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

Bakalářská práce

Add-in do nástroje Enterprise Architect

Plzeň 2016 Martin Košák

Místo této strany bude zadání práce.

Prohlášení

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

V Plzni dne 20. června 2016

Martin Košák

Poděkování

Chtěl bych poděkovat všem, kteří mi pomáhali při tvorbě této bakalářské práce. Můj největší dík patří vedoucímu práce Ing. Petrovi Přibylovi a konzultantovi Ing. Jakubovi Daňkovi.

Abstract

Main goal of this bachelor thesis is to develop a methodology for creating add-ins for the Enterprise Architect tool and verify correctness of this methodology. First part is focused on introduction of the Enterprise Architect and possibilities of adding a new functionality. Next part contains a methodology for the creation of add-ins and their distribution. Last part is focused on design of sample add-ins, their implementation using the developed methodology and their testing. Implementation of designed add-ins is mainly used as verification of correctness of the developed methodology.

Abstrakt

Hlavním cílem této práce je vytvoření metodiky pro tvorbu add-inů do nástroje Enterprise Architect a ověření její správnosti. První části práce se věnuje stručnému seznámení s nástrojem Enterprise Architect a s možnostmi jeho rozšíření o novou funkčnost. Další část práce obsahuje vytvořenou metodiku pro tvorbu add-inů a jejich následnou distribuci. Poslední část práce se věnuje návrhu vzorových add-inů, jejich následné implementaci podle vytvořené metodiky a jejich testování. Implementace navržených add-inů slouží zejména k ověření správnosti vytvořené metodiky.

Obsah

1 Úvod

Enterprise Architect je komplexní nástroj pro systémovou analýzu a návrh, který pokrývá celý životní cyklus vývoje systému, tzn. od zadání požadavků přes analýzu stavů, návrh modelu, testování a údržbu, to vše s využitím diagramů standardu Unified Modeling Language (UML). Kromě UML Enterprise Architect podporuje i řadu dalších standardů, např. Business Process Modeling Notation (BPMN).

Vzhledem k tomu, že každá firma při návrhu systémů používá vlastní metodiky a technologie, poskytuje Enterprise Architect řadu možností, jak jej rozšířit.

Prvním cílem této práce je prozkoumat možnosti rozšíření nástroje Enterprise Architect o nové funkčnosti.

Druhým cílem práce je navrhnout a sepsat metodiku vytváření nových komponent rozšiřujících funkčnosti nástroje Enterprise Architect.

Třetím cílem práce je navrhnout a implementovat rozšíření funkčnosti Enterprise Architect podle vytvořené metodiky. Tato rozšíření budou následně používána při návrhu informačních systémů v tomto nástroji. Funkčnost se rozšiřuje s cílem zefektivnění práce při návrhu systémů a přizpůsobení nástroje potřebám firmy.

Zadavatelem této práce je firma CCA Group a.s., jejímž metodikám a technologiím je práce přizpůsobena. Po přečtení čtenář získá představu o možnostech vytváření rozšíření pro Enterprise Architect. Dále je práce zaměřena na následný návrh, samostatnou implementaci a distribuci těchto rozšíření ke koncovým uživatelům.

2 Enterprise Architect

Enterprise Architect vytvořila australská softwarová společnost Sparx Systems. Je to kompletní nástroj pro systémovou analýzu a návrh, který pokrývá celý životní cyklus vývoje systému, tzn. od zadání požadavků přes analýzu stavů, návrh modelu, testování a údržbu, vše s využitím diagramů [2]. V současnosti (prosinec 2015) je nejnovější Enterprise Architect verze 12.1.

Verze 12.1 používá k vizualizaci systémů nejnovější standard UML 2.5 (více o UML v kapitole 2.1). K dokumentaci a popisu podnikových procesů Enterprise Architect využívá standard BPMN 2.0 (více o BPMN v kapitole 2.2). Dále je možné v Enterprise Architect využívat řadu dalších otevřených standardů pro návrh a modelování softwarových a podnikových systémů, jako např. Systems Modeling Language (SysML), Service Oriented Architecture Modeling Language (SoaML), ArchiMate a další [10, s. 7].

Enterprise Architect umožňuje generovat dokumenty na základě šablon distribuovaných spolu s nástrojem i na základě uživatelských šablon vytvářených podle potřeby přímo v tomto nástroji. Dále umožňuje generovat zdrojový kód (více o možnostech generování dokumentů a zdrojového kódu v [10, s. 11]).

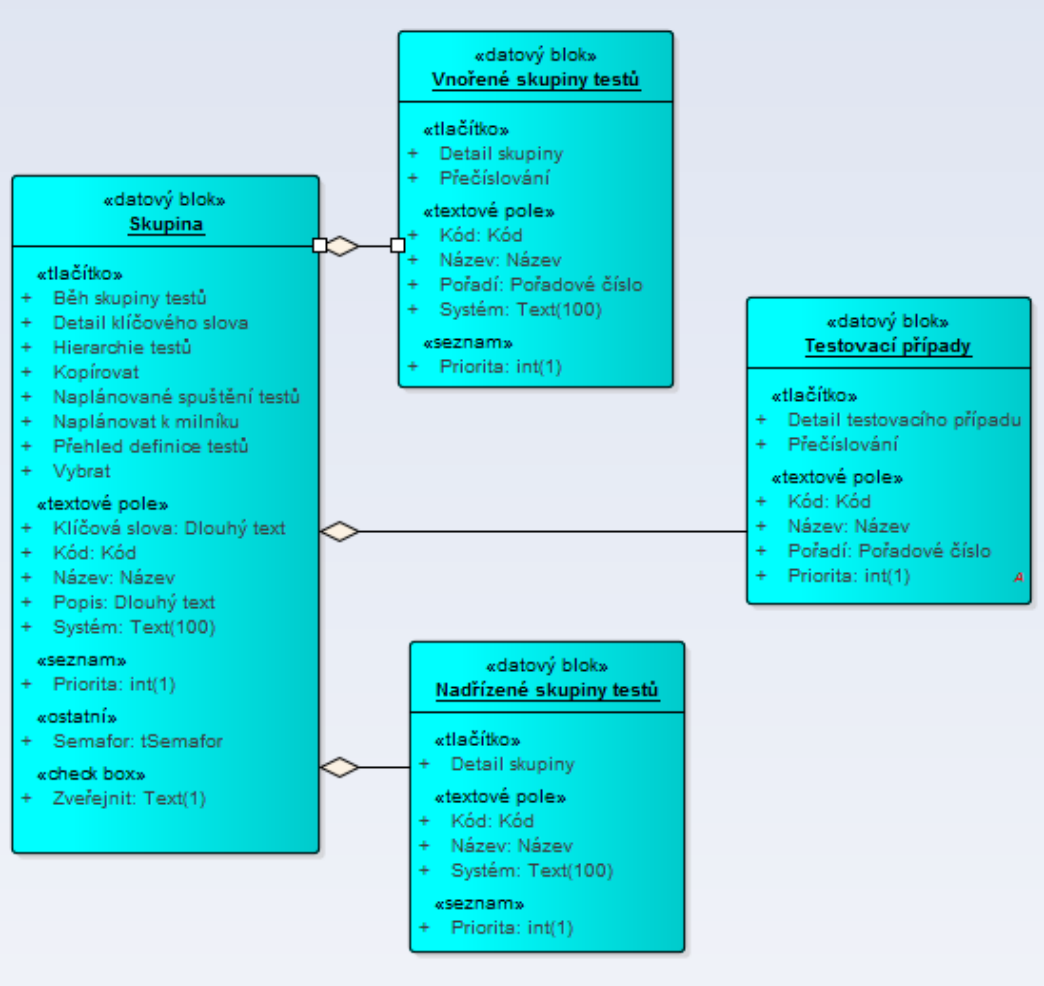
Nástroj ve firmě CCA Group a.s. využívají zejména analytici pro analýzu požadavků a návrh informačního systému a následné vygenerování projektové dokumentace, jako je například dokument Analýza a návrh pro zákazníka, který slouží jako podklad pro programátory.

2.1 UML

Unified Modeling Language (viz obr. 2.1), zkráceně UML), je průmyslový standard vytvořený za účelem vizuálního modelování a popisu procesu vývoje objektově orientovaných systémů (dále jen systémů). Byl standardizován skupinou Object Managment Group (OMG) [3].

Standard obsahuje sadu pravidel a symbolů. Symboly graficky popisují prvky systémů. Pravidla definují, jakým způsobem mohou být symboly mezi sebou propojeny a jak mohou být používány [3].

UML je možné využít k modelování podnikových procesů a jiných nesoftwarových systémů nebo k popisu implementace.

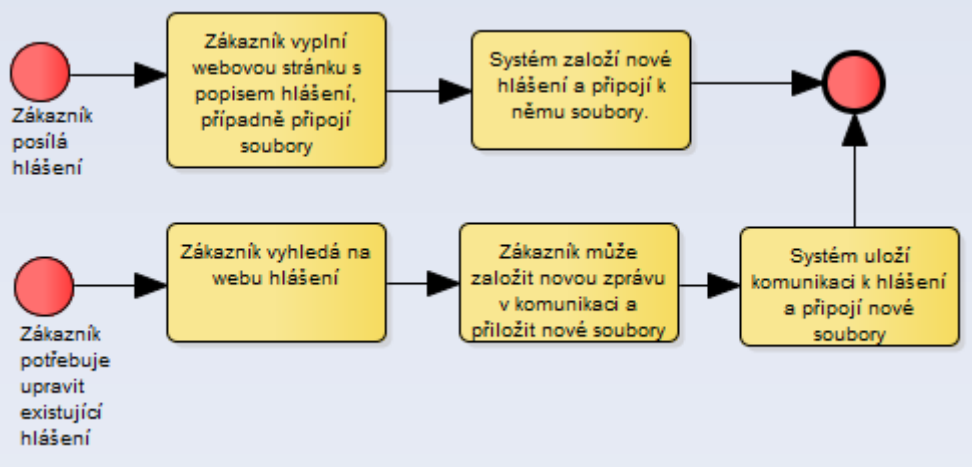


Obrázek 2.1: Model dat obrazovky pro správu testů navržený s použitím UML v nástroji Enterprise Architect.

2.2 BPMN

BPMN (viz obr. 2.2) nebo-li Business Process Modeling Notation je standard, soubor grafických prostředků a pravidel, pro popis podnikových procesů vytvořený iniciativou Business Process Managment Initiative (BMNI) [11].

BPMN definuje pouze jediný diagram pojmenovaný Business Process Diagram, který je tvořen grafickými objekty reprezentujícími zejména aktivity a zobrazení toků informací mezi nimi [11].



Obrázek 2.2: Business Process Diagram popisující založení a úpravu hlášení zákazníkem na helpdesku.

2.3 Základní prvky Enterprise Architectu

Aby bylo možné v nástroji Enterprise Architect navrhovat a modelovat systémy, je třeba mít k dispozici prvky, které toto umožní. Níže jsou popsány základní prvky, které Enterprise Architect poskytuje.

2.3.1 Packages

Package je kontejner, který ostatní objekty člení do složek. Může obsahovat i podmnožinu jiných packagů. Kromě klasických packagů má Enterprise Architect dva speciální typy, model a view.

V Enterprise Architect je model speciálním typem package. Model je kořenovým uzlem hierarchie projektu. Každý projekt může obsahovat jeden nebo více modelů. Model je tedy první package, který se po vytvoření projektu zakládá. Při založení nového projektu Enterprise Architect automaticky vytvoří model. Strukturu modelu lze vygenerovat nebo vytvořit postupně podle potřeb daného projektu. Další modely lze do projektu přidat samostatně [10, s. 753].

Další úroveň hierarchie projektu tvoří views. View je, stejně jako model, speciálním typem package. Používají se pro rozdělení a rozšíření struktury modelu do jednotlivých skupin podle požadavků a potřeb daného projektu. Mohou být vytvořeny pouze pod kořenovým uzlem (modelem). Enterprise Architect má šest různých typů view, které jsou popsány níže [10, s. 769].

- **Use Case View** - může například obsahovat případy užití a diagramy popisující analýzu.
- **Dynamic View** - pod těmito view je možné například vytvářet a ukládat activity diagramy, diagramy popisující komunikace, sekvenční diagramy.
- **Class View** - obsahuje například diagramy tříd a datové modely.
- **Component View** - pod tímto view můžeme vytvářet a ukládat diagramy různých komponent.
- **Deployment View** - může obsahovat například deployment diagramy.
- **Simple View** - slouží k přizpůsobení a vytvoření vlastních view.

2.3.2 Elementy

Elementy jsou společně s konektory (viz kapitola 2.3.3) základními objekty, pomocí kterých jsou tvořeny diagramy a celé modely. Různé elementy mají odlišný význam, pravidla a různé vlastnosti [10, s. 900]. Druhy elementů například jsou „Class“, „Object“, „Table“, „Datový blok“ a „Screen“.

2.3.3 Konektory

Konektory spolu s elementy tvoří základ modelu. Konektory propojují elementy a označují logický nebo funkční vztah, který mezi sebou jednotlivé elementy mají. Stejně jako elementy i konektory se mezi sebou liší významem, pravidly a vlastnostmi [10, s. 1102]. Druhy konektorů například jsou „Associate“, „Generalize“, „Aggregate“, „Abstraction“ a „Usage“.

2.3.4 Diagramy

Diagram je grafická reprezentace elementů a jejich vztahů, společně pak vytváří základ modelu. Vytváří se pod packages(2.3.1) a mohou mít případně rodičovské objekty. Lze je přesouvat mezi různými packages [10, s. 778].

Podle [10, s. 778] Enterprise Architect podporuje všechny diagramy definované standardem UML. Podporuje také diagramy jiných standardů, jako např. BPMN a umožňuje definovat i vlastní diagramy (viz 3.2.1).

2.3.5 Tagované hodnoty

Pomocí tagovaných hodnot můžeme objekty rozšířit o nové vlastnosti podle potřeb a pravidel organizace. Tagované hodnoty jsou kombinací vlastnosti a její hodnoty. Každému prvku Enterprise Architectu je možné definovat různé tagované hodnoty [10, s. 1134].

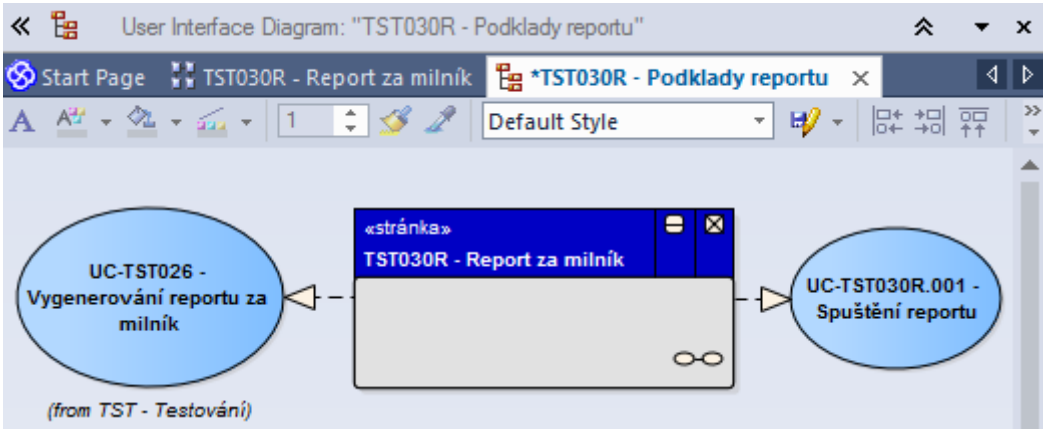
2.4 Uživatelské rozhraní

V Enterprise Architect má uživatel k dispozici grafické rozhraní, pomocí kterého navrhuje a modeluje systémy. Hlavní součásti rozhraní jsou Diagram View (kapitola 2.4.1), Diagram Toolbox (kapitola 2.4.2) a Project Browser (kapitola 2.4.3).

