INFORMATYKA 5/98 s.18-22

Zygmunt Ryznar

OBIEKTOWO-DYNAMICZNE PODEJŚCIE I JEGO JĘZYK SPECYFIKACYJNY

(próba definicji na przykładzie bankowości)

Uwagi na temat obiektowego podejścia

Nie jest moim zamiarem dokonanie oceny obiektowych metod analizy i projektowania. Jest ich wiele (ok.50) i przeważnie są niespójne ze sobą pojęciowo i notacyjnie (każda idzie „swoją drogą”)*, co właściwie prowadzi do specyfikacji różnic zamiast syntezy. Zostawiam więc to zadanie „badaczom-historykom”.*

Jednakże coś *zawdzięczamy obiektowemu podejściu lub moglibyśmy zawdzięczać* (gdyby kierunek ten konsekwentnie doprowadzono do pomyślnego końca)*.*

Na to „coś” składają się moim zdaniem:

- powrót do „natury” czyli do integracji struktur danych i kodu programistycznego, a więc rezygnacja z niezależności danych typowej dla klasycznych systemów zarządzania bazami danych,
- możliwość uzyskania transparentności (przejrzystości wzajemnej) na dowolnym poziomie ujęcia (biznesowym, projektowym, programistycznym) pod warunkiem stosowania obiektowego podejścia na wszystkich etapach budowy systemu;
- analiza obiektowa jako narzędzie dotarcia do istoty problemu biznesowego,
- stworzenie przesłanek realizacji „marzenia”, polegającego na tworzeniu elastycznych systemów informatycznych poprzez „montaż” jego elementarnych cząstek zwanych obiektami.

Obiektowe podejście stwarza więc pewne możliwości. Na przykład, transparentność stwarza podstawy komunikacyjne pomiędzy różnymi specjalistami pracującymi nad budową systemu (konsultanci, biznesowcy, analitycy,. programiści) oraz przesłanki automatycznego przejścia od specyfikacji problemu do automatycznie wygenerowanego oprogramowania.
Na ile możliwości te są realizowalne już dzisiaj i czy nie będą zastąpione kolejnym modnym podejściem , niestety, nie potrafię odpowiedzieć.

Mogę natomiast skomentować dotychczasowy rozwój podejścia obiektowego z punktu widzenia uwzględniania dynamicznej strony obiektów, czyli procesów i zdarzeń, które zachodzą z ich udziałem.

Wydaje się, że *podejście ER* (Entity Relationship) zaproponowane przez Petera Chena w 1976 roku narzuciło pewien standard rozumowania, nie doceniający dynamicznych aspektów obiektu. Krok pewien w kierunku odwzorowania dynamicznego stanowił model informacyjny *Shlaera i Mellora* (1988 r), w ktorym obiekt charakteryzowany (czy też identyfikowany) jest przez rolę jaką odgrywa, przez zdarzenia pojawiające się w określonym momencie czasu czy też przez tzw.interakcje w jakie wchodzi obiekt (np.zawarcie kontraktu). W metodzie *OMT (Object Modeling Technique) Rumbaugh* docenione zostaje zachowanie się obiektu określane jako operacja, w szczególności dotyczy to operacji polimorficznej znajdującej zastosowanie w stosunku do obiektów należących do różnych klas. Inną zaletą Rumbaugh jest oderwanie się od technologii relacyjnych baz danych dzięki zaprzestaniu używania kluczy pierwotnych i obcych (primary and foreign keys).

Sądzę, że mimo 25 letniej historii strukturalnego i obiektowego podejścia, wiele jeszcze pozostaje do dokonania w kierunku *dynamicznego modelowania biznesowej symulacji obiektów*, opartego na symulacji działalności przedsiębiorstwa i posiadającego odpowiednie przełożenie na komputerową technologię realizacji od strony obiektowych języków programowania, obiektowych baz danych, obiektowych baz wiedzy oraz narzędzi CASE wyposażonych w odpowiednie repozytorium do utrzymywania definicji klas i obiektów.

