Špecifikácia fyzického návrhu WEB aplikácií v CASE nástroji a jeho mapovanie do implementačného prostredia

Peter BLŠTÁK

FIIT STU Bratislava,
14.októbra, Bratislava
peter.blstak@softec.sk

Vedúci práce: Ing. Juraj Červeň, CSc.

Abstrakt. Tvorba softvérových systémov vyžaduje stále rýchlejší a systematickejší prístup. Z tohto dôvodu sa stále hľadajú lepšie spôsoby špecifikácie softvérových systémov a vytvárajú podporné nástroje na ich tvorbu. V tomto dokumente sa venujeme problematike návrhu agendových web aplikácií v CASE nástroji s využitím UML a problematike prechodu z návrhu aplikácie do jej implementácie prostredníctvom generovania. Opísaná špecifikácia návrhu vychádza z doménového objektovo-orientovaného návrhu, realizovaného prostredníctvom diagramu tried, ktorý modeluje aplikačnú vrstvu aplikácie. Súčasťou špecifikácie návrhu je aj dynamický model prezentačnej vrstvy aplikácie, ktorý je realizovaný prostredníctvom stavového diagramu. Pre zjednodušenie generovania aplikácií a pre zvýšenie množstva vygenerovaného kódu sme v rámci projektu vytvorili cieľové prostredie – TF framework. Na základe špecifikácie návrhu web aplikácií bol navrhnutý, implementovaný a overený generátor aplikácií do TF pre CASE nástroj Poseidon for UML.

Kľúčové slová: web aplikácia, špecifikácia, návrh, modelovanie, UML, generovanie

1 Úvod

S príchodom rýchlejších komunikačných liniek, rozširovaním Internetu a lokálnych počítačových sietí možno pozorovať narastajúce počty existujúcich klient-server aplikácií, ktoré poskytujú svojím používateľom cez počítačové siete širokú funkcionalitu. V súčasnosti ide veľmi často o klient-server aplikácie, ktorých funkčnosť je prístupná prostredníctvom web prehliadača – tenkého klienta (ďalej len *web aplikácie*).

V súvislosti s rozširovaním web aplikácií boli vyvinuté mnohé technológie umožňujúce ich rýchlejší a jednoduchší vývoj. V súčasnosti si však trh vyžaduje ešte rýchlejší vývoj web aplikácií, ktoré by zároveň boli vytvárané viac systematicky s možnosťou ich jednoduchšieho rozširovania a prispôsobovania novým požiadavkám. Odpoveďou na požiadavku rýchlejšieho a systematickejšieho vývoja

web aplikácií môže byť vytvorenie framework-u, ktorý by zjednodušoval ich budovanie a realizácia generátora, ktorý by jednoduchým spôsobom umožňoval vytvárať funkčnú kostru aplikácie z jej fyzického návrhu. Pod fyzickým návrhom aplikácie rozumieme podrobný návrh aplikácie, ktorý zohľadňuje cieľové implementačné prostredie [6]. Fyzický návrh teda predstavuje špecifikáciu pre fázu implementácie a generátor zabezpečuje rýchlejší a automatizovaný prechod z návrhu do implementácie. Ak je návrh vyjadrený prostredníctvom UML s použitím CASE nástroja [5], je zároveň dostupná kvalitná dokumentácia k vytváranému systému. Týmto spôsobom je zabezpečená jednotná štruktúra vytváraných aplikácií a použitie štandardnej notácie pre realizáciu fyzického návrhu.

Pre úspešné použitie generátora je potrebné obmedziť generátor aj cieľové prostredie na určitú doménu. V našom prípade sme sa rozhodli pre tvorbu a generovanie agendových web aplikácií menšieho až stredného rozsahu. Keďže tento typ aplikácií je zameraný prevažne na uskladnenie a manipuláciu väčšieho množstva údajov s relatívne statickou štruktúrou a má zväčša jednoduchú funkcionalitu, predstavuje ideálny typ aplikácií pre modelovanie a následné generovanie. Ide pritom o triedu aplikácií s veľmi širokou oblasťou uplatnenia.