2.4.1 Diagram View

Diagram View (obr. 2.3) je hlavním pracovním prostorem Enterprise Architect pro vytváření struktury modelu pomocí různých objektů [10, s. 784].

Po kliknutí na diagram v Project Browseru (viz 2.4.3), zobrazí se právě v Diagram View. Můžou se zde vytvářet nové objekty a relace mezi nimi. Vytváří se zde struktura diagramu. Operací Drag And Drop lze vytvářet nové objekty například přetažením příslušného objektu z Diagram Toolboxu na Diagram View příslušného diagramu.

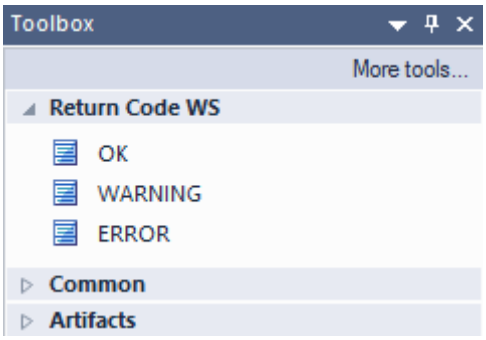


Obrázek 2.3: Diagram „TST030R - Podklady reportu“ otevřený v Diagram View.

2.4.2 Diagram Toolbox

Diagram Toolbox (obr. 2.4) je panel, pomocí kterého se vytvářejí elementy a konektory v diagramu. Konektory a další prvky jsou v panelu uspořádány do sekcí. Sekce obsahují prvky a konektory související s určitým typem diagramu. [10, s. 792].

Při otevření diagramu v Diagram View se v Diagram Toolboxu automaticky otevřou i sekce k příslušnému typu diagramu.



Obrázek 2.4: Diagram Toolbox s prvky náležejícími diagramům typu „Return Code WS“.

2.4.3 Project Browser

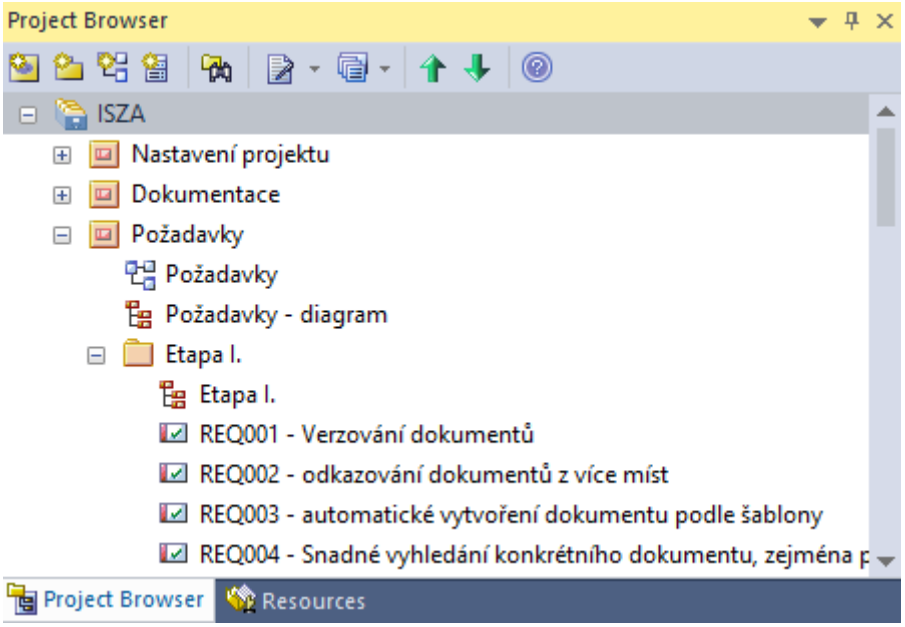
Project Browser (obr. 2.5) zobrazuje hierarchickou strukturu projektu. Každý objekt kromě objektu kořenového má nějaké nadřízené a podřízené objekty.

Jedná se o primární prostředek pro orientaci v projektu. Slouží k procházení různých objektů projektu a jejich podřazených objektů [10, s. 646]. Kromě procházení objektů projektu nabízí Project Browser mnoho jiných funkcí, např. z něj můžeme spouštět skripty a add-iny.

Podle [10, s. 646] nabízí kromě spouštění skriptů řadu dalších funkcí a může sloužit například k:

- Přesouvaní elementů a packagů v rámci projektu.
- Kopírovaní kompletních packagů.
- Importování a exportování informací o modelech.
- Importovaní kódu, xml a csv souborů, schémat databází a dalších externích zdrojů.

V Project Browseru je také možné vytvářet nové objekty, mazat již existující objekty nebo je editovat.



Obrázek 2.5: Project Browser.

3 Možnosti rozšíření

Každá firma používá během vývoje systémů své vlastní metodiky a technologie. Ne všechny funkce nástroje tak vyhovují a dostačují potřebám zaměstnanců firmy. Je třeba mít možnost si nástroj individuálně přizpůsobit.

Enterprise Architect je za tímto účelem do značné míry přizpůsobitelný. Uživatelé si mohou pomocí Model Driven Generation (MDG) technologie například vytvářet vlastní šablony pro generování dokumentace, vytvářet a definovat vlastní typy diagramů, elementů a dalších objektů (viz 3.2).

Enterprise Architect také umožňuje uživatelům implementovat vlastní funkčnost. Jednoduchou funkčnost je možné naimplementovat pomocí skriptů (kapitola 3.3). Složitější funkčnost lze dodat s využitím technologie Component Object Model (COM) ve formě komponent napsaných například v programovacím jazyku C# nebo C++ (viz kapitola 3.4). Při vytváření skriptů nebo add-inů je nutné pracovat s objekty nástroje Enterprise Architect. Pro práci s těmito objekty poskytuje nástroj různé třídy (viz kapitola 3.1).

3.1 Programové rozhraní

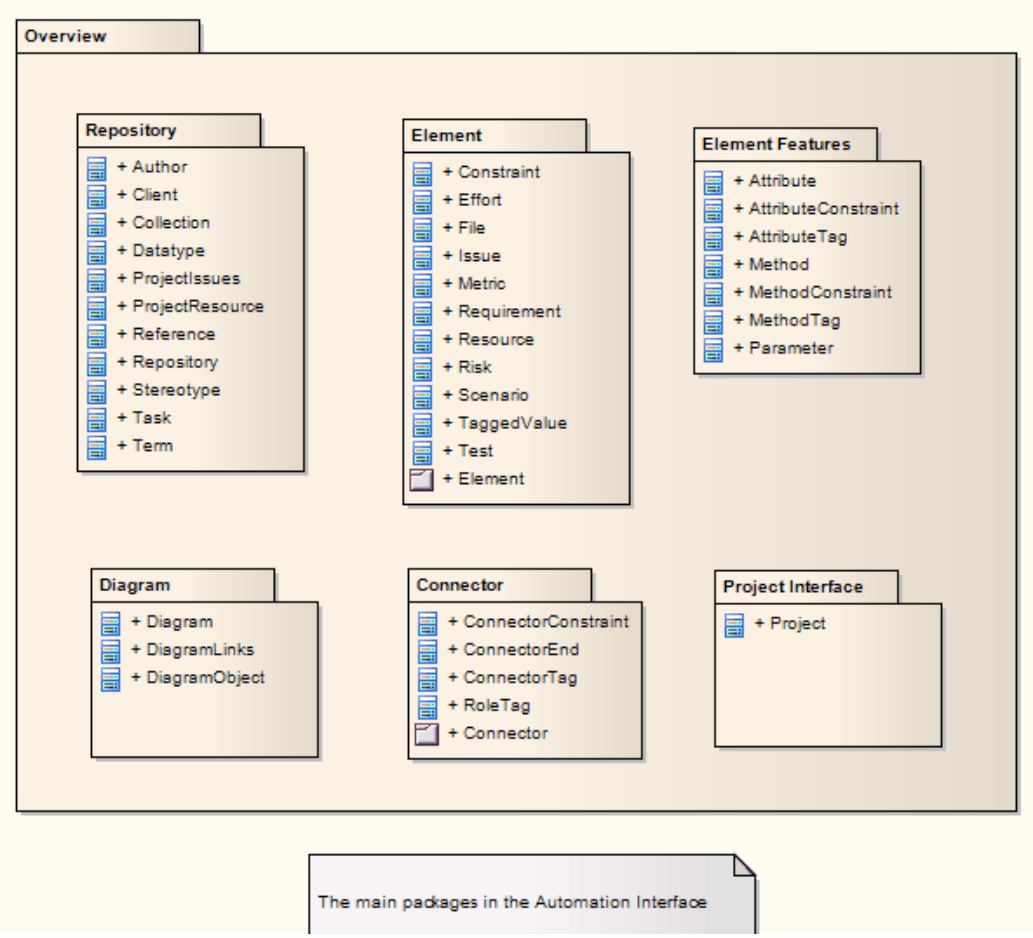
Přístup k objektů a práci s nimi umožňuje rozhraní **`Automation`** nástroje Enterprise Architect. Toto rozhraní poskytuje přístup k různým balíkům, které dále poskytují přístup ke konkrétním metodám a vlastnostem tříd spadajícím pod tyto balíky. Nejdůležitějšími balíky jsou **`Repository`**, **`Element`** a **`Diagram`** (více viz [10, s. 2813]). Strukturu hlavních balíků a do nich spadajících tříd zobrazuje obrázek 3.1.

Ze tříd je nejdůležitější třída **`Repository`**. Tato třída je podle [10, s. 2850] hlavním kontejnerem všech struktur jako je např. **`model`** nebo jakými jsou **`elementy`** a **`packages`**. Pomocí této třídy je tedy možné k těmto strukturám přistupovat a dále s nimi pracovat.

Jinou důležitou třídou je třída **`Element`**. Tato třída poskytuje metody a atributy potřebné pro práci s jednotlivými elementy. Další informace o této třídě lze nalézt v [10, s. 2881]

Další důležitou třídou je třída **`Attribute`**. Třída **`Attribute`** poskytuje metody a atributy pro práci s jednotlivými atributy elementů. Podrobnější informace o této třídě je možné nalézt v [10, s. 2911].

Informace o dalších třídách je možné nalézt v [10].



Obrázek 3.1: Hlavní balíky poskytované rozhraním Automation nástroje Enterprise Architect a třídy k jejichž metodám a vlastnostem tyto balíky poskytují přístup. Obrázek je převzat z [10, s. 2814]

3.2 MDG technologie

MDG neboli Model Driven Generation technologie je nástroj, který umožňuje rozšířit Enterprise Architect o vlastní nástroje a metody, jako jsou UML Profiles, skripty, vzory, různé šablony dokumentů, panel nástrojů, diagramy a další [10, s. 1475].

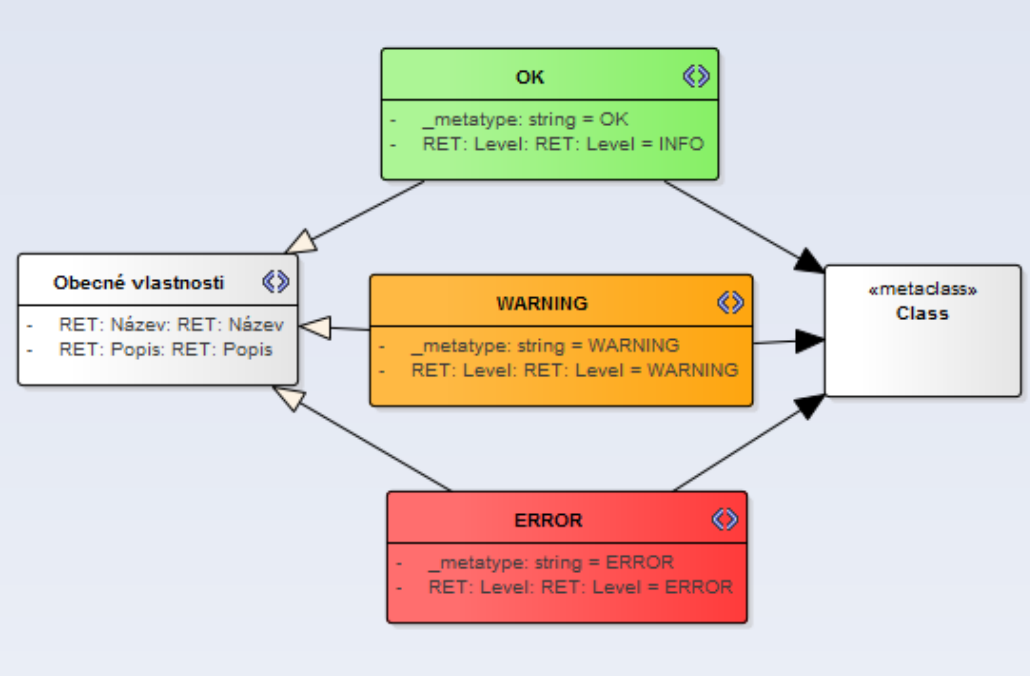
Ne vždy se nám např. hodí typy elementů, atributů nebo tagovaných hodnot v takové podobě, v jaké jsou dodány spolu s nástrojem Enterprise Architect. Většinou při vývoji projektu potřebujeme definovat pro daný projekt vlastní objekty, které budou lépe vyhovovat našim potřebám. Pro takové účely můžeme vytvořit vlastní MDG technologii. V MDG technologii lze definovat například vlastní typy tagovaných hodnot a šablon pro generování

dokumentů. Mohou také obsahovat námi vytvořené UML Profiles a Toolbox Profiles.

3.2.1 UML Profiles

UML Profiles jsou kolekcemi rozšíření. Konkrétně rozšíření stereotypů a s nimi spojených tagovaných hodnot přidaných do klasických UML elementů. Společně pak popisují nějaký konkrétní problém v návrhu systému [10, s. 1472].

Kolekce lze vytvářet, upravovat a přizpůsobovat vlastním potřebám. Umožňují definovat vlastní stereotypy, tagované hodnoty, diagramy a další objekty pomocí UML (viz obr. 3.2).



Obrázek 3.2: Nové objekty „OK“, „WARNING“ a „ERROR“ definované v UML Profile v rámci MDG technologie.

