DSM nástroj UML .FRI

Ján Janech a Tomáš Bača

Katedra softvérových technológií, FRI, Žilinská univerzita v Žiline,
Univerzitná 1, 010 26, Žilina
jan.janech@kst.uniza.sk
tomas.baca@kst.uniza.sk

Abstrakt V prvej časti článku sú popísané všeobecné princípy DSM (*Domain Specific Modeling*) – čo to je a ako umožňuje efektívnejšie navrhovať softvérové systémy. Vysvetľujú sa tu základné pojmy metamodelovania a dve roviny abstrakcie. V druhej časti je popísaný UML .FRI, ako zástupca DSM nástrojov, ktorý je navyše vyvíjaný samotnými autormi článku. Jeho zvláštnosťou voči iným DSM nástrojom je jeho cieľová skupina – vyučovací proces na univerzite. V tomto duchu je ladená aj druhá časť článku.

Kľúčové slová: DSM, výučba, UML .FRI

1 Úvod

Ak chceme niekoho uviesť do novej problematiky, obyčajne ho najskôr zoznámime so zjednodušeným pohľadom na vec. Niektoré menej podstatné informácie na začiatku vynecháme. Rovnako, ak opisujeme väčší systém kolegovi, nevenujeme sa detailom, ktoré nie sú v danom čase podstatné. Tento postup je bežný vo všetkých vedných oboroch, ba dokonca, dalo by sa povedať, že v každodennej praxi. Zjednodušený obraz reality nazývame modelom a vytvárame ho modelovaním.

Keď začneme popisovať objekt reálneho sveta, obyčajne si zvolíme jeden konkrétny uhol pohľadu. Informácie, ktoré sme na začiatku v modeli neuviedli, pretože neboli zaujímavé, budeme však pravdepodobne niekedy inokedy potrebovať. Rozširovaním pôvodného modelu by sme nakoniec dospeli k znovuvytvoreniu pôvodného systému. Hlavne by sme však stratili jednoduchosť vyjadrenia, pre ktorú model vytvárame, pretože v modeli zostávajú aj informácie, ktoré sú už v novej situácii nezaujímavé, alebo notoricky známe. Preto je rozumnejšie vytvoriť nový model, ktorý bude vyhovovať novej situácii.

Model môže nadobúdať rôzne formy. Modelom môže byť slovný popis, nákres, miniatúra, ale tiež sústava matematických rovníc. V tomto článku sa budeme venovať vizuálnemu modelovaniu, to znamená, tvorbe nákresov a diagramov, popisujúcich objekty reálneho sveta.

Pre vizuálne modelovanie a výmenu modelov s inými ľuďmi sa v súčasnosti výrazne uplatňujú rôzne počítačové programy. Väčšina špecializovaných programov v sebe obsahuje nejaký konkrétny modelovací jazyk, vymyslený tak, aby sa

ním dobre modelovala určitá konkrétna skupina objektov reálneho sveta, resp. pri pozeraní sa na svet z konkrétneho uhlu pohľadu.

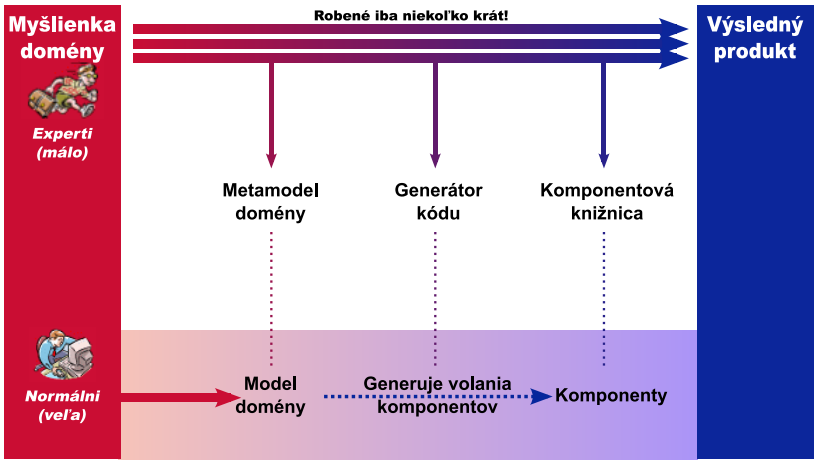
Na jednej strane je používanie štandardného modelovacieho jazyka výhodou, pretože ho pozná pomerne veľká skupina ľudí. Na strane druhej, takýto modelovací jazyk býva obyčajne navrhnutý tak, aby pokrýval pomerne veľkú skupinu problémov. Takýto jazyk sa často označuje ako GPML (*General Purpose Modeling Language*).

