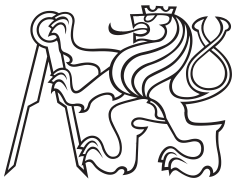
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická

Rozšíření nástroje PCTgen o import a export externích formátů

Adam Lenger

Školitel: Ing. Miroslav Bureš Ph.D.
Leden 2016

České vysoké učení technické v Praze
Fakulta elektrotechnická

katedra počítačů

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: **Adam Lenger**

Studijní program: Softwarové technologie a management
Obor: Softwarové inženýrství

Název tématu: **Rozšíření nástroje PCTgen o import a export externích formátů**

Pokyny pro vypracování:

Podle zadání školitele navrhněte a implementujte rozšiřující modul do aplikace PCTgen pro import externích formátů. Tento modul bude převádět UML Activity diagramy nebo UML Statechart diagramy vyexportované ze zvolených nástrojů na orientovaný graf a bude jej načítat do interní reprezentace aplikace PCTgen. Stejně tak bude umožňovat konverzi a export orientovaného grafu vytvořeného v aplikaci PCTgen na UML Activity diagram nebo UML Statechart diagram do zvolených externích formátů. Modul implementujte v jazyce Java a otestujte vhodnou sadou jednotkových a uživatelských testů.

Seznam odborné literatury:

Ray, E. T.: Learning XML, Second Edition. O'Reilly, 2003
Pecinovský, R.: Návrhové vzory, Computer Press, 2007

Vedoucí: Ing. Miroslav Bureš, Ph.D.

Platnost zadání: do konce zimního semestru 2016/2017

doc. Ing. Filip Železný, Ph.D.
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 8. 10. 2015

Poděkování

Děkuji vedoucímu práce za trpělivost, rady a odborné vedení. Svými názory a znalostmi mi velmi pomohl při tvorbě této práce.

Dále bych rád také poděkoval celé svojí rodině a kamarádům, kteří mě podporovali a dodávali vše, co jsem potřeboval.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne 11. 1. 2016

Abstrakt

Cílem této práce bylo vytvořit nástroj pro import diagramů vytvořených v programu Enterprise Architect do programu PCTgen. Daný nástroj umožní uživatelům těchto aplikací jednoduše generovat testovací scénáře z již hotových workflow.

Byl vytvořen návrh, který byl následně implementován a otestován pomocí testovacích scénářů. Implementace byla provedena v jazyce Java. Pro konverzi diagramů na graf byl vytvořen algoritmus, jelikož tato konverze je ztrátová.

Nástroj byl vyvinut jako doplněk do stávající aplikace PCTgen a umožňuje import přímo z jejího rozhraní. Umožňuje konverzi diagramu na graf za pomoci přepínačů, které určí, jakým způsobem se konverze provede. Uživatel tedy sám zvolí, jaký chce mít výsledný graf, se kterým bude dále pracovat.

Klíčová slova: Enterprise Architect, PCTgen, Java, diagram, graf, import

Školitel: Ing. Miroslav Bureš Ph.D.

Abstract

The goal of this thesis was to create a tool for import of diagrams created in program Enterprise Architect into program PCTgen. This tool will enable generation of test cases from completed workflow for users of these two applications.

It was created design, which was then implemented and tested by set of test cases. Implementation was done in Java language. For conversion of diagrams to graph was created algorithm because this conversion is loss.

Tool was developed as add-in into curren application PCTgen and enables import directly from its interface. It enables conversion of diagram to graph by set of switches, which sets how the conversion is done. Therefore user sets by himself/herself how the final graph will look like.

Keywords: Enterprise Architect, PCTgen, Java, diagram, graph, import

Title translation: Extension of PCTgen tool by import and export of external data formats

Obsah

Obrázky

Kapitola 1
Úvod

V dnešní době, kdy se klade velký důraz na vývoj softwaru, se občas opomíjí či zapomíná na testování. Z toho pak ovšem vyplývá větší chybovost a horší udržitelnost aplikace. Vzniká tedy potřeba vytvářet testy a testovací scénáře, které by zmiňované vlastnosti zlepšovaly. Ovšem je potřeba toto vytváření testů zefektivnit a zautomatizovat. Kvůli tomu byl vyvinut nástroj PCTgen, který z aplikačních workflow umožňuje vytvářet testovací scénáře.[1]

Tento nástroj je již vyvinut a standardně používán. Práce s ním je relativně jednoduchá a intuitivní. Pro znalého člověka tedy není problém vytvořit aplikační workflow a poté nechat nástroj vygenerovat testovací scénář. Stále je ovšem nutno vytvářet aplikační workflow. Pro zjednodušení je zde i možnost importu ze souborů XML a CSV. Pro vytváření diagramů aktivit se používá řada komerčních nástrojů, jedním z nich je i Enterprise Architect. Objevuje se tedy varianta, kdy z Enterprise Architectu předám workflow do PCTgenu a ten automaticky vygeneruje testovací scénáře. A přesně vývojem tohoto doplňku se zabývá má bakalářská práce. Z Enterprise Architect se dají vyexportovat již hotové diagramy a projekty do souboru typu XML. Doplněk se tedy bude věnovat importu diagramů do PCTgen přes tento soubor.

Jelikož je konverze ztrátová (některé konstrukty UML nejdou jednoduše zobrazit v grafu), vyvstává potřeba řešit, jakým způsobem se zachovat v klíčových situacích. Tyto situace je nutné zanalyzovat a navrhnout vhodné řešení. Výstup pro uživatele musí být srozumitelný a jednoduše pochopitelný. Jako vhodné se jeví nabídnutí přepínačů a možností, aby si uživatel mohl vybrat variantu, která vyhovuje jeho požadavkům.

Doplněk nejprve vezme soubor vyexportovaný z Enterprise Architect. Poté zpracuje v něm uložené diagramy a nabídne uživateli, jaké diagramy a jakým způsobem chce importovat. Následně budou diagramy zkonvertovány na grafy používané v PCTgen a naimportovány do projektu.

Po dohodě se školitelem jsme se rozhodli neimplementovat tuto část zadání: "export orientovaného grafu vytvořeného v aplikaci PCTgen na UML Activity diagram nebo UML Statechart diagram do zvolených externích formátů". Zjistili jsme totiž, že tato podúloha nejde díky chybějícím datům implementovat.

Kapitola 2
Analýza a návrh

V této kapitole se věnuji analýze a návrhu mého doplňku. Představím oba nástroje, mezi kterými probíhá import, poté požadavky a typický uživatelský scénář. Dále se již v kapitole zaobírám návrhem procesů a architektury.

