Autor

Receptury projektowe – niezbędnik początkującego architekta

Część VI: Wzorce analityczne modeli biznesowych na przykładzie Party – odkrywanie krok po kroku kolejnych rozwiązań.

Techniki takie jak Domain Driven Design służą do odkrywania wiedzy na temat nieznanych domen biznesowych. Jednak często pracujemy nad problemem, który został już dobrze przemyślany i opracowany kilka dekad temu. W niniejszym artykule przedstawimy koncepcję Wzorców analitycznych – czyli wzorców adresujących rozwiązania na poziomie analizy systemowej. Koncepcja wzorców analitycznych będzie ilustrowana na przykładzie kilku wybranych, w tym najbardziej popularnym z nich: Party – będą one również alternatywą dla typowych naiwnych książkowych modeli struktur organizacyjnych.

O serii „Receptury projektowe"

Intencją serii „Receptury projektowe" jest dostarczenie usystematyzowanej wiedzy bazowej początkującym projektantom i architektom. Przez projektanta lub architekta rozumiem tutaj rolę pełnioną w projekcie. Rolę taką może grać np. starszy programista lub lider techniczny, gdyż bycie architektem to raczej stan umysłu niż formalne stanowisko w organizacji.

Wychodzimy z założenia, że jedną z najważniejszych cech projektanta/architekta jest umiejętność klasyfikowania problemów oraz dobieranie klasy rozwiązania do klasy problemu. Dlatego artykuły będą skupiały się na rzetelnej wiedzy bazowej i sprawdzonych recepturach pozwalających na radzenie sobie z wyzwaniami niezależnymi od konkretnej technologii, wraz z pełną świadomością konsekwencji płynących z podejmowanych decyzji projektowych.

MODELOWANIE RÓL POPRZEZ DZIEDZICZENIE – KLASYCZNY STRZAŁ W STOPĘ

W podręcznikach do nauki projektowania obiektowego możemy często spotkać się z przykładem modelu użytkowników systemu, gdzie pojawiają się klasy:

- Użytkownik – klasa bazowa zawierająca imię, nazwisko, login itp.
- Klient – klasa dziedzicząca po Użytkowniku modelująca klientów firmy
- Pracownik – klasa dziedzicząca po Użytkowniku modelująca pracowników firmy

Twórcy takich przykładów zapominają o dosyć powszechnym scenariuszu, w którym pracownik firmy chciałby zostać jej klientem...

Antywzorzec: Wyciąganie przed nawias

Powyższy przykład jest typowym antywzorcem modelowania, gdzie dziedziczenia użyto naiwnie w celu „wyciągnięcia przed nawias" wspólnych danych. Jest to tok rozumowania analogiczny do uczonej w szkole podstawowej reguły rozdzielności mnożenia względem dodawania: a*b + a*c = a * (b+c).

Tak więc skoro klient i pracownik mają imię i nazwisko, to „wyciągnijmy przed nawias" do klasy bazowej. Doświadczeni projektanci obiektowi widzą zapewne w tym miejscu kolejny błąd koncepcyjny: modelowanie danych zamiast zachowań jest wbrew podejściu OO.

Problem kruchej klasy bazowej

Zwykle tego typu błąd projektowy powoduje pojawienie się na poziomie implementacji syndromu „kruchej klasy bazowej": zmiany w klasie bazowej powodują błędy w klasach dziedziczących. W efekcie następuje blokada możliwości wprowadzania zmian w systemie i dziwne obejścia typu sprawdzanie konkretnego typu klasy w runtime.

Liskov Substitution Principle w praktyce

Dziedziczenie jest relacją pomiędzy klasami, które występuje relatywnie rzadko w modelach biznesowych – jest po prostu nienaturalna w większości przypadków. Pojęcie „naturalności" jest oczywiście subiektywne, dlatego warto stosować test Liskov Substitution Principle (jedną z zasad SOLID – zobacz ramkę „W sieci").

W praktyce możemy sprowadzić zasadę LSP do prostej reguły: używamy dziedziczenia tylko wówczas, gdy będziemy stosować polimorfizm. W naszym przykładzie musimy odpowiedzieć sobie na pytanie, czy traktujemy/przetwarzamy pracowników i klientów tak samo?

WZORCE ANALITYCZNE

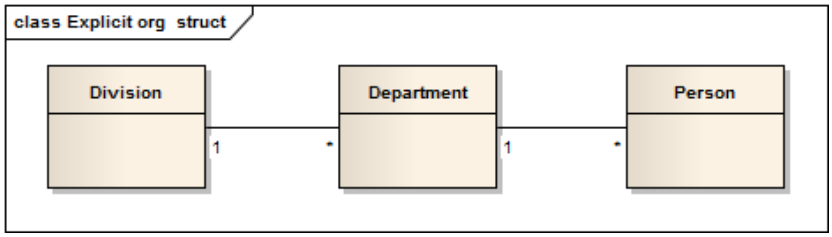
Krótki wstęp miał na celu uświadomienie nam typowych problemów i błędów, jakie spotykamy w modelach obiektowych. Przyjrzymy się teraz technikom i wzorcom modelowania, które ze swej natury podchodzą do problemu nieco inaczej.

