Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Projekt 5

Generování konfigurací softwarových komponent z modelů vlastností

Plzeň 2018 Vaněk Jakub

Místo této strany bude zadání práce.

Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 19. prosince 2018

Vaněk Jakub

Abstract

The text of the abstract (in English). It contains the English translation of the thesis title and a short description of the thesis.

Abstrakt

Text abstraktu (česky). Obsahuje krátkou anotaci (cca 10 řádek) v češtině. Budete ji potřebovat i při vyplňování údajů o bakalářské práci ve STAGu. Český i anglický abstrakt by měly být na stejné stránce a měly by si obsahem co možná nejvíce odpovídat (samozřejmě není možný doslovný překlad!).

Obsah

1 Úvod

Generování softwarových komponent bude využívat generativního programování. Generativní programování je způsob programování, kdy je zdrojový kód programu generován na základě šablony. Tuto šablonu tvoří různé vzory.

K vytvoření šablony, pro vygenerování finálního zdrojového kódu bude v práci využíván model vlastností. Model vlastností zachycuje všechny různorodosti a podobnosti všech možných variant finálního produktu. Různé charakteristiky, které finální produkty rozlišují se nazývají právě vlastnosti.

Cílem této práce bude vygenerovat šablonu jako model vlastností ze zápisu gramatiky jazyka tesa pomocí frameworku Xtext a na základě vybraných vlastností z tohoto modelu vygenerovat finální zdrojový kód a ušetřit tak programátorům čas strávený přepisováním zdrojových kódů při změně gramatiky nebo změně konfigurace, či vytvoření nových variant konfigurace.

Čtenář je v úvodní části seznámen s problematikou modelování vlastností, nástroji které jsou pro modelování, nebo tvorbu modelů využívány. Dále bude seznámen s problematikou generativního programování a frameworkem Xtext, který je jednoduchým frameworkem pro tvorbu vlastního jazyka.

2 Modelování vlastností

Účelem této kapitoly je seznámit čtenáře s problematikou modelování vlastností, které je v práci použito jako šablona pro generování zdrojového kódu konfigurátoru a s nástroji které modelování vlastností umožňují.

2.1 Motivace

Motivaci pro modelování vlastností si ukážeme na příkladě. Představme si, že společnost vyvíjí kávovar. Kávovar bude vždy připravovat kávu, avšak jeho vlastnosti se mohou lišit na základě předem určených specifikací. Součástí specifikace může být požadavek, že kávovar bude vyvíjen pro různé trhy, například pro Evropský trh a trh v USA. Dále může být vyžadováno vytvoření dvou různých edic kávovaru, pro příklad standartní edici a deluxe edici, kde deluxe edice bude oproti standartní edici obsahovat trysku na čistou horkou vodu a displej. Za těchto předpokladů je třeba si uvědomit, že kávovar pro Americký a Evropský trh bude využívat jiný adaptér. Pro Evropský trh je potřeba klasických 220V a pro USA 120V. Zároveň je u kávovaru možno si zvolit, zda bude nebo nebude mít nastavitelné množšství kávových zrn a množství vody, ze které bude káva připravena.

Kávovar se v tomto případě nazývá produktovou řadou, produktovou rodinou, nebo také konceptem. Produktová rodina, či koncept, jsou pojmy, které označují skupinu produktů, které fungují na stejném principu, ale liší se od sebe různými vlastnostmi, tudíž je vytvořeno několik variant. Varianty je třeba spravovat. Je potřeba znázornit, které vlastnosti bude jaká varianta obsahovat, jak na sobě vlastnosti závisí, které a zda jsou potřeba. K tomu nám slouží právě model vlastností.

Tento model je tvořen procesem, který nazýváme modelování vlastností. Vytvořit takovýto model má hned několik výhod. Základní výhodou je přehlednost. Pokud zkonstruujeme správný model vlastností, je v něm na první pohled vidět, jaké produkty můžeme z vlastností sestavit. Model se poté dá použít pro nové verze stejného produktu tím, že se některé vlastnosti změní, nebo také jednoduše přidají.

Hlavní motivací je tedy identifikace a zachycení variability, znovupoužitelnost a rozšířitelnost systému či produktu.