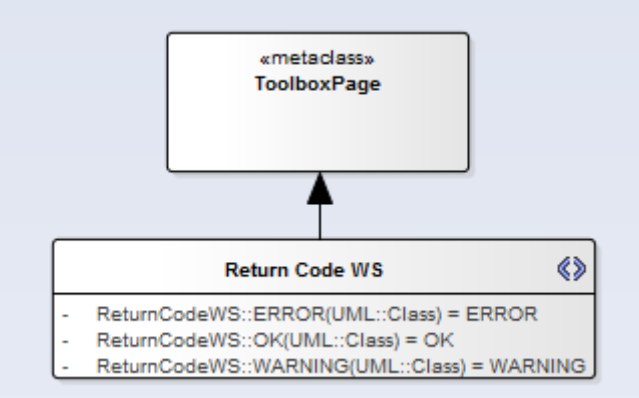
Na obrázku 3.2 jsou definované objekty „OK“, „WARNING“ a „ERROR“. Tyto objekty jsou rozšířením obecného objektu typu `Class`. Nové objekty mají společné „Obecné vlastnosti“. Tyto vlastnosti jsou definovány jako tagované hodnoty. Každý z objektů má také definovanou vlastní tagovanou hodnotu s názvem „RET:Level“ s rozdílnými inicializačními hodnotami.

3.2.2 Toolbox Profiles

Toolbox Profiles zařizují přístup ke všem prvkům vytvořeným v rámci MDG technologie [10, s. 1560]. V případě vytváření vlastní MDG technologie je vhodné v jejím rámci vytvořit i příslušný Toolbox Profile. Vytváří se pomocí UML stejně jako UML Profiles (viz obr. 3.3).

Bez vytvoření `Toolbox Profile` nebudou mít nově definované objekty vlastní toolbox v okně `Toolbox View`.

Na obrázku 3.3 je zobrazena definice nového toolboxu. Výsledný toolbox je zobrazen na obrázku 2.4 v kapitole 2.4.2.



Obrázek 3.3: Nově definovaný Toolbox Profile s názvem „Return Code WS“ definovaný v rámci MDG technologie.

3.3 Skripty

Pomocí skriptů můžeme do Enterprise Architectu dodat jednoduchou funkčnost. To se obzvláště hodí, pokud často provádíme ručně nějakou činnost, která je zdlouhavá nebo se stále opakuje. Například v projektech často vytváříme diagramy, ve kterých vždy zakládáme dva stejné objekty. V momentě, kdy diagram s takovouto strukturou v jednom projektu zakládáme např. desetkrát, by se nám již hodil nějaký skript, který bychom spustili a on by automaticky založil diagram i se zmíněnými objekty místo nás.

K usnadnění či zrychlení takové činnosti nám umožňuje Enterprise Architect vytvořit skript. Skripty pro Enterprise Architect lze vytvářet ve třech skriptovacích jazycích, a to v JavaScriptu, JScriptu a VBScriptu [10, s. 2791].

- **JavaScript** - jedná se o multiplatformní objektově orientovaný skriptovací jazyk. Nejčastěji se s ním můžeme setkat na webových stránkách.

- **JScript** - JScript pochází z dílny Microsoft Corporation. Jedná se o objektově orientovaný skriptovací jazyk [6].
- **VBScript** - Microsoft Visual Basic Scripting Edition pochází stejně jako JScript z dílny Microsoft Corporation. Je podobný programovacímu jazyku Visual Basic [9].

3.4 Add-iny

Add-iny, stejně jako skripty, slouží k přidání nové funkčnosti do nástroje Enterprise Architect. S jejich pomocí je možné implementovat novou funkčnost včetně uživatelského rozhraní a reakcí na různé události.

Můžeme v nich definovat menu a sub-menu Enterprise Architect. Add-iny také dostávají zprávy o různých událostech, které nastaly v uživatelském rozhraní. Díky zprávám o událostech, jako je třeba kliknutí myši na položku menu nebo vytvoření nového objektu v diagramu, můžou add-iny na tyto události nějakým způsobem reagovat [10, s. 3010].

Podle [10, s. 3097] existuje speciální typ add-inů nazývaný MDG add-iny. Tyto add-iny mají oproti klasickým add-inům vlastní události, funkce a požadavky. Na rozdíl od klasických add-inů a jejich událostí musí MDG add-iny mít implementované metody pro každou MDG událost.

Add-iny jsou ActiveX COM objekty, využívající funkce rozhraní Automation Enterprise Architectu[10, s. 3010].

Toto rozhraní poskytuje možnost přístupu jiným aplikacím k informacím Enterprise Architect modelů. Tento přístup je prováděn pomocí technologie Object Linking and Embedding (OLE) Automation firmy Microsoft [10, s. 2804]. Rozhraní Automation nástroje Enterprise Architect potom poskytuje přístup k vnitřní části modelů (viz kapitola 3.1).

Add-iny lze vytvářet ve formě knihoven a následně distribuovat k uživatelům například ve formě Microsoft Windows Installer (MSI) balíčků.

3.4.1 COM

Podle [1, s. 836] definuje technologie COM komponentový model a je předchůdcem technologie .NET. Pro tento model je možné vytvářet komponenty v různých programovacích jazycích. Komponenta vytvořená v jazyce C++ tedy může být například použita z klienta vytvořeného v jazyce Visual Basic.

Programovací jazyk C# neumožňuje vytváření COM komponent, ale umožňuje vytváření .NET komponent. Vytvořenou .NET komponentu je možné následně prostřednictvím obalujícího COM objektu využívat jako

COM objekt [1, s. 842]. Mezi technologiemi .NET a COM tedy existuje možnost vzájemné spolupráce.

3.4.2 OLE Automation

OLE Automation umožnuje podle [7] softwarovým komponentám poskytovat jejich jedinečné vlastnosti jiným nástrojům a aplikacím. Za tímto účelem používá technologii COM (kapitola 3.4.1).

Objekty, které poskytují své vlastnosti, metody a události jiným aplikacím či nástrojům, se nazývají ActiveX objekty. Jedná se o instance tříd. Aplikace nebo knihovna, která může vytvořit jeden či více ActiveX objektů, se nazývá ActiveX komponenta [5]. Add-in, který je dodán ve formě knihovny je tedy komponentou.

Aplikace a nástroje přistupující k ActiveX objektům se nazývají ActiveX klienti. Dokáží manipulovat jedním či více ActiveX objekty. Mohou používat již existující objekty, vytvářet jejich nové instance, měnit a číst hodnoty jejich vlastností a volat metody objektů [4].

4 Vytváření nových add-inů

V předešlých kapitolách jsou popsány základní prvky nástroje Enterprise Architect a možnosti jeho rozšíření. Tato kapitola je zaměřena na vytvoření add-inu a jeho následné distribuce ve formě ActiveX COM komponenty.

V následujícím textu je popsáno vytváření add-inů pro Enterprise Architect verze 11 a novější. Metodika je sepsána pro programovací jazyk C# s použitím vývojového prostředí Microsoft Visual Studio 2013.

Vytváření add-inu se skládá z několika kroků. Těmito kroky jsou vytvoření a nastavení nového projektu (kapitola 4.1), přidání nového klíče do registrů (kapitola 4.2), připojení projektu k rozhraní Automation Enterprise Architect (kapitola 4.3), implementace potřebných událostí Enterprise Architect (kapitola 4.4), ladění implementovaného kódu (kapitola 4.5) a distribuce ke koncovým uživatelům (kapitola 5).

4.1 Vytvoření a nastavení nového projektu

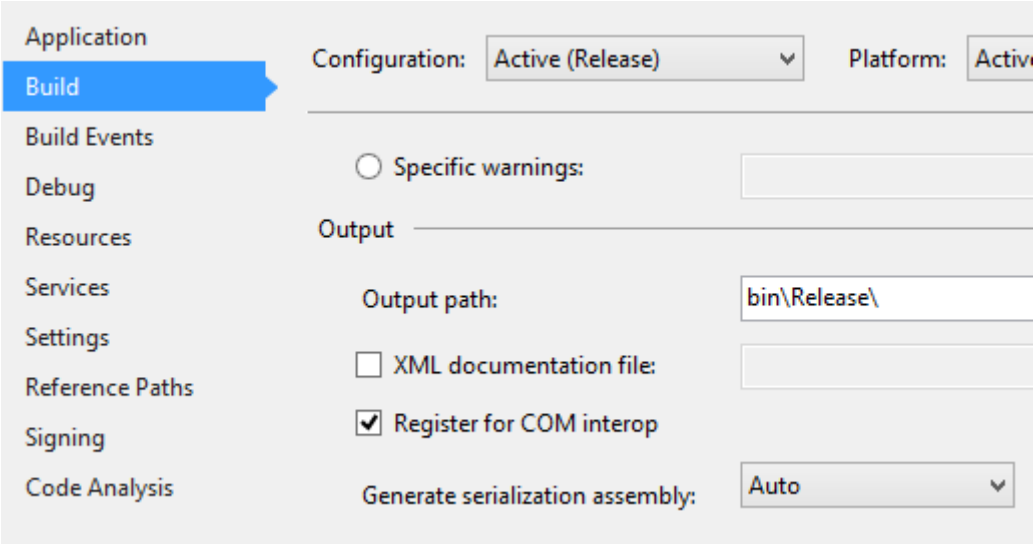
Prvním krokem při vytváření add-inů je vytvoření nového projektu a jeho nastavení. Visual Studio se musí spustit v režimu správce, protože při překladu bude přistupovat k registrům, a proto musí mít příslušná práva. Bez těchto přístupových práv překlad projektu selže.

Po spuštění Visual Studia je nutné založit nový projekt pomocí položky **`File -> New -> Project`** a dále pod položkou **`Installed -> Templates`** zvolit **`Visual C#`**. Add-in bude distribuován ve formě knihovny, proto je vhodné zvolit **`Class Library`** jako typ projektu. Dále je nutné zapsat název projektu, zvolit umístění projektu na disku, případně změnit název **`Solution`** a na závěr potvrdit zvolená nastavení.

Aby se ze tříd projektu stala komponenta v podobě knihovny, je třeba upravit nastavení pro překlad a sestavení. Konkrétně je potřeba nastavit registraci knihovny v operačním systému (kapitola 4.1.1) a zpřístupnit metody knihovny ActiveX klientům (kapitola 4.1.2). Tato nastavení se nachází pod položkou **`Properties`**. **`Properties`** je možné otevřít z okna **`Solution Explorer`**. Dvojklikem na název vytvořeného projektu se rozbalí položky menu (pokud již nejsou rozbaleny), mezi nimiž je i položka **`Properties`**. Další možností je kliknout na název projektu levým tlačítkem myši. Po kliknutí se objeví menu, ve kterém je na konci položka **`Properties`**.

4.1.1 Registrace knihovny

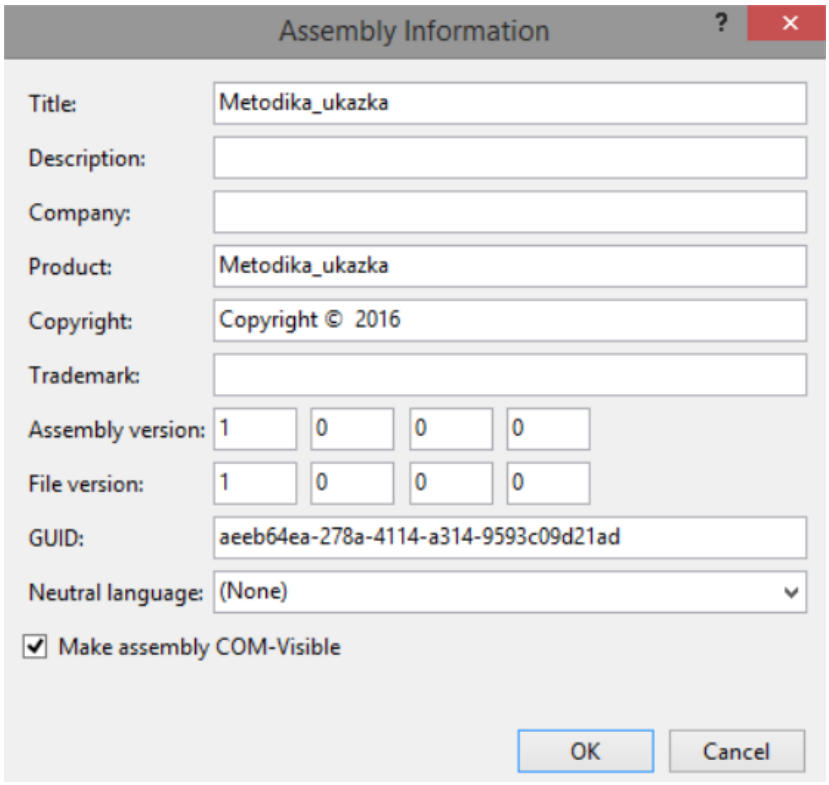
Aby operační systém věděl, že vytvořená knihovna je komponentou, je třeba ji do systému zaregistrovat. Nejjednodušší cestou je nastavit Visual Studio tak, aby při překladu a následném sestavení projektu samo zaregistrovalo knihovnu do systému. Registraci lze nastavit v položce **`Properties`** na záložce **`Build`** zaškrtnutím volby „Register for COM interop“(viz obr. 4.1). Toto nastavení se uloží současně s uložením projektu.



Obrázek 4.1: Nastavení automatické registrace knihovny.

4.1.2 Zpřístupnění metod add-inu

V položce **`Properties`** na záložce **`Application`** se nachází tlačítko **`Assembly Information`**. Po kliknutí na toto tlačítko se otevře nové okno, ve kterém je potřeba zaškrtnout volbu „Make assembly COM-Visible“. Zaškrtnutím této volby se zajistí, že metody výsledné knihovny budou viditelné pro ActiveX klienty (konkrétně pro Enterprise Architect). Kromě této volby je možné v okně nastavit další informace o sestavení, jako např. verzi sestavení, verzi souboru, popis a další doplňující informace (viz obr. 4.2). Na závěr je nutné potvrdit zvolená nastavení.



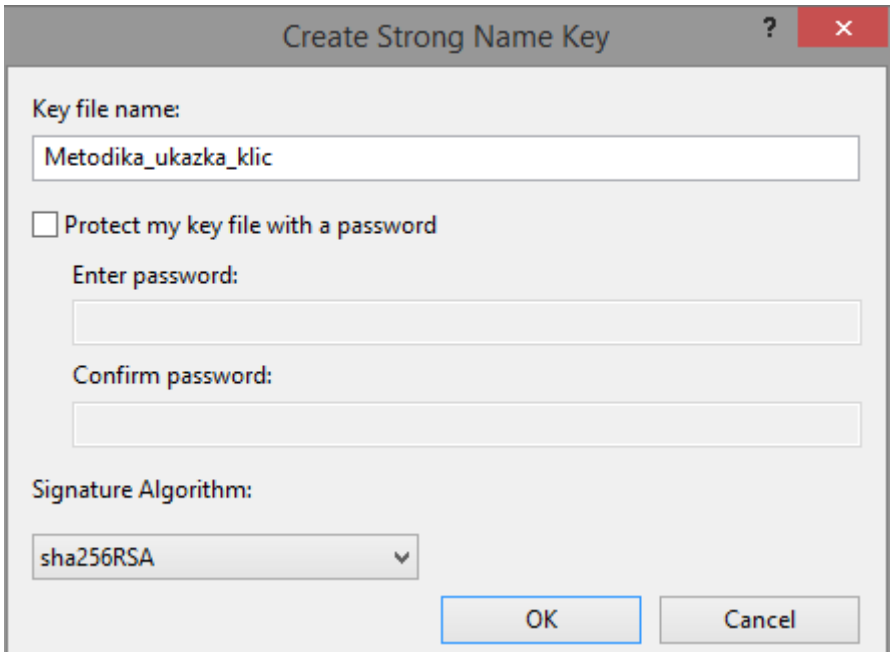
Obrázek 4.2: Nastavení atributů sestavení se zaškrtnutou volbou „Make assembly COM-visible“.