Mantra: nie pracuję w NASA

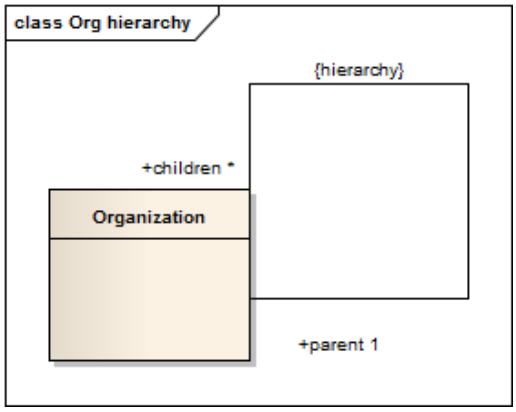
Jeżeli tworzysz standardowe oprogramowanie biznesowe, to najprawdopodobniej pracujesz nad problemem, który został już wielokrotnie rozwiązany i być może nawet dobrze udokumentowany.

Idea wzorców analitycznych

Wzorce analityczne (*analysis patterns*) zostały po raz pierwszy zdefiniowane przez Martina Fowlera. Są to grupy konceptów, które reprezentują wspólne, powszechne konstrukcje w modelowaniu biznesowym. Mogą być adekwatne dla jednej domeny lub rozprzestrzeniać się na wiele domen.



Rysunek 1. Diagram ilustrujący jawną strukturę organizacyjną



Rysunek 2. Diagram ilustrujący hierarchię organizacyjną

Wzorce analityczne wg Martina Fowlera

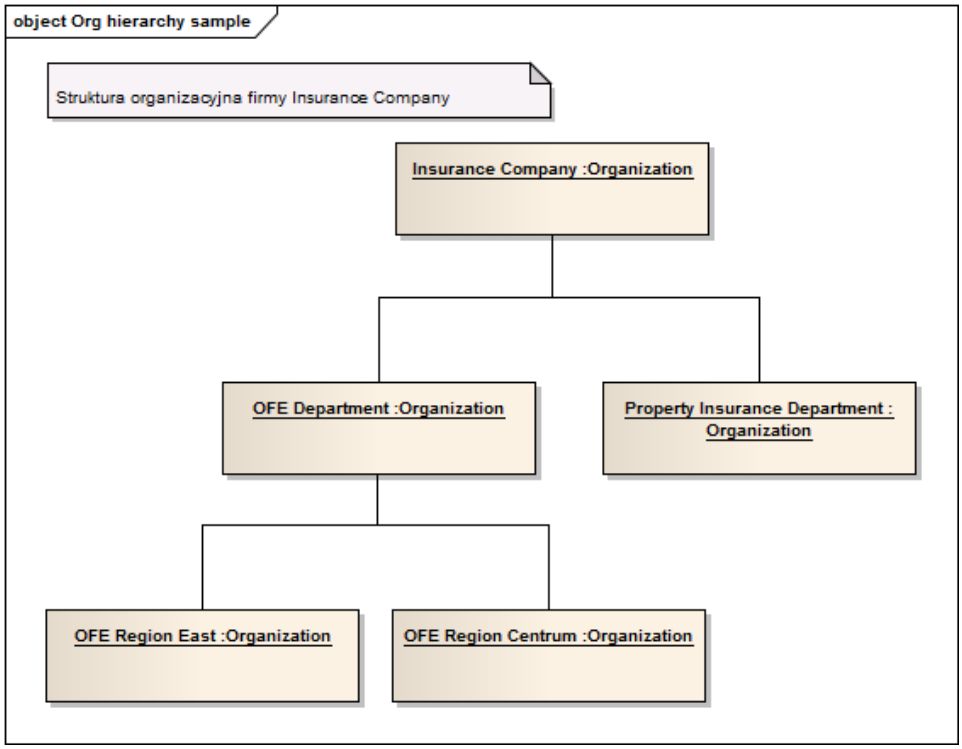
- Struktury organizacyjne (Odpowiedzialność)
- Wzorce dla informacji zmiennych w czasie
- Ilość
- Zakres
- Wzorce księgowe
- Role
- Specyfikacje
- Powtarzające się zdarzenia kalendarzowe
- Własności

W przypadku wzorców dotyczących samej tylko struktury organizacyjnej mamy następujące wzorce:

- Explicit organizational structure
- Organization Hierarchy
- Party
- Accountability
- Knowledge Level

PODEJŚCIE „KROK PO KROKU”

W niniejszym artykule przyjrzymy się bliżej wzorcom dotyczącym struktury organizacyjnej, poznając krok po kroku proces odkrywania kolejnych rozwiązań. Dla każdego z podejść omówimy konsekwencje wynikające z jego zastosowania.