Pri použití GPML sa môže stať, že pri riešení jedného problému z jazyka často nevyužijeme značnú časť. Taktiež sa stáva, že si ho zvolíme aj pre špecifický problém, ktorý nie je jazykom dostatočne pokrytý, alebo naň nemá dostatočne jednoduché vyjadrovacie prostriedky. V takej situácii potom treba prispôsobovať model modelovaciemu jazyku, hoci by bolo logickejšie a lepšie prispôsobiť modelovací jazyk modelovanému problému.

Prístup, keď si najskôr navrhneme modelovací jazyk, špecificky k nášmu problému a potom v tomto jazyku modelujeme náš problém, prípadne niekoľko veľmi podobných problémov sa označuje ako DSM (*Domain Specific Modeling*) – modelovanie v jazyku špecifickom pre danú doménu.

2 Všeobecné princípy DSM

Modelovanie, v jazyku špecifickom danej doméne, sa skladá z dvoch fáz. Prvá fáza spočíva vo vytvorení modelovacieho jazyka. V druhej potom nasleduje samotné modelovanie. Tento postup je znázornený na Obr. 1.



Obr. 1. Postup pri tvorbe produktu s použitím nástrojov DSM [3]

DSM fórum [3] kladie prvú fázu za úlohu doménovému expertovi. Jeho hlavnými úlohami sú vytvorenie modelovacieho jazyka, šablóny pre generovanie objektov z modelov a prípadne aj knižnice komponentov.

Vytvorenie modelovacieho jazyka spočíva vo vymenovaní stavebných prvkov a v definovaní gramatiky. Vymenovaným stavebným prvkom treba určiť ich význam. Gramatika zase vytvára zoznam pravidiel, ktoré definujú, ako sa jednotlivé stavebné prvky môžu spájať do väčších celkov. V prípade vizuálneho modelovacieho je potrebné určiť aj notáciu. Notácia stavebných prvkov vizuálneho jazyka určuje, ako budú vyzerať (tvar, celistvosť čiar a pod.).

Súčasťou definície jazyka býva aj generátor kódu, alebo šablóna pre generátor. Slúžia na transformáciu modelu do objektov reálneho sveta. Môže to byť napríklad funkcia, ktorá na základe diagramov vygeneruje zdrojové kódy, skript alebo niečo podobné. Nepovinne býva mapovaná na knižnicu komponentov. V prípade, že je knižnica komponentov k dispozícii, generovanie objektov spočíva predovšetkým v mapovaní modelu na volania komponentov.

Doménový expert je človek, ktorý veľmi dobre pozná kategóriu problémov, ktoré budú modelované. To znamená, že vie, ktoré charakteristické črty objektov reálneho sveta v modeli nesmú chýbať. Zároveň však vie, čo pre model nie je zaujímavé a preto ani modelovaciemu jazyku neumožní takéto informácie vyjadriť. Väčšinou za experta považujeme človeka, ktorý v danej oblasti už vyvinul nejaké aplikácie, vytvoril architektúru vznikajúceho produktu na pozadí, alebo vytvoril knižnicu komponentov, ktorá sa využije pri generovaní kódu.[3]

3 Abstrakcia, metamodelovanie

Spoločné označenie pre definíciu modelovacieho jazyka nazývame metamodelom. Je vytvorený pomocou metamodelovacieho jazyka procesom nazývaným metamodelovanie. Dá sa povedať, že metamodel je abstrakciou modelu a teda, že sú v podobnom vzťahu, ako model voči modelovanému objektu reálneho sveta.

Na tomto mieste treba však upozorniť, že abstrakcia môže prebiehať dvomi samostatnými smermi, ktoré v bežnej reči môžu niekedy splývať. Dôvod, prečo môže dochádzať k zámene, je ten, že v oboch prípadoch sa na existenciu vzťahu používa slovo „JE"[1]. Jeden spôsob abstrakcie je generalizácia. Druhý spôsob je vzťah triedy a inštancie. [2]

Generalizácia je vzťah dvoch generických objektov, tried, množín a pod. Ako príklad môžu poslúžiť množiny čísel – prirodzené čísla, celé čísla, reálne čísla. Pre každý prvok z množiny prirodzených čísel platí, že je zároveň aj prvkom množiny celých a reálnych čísel. Môžeme napísať napríklad: $N \subset Z \wedge 1 \in N \Rightarrow 1 \in Z$. Čiže, prirodzené číslo JE aj celým číslom.