2.2 Model vlastností

2.2.1 Vlastnosti

Každý koncept je tvořen z určitých charakteristik. Tyto charakteristiky nazýváme vlastnostmi. Pojem vlastnosti můžeme aplikovat jak na vyráběné produkty, tak na různé vlastnosti systému. Příkladem vlastností u vyráběných produktů mohou být vlastnosti znázorněné v příkladu s kávovarem. Vlastnosti jsou také užitečné při vyvíjení softwarových produktů. Požadavkem může být například kompatibilita s různými operačními systémy. Ve výsledném systému bude mít pak každá vlastnost odlišnou implementaci a na základě poskládání těchto vlastností v jeden celek získáme finální systém. Vlastnosti nám tedy zajišťují variabilitu mezi odlišnými produkty se stejným základem.

Model vlastností znázorňuje závislosti mezi těmito vlastnostmi. Vlastnosti v modelu tvoří strom, kde kořenovou vlastností je koncept. Koncept obsahuje další vlastnosti jako své potomky.

2.2.2 Typy vlastností

Jak už bylo řečeno, každý produkt u kterého je požadavek na určitou variabilitu, znovupoužitelnost či rozšiřitelnost obsahuje vlastnosti. Tyto vlastnosti mohou být několika typů. V této části se budeme věnovat základním typům těchto vlastností.

Povinné vlastnosti

Povinné vlastnosti jsou vlastnosti, které výsledný produkt musí obsahovat. Jsou to vlastnosti, na kterých je koncept založen a které jsou zahrnuty v jeho popisu. Povinná vlastnost musí být ve výsledném modelu zahrnuta, pokud je zahrnutý i její rodič. Například náš kávovar bude mít vždy mlýnek na kávu, odkapávač, zásobník zbytků, zásobník vody a další. Bez těchto vlastností se neobejde žádná varianta výsledného produktu.

Volitelné vlastnosti

Volitelné vlastnosti mohou být zahrnuty v popisu konceptu. Jinými slovy, pokud je zahrnut rodič, volitelná vlastnost může a nemusí být zahrnuta. Pokud rodič volitelné vlastnosti zahrnutý není, nemůže být zahrnuta ani volitelná vlastnost na něm závislá. V příkladě s kávovarem může být touto vlastností například zmíněné nastavitelné množství kávových zrn a množství

vody. Tuto vlastnost mít kávovar může,ale zároveň nemusí a tvoří tedy další varianty produktu.

Alternativní vlastnosti

Alternativní vlastnosti jsou vlastnosti, kde existuje možnost výběru mezi více vlastnostmi. V kávovaru je takovou vlastností edice, kdy je potřeba si vybrat mezi standart nebo deluxe edicí, ale výsledný produkt nemůže obsahovat obě. Alternativní vlastnosti mohou být volitelné, nebo povinné. Pokud je soubor alternativních vlastností povinný, je třeba si mezi nimi vybrat, ale nelze nevybrat žádnou z nich. Pokud jsou alternativní vlastnosti volitelné, je třeba mezi nimi vybrat, nebo nepoužít žádnou z nich.

2.2.3 Grafické znázornění

Modely vlastností jsou zobrazovány v diagramu vlastností. Diagram vlastností je zakreslován jako stromový graf, kde koncept je kořenovým uzlem a všechny další uzly jsou jeho vlastnostmi. Každý uzel může mít N potomků.

Povinné vlastnosti

Povinnou vlastnost v diagramu značíme jednoduchou hranou zakončenou vybarveným kruhem.

ZDE BUDE OBRÁZEK POVINNÝCH VLASTNOSTÍ

Každá instance konceptu C má vlastnost *f1* a *f2* a každá co má *f1* má *f3* a *f4*. Z toho vyplývá, že každá instance konceptu C má vlastnosti *f3* a *f4*. Můžeme tedy říct, že koncept C je popsán sadou vlastností:
{C,f1,f2,f3,f4}.

Volitelné vlastnosti

Volitelné vlastnosti v diagramu značíme jednoduchou hranou zakončenou prázdným kruhem.

