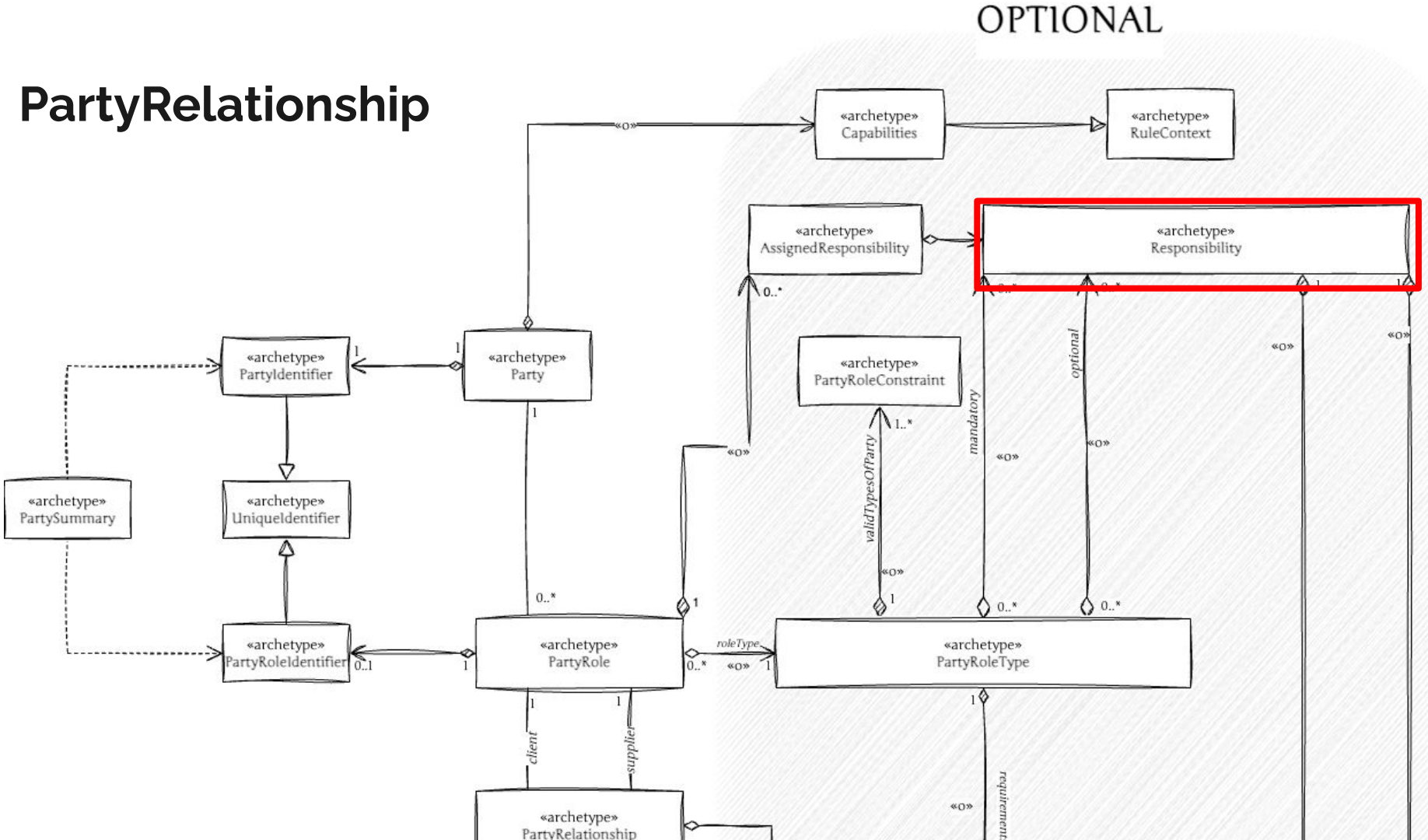
Party


«archetype»
Party
«archetype»
Ethnicity
<<pleomorph>>
<<o>>
«archetype»
PersonName
1
main name
1
0...∞
1
other names
«archetype»
Person
«archetype»
Organization
1
1
«o»
main
name
other
names
1
0...∞
«archetype»
BodyMetrics
«archetype»
ISOGender
«archetype»
OrganizationName
«archetype»
OrganizationUnit
«archetype»
Company
«archetype»
CompanyGroup