4.1.3 Vytvoření silného jména

Add-in musí být v systému jednoznačně identifikovatelný. K jednoznačné identifikaci bohužel nestačí add-in jen pojmenovat. V momentě, kdy by byly ve stejném systému zaregistrované dva stejně pojmenované add-iny, by systém nevěděl, která knihovna se má použít. Aby k této kolizi nedošlo, je nutné sestavení podepsat silným jménem. [1, s. 711] říká, že silné jméno je jednoznačným identifikátorem výsledného sestavení. Podle [8] je možné vytvořit silné jméno pomocí nástroje Visual Studio níže uvedeným postupem.

V položce **`Properties`** na záložce **`Signing`** je nutné zaškrtnout volbu „Sign the assembly“. Po zaškrtnutí této volby se zpřístupní dialog „Choose a strong name key file“, ve kterém lze vybrat již existující soubor nebo vytvořit nový. Pro vytvoření nového souboru je nutné zvolit položku **`<New...>`**. Pro výběr již existujícího souboru je nutné vybrat položku **`<Browse...>`**.

Po zvolení položky **`<New...>`** se otevře nové okno, ve kterém je možné vytvořit nový soubor s klíčem (viz obr. 4.3). Po vybrání položky **`<Browse...>`** se otevře standardní okno windows pro výběr souboru. V něm je možné vybrat již existující soubor s klíčem.



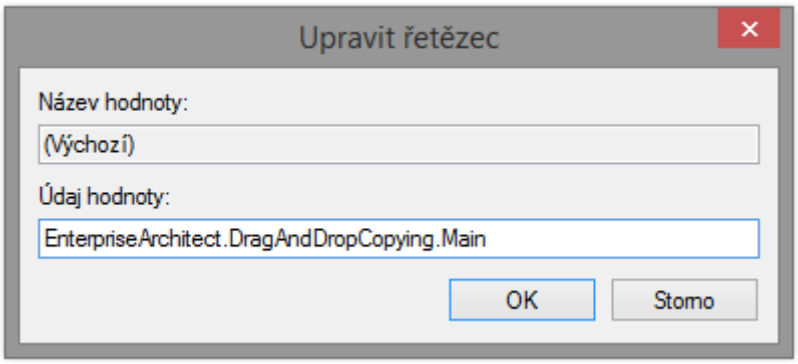
Obrázek 4.3: Okno pro vytvoření souboru se silným jménem.

Před vytvořením souboru je nutné upravit nastavení. Soubor je třeba pojmenovat v poli **`Key file name`**. Dále je možné soubor zabezpečit heslem a případně vybrat algoritmus, pomocí kterého bude silné jméno vytvořeno. Úprava nastavení se musí potvrdit kliknutím na tlačítko **`OK`**. Po potvrzení dojde k vytvoření souboru. Tento soubor má příponu **`.snk`** a nachází se v umístění projektu.

4.2 Přidání klíče do registrů

Aby Enterprise Architect add-in viděl, je třeba přidat klíč do jeho registrů. Klíč je třeba přidat do lokace **`HKEY_CURRENT_USERS\Software\Sparx Systems\EAAddins`**.

Přidání klíče lze provést např. pomocí nástroje **`regedit`**. Ve výše zmíněné lokaci stačí pro vytvoření nového klíče kliknout levým tlačítkem myši na **`EAAddins`** a zvolit **`Nový -> Klíč (New -> Key)`**. Nový klíč by měl mít stejný název, jako je název projektu ve Visual Studiu. Editor registrů automaticky vytvoří defaultní hodnotu pro nový klíč. Dvojklikem na vytvořenou defaultní hodnotu lze otevřít okno, do kterého je nutné zadat atribut **`Údaj hodnoty`**. **`Údaj hodnoty`** musí být ve tvaru **`<NameSpace>.<ClassName>`**, kde **`<NameSpace>`** je celý název „namespace“ projektu a **`<ClassName>`** je název hlavní třídy projektu (viz obr. 4.4).



Obrázek 4.4: Hodnota v registrech, kde „EnterpriseArchitec.DragAndDrop“ je celý název namespace projektu a „Main“ je název hlavní metody projektu.

4.3 Připojení k rozhraní Automation

Aby bylo možné využívat metody, atributy a další možnosti nástroje Enterprise Architect, je třeba projekt propojit s rozhraním Automation tohoto nástroje. Propojení je možné provést přidáním reference na knihovnu **`Interop.EA.dll`** do projektu.

Referenci je možné přidat z okna **`Solution Explorer`**. Dvojklikem na název projektu se rozbalí nabídka, ve které se nachází položka **`References`**. Kliknutím pravým tlačítkem myši na tuto položku se rozbalí menu s dalšími volbami. Pro přidání reference do projektu slouží volba **`Add Reference`**. Kliknutím na tuto volbu se otevře okno **`Reference Manager`**. Pro přidání reference na rozhraní Enterprise Architect je potřeba pokračovat kliknutím na tlačítko **`Browse`**, které otevře standardní okno windows pro výběr souboru. V tomto okně je třeba vybrat výše zmíněný soubor **`Interop.EA.dll`**, který se nachází v instalační složce Enterprise Architectu (např. v **`C:\Program Files (x86)\Sparx Systems\EA`**). Po vybrání tohoto souboru je ještě třeba zapsat do hlavní třídy projektu příkaz „**`using EA;`**“. Nyní by mělo být možné využívat všechny metody, atributy a další možnosti, které Enterprise Architect poskytuje.

4.4 Důležité události

Před programováním konkrétní funkčnosti je nutné nejprve implementovat v hlavní třídě projektu metody zpracovávající některé události nástroje Enterprise Architect. Konkrétně se jedná o události zajišťující připojení a odpojení add-inu do Enterprise Architect (kapitoly 4.4.1 a 4.4.2) a události zajišťující získání a úpravu položek menu (kapitoly 4.4.3 a 4.4.4).

4.4.1 EA_Connect

Událost **EA_Connect** slouží podle [10, s. 3022] k identifikaci typu add-inu a umožňuje reagovat na spuštění Enterprise Architect.

Hlavičku metody pro zpracování této události zobrazuje kód 4.1. Hlavička metody je uvedena v pseudokódu. V tabulce 4.1 je uveden datový typ a význam vstupního parametru metody.

Kód 4.1: Hlavička metody pro zpracování události **EA_Connect.**

```
String EA_Connect(Repository)
```

Parametr	Typ	Informace
Repository	**EA.Repository**	Objekt reprezentující aktuálně otevřený Enterprise Architect model.

Tabulka 4.1: Vstupní parametr metody **EA_Connect**.

Návratová hodnota je datového typu String a může nabývat hodnoty „MDG“, pokud se jedná o MDG add-in dostávající MDG události a zvláštní nastavení menu. Pokud add-in není specializovaný, tvoří návratovou hodnotu prázdný řetězec.

4.4.2 EA_Disconnet

Tato událost nastane při ukončování Enterprise Architect. Lze ji využít pro úklid paměti. Nemá žádné vstupní parametry ani návratovou hodnotu.

Uvnitř metody pro zpracování této události musí být podle [10, s. 3017] zavolány metody **GC.Collect** a **GC.WaitForPendingFinalizers**. Jedná se o metody pro správu paměti. Bez zavolání těchto metod by mohlo dojít k nesprávnému ukončení Enterprise Architect.

Příklad metody pro zpracování události **EA_Disconnect** zobrazuje kód 4.2.

Kód 4.2: Příklad metody pro zpracování události **EA_Disconnect.**

```
void EA_Disconnect(){
        GC.Collect();
        GC.WaitForPendingFinalizers();
}
```

4.4.3 EA_GetMenuItems

Událost **EA_GetMenuItems** nastane podle [10, s. 3023] pokud uživatel klikne na nějakou položku menu. Událost nastane před tím, než jsou zobrazeny položky zvoleného menu. Jedná se konkrétně o menu **EXTENSIONS**, které se nachází v horní liště uživatelského rozhraní Enterprise Architect, nebo které se zobrazí po kliknutí pravým tlačítkem myši na element v Project Browseru či v Diagram View. Tato událost nastane také v případě rozbalení submenu definovaného add-inem.

Hlavičku metody pro zpracování této události zobrazuje kód 4.3. Hlavička metody je uvedena v pseudokódu. V tabulce 4.2 jsou uvedeny datové typy a význam vstupních parametrů této metody.

Kód 4.3: Hlavička metody pro zpracování události **EA_GetMenuItems**.

```
object EA_GetMenuItems(Repository, Location, MenuName)
```

Parametr	Typ	Info
Repository	**EA.Repository**	Objekt reprezentující aktuálně otevřený Enterprise Architect model.
Location	**String**	Řetězec reprezentující část uživatelského rozhraní, ze které bylo menu otevřeno. Hodnoty mohou být TreeView, MainMenu nebo Diagram .
MenuName	**String**	Název menu, ve kterém se položka nebo submenu nachází.

Tabulka 4.2: Parametry metody **EA_GetMenuItems**.

Návratovou hodnotou události je objekt. Konkrétně se jedná o String reprezentující jednu položku menu, pole Stringů reprezentující položky ve zvoleném menu nebo null, pokud nelze zobrazit žádné položky.

4.4.4 EA_MenuClick

Událost **EA_MenuClick** nastane v případě, že uživatel zvolí nějakou položku menu, která není rodičovská [10, s. 3025]. Tuto metodu je vhodné využít pro spuštění vlastních metod add-inu.

Hlavičku metody pro zpracování této události zobrazuje kód 4.4. Hlavička metody je uvedena v pseudokódu. V tabulce 4.3 jsou uvedeny datové typy a význam vstupních parametrů této metody.

Kód 4.4: Hlavička metody pro zpracování události **EA_MenuClick**.

```
void EA_MenuClick(Repository, Location, MenuName, ItemName)
```

Parametr	Typ	Info
Repository	**EA.Repository**	Objekt reprezentující aktuálně otevřený Enterprise Architect model
Location	**String**	Řetězec reprezentující část uživatelského rozhraní, ze které bylo menu otevřeno. Hodnoty mohou být TreeView, MainMenu nebo Diagram .
MenuName	**String**	Název menu, ve kterém se položka nebo submenu nachází.
ItemName	**String**	Název vybrané možnosti.

Tabulka 4.3: Parametry metody **EA_MenuClick**.

Tato událost nevrací žádnou hodnotu.

4.5 Ladění

Během vytváření add-inu je třeba zkoušet nově implementovanou funkčnost a v případě neočekávaného chování nalézt a opravit chyby.

Funkčnost je možné ověřit přímo v Enterprise Architect spuštěním příslušného add-inu a jeho ručním vyzkoušením. Add-iny se spouští z menu **Extensions** kliknutím na příslušnou položku. Před opravením případné chyby je nutné Enterprise Architect vypnout. V případě běhu tohoto nástroje nebude moci Visual Studio projekt přeložit z důvodu nemožnosti přepsání používaných souborů.

Visual Studio nabízí také možnost debugování add-inu. Pro spuštění debugování stačí nastavit v kódu breakpointy, spustit Enterprise Architect a pod položkou **DEBUG** zvolit **Attach to Process**. Po této volbě se objeví nové okno obsahující názvy běžících procesů. Nástroji Enterprise Architect patří proces s názvem EA.exe, který je potřeba zvolit a následně potvrdit tlačítkem **Attach**. Po spuštění add-inu se poté vykonávání kódu zastaví na nastaveném breakpointu.

5 Distribuce add-inu

Pokud je add-in již hotov, otestován a funguje správně, je možné ho distribuovat koncovým uživatelům. Distribuci je možné provést dvěma způsoby.

Prvním způsobem je poskytnout uživatelům pouze vytvořenou knihovnu s návodem na zprovoznění (kapitola 5.1). Tento způsob ovšem není z pohledu koncového uživatele příliš ideální.

Další možností je vytvořit instalační balíček, který uživatel pouze spustí a instalátor provede všechny potřebné úkony za něj (kapitola 5.2). Tento způsob je pro koncového uživatele jistě mnohem pohodlnější.

5.1 Manuální instalace

Před tím, než bude uživatel moci vytvořený add-in používat v Enterprise Architect, je potřeba add-in nainstalovat do jeho počítače.

Instalaci lze rozdělit do tří základních kroků. Prvním krokem je umístění vytvořené knihovny s add-inem do požadované složky. Druhým krokem je zaregistrování dané knihovny do operačního systému jako COM komponenty. Posledním krokem je zapsání add-inu do registrů nástroje Enterprise Architect.

Knihovna je ve formě souboru s příponou **`dll`**. Tento soubor je výstupem přeloženého a sestaveného projektu s add-inem. Nachází se ve složce projektu v **`\bin\Release`**. Knihovnu je nutné zkopírovat do počítače koncového uživatele do složky, kterou si zvolí. Vhodnou volbou pro umístění knihovny je například vytvoření nové složky ve složce **`Program Files`**.

Po zkopírování knihovny do zvolené složky je třeba ji zaregistrovat do operačního systému. Registraci je možné provést pomocí nástroje **`Assembly Registration Tool (RegAsm)`**. Tento nástroj lze spustit z příkazové řádky. Příkazová řádka musí být spuštěna v režimu správce. Po spuštění příkazové řádky je nutné zadat cestu **`C:\WINDOWS\Microsoft.NET\Framework\`**<**`verze frameworku`**>, kde <**`verze frameworku`**> je označení nejvyšší verze frameworku přítomné na uživatelském počítači (např. v4.0.30319). Z této lokace je možné příkazem **`RegAsm.exe "`**<**`cesta ke knihovně`**>**`"/codebase`**, kde <**`cesta ke knihovně`**> je úplná cesta ke knihovně obsahující požadovaný add-in, spustit nástroj **`RegAsm`** (viz kód 5.1). Cesta musí vést do umístění, do kterého byla knihovna umístěna na počítači koncového uživatele. Po úspěšném spuštění tohoto příkazu se na obrazovce objeví stejný výstup, jako je zobrazený na obrázku 5.1.

Kód 5.1: Příkaz pro spuštění nástroje RegAsm.

```
RegAsm.exe "D:\Program Files\CCA EA add-ins\Metodika_ukazka.dll" /codebase
```

Obrázek 5.1: Výstup po spuštění nástroje RegAsm a úspěšné registraci knihovny.

Posledním krokem manuální instalace je zapsání add-inu do registrů Enterprise Architect. Bez tohoto kroku nebude Enterprise Architect o add-inu vědět. Postup zápisu do registrů byl již podrobně popsán v kapitole 4.2.

5.2 Vytvoření instalačního balíčku

Jelikož manuální instalace není pro uživatele příliš pohodlná, je vhodné celý proces popsaný v kapitole 5.1 zautomatizovat.

K automatizaci lze využít například Windows Installer XML Toolset (WiX)[1]. Pomocí WiX lze vytvářet instalační balíčky s využitím značkovacího jazkyka Extensible Markup Language (XML). Po nainstalování je možné tento nástroj používat ve Visual Studiu. Ze všeho nejdříve je potřeba založit nový projekt (viz kapitola 5.2.1). Dalšími kroky jsou vložení knihovny a její registrace jako COM komponenty (kapitola 5.2.2), vložení hodnot do registrů (kapitola 5.2.3), výběr místa pro instalaci (kapitola 5.2.4) a otestování výsledného instalačního balíčku (kapitola 5.2.5).

