A TOOL FOR MODELLING OF COMPONENT-BASED SYSTEMS

Zoltán Zemko

Master Degree Programme (3), FIT BUT

E-mail: xzemko01@stud.fit.vutbr.cz

Supervised by: Marek Rychlý

E-mail: rychly@fit.vutbr.cz

Abstract: The paper deals with the development of a graphical tool for modeling of component-based software systems It describes basic concepts and features of the component-based systems, provides a short introduction to Domain-Specific Languages, and proposes the utilizations of Eclipse Modeling Framework and Graphical Modeling Framework in descriptions of abstract and concrete syntaxes during development phases of the graphical modeling tool for component-based systems.

Keywords: EMF, GMF, Component Software, Component Diagram, Eclipse

1. ÚVOD

Cieľom projektu bolo vyvinúť jednoduchý, efektívne použiteľný modelovací nástroj pre podporu grafického návrhu komponent systémov pod platformu Eclispe. Výsledný produkt musí poskytovať podrobný validačný mechanizmus na overovanie konzistentnosti údajov v modeli. Vynikajúcou súčasťou tohto mechanizmu je, oproti väčšine modelovacím nástrojom zamerajúcich sa na túto problémovú doménu, že produkt nielenže dokáže zistiť, ale v niektorých prípadoch dokáže aj zabrániť vzniku chýb.

V prvej, teoretickej časti príspevku sú vysvetlené pojmy ako komponent software a Doménovo-špecifický jazyk. V druhej polovici sa príspevok zameriava na postup vývoja modelovacieho nástroja pre grafický návrh komponent systémov, resp. na postup vytvorenia abstraktnej a konkrétnej syntaxe. V závere príspevku sa uvedie jednoduchý prípad použitia modelovacieho nástroja na modelovanie informačného systému zloženého z komponent.

2. KOMPONENT SOFTWARE

Software tvorený množinou komponentov, vytvárajúcich kompozíciu funkčného systému, sa nazýva komponent software. Komponent je samostatná výkonná jednotka, ktorá je ľahko udržiavateľná a jednoducho vymeniteľná za iný komponent komunikujúci s okolím cez rovnaké rozhranie. Ďalšou výhodou použitia komponent je ich *znovu použiteľnosť* vo viacerých projektoch. [1]

3. DOMÉNOVO-ŠPECIFICKÝ JAZYK

Doménovo-špecifické jazyky (DSL) tvoria triedu programovacích alebo špecifikačných jazykov na popis a riešenie úloh obmedzenej, konkrétnej problémovej domény. Zložitejšiu problémovú oblasť je možné riešiť vytvorením viacerých DSL jazykov, ktoré tvoria jednu rodinu DSL jazykov. [2]

3.1. ABSTRAKTNÁ SYNTAX

Jadrom každého DSL jazyka je *abstraktná syntax*, ktorá sa používa pri vývoji pre každý artefakt, vrátane grafickej konkrétnej syntaxe, model na model transformácií a model na text transformácií. *Abstraktná syntax* nového DSL popisuje štruktúru modelu konkrétnej modelovanej oblasti.

Pre tento účel sa používa *Eclipse Modeling Framework* (EMF). Využitím *EMF ECore* modelu sa vytvorí meta-model reprezentujúci abstraktnú syntax nového DSL jazyka. [2]

3.2. KONKRÉTNA SYNTAX

Konkrétna syntax definuje textovú alebo grafickú reprezentáciu prvkov v modelovacom nástroji definovaných v abstraktnej syntaxi DSL. *Graphical Modeling Framework* (GMF) slúži na vývoj grafickej konkrétnej syntaxe. Využitím GMF je možné vytvoriť grafický zápis pre DSL, mapovať ho do abstraktnej syntaxe a vygenerovať grafický editor modelu DSL jazyka. [2]

4. VÝVOJ ABSTRAKTNEJ SYNTAXE MODELOVACIEHO NÁSTROJA

4.1. ECORE MODEL

Využitím *EMF ECore* bol vytvorený model, ktorý reprezentuje *abstraktnú syntax* nového modelovacieho nástroja. Bol navrhnutý tak aby umožnil vytvoriť jednoduchú, prehľadnú hierarchickú dekompozíciu komponent, špecifikáciu jednoduchých vlastností a prepojení medzi nimi.

Základným prvkom modelovacieho nástroja je abstraktný prvok kontajner (*Container*), ktorý bude uchovávať zoznam do neho vnorených prvkov. Kontajner má dva konkrétne špecializované prvky ako model (*Model*) a zložený komponent (*CompositeComponent*).

Vytvorený *Model* sa môže vyskytnúť iba ako koreňový prvok hierarchického stromu modelu, kým prvok typu *CompositeComponent* sa môže vyskytovať ľubovoľný krát, totiž reprezentuje špeciálny prípad komponenty.

Ako bolo už naznačené, prvok abstraktného typu *Container* obsahuje zoznam vnorených prvkov (abstraktný typ *Element*). Abstraktný typ *Element* sa ďalej špecializuje na typy *Entity* a *Relationship*.

Entity v modelovacom nástroji tvoria komponenty (*Component*), rozhrania (*Interface*) a artefakty (*Artifact*). Typ *Component* je možné stále špecializovať na obyčajný komponent (*SimpleComponent*) a zložený komponent (*CompositeComponent*).

Relácie (*Relationship*) medzi entitami môžu tvoriť relácia závislosti (*Dependency*), relácia použitia (*Usage*), relácia realizácie (*Realization*), relácia delegácie (*Delegation*) a relácia dedičnosti (*Generalization*).