2.1 PCTgen

PCTgen je freeware nástroj pro automatickou generaci testovacích scénářů. Byl vyvinut na Fakultě Elektrotechnické ČVUT. Tým z katedry počítačů si uvědomil, že jim současné nástroje nevyhovují, a rozhodli se ho sami vytvořit. Program umožňuje vytvářet testovací scénáře z aplikačních workflow. Tato workflow jsou reprezentována orientovanými grafy pro lepší čitelnost a abstrakci.

Před samotným generováním testovacích scénářů je možné zvolit několik různých nastavení jako Test Level Depth (nastavení hloubky testování) či redukci cyklů. Díky importu a exportu z a do XML a CSV je možné propojení PCTgen s dalšími nástroji pro práci týkající se testování.

Grafy je kromě importu samozřejmě možné vytvářet a upravovat i ručně. Slouží pro to jednoduché grafické prostředí, kde si uživatel přidá začátek grafu a další uzly. Ty pak tahem myši jednoduše pospojuje hranami do požadovaného orientovaného grafu. Každý uzel a hranu lze pojmenovat, případně k nim přiřadit popis.

2.2 Enterprise Architect

Na rozdíl od malého PCTgen nástroje je Enterprise Architect komplexní software určený pro systémovou analýzu a návrh. Pokrývá celý životní celek vývoje systému, tedy zadání, analýzu, návrh, testování a údržbu. Používá k tomu UML diagramy. Může ho tedy používat celý tým od analytiků až po testery.

Enterprise Architect byl vydán již v roce 2000. Od té doby prošel rozsáhlým vývojem, kdy se přizpůsoboval aktuálním požadavkům a novým verzím UML. V současné verzi podporuje standard UML 2.5.[7]

Jako student ČVUT mám k dispozici Enterprise Architect ve verzi 11, přičemž nejnovější je verze 12. Ve verzi 12 proběhly velké úpravy designu, ale i změny ve funkčnosti softwaru. Podle dohledatelných informací by ale neměl být změněn export do XML souboru. Pokud by byl změněn, bude nutné poté upravit i mnou vytvořený doplněk. Pro vývoj své aplikace tedy použiji verzi 11.

2.3 Požadavky na implementovaný modul

2.3.1 Funkcionální požadavky

- Import UML diagramu aktivit z Enterprise Architect.
- Import UML stavového diagramu z Enterprise Architect.
- Možnost zvolit zachování koncových uzlů.
- Možnost určit typ převodu paralelismu.
- Možnost zvolit jeden či více diagramů k importu najednou.
- Zabudování doplňku do PCTgen.

2.3.2 Nefunkcionální požadavky

- Stabilita - program zvládne i neočekávané situace a bude s ním možno dále pracovat.
- Doplněk bude napsán v jazyce Java, aby ho bylo možné zakomponovat do aplikace.
- Doplněk musí být jednoduchý a přehledný pro uživatele.
- Konverzní pravidla budou provedena formou přepínačů a výběrových polí.
- Funkčnost musí být dostatečně rychlá, aby uživatel mohl pohodlně pracovat s aplikací.

2.4 Typický uživatelský scénář

Uživatel nejprve vyexportuje hotový Activity Diagram z Enterprise Architect do XML souboru. Poté otevře PCTgen a zde zvolí v horním menu Export/Import, z možné nabídky vybere Import z EA. Otevře se okno pro zvolení vygenerovaného souboru. Po vybrání se zobrazí seznam diagramů, ze kterých si uživatel vybere požadované diagramy pro import. Zároveň zde zvolí zachovávání koncových uzlů a typ převodu paralelismu. Po potvrzení se zvolené diagramy naimportují do PCTgen.

2.5 Návrh procesu pro import diagramu

2.5.1 Zpracování XMI souboru - Exportu z Enterprise Architect

Enterprise Architect umí vygenerovat XMI soubor buď z celého projektu, nebo pouze z vybraných částí projektu. Více informací k tomuto souboru se nachází v kapitole 2.8.1. Struktura souboru je v obou případech podobná, v prvním obsahuje několik informací (uzlů) navíc, které ovšem pro přenos nebudou potřeba. Informace o jednom diagramu jsou uloženy v jednom uzlu a jeho potomcích.

Pro zpracování bude tedy použit standardní SAXParser, který s dokumentem pracuje postupně a bude stejně přistupovat k uzlům.

2.5.2 Načtení do vlastní struktury

Pro uchování a další zpracování grafu bude vytvořena nová samostatná struktura - graf. Bude se skládat z grafu samotného, jeho uzlů a hran. Každá z těchto tří komponent (graf, uzel, hrana) si bude u sebe uchovávat potřebné informace a případně obsahovat potřebnou funkčnost.

2.5.3 Konverzní pravidla

Jelikož PCTgen pracuje pouze s jednoduchými grafy, je potřeba konvertovat složitější konstrukce v diagramech. Z hlediska programu jsou zde dva hlavní body ohledně konverzních pravidel, a to jak se chovat ke koncovým uzlům a jakým způsobem zpracovat paralelismus.

U koncových uzlů bude možné je ponechat v samotném grafu či je vynechat. Stačí tedy mít koncové uzly označené a případně je vynechat. Implementace této funkčnosti je relativně jednodušší.

2.5.4 Převedení paralelismů

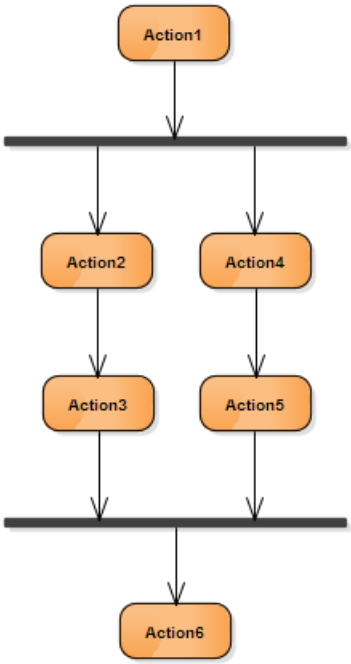
Jedním z nejtěžších problémů, který je potřeba vyřešit, je konverze paralelismů podle volby přepínače. Přepínače jsou nutné, jelikož není možné udělat bezeztrátovou konverzi z diagramu v Enterprise Architect do PCTgen. PCTgen má totiž pouze jednoduchou grafovou reprezentaci pomocí uzlů a hran. Nerozděluje uzly na další typy, a tím pádem nelze převést bezeztrátově paralelismus.

