Zygmunt Ryznar

Propozycja języka obiektowej specyfikacji OSL (na przykładzie bankowości)

Język specyfikacyjny OSL jest próbą sprawdzenia na ile złożone biznesy i obiekty mogą być poddane takiej formalizacji, by możliwa stała się analiza porównawcza wielu rozwiązań biznesowych, opisanych w tej samej notacji.
Ponadto sądzę, że OSL może być dobrze osadzony w środowisku komputerowego wspomagania poprzez użycie specjalizowanego edytora (zawierającego narzędzia sprawdzania poprawności formalnej opisu oraz podpowiedzi słów kluczowych i fraz). Generacje opisu poszczególnych obiektów byłyby utrzymywane w bibliotece specyfikacji, zaś specyfikacja całości tworzona będzie automatycznie w wyniku konsolidacji zmian.
W dalszej kolejności w grę wchodzi generator tablic i graficznej prezentacji.
Jeśli zainspiruję kogolwiek do dalszych prac rozwojowych i implementacyjnych (co oczywiście nie będzie wymagać mojej zgody lecz jedynie powołania się na źródło) będzie to dla mnie wystarczającą satysfakcją. Sądzę, że w fazie implementacji software'owej ciekawa może być próba wykorzystania możliwości języka UML (Unified Modeling Language).

Prezentuję zarys obiektowo zorientowanego języka specyfikacyjnego OSL (Object Specification Language), służącego do obiektowego opisu różnorodnych obiektów. Rdzeń języka nosi charakter uniwersalny i po pewnych uzupełnieniach może służyć do opisu obiektów biznesowych (co staram się udowodnić na przykładzie bankowości) oraz zupełnie innych (np. używać go można do opisu człowieka - HSL HumanBeing Specification Language).

Język OSL nie należy do języków symulacyjnych, tzn. nie mozna za jego pomocą odwzorować dynamicznego zachowania obiektu w czasie, czyli wygenerować kolejnych wartości zmiennych (np. zysku) w miarę upływu czasu, aczkolwiek mozna zdefiniować statycznie procesy i relacje. Nie wykluczam, że wzbogacenie tego języka o własciwości symulacyjne będzie możliwe i użyteczne.

Język OSL składa się ze zwrotów sformalizowanych, służących do definiowania obiektów oraz słów kluczowych specyfikowanych do używania w lokalnym obszarze biznesowym (AREA) lub globalnie w ramach tzw. świata (UNIVERSE).
Opis dokonywany jest w imieniu głównego podmiotu (SUBJECT) jakim może być np. instytucja realizująca działalność biznesową. UNIVERSE, AREA i SUBJECT są więc jakby superklasami. Klasy w ramach AREA mogą być definiowane swobodnie (user-defined} zgodnie z potrzebami głównego podmiotu. W OSL wyróżnia się również tzw. obiekty wykonawczo-operacyjne, którymi np. są pracownicy podmiotu obsługujący klasy (np. menadżer konta, opiekun klienta, kasjer, dysponent itp.) oraz infrastruktura techniczna (w tym serwery, bazy danych itp.) i organizacyjna.
Notacja pod względem formalnym jest stosunkowo prosta i nie nosi charakteru matematycznego. Ze względu na róznorodność zwrotów opanowanie języka zapewne wymagać będzie pewnego wysiłku.
Poza "standardami" starałem się wprowadzać do notacji takie oryginalne rozwiązania jak np.
geometryczna prezentacja procesu i populacji
- repetition :=(iteration, single spiral, multiband spiral)
- population layout := (free-space,swarm,network,hierachy,line,triangle,tunnel...)
Zdefiniowanie powyższych konfiguracji geometrycznych stanowi odrębne zagadnienie i nie wchodzi do niniejszego opisu.