W obiektowym podejściu często stosowany jest pogląd, iż właściwą metodą wykrywania obiektów oraz ustalania ich zachowania się jest *modelowanie działalności biznesowej a więc odwzorowanie rzeczywistości niejako na poziomie ontologicznym czyli w kategoriach teorii bytu.* Oznacza to jakościową zmianę w stosunku do tradycyjnego etapu zwanego zwykle „analizą dotychczasowego stanu” i ograniczającego się do analizy *dokumentów (zamiast obiektów)*, przepływu danych, opisu

danych i zebrania algorytmów. Obiektowe ujęcie powinno umożliwić oderwanie się od statycznego spojrzenia poprzez dokumenty, raporty, struktury danych i pliki, na rzecz odzwierciedlania międzyobiektowych reakcji i relacji dynamicznych (zależnych od następstwa zdarzeń) bez uwzględnienia których nie można odwzorować rzeczywistości.

Kluczową rolę w obiektowym podejściu odgrywają pojęcia klasy i obiektu.
Klasa jest definicją właściwości grupy obiektów i służy do ich generowania. W bankowości klasą może być grupa rachunków depozytowych lub kredytowych, akredytywy itp. Generowanie polega na dziedziczeniu przez obiekt charakterystyki klasy, co nie wyklucza posiadania przez niego swoich własnych cech.

Obiekt jest wystąpieniem końcowego elementu klasy, dziedziczącym jej właściwości, posiadającym dobrze określone granice fizyczne i funkcjonalne (wartości atrybutów i funkcje), dzięki czemu możliwe jest formalne zdefiniowanie - na poziomie klasy - mechanizmów tworzenia i opuszczania obiektu (w programowaniu obiektowym zwanych konstruktorami i destruktorami).

Obiekt jest unikalnym scaleniem (enkapsulacją, hermetyzacją) struktur danych, procedur i reguł zachowania się (w tym reagowania na przesyłane do niego wiadomości), charakterystycznych dla jednostki wyodrębnionej z analizowanego rzeczywistego lub abstrakcyjnego „świata". Obiekt powinien być kompletnym zestawem w tym sensie, że zawiera w sobie mechanizmy uruchamiania wszystkich zasobów (własnych lub wywoływanych z innych obiektów) niezbędnych do działania.

Obiektowo-dynamiczne podejście w bankowości

W działalności bankowej istnieje co najmniej kilkanaście (może nawet kilkadziesiąt) biznesowych obiektów, które grupowane są w klasy. Przykładowo, klasą może być grupa rachunków oszczędnościowo-rozliczeniowych i bieżących, depozytów, kredytów, grupa płatności zagranicznych dokumentowych i akredytyw itp. W zależności od stopnia złożoności klasy możemy ją dzielić na podklasy, te na grupy itp.

Z obiektami biznesowymi stowarzyszony jest zestaw wywołań obiektów programistycznych takich jak moduły kodu wykonywalnego (procedury, funkcje), lista odwołań do tablic (np. księgowań), transakcji, fomatek ekranowych, struktur danych, notatek i korespondencji towarzyszącej.

Wśród obiektów biznesowych rozróżnić można obiekty elementarne, złożone, dynamiczne i informacyjne.
W grupie *obiektów dynamicznych* występują *zdarzenia, transakcje* i *procesy. Z*darzenie jest to elementarny niepodzielny fakt, czyn, działanie lub zaniechanie spodziewanego działania, który wpływa na stan obiektu, np. niedokonanie spłaty raty kredytu w terminie zmienia status kredytu. Transakcja jest to sekwencja zdarzeń mająca na celu realizację krótkoterminowego celu, np. otwarcie lokaty i dokonanie wpłaty. Akcja jest to złożone działanie np. uzgodnienia transakcji, ocena kategorii kredytowej i tworzenie rezerw na trudne kredyty. Proces jest to ciąg akcji i zdarzeń posiadający wspólne zadanie inicjowany przez zdarzenie inicjujące i kończony przez zdarzenie końcowe np. proces obsługi rachunku na koniec dnia obejmujący naliczenie odsetek, pobranie opłat, kapitalizację odsetek itd.