První krok je určit, jaké přepínače bude mít uživatel k dispozici. Několik variant vyplývá přímo z dané struktury, např. převedení začátku a konce paralelismu na uzel. Další varianty byly předmětem diskuze a zjišťování požadavků uživatelů aplikace. Nabízelo se jich větší množství, z nich nakonec vyplynuly hlavní tři, které byly nejvíce požadované a nejpřínosnější:

- Převedení začátku a konce paralelismu na uzel
- Převedení celého paralelismu na jeden uzel

- Serializace jednotlivých proudů paralelismu

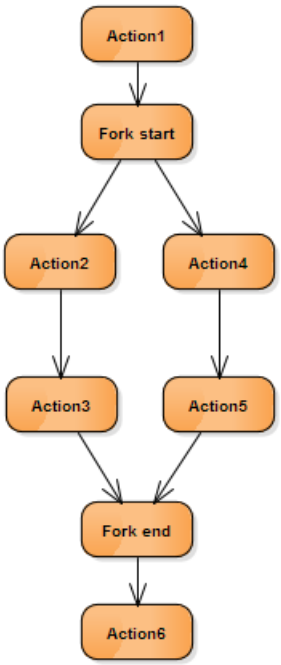
Po určení těchto možností je potřeba vymyslet, jakým způsobem se bude každá z nich implementovat. Způsoby budou interpretovány i obrázky již převedeného paralelismu. Všechny budou vycházet z tohoto základního paralelismu:



Obrázek 2.1: Základní paralelismus

Převedení začátku a konce paralelismu na uzel

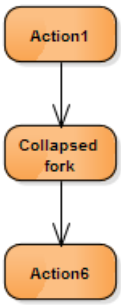
Tato možnost je ze všech nejjednodušší. Enterprise Architect totiž ve vygenerovaném XML drží začátek i konec paralelismu jako uzel s nastaveným atributem paralelismu. Není tedy potřeba nějakých speciálních úprav pro převedení začátku a konce paralelismu na uzel.



Obrázek 2.2: Převedení začátku a konce paralelismu na uzel

Převedení celého paralelismu na jeden uzel

Tento úkol se jeví jako nejtěžší. Uzly, které se vyskytují v paralelismu, totiž nejsou v XML souboru nijak zvláštně označeny. To velmi výrazně zhoršuje možnosti jak rychle a efektivně zkonvertovat paralelismus na jeden uzel. Je tedy potřeba nejprve sestavit graf a v něm si držet označené začátky a konce paralelismu a až poté provádět konverzi.



Obrázek 2.3: Převedení celého paralelismu na jeden uzel

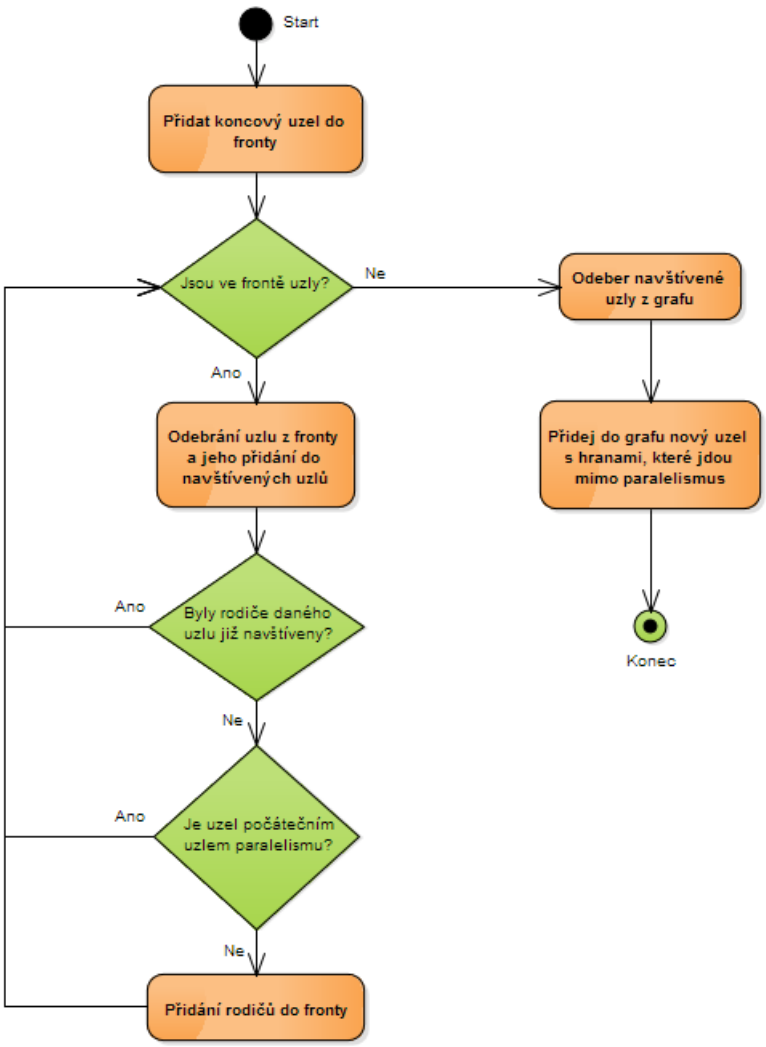
Vzhledem k relativně specifickým podmínkám je potřeba vymyslet vlastní algoritmus. Prvním úkolem je najít všechny uzly, které se nacházejí uvnitř

paralelismu. Dále je potřeba zachovat všechny hrany, které by mohly vést mimo paralelismus. A samozřejmě je také nutné nově vytvořený uzel správně připojit do vytvořeného grafu.

Algoritmus prochází graf postupně od všech koncových uzlů směrem k počátečním uzlům. Pokud narazí na uzel, který je koncem paralelismu, je spuštěno převedení jednoho paralelismu na jeden uzel.

Koncový uzel paralelismu je zapamatován a pokračuje se dále, dokud se nenarazí na počáteční uzel paralelismu. Uzly, které jsou rámci toho navštíveny, jsou zapamatovány i s jejich hranami. To je nutné pro budoucí odstranění těchto uzlů z původního grafu a jejich nahrazení jedním uzlem. Zároveň se v průběhu kontroluje, zda některý z uzlů není koncem paralelismu. Pokud ano, je třeba s tím počítat, jelikož další počáteční uzel paralelismu není hledaný počátek paralelismu. Tímto jsou vyřešeny vnořené paralelismy. Pokud se došlo na hledaný počáteční uzel, je tento zapamatován a dále za něj se již nepokračuje. Nechají se doběhnout zbylé proudy, které ještě nedošly k tomuto počátečnímu uzlu. Poté, co všechny proudy doběhly, máme nalezeny všechny uzly, které jsou uvnitř paralelismu. Uzly odstraníme z původního grafu a zachováme pouze hrany, které jdou mimo tento paralelismus. Tím se docílí správného napojení do grafu a zachování případných hran, které vystoupily z paralelismu. Zjednodušené blokové schéma vnitřního algoritmu můžete vidět na obrázku 2.4.