Vzťah triedy a inštancie vyjadruje, že inštancia je vo všeobecnej rovine popísateľná svojou triedou [2]. Ak využijeme predošlý príklad, tak 1 JE prirodzené číslo. To znamená, že číslo 1 je inštanciou triedy prirodzených čísel. Aby sme pokračovali tým istým smerom abstrakcie, treba sa posunúť na vyššiu úroveň, na

[1] V anglickej literatúre sa tento vzťah označuje ako „IS-A"

ktorej je množina prirodzených čísel inštanciou. Jej triedu možno nazvať číselný typ, rovnako ako JE typom čísla aj množina celých a reálnych čísel.

Vrátime sa späť ku vzťahu objektov a modelu. Abstrakcia medzi nimi je teda vzťah inštancie a triedy. Rovnako, model je inštanciou metamodelu. Číselný typ, ktorého inštanciou je napríklad množina prirodzených čísel teda môžeme označiť ako metatriedu.

Teraz sa opäť vráťme na začiatok kapitoly, kde bolo uvedené, že modelovací jazyk je metamodelom zapísaným v metamodelovacom jazyku. Pre úplnosť informácie treba ešte uviesť, že metamodel je inštanciou metametamodelu. Metametamodel už potom definuje sám seba – vyššie úrovne nemajú vlastné označenie.

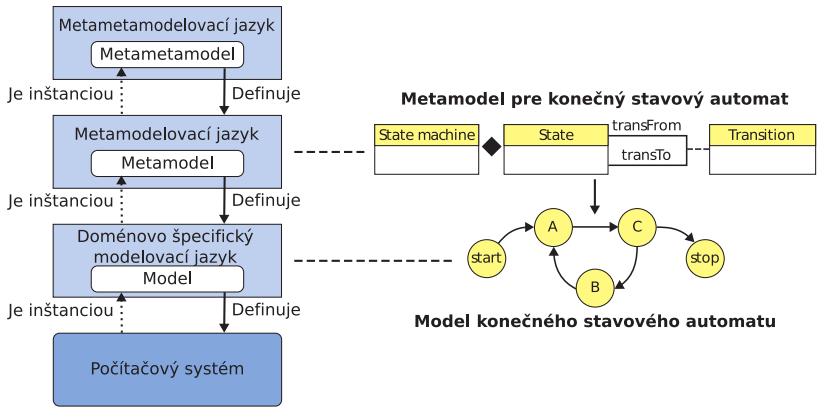
Obr. 2 znázorňuje úrovne abstrakcie, popísané vyššie v tejto kapitole. DSM nástroje implementujú každú z týchto úrovní samostatným subsystémom.

- Metametamodel býva štandardne integrovaný do samotného DSM nástroja. Je to tá časť aplikácie, ktorá určuje sémantiku metamodelu a taktiež časť, ktorá dokáže metamodel prečítať a interpretovať.
- Metamodel musí byť od samotnej aplikácie DSM nástroja oddelený. Aplikácia ho pred použitím spracuje, aby na jeho základe ponúkla používateľovi modelovanie v zadefinovanom modelovacom jazyku. Vytvára a upravuje ho používateľ v roli doménového experta.
- Model je to, čo vytvára a upravuje bežný používateľ nástroja. Aplikácia obsahuje ho dokáže vykresliť, uložiť do projektového súboru a opätovne ho z neho načítať a pod. Na to, aby bolo možné s modelom v aplikácii pracovať, je potrebné, aby bol k dispozícii aj jeho metamodel.
- Objekty reálneho sveta vznikajú generovaním, ak to nástroj umožňuje. V takom prípade musí metamodel obsahovať šablónu, podľa ktorej sa dá generovať.

4 Využitie princípov DSM v praxi

Dôvodov, prečo sa DSM používa na mnohé veľké projekty je niekoľko. Najdôležitejší z nich je niekoľkonásobné zvýšenie produktivity. To je spôsobené zjednodušením procesu tvorby samotného výsledného (softvérového) produktu na základe vykonanej analýzy. Kým iné prístupy ku tvorbe softvéru vyžadujú niekoľko krokov od modelu vytvoreného v CASE nástroji po finálny produkt, DSM nástroje umožňujú v niektorých prípadoch až 100% generovanie zdrojových kódov aplikácie z navrhnutého modelu.