5.2.1 Vytvoření a nastavení projektu

Nový projekt je vhodné vytvořit v **`solution`**, ve kterém se nachází projekt s add-inem. Prvním krokem je tedy spuštění Visual Studia v režimu správce a otevření příslušného **`solution`**.

Po otevření příslušného **`solution`** je nutné založit nový projekt. To lze provést z okna **`Solution Explorer`** po kliknutí levým tlačítkem myši na název **`solution`** a výběrem volby **`Add -> New Project`**. Po výběru této volby se otevře okno s názvem **`Add New Project`**. Pro vytvoření instalačního balíčku je nutné zvolit pod položkou **`Installed -> Windows Installer`**

[1] stránky distributora: wixtoolset.org

XML typ projektu **Setup Project**. V tomto okně také lze nastavit název a umístění projektu. Vytvoření nového projektu je nutné po provedení všech potřebných nastavení potvrdit tlačítkem **OK**.

Nově vytvořený projekt obsahuje soubor **Product.wxs**. V tomto souboru se vytváří instalační balíček pomocí značkovacího jazyka **XML**. Po vytvoření projektu se v souboru **Product.wxs** již nachází základní kostra. Před začátkem vytváření balíčku je třeba v této kostře doplnit u značky **Product** hodnotu atributu **Manufacturer**. Bez doplnění této hodnoty sestavení instalátoru selže. Dále je možné upravit různé atributy u dalších značek. Například atributem **Name** u značky **Directory**, která má **INSTALLFOLDER** jako hodnotu atributu **Id**, lze měnit název složky, pod kterou budou na disku uloženy potřebné soubory (konkrétně knihovna s add-inem). Dále je potřeba vytvořit v bloku ohraničeném značkami **ComponentGroup** dvě nové komponenty. Jednu komponentu pro knihovnu a její registraci a druhou komponentu pro zápis do registrů Enterprise Architect. Komponenty lze vytvořit pomocí značky **Component**, kde je třeba vyplnit hodnotu atributu **Guid**. Lze vyplnit i další atributy jako například **Id** nebo **Name**. Visual Studio umožňuje **Guid** vygenerovat vybráním položky **TOOLS -> Create GUID**. Po kliknutí na tuto položku se otevře okno **Create GUID**, ve kterém je možné vybrat formát GUID a následně jej vygenerovat tlačítkem **New GUID**. Ideální formát je v tomto případě **Registry Format**. Tlačítkem **Copy** je možné vygenerované **Guid** zkopírovat. Po vložení **Guid** do kódu je třeba z hodnoty odstranit složené závorky, které ji ohraničují. Výsledný blok **ComponentGroup** zobrazuje kód 5.2.

Kód 5.2: Vytvořené komponenty v bloku ComponentGroup.

```
<ComponentGroup Id="ProductComponents" Directory="INSTALLFOLDER">
      <Component Id="Files" Guid="409A86D8-1502-4A8B-B27C-13A4CDB42A1D">

      </Component>

      <Component Id="Regs" Guid="90EC3823-2048-4769-9F9B-1029A5258EC4">

      </Component>
</ComponentGroup>
```

5.2.2 Vložení knihovny a její registrace

Od instalátoru se vyžaduje, aby do uživatelského počítače nahrál všechny soubory potřebné pro chod add-inu. Tímto souborem je knihovna, která je výstupem přeloženého a sestaveného projektu. Nachází se ve složce projektu

add-inu v lokaci **bin****Release**. Do kódu je tedy třeba vložit značku definující soubor, který se má stát součástí instalátoru.

Tuto knihovnu je také potřeba zaregistrovat do operačního systému. Registrace je pouze vložení několika hodnot do registrů operačního systému. V kapitole 5.1 byl využit pro registraci nástroj **RegAsm**, který do registrů vložil příslušné hodnoty. U instalátoru ovšem lze zařídit, aby nevyužíval pro registraci žádný vnější nástroj, ale vložil tyto hodnoty do registrů přímo.

K vytvoření kódu pro vložení a registraci knihovny je možné využít nástroj **Harvest Tool (Heat)**, který je součástí **WiX**. Tento nástroj podle [12] vygeneruje na základě vstupního souboru příslušný XML kód a uloží ho do souboru. Pro vygenerování kódu je nutné spustit příkazovou řádku a přesunout se do lokace, ve které se nachází příslušná knihovna. Kód 5.3 zobrazuje příkaz pro vygenerování potřebného kódu.

Kód 5.3: Ukázka příkazu pro spuštění nástroje Heat, jenž pro knihovnu „Addin.dll“ vygeneruje do výstupního souboru „Vystup.wxs“ příslušný kód.

```
"C:\Program Files (x86)\WiX Toolset v3.8\bin\heat.exe" file Addin.dll -out
 Vystup.wxs
```

Po spuštění příkazu 5.3 dojde k vytvoření souboru „Vystup.wxs“ ve složce s příslušnou knihovnou. Z tohoto souboru je třeba zkopírovat obsah bloku ohraničeného značkou **Component** a vložit jej do jednoho z **Component** bloků vytvořeného v projektu (viz kapitola 5.2.1). Po zkopírování je dále nutné do značky **File** přidat atribut **Name** a jako hodnotu nastavit kompletní název knihovny (např. Addin.dll). V téže značce je potřeba upravit i hodnotu atributu **Source**. Hodnotou tohoto atributu je cesta k příslušné knihovně.

5.2.3 Vložení hodnot do registrů

Aby Enterprise Architect o add-inu věděl, je třeba přidat do jeho registrů nové hodnoty. Jedná se o stejné registry a hodnoty, jejichž úprava byla popsána v kapitole 4.2. Aby uživatel nemusel registry upravovat ručně, je dobré zařídit automatickou úpravu registrů instalátorem.

Úpravu registrů je možné definovat pomocí značek **RegistryKey** a **RegistryValue**. Značka **RegistryKey** určuje, který klíč se má upravit, kde se tento klíč nachází, a upřesňuje konkrétní akci (vytvoření, mazání atd.). Značka **RegistryValue** určuje typ a obsah hodnoty příslušného klíče.

Značka **RegistryKey** obsahuje tři důležité atributy. Jedná se o atributy **Root**, **Key** a **Action**. Atribut **Root** určuje kořenový klíč registrů. Pro add-in jsou podstatné dvě hodnoty tohoto atributu. Použití konkrétní hod-

noty závisí na tom, zda má add-in být k dispozici pouze aktuálnímu uživateli, nebo zda má být k dispozici všem uživatelům daného počítače. Kořenovým klíčem pro aktuálního uživatele je hodnota **HKEY_CURRENT_USER (HKCU)** [10, s. 3014]. Kořenovým klíčem pro všechny uživatele lokálního počítače je **HKEY_LOCAL_MACHINE (HKLM)** [10, s. 3014]. Tyto hodnoty se zapisují do atributu jako zkratky (viz kód 5.4). Atribut **Key** určuje klíč, který má být upraven. Hodnotou tohoto atributu je název klíče včetně cesty, ve které se má nacházet. Hodnoto tohoto atributu je vždy **Software\Sparx Systems\EAAddins**<**název_projektu**>, kde **název_projektu** je jméno projektu add-inu ve Visual Studiu. **Action** je posledním důležitým atributem této značky. Určuje, co se má s daným registrem stát. Podle [13] může tento atribut nabývat hodnoty **create**, **createAndRemoveOnUninstall** a **none**. Defaultní hodnotou atributu je hodnota **none**, která neprovádí s registry žádnou akci. Hodnota **create** vytvoří klíč, pokud již není vytvořen. Hodnota **CreateAndRemoveOnUnistall** vytvoří při instalaci klíč, pokud již není vytvořen, a při odinstalaci tento klíč z registrů odstraní se všemi jeho hodnotami. Další možné atributy a jejich hodnoty lze nalézt v [13].

Značka **RegistryValue** obsahuje dva důležité atributy, těmito atributy jsou **Type** a **Value**. Atribut **Type** určuje typ hodnoty klíče a **Value** určuje konkrétní hodnotu. **Type** může nabývat několika hodnot. Pro účely add-inu je ovšem důležitá pouze hodnota **string**, která uloží hodnotu klíče jako řetězec. Atribut **Value** reprezentuje obsah položky **Údaj hodnoty** příslušného klíče v registrech (viz kapitola 4.2). Hodnota tohoto atributu musí být ve tvaru **<NameSpace>.<ClassName>**, kde **<NameSpace>** je celý název „namespace“ projektu a **<ClassName>** je název hlavní třídy projektu (viz kapitola 4.2).

Příklad nadefinovaných registrů pomocí výše uvedených značek ukazuje algoritmus 5.4.

Kód 5.4: Ukázka registrů definovaných pomocí značek **RegistryKey** a **RegistryValue**.

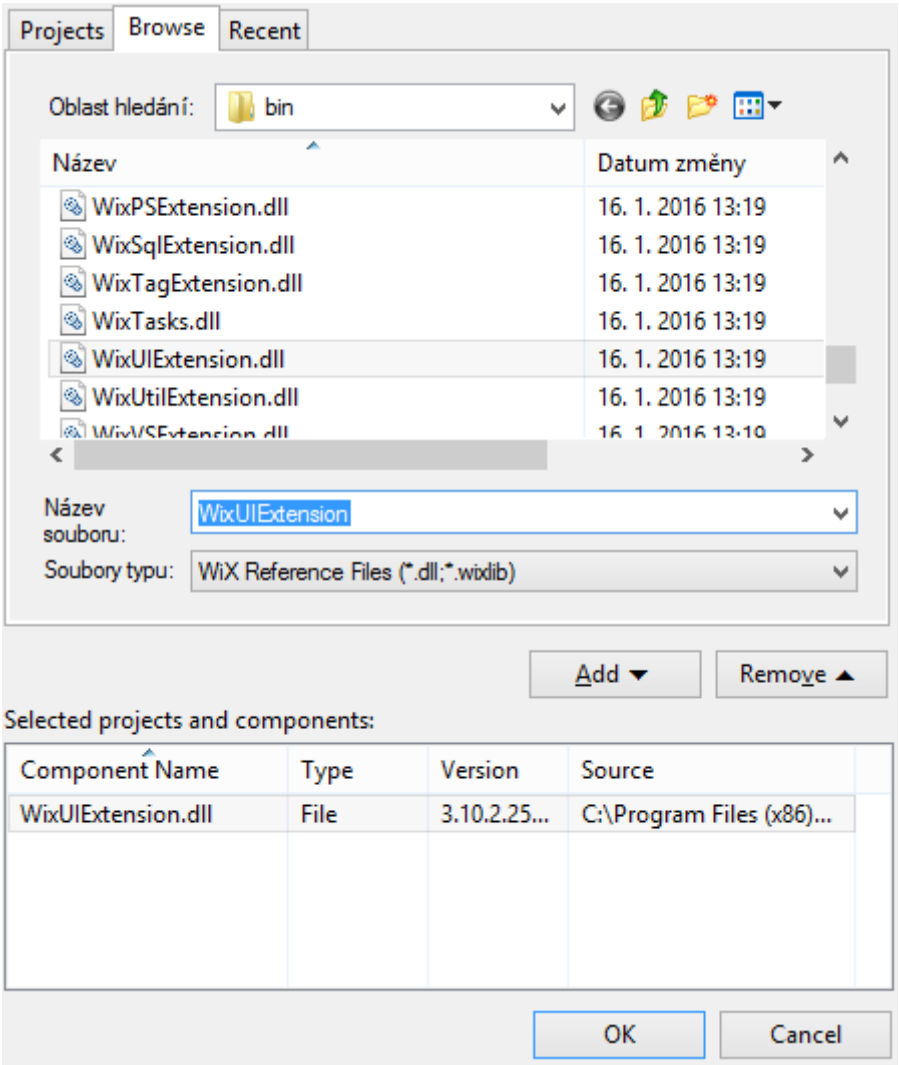
```
<RegistryKey Root="HKCU"
             Key="Software\Sparx Systems\EAAddins\Metodika_ukazka"
             Action="createAndRemoveOnUninstall">
       <RegistryValue Type="string" Value="Metodika_ukazka.Main" />
</RegistryKey>
```

5.2.4 Dialog pro výběr umístění souboru

WiX obsahuje již předdefinované vestavěné dialogové sady. Dialogových sad je pět, pro účely add-inu je ovšem dostačující sada **WixUI_InstallDir**.

WixUI_InstallDir umožňuje uživateli vybrat místo na disku, do kterého se má produkt nainstalovat. Pro použití těchto sad je potřeba přidat do projektu referenci na knihovnu **WixUIExtension**.

Referenci na knihovnu lze přidat z okna **Solution Explorer** kliknutím levým tlačítkem myši na položku **References** daného projektu. Po kliknutí se rozbalí menu. Kliknutím na položku **Add Reference** v menu lze otevřít okno pro výběr souboru, na který má reference ukazovat. Pro výběr knihovny **WixUIExtension** je nutné zvolit záložku **Browse** a v kolonce **Oblast hledání** vybrat složku **bin**. Tato složka se nachází v místě, do kterého byl WiX nainstalován. Ve složce **bin** se již nachází soubor **WixUIExtension.dll**, který je nutné vybrat. Dále je nutné tento soubor přidat do projektu kliknutím na tlačítko **Add** a potvrzením tlačítkem **OK** (viz obr. 5.2).



Obrázek 5.2: Přidání reference na knihovnu WixUIExtension.dll.

Pro implementaci dialogové sady **WixUI_InstallDir** je nutné přidat do fragmentu obsahujícího definici instalační složky (část ohraničená značkami **Directory**) značky **Property** a **UIRef**. Přesná podoba těchto značek je uvedena v kódu 5.5. Hodnota atributu **Value** značky **Property** reprezentuje **Id** definované složky, pro kterou by měl mít uživatel možnost specifikovat umístění. Hodnoty atributu **Id** obou značek jsou pevně dány.

Kód 5.5: Implementace dialogové sady **WixUI_InstallDir**.

```
<Property Id="WIXUI_INSTALLDIR" Value="INSTALLFOLDER" />
<UIRef Id="WixUI_InstallDir" />
```

5.2.5 Ověření funkčnosti instalátoru

Před předáním instalačního balíčku uživatelům je dobré ověřit, zda instalátor skutečně funguje. Správnou funkčnost lze ověřit jednoduchým spuštěním instalátoru. Před tímto ověřením je ovšem nutné odinstalovat knihovnu, která byla použita při vytváření add-inu ve Visual Studiu. Bez odinstalování knihovny nebude možné ověřit zda instalátor skutečně fugnuje. Nebude možné ověřit, zda instalátor skutečně vytvořil registry, protože tyto registry budou již vytvořeny z procesu vývoje komponenty.

K této odinstalaci stačí pouze odstranit příslušné registry daného add-inu. Registry add-inu je možné odstranit například pomocí nástroje **regedit**. Tyto registry se nachází ve větvi **HKEY_CURRENT_USERS**. V této větvi je třeba odstranit klíč nacházející se v lokaci **Software\Sparx Systems\EAAddins** (viz kapitola 4.2). Je třeba odstranit klíč, který má stejné jméno jako projekt ve Visual Studiu. Pro odstranění stačí na klíč kliknout levým tlačítkem myši a v otevřeném menu zvolit položku **Odstranit**. Po odstranění klíče by Enterprise Architect neměl vidět příslušný add-in.