Rysunek 3. Hierarchia organizacyjna - każdy element struktury jest klasy Organization

Explicit organizational structure

Wzorzec analityczny Explicit organizational structure opisuje strukturę organizacyjną wprost: osoby pracują w oddziałach (departamentach), oddziały są zorganizowane w wydziały (dywizje). Każda część struktury jest oddzielną klasą (Rysunek 1).

Jawna struktura organizacyjna ma pewne oczywiste wady. Przede wszystkim jest zbyt sztywna. Przykładowo, jeśli chcemy dodać dodatkowy poziom, np. kanał czy region, to musimy tworzyć nowe dedykowane klasy dla tych jednostek. Ponadto istnieją problemy ze wspólnym zachowaniem pomiędzy różnymi typami organizacji, brak możliwości wydzielenia wspólnych "rzeczy" i zachowań.

Organization Hierarchy

Wzorzec analityczny Organization Hierarchy rozwiązuje pewne problemy, które były obecne we wzorcu Explicit organizational structure.

Funkcjonuje dobrze, jeśli nie ma zbyt wielu różnic w zachowaniu dla różnych elementów struktury, jeśli są, to warto wyróżnić podtypy.

Wzorzec jest elastyczny, jeśli pojawią się nowe rodzaje organizacji (Rysunek 2).

Z uwagi na brak oficjalnego oznaczenia w UML stosuje się oznaczenie poprzez constraint {hierarchy} na asocjacji.

Hierarchia posiada pewne reguły, tj. nie mogą istnieć cykle – dziecko nie może być własnym przodkiem.

Można zauważyć, iż wzorzec Organization Hierarchy jest niczym innym jak zastosowaniem wzorca projektowego Composite wg GOF w swej zdegenerowanej formie (liście i węzły razem).

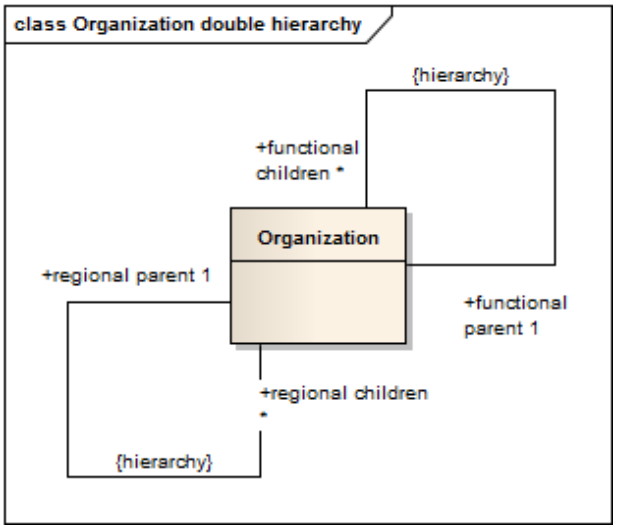
Przykład hierarchii organizacyjnej

Firma o nazwie Insurance Company posiada 2 wydziały, tj. Wydział OFE (Otwarty fundusz emerytalny) i Wydział ubezpieczeń majątkowych. W ramach Wydziału OFE jest podział na regiony: Region OFE Wschód i Region OFE Centrum. Każdy element struktury jest instancją klasy Organization (Rysunek 3).

Podwójna hierarchia organizacyjna

Jeżeli organizacja ma dwie różne struktury, np. regionalną i funkcjonalną, to możemy wprowadzić podwójną hierarchię.

