Otvorený softvér vo vzdelávaní, výskume a v IT riešeniach
Žilina 2.-5. júla 2009

ZÁSUVNÉ MODULY V NÁSTROJI UML .FRI

BAČA, Tomáš, (SK)

1 Úvod

Aplikácia UML .FRI je vizuálny editor diagramov, aké bežne používajú študenti a pracovníci v oblasti IT. Vyvíjajú ho študenti inžinierskeho štúdia na Fakulte riadenia a informatiky Žilinskej univerzity v rámci svojej projektovej výučby.

Nástroj má síce v názve *UML*, radíme ho však medzi *DSM*[1] modelovacie nástroje. To znamená, že nie je zameraný na jeden modelovací jazyk, ale obsahuje mechanizmus, ktorý interpretuje *metamodel*[2]. Preto je v ňom možné modelovanie napríklad v UML, vývojových diagramoch, DFD, vytváranie grafov či schém zapojenia počítačov do počítačovej siete. Okrem toho, pridanie nového modelovacieho jazyka spočíva iba v pomerne jednoduchom vytvorení nového metamodelu.

Pre profesionálne modelovacie nástroje je charakteristické, že okrem modelovania samotného, obsahujú aj pokročilejšie funkcie. Napríklad umožňujú generovanie zdrojových kódov na základe UML diagramu tried alebo automatické usporiadanie prvkov diagramu podľa určitých pravidiel. Väčšina týchto funkcií je však špecifická pre jeden typ alebo malú skupinu typov diagramov. Do spomínaných profesionálnych nástrojov mohli byť teda zaradené relatívne ľahko hlavne preto, že nástroje samotné sú orientované iba na vybrané modelovacie jazyky.

Tím, starajúci sa o vývoj UML .FRI, považuje takéto funkcie za potrebné a rád by ich do svojho nástroja zaradil. Ak sú však závislé na modelovacom jazyku, musia byť dodávané spoločne s príslušným metamodelom a teda vo forme *zásuvných modulov*.[3]

Z toho dôvodu som si pre svoju diplomovú prácu vybral tému *Návrh a tvorba systému zásuvných modulov pre nástroj UML .FRI*. [4] V rámci jej riešenia som vytvoril prvú verziu,

[1] Domain Specific Modeling

[2] Model definujúci model, alebo tiež gramatika modelovacieho jazyka

[3] Zásuvný modul je počítačový program, ktorý spolupracuje s hlavnou (hostiteľskou) aplikáciou. Poskytuje jej konkrétnu a obyčajne aj veľmi špecifickú funkcionalitu „na požiadanie používateľa".

[4] Systém umožňujúci pripájanie zásuvných modulov k aplikácii a vytvárajúci komunikačné rozhranie medzi aplikáciou a zásuvnými modulmi.

ktorú budem postupne rozširovať. V článku opisujem princípy, na ktorých je tento systém postavený a možnosti, ktoré to do budúcnosti prináša.

2 Požiadavky na systém podpory zásuvných modulov

Aplikácia UML .FRI je vyvíjaná v programovacom jazyku Python, pre grafické rozhranie používa knižnicu GTK+, informácie ukladá pomocou XML. Všetky tieto technológie boli vybrané tak, aby aplikácia mohla fungovať na rôznych operačných systémoch a hardvérových platformách. Z toho vyplýva najpodstatnejšia podmienka, ktorú musí aj nový systém zásuvných modulov spĺňať – musí fungovať rovnako univerzálne ako aplikácia samotná.

Aplikácia je vyvíjaná ako otvorený projekt a najmä oddelenie metamodelov od aplikácie umožňuje pomerne jednoduché rozširovanie zo strany používateľov. Rovnako, zásuvné moduly sú zamerané na možnosť rozšírenia funkcionality programátormi nezávislými na vývojovom tíme UML .FRI. Nedostatočne otestovaný zásuvný modul však môže predstavovať riziko pre stabilitu samotnej aplikácie. Systém zásuvných modulov musí byť preto odolný voči chybám zásuvných modulov.