2 Cieľové prostredie

Aby bolo využitie generátora pri jeho použití naozaj dobré, je potrebné venovať veľké úsilie výberu a úprave, príp. tvorbe cieľového prostredia. Jeho dobré vlastnosti (najmä vhodnosť pre riešenie danej triedy aplikácií a vysoká deklaratívnosť kódu) môžu v značnej miere zlepšiť možnosti samotného modelovania a generovania aplikácií.

Z tohto dôvodu sme po analýze dostupných technológií a framework-ov na tvorbu web aplikácií vytvorili v rámci diplomovej práce TF framework. Framework integruje framework-y Struts [11] a OJB [9], rozširuje ich o ďalšiu funkcionalitu a zvyšuje deklaratívnosť zdrojových kódov pre potreby generovania agendových web aplikácií.

2.1 Architektúra

TF framework využíva klient-server architektúru s tenkým klientom. Vytváraným aplikáciám ponúka na strane servera trojvrstvovú architektúru s využitím architektonického vzoru MVC v prezentačnej vrstve. Prezentačná vrstva je realizovaná framework-om Struts, doplneným o niekoľko ďalších čŕt vrátane prepojenia na aplikačnú vrstvu TF framework-u. Aplikačná vrstva je realizovaná prostredníctvom JavaBeans perzistentných tried mapovaných do RDBMS OR mapovačom OJB. Databázová vrstva je realizovaná prostredníctvom OJB a voľne dostupným RDBMS – MySQL.

2.2 Aplikačná vrstva

Agendové aplikácie sú primárne zamerané na manipuláciu s veľkým množstvom perzistentných údajov, ktoré sú zväčša uchovávané v nejakom externom úložisku (napr. RDBMS). Keďže Struts neposkytoval konkrétne riešenie modelu aplikácie ani perzistenciu údajov, bolo potrebné navrhnúť spôsob ako vytvárať aplikačnú vrstvu a ako pracovať s perzistentnými údajmi.

Zvolili sme doménovo orientovaný objektový prístup s mapovaním dómenových objektov do relačnej databázy, podobne ako je tomu v iných framework-och (napr. FAAST, JavaTEC [7]). Tento prístup umožňuje dobre štrukturovať aplikáciu spôsobom blízkym ľudskému zmýšľaniu a je vhodný aj pre potreby modelovania a generovania (diagram tried). Model teda pozostáva z množiny perzistentných tried (resp. objektov), ktoré reprezentujú základné entity doménovej oblasti a z množiny väzieb medzi týmito triedami (objektami).

V TF framework-u sú triedy modelu riešené ako obyčajné JavaBeans triedy. Perzistencia objektov je definovaná deklaratívnym spôsobom pomocou konfiguračného súboru pre OJB mapovač. Vzhľadom na širokú podporu RDBMS systémov, ktorú poskytuje OJB, možno povedať, že TF framework je nezávislý od použitého RDBMS.

Podstatná časť funkcionality tried modelu je sústredená do tzv. aplikačných metód (podobne ako vo FAAST-e), ktoré možno jednoduchým spôsobom spúšťať z používateľského rozhrania. Sú definované tri základné typy aplikačných metód:

- objektové – vykonávajú funkciu nad aktuálnym objektom
- globálne – realizujú funkciu nezávislú od konkrétneho objektu (napr. dávkové spracovanie údajov)
- kolekciové – vykonávajú funkciu nad vybranou množinou objektov (napr. tlačenie hromadnej zostavy)

2.3 Prezentačná vrstva

Základnými grafickými prvkami prezentačnej vrstvy sú formuláre. Keďže Struts poskytuje podporu na tejto úrovní grafických komponentov, aj nadstavba vytvorená v TF framework-u je realizovaná na úrovni formulárov.