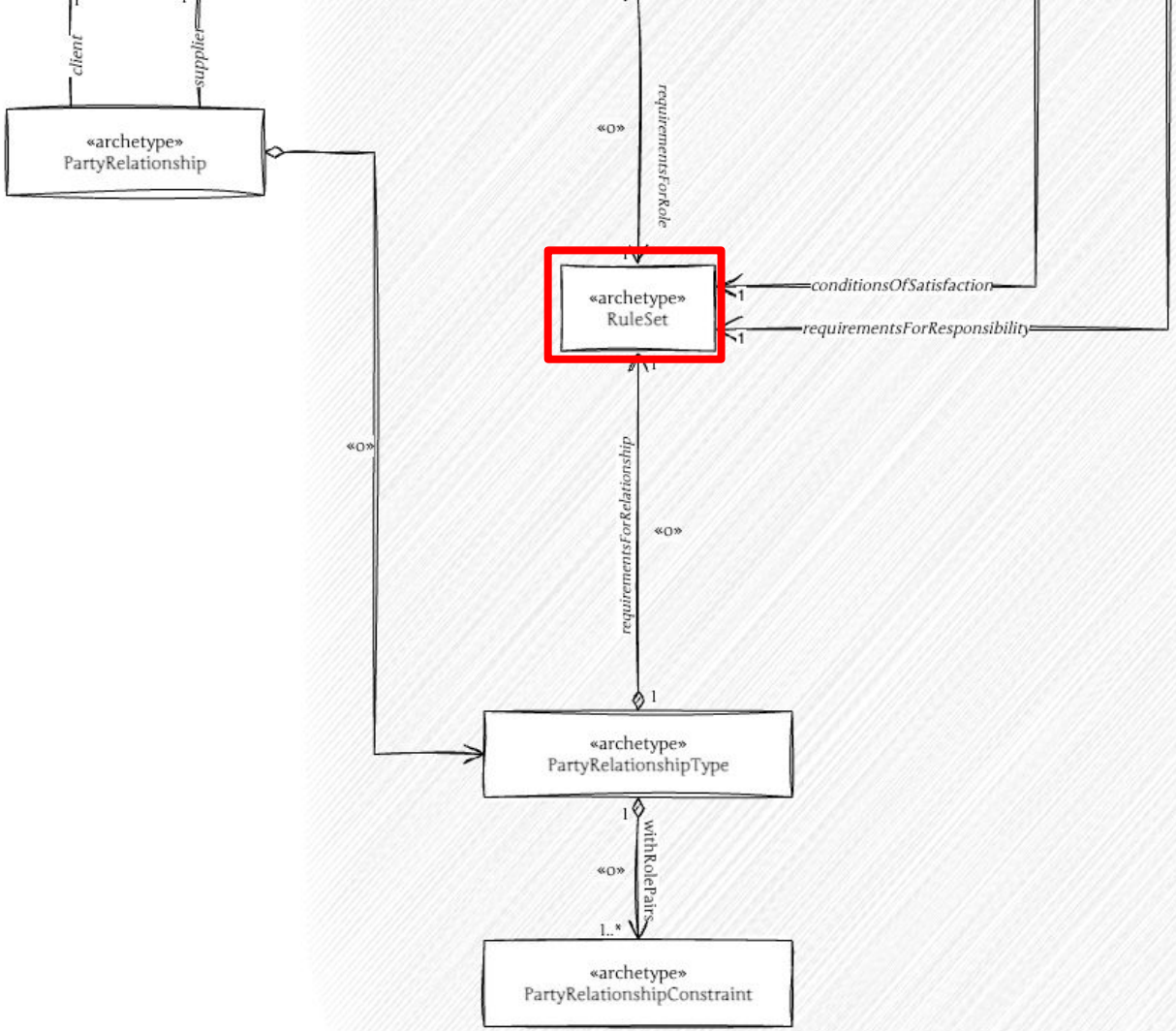
PartyRelationship



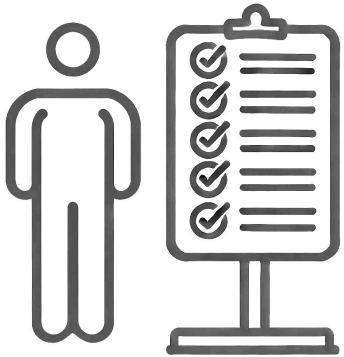
PartyRelationship



PartyRelationship