Prečo je tomu tak, je schematicky uvedené na Obr. 3. Ako je vidieť, najjednoduchší spôsob prevodu modelu na finálny produkt je pri tvorbe „softvérového modelu" v jazyku symbolických adries. Mapovanie problému priamo do tohto jazyka je však komplikované a zdĺhavé. Boli preto vytvorené vyššie programovacie jazyky, ktoré umožňujú jednoduchší zápis modelu. Konverzia z vyššieho programovacieho jazyka do jazyka symbolických adries, alebo priamo do finálneho produktu je plne automatická. Vyššie programovacie jazyky zaviedli vyššiu



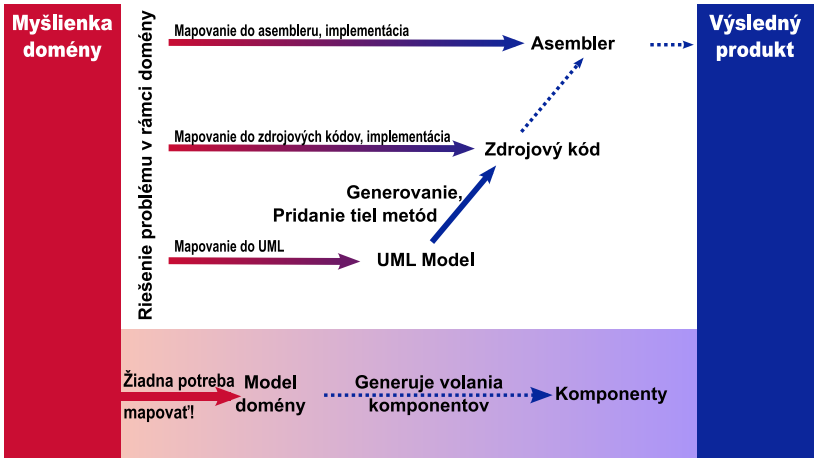
Obr. 2. Štvorvrstvová metamodelovacia architektúra. Príklad ukazuje definíciu konečného stavového automatu pomocou metamodelu v UML [7]

mieru abstrakcie a teda umožnili návrhárom softvéru sústrediť sa viac na riešenie problémov. Stále sa však jedná o riešenie na úrovni blízkej počítaču a nie samotnému problému. Na ďalšie zvýšenie abstrakcie sa používajú modelovacie jazyky, najčastejšie UML. Umožňujú analýzu problému na vysokej miere abstrakcie. Cenou za to je, že konverzia na finálny produkt už nie je úplne automatická a je nutné do tohto procesu zasahovať na úrovni zdrojového kódu. Prekážkou plnému generovaniu kódov je práve všeobecnosť týchto modelovacích jazykov. Kvôli nej nie je možné jednoducho popísať doménu tak, aby bol počítač schopný vykonať úplné generovanie zdrojových kódov.

Ako už bolo povedané, DSM nástroje umožňujú doménovému expertovi zadefinovať modelovací jazyk, ktorý presne pokrýva požiadavky dané doménou. Dôsledok tejto možnosti je tiež vidieť na Obr. 3. Pri riešení problému modelovacím jazykom špecifickým pre doménu je možné navrhnúť model bez potreby mapovania problému do modelovacieho jazyka. Ak je správne navrhnutý modelovací jazyk, model vzniká priamo ako abstrakcia problému. Rovnako, transformácie potrebné na generovanie kódu, môžu byť súčasťou definície domény. Vďaka tomu DSM nástroje umožňujú takmer kompletné generovanie zdrojových kódov produktu.

Prijatie princípov DSM zo strany veľkých firiem je veľmi dobré. Príkladom môže byť firma Nokia, ktorá sa rozhodla využiť DSM pri tvorbe softvéru pre mobilné telefóny. Výsledkom bolo až 10 násobné zvýšenie produktivity. [8].