Autor



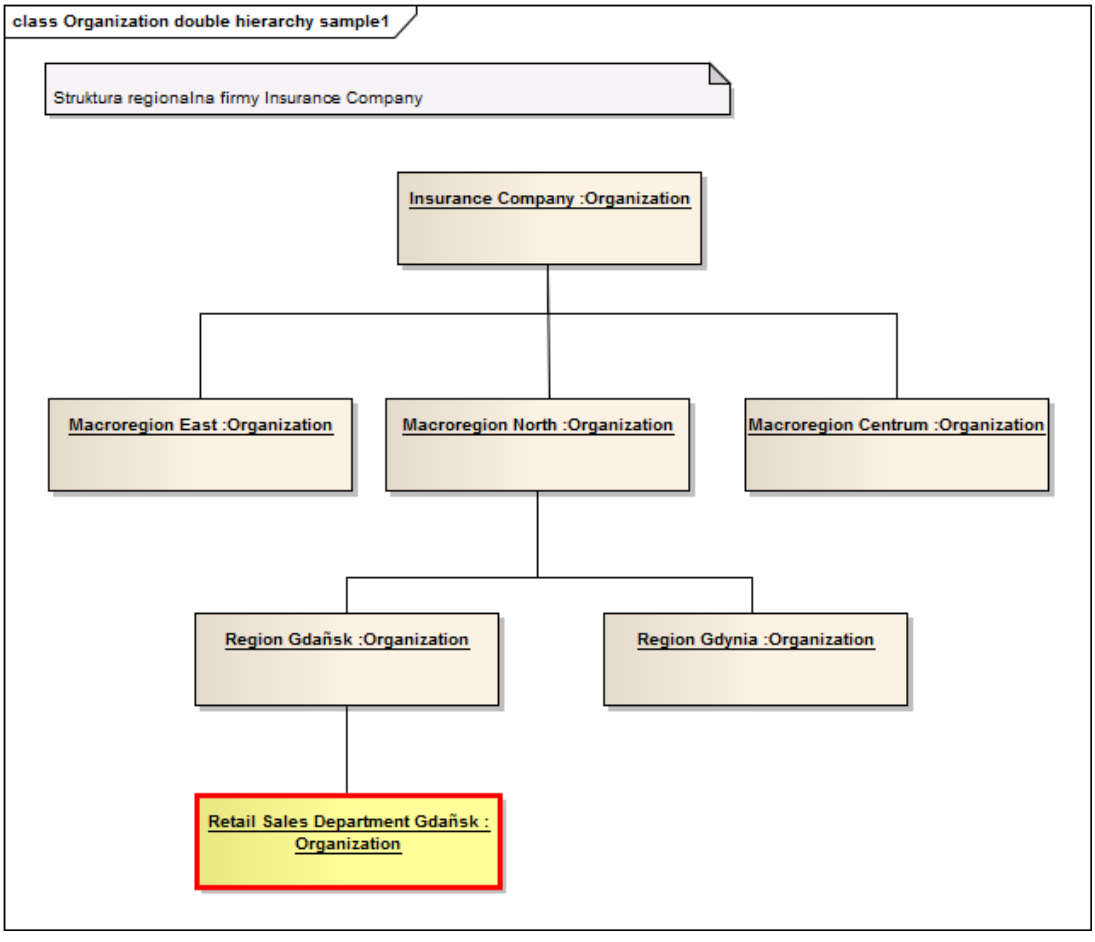
Rysunek 4. Podwójna hierarchia organizacyjna

Podwójna hierarchia organizacyjna – Struktura regionalna

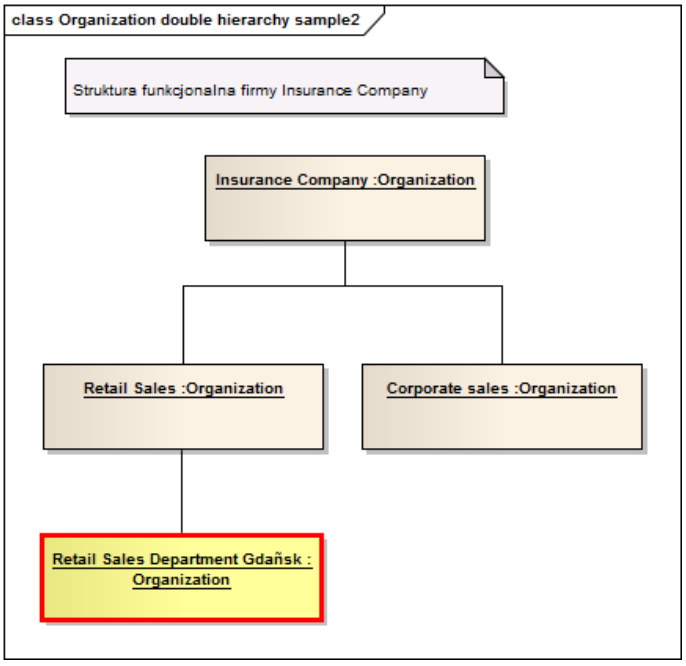
Dział sprzedaży detalicznej w Gdańsku jest umiejscowiony w strukturze regionalnej (w Regionie Gdańsk) (Rysunek 5).

Podwójna hierarchia organizacyjna – Struktura funkcjonalna

Dział sprzedaży detalicznej w Gdańsku jest umiejscowiony w strukturze funkcjonalnej (w Sprzedaży detalicznej).



Rysunek 5. Podwójna hierarchia organizacyjna - Struktura regionalna



Rysunek 6. Podwójna hierarchia organizacyjna - Struktura funkcjonalna

Problem powstaje, gdy elementy struktury są różnego typu (osoby, organizacje). Zakładamy, że pewne cechy i zachowania są wspólne.