V agendových aplikáciách sa používajú formuláre, ktoré poskytujú dva základné pohľady na existujúce údaje: **detail** jedného objektu, poprípade aj so zoznamom(ami) asociovaných objektov (napr. titul a jeho knihy) a **zoznam** viacerých objektov rovnakého typu (napr. zoznam titulov), ktoré možno prehľadávať, triediť a filtrovať podľa zvolených kritérií. Prvý typ pohľadu je vhodný na vytváranie, modifikáciu a detailné prezeranie objektov, kým druhý je vhodný na manipuláciu s množinou objektov, vyhľadávanie a spúšťanie kolekciových a objektových aplikačných metód. TF framework poskytuje podporu pre oba spomenuté typy formulárov s podporou pre filtrovanie, usporadúvanie kolekcií a asociácií. Okrem toho poskytuje podporu pre budovanie **jednoduchých** formulárov, ktoré nezobrazujú perzistentné objekty (napr. prihlasovací formulár). Všetky typy formulárov TF framework-u majú zabudovanú podporu pre vykonávanie akcií nad aktuálnym formulárom, spúšťanie aplikačných metód nad doménovými objektami aplikačnej vrstvy a spúšťanie špecifickej podpory framwork-u (napr. otvorenie okna na vytvorenie nového objektu, odstraňovanie objektov z kolekcie a pod.). Všetky akcie sú realizované prostredníctvom TF forwad-ov, pričom každý TF forward zodpovedá konkrétnemu grafickému prvku formulára (tlačidlo, linka).

Framework poskytuje správu otvorených formulárov, ktorá umožňuje návrat k ľubovolnému z predchádzajúcich otvorených formulárov. Otvorené formuláre sú uchovávané v zásobníku, do ktorého sa formuláre automaticky pridávajú pri ich otváraní. Otvorenie formulára môže byť realizované v troch režimoch:

- `add` – pridanie na koniec zásobníka (štandardné otvorenie formulára)
- `replace` – pridanie na koniec zásobníka s odstránením posledného formulára
- `clear` – pridanie formulára, ktoré zabezpečuje odstránenie predchádzajúcich formulárov so zásobníka

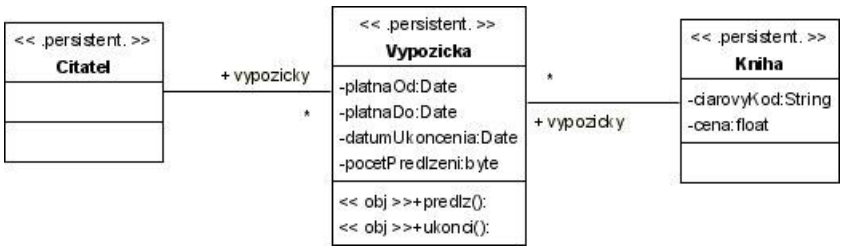
3 Špecifikácia návrhu WEB aplikácií

Pri špecifikácii návrhu sme vychádzali z možností implementovaného cieľového prostredia (pozri *2 Cieľové prostredie*). Snažili sme sa splniť dva základné ciele. Na jednej strane sme chceli zabezpečiť jednoduchosť a prehľadnosť návrhu agendových web aplikácií a na druhej strane sme chceli, aby sa z takéhoto návrhu dala vygenerovať pokiaľ možno čo najväčšia časť hotovej aplikácie. Keďže tieto dva ciele boli v protiklade, išlo o hľadanie kompromisu medzi množstvom informácií, ktoré bude obsahovať návrh aplikácie a množstvom vygenerovaného kódu. Išlo teda o nájdenie stavu, kedy by už pridávanie ďalších čŕt do návrhu aplikácie v UML modeli neprinášalo zjednodušenie tvorby generovaných aplikácií.