Po převedení jednoho paralelismu na uzel se pokračuje dále v původním algoritmu, dokud se nenarazí na počáteční uzel grafu. Pokud algoritmus již doběhl ze všech koncových uzlů, tak skončí.

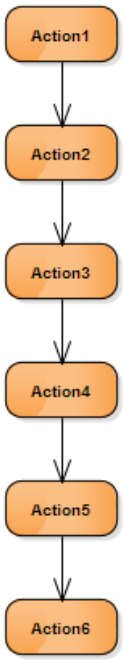


Obrázek 2.4: Zjednodušené schéma algoritmu

Serializace jednotlivých proudů paralelismu

Serializace funguje na podobném principu jako převedení paralelismu na jeden uzel. Hlavní kostra algoritmu je stejná, rozdíl je v tom, když se narazí na koncový uzel paralelismu.

Od koncového uzlu paralelismu se opět hledá počáteční uzel paralelismu. Pokud je nalezen, zastavíme další prohledávání paralelismu. Máme tím totiž nalezeny počátky a konce jednotlivých proudů paralelismu, jelikož ty jsou napojeny na počáteční a koncový uzel. Tyto počátky a konce na sebe napojíme. Jediné co je potřeba hlídat, je, aby se nenapojil počátek jednoho proudu na konec toho samého. Tím docílíme serializaci jednotlivých proudů v paralelismu.



Obrázek 2.5: Serializace jednotlivých proudů paralelismu

2.6 Architektura modulu

Aplikace se skládá z těchto hlavních částí:

- Modul struktury grafu

  Slouží pro uložení grafu, uzlů a hran. Drží si jejich informace a případně potřebnou funkčnost.

- Modul uživatelského rozhraní

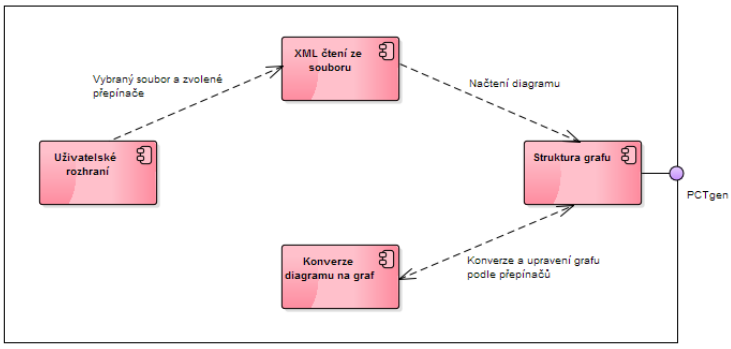
  Zobrazuje uživateli potřebná okna, pomocí kterých vybere soubor a zvolí možnosti. Zároveň tyto informace předává dále.

- Modul pro čtení XML souboru

  Zajišťuje čtení XML souboru a předání dat do struktury grafu a uživatelského rozhraní.

- Modul konverze diagramu na graf

  Převádí načtený diagram na graf použitelný v PCTgen.

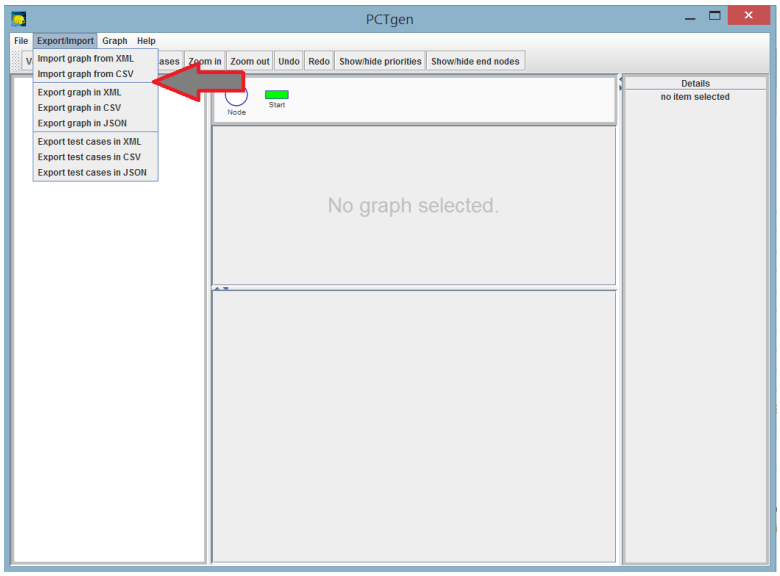


Obrázek 2.6: Návrh blokového schématu aplikace

2.7 Návrh uživatelského rozhraní modulu

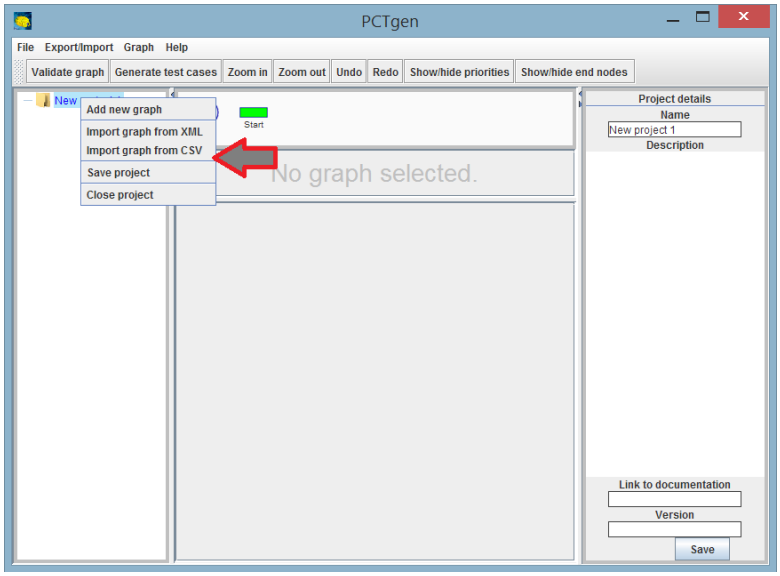
2.7.1 Zakomponování importu do PCTgen

Současné uživatelské rozhraní aplikace PCTgen je celkem jednoduché. V podobném duchu tedy bylo navrhnuto i zakomponování importu souboru z Enterprise Architect. Položka "Import graph from EA" bude přidána do menu "Export/Import" k dalším položkám importování.