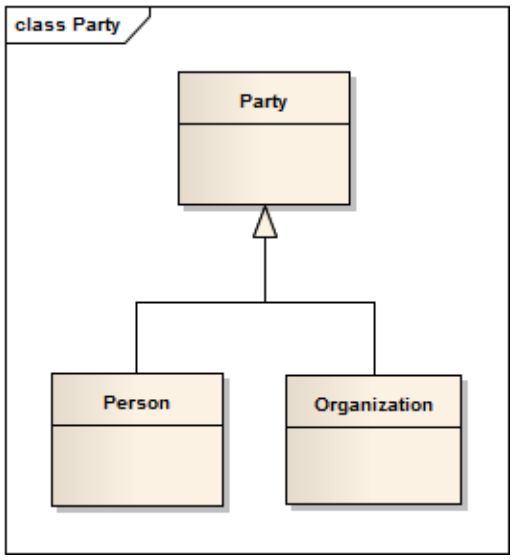
Party

W ten sposób dochodzimy do wzorca Party, który można tłumaczyć jako: uczestnik, strona, jednostka. Party reprezentuje jednostkę, którą można

zidentyfikować (oznaczyć), można określić jej adres, często posiadającą status prawny. Zwykle reprezentuje osobę lub organizację. Wzorzec Party opisuje, jak reprezentować niezbędne informacje o osobach i organizacjach.

Supertyp Party często w zastosowaniach przybiera nazwy: Player, Person/Org, Contractor, Legal Entity, Unit,...

Aktualne nazwa Party jest już standardem, przez co znacznie ułatwia komunikację w procesie wytwarzania oprogramowania.

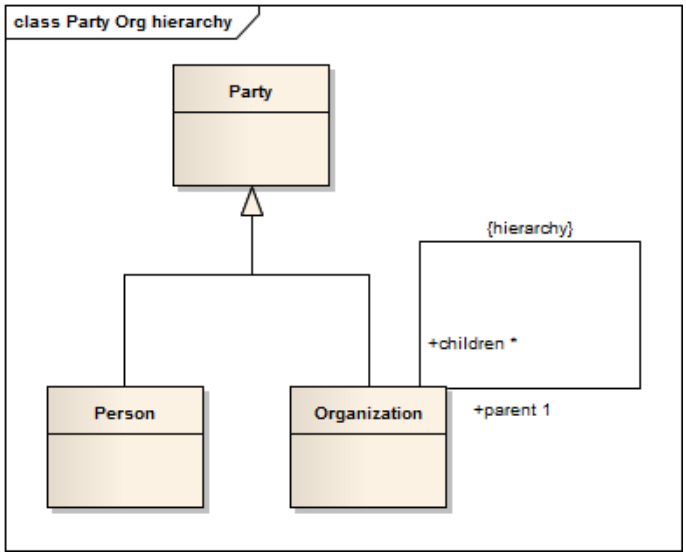


Rysunek 7. Klasa bazowa Party i podklasy Person i Organization

Kiedy używać wzorca Party? Koniecznie gdy w modelu są osoby i organizacje i posiadają wspólne cechy i zachowanie. Również gdy nie ma potrzeby rozróżniania na osoby i organizacje, wówczas wystarczy ograniczyć się do supertypu Party i nie tworzyć podtypów Party i Organization.

Modelowanie struktury organizacyjnej: Party i Organization Hierarchy

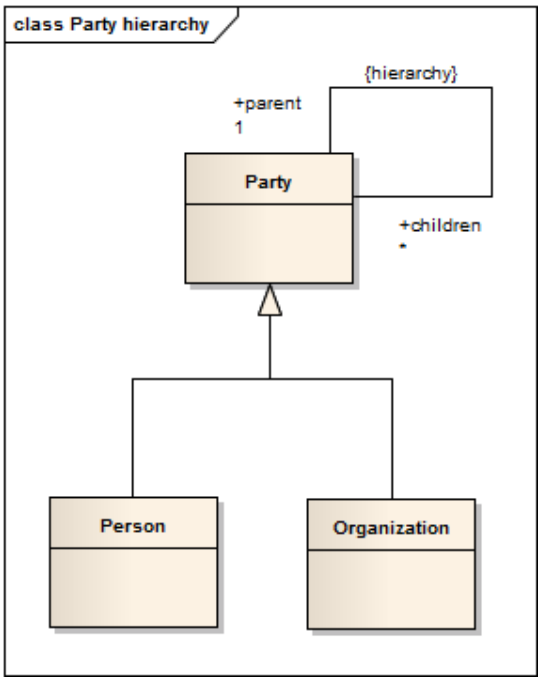
W modelowaniu struktury organizacyjnej przy pomocy Party możemy zastosować hierarchię dotyczącą samej organizacji.