ZDE BUDE OBRÁZEK VOLITELNÝCH VLASTNOSTÍ

Každá instance konceptu může mít vlastnost *f1*, vlastnost *f2*, obě, nebo žádnou. Pokud má vlastnost *f1*, může mít také vlastnosti *f3* a *f4*. Koncept můžeme popsat sadou vlastností:

- *{C}*
- *{C,f1}*
- *{C,f2}*
- *{C,f1,f2}*
- *{C,f1,f3}*
- *{C,f1,f2,f3}*

Alternativní vlastnosti

Alternativní vlastnost v diagramu značíme obloukem mezi hranami.

ZDE BUDE OBRÁZEK ALTERNATIVNÍCH VLASTNOSTÍ

V instanci jsou znázorněný povinné alternativní vlastnosti. Musíme si tedy v levé větvi vybrat mezi vlastnostmi *f1* a *f2* a na pravé větvi mezi vlastnostmi *f3*, *f4* a *f5*. Koncept můžeme popsat sadou vlastností:

- *{C,f1,f3}*
- *{C,f1,f4}*
- *{C,f1,f5}*
- *{C,f2,f3}*
- *{C,f2,f4}*
- *{C,f2,f5}*

Shrnutí

Na základě těchto základních typů dokážeme z libovolného konceptu sestavit hotový model vlastností. Díky němu jsme schopni zachytit veškeré varianty finálního produktu. Jsme schopní zjistit co jaká varianta vyžaduje a co musí obsahovat.

Diagram by krom názvů vlastností a typu závislostí měl obsahovat i informace o vlastnostech. Tyto informace by měly obsahovat důvod, proč je tato vlastnost vyžadována, jak souvisí se zbytkem modelu a veškeré další informace, které mohou být při vývoji užitečné.

3 Modelovací nástroje

V této kapitole se budeme zabývat nástroji, které umožňují modelování vlastností a které by bylo vhodné použít pro účely práce. Cílem je ukázat, jaké modelovací nástroje existují, k čemu se používají a zachytit rozsah jejich funkcí. Bude vysvětleno, jaké mají nástroje výhody a nevýhody a odůvodnění výběru nástroje.

3.0.1 pure::variants

Pure::variants je jeden z mála komerčně využívaných modelovacích nástrojů od společnosti pure-systems. Není zaměřen pouze na modelování vlastností, ale svou funkcionalitou se snaží pokrýt všechny fáze vývoje software. Samotný software je plug-inem do vývojového prostředí Eclipse. Umožňuje práci s několika modely a pro každý z těchto modelů má vlastní editor.

Hlavním modelem je *Feature Model* nebo-li model vlastností. Software zobrazuje model vastností ve stromové architektuře. Umožňuje vytvoření čtyř různých závislostí: povinné, volitelné, alternativní a slučitelné. Umožňuje také přidávat závislosti vlastností napříč modelem. Například pokud bude ve finální variantě vybrána vlastnost, software automaticky vyber povinné vlastnosti, které jsou na ní závislé a nedovolí vytvořit variantu bez těchto vlastností.

Dalšími modely jsou *Family model* nebo-li model rodiny produktů. Tento model zobrazuje elementy této rodiny a dokáže na ně namapovat vlastnosti. Dále *Variant description model*, který vlastnosti dokáže konfigurovat. Posledním modelem je *Variant result model*, který narozdíl od předchozích jako jediný nemá vlastní editor. Tento model popisuje konkrétní výstupní variantu produktu, její popis a informace pro její sestavení.

Pro naše účely budeme využívat *Feature model*, který dokáže jednoduchým způsobem sestavit výslednou variantu z vytvořených vlastností pomocí zaškrtávání.

ZDE BUDE OBRÁZEK PURE::VARIANTS

Výhody

Hlavní výhodou je uživatelsky přívětivé ovládání. Vytvoření modelu je intuitivní a jednoduché. Vlastnosti se přidávají pomocí kontextového menu a u každé nové vlastnosti je uživatel vyzván k popisu této vlastnosti. Z výsledného stromu se dá jednoduše vytvořit finální varianta a software si sám hlídá, zda jsou dodržena veškerá pravidla, která jsou v modelu zanesena.

Další velkou výhodou, která je pro tuto práci klíčová, je možnost importu a exportu modelu a výsledné varianty pomocí souborů ve formátech *CSV*, *XML*, *HTML* a *DOT*.