NOTACJA JĘZYKA OSL

<! *komentarz* >
<! w nawiasach ostrych <> podawane są tylko głowne frazy strukturalne: def, spec oraz komentarze, nazwy obiektów biznesowych i predefiniowane słowa kluczowe pisane są dużymi literami, obiekty niebiznesowe pisane są małymi literami>

< def *nazwa obiektu* > <! początek definicji obiektu >

< / def> <! koniec definicji obiektu >

< spec *nazwa specyfikacji* > <! początek specyfikacji >

< / spec > <! koniec specyfikacji >

: : = <! przypisanie typu (np. klasy, obiektu) >

:= <! przypisanie do listy >

= <! przypisanie wartości czyli podstawienie >

: <! przypisanie obiektowi atrybutu >

: : <! przynależność>

= = <!ekwiwalentność >
{.....} <!ograniczniki frazy specyfikacyjnej >
(x,y,..) <!lista >
YYYYYY.UUUUU(XXXXXX) <!kwalifikowana nazwa obiektu biznesowego >
[name] <!obiekt wykonawczo-operacyjny >
(XXXX)<!obiekt biznesowy >
keywords <!kluczowe słowa specyfikacyjne, są dziedziczone przez obiekty niższego rzędu >

Środowisko (ENV) w jakim działa aplikacja opisywane jest następująco:
ENV:=(REGULATIONS, INFRASTRUCTURE)
ENV.REGULATIONS:=(*Akty prawne, regulaminy, zarządzenia*)
ENV.INFRASTRUCTURE:=(*ServeryDanych, SystemyOperacyjne, Transakcyjne BazyDanych, HurtownieDanych , SystemZarządzaniaSiecią, Struktura organizacyjna firmy*)
Specyfikacja obiektu obejmuje takie elementy jak:
-id <!identyfikator obiektu>
-infoWindow <!"okno na świat " - informacje uzewnętrzniane przez obiekt >
-dataTable <!tablica danych. >
-typ obiektu ObjectTypes : (eOBJECT<!obiekt elementarny np. klient >
-rola obiektu (ROLE)
-historia życia (OLH -Object Life History)
-relacje (RELATIONS)
-charakterystyka dynamiki(poprzez procesy, transakcje, zdarzenia; sposób ich inicjowania itp.)
-tryb wykonania (EXECUTION_MODE:= modeRT <!real time >,modeBAT<!batch > modeOND <!on demand >)
-przepływ danych (DATA_FLOW)
-przepływ sterowania (CONTROL_FLOW)
Poniżej wymienimy niektóre elementy (szczególnie te, których nie udało się użyć w przykładzie).
ROLE :=(commander <!obiekt sterujący procesem, odpowiedzialny >,
trigger <!obiekt inicjujący/uruchamiający proces, zdarzenie >,
generator <! obiekt generujący np. zdarzenia, cash flow >,
agent <! reprezentuje usługę np. agent w call-center >,
integrator <! obiekt integrujący obiekty>
monitor <! śledzi wykonanie/przebieg procesu >,
executor <! obiektwykonawczy >,
participator <!obiekt uczestniczący>,
owner, stockholder, customer, supplier; partner, employee,
component <! składnik >
performance center <! obiekt będący miejscem wykonania >)
RELATIONS:=(*(activated by / activates, activated when/if,*
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 <! np. istnienie obiektu zależy od .... >
exists when/in/for, evaluated as critical/most wanted , included in ,
linked to ...by / links, matched/matches <! np. uzgodnić transakcje >
refers to <!bezpośrednie odniesienie> *relates to, related by affinity, represented by /*
represents , involved in, shared by / shares, used by / uses)
STATE := (active,inactive,dormant,suspended,aborted,idle, lost)
STATUS := (generic, real, virtual)
dOBJECT<! obiekt dynamiczny np. transakcja >
iOBJECT<!obiekt informacyjny np. informacje o tzw. pozycji klienta >
vOBJECT < obiekt wirtualny np. lustrzane rachunki Nostro >.
dOBJECT ::= ZDARZENIE,TRANSAKCJA,AKCJA,PROCES)
== (EVENT,TRANSACTION,ACTION,PROCESS) <! przykład równoważnego definiowania dwujęzycznego >
<! ZDARZENIE jest to elementarny niepodzielny fakt, czyn, działanie lub zaniechanie spodziewanego działania 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.) >
{CONTROL_FLOW
repetition :=(iteration, single spiral, multiband spiral <!spirala znajduje zastosowanie w procesie uczącym się i od iteracji różni się

tym, iż każdy jej zwój posiada inne mechanizmy i treść >)
activated .... BY ...with <initial-value> AT <time-point > WHEN ..
finished AT < > with <value> WHEN ...}
{ DATA FLOW
<.> from (<.>) <!lista danych przychodzących od...>
<.> to (<.>) <!lista danych wysyłanych do >
<> dataGeneration Method <*nazwa* >
<>dataEncoding Method <*nazwa* >
<>dataCompressing Method <*nazwa* >
}
{BODY ::= (Contents, Script, Metadata)
contents <!zawartość ciała np. treść dokumentu, kod programu >
script <!lista operacji generowanych wewnątrz obiektu odpowiednio do jego zachowania się np. oczekiwanie na dostęp do danych, zawieszenie się >
population layout := (free-space, swarm, network, hierachy, line, triangle, tunnel...)

metadata <! np. w XML- opis struktury ciała obiektu >}

Przykład specyfikacji obiektowej w języku OSL w zakresie bankowości

Zamieszczone poniżej zwroty dotyczą fragmentów biznesu bankowego i stanowią jedynie ilustrację metody.