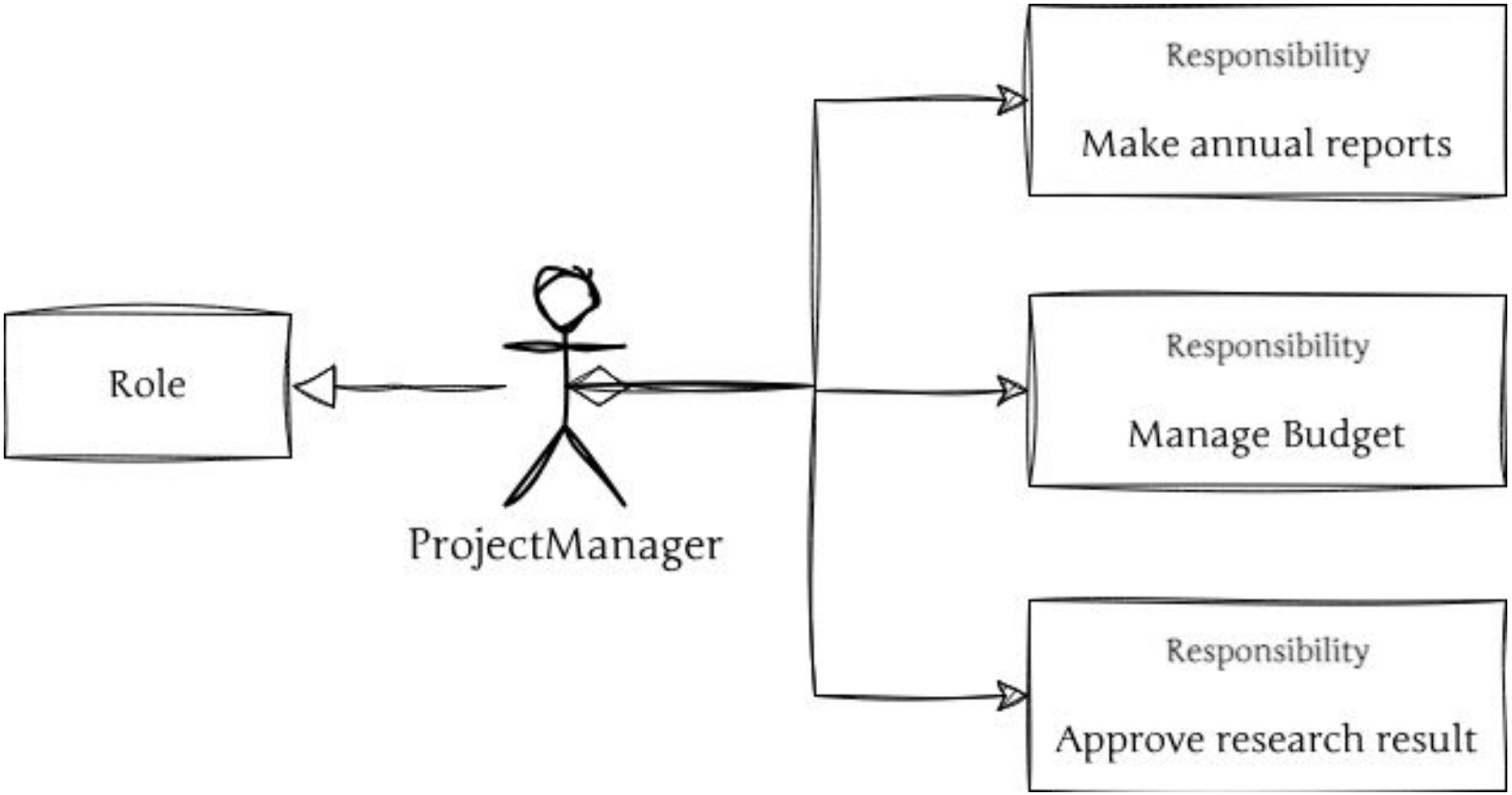
Responsibility

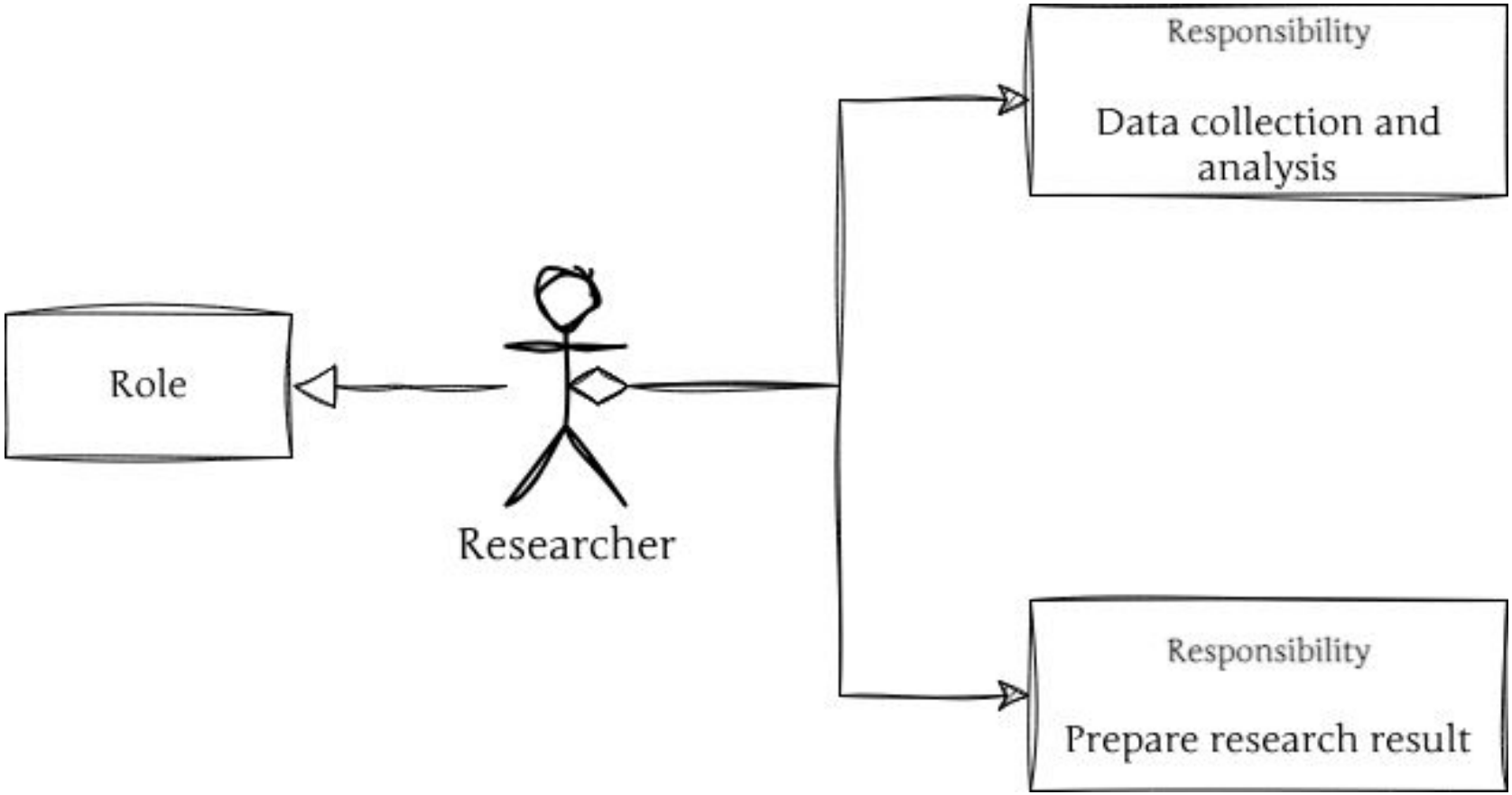


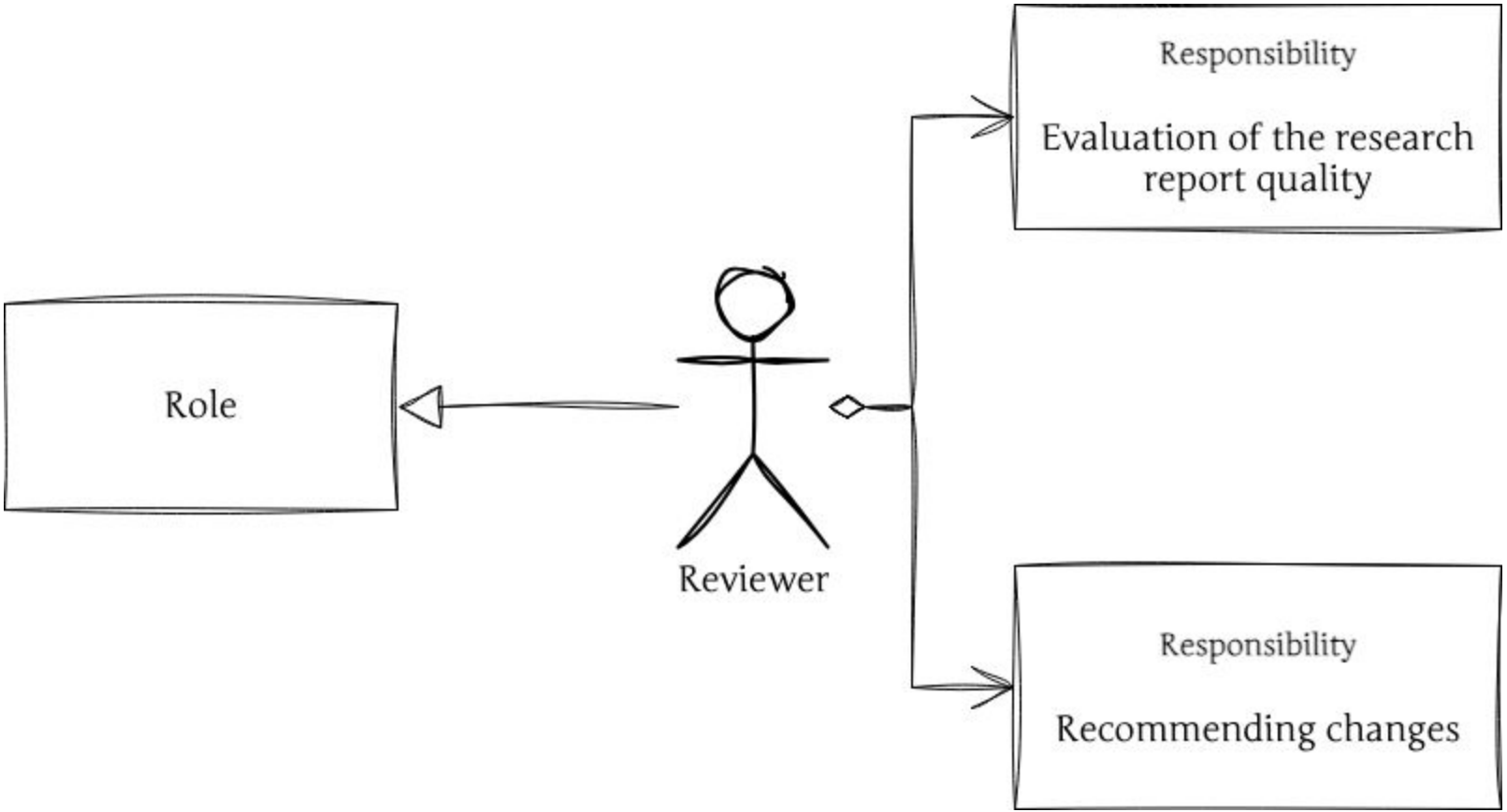
ProjectManager

Researcher

Reviever


Role
ProjectManager
Responsibility
Make annual reports
Responsibility
Manage Budget
Responsibility
Approve research result