W grupie *obiektów informacyjnych* wyróżnia się takie zestawy informacji jak pozycja klienta, przepływ pieniężny, bilans księgowy. W grupie biznesowych *obiektów elementarnych* występują : podmioty (klient, bank, oddział banku), rachunki klientów, konta księgowe, waluty, limity itp.

W grupie *obiektów złożonych* występują produkty bankowe oraz linie obsługi klientów.
Wśród *produktów bankowych* występują operacje zagraniczne forex (spot, forward, swap), operacje komercyjne na rynku pieniężnym (np. międzybankowym), depozyty, kredyty, akredytywy, papiery wartościowe itp. Produkty w ujęciu obiektowym mają zwykle złożoną kilkupoziomową strukturę dziedziczenia (klasy, podklasy, grupy itp.).
Linie obsługi klienta to formy obsługi np. tradycyjna obsługa okienkowa albo bankowość elektroniczna („call center"). Obiekty tego typu zawierać powinny szczegółowy opis inftrastruktury technicznej i software'owej.

Zarys propozycji obiektowo zorientowanego języka specyfikacyjnego OOSL (Object Oriented Specification Language)

Zwroty specyfikacyjne OOSL wyrażane są w języku angielskim, gdyż sądzę, iż jakakolwiek myśl twórcza w informatyce aby być dostrzeżoną musi być w tym języku sformułowana.

Zarys języka OOSL jest jedynie ilustracją zagadnienia (a czasem jedynie wykazem problemów do rozwiązania), nie zaś zakończonym produktem. Zamieszczam go, ponieważ może zainspirować osoby pracujące nad językami specyfikacyjnymi, podczas gdy ja „dotykając" problemu jedynie hobbystycznie (od czasu do czasu) nie będę w stanie rozpracować do końca pomysłu, którego realizację rozpocząłem dawno temu[1]*.*

Spotkać się można z różnymi technikami obiektowego podejścia. Większość z nich tkwi w sferze oprogramowania, a więc w metodach, konstruktorach, destruktorach, okienkach traktowanych jako obiekty itp.), a tylko niektóre za najistotniejszą sprawę uważają odwzorowanie obiektu jako elementu biznesowego istniejącego w świecie rzeczywistym. *Język OOSL aczkolwiek posiada punkt ciężkości po stronie biznesu ma być próbą zapewnienia możliwości przechodzenia ze specyfikacji biznesowej na specyfikację programistyczną.*

Zwroty języka OOSL zapisywane są w następującej notacji:
//komentarz//
def //początek definicji obiektu//
endef //koniec definicji obiektu//
spec //początek specyfikacji//
endsp //koniec specyfikacji//
::= //przypisanie typu (np. klasy, obiektu) //
:= //przypisanie do listy//
= //przypisanie wartości //
: //przypisanie obiektowi atrybutu//
= = //ekwiwalentność//
{.....} //ograniczniki frazy specyfikacyjnej//
(x,y,..) //lista//
YYYYYY.UUUUU(XXXXXX) //kwalifikowana nazwa obiektu biznesowego//
(XXXX)//obiekt biznesowy//
[name] //obiekt wykonawczo-operacyjny//
keywords //kluczowe słowa specyfikacyjne//

Zakłada się, że poszczególne generacje opisu obiektów będą utrzymywane w bibliotece specyfikacji i tworzone automatyczne w wyniku konsolidacji zmian

Język OOSL składa się ze zwrotów standardowych, niezależnych od rodzaju biznesu, służących do definiowania obiektów oraz słów kluczowych specyfikowanych do używania w lokalnym obszarze biznesowym (AREA) lub globalnie w ramach tzw. świata (UNIVERSE). W obszarze można ponadto wyróżnić moduły biznesowe (MODULES).