Obrázek 2.7: Umístění možnosti importu z Enterprise Architect

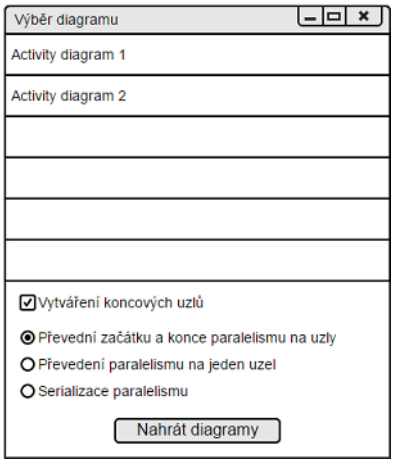
Zároveň je import možný i z kontextové nabídky po pravém kliknutí na projekt. Položka "Import graph from EA"tedy bude přidána i do tohoto kontextového menu.



Obrázek 2.8: Umístění možnosti importu z Enterprise Architect - kontextová nabídka

2.7.2 Okno pro import diagramů

Okno pro výběr diagramů bude také jednoduché. Bude se na něm nacházet samozřejmě výběr diagramů pro nahrání a dále přepínače pro uživatelskou volbu. První přepínač bude pro zachování či vynechání koncových uzlů. Další přepínač bude výběr jak převést paralelismus na graf, který dokáže PCTgen zpracovat.



Obrázek 2.9: Okno pro import diagramů

2.8 Zpracování a uložení dat v aplikaci

Aplikace musí být schopná zpracovat XML soubor z Enterprise Architect, vhodně si reprezentovat data vnitřně a pak je předat ve správném formátu

do PCTgen.

2.8.1 Exportní formát z aplikace Enterprise Architect

Enterprise Architect pro export používá soubor typu XMI. Zkratka XMI zastupuje "XML Metadata Interchange"[8]. Více konkrétně zahrnuje čtyři standardy:

- XML - Extensible Markup Language
- UML - Unified Modeling Language
- MOF - Meta Object Facility
- MOF - Mapování na XMI

Jedná se o standard pro výměnu metadat informací prostřednictvím XML souboru. Nejčastěji se používá pro výměnu UML modelů, ale může se použít také pro serializaci modelů z jiných metamodelů.

V souboru se tedy nachází mnoho informací - informace o samotném diagramu, které jsou pro nás podstatné, ale také informace potřebné pouze pro Enterprise Architect, kterých je bohužel většina.

2.8.2 Vnitřní reprezentace dat

Uvnitř aplikace bude vhodné si držet data jako standardní graf. Budeme tedy držet informace o grafu samotném, uzlech a hranách. Pro každý z těchto tří prvků bude vytvořena samostatná třída. V této třídě se budou držet informace o části grafu (či grafu samotném), jeho vazby na ostatní části grafu a také případné metody upravující tu danou komponentu.

2.8.3 Formát pro předání dat do PCTgen

PCTgen pro reprezentaci dat používá vlastní strukturu. Ta využívá knihovnu jgraphx a její objekt mxgraph. Tento objekt rozšiřuje a přidává k němu další doplňující informace potřebné pro správný běh programu. Aplikace tudíž musí převést data do této struktury, aby se poté správně reprezentovala v PCTgen.

Kapitola 3
Implementace

V kapitole Implementace vám představím, jaké prostředky jsem pro ni použil. Také ukáži, jakým způsobem jsem zpracoval důležité části a PCTgen s přidaným doplňkem.

3.1 Zvolený jazyk a knihovny

Vzhledem k tomu, že aplikace PCTgen je napsaná v programovacím jazyku Java, byl tento jazyk zvolen i k implementaci mého rozšíření. Použil jsem také jgraphx knihovnu, která je použita v PCTgen. Závěrečná integrace do mého doplňku do PCTgen tedy nevyžadovala import dalších knihoven.

3.2 Důležité části implementace

3.2.1 Výběr rozhraní pro čtení XML

Na počátku jsem se rozhodoval mezi rozhraními DOM a SAX. Po analýze XML souboru generovaného z Enterprise Architect jsem se rozhodl pro rozhraní SAX. Tato zkratka stojí za "Simple API for XML". Na rozdíl od DOM (Document Object Model), které si nahraje celý dokument do paměti a pak s ním pracuje na principu dotazování, SAX postupně prochází dokument a komunikuje pomocí událostí. Tyto události jsou vyvolány např. na začátku a konci elementu či dokumentu. S těmito událostmi jsou předány i další parametry jako atributy či názvy elementů.

V metodě, která reaguje na začátek elementu, tedy zjišťuji, o jaký element se jedná a zda je podstatný pro vytvoření grafu. Pokud ano, jsou předány všechny podstatné informace do metody určené pro zpracování toho daného prvku. Tímto způsobem je tedy postupně vytvořen graf a do něho přidány uzly a hrany se všemi potřebnými informacemi.

V průběhu implementace vznikl požadavek na možnost vybrání více diagramů a ne pouze jednoho. Pro nalezení všech diagramů v XML souboru se ukázalo vhodnější použít rozhraní DOM. Hodí se lépe pro nalezení konkrétních dat a vytažení pouze potřebných informací a ne pro postupnou práci s prvky.[2]

Pro hlavní část programu tedy bylo zvoleno rozhraní SAX a pro doplňkovou funkci rozhraní DOM. Obě dvě rozhraní se dobře hodí na zvolené části.

3.2.2 Prvky grafického uživatelského rozhraní

Pro uživatele bylo nutné vytvořit okno pro výběr souboru a pak okno pro výběr diagramů s přepínači. Vzhledem k použitým základním Java technologiím v PCTgen jsem pro tyto komponenty nehledal speciální grafické prvky a to z toho důvodu, aby dobře zapadly do aplikace. Pro výběr souboru jsem zvolil JFileChooser ze Swing knihovny.[6] Pro zvolení diagramů a přepínače jsem vybral standardní JPanel, JCheckbox a JRadioButton. Tedy vše základní Java komponenty, nicméně plně dostačující pro fungující aplikaci. Navíc vhodně korespondují se současným vzhledem PCTgen.

3.2.3 Struktura balíků v aplikaci

Jelikož je moje aplikace integrována do PCTgen, volil jsem i podobnou strukturu, aby závěrečné spojení bylo jednoduché. Zde je seznam balíků, kam jsem umístil svou část implementace:

- *main.structures*

  Zde jsem umístil svoji strukturu grafu.

- *main.utils*

  Do tohoto balíku je přidána potřebná pomocná třída a třída pro načtení XML souboru.

- *main.gui*

  V balíku *main.gui* se nachází okno pro výběr diagramu.