Nevýhody

Hlavní nevýhodou tohoto software, je dle mého názoru vysoká cena za licenci. Oficiální cena není ani na oficiálních stránkách produktu uvedena a zákazník se jí dozví přímým e-mailovým dotazem na společnost. Naše společnost zakoupila několik licencí na tento software a cena jedné licence se pohybovala okolo 10 000 eur při čemž je potřeba zaplatit 20% každý rok za prodloužení licence.

Další nevýhodou může být přílišná komplexnost nástroje. Nástroj obsahuje opravdu velké množství funkcí a plně využít jeho potenciál může být náročné.

Shrnutí

Nástroj Pure::variants je zástupcem komerčně využívaných nástrojů. Tento nástroj bude v práci využit k zachycení šablony díky jeho možnosti importu a exportu. Je pravděpodobobné, že pro účel ke kterému bude nástroj využíván by byl jiný nástroj vhodnější, ale jelikož firma ZF vlastní několik licencí a jedna z nich mi byla k účelům bakalářské práce poskytnuta, bude využit tento nástroj.

3.0.2 Xfeature

Nástroj Xfeature je dalším plug-inem do vývojového prostředí Eclipse, který poskytuje grafické uživatelské rozhraní pro práci s modely vlastností. Modely vlastností zde vyjadřují model rodiny produktů a modely aplikací. K modelu vlastností přistupují jako k meta-modelu vlastností. Model vlastností i jeho konfigurace jsou zde popsány pomocí XML dokumentu, který odpovídá určitému XML schématu (meta-modelu). Uživatel je schopen vytvořit vlastní XML schéma, které však musí odpovídat jeho meta-modelu. Nové vlastnosti

jdou tedy tvořit pouze v souladu s meta-modelem. Tvorba modelů vlastností je zde prováděna pomocí kontextového menu. Umožňuje tvorbu povinných, volitelných i alternativních vlastností.

ZDE BUDE OBRÁZEK XFEATURE

Výhody

Výhodou nástroje Xfeature je hlavně jeho přístup. Jelikož k modelům přistupuje pomocí XML, které odpovídá jistému XML schématu, je jednoduchá jeho validace. Zároveň nástroj hlídá, zda nejsou pravidla porušena. Umožňuje uživateli vytvořit si vlastní meta-model dle vlastních potřeb.

Nevýhody

Hlavní nedostatkem nástroje je odlišení grafického ztvárnění diagramů, které neodpovídá žádnému jinému ztvárnění. Tato změna může mít za následek zmatení uživatele, který je na práci s modely vlastností zvyklí a je potřeba adaptace na grafické znázornění v nástroji Xfeature. Dalším nedostatkem může být editace modelů přes kontextové menu, které může být matoucí a uživatele může značně zpomalit.

Shrnutí

Ačkoliv nástroj graficky reprezentuje vnitřně uložené struktury ve formátu XML, které by bylo možné do nástroje importovat, neumožňuje vytvoření jednotlivé varianty a neumožňuje její export. Z tohoto důvodu nemůže být pro práci využit. Může však být užitečným nástrojem při vývoji software a může mít široké využití.

3.0.3 Vlastní nástroj

Další možností využití nástrojů pro modelování vlastností je vytvořit nástroj vlastní. Jelikož je zadání velmi specifické, bylo by možné vytvořit vlastní nástroj, který bude umět pracovat se specifickými daty. Požadavky na takový nástroj by byly:

- graficky zobrazit model vlastností z gramatiky psané v Xtextu.
- umožnit vytvoření varianty konfigurace pomocí zaškrtávání vlastností

- validace vytvořené varianty
- vygenerování šablony v jazyce tesa

nástroj by mohl být opět plug-inem do vývojového prostředí Eclipse, nebo samostatnou aplikací.

Výhody

Největší výhodou vytvoření vlastního nástroje by bylo specifické řešení všech požadavků. Další výhodou by byl malý rozsah funkčí aplikace, která by nemusela umět model vlastností tvořit ani žádným jiným způsobem upravovat. Model vlastností by byl pouze grafickým znázorněním popsané gramatiky.