Responsibility
Data collection and analysis
Role
Researcher
Responsibility
Prepare research result


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

```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());
    }
}
```

```
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
$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
);
```

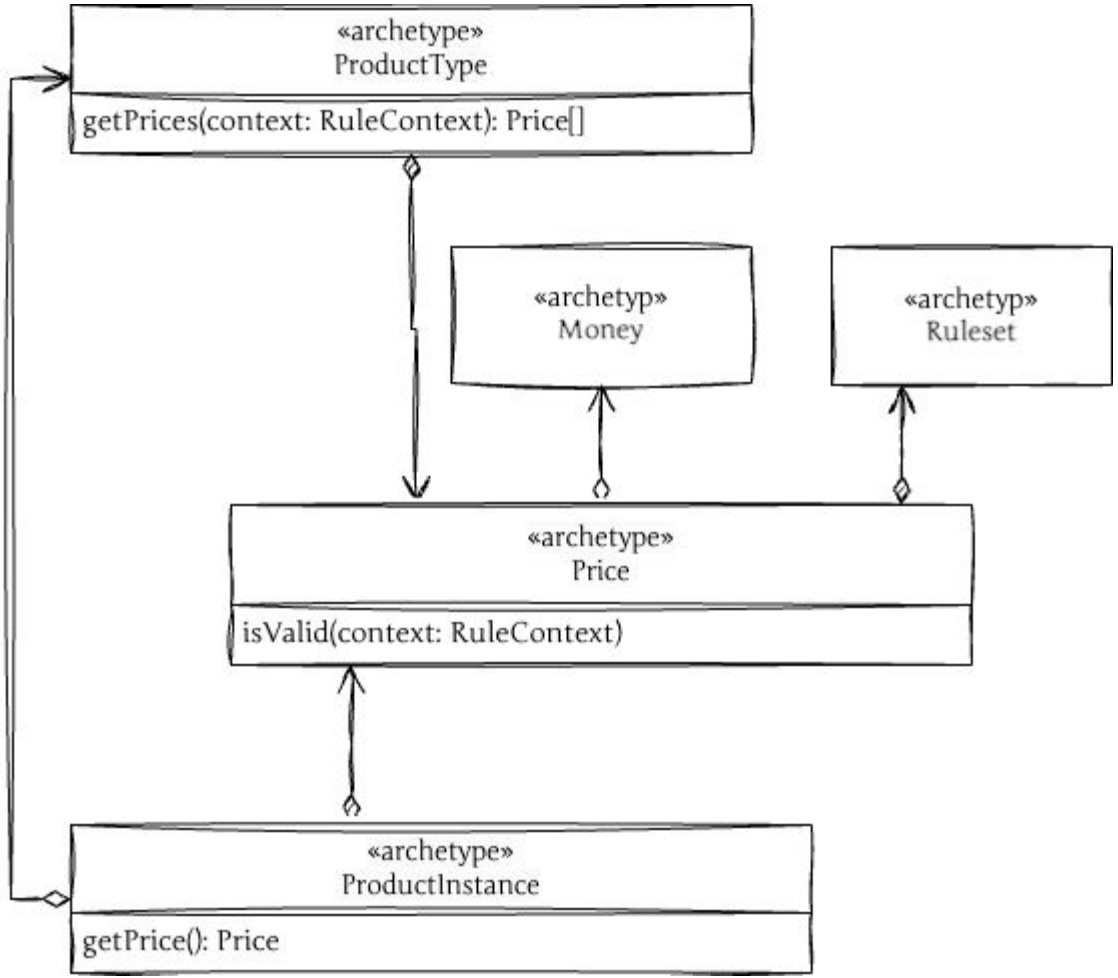
```php
$researcherRole = new RoleOfScientist();
$researcherRole->addResponsibility($analyzeDataResponsibility);
$researcherRole->addResponsibility($prepareReportResponsibility);
$concreteResearcherEmployee = new ScienceEmployee();
$concreteResearcherEmployee->addRole($researcherRole);

// 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,
);
```