- *main.controllers*

  Do třídy *MainGuiPresenter* jsem přidal okno pro výběr souboru. Tato funkcionalita je umístěna zde, aby korespondovala s dalšími podobnými komponentami. Zároveň jsem využil již naimplementované funkčnosti ve třídě *GuiUtils*, která má metodu pro vytvoření okna pro výběr souboru.

3.2.4 Seznam hlavních tříd

V této podkapitole více rozeberu jednotlivé třídy, které jsem implementoval v rámci svého doplňku.

- *EAUtility*

  Zajišťuje komunikaci mezi strukturou grafu a čtením XML souboru. Zároveň poskytuje metody pro poskytnutí grafu do aplikace.

- *EASaxHandler*

  Vlastní implementace výchozího SAX Handler. Slouží k načtení diagramu z XML souboru. Jako parametr přijímá cestu k souboru.

- *EAGraph*

  Drží informace o grafu (kořen, uzly a jeho hrany). Také nabízí metody k úpravě načteného grafu.

- *EANode*

  Reprezentuje uzel grafu. Ten si u sebe drží všechny příchozí a odchozí hrany. Uvádí, zda se jedná o koncový uzel, či začátek nebo konec paralelismu.

- *EAEdge*

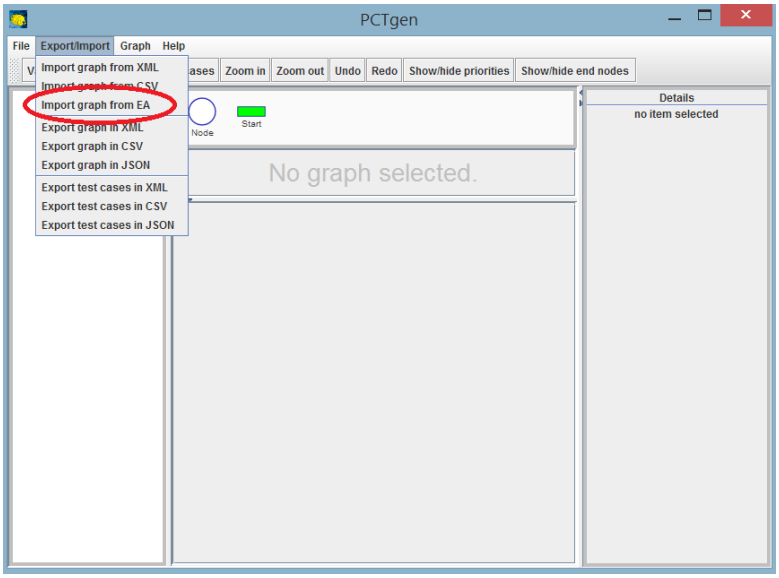
  Reprezentuje hranu grafu. Má u sebe uložen zdrojový uzel, z kterého vychází, a cílový uzel, do něhož směřuje.

- *DiagramList*

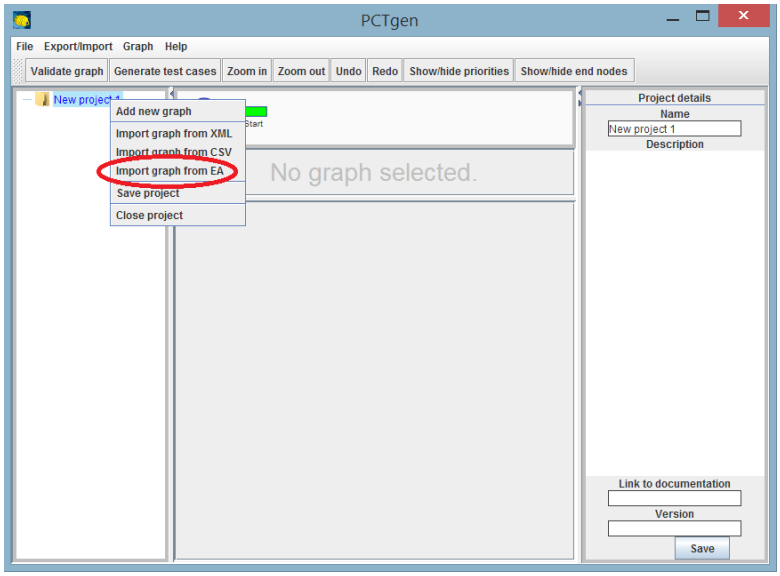
  Používá se k zobrazení dialogového okna pro výběr diagramů a zvolení přepínačů.

3.3 Ukázky obrazovek PCTgen s přidanými komponentami

Přidané položky do Import/Export menu a kontextového menu projektu.

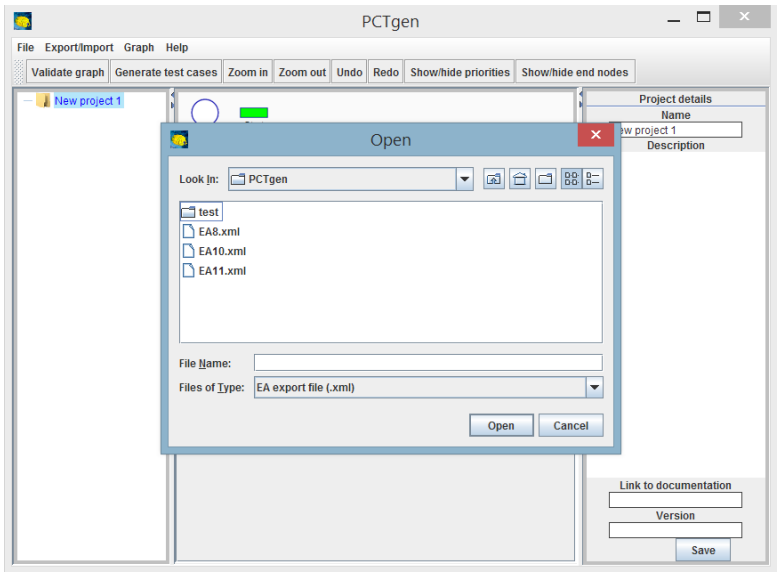


Obrázek 3.1: Umístění možnosti importu z Enterprise Architect - nově



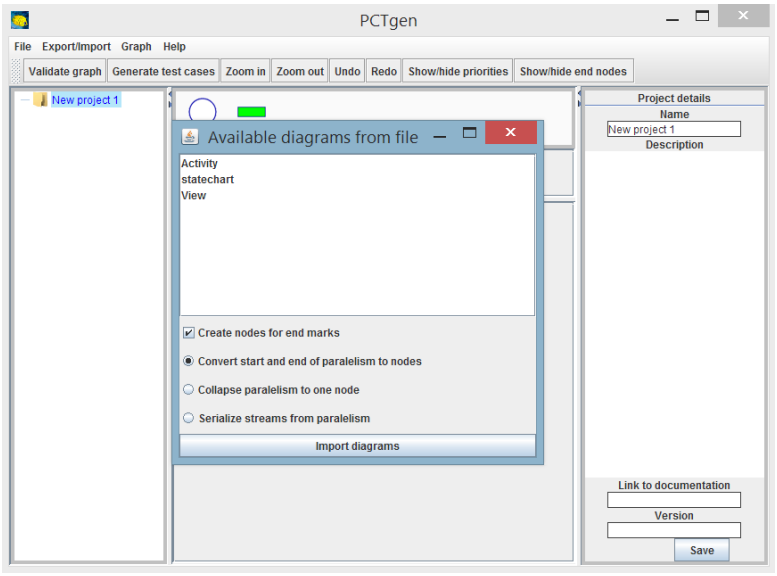
Obrázek 3.2: Umístění možnosti importu z Enterprise Architect - kontextové menu - nově

Okno pro výběr XML souboru z Enterprise Architect.