Jazyk Python má mnoho nesporných výhod voči iným programovacím jazykom a preto je aj použitý pre naprogramovanie aplikácie UML .FRI. Napriek tomu však nie je medzi programátormi príliš rozšírený. Ak by mal byť vývoj zásuvných modulov obmedzený iba na Python, pravdepodobne by to odradilo veľmi veľkú časť potencionálnych tvorcov a to by bola škoda. Musí byť preto umožnené pripájať k aplikácii zásuvné moduly naprogramované v rôznych programovacích jazykoch.

3 Implementácia systému podpory zásuvných modulov

V záujme uspokojenia všetkých požiadavkov a inšpirovaný novým webovým prehliadačom Google Chrome, som sa rozhodol, že každý zásuvný modul bude vykonávaný v samostatnom procese. Dôjde tak k dostatočnej izolácii aplikácie od chýb, ktoré by mohli v zásuvných moduloch vzniknúť. Navyše, univerzálny systém medziprocesovej komunikácie umožní implementáciu zásuvných modulov principiálne v ľubovoľnom programovacom jazyku.

V prvej fáze vývoja som navrhol, že zásuvný modul sa bude k aplikácii pripájať pomocou TCP/IP *socketu*. Je to pomerne jednoducho implementovateľný spôsob. Má však jednu závažnú nevýhodu. Niektorí používatelia môžu mať blokovanú IP komunikáciu na väčšine portov, čím by znemožnili fungovanie tohto systému. Do druhej fázy preto plánujem implementovať komunikáciu aj pomocou rúr (*pipes*).

Pre zachovanie maximálnej prenosnosti systému medzi platformami ale tiež medzi programovacími jazykmi, komunikačný protokol je kódovaný textovo. Konkrétne, vychádza zo špecifikácie *RFC 2822*. Formát správ sa preto podobá na protokoly známe z internetovej

komunikácie (*HTTP*, *SIP*, *SMTP*). [8]

Správy možno rozdeliť na tri typy:

- požiadavka – zaslaná zásuvným modulom,
- odpoveď – zaslaná aplikáciou ako reakcia na požiadavku zásuvného modulu,
- notifikácia – zaslaná aplikáciou v prípade nejakej udalosti

Zásuvný modul môže zasielať požiadavky na tri rôzne časti aplikácie:

- zmena stavu *GUI*[5] aplikácie – aby si pridal svoje vlastné položky do menu aplikácie,
- získanie informácie o aktuálne načítanom metamodeli,
- vzdialené volanie metódy nad objektmi samotného modelu.

Knižničné rozhranie, ktoré môžu použiť programátori zásuvných modulov, je navrhnuté tak, aby abstrahovalo od komunikačného kanála medzi aplikáciou a zásuvným modulom. Celé je to spravené tak, aby mal programátor dojem, že pracuje s objektmi aplikácie. V skutočnosti tu však dochádza k vzdialenému volaniu metód.

4 Možnosti použitia systému zásuvných modulov

Pre lepšiu predstavu ešte raz pripomeniem základné schopnosti aplikácie UML .FRI. Aplikácia samotná umožňuje vytváranie diagramov zložených z elementov a spojení. Obsahuje iba všeobecné funkcie pre prácu s týmito diagramami. Sú to napríklad preskupovanie elementov, ukladanie projektu, či tlačenie diagramu na tlačiarni.

Pokročilejšie funkcie, ako napríklad generovanie zdrojových kódov z UML, však neobsahuje a ani obsahovať nebude. Je to najmä preto, lebo takéto funkcie sú špecifické pre konkrétny modelovací jazyk, čiže pre konkrétny metamodel. Metamodely sú však k aplikácii dodávané samostatne a navyše si ich môžu vytvárať aj samotní používatelia.