Pri návrhu agendových web aplikácií sme sa rozhodli vychádzať z doménového objektovo orientovaného návrhu aplikačnej vrstvy aplikácie. Okrem aplikačnej vrstvy sme sa rozhodli modelovať aj prezentačnú vrstvu aplikácie. Návrh web aplikácie teda pozostáva z dvoch častí – návrh aplikačnej vrstvy vyjadrenej pomocou diagramu tried a návrh prezentačnej vrstvy, ktorú sme sa rozhodli modelovať pomocou stavového diagramu.

3.1 Návrh aplikačnej vrstvy

Ako sme už spomínali aplikačná vrstva aplikácie je modelovaná diagramom tried, ktorý pokrýva doménu aplikácie. Každá trieda reprezentuje entitu domény, ktorú má navrhovaná aplikácia evidovať a vykonávať nad ňou aplikačnú funkčnosť. Príkladom takýchto entít môžu byť kniha, výpožička a čitateľ knižničného systému, ktoré sú modelované triedami `Kniha`, `Vypozicka` a `Citatel` a asociáciami medzi nimi (pozri *Obr. 1*).



Obr. 1 – Model výpožičky knihy

Model teda opisuje entity aplikácie, ich vlastnosti (atribúty), vzťahy medzi entitami (asociácie) a funkcie, ktoré sa dajú nad entitami vykonať (metódy).

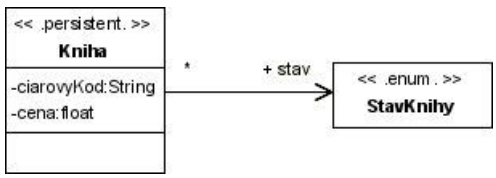
Každá perzistentná doménová trieda (trieda so stereotypom `.persistent.`) má automaticky schopnosť uloženia v perzistentnom dátovom úložisku, do ktorého sú

mapované všetky jej nederivované atribúty a asociácie. Atribúty a asociácie môžu byť perzistentné alebo derivované. Derivované atribúty a asociácie je potrebné vypočítať zo stavu aktuálneho objektu a nie je možné ich priamo meniť.

Manipulácia s asociovanými objektami je možná len v prípade, že zodpovedajúce asociácie sú navigovateľné alebo pomenované na vzdialených koncoch. Pre takéto asociácie je možné automaticky vykonávať získavanie hodnoty, pridávanie a odstraňovanie pre asociácie k N, resp. získavanie a nastavovanie hodnoty pre asociácie k 1. Podporované sú všetky typy asociácií okrem typov 0..1 : 0..1 a 1 : 1, kde by bol problém s určením pozície cudzieho kľúča v RDBMS, resp. s udržiavaním konzistencie v prípade existencie cudzích kľúčov pre každú entitu.

Metódy triedy reprezentujú funkčnosť, ktorú možno vykonať nad objektom triedy, nad skupinou objektov, resp. nad všetkými objektami danej triedy. Metódy môžu byť obyčajné alebo aplikačné. Aplikačné metódy sú dostupné z prezentačnej časti aplikácie. Aplikačné metódy sú troch typov – objektové, kolekciové a globálne (pozri *2.2 Aplikačná vrstva*) a ich typ je určený stereotypom (`obj`, `coll`, `glob`).

Pre každý atribút, asociáciu a metódu je možné okrem štandardných vlastností UML nastaviť aj zobrazovaný názov (`caption`) a pomocný vysvetľujúci text – popis (`description`), ktoré sú zobrazované v prezentačnej časti aplikácie. Pre aplikačné metódy je možné špecifikovať aj potvrdzujúcu správu, ktorá sa má zobraziť pred vyvolaním metódy (`confirmation`) a zoznam používateľských rolí, pre ktoré je metóda dostupná.