Rysunek 8. Party i hierarchia dotycząca organizacji

Hierarchia dotycząca Party

Możemy również przesunąć hierarchię do poziomu supertypu Party.



Rysunek 9. Hierarchia dotycząca Party

Jest to wystarczające rozwiązanie dla większości struktur organizacyjnych.

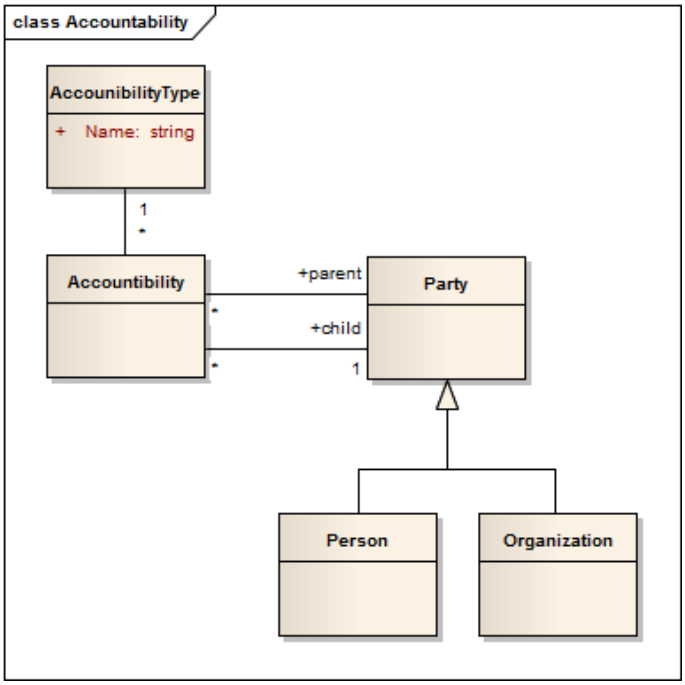
Problem powstaje, gdy rośnie liczba powiązań pomiędzy Party, np. ludzie są zorganizowani jednocześnie pod względem stanowiska i podziału regionalnego.

Można wprowadzić drugą i kolejną hierarchię, ale wtedy poziom komplikacji wzrasta.

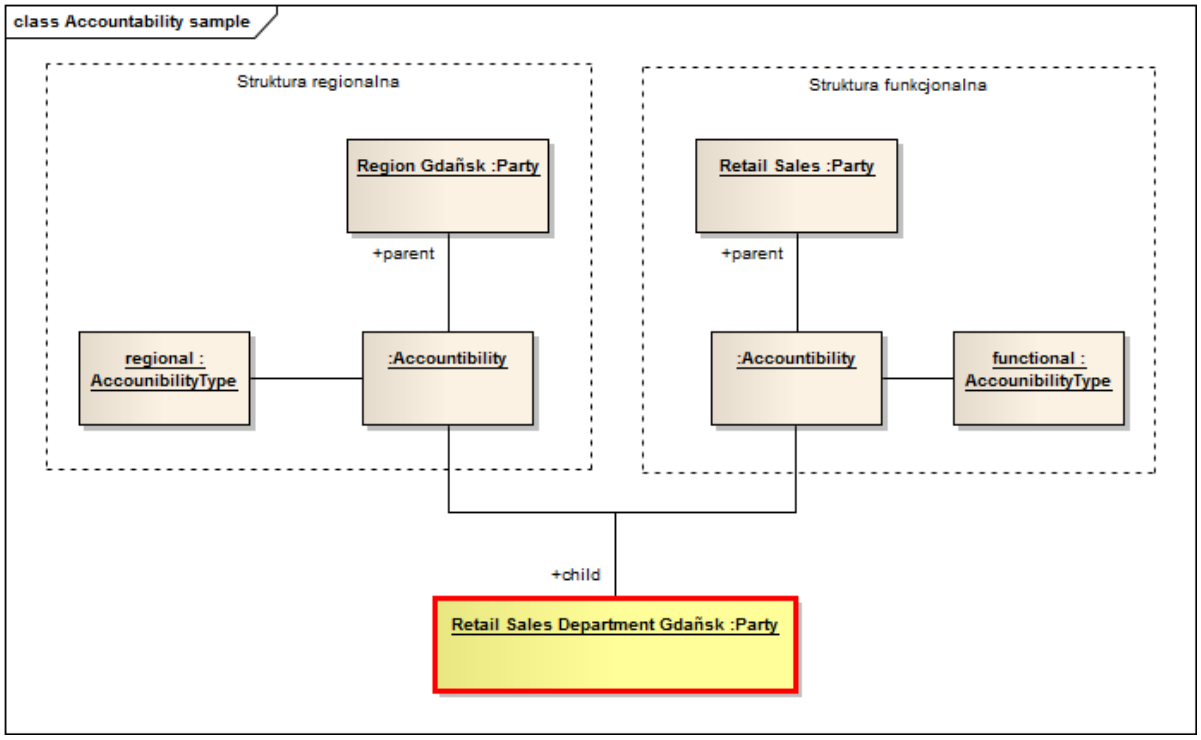
Rozwiązaniem jest Wzorzec analityczny Accountability.