Zásuvné moduly preto predstavujú vhodný spôsob, ako rozšíriť funkcionalitu aplikácie o rôzne špecifické algoritmy a zachovať pritom jadro aplikácie malé a univerzálne. Konkrétne príklady použitia zásuvných modulov v aplikácii UML .FRI sú:

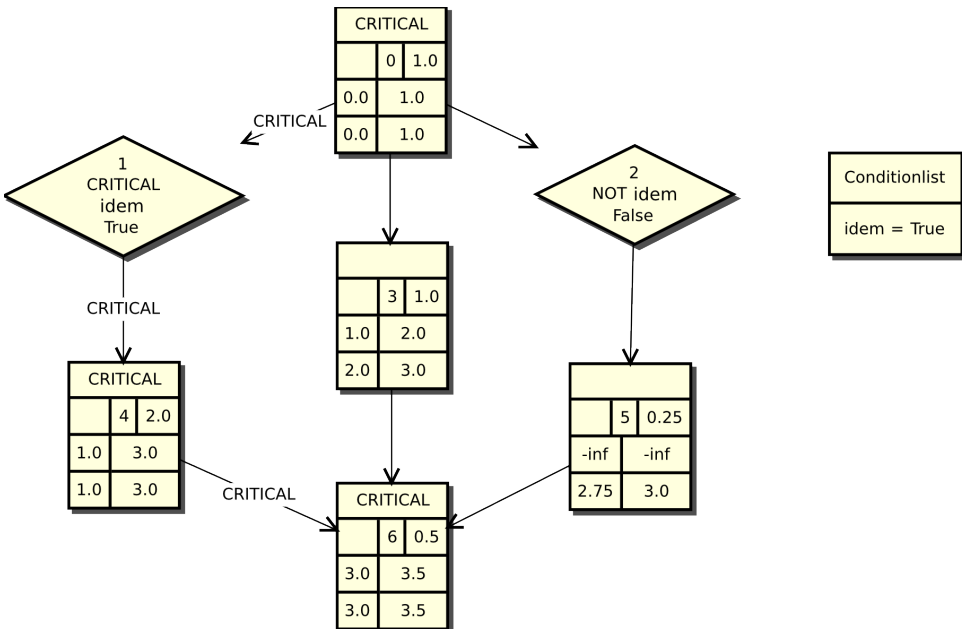
- generovanie zdrojových kódov z diagramov,
- spätné vytváranie diagramov zo zdrojových kódov,
- transformácie model2model,
- implementácia a vizualizácia rôznych algoritmov z teórie grafov,

[5] Graphical User Interface – grafické používateľské rozhranie.

- generovanie metamodelov,
- automatické usporiadanie prvkov diagramu,
- procesne orientovaná simulácia,
- a iné.

Pre úplnosť treba povedať, že nie všetky návrhy sú realizovateľné so súčasnou verziou UML .FRI. V nástroji chýba napríklad ešte podpora pre animáciu, ktorá by bola potrebná pre potreby simulácie.

5 Demonštračný zásuvný modul



Obrázok 1: Príklad sieťového grafu s podmienenými vetvami a vyznačenou kritickou cestou

Napriek tomu, že na realizáciu niektorých nápadov ešte nie je aplikácia pripravená, iné môžu začať vznikať už teraz. Jeden z nich som implementoval ako súčasť svojej diplomovej práce, aby som ním demonštroval funkčnosť svojho riešenia. Zásuvný modul slúži na hľadanie kritickej cesty v sieťovom grafe, ktorý obsahuje podmienené vetvy. Teória ohľadom tejto problematiky je popísaná v [3, 4, 6, 7] a preto sa jej venovať nebudem. Popíšem však v niekoľkých krokoch, ako prebiehal samotný vývoj zásuvného modulu.

V prvom rade bolo potrebné vytvoriť nový metamodel. V tomto prípade šlo o definovanie metamodelu s jedným typom diagramu – sieťovým grafom. Sieťový diagram obsahuje tri typy elementov (činnosť, podmienka, zoznam hodnôt podmienok) a jeden typ spojenia (náväznosť činností). Pre každý typ elementu a spojenia je potrebné vytvoriť definíciu logickej štruktúry dát, ktoré nesie (atribúty), vzhľad a okrem toho treba vymenovať, ktorými typmi spojenia možno spájať jednotlivé typy elementov.