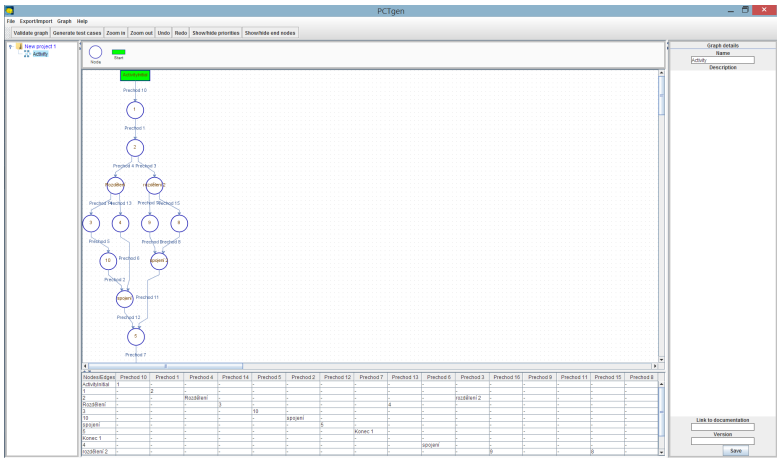
Obrázek 3.3: Okno pro vybrání souboru vyexportovaného z Enterprise Architect

Okno pro výběr diagramu, zvolení přepínačů a import diagramů.



Obrázek 3.4: Okno s diagramy

Aplikace s importovaným diagramem.



Obrázek 3.5: Aplikace s importovaným diagramem

Kapitola 4

Testovací scénáře

V této kapitole jsou testovací scénáře sloužící k otestování aplikace. Pro všechny testovací scénáře je nutné mít v PCTgen vytvořený projekt, do kterého se diagramy budou importovat. V případě, že projekt není vytvořen, je po zvolení možnosti *Import graph from EA* zobrazena chybová hláška *In the project tree (left panel), select a project to which you would like to import the graph.* Zároveň se v každém testu nejprve provádí export diagramu do XML v Enterprise Architect. Kroky pro export:

1. Pravý klik na model, v němž se diagram nachází.
2. Vybrání položky *Export Model to XMI...*
3. V poli *filename* nově vytvořený soubor a určit cestu, kam se uloží. Export type nechat nastaven na XMI 2.1.
4. Kliknout na tlačítko *Export*.
5. Exportovaný soubor je uložen na určené cestě.

4.1 Testovací scénář č. 1 - Jednoduchý activity diagram

Výchozí podmínky:

- Vytvořený activity diagram bez paralelismů v Enterprise Architect.

Kroky testu:

1. Export modelu z Enterprise Architect.
2. V aplikaci PCTgen v položce *Export/Import* vybrat možnost *Import graph from EA*.
3. Vybrat XML soubor s vyexportovaným modelem.
4. Vybrat námi vytvořený activity diagram. Nechat zaškrtnutou položku *Create nodes for end marks* a nechat vybranou položku *Convert start and end of paralelism to nodes*.

5. Kliknout na tlačítko *Import diagrams*.

Očekávaný výsledek: Naimportovaný graf v aplikaci PCTgen, který koresponduje s vytvořeným activity diagramem z Enterprise Architect.

4.2 Testovací scénář č. 2 - Jednoduchý activity diagram - Vynechání koncových uzlů

Výchozí podmínky:

- Vytvořený activity diagram bez paralelismů v Enterprise Architect.

Kroky testu:

1. Export modelu z Enterprise Architect.
2. V aplikaci PCTgen v položce *Export/Import* vybrat možnost *Import graph from EA*.
3. Vybrat XML soubor s vyexportovaným modelem.
4. Vybrat námi vytvořený activity diagram. Položku *Create nodes for end marks* změnit na nezaškrtnutou a nechat vybranou položku *Convert start and end of paralelism to nodes*.
5. Kliknout na tlačítko *Import diagrams*.

Očekávaný výsledek: Naimportovaný graf v aplikaci PCTgen, který koresponduje s vytvořeným activity diagramem z Enterprise Architect. Koncové uzly se v grafu nevyskytují.

4.3 Testovací scénář č. 3 - Activity diagram s paralelismem - Převod na uzly

Výchozí podmínky:

- Vytvořený activity diagram s paralelismem v Enterprise Architect.

Kroky testu:

1. Export modelu z Enterprise Architect.
2. V aplikaci PCTgen v položce *Export/Import* vybrat možnost *Import graph from EA*.
3. Vybrat XML soubor s vyexportovaným modelem.
4. Vybrat námi vytvořený activity diagram. Nechat zaškrtnutou položku *Create nodes for end marks* a nechat vybranou položku *Convert start and end of paralelism to nodes*.

5. Kliknout na tlačítko *Import diagrams*.

Očekávaný výsledek: Naimportovaný graf v aplikaci PCTgen, který koresponduje s vytvořeným activity diagramem z Enterprise Architect. Začátek a konec paralelismu jsou převedeny na uzly.

4.4 Testovací scénář č. 4 - Activity diagram s paralelismem - Převedení celého paralelismu na jeden uzel

Výchozí podmínky:

- Vytvořený activity diagram s paralelismem v Enterprise Architect.

Kroky testu:

1. Export modelu z Enterprise Architect.
2. V aplikaci PCTgen v položce *Export/Import* vybrat možnost *Import graph from EA*.
3. Vybrat XML soubor s vyexportovaným modelem.
4. Vybrat námi vytvořený activity diagram. Nechat zaškrtnutou položku *Create nodes for end marks* a vybrat položku *Collapse paralelism to one node*.
5. Kliknout na tlačítko *Import diagrams*.