<OSL-1.Beta>
<spec(Banking)>
<def SUBJECT(MÓJ_BANK)>
<!MÓJ_BANK podmiot, którego działalność biznesową definiujemy >

infoWindow: =(idBanku, TypBiznesu <!bank uniwersalny, komercyjny, detaliczny>,
kraj, WalutaBazowa, DługośćRokuFinansowego, LiczbaOddziałów, PozycjaRankingowa, LimityWalutowe)
dataTables: = (TablicaBankówKorespondentów , TablicaOddziałów, KalendarzDniRoboczych, PlanKont)

</def>
<def UNIVERSE (INTERNATIONAL_BANKING)>

<!definiowanie globalnego biznesu międzynarodowego, w którym MÓJ_BANK uczestniczy >
<!słowa kluczowe wymienione w klasie UNIVERSE są dostępne globalnie dla wszystkich obiektów specyfikacji >

id :=BIC<!międzynarodowy identyfikator banku macierzystego >
keywords:= (IBAN<!International Bank Account Number >,
ICC <!International Chamber of Commerce >,
BIC<!Bank Identification Code >
dataTables:= (*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 >, *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.*)

</def>
<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)
dataTables := (*TabliceWalut, TablicеProduktów*)
procedures:= (NaliczOds, ...)
StopaOds *refers to* <*identyfikator 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)
RACHUNEK.BOR ::= (ROR, A'VISTA, OSZCZĘDNOŚCIOWY, BIEŻĄCY)
LIMIT ::= (KRAJU, BRANŻY, KLIENTA, WALUTY)}
{<!wyróżnienie typów obiektów >
eOBJECT : : = (KLIENT,BANK,RACHUNEK,WALUTA, PAPIERwart) <!obiekty elementarne>
iOBJECT::= (POZKLIENTA, WYCIĄG-MIESIĘCZNY) <!obiekty informacyjne>
}
{ <!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,BOR,ROR. Rachunek fizyczny (założony dla konkretnego klienta) dziedziczy specyfikację powyższych klas >
<def PRODUKT>

infoWindow: = (rodzPodmiotu, TypProduktu, waluta)
<!niektóre produkty mogą być aktywne tylko dla wybranych walut>
relates to <TabliceSystemoweZakresuProduktów>
<! klasa PRODUKT zawiera informacje dziedziczone przez typy produktów >

</def>
<def RACHUNEK >

infoWindow :=(Właściciel,Wspólnicy, Pełnomocnicy,MinSaldo,Saldo,Obroty,HistoriaRachunku)
Relates to idKlienta
rTRANSAKCJE := (OtwRku, LikwRku, Wpłata, Wypłata)
eodTRANSAKCJE: = (drukWyciągu)

</def>
<def RACHUNEK.BOR >
<! grupa rachunków BOR-bieżący, a'vista, oszczędnościowy, rozliczeniowy >
Id: = typRachunku <! ROR - oszczędnościowo-rozliczeniowy, BIEZ - bieżący>

</def>
<def RACHUNEK.BOR(ROR)>

id: : = (NrRachunku)
exists for fzPODMIOT <!tylko dla osób fizycznych >
initiated by PierwszaWpłata

ROR: :*infoWindow* :=(LimitDebetu, ZleceniaStałe, TypWyciągu, KartaVisa,Opłaty, WarunkiOdnowienia.)
Procedures:=(GenerKsięgowań, NaliczOds, OdsKarne)
rTRANSAKCJE: =(ZlecPrzelewuPoborów, Przelewy, TrBankomatowe, ...)
eodTRANSAKCJE: = (StałeZlecenia, PobrOpłat, OdsKarne if SaldoDebet)
dataTables:= (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 >)

</def>
</def>AREA>
</spec>