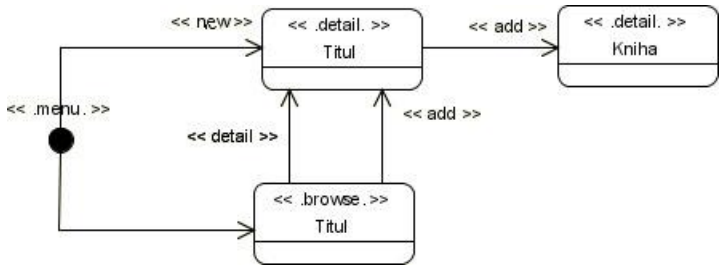
Obr. 2 – Vymenované hodnoty stav knihy

Ďalšími špeciálnymi entitami doménového modelu sú atribúty, typu vymenované hodnoty – enumerácia. V modeli návrhu aplikácie im zodpovedá asociácia k 1 na triedu so stereotypom `enum`. Pre tieto triedy je možné špecifikovať zoznamy kódov a hodnôt (`code`, `value`), ktoré má daná enumerácia obsahovať. Príkladom číselníka je stav knihy (pozri *Obr. 2*).

3.2 Návrh prezentačnej vrstvy

Prezentačná vrstva agendovej web aplikácie je tvorená formulármi, ktoré prevažne zobrazujú doménové objekty aplikácie. Zväčša ide o zobrazenie jedného objektu v detailnom pohľade alebo zobrazenie zoznamu objektov rovnakého typu v tabuľke s možnosťou filtrovania a usporadúvania. Vo formulároch sú tiež k dispozícii funkcie aplikácie, ktoré sú prístupné prostredníctvom liniek a tlačidiel. Súčasťou prezentačnej vrstvy je aj navigačné menu.

Návrh prezentačnej vrstvy aplikácie realizujeme prostredníctvom stavového diagramu, ktorý špecifikuje formuláre a navigáciu medzi nimi [1, 2, 3, 4, 5]. Formuláre sú v modeli reprezentované stavmi a prechody medzi formulármi sú v modeli reprezentované prechodmi medzi stavmi (pozri *Obr. 3*).



Obr. 3 – Formuláre a navigácia medzi nimi

Formuláre môžu byť jedného z troch typov – detailný formulár, browse formulár a jednoduchý formulár (pozri *2.3 Prezentačná vrstva*). Typ formulára je špecifikovaný stereotypom stavu, ktorý ho reprezentuje v modeli (`detail`, `browse` alebo `simple`). Detail a browse formuláre zobrazujú doménové entity modelu – niektorú z tried špecifikovanú v návrhu aplikačnej vrstvy web aplikácie (pozri *3.1 Návrh aplikačnej vrstvy*). Doménovú triedu, ku ktorej sa vzťahuje formulár je možné určiť špeciálnym atribútom.

Detail formulár je zobrazenie jednej entity, s možnosťou vykonávania úprav, uloženia do perzistentného úložiska, usporadúvaním asociovaných objektov a vykonávaním objektových a globálnych aplikačných metód nad aktuálnym objektom formulára a objektových a kolekciových aplikačných metód nad jeho asociovanými objektami vrátane ich odstraňovania. Browse formulár je zobrazenie zoznamu entít modelu, ktoré je možné filtrovať, usporadúvať, odstraňovať a vykonávať nad nimi všetky objektové a kolekciové aplikačné metódy. Tretí typ formulárov neslúži na zobrazenie doménového perzistentného objektu a ich obsah nie je v UML modeli aplikácie určený. Pre takéto formuláre je možné špecifikovať len prechody zodpovedajúce TF forward-om, ktoré zabezpečujú volanie metód formulára (nemôžu volať priamo aplikačné metódy).