Treba však povedať, že ani prístup DSM nemusí byť ideálny na nasadenie za každých okolností. V prípade jednoduchých projektov sa môže stať, že počiatočné náklady na zadefinovanie domény v DSM nástroji prekročia náklady na riešenie pomocou GPML nástroja. [9]



Obr. 3. Produktivita DSM [4]

5 DSM nástroj UML .FRI

Nástroj UML .FRI je vyvíjaný na Žilinskej univerzite[2] v rámci projektovej výučby. Patrí medzi DSM nástroje, ale nemá tak široké zameranie, ako ostatné nástroje z tejto kategórie. Jeho zameranie je čisto na potreby školy v rámci vyučovacieho procesu. Aj keď by sa mohlo zdať, že takéto úzke zameranie spraví nástroj jednoduchším, ako bude uvedené neskôr, prináša to so sebou rôzne komplikácie.

Nástroj vznikol z počiatku ako náhrada za drahé komerčné nástroje na modelovanie UML, na ktorých sa študenti učili využívať tento jazyk [6]. Z toho pramení aj jeho názov. Jeho architektúra však umožňovala jednoduchým spôsobom nahradiť aj iné na škole využívané nástroje. Bol preto dodatočne upravený tak, aby podporoval rôzne typy modelov. Tieto modely je možné do nástroja zavádzať v podobe jednoduchých XML definícií metamodelu. Vzhľadom na to, že si vďaka tomu môže ktokoľvek vytvoriť vlastný metamodel a zaviesť ho do nástroja, je možné nástroj označiť ako DSM.

Keďže má nástroj fungovať ako náhrada rôznych iných CASE nástrojov, doména je definovaná typom požadovaného modelu. Nástroj sa teda vo väčšine prípadov nepoužíva s vlastným navrhnutým typom modelu, ale s presne definovaným, často štandardizovaným jazykom. Z toho vyplýva, že metametamodel musí byť zadefinovaný tak, aby metamodel bol schopný popísať všetky črty daného modelovacieho jazyka. Toto neplatí pre klasické DSM nástroje. Je síce výhodné, keď nástroj nekladie doménovému expertovi žiadne prekážky pri návrhu modelovacieho jazyka, ale na druhú stranu sa nový jazyk, na rozdiel od existujúceho, dá čiastočne prispôsobiť možnostiam nástroja. Keď sa však jedná

[2] Fakulta riadenia a informatiky, Katedra softvérových technológií

o implementáciu už existujúceho jazyka, je nutné dodržiavať všetky jeho časti do čo najmenších podrobností.

Zvláštna pozornosť sa v nástroji UML .FRI venuje pri definícii notácie modelovacieho jazyka. Nasadenie pri vyučovaní vyžaduje čo najpresnejšie dodržiavanie notácie podľa štandardov daného modelovacieho jazyka, aby neboli študenti zmätení, keď začnú pracovať s konkurenčným nástrojom [5]. Rôzne existujúce modelovacie jazyky obsahujú nezanedbateľné rozdiely v notácii. Niektoré sú zadefinované ako diagram, zobrazujúci množinu elementov, prepojených spojeniami, na základe určitých pravidiel. Niektoré pridávajú definície špeciálnych vlastností diagramov (napr. plavecké dráhy v UML), alebo elementov (čiary života v UML). Iné zase definujú odlišný zápis diagramov – v tabuľke (štrukturogramy, CRC karty), vo forme grafov funkcií (časový diagram v UML)... Aby bol nástroj schopný pracovať so všetkými týmito notáciami, musí obsahovať dostatočne flexibilný systém na definíciu notácie a vykresľovanie diagramov. Tento systém je v UML .FRI implementovaný zatiaľ len čiastočne. Popísaný bude ďalej.

6 Princípy metamodelovania v UML .FRI

V UML .FRI sa v súčasnosti model skladá z diagramov. Každý diagram je v podstate graf, zložený z uzlov a hrán. Uzly sú označované ako elementy, hrany sú označované ako spojenia. Každé spojenie musí začínať aj končiť nejakým elementom, principiálne môže aj v tým istým. Na element sa môže pripájať principiálne ľubovoľný počet spojení.

Definícia metamodelu v UML .FRI sa skladá z niekoľkých základných stavebných prvkov:

- Element – Hlavnou časťou definície elementu je jeho notácia (vizuálna reprezentácia). Okrem toho, ešte obsahuje obmedzenia, akým typom spojenia sa môže spájať s inými typmi elementov.
- Spojenie – Definícia spojenia obsahuje iba notáciu spojenia.
- Doména[3] – Predstavuje definíciu štruktúrovaných údajov, ktoré sú obsiahnuté v elementoch a spojeniach.
- Diagram – Definícia diagramu spočíva vo vymenovaní typov elementov a spojení, ktoré je možné na daný typ diagramu umiestniť.