Následne na to som mohol začať programovať samotný zásuvný modul. Zvolil som preň štruktúru triedy, ktorá obsluhuje udalosti vznikajúce v GUI. To znamená, že metóda, ktorá je volaná pri inicializácii zásuvného modulu, pridá do menu aplikácie svoju položku a pre udalosť jej aktivácie zaregistruje ďalšiu svoju metódu. Zásuvný modul potom prechádza do stavu čakania. V prípade, že používateľ aktivuje túto položku, bude vyvolaná obslužná metóda.

Obslužná metóda sa najskôr musí presvedčiť, či je v aplikácii práve otvorený sieťový graf, v opačnom prípade nemá totiž význam niečo počítať. Následne na to si načíta celý diagram a interpretuje ho do svojej internej štruktúry. Potom sa pokúsi o monotónne očíslovanie vrcholov a výpočet najskorších a najneskorších možných začiatkov a koncov činností. Napokon označí vrcholy so zhodnou hodnotou najskoršieho aj najneskoršieho možného začiatku za kritické činnosti a hrany, ktoré ich prepájajú za kritické hrany. Na záver sú výsledky zobrazené v aplikácii.

6 Záver

Úlohou práce bolo umožniť rozširovanie aplikácie UML .FRI pomocou zásuvných modulov. Keď vedúci mojej diplomovej práce vymyslel túto tému, očakával, že na jej základe bude môcť vypísať celú sériu ďalších záverečných prác s rôznou tématikou. Dá sa povedať, že každý z bodov štvrtej kapitoly predstavuje námet na záverečnú prácu minimálne jedného študenta. A treba povedať, že nové nápady sa postupne vynárajú aj teraz.

Moja práca sa však ešte stále nekončí. Súbežne vytváral svoju diplomovú prácu aj môj kolega na tému „Návrh a implementácia systému Undo/Redo[6] do aplikácie UML .FRI". Obaja sme vyvíjali svoje vetvy oddelene od hlavnej aj od seba navzájom. Našou spoločnou úlohou v najbližšom čase bude zaradenie týchto nových systémov do hlavnej vetvy a ich vzájomné prepojenie. Spoločným termínom celého tímu UML .FRI na vydanie novej verzie je september 2009. Vtedy začne nasledujúci akademický rok a my všetci dúfame, že si v ňom UML .FRI získa pevné postavenie medzi učiteľmi aj študentmi Fakulty riadenia a informatiky.

Poďakovanie

[6] Operácia *Undo* znamená vrátenie predchádzajúcej používateľovej akcie. Operácia *Redo* zruší predchádzajúcu *Undo* operáciu.

Táto práca vznikla s podporou grantovej agentúry KEGA v rámci riešenia projektu 3/2158/04 „Využitie open source softvéru vo výučbe na vysokých školách“.

Literatúra

[1] BAČA, T.: *Návrh a tvorba systému zásuvných modulov pre nástroj UML .FRI*. 2009

[2] BAČA, T. – JURÍČEK, M. – ODLEVÁK, P. *Case tool development* Journal of Information, Control and Management Systems 2008. vol.6, ISSN 1336-1716.

[3] BACHRATÝ, H. – SADLOŇ, Ľ.: *Modifikácia zovšeobecnených sieťových grafov*. 2005. nepublikované.

[4] BACHRATÝ, H. – SADLOŇ, Ľ.: *Výpočet prevádzkových intervalov pomocou sieťových grafov*. INFOTRANS 2005. Pardubice : DPJF Pardubice, september 2005. ISBN 80-7194-792-X

[5] JANECH, J. – BAČA, T. – ODLEVÁK, P – a i.: *Projektová dokumentácia*. 2008. nepublikované.

[6] LUKÁČ, M.: Softvérový nástroj na návrh a výpočty sieťových grafov s podmienkovými hranami. 2005.

[7] PALÚCH, S.: *Skriptá z teórie grafov*. 2001. Dostupné online: `http://frcatel.fri.uniza.sk/users/paluch/grafy-pdf-pics.zip`

[8] RFC 2822 Internet Message Format. 2001. Dostupné online: `http://www.ietf.org/rfc/rfc2822.txt`

Kontaktná adresa

Tomáš Bača
`tomas.baca@kst.uniza.com`