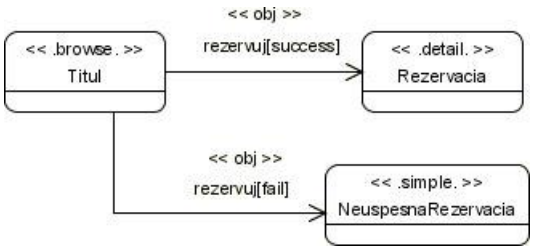
Prechody medzi stavmi zodpovedajú aplikačným TF forward-om, ktoré sú zobrazované na obrazovke v podobe liniek alebo tlačidiel. Pre všetky prechody možno špecifikovať roly, ktoré ich majú možnosť vykonať ako aj typ otvorenia cieľového formulára (`add`, `replace`, `clear`).

Špeciálnym stavom je počiatočný stav so stereotypom `menu`, ktorý reprezentuje úvodnú obrazovku aplikácie a prechody z tohto stavu reprezentujú položky navigačného menu aplikácie.

Prechody môžu byť špeciálne, alebo môžu zodpovedať aplikačným metódam resp. metódam formulárov. Medzi špeciálne typy prechodov patria prechody `new`, `detail`, `add` a `select`. Typ je vyjadrený stereotypom prechodu. Prechod typu `new` reprezentuje prechod na detailný formulár, ktorý zobrazuje vytvorený objekt s iniciálnymi hodnotami. Tieto hodnoty je možné zmeniť a uložiť do perzistentného úložiska. Prechod typu `detail` reprezentuje prechod na detail asociovaného objektu aktuálneho formulára. Má význam najmä pri prechode na detail objektu zobrazeného v tabuľke aktuálneho formulára. Prechod typu `add` reprezentuje otvorenie detailného formulára, pri ktorého potvrdení sa zobrazovaný objekt pridá do špecifikovanej kolekcie, resp. asociácie objektu zdrojového formulára. Prechod typu `select` zabezpečuje otvorenie browse formulára pre výber, po ktorého potvrdení sa výber

priradí alebo pridá do špecifikovanej kolekcie, resp. asociácie objektu zdrojového formulára. Pri výbere objektov v browse formulári je možné vyberať jeden alebo viacero objektov v závislosti od kardinality asociácie, do ktorej sa má tento výber priradiť.

V prípade prechodov typu `detail`, `add` a `select` je možné špecifikovať názov asociácie, na ktorú sa zodpovedajúca akcia vzťahuje. Názov asociácie sa špecifikuje pomocou názvu prechodu.



Obr. 4 – Prechody po vykonaní aplikačnej metódy

Ako sme spomínali prechod môže zodpovedať aj ľubovoľnej aplikačnej metóde doménového objektu, ktorý je zobrazovaný formulárom, resp. metóde formulára. V takýchto prípadoch udalosť prechodu (event) špecifikuje názov metódy a podmienka prechodu (guard) špecifikuje logický výsledok akcie, ktorý podmieňuje prechod na nasledujúci formulár. Špecifikovaním viacerých prechodov s rovnakým názvom metódy a možno docieliť vetvenie toku obrazoviek závislé od výsledku spúšťanej metódy (pozri *Obr.* 4).

Pre aplikačné metódy, po ktorých vykonaní nenasleduje prechod na iný formulár nie je potrebné v stavovom diagrame špecifikovať žiaden prechod. Všetky aplikačné metódy danej triedy sú automaticky obsiahnuté vo formulári, ktorý túto triedu zobrazuje.

Pre každý prechod je možné špecifikovať podobne ako pre aplikačné metódy zobrazovaný názov (`caption`) a pomocný vysvetľujúci text (`description`), prípadne potvrdzujúcu správu (`confirmation`).

4 Realizácia a overenie riešenia

Na základe špecifikácie návrhu web aplikácií a možností cieľového prostredia sme podrobne navrhli a vytvorili generátor aplikácií pre CASE nástroj Poseidon for UML [10]. Generátor sme realizovali ako cartridge (náplň) pre opensource nástroj AndroMDA [8]. Generátor a špecifikáciu návrhu web aplikácií sme overili vytvorením modelu a vygenerovaním aplikácie knižničného systému.