Dá sa povedať, že definícia štruktúry dát, spoločne s obmedzeniami, definovanými v elemente a diagrame, tvoria metamodel ako taký. Notácia nie je z pohľadu teórie metamodelovania dôležitá. Napriek tomu to je tá časť, ktorú si používateľ všimne ako prvú a pri vyučovanie je dôležitá (viď. predošlá kapitola). Preto jej v UML .FRI venujeme zvýšenú pozornosť.

V UML .FRI je notácia riešená pomocou systému jednoduchých kontajnerov, komplexných kontajnerov a koncových prvkov. Jednoduché kontajnery sú geometrické tvary, podmienené bloky a cyklus. Jednoduchý kontajner môže

[3] V tomto prípade má slovo doména iný význam ako v DSM

priamo obsahovať iba jeden vnorený prvok. Komplexný kontajner sa od neho odlišuje tým, že môže obsahovať ľubovoľný počet prvkov. Komplexné kontajnery radia obsiahnuté prvky pod seba alebo vedľa seba. Spoločným znakom všetkých kontajnerov je, že svoj vzhľad prispôsobujú svojmu obsahu. Koncový element už nemôže obsahovať ďalšie prvky. Do tejto skupiny patria texty, vektorové a bitmapové obrázky.

Hlavnou myšlienkou tohto konceptu je, aby bolo možné vytvoriť ľubovoľnú notáciu jednotlivých prvkov. Systém umožňuje „poskladať si" veľké množstvo rôzne zobrazených elementov. Podmienené bloky umožňujú meniť ich vzhľad na základe hodnôt atribútov. Cyklus zase umožňuje prechádzať kolekciou atribútov – napríklad zoznam variabilnej dĺžky.

Tento systém však ešte nie je v súčasnosti dokončený. Umožňuje modelovanie v modelovacích jazykoch, ktoré využívajú klasickú štruktúru grafov. Problematické sú však niektoré špecifické typy UML diagramov a okrem nich aj niektoré ďalšie. Zatiaľ však nemá prostriedky na nič podobné, ako sú čiary života v sekvenčnom diagrame. Taktiež nedokáže pripojiť spojenie na iné spojenie a teda napríklad pripojiť poznámku k spojeniu. Taktiež je implementovaný iba jednoduchý systém obmedzení, ktorý nedokáže napríklad zakázať cyklickú dedičnosť.

7 Využitie systému zásuvných modulov

Pre lepšiu predstavu v krátkosti ešte raz uvedieme funkcionalitu aplikácie UML .FRI. Aplikácia samotná dokáže spracovať metamodel, podľa ktorého potom umožňuje vytváranie a úpravu diagramov, zložených z elementov a spojení. To znamená, pridávanie a preskupovanie elementov, vytváranie spojení medzi nimi, kontrolovanie gramatických pravidiel, ukladanie projektu do súboru a jeho obnovenie, export diagramu do obrázkových formátov a tlač diagramu na tlačiarni.

Avšak, pokročilejšie funkcie nie sú do samotnej aplikácie zahrnuté a väčšina z nich ani zahrnutá nebude. Sú to predovšetkým funkcie, ktoré nejakým spôsobom spracúvajú model, upravujú ho, alebo z neho vytvárajú nejaký produkt. Hlavný dôvod, prečo zahrnutie do aplikácie nie je ani plánované ani žiadúce, je ten, že sú, takmer bez výnimky, špecifické pre jeden konkrétny metamodel. [1]

Vhodnými príkladmi pokročilých funkcií, s ohľadom na cieľovú skupinu, vyučovací proces na univerzite, sú napríklad:

- generovanie zdrojových kódov z UML diagramov,
- on-line modifikácia databázy na základe ERA diagramu,
- vizualizácia algoritmov z teórie grafov,
- procesne orientovaná simulácia,
- a iné.

Dôležitým rysom všetkých uvedených príkladov je, že ich realizácia pomocou deklaratívneho zápisu metamodelom by bola veľmi komplikovaná, ak vôbec možná. Podstatne pohodlnejší prístup by bol s využitím imperatívneho prístupu, čiže naprogramovaním samostatného modulu, ktorý by požadovanú funkcionalitu poskytoval.