Instalační balíček se po překladu a sestavení projektu nachází v umístění projektu ve složce **\bin\Debug**. K otestování jeho správné funkčnosti je nutné tento balíček spustit. Po spuštění se objeví úvodní okno s tlačítky **Back**, **Next** a **Cancel**. Pomocí těchto tlačítek je možné přesouvat se mezi okny spuštěné instalace nebo instalaci ukončit. Po dokončení instalace je třeba ověřit, že se v příslušném umístění na disku nachází knihovna add-inu. V registrech by se také měly znovu objevit klíče patřící danému add-inu. Dále je dobré otestovat, zda Enterprise Architect po instalaci skutečně add-in vidí. To lze provést spuštěním nástroje Enterprise Architect a následným spuštěním add-inu. Add-in je možné spustit z menu **EXTENSION** nebo spuštěním události, která by add-in měla vyvolat.

Po vyzkoušení instalace je třeba ověřit i funkčnost odinstalace. Odinstalaci je možné spustit opětovným spuštěním instalačního balíčku. Pokud

je add-in nainstalován, nabídne instalátor možnosti **Repair** a **Remove**. Po kliknutí na tlačítko **Remove** se spustí odinstalace. Po úspěšné odinstalaci by na disku již neměla být přítomna knihovna add-inu. V registrech by také již neměly být klíče daného add-inu. Pokud instalace i odinstalace proběhla v pořádku, je instalační balíček připraven na předání uživatelům.

6 Návrh vzorových komponent

Tato kapitola se věnuje návrhu vzorových komponent pro nástroj Enterprise Architect. Komponenty jsem navrhl s ohledem na metodiky a technologie zadavatele. Návrhy jednotlivých komponent jsou uvedeny v kapitolách 6.2, 6.3 a 6.4. V kapitole 6.1 je popsán důvod vytvoření těchto komponent.

6.1 Motivace

V projektech zadavatele se v mnoha případech vyskytuje jeden element na více místech projektu. Ruční vytváření nových elementů s totožnými parametry a jejich následná úprava je zdlouhavá a náročná na údržbu. Z tohoto důvodu vznikl požadavek na vytvoření komponent pro kopírování a aktualizaci elementů.

Jelikož elementy zkopírované komponentou pro kopírování budou mít odkaz na původní elementy, nebude muset uživatel ručně hledat každý zkopírovaný element. Komponenta pro aktualizaci nalezne všechny elementy nesoucí si v tagovaných hodnotách odkaz na příslušný element. Na základě hodnot v původním elementu přepíše informace vybraných nalezených elementů. Komponenta pro aktualizaci nebude bez komponenty pro kopírování fungovat.

Hlavním účelem těchto dvou komponent bude ulehčit uživatelům výše popsanou údržbu projektu při změnách hodnot elementů, které jsou rozkopírované ve více částech projektu. Tyto komponenty byly navrženy pro použití společně s MDG technologií `CCA_Prostředí`.

Podobným problémem jako změna hodnot rozkopírovaných elementů je i číslování nových elementů. Z tohoto důvodu vznikl také požadavek na vytvoření komponenty pro číslování elementů.

Elementy jsou v rámci projektu systematicky označovány a číslovány. Pokud uživatel založí nový element, např. požadavek, potřebuje jej očíslovat pořadovým číslem. Pokud je v celém projektu elementů příslušné skupiny několik desítek, může být zdlouhavé nalézt element s nejvyšším číslem. Třetí komponenta tedy bude mít za úkol tento proces zautomatizovat.

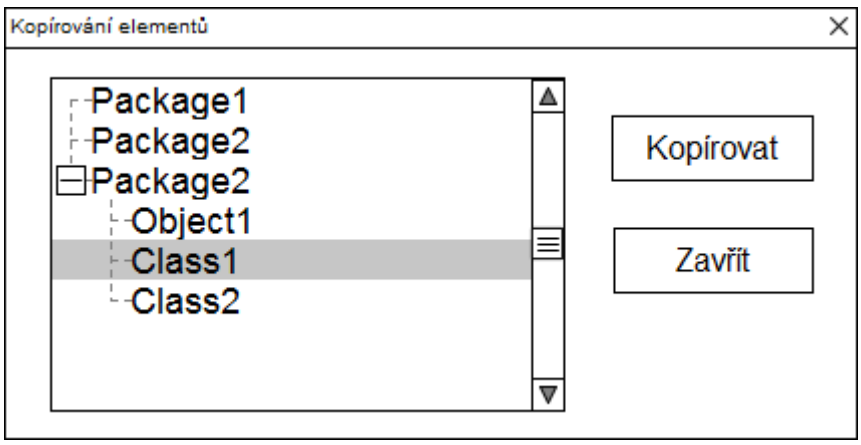
6.2 Kopírování elementů

Tato komponenta bude sloužit pro kopírování vybraného elementu. Do tagovaných hodnot kopie také přidá odkaz na původní element. Tento odkaz bude následně používat komponenta pro aktualizaci hodnot (viz kapitola 6.3).

Pro spuštění komponenty musí uživatel vytvořit nový element typu **Datový blok** přetažením z okna **Toolbox** do příslušného otevřeného diagramu. Komponenta vytvoří dialogové okno.

Dialogové okno bude zobrazovat zprávu dotazující se, zda chce uživatel element zkopírovat. Dialogové okno bude obsahovat tlačítka **Ano** a **Ne**. Pokud uživatel klikne na tlačítko **Ne**, neprovede komponenta žádnou další akci a Enterprise Architect bude dále pokračovat standardním vytvářením nového elementu. Pokud uživatel klikne na tlačítko **Ano**, vytvoří komponenta nové okno pro výběr elementu ke zkopírování.

Okno pro výběr elementu ke zkopírování bude zobrazovat elementy, které je možné zkopírovat, v podobě stromové struktury. Zkopírovat bude možné pouze elementy typu **Class** a **Object**. Elementy jiného typu nebudou ve stromové struktuře obsaženy, případně bude uživateli jiným způsobem znemožněno element vybrat. Dále bude toto okno obsahovat tlačítka pro zrušení a potvrzení výběru elementu. Pokud uživatel klikne na tlačítko pro zrušení, okno pro výběr se zavře a Enterprise Architect bude pokračovat ve standardním vytváření nového elementu. Návrh okna pro výběr elementu ke zkopírování je zobrazen na obrázku 6.1.



Obrázek 6.1: Návrh okna pro výběr elementu ke kopírování.

V případě, že uživatel ze stromové struktury vybere element a volbu potvrdí kliknutím na tlačítko pro potvrzení, okno pro výběr elementu se zavře a komponenta provede zkopírování vybraného elementu do otevřeného diagramu.

Komponenta bude kopírovat veškeré základní informace o elementu, např. název, stereotyp, poznámky a další. Dále bude kopírovat tagované hodnoty elementu, atributy elementu a tagované hodnoty jednotlivých atributů. Po zkopírování bude do tagovaných hodnot elementu přidána nová tagovaná hodnota s názvem **`RefGUID`**, která bude obsahovat globally unique identifier (GUID) původního elementu. Tato tagovaná hodnota bude přidána i ke všem zkopírovaným atributům a bude obsahovat GUID atributů původního elementu.

6.3 Aktualizace zkopírovaných elementů

Tato komponenta bude sloužit k aktualizaci zkopírovaných elementů na základě hodnot v elementu původním. Komponenta bude sloužit pouze pro aktualizaci elementů zkopírovaných pomocí komponenty pro kopírování (viz kapitola 6.2).

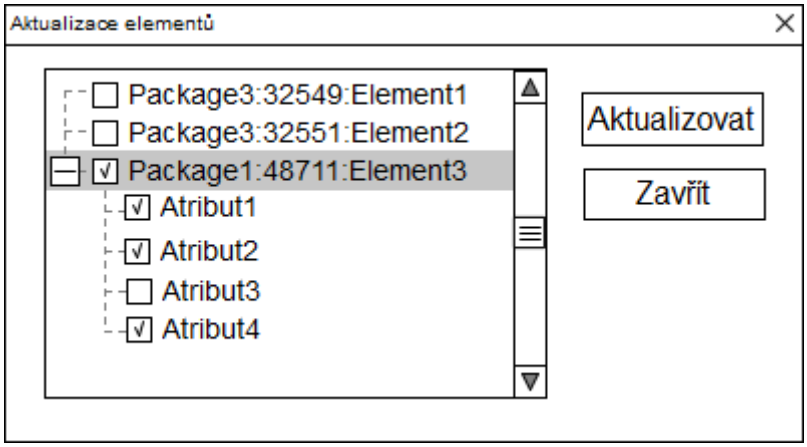
Před spuštěním komponenty bude uživatel muset označit původní element, ze kterého byly ostatní elementy zkopírovány. Element bude moci uživatel označit buď v okně **`Diagram View`**, nebo v okně **`Project Browser`**. Komponentu bude možné spustit pouze z elementů, které mají stereotyp **`table`** nebo **`datový blok`**, případně pokud jsou stereotypy **`table`** nebo **`datový blok`** součástí složeného stereotypu.

Tuto komponentu bude uživatel moci spustit z horní lišty Enterprise Architect zvolením menu **`Extensions`**. Následně uživatel bude muset zvolit položku **`CCA Aktualizace elementů -> Aktualizovat`**. Dalším možným způsobem spuštění komponenty bude kliknutí levým tlačítkem myši po označení původního elementu. Po tomto kliknutí se otevře menu, ve kterém uživatel zvolí položku **`Extensions -> CCA Aktualizace elementů -> Aktualizovat`**.

Po spuštění vyhledá komponenta všechny elementy, které obsahují referenci na označený element. Tato reference je obsažena ve formě tagované hodnoty **`RefGUID`** a její hodnota je GUID původního elementu (viz kapitola 6.2). Elementy může komponenta hledat rekurzivním procházením celého modelu projektu. Další možností k nalezení příslušných elementů je použití dotazu jazyka Structured Query Language (SQL). Výběr konkrétní metody hledání bude záviset na rychlosti. Pokud budou nalezeny elementy s přísluš-

nou referencí, najde komponenta v těchto elementech atributy obsahující tagovanou hodnotu **`RefGUID`** s odkazem na atribut v původním elementu. Po nalezení všech zkopírovaných elementů a atributů vytvoří komponenta nové okno. Pokud žádné elementy nenalezne, otevře dialogové okno s informací, že neexistují žádné zkopírované elementy.

Okno, které se vytvoří po nalezení elementů, bude obsahovat pole s nalezenými elementy. Každý element v tomto poli bude uživatel moci rozbalit. Po rozbalení se objeví jednotlivé atributy elementu. Budou zobrazeny pouze atributy, které byly zkopírovány (obsahují tagovanou hodnotu **`RefGUID`** s GUID původního elementu). Každý element bude pro snadnou identifikaci popsán pomocí názvu nadřazeného objektu, svého identifikátoru a svého názvu. Každý zobrazený atribut bude popsán svým názvem. U každého elementu a atributu bude také zaškrtávací políčko (checkbox), jehož zaškrtnutím bude moci uživatel označit elementy a atributy určené k aktualizaci. V okně se dále budou nacházet tlačítka pro potvrzení aktualizace a pro zavření okna. Návrh tohoto okna je zobrazen na obrázku 6.2.



Obrázek 6.2: Návrh okna pro výběr elementů a atributů k aktualizaci.

Pokud uživatel klikne na tlačítko pro zavření okna, komponenta pouze zavře okno a neprovede žádné změny. Po kliknutí na potvrzující tlačítko se objeví dialogové okno s kontrolním výpisem. Toto okno bude také obsahovat tlačítko pro potvrzení a tlačítko pro zrušení akce. Po zamítnutí akce uživatelem se dialogové okno zavře a okno s výběrem elementů zůstane otevřené. Uživatel tak bude mít možnost výběr změnit. Po potvrzení akce uživatelem provede komponenta aktualizaci označených elementů a jejich atributů. Aktualizace bude provedena načtením hodnot původního elementu a jeho příslušných atributů a jejich následným zapsáním do zvolených elementů a atributů. Po aktualizaci dojde k zavření okna s výběrem nalezených ele-

mentů.

Komponenta bude umožňovat také zjednodušenou verzi aktualizace pro jednotlivé atributy konkrétního elementu. Před spuštěním této aktualizace bude uživatel muset označit atribut, který bude chtít aktualizovat. Atribut bude moci uživatel v tomto případě označit pouze v okně **Project Browser**. Po označení bude muset kliknutím pravým tlačítkem myši otevřít menu. Po otevření menu spustí aktualizaci atributu zvolením položky **Extensions -> CCA Aktualizace elementů -> Aktualizovat**.

Po spuštění tohoto druhu aktualizace vyhledá komponenta pomocí tagované hodnoty **RefGUID** označeného atributu atribut původní. Pokud původní atribut nebude nalezen, zobrazí komponenta o této události informaci. V případě nalezení původního atributu načte komponenta jeho hodnoty a následně je zkopíruje do atributu zvoleného uživatelem.

6.4 Číslování elementů

Tato komponenta bude sloužit k očíslování elementu. Aby element mohl být očíslován, bude muset mít vyplněn prefix, případně i popisek v názvu nebo v aliasu. Tento prefix slouží komponentě jako maska pro vyhledání posledního použitého čísla.

Před spuštěním této komponenty bude uživatel muset vyplnit název nebo alias elementu. Název a alias se skládá z prefixu, oddělovacího znaku a popisku. Součástí prefixu je pořadové číslo. Prefix může obsahovat i jiné číslo než pořadové. Za tímto číslem musí být vždy umístěn jiný znak, než je číslice nebo znak oddělovací. Před oddělovacím znakem musí být vždy mezera. Pro použití této komponenty uživatel nebude do prefixu pořadové číslo vyplňovat. Název bez pořadového čísla zadaný uživatelem může vypadat například takto: „UC-PD0154F - událost na komponentě aplikace“. Prefixem bez pořadového čísla je „UC-PD0154F“. Pořadové číslo bude doplňovat komponenta. Oddělovacím znakem je v tomto případě pomlčka. Text za oddělovacím znakem je popisek. Struktura názvů a aliasů elementů je stanovena metodikou firmy.

Po vyplnění názvu nebo aliasu musí uživatel element označit, pokud již není označen. Element může být označen v okně **Diagram View** nebo v okně **Project Browser**. Následně bude uživatel moci komponentu spustit z horní lišty Enterprise Architect kliknutím na položku **Extensions -> CCA Číslování elementů -> Číslovat**. Dalším způsobem spuštění bude možnost pravým tlačítkem myši vyvolat menu a v něm opět zvolit položku **Extensions -> CCA Číslování elementů -> Číslovat**.