Nevýhody

Hlavní nevýhodou tvorby vlastního nástroje by byl čas strávený na jejím vývoji. Ačkoliv by použití bylo velmi specifické, graficky generovat diagram by nebyl jednoduchý úkol a implementace takového problému by zabralo spoustu času. Další nevýhodou by byla jeho jednoúčelovost. Takový nástroj by se dal použít pouze pro tento specifický problém a nedokázal by řešit žádné jiné scénáře.

Shrnutí

Vytvořit vlastní nástroj by bylo možné a pravděpodobně i levnější než použití nástroje pure::variants, avšak časová náročnot vývoje takového nástroje by byla nad rámec času, který je na práci k dispozici. Zároveň by vývoj takového nástroje musel procházet určitými fázemi, které by se ve společnosti mohli značně protáhnout a čas, který by byl potřeba k dokončení takového nástroje je neodhadnutelný.

4 Generativní programování

Generativní programování je metodika, která odděluje model od implementace. Realizuje se pomocí generátorů. Generátor využívá modelu jako šablony, ze které je generován výsledný kód. Generátory jsou založeny na modelech, které definují sémantiku. Pro generování jsou použity dvě základní metody - kompoziční a transformační. Generátory založené na kompozici se nazývají *kompoziční generátory* a generátory založené na transformaci se nazývají *transformační generátory*.

4.1 Kompozice a transformace

Existují dvě základní metody generování instance konceptu. Obě dvě si ukážeme na příkladě, kde budeme vytvářet instanci hvězdy z různých komponent. Hvězda je sestavena z modelu vlastností, který obsahuje:

- počet cípů
- vnitřní poloměr
- vnější poloměr
- úhel popisující naklonění prvního cípu

Nyní si ukážeme, jak k vytvoření hvězdy přistupují obě dvě metody.

ZDE BUDE OBRÁZEK CZARNECKI FIGURE 52

Kompozice

V kompozičním modelu spojíme několik komponent dohromady, které tak vytvoří požadovaný celek. K tomu, abychom byli schopni generovat různé hvězdy, je potřeba mít sadu konkrétních komponent různých velikostí a tvarů.

ZDE BUDE OBRÁZEK CZARNECKI FIGURE 55.

Kruh je popsán pouze vnitřním poloměrem, zatímco cípy jsou popsány vnitřním poloměrem, jejich počtem a vnějším poloměrem. K tomu abychom sestavili čtyřcípou hvězdu, která je vidět na obrázku, je potřeba jeden kruh a čtyři cípy vybrané ze sady konkrétních komponent, na základě jejich vlastností.

ZDE BUDE OBRÁZEK CZARNECKI FIGURE 53.

Efektivnějším způsobem sestavení instance je použití *generativních komponent* místo konkrétních komponent. Generativní komponenta využívá abstraktního popisu komponent a generuje komponentu na základě popisu. Například místo celé sady všech možných kruhů a cípů potřebujeme dvě generativní komponenty, a to generativní kruh a generativní cíp. Generativní kruh má jako parametr vnitřní poloměr a generativní cíp má jako své parametry vnitřní poloměr, vnější poloměr a úhel.

ZDE BUDE OBRÁZEK CZARNECKI FIGURE 54

Konkrétní komponenty jsou následně vygenerovány a poskládány do požadovaného obrazce.

Transformace

Narozdíl od kompozice nespojujeme jednotlivé komponenty, ale provádíme určitý počet transformací, které vyústí v požadovaný výsledek. Není potřeba mít nadefinovanou sadu komponent, nebo jejich generování, ale je třeba mít nadefinované transformace, které můžeme s instancí provádět. V tomto příkladě jsou to transformace:

- přidej 4 cípy
- zvětš vnější poloměr
- otoč o 45 stupňů

ZDE BUDE OBRÁZEK CZARNECKI FIGURE 53

Shrnutí

I když se transformační generátor zdá být jednodušší, většina generátorů, které známe, jsou kompoziční. Příkladem takového generátoru může být editor GUI, který na základě grafického editoru, ve kterém poskládáme grafické komponenty dohromady, vygeneruje kód, který je popisuje.

Kompoziční generátor bude použit v této práci. Na základě vytvořené varianty z modelu vlastností bude generován kód jazyka tesa. Komponenty zde budou zaznamenány jako záznamy v souboru formátu XML. Z těchto komponent bude následně vygenerován výsledný kód.