Implementacja

Ceny uzależnione od różnych czynników

Price


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

Ceny uzależnione od różnych czynników

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

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

Ceny uzależnione od różnych czynników

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

    2 usages   new *
    public function getPrice(RuleContext $ruleContext): Price
    {
        return $this->prices->filter(fn(Price $price) => $price->isApplicable($ruleContext))->first();
    }
}
```

Ceny uzależnione od różnych czynników

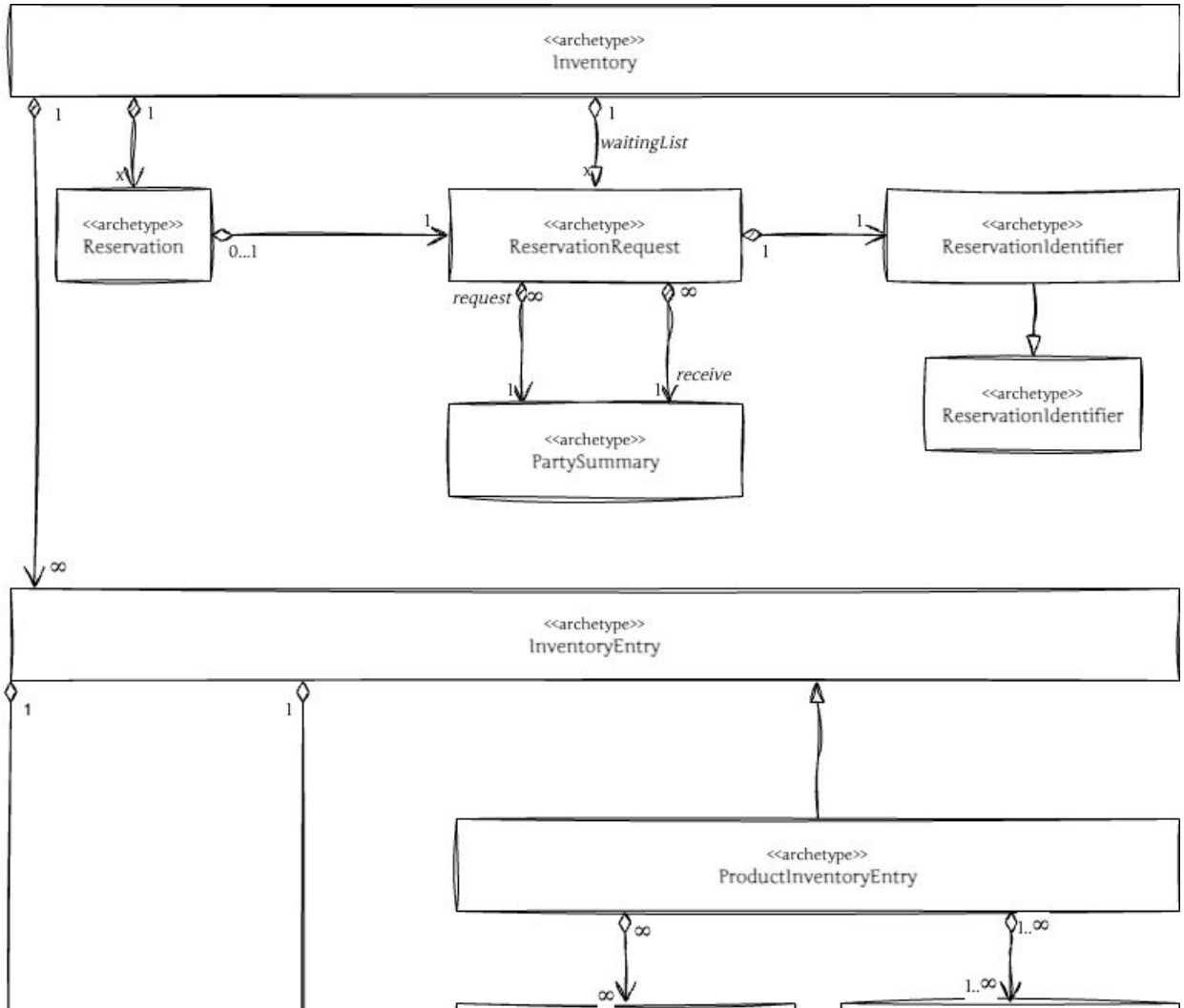
```php
class Pricing
{
    2 usages  new *
    public static function createRuleContext(
        Product $product,
        UnsignedQuantity $quantity,
        CountryCode $location
    ): RuleContext {
        $elements = [
            new Variable('productCategory', $product->getCategory()),
      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

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

Ceny uzależnione od różnych czynników

```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
            )
        );
    }
}
```

Zarządzanie zapasami

Inventory


<<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..∞

Zarządzanie zapasami

```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')));
    }
}
```

Zarządzanie zapasami

```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')));
    }
}
```

Zarządzanie zapasami

```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))

        );
    }
}
```

Zarządzanie zapasami

```php
class InventoryEntry
{
    3 usages  new *
    public function __construct(
        private ProductTypeIdentifier $productIdentifier,
        private GenericList $productInstances,
        private Ruleset $availabilityRules
    ) {
    }

    3 usages  new *
    public function makeReservation(ReservationRequest $request): Either
    {
        $reservationContext = $this->fillContextData($request->reservationContext());

        $isAvailable = $this->availabilityRules->evaluate($reservationContext)->getValue();

        if (!$isAvailable) {
            return Either::left(new ReservationRejected( reason: 'Product is not available'));
        }

        $reservation = Reservation::fromRequest($request);

        /** @var Either<Reservation> $reservationResult */
        $reservationResult = $this->productInstances->fold(
            Either::right($reservation),
            fn(Either $either, ProductInstance $instance) => $either->map(
                fn(Reservation $current) => $instance->reserve($reservation)
            )
        );

        return Either::right($reservationResult->get());
    }
```