Po spuštění komponenta načte označený element. Pokud načtený objekt není elementem nebo jej není možné načíst, zobrazí komponenta uživateli informaci o nemožnosti očíslování objektu. Při úspěšném načtení označeného elementu načte komponenta hodnoty názvu a aliasu. Z názvu a aliasu načte prefix a vyhledá veškeré elementy, jejichž prefix je shodný. Názvy a aliasy bude komponenta porovnávat zvlášť, bude tedy možné číslovat alias nezávisle na názvu. Ze všech nalezených elementů se stejným prefixem nalezne komponenta element s největším pořadovým číslem a číslo uloží. Komponenta také zjistí počet číslic uloženého čísla. Toto číslo poté zvětší o jedna a vloží jej za prefix před oddělovací znak vybraného elementu. Číslo vložené za prefix bude mít stejný počet číslic jako číslo, z něhož bylo získané. Znaky budou případně do počtu doplněny zleva nulami. Oddělovací znak může být pomlčka nebo dvojtečka. Tímto způsobem bude číslován název i alias vybraného elementu.

Pokud žádné elementy se stejným prefixem nebudou nalezeny nebo pokud vybraný element bude již očíslován, neprovede komponenta žádnou akci.

7 Implementace vzorových komponent

Tato kapitola je věnována popisu implementace komponent navržených v kapitole 6. Hlavním účelem implementace vzorových komponent bylo ověření správnosti metodiky, kterou jsem sepsal v kapitolách 4 a 5.

Každou komponentu jsem vytvořil jako samostatný projekt. Dalším samostatným projektem je instalační balíček. Projekty s komponentami a instalačním balíčkem jsou obsaženy ve společném solution nástroje Visual Studio. Každý projekt s komponentou jsem vytvořil podle kapitoly 4.1. Podle kapitoly 4.2 jsem vložil klíč do registrů pro každou komponentu. Dále jsem všechny komponenty podle kapitoly 4.3 připojil k rozhraní Automation nástroje Enterprise Architect. Další postup implementace je popsán jednotlivě pro každou komponentu. Kapitola 7.1 popisuje implementaci komponenty pro kopírování elementů. Implementaci komponenty pro aktualizaci elementů popisuje kapitola 7.2. Implementace komponenty pro číslování popisuje kapitola 7.3.

Projekt s instalačním balíčkem jsem vytvářel podle kapitoly 5.2.1. Implementace instalačního balíčku je popsána v kapitole 7.4.

7.1 Kopírování elementů

Projekt s touto komponentou jsem pojmenoval **`DragAndDropCopying`**. Projekt obsahuje třídy **`Main`**, **`EAOperations`** a **`ElementSelectionForm`**. Každá třída má samostatný stejně pojmenovaný soubor.

Třída **`Main`** je hlavní třídou projektu. V této třídě jsem podle kapitoly 4.4 implementoval metody pro zachycení událostí **`EA_Connect`** a **`EA_Disconnect`**. Podle návrhu této komponenty (viz kapitola 6.2) jsem do třídy implementoval metodu pro zachycení události **`EA_onPostNewElement`**. Tato metoda je omezena pouze na elementy typu **`Datový blok`** a spouští další činnost komponenty.

Třída **`EAOperations`** obsahuje metody, které přímo pracují s objekty Enterprise Architect. V této komponentě se jedná o všechny metody potřebné ke zkopírování vybraného elementu. Enterprise Architect bohužel neobsahuje žádnou metodu pro zkopírování elementu. Bylo tedy nutné celou proceduru kopírování naprogramovat. V této třídě se také nachází metody

pro vložení tagované hodnoty **RefGUID** do tagovaných hodnot elementu a jeho atributů. Datový typ tagované hodnoty elementů a datový typ tagované hodnoty atributů se liší. Pro tagovanou hodnotu elementů je datový typ **EA.TaggedValue**, pro tagovanou hodnotu atributu je datový typ **EA.AttributeTag**. Z tohoto důvodu jsem musel implementovat metodu pro vložení **RefGUID** samostatně pro atributy a samostatně pro elementy.

Třída **ElementSelectionForm** slouží k vytvoření dialogového okna s dotazem, zda chce uživatel element zkopírovat. Instance této třídy se vytvoří pouze v případě vyvolání události **EA_onPostNewElement** nad elementem typu **datový blok**. Dialog obsahuje možnost odsouhlasit, případně zamítnout kopírování elementu. Při zvolení souhlasu ke kopírování se otevře okno pro výběr elementu. Toto okno jsem implementovat nemusel, protože jej již obsahuje třída **Repository** nástroje Enterprise Architect a poskytuje jej k použití. Metoda poskytovaná třídou **Repository** se jmenuje **InvokeConstructPicker** a jejím vstupním parametrem je řetězec, kterým je možné omezit zobrazené objekty podle typu. Tato metoda také vrací celočíselný identifikátor vybraného objektu. Podle návrhu 6.2 jsem omezil elementy pouze na typy **Object** a **Class**.

Volání tříd a důležitých metod této komponenty znázorňuje diagram A.1 v příloze A.

Výstupem projektu je soubor **DragAndDropCopying.dll**. Tento soubor je komponentou ve formě knihovny a nachází se v umístění projektu ve složce **\bin\Release**.

7.2 Aktualizace zkopírovaných elementů

Projekt této komponenty se jmenuje **ElementUpdate** a obsahuje třídy **Main**, **EAOperations**, **ElementsSelectionForm** a **ConfirmForm**.

Třída **Main** je hlavní třídou projektu. V této třídě jsem implementoval metody všech důležitých událostí popsaných v kapitole 4.4. Dále třída obsahuje metodu **update**, která vytváří instanci třídy **EAOperations** a zajišťuje další činnost komponenty. Tato metoda je zavolána v případě, že nastane událost **EA_MenuClick** (viz kapitola 4.4.4).

Ve třídě **EAOperations** se nachází všechny metody pracující s objekty Enterprise Architect. Jedná se o všechny metody potřebné pro aktualizaci elementů. Metody, které aktualizují hodnoty v konkrétním elementu, jsou totožné s metodami třídy **EAOperations** komponenty pro kopírování. Dále se v této třídě také nachází metoda pro získání všech elementů, které obsahují v tagovaných hodnotách **RefGUID** označeného elementu. Pro nalezení

příslušných elementů jsem zvolil použití dotazu jazyka SQL. Rekurzivní prohledávání celého modelu projektu jsem vyzkoušel. V komponentě jsem jej nepoužil, protože bylo příliš pomalé. SQL dotaz pro získání těchto elementů zobrazuje kód 7.1.

Kód 7.1: SQL dotaz pro nalezení všech elementů, které obsahují **RefGUID** označeného elementu. Hodnota **element_ref** je GUID označeného elementu.

```
SELECT o.name, o.object_id
FROM t_objectproperties v, t_object o
WHERE v.object_id = o.object_id
AND v.property = 'RefGUID'
AND v.value = '" + element_ref + "'"
```

Třída **ElementsSelectionForm** obsahuje metody potřebné k vytvoření formuláře pro výběr elementů a atributů, které uživatel bude chtít aktualizovat. Enterprise Architect bohužel neposkytuje žádné okno, které by bylo vhodné pro účely této komponenty. Formulář jsem tedy musel vytvořit podle návrhu v kapitole 6.3. Formulář vytváří metoda **createForm**. Z této třídy se také vytváří instance třídy **ConfirmForm**.

Třída **ConfirmForm** obsahuje metody potřebné k vytvoření potvrzovacího dialogového okna. Formulář vytváří metoda **createForm**. Dále třída obsahuje metody **yesClick** a **noClick**. Tyto metody se spouští v případě kliknutí na potvrzovací nebo zamítací tlačítko v tomto formuláři.

Volání tříd a důležitých metod této komponenty znázorňuje diagram A.2 v příloze A.

Výstupem projektu je soubor **ElementUpdate.dll**. Tento soubor je komponentou ve formě knihovny. Soubor se nachází v umístění projektu ve složce **\bin\Release**.

7.3 Číslování elementů

Projekt s komponentou pro číslování elementů se jmenuje **Numbering** a obsahuje třídy **Main**, **EAOperations**, **NameNumbering** a **AliasNumbering**.

Hlavní třídou projektu je třída **Main**. V této třídě jsem implementoval všechny metody pro zachycení událostí podle kapitoly 4.4. V případě, že nastane událost **EA_MenuClick**, vytvoří metoda pro zpracování této události instanci třídy **EAOperations** a spustí další činnost komponenty.

Třída **EAOperations** obsahuje metody, které přímo pracují s objekty Enterprise Architect. V této třídě se nachází metody, které nastaví nové, již očíslované jméno nebo alias označenému elementu.

Ve třídě **NameNumbering** se nachází metody pro sestavení jména elementu včetně pořadového čísla. Metody v této třídě získají z nalezených jmen elementů jméno s nejvyšším číslem. Dále na základě nalezeného čísla sestaví nové jméno pro element označený uživatelem. Třída také obsahuje metodu, která nalezne všechna jména elementů s příslušným prefixem. Metoda hledá jména pomocí SQL dotazu (viz kód 7.2).

Kód 7.2: SQL dotaz pro nalezení všech jmen elementů s příslušným prefixem. Hodnota **name_prefix** je prefix hledaný ve jméně elementu.

```
SELECT o.Name
FROM t_object o
WHERE o.Name
LIKE '" + this.name_prefix + "*'"
```

Třída **AliasNumbering** obsahuje metody pro sestavení aliasu elementu včetně pořadového čísla. Metody v této třídě jsou podobné metodám z třídy **NameNumbering**. Požadavkem zadavatele bylo, aby číslování názvu a číslování aliasu bylo jednoznačně odděleno. Metody v této třídě získají z nalezených elementů element s nejvyšším číslem v aliasu. Dále na základě nalezeného čísla sestaví nový alias pro element označený uživatelem. Třída dále obsahuje metodu pro nalezení všech aliasů elementů s příslušným prefixem. Stejně jako v případě hledání jmen jsou aliasy elementů hledány pomocí SQL dotazu (viz kód 7.3).

Kód 7.3: SQL dotaz pro nalezení všech aliasů elementů s příslušným prefixem. Hodnota **alias_prefix** je prefix hledaný ve jméně elementu.

```
SELECT o.Alias
FROM t_object o
WHERE o.Alias
LIKE '" + this.alias_prefix + "*'"
```

Volání tříd a důležitých metod této komponenty znázorňuje diagram A.3 v příloze A.

Výstupem projektu je soubor **Numbering.dll**. Tento soubor je komponentou ve formě knihovny. Soubor se nachází v umístění projektu ve složce **\bin\Release**.

7.4 Instalační balíček

Projekt s instalačním balíčkem má název **CCA_EA_ReusableComponents**. Instalační balíček obsahuje všechny komponenty navržené v kapitole 6. V projektu se nachází jediný soubor s názvem **Product**.

V souboru **Product** jsem implementoval podle kapitoly 5.2.2 soubory s knihovnou pro každou komponentu. Každá knihovna je uzavřena ve svém vlastním **Component** bloku. Dále jsem také každé knihovně podle kapitoly 5.2.3 definoval registry. Definice registrů je opět vložena pro každou knihovnu v samostatném **Component** bloku. Podle kapitoly 5.2.4 jsem implementoval dialog pro výběr umístění souborů na disku.

Výstupem projektu je soubor **CCA_EA_ReusableComponents.msi**. Tento soubor je spustitelným instalačním balíčkem a nachází se v umístění projektu ve složce **bin****Release**.

8 Testování vzorových komponent

Testování probíhalo podle standardizovaných postupů zadavatele. Pro otestování jednotlivých komponent jsem provedl funkční testy podle případů užití. Případy užití jsou popsány pro jednotlivé komponenty v kapitolách níže. Instalační balíček jsem otestoval podle kapitoly 5.2.5.

Správnost metodiky vytvořené v kapitole 4 jsem ověřil vytvořením komponent navržených v kapitole 6 podle této metodiky. Správnost metodiky pro distribuci vytvořené v kapitole 5 jsem ověřil vytvořením instalačního balíčku podle této metodiky.

Komponenty prošly všemi funkčními testy provedenými podle případů užití, které jsou sepsány v kapitolách níže. Tím že byly komponenty úspěšně otestovány, byla potvrzena i správnost vytvořené metodiky.

8.1 DragAndDropCopying

Případy užití pro tuto komponentu jsou uvedeny v tabulkách 8.2, 8.3 a 8.4.

Před testováním podle výše zmíněných případů užití musí být splněny některé podmínky. Tyto podmínky jsou uvedeny v tabulce 8.1.

Vstupní podmínky	
Pořadí	Podmínka
1	V Enterprise Architect je nainstalována MDG technologie `CCA_Prostredi`.

Tabulka 8.1: Vstupní podmínky.

TST001 - zamítnutí kopírování		
Krok	Popis	Očekávaný výsledek
1	Uživatel přetáhne „datový blok“ z toolboxu „CCA Screens“ do otevřeného diagramu.	Komponenta otevře dialogové okno s dotazem, zda si uživatel přeje element zkopírovat.

2	Uživatel klikne na tlačítko „Ne“.	Komponenta zavře dialogové okno. Komponenta neprovede žádnou další akci. Enterprise Architect otevře nastavení nově vytvořeného elementu.

Tabulka 8.2: Případ užití TST001.

TST002 - zkopírování vybraných elementů		
Krok	Popis	Očekávaný výsledek
1	Uživatel přetáhne z toolboxu „CCA Screens“ element „datový blok“ do otevřeného diagramu.	Komponenta otevře dialogové okno s dotazem, zda si uživatel přeje element zkopírovat.
2	Uživatel klikne na tlačítko „Ano“.	Komponenta otevře okno pro výběr elementu. V okně se bude nacházet stromová struktura, ve které bude možné vybrat pouze elementy typu class a object.
3	Uživatel vybere ve stromové struktuře element typu „class“ nebo „object“. Výběr potvrdí tlačítkem „OK“.	Komponenta zkopíruje vybraný element do diagramu, do kterého uživatel původně přetáhl „datový blok“. Komponenta také přidá do kopie elementu novou tagovanou hodnotu s názvem „RefGUID“, která bude ukazovat na původní objekt. Stejnou tagovanou hodnotu také komponenta přidá ke všem zkopírovaným atributům.

Tabulka 8.3: Případ užití TST002.

TST003 - zrušení výběru elementů		
Krok	Popis	Očekávaný výsledek
1	Uživatel přetáhne z toolboxu „CCA Screens“ element „datový blok“ do otevřeného diagramu.	Komponenta otevře dialogové okno s dotazem, zda si uživatel přeje element zkopírovat.
2	Uživatel klikne na tlačítko „Ano“.	Komponenta otevře okno pro výběr elementu. V okně se bude nacházet stromová struktura, ve které bude možné vybrat pouze elementy typu class a object.
3	Uživatel klikne na tlačítko „Storno“.	Komponenta zavře okno pro výběr elementu. Komponenta neprovede žádnou další akci. Enterprise Architect otevře nastavení nově vytvořeného elementu.

Tabulka 8.4: Případ užití TST003.

8.2 ElementUpdate

Případy užití pro testování této komponenty jsou uvedeny v tabulkách 8.6, 8.7, 8.8, 8.9 a 8.10.

Před testováním této komponenty musí být splněny podmínky uvedené v tabulce 8.5.

Vstupní podmínky	
Pořadí	Podmínka
1	V Enterprise Architectu je nainstálována MDG technologie **CCA_Prostredi**.
2	V projektu se nachází elementy se stereotypem **datový blok** nebo **table**, které byly zkopírovány pomocí komponenty **DragAndDropCopying**. Každý element musí obsahovat minimálně jeden atribut.

Tabulka 8.5: Vstupní podmínky.