Opis dokonywany jest w imieniu głównego podmiotu (SUBJECT) jakim jest instytucja realizująca działalność biznesową. UNIVERSE, AREA i SUBJECT są więc jakby superklasami. Klasy biznesowe w ramach AREA mogą być definiowane swobodnie (user-defined) zgodnie z potrzebami głównego podmiotu. Tak więc struktura obiektowa wygląda następująco: SUBJECT, UNIVERSE, AREA, <user-defined class>.

Obiekty definiowane można podzielić na elementarne, dynamiczne i wirtualne.
ObjectTypes : (eOBJECT//obiekt elementarny np. klient//,

[1] OOSL korzeniami swoimi sięga do języka specyfikacyjnego S&DL ukierunkowanego na projektowanie strukturalne. Był on również mojego autorstwa i opublikowano go w niemieckim miesięczniku Angewandte Informatik (Applied Informatics) 12/81 oraz w polskim Informatyka 2-3/82.

dOBJECT// obiekt dynamiczny np. transakcja, wystepujący jako podobiekt w obiekcie elementarnym//,
iOBJECT//obiekt informacyjny np. przepływ pieniężny//,
vOBJECT//obiekt wirtualny//).
W obiekcie dynamicznym można wyróżnić obiekty mogące być przedmiotem odrębnego opisu.
dOBJECT ::= ZDARZENIE,TRANSAKCJA,AKCJA,PROCES) ==
(EVENT,TRANSACTION,ACTION,PROCESS) //przykład równoważnego definiowania dwujęzycznego//

W OOSL wyróżnia się również *obiekty wykonawczo-operacyjne*, którymi są pracownicy podmiotu obsługujący klasy biznesowe (np. menadżer konta, opiekun klienta, kasjer, dysponent itp.) oraz infrastruktura techniczna (w tym komputery).

Środowisko w jakim działa aplikacja opisywane jest następująco:
ENVIRONMENT:=(<Ustawodawstwo>)(<Komputery>, <SystemyOperacyjne>, <BazyDanych>, <HurtownieDanych>, <ServeryDanych>, <SystemZarządzaniaSiecią>)

Specyfikacja obiektu obejmuje takie elementy jak:

-środowisko lokalne (w tym tablice danych, procedury, formaty ekranów, parametry wywołań)
-rola obiektu (ROLE)
-historia życia (OLH -Object Life History)
-relacje (RELATIONS)
-charakterystyka procesów, transakcji, zdarzeń
-tryb wykonania (EXECUTION_MODE:= RT //real time//, BT//batch//, OD //on demand//)
-przepływ danych (DATA_FLOW)
-przepływ sterowania (CONTROL_FLOW)

ROLE :=(commander //kierujący procesem, odpowiedzialny//,
trigger //inicjujący/uruchamiający proces, zdarzenie//,
generator //generujący np. zdarzenia//,
agent //reprezentuje usługę np. call-center agent//,
integrator, monitor //śledzi wykonanie/przebieg procesu//, executor,component,
performance center //miejsce wykonania//)

RELATIONS:=(*(activated by / activates, activated when,*
appearence depends on // np.pojawienie się dziecka zależy od istnienia rodziców//,
assisted by, belongs to /is owned by , *built from* ,
calls <obiekt> (xxx,yyy) //xxx elementy przekazywane., yyy elementy zwracane//,
consists of <parts> , *contained in/contains, controlled by / controls,derived from,*
driven by transaction,driven by product ,driven by banking regulations,
driven by customer, driven by date, driven by schedule,driven by frequency,
existence depends on, exists when/in/for, evaluated as critical/most wanted ,
included in , l*inked to ...by / links, matched/matches // np. uzgodnić transakcje//,*
refers to //bezpośrednie odniesienie//, relates to ,represented by / represents ,
involved in, shared by / shares, used by / uses)

STATE := (active,inactive,dormant,suspended,aborted,idle)
STATUS := (generic, real, virtual, temporary)

{CONTROL_FLOW
repetition :=(iteration, spiral //dla procesów uczących się, każdy zwój spirali posiada nową jakość)
activated .... BY ...with <initial-value> AT <time-point > WHEN ..
finished AT < > with <value> WHEN ...}