Ak je však takýto modul závislý na konkrétnom metamodeli a pri inom nemá jeho používanie zmysel, je logické, aby bol v aplikácii aktívny iba v čase, keď ho používateľ potrebuje. Takéto správanie sa je charakteristické pre tzv. zásuvné moduly. UML .FRI bolo preto nedávno rozšírené o systém, ktorý umožňuje pripájanie zásuvných modulov k aplikácii.

Ak by bolo teraz UML .FRI použité ako modelovací nástroj tak, ako je znázornené na Obr. 1, tak aplikácia samotná pokrýva iba prvú časť (vľavo) tvorby výsledného produktu – spracovanie metamodelu a tvorba modelu. Druhú časť, generovanie kódu, resp. mapovanie na volania komponentov, by už obstarávali príslušné zásuvné moduly. Tretia nepovinná časť, knižnica komponentov, jej vytvorenie a použitie by bolo plne v kompetencii tvorcu zásuvného modulu.

8 Záver

Napriek tomu, že princípy DSM boli navrhnuté na zvýšenie produktivity oproti klasickým GPLM nástrojom, ich uplatnenie sa dá nájsť aj pri vyučovaní na vysokej škole. Dôkazom toho je nástroj UML .FRI.

UML .FRI je vyvíjaný s priamo cieľom začlenenia do vyučovacieho procesu. Má nahradiť niekoľko rôznych nástrojov (modelovanie UML, E-R, vývojové diagramy, DFD. . .). S ohľadom na to je volený zápis metamodelu. Maximum pozornosti sa venuje zápisu notácie, aby sa študenti dokázali jednoducho zorientovať po prechode na iný nástroj.

Nástroj UML .FRI už bol úspešne nasadený na predmety, ktoré vyžadovali jednoduchší tvar modelov. Diagram tried z modelovacieho jazyka UML sa využil na predmete Objektové programovanie. Okrem toho sa na predmete Základy informatiky 1 využila podpora vývojových diagramov. V dobe písania článku bol v príprave nový predmet Informatika 1 začínajúci v tomto akademickom roku, na ktorom sa bude využívať čiastočne aj nástroj UML .FRI ako podpora pre metodiku Objects First.

Literatúra

1. Bača, Tomáš: Zásuvné moduly v nástroji UML .FRI. In: *Otvorený softvér vo vzdelávaní, výskume a v IT riešeniach.* Žilina : Vedecko-technická spoločnosť pri Žilinskej univerzite, Júl 2009. ISBN 978-80-89276-16-5.
2. Brachman, Ronald J.: *What IS-A Is and Isn't: An Analysis of Taxonomic Links in Semantic Networks.* In: *IEEE Computer.* 1983. vol.16, n°10.
3. *DSM Forum: How to get started with DSM.* [online]. Dostupné online: <http://www.dsmforum.org/how.html>.
4. *DSM Forum: Why DSM?.* [online]. Dostupné online: <http://www.dsmforum.org/why.html>.
5. Janech, Ján: UML/DSM nástroj UML .FRI. In: *Otvorený softvér vo vzdelávaní, výskume a v IT riešeniach.* Žilina : Vedecko-technická spoločnosť pri Žilinskej univerzite, Júl 2009. ISBN 978-80-89276-16-5.
6. Janech, Ján – Sadloň, Ľubomír: CASE UML .FRI. In: *OBJEKTY 2007.* Ostrava : Fakulta elektroniky a informatiky, VŠB – Technická univerzita Ostrava, November 2007. ISBN 978-80-248-1635-7.

7. Lédeczi, Ákos – Bakay, Árpád – Maróti, Miklós a i.: *Composing Domain-Specific Design Environments.* In: *Computer.* Los Alamitos, CA, USA : IEEE Computer Society Press, 2001. vol.34, n°11. ISSN 0018-9162.
8. *MetaCase: Nokia Case Study.* [s.l.]. MetaCase. 2007. Dostupné online: <http://www.metacase.com/papers/MetaEdit_in_Nokia.pdf>.
9. Sprinkle, Jonathan – Mernik, Marjan – Tolvanen, Juha-Pekka a i.: *Guest Editors' Introduction: What Kinds of Nails Need a Domain-Specific Hammer?.* In: *IEEE Software.* Los Alamitos, CA, USA : IEEE Computer Society, 2009. vol.26, n°4. ISSN 0740-7459.