5 Záver

V dokumente sa venujeme problematike špecifikácie návrhu agendových web aplikácií a problematike prechodu z tohto návrhu do implementácie prostredníctvom generovania. Špecifikovali sme návrh web aplikácií, ktorý využíva diagramy UML notácie – diagram tried a stavový diagram. Diagram tried reprezentuje aplikačnú

a databázovú vrstvu aplikácie a stavový diagram modeluje prezentačnú vrstvu aplikácie. Návrh aplikácie vytvorený podľa tejto špecifikácie pokrýva všetky časti vytváranej aplikácie a pritom zostáva dostatočne jednoduchý a prehľadný.

Špecifikáciu návrhu agendových web aplikácií sme overili vytvorením ukážkovej aplikácie knižničného systému, čím sme potvrdili možnosť použitia vytvoreného riešenia a jeho uplatnenia v praxi.

Možnosti pre ďalšie rozvíjanie realizovaného systému vidíme najmä v dorobení podpory pre modelovanie transient a session objektov doménovej vrstvy, zdokonaľovaní prezentačnej vrstvy TF framework-u a realizovaní generovania JSP stránok v dvoch fázach: najskôr generovať obrazovky do špeciálneho XML súboru, z ktorého by sa potom mohli vytvárať rôzne výzory JSP stránok XSLT transformáciou.

Literatúra

Referencie na články:

1. Ambler, S. W.: *User interface design: tips and techniques.* AmbySoft Inc. White Paper, 1998. http://www.ambysoft.com/userInterfaceDesign.pdf
2. Coldwey, J., Kruger, I.: *Form-Based User Interface – The Architectural Patterns*, 1997. http://www.riehle.org/community-service/hillside-group/europlop-1997/p9final.pdf
3. Dolog, P., Bieliková, M.: *Transforming State Diagrams into Navigation Objects in Hypermedia Applications*, KIVT FEI Slovenská technická univerzita, 2002.
4. Draheim, D., Weber, G.: *Modeling Submit / Response Style Systems with Form Charts and Dialogue Constraints*, Institute of Computer Science, Freie Universität Berlin, August, 2003. http://www-staff.it.uts.edu.au/~wgardner/papers/HCISWWA12.pdf

Referencie na knihy:

5. Fowler, M., Scott, K.: *UML distiled – Applying the Standard Object Modeling Language*, Addison Wesley, 5th Printing, December, 1997, ISBN 0-201-32563-2.
6. Sommerville, I.: *Software engineering*, Addison-Wesley Publ. Company, 5th Eddition, 1996.
7. Pribula, E., Slyško P., Šolc, R.: *Návrh frameworku JavaTEC – 1.etapa.* Softec s.r.o., 9.2001.

Referencie na internetové zdorje:

8. AndroMDA, domovská stránka produktu, http://www.andromda.org/
9. OJB, domovská stránka produktu, http://db.apache.org/ojb/
10. Poseidon for UML, domovská stránka produktu, http://www.gentleware.com/
11. Struts, domovská stránka framework-u, http://jakarta.apache.org/struts

Peter Blšták

Annotation:

Specification of web applications physical design in CASE and its mapping to an implementation environment

Software systems design requires constantly prompter and more systematic approach. Therefore, new ways of software systems specification are being searched for and support tools for software systems development are being created. In this paper we discuss an issue of web applications design using UML and an issue of transition from the applications design to its implementation via code generation. We describe the design specification, which is based on domain object-orientated design realized using class diagrams by which the application layer is modeled. A part of design specification is dynamic model of web application presentation layer, which is built on state diagrams. For simplifying the application code generation and for increasing the amount of generated code, within this project, we have created target framework – TF framework. On the bases of presented web applications design specification, we have designed, implemented and attested the application code generator for created TF framework using CASE tool Poseidon for UML.