{BODY_DESCRIPTION

//ciało obiektu fizycznego lub programistycznego opisywane za pomocą pseudokodu[2] czyli niezależnego -sprzętowo - programistycznie - języka tak analitycznego że pozwala odwzorować zarówno strukturę fizyczną jak i działania obiektu //
<opis w pseudokodzie>}

Przykład specyfikacji obiektowej w języku OOSL w zakresie bankowości

Ze względu na ograniczenia objętościowe specyfikacja poniższa nie zawiera wszystkich możliwych zwrotów.

spec(Banking9712311210)

def_SUBJECT(MÓJ_BANK)
//MÓJ_BANK jest bankiem macierzystym czyli podmiotem, którego definiujemy działalność biznesową //
Parameters/DaneWejsciowe: =(idBanku, TypBiznesu //bank uniwersalny, komercyjny, detaliczny//,
kraj, WalutaBazowa, DługośćRokuFinansowego, LiczbaOddziałów, PozycjaRankingowa,
LimityWalutowe)
Tables:= (<TablicaBankówKorespondentów> , <TablicaOddziałów>, <KalendarzDniRoboczych>)
<PlanKont>)
endef

def_UNIVERSE (INTERNATIONAL_BANKING)
//definiowanie globalnego biznesu międzynarodowego, w którym MÓJ_BANK uczestniczy//
Parameters:= idBankuBIC //międzynarodowy identyfikator banku macierzystego//
keywords:= (IBAN//International Bank Account Number//, ICC //International Chamber of Commerce//, BIC//Bank Identification Code//}
//słowa kluczowe wymienione w klasie UNIVERSE są dostępne globalnie dla wszystkich obiektów //
tables:= (<międzybankowe stopy procentowe na rynku pieniężnym>//typu LIBOR//,
<międzynarodowe kursy wymiany walut>//monitoring REUTERS 2000//,
<międzynarodowe standardy finansowe>//np. UCP 500, ICC 500, URR525 dla akredytyw,
URC dla inkasa eksportu i importu, itp.//.
<wykaz międzynarodowych papierów wartościowych i instrumentów finansowych> ,
<wykaz walut wg standardu ISO>,
<wykaz sieci finansowych>,
<wykaz banków o charakterze międzynarodowym>,
<wykaz giełd i depozytoriów papierów wartościowych>,
<wykaz komunikatów SWIFTowych> itp.)
endef

def AREA(Banking)
BUSINESS_MODULES::=(DEPOZYTY, KREDYTY, INFO_O_KLIENTACH,
RYNEK_PIENIĘŻNY, AKREDYTYWA, PŁATNOŚCI, PAPIERY_WARTOSCIOWE)
keywords:=(NrOddz, idKlienta, NrRachunku, StopaOds, KursWym, DataKs, DataEfekt, SaldoDostępne, SaldoKsięgowe, kwota, kapitał, OdsetkiNal, OdsetkiSkapit, DataWygas, LimitDebetu, SaldoDebet)
procedures:= (NaliczOds, ...)
StopaOds *refers to* <opis danej w repozytorium danych>
NaliczOds *refers to* <nazwa procedury w bibliotece obiektów programistycznych>
.....
OperationalObjects ::= [dysponent, kasjer, MenadżerRachunku, MenadżerKlienta,MenadżerProduktu]

{//strukturalny podział obiektów//
CLASS ::= (PODMIOT, PRODUKT, WALUTA, LIMIT, KONTOKS, TRANSAKCJA, PROCES,
ZDARZENIE)
//Takie obiekty jak WALUTA, LIMIT, KONTOKS, TRANSAKCJA, PROCES, ZDARZENIE wchodzą do opisu większości obiektów bankowych np.PRODUKT i również same są przedmiotem odrębnych definicji obiektowych//
PRODUKT ::= (RACHUNEK, PAPIER_WARTOŚCIOWY, DERYWAT,AKREDYTYWA)
RACHUNEK ::= (BOR, DEPOZYT, KREDYT)

[2] Definicja pseudokodu nie wchodzi do standardu OOSL.