Zložený komponent poskytuje/požaduje služby od okolia cez porty (*Port*). Porty sa špecializujú na dve kategórie, ako sú poskytujúci port (*ProvidingPort*) a požadujúci port (*RequiringPort*). Poskytujúci port okoliu poskytuje určitú službu realizovanú vo vnútri zloženého komponentu, kým zložený komponent cez požadujúci port využíva konkrétnu službu od okolia.

Z ECore modelu bol vygenerovaný, použitím *EMF generátora*, zdrojový kód meta-modelu DSL jazyka a jednoduchého EMF editoru založeného na editácií hierarchickým stromom.

4.2. VALIDÁCIA MODELU

EMF ECore model popisuje štruktúru modelu tvoreného modelovacím nástrojom a takto dokáže predísť vloženiu nekonzistentných prvkov podľa pravidiel štruktúry modelu. Nedokáže však odhaliť porušenie komplexnejších pravidiel, ako napr. cyklická dedičnosť, preto základnú sémantiku abstraktnej syntaxe bolo treba rozšíriť o validačný mechanizmus.

Na overovanie konzistentného stavu modelu bol vytvorený plug-in pre Eclipse, ktorý obsahuje validátori definujúce pravidlá, ktoré musia byť splnené, pre každý prvok v modeli. EMF na validáciu modelu používa *Eclipse Validation Framework*, ktorý pre každý validovaný objekt volá metódu *validate(..)* z typu *EObjectValidator*. Z tohto dôvodu bola vytvorená špecializovaná trieda typu *EObjectValidator*, ktorá sa zakladá na návrhovom vzory Adaptér a deleguje úlohu validácie, podľa typu vstupného objektu v metóde *validate(..)*, na konkrétny validátor.

Validačný mechanizmus dokáže odhaliť chyby v modeli a v niektorých prípadoch i zabrániť vzniku týchto chýb. Modelovací nástroj vďaka tomuto mechanizmu vyniká oproti väčšine nástrojov zameraj úce sa na modelovanie komponent systémov.

5. VÝVOJ KONKRÉTNEJ SYNTAXE MODELOVACIEHO NÁSTROJA

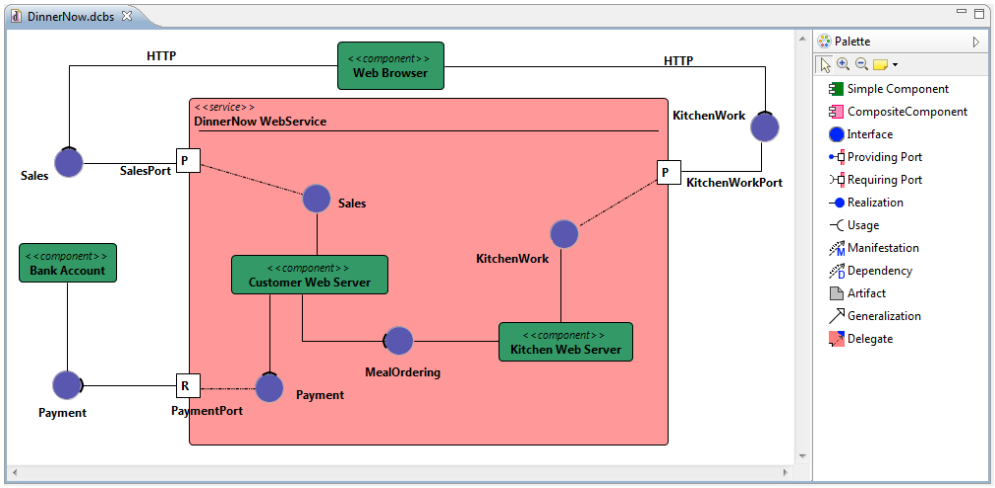
V prvej fáze vývoja konkrétnej syntaxe modelovacieho nástroja bol vytvorený *model grafických definícií* a *model nástrojov*. V modeli grafických definícií sú definované tvary, ktoré slúžia na reprezentáciu uzlov a spojení v modelovacom nástroji. Model nástrojov definuje paletu na vytvorenie prvkov v modelovacom nástroji.

V nasledujúcom kroku sa vytvoril *model mapovania*, ktorý mapuje grafickú reprezentáciu prvkov z modelu grafických definícií a prvok na palete z modelu nástrojov na entitu z ECore modelu, čiže z abstraktnej syntaxe.

Z modelu mapovania bol vytvorený *model GMF generátora*, ktorý umožnil vygenerovať zdrojový kód grafického modelovacieho nástroja. V tomto modeli bolo treba nakonfigurovať vlastnosti generovania ako napr. prípona súboru diagramu, či povolenie validácie v modelovacom nástroji.

6. PRÍPAD POUŽITIA MODELOVACIEHO NÁSTROJA

Použitie výsledného modelovacieho nástroja je demonštrované na prípade modelovania webovej služby *DinnerNow*, ktorá poskytuje služby objednávania večery pre zákazníkov cez internet.



Obrázek 1: Prípad použitia modelovacieho nástroja.

7. ZÁVER

Sila modelovacieho nástroja sa vyniká v jej jednoduchej a prehľadnej grafickej syntaxe. Ďalej poskytuje podrobnú validáciu modelu, oproti už existujúcim riešeniam, ktorá dokáže odhaliť chyby a prípadne zabrániť vzniku už počas modelovania.

Tento príspevok vznikol za podpory grantu FIT-S-11-2 a výskumného zámeru MSM 0021630528.

REFERENCIE

[1] C. Szypersky: Component Software, Beyond Object-Oriented Programming, Second Edititon, Addison-Wesley, 2002, ISBN-978-0201745726

[2] R.C. Gronback: A Domain-Specific Language Toolkit, Addison-Wesley, 2009, ISBN-13: 978-0321534071