Ujarzmienie różnych modeli sprzedażowych

Różne modele sprzedażowe

```php
abstract class Party
{
    protected $partyId;
    protected $name;


    public function __construct($partyId, $name)
    {
        $this->partyId = $partyId;
        $this->name = $name;
    }

    public function getPartyId()
    {
        return $this->partyId;
    }

    public function getName()
    {
        return $this->name;
    }

    abstract public function getPartyType(): PartyType;
    abstract public function getSummary(): PartySummary;
}
```

Różne modele sprzedażowe

```php
class Person extends Party
{
    private string $firstName;
    private string $lastName;

    public function __construct(PartyId $partyId, string $firstName, string $lastName)
    {
        parent::__construct($partyId, $firstName . ' ' . $lastName);
        $this->firstName = $firstName;
        $this->lastName = $lastName;
    }

    public function getPartyType(): PartyType{...}

    public function getFirstName(): string{...}

    public function getLastName(): string{...}

    public function getSummary(): PartySummary
    {
        return new PartySummary(
        // potrzebne dane
        );
    }
}
```

Różne modele sprzedażowe

```php
class Organization extends Party
{
    private string $organizationName;

    public function __construct(PartyId $partyId, string $organizationName)
    {
        parent::__construct($partyId, $organizationName);
        $this->organizationName = $organizationName;
    }

    public function getOrganizationName(){...}

    public function getPartyType(): PartyType{...}

    public function getSummary(): PartySummary
    {
        return new PartySummary(
        // potrzebne dane
        );
    }
}
```

Różne modele sprzedażowe

```php
interface MarketplaceCustomerRole
{
    public function handleMarketplaceOrder(MarkerplaceOrderRequest $request): Result;
}
```

```php
class BusinessClientRole extends PartyRole implements MarketplaceCustomerRole
{
    public function handleMarketplaceOrder(MarketplaceOrderRequest $request): Either
    {
        if (!$this->validate($request)) {
            return Either::left(new OrderRejected('Business requirements not met'));
        }

        if ($request->isBulkOrder()) {
            $request->applyDiscount(10);
        }
        return Either::right(new OrderAccepted($request));
    }

    private function validate(MarketplaceOrderRequest $request): bool
    {
        // Business specific requirements
        return true;
    }
}
```

```php
class IndividualClientRole extends PartyRole implements MarketplaceCustomerRole
{
    public function handleMarketplaceOrder(MarketplaceOrderRequest $request): Ei
    {
        if (!$this->validate($request)) {
            return Either::left(new OrderRejected('Individual requirements not m
        }

        return Either::right(new OrderAccepted($request));
    }

    private function validate(MarketplaceOrderRequest $request): bool
    {
        // Individual-specific validation logic
        return true;
    }
}
```

Różne modele sprzedażowe

```php
class BusinessClientRole extends PartyRole implements MarketplaceCustomerRole
{
    public function handleMarketplaceOrder(MarketplaceOrderRequest $request): Either
    {
        if (!$this->validate($request)) {
            return Either::left(new OrderRejected('Business requirements not met'));
        }

        if ($request->isBulkOrder()) {
            $request->applyDiscount(10);
        }
        return Either::right(new OrderAccepted($request));
    }

    private function validate(MarketplaceOrderRequest $request): bool
    {
        // Business specific requirements
        return true;
    }
}
```

Różne modele sprzedażowe

```php
class IndividualClientRole extends PartyRole implements MarketplaceCustomerRole
{
    public function handleMarketplaceOrder(MarketplaceOrderRequest $request): Either
    {
        if (!$this->validate($request)) {
            return Either::left(new OrderRejected('Individual requirements not met'));
        }

        return Either::right(new OrderAccepted($request));
    }

    private function validate(MarketplaceOrderRequest $request): bool
    {
        // Individual-specific validation logic
        return true;
    }
}
```