RACHUNEK.BOR ::= (ROR, A'VISTA, OSZCZĘDNOŚCIOWY, BIEŻĄCY)
LIMIT ::= (KRAJU, BRANŻY, KLIENTA, WALUTY)}

{//wyróżnienie typów obiektów//
eOBJECT ::= (KLIENT,BANK,RACHUNEK,WALUTA, PAPIERwart)
iOBJECT::= (POZKLIENTA)//pozycja klienta//
vOBJECT:: (NOSTRO_ACCOUNTS)//lustrzane odbicie rachunków w bankach zagranicznych//}
{//Atrybutowa klasyfikacja obiektów//
PODMIOT : (prPODMIOT//OsobyPrawne//, fzPODMIOT//OsobyFizyczne//)
BANK: (krBANK //krajowy// , zagrBANK //zagraniczny//, korBANK//korespondent//)
KONTOKS: (bilKONTOKS //KontoBilansowe//, pozKONTOKS //KontoPozabilansowe//
TRANSAKCJA: (rTRANSAKCJA//transakcja w czasie rzeczywistym//,eodTRANSAKCJA//transakcja
generowana podczas zamykania dnia//)}
ZDARZENIE: (zdi//inicjujące//, zdk// końcowe//, zdz//zawieszające//, zdu//usuwające//)}

//poniżej przykład definiowania rachunku oszczędnościowo-rozliczeniowego ROR, który dziedziczy parametry, procedury, transakcje, formatki ekranów i tablice danych wyspecyfikowane kolejno w klasach PRODUKT, RACHUNEK i BOR. Rachunek fizyczny (założony dla konkretnego klienta) dziedziczy ponadto specyfikację typu produktu ROR//

def PRODUKT
Parameters:= (rodzPodmiotu, TypProduktu, waluta)
//niektóre produkty mogą być aktywne tylko dla wybranych walut//
relates to <TabliceSystemoweZakresuProduktów>
// klasa PRODUKT zawiera głównie tablice systemowe wchodzące w skład rdzenia systemu bankowego//
Screens:=(Scr00)
Tables := (<TabliceWalut>, <TabliceProduktów>)
endef

def RACHUNEK
Parameters:=(Właściciel,Wspólnicy, Pełnomocnicy,Waluta,MinSaldo)
Relates to idKlienta
rTRANSAKCJE *:*= (OtwRku, LikwRku, Wpłata, Wypłata)
eodTRANSAKCJE*:*= (drWyciągu)
Screens:= (Scr01-02)
endef

def RACHUNEK.BOR //grupa rachunków BOR-bieżący, oszczędnościowy, rozliczeniowy//
Parameters:= typRachunku
Screens:= (Scr03)
.....
endef

def RACHUNEK.BOR(ROR) //rachubek oszczędnościowo-rozliczeniowy//
id::= (NrRachunku)
exists for fzPODMIOT //można go założyć tylko dla osób fizycznych//
initiated by PierwszaWpłata
Parameters:=(LimitDebetu, ZleceniaStałe, TypWyciągu, KartaVisa,Opłaty, WarunkiOdnowienia.)
Procedures:=(GenerKsięgowań, NaliczOds, OdsKarne)
rTRANSAKCJE:=(ZlecPrzelewuPoborów, Przelewy, TrBankomatowe, ...)
eodTRANSAKCJE*:= (*StałeZlecenia, PobrOpłat, OdsKarne if SaldoDebet)
Screens:= (Scr04-12)
Tables:= (Opłaty, IndeksStóp, Księgowania)
Calls NaliczOds=(30,360,90,Zmienne, IndeksStóp)
Calls zdiArch(30c,eod) //zdarzenie inicjące archiwizację transakcji, zawartych przed 30 dniami kalendarzowymi , wykonywany na koniec dnia//
Calls zduArch(5y,eoy) //zdarzenie usuwające z archiwum transakcje starsze od 5 lat wykonywane podczas zamykania roku //)
endef

endefAREA
endspec