Očekávaný výsledek: Naimportovaný graf v aplikaci PCTgen, který koresponduje s vytvořeným activity diagramem z Enterprise Architect. Celý paralelismus je převeden na jeden uzel.

4.5 Testovací scénář č. 5 - Activity diagram s paralelismem - Serializace paralelismu

Výchozí podmínky:

- Vytvořený activity diagram s paralelismem v Enterprise Architect.

Kroky testu:

1. Export modelu z Enterprise Architect.
2. V aplikaci PCTgen v položce *Export/Import* vybrat možnost *Import graph from EA*.
3. Vybrat XML soubor s vyexportovaným modelem.

4. Vybrat námi vytvořený activity diagram. Nechat zaškrtnutou položku *Create nodes for end marks* a vybrat položku *Serialize streams from paralelism*.
5. Kliknout na tlačítko *Import diagrams*.

Očekávaný výsledek: Naimportovaný graf v aplikaci PCTgen, který koresponduje s vytvořeným activity diagramem z Enterprise Architect. Jednotlivé proudy v paralelismu jsou serializovány za sebe v nově importovaném grafu.

4.6 Testovací scénář č. 6 - Více activity diagramů

Výchozí podmínky:

- Vytvořený model s více activity diagramy v Enterprise Architect.

Kroky testu:

1. Export modelu z Enterprise Architect.
2. V aplikaci PCTgen v položce *Export/Import* vybrat možnost *Import graph from EA*.
3. Vybrat XML soubor s vyexportovaným modelem.
4. Vybrat námi vytvořené activity diagramy (větší množství než jeden). Nechat zaškrtnutou položku *Create nodes for end marks* a nechat vybranou položku *Convert start and end of paralelism to nodes*.
5. Kliknout na tlačítko *Import diagrams*.

4.7 Testovací scénář č. 7 - Stavový diagram

Výchozí podmínky:

- Vytvořený model se stavovým diagramem v Enterprise Architect.

Kroky testu:

1. Export modelu z Enterprise Architect.
2. V aplikaci PCTgen v položce *Export/Import* vybrat možnost *Import graph from EA*.
3. Vybrat XML soubor s vyexportovaným modelem.
4. Vybrat námi vytvořený stavový diagram. Nechat zaškrtnutou položku *Create nodes for end marks* a nechat vybranou položku *Convert start and end of paralelism to nodes*.
5. Kliknout na tlačítko *Import diagrams*.

Očekávaný výsledek: Naimportovaný graf v aplikaci PCTgen, který koresponduje s vytvořeným stavovým diagramem z Enterprise Architect.

4.8 Testovací scénář č. 8 - XML bez diagramu

Výchozí podmínky:

- Vytvořený model bez diagramů v Enterprise Architect.

Kroky testu:

1. Export modelu z Enterprise Architect.
2. V aplikaci PCTgen v položce *Export/Import* vybrat možnost *Import graph from EA*.
3. Vybrat XML soubor s vyexportovaným modelem.

Očekávaný výsledek: PCTgen ohlásí, že v souboru se nenachází žádné diagramy.

4.9 Výsledky testů

Všechny scénáře byly postupně vyzkoušeny a použity k otestování správné funkčnosti programu a nalezení chyb. V případě nalezení chyby se upravila funkčnost pro správné chování podle scénáře. Poté proběhlo opětovné testování i již funkčních scénářů, zda nedošlo k zanesení nové chyby.

Kapitola 5

Závěr

Cílem této práce bylo vytvořit doplněk do aplikace PCTgen pro import diagramů z programu Enterprise Architect. Tyto diagramy bylo nutno převést na grafy, se kterými PCTgen pracuje. Jelikož je tato konverze ztrátová, bylo nutné vymyslet a nabídnout uživateli přepínače, pomocí kterých může konverzi ovlivnit. Všechny tyto body byly splněny a implementovány a nyní může uživatel importovat diagram vytvořený v Enterpise Architect do PCTgen.

V rámci implementace se řešilo hned několik problémů. Prvním byla volba přepínačů pro uživatele. Bylo potřeba zvolit ty nejvíce vyhovující potřebám uživatelů. Vybrané změny pak byly implementovány. Je možné, že se v budoucím používání aplikace vyskytne potřeba pro změnu či přidání více přepínačů. Další úloha, kterou bylo potřeba vyřešit, byla konverze paralelismů z diagramu. Protože je tato konverze specifická, bylo nutné vymyslet vhodný algoritmus, který různé typy konverze provede. Doplněk navíc umožňuje importovat více diagramů z jednoho modelu najednou.

Pokud by v budoucnosti bylo potřeba importovat grafy z dalších programů schopných své diagramy exportovat do XML, je určitě možné využít již hotové implementace z této práce. Je zde přidána nová generická struktura grafu a nástroje, které získávají tento graf z XML souboru. Bude tedy potřeba několika málo úprav a existuje možnost importovat grafy i z jiných programů než Enterprise Architect.

Po dohodě se školitelem jsme se rozhodli neimplementovat tuto část zadání: "export orientovaného grafu vytvořeného v aplikaci PCTgen na UML Activity diagram nebo UML Statechart diagram do zvolených externích formátů". Zjistili jsme totiž, že tato podúloha nejde díky chybějícím datům implementovat.

Literatura

[1] Bureš, M. "PCTgen: Automated Generation of Test Cases for Application Workflows". In: *New Contributions in Information Systems and Technologies, Advances in Intelligent Systems and Computing.* vol.353, Springer, 2015, s. 789-794.

[2] Ray, E. T.: Learning XML, Second Edition. O'Reilly, 2003

[3] Pecinovský, R.: Návrhové vzory. Computer Press, 2007

[4] Harold, E. R.; Means, W. S.: XML in a Nutshell : A Desktop Quick Reference (Nutshell Handbook). O'Reilly, 2001

[5] Bloch, J.: Effective Java (2nd Edition). Addison-Wesley, 2008

[6] KOGENT LEARNING SOLUTIONS INC: Java 7 Programming - Black Book. Dreamtech Press, 2013

[7] Steinpichler, D.: Project Management with UML and Enterprise Architect from Sparx Systems, Version 9.2. SparxSystems Software GmbH, 2010

[8] *XML Metadata Interchange (XMI).* [vid. leden 2016]. http://www.omg.org/spec/XMI/

Příloha A

Obsah přiloženého CD

- **example_xml** - složka obsahuje vzorový XML soubor vyexportovaný z Enterprise Architect a obrázky diagramů z tohoto souboru
- **java_classes** - složka obsahuje zdrojové kódy tříd, které jsem v rámci práce vytvořil či upravil
- *PCTGen.jar* - spustitelný jar soubor s PCTgen aplikací
- *BP-Adam_Lenger.pdf* - pdf s textem práce