Różne modele sprzedażowe

```
interface MarketplaceCustomerRole
```

Abstract Strategy

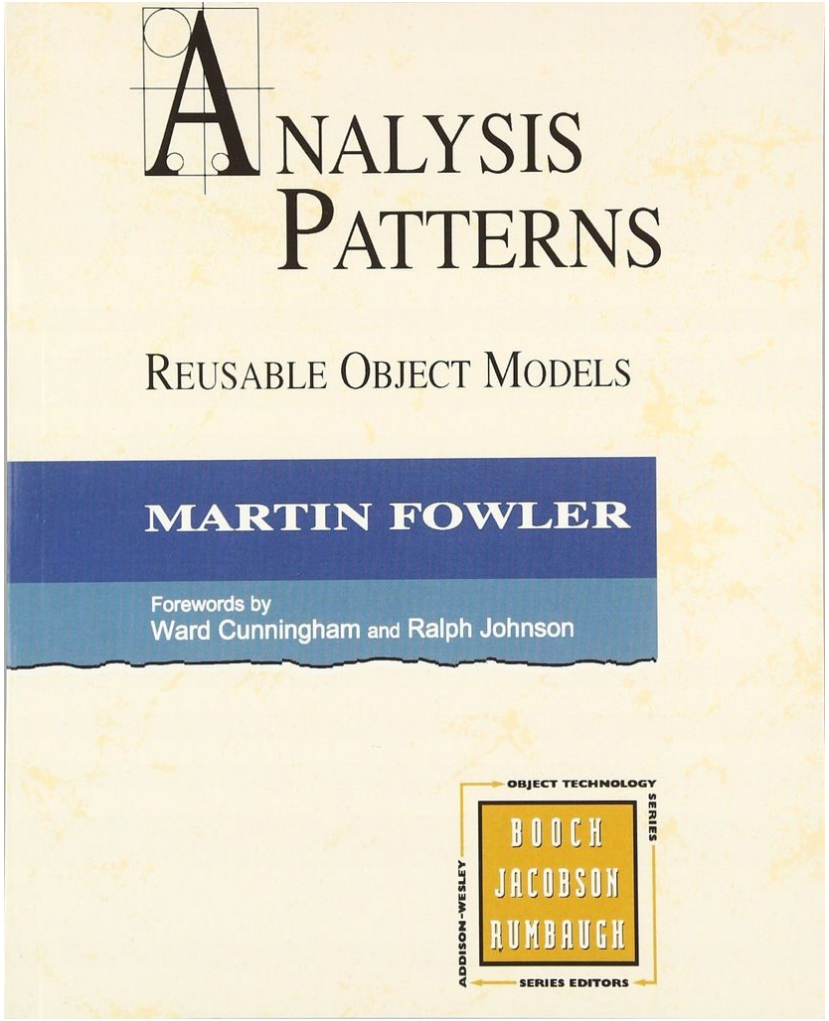
```
class BusinessClientRole extends PartyRole implements MarketplaceCustomerRole
```

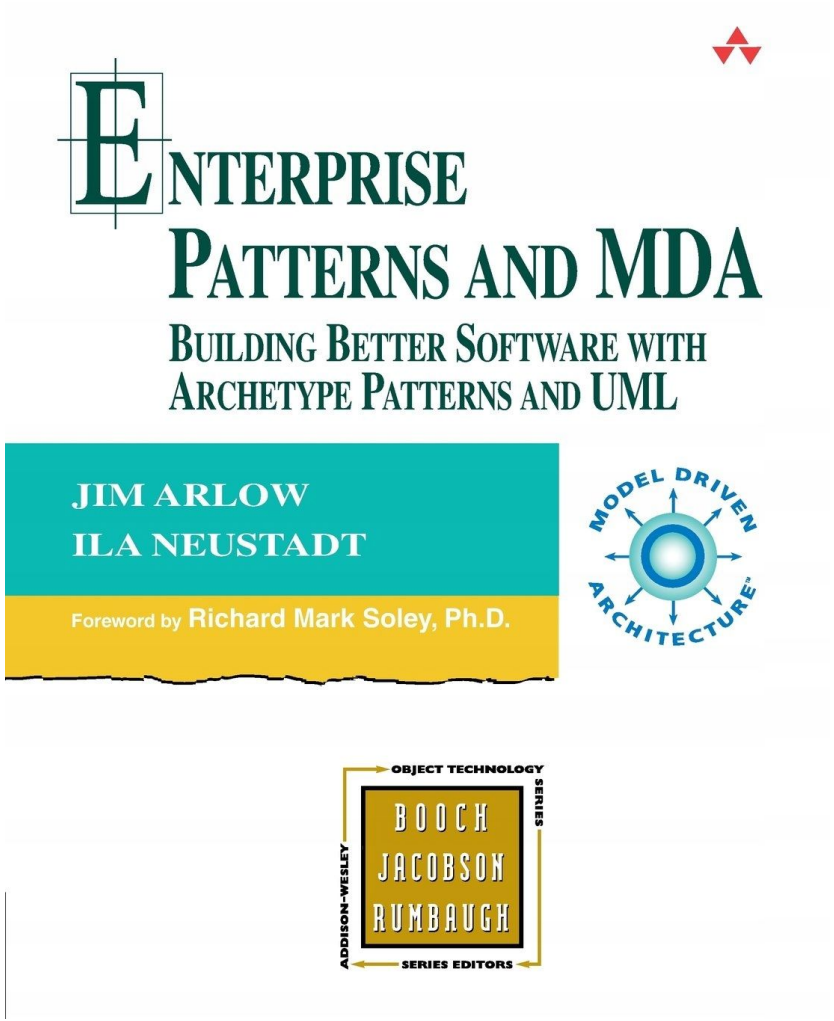
Concrete Strategies

```
class IndividualClientRole extends PartyRole implements MarketplaceCustomerRole
```


phpcon
POLAND
23


phpcon
POLAND
23


ANALYSIS PATTERNS
REUSABLE OBJECT MODELS
MARTIN FOWLER
Forewords by
Ward Cunningham and Ralph Johnson
OBJECT TECHNOLOGY SERIES
BOOCH
JACOBSON
RUMBAUGH
ADDISON-WESLEY
SERIES EDITORS


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

https://softwarearchetypes.com/

Ktoś już to wymyślił

Modelowanie z użyciem archetypów biznesowych.

Jakub Ciszak

phpcon POLAND 25

Oceń prezentację!