Accountability (Odpowiedzialność)

Jest to złożony graf relacji pomiędzy Party. Każda instancja Accountability stanowi połączenie pomiędzy dwoma instancjami Party, natomiast Accountability Type wskazuje naturę tego połączenia. Rozwiązanie pozwala obsłużyć dowolną ilość relacji pomiędzy jednostkami Party. Istnieje reguła określająca, iż nie mogą istnieć cykle.



Rysunek 10. Modelowanie relacji poprzez wzorzec Accountability



Rysunek 11. Dział sprzedaży detalicznej w Gdańsku jest dzieckiem Regionu Gdańsk w strukturze regionalnej i dzieckiem Sprzedaży Detalicznej w strukturze funkcjonalnej

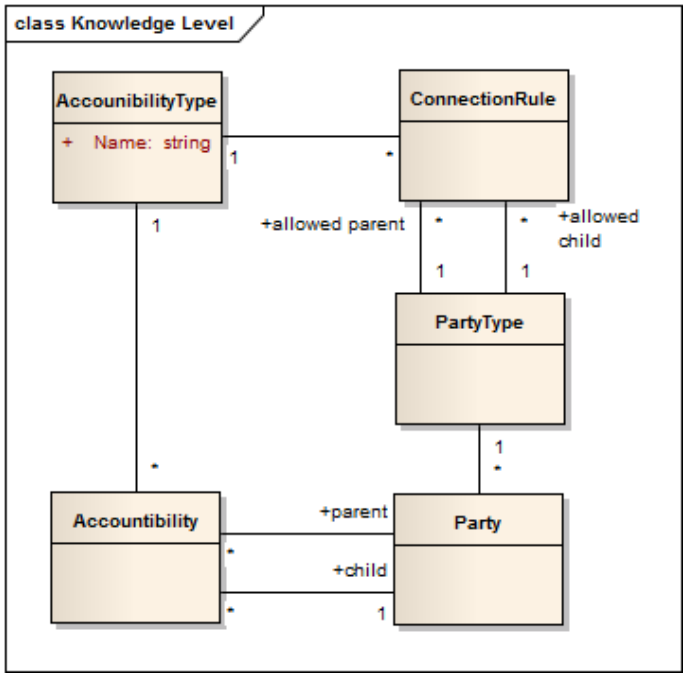
Wariacje wzorca Accountability

Podstawowa forma wzorca Accountability nie ogranicza, które jednostki mogą się łączyć z innymi w relacji. Problem powstaje, gdy w pewnych okolicznościach dana relacja nie ma sensu, np. Dział sprzedaży detalicznej nie może być podległy wobec Działu sprzedaży korporacyjnej lub Agent Kowalski nie może być szefem Kierownika Oddziału.

Rozwiązaniem jest wprowadzenie Poziomu Wiedzy (tzw. Knowledge Level) zawierającego reguły określające, które typy Party mogą wchodzić razem w relację i jakiego typu jest ta relacja.

Accountability Type zawiera grupę obiektów klasy Connection Rule.

Każda z reguł Connection Rule definiuje legalną parę (dozwolony rodzic i dziecko) pomiędzy typami Party.



Rysunek 12. Knowledge Level

W przykładzie z Rysunku 12 Knowledge Level służy do sprawdzania poprawności powiązań. W momencie tworzenia instancji klasy Accountability, sprawdzane jest, czy dane typy PartyType mogą utworzyć żądany AccountabilityType.

Knowledge Level

Wzorzec Knowledge Level jest znany również jako Meta Level.

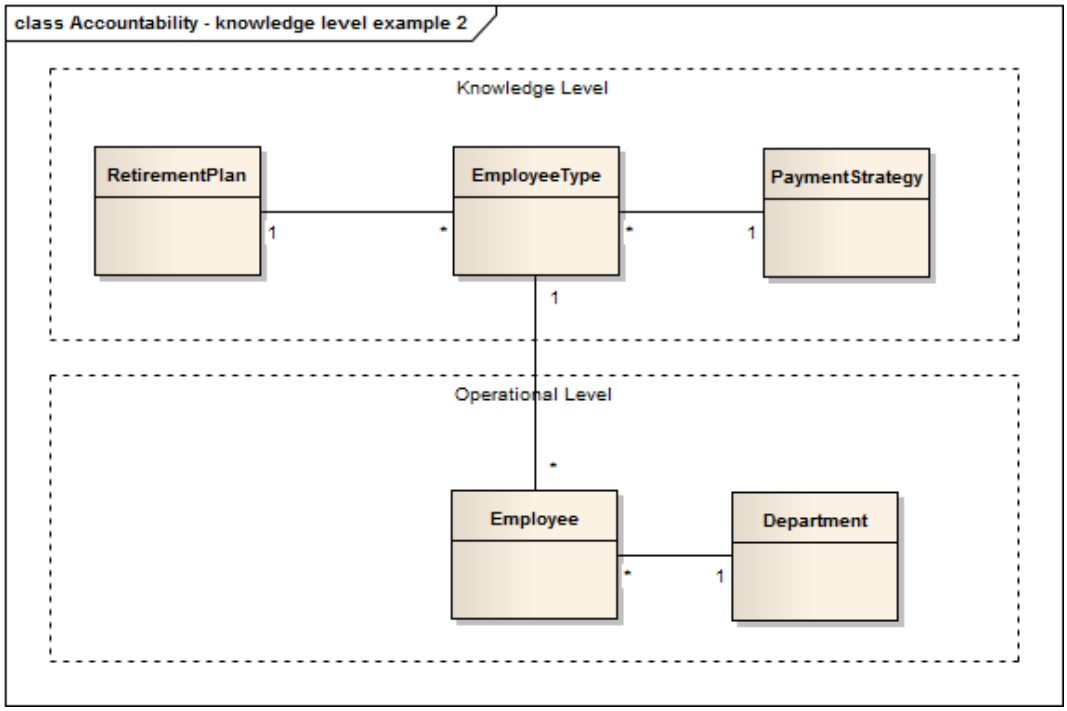
Jest to grupa obiektów, która definiuje, jak inna grupa obiektów powinna się zachowywać. Pozwala konfigurować system bez programowania lub przynajmniej łatwiej wprowadzać zmiany w kodzie.

Jako przykład wyobraźmy sobie sytuację (Rysunek 13), w której mamy dobrze zdefiniowany model pracowników (szczególny rodzaj party), jednak dyrektor HR dokonuje częstych zmian co do stanowisk (typów pracowników). Dzięki wprowadzeniu wzorca Knowledge Level możemy przygotować się na takie zmiany.

Typ pracownika określa strategię płatności dla planu emerytalnego. Pracownik pracuje w wydziale. Można dodać nowy typ pracownika, nowe strategie płatności bez wpływu na model na poziomie operacyjnym (Rysunek 13).

WZORCE ANALITYCZNE JAKO POCZĄTEK ANALIZY DOMENOWEJ

Eric Evans (twórca Domain Driven Design) powiedział, że DDD służy do iteracyjnego odkrywania nowych, nieznanych domen w procesie Knowledge Crunching; natomiast wiele domen jest dobrze poznanych i udokumentowanych. Dlatego stosując DDD do takich przypadków, warto posiłkować się katalogiem wzorców analitycznych jako punktem wyjścia. Wzorzec pozwala szybko nadać zgrubną strukturę modelowi i wyłonić główne Agregaty (jeden z Building Blocks DDD). Struktury są poddawane dalszej analizie DDD, czyli szukamy ich niezmienników (reguł biznesowych), które będą hermetyzowane wewnątrz klas oraz odpowiedzialności (metody) modelujących dynamikę działania biznesu.



Rysunek 13. Accountability – Knowledge Level

WZORCE ANALITYCZNE JAKO CZYNNIK PRZEWAGI

Już nawet tak podstawowe wzorce analityczne jak Party i Knowledge Level wprowadzają więcej punktów swobody do modelu. Dzięki nim możemy szybciej reagować na zmiany w biznesie.

Przykładowo systemy wspierające działanie firm ubezpieczeniowych, które pozwalają na szybkie wprowadzanie zmian na poziomie technicznym, w efekcie pozwalają szybciej reagować na potrzeby rynku na poziomie biznesowym, co daje niebagatelną przewagę nad konkurencją.

W sieci

- Prace Martina Fowlera: http://martinfowler.com/tags/analysis%20patterns.html
- Liskov substitution principle: http://en.wikipedia.org/wiki/Liskov_substitution_principle
- SOLID: http://en.wikipedia.org/wiki/Solid_(object-oriented_design)

Dorota Pawelec-Sobótka dorota.pawelec@bottega.com.pl

Konsultantka i koordynatorka szkoleń w firmie Bottega IT Solutions. W swej karierze prowadziła projekty, dokonywała analiz systemowych i projektowała systemy informatyczne klasy enterprise oparte o platformę .NET.

215x80