TST004 - elementy, které nelze aktualizovat		
Krok	Popis	Očekávaný výsledek
1	Uživatel označí element, ze kterého nebyly zkopírovány žádné elementy. Po označení elementu zvolí v menu **Extensions** položku **CCA Aktualizace elementů -> Aktualizovat**.	Komponenta otevře upozornění, že není možné aktualizovat žádný element.

Tabulka 8.6: Případ užití TST004.

TST005 - zrušení aktualizace		
Krok	Popis	Očekávaný výsledek
1	Uživatel označí element stereotypu **datový blok** nebo **table**, ze kterého byly zkopírovány elementy do více částí projektu. Po označení elementu zvolí v menu **Extensions** položku **CCA Aktualizace elementů -> Aktualizovat**.	Komponenta otevře okno pro výběr elementů a atributů k aktualizaci. Okno bude obsahovat elementy s atributy zobrazenými ve stromové struktuře. Dále bude obsahovat tlačítka pro aktualizaci a zrušení.
2	Uživatel vybere některé elementy a atributy.	U vybraných elementů a atributů bude zaškrtnutý checkbox.
3	Uživatel klikne na tlačítko „Zavřít“.	Komponenta zavře okno pro výběr elementů. Komponenta neprovede žádnou další akci.

Tabulka 8.7: Případ užití TST005.

TST006 - změna výběru elementů a jejich následná aktualizace		
Krok	Popis	Očekávaný výsledek
1	Uživatel označí element stereotypu **datový blok** nebo **table**, ze kterého byly zkopírovány elementy do více částí projektu. V elementu provede změny některých hodnot a atributů. Po označení elementu a provedení změn zvolí v menu **Extensions** položku **CCA Aktualizace elementů -> Aktualizovat**.	Komponenta otevře okno pro výběr elementů a atributů k aktualizaci. Okno bude obsahovat elementy s atributy zobrazené ve stromové struktuře. Dále bude obsahovat tlačítko pro aktualizaci a zrušení.
2	Uživatel vybere některé elementy a atributy.	U vybraných elementů a atributů bude zaškrtnutý checkbox.
3	Uživatel klikne na tlačítko „Aktualizovat“.	Komponenta otevře okno s kontrolním výpisem vybraných elementů a atributů. Okno bude obsahovat tlačítka pro potvrzení a zamítnutí.
4	Uživatel klikne na tlačítko „Ne“.	Komponenta zavře okno s kontrolním výpisem. Okno pro výběr elementů zůstane otevřené. Výběr elementů a atributů zůstane nezměněný.
5	Uživatel klikne na tlačítko „Aktualizovat“.	Komponenta otevře okno s kontrolním výpisem vybraných elementů a atributů. Okno bude obsahovat tlačítka pro potvrzení a zamítnutí.
6	Uživatel klikne na tlačítko „Ano“.	Komponenta zavře okno s kontrolním výpisem. Dále komponenta zavře okno pro výběr elementů. Po zavření oken provede aktualizaci vybraných elementů a atributů.

Tabulka 8.8: Případ užití TST006.

TST007 - zrušení aktualizace atributu		
Krok	Popis	Očekávaný výsledek
1	Uživatel v okně **Project Broweser** označí atribut elementu, který byl zkopírován. Po označní atributu zvolí v menu **Extensions** položku **CCA Aktualizace elementů -> Aktualizovat**.	Komponenta otevře dialogové okno s dotazem, zda chce uživatel vybraný atribut aktualizovat.
2	Uživatel klikne na tlačítko „Ne“.	Komponenta zavře dialogové okno. Dále neprovede žádnou další akci.

Tabulka 8.9: Případ užití TST007.

TST008 - aktualizace atributu		
Krok	Popis	Očekávaný výsledek
1	Uživatel v okně **Project Browser** označí atribut elementu, který byl zkopírován. Po označení atributu zvolí v menu **Extensions** položku **CCA Aktualizace elementů -> Aktualizovat**.	Komponenta otevře dialogové okno s dotazem, zda chce uživatel vybraný atribut aktualizovat.
2	Uživatel klikne na tlačítko „Ano“.	Komponenta zavře dialogové okno a následně provede aktualizaci označeného elementu.

Tabulka 8.10: Případ užití TST008.

8.3 Numbering

Případy užití pro otestování komponenty **Numbering** jsou uvedeny v tabulkách 8.12, 8.13, 8.14 a 8.15.

Před testováním této komponenty musí být splněny podmínky uvedené v tabulce 8.11.

Vstupní podmínky	
Pořadí	Podmínka
1	V projektu se musí nacházet alespoň jeden očíslovaný element s požadovaným prefixem.

Tabulka 8.11: Vstupní podmínky.

TST009 - číslování názvu a aliasu		
Krok	Popis	Očekávaný výsledek
1	Uživatel vytvoří nový element typu class. Do názvu a aliasu zadá „TST050F - Testovací případ“. Vytvořený element označí. Spustí komponentu kliknutím na položku **Extensions -> CCA Číslování elementů -> Číslovat**.	Komponenta doplní v názvu a aliasu za „TST050F“ pořadové číslo. Výsledný název a alias může vypadat např. takto: „TST050F002 - Testovací případ“.

Tabulka 8.12: Případ užití TST009.

TST010 - číslování názvu samostatně		
Krok	Popis	Očekávaný výsledek
1	Uživatel vytvoří nový element. Do názvu zadá „TST050F - Testovací případ“. Vytvořený element označí. Spustí komponentu kliknutím na položku **Extensions -> CCA Číslování elementů -> Číslovat**.	Komponenta doplní v názvu za „TST050F“ pořadové číslo. Výsledný název může vypadat např. takto: „TST050F002 - Testovací případ“.

Tabulka 8.13: Případ užití TST010.

TST011 - číslování aliasu samostatně		
Krok	Popis	Očekávaný výsledek
1	Uživatel vytvoří nový element typu class. Do aliasu zadá „TST050F - Testovací případ“. Vytvořený element označí. Spustí komponentu kliknutím na položku **Extensions -> CCA Číslování elementů -> Číslovat**.	Komponenta doplní v aliasu za „TST050F“ pořadové číslo. Výsledný alias může vypadat např. takto: „TST050F002 - Testovací případ“.

Tabulka 8.14: Případ užití TST011.

TST012 - číslování objektů, které nejsou elementy		
Krok	Popis	Očekávaný výsledek
1	Uživatel označí objekt, který není elementem. Spustí komponentu kliknutím na položku **Extensions -> CCA Číslování elementů -> Číslovat**.	Komponenta zobrazí chybovou hlášku s upozorněním, že vybraný objekt nelze očíslovat.

Tabulka 8.15: Případ užití TST012.

9 Závěr

V rámci své práce jsem se podrobně seznámil s nástrojem Enterprise Architect verze 11 a 12. Při seznamování s nástrojem jsem se zaměřil zejména na možnosti rozšíření jeho funkčnosti. Část své práce jsem věnoval studiu vytváření add-inů a studiu jejich distribuce. Nejvíce času jsem věnoval vytváření metodiky, která popisuje tvorbu add-inů a jejich distribuci ke koncovému uživateli. Metodiku jsem vytvořil zejména na základě svých vědomostí, které jsem získal při studiu této problematiky a na základě oficiální dokumentace nástroje Enterprise Architect.

Pro ověření správnosti metodiky jsem navrhl komponenty, které jsem následně podle mnou vytvořené metodiky implementoval. Funkčnost jednotlivých komponent jsem ověřil pomocí funkčních testů, ke kterým jsem vytvořil případy užití. Funkčnímy testy komponenty prošly. Komponenty kromě ověření metodiky měly také ulehčit zaměstnancům firmy některé úkony prováděné v nástroji Enterprise Architect. Po otestování tedy byly komponenty nasazeny do rutinního provozu pro použití při navrhování nových informačních systémů analytiky. V budoucnu je možné na základě námětů uživatelů komponenty upravit. Funkčnost komponent vytvořených podle metodiky prokázala její správnost.

Podle metodiky mohou vývojáři snadněji vytvářet další add-iny pro nástroj Enterprise Architect.

Seznam zkratek

BMNI Business Process Managment Initiative. 11

BPMN Business Process Modeling Notation. 9–11

COM Component Object Model. 17, 21, 22, 24, 32

GUID globally unique identifier. 41, 42, 47, 68

Heat Harvest Tool. 34

HKCU HKEY_CURRENT_USER. 35

HKLM HKEY_LOCAL_MACHINE. 35

MDG Model Driven Generation. 17–20, 39, 50, 52, 66–68

MSI Microsoft Windows Installer. 21

OLE Object Linking and Embedding. 21, 22

OMG Object Managment Group. 10

RegAsm Assembly Registration Tool. 31, 32, 34

SoaML Service Oriented Architecture Modeling Language. 10

SQL Structured Query Language. 41, 47, 48

SysML Systems Modeling Language. 10

UML Unified Modeling Language. 9–11, 18–20

WiX Windows Installer XML Toolset. 32, 34–36

XML Extensible Markup Language. 32–34

Literatura

[1] Christian, N. – aj. *C# 2008: Programujeme profesionálně*. Computer Press a.s., 2009. 1112 s. ISBN 978-80-251-2401-7.

[2] *Představení nástroje Enterprise Architect* [online]. EA Team, 2012. [cit. 2015/12/13]. Dostupné z: http://www.enterprise-architect.cz/blog/115.

[3] Jim, A. – Ila, N. *UML2 a unifikovaný proces vývoje aplikací*. Computer Press a.s., 2011. 568 s. ISBN 978-80-251-1503-9.

[4] *ActiveX Clients* [online]. Microsoft, 2015. [cit. 2015/12/18]. Microsoft Developer Network. Dostupné z: https://msdn.microsoft.com/en-us/library/windows/desktop/ms221336(v=vs.85).aspx.

[5] *ActiveX Objects* [online]. Microsoft, 2015. [cit. 2015/12/18]. Microsoft Developer Network. Dostupné z: https://msdn.microsoft.com/en-us/library/windows/desktop/ms221401(v=vs.85).aspx.

[6] *JScript (ECMAScript3)* [online]. Microsoft, 2015. [cit. 2015/12/18]. Microsoft Developer Network. Dostupné z: https://msdn.microsoft.com/en-us/library/hbxc2t98(v=vs.84).aspx.

[7] *Automation* [online]. Microsoft, 2015. [cit. 2015/12/18]. Microsoft Developer Network. Dostupné z: https://msdn.microsoft.com/en-us/library/windows/desktop/ms221375(v=vs.85).aspx.

[8] *How to: Sign an Assembly with a Strong Name* [online]. Microsoft, 2016. [cit. 2016/4/16]. Microsoft Developer Network. Dostupné z: https://msdn.microsoft.com/en-us/library/xc31ft41(v=vs.110).aspx.

[9] *VBScript* [online]. Microsoft, 2015. [cit. 2015/12/18]. Microsoft Developer Network. Dostupné z: https://msdn.microsoft.com/en-us/library/t0aew7h6(v=vs.84).aspx.

[10] *Enterprise Architect User Guide* [online]. Sparx Systems Pty Ltd., 2014. [cit. 2015/12/15]. 3317 s. Dostupné z: http://www.sparxsystems.com.au/bin/EAUserGuide.pdf.

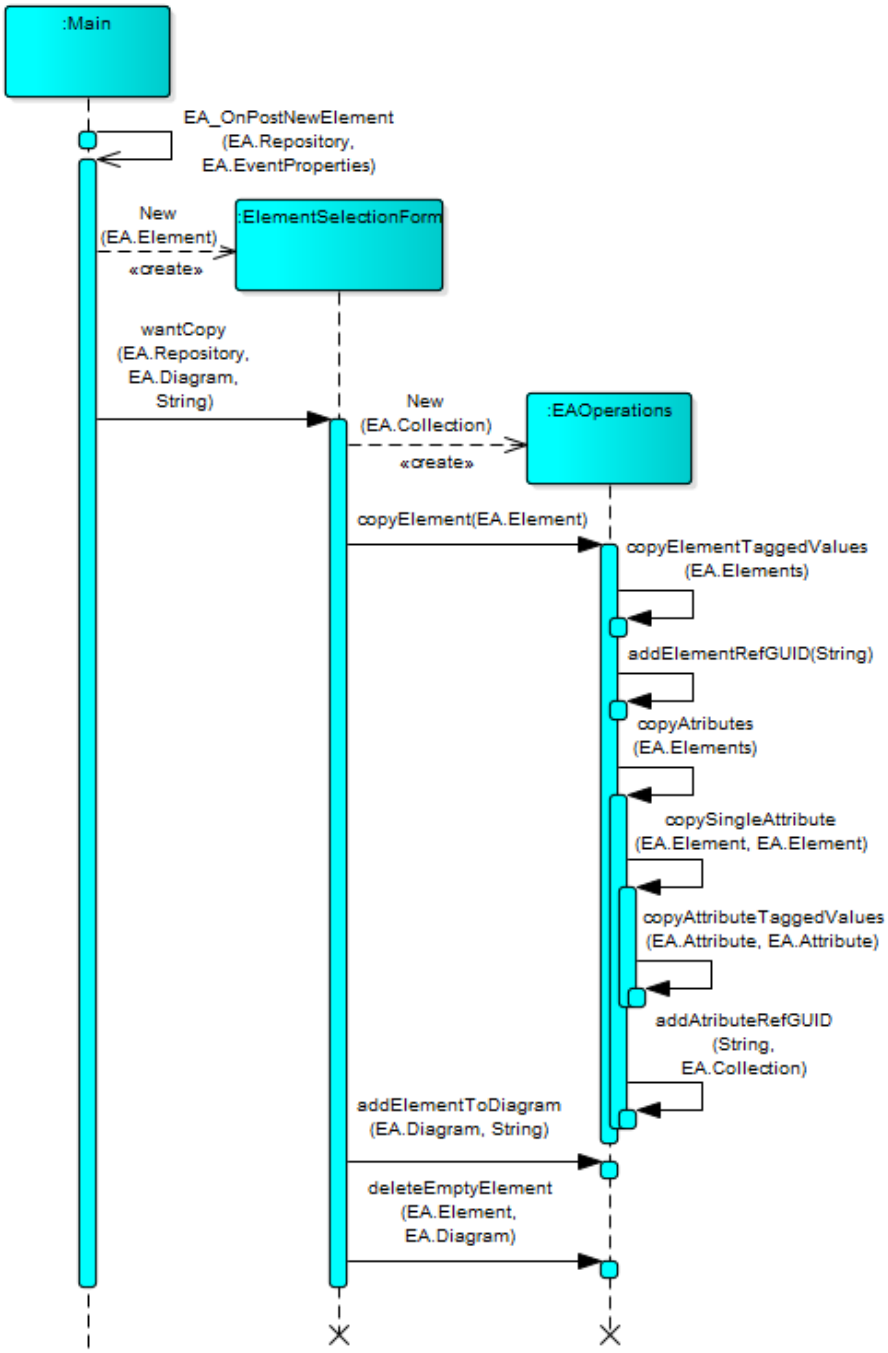
[11] Vašíček, P. *Úvod do BPMN* [online]. BPS Business Process Services s.r.o., 2008. [cit. 2015/12/14]. Dostupné z: http://bpm-sme.blogspot.cz/2008/03/3-uvod-do-bpmn.html.

[12] *Harvest Tool (Heat)* [online]. Outercurve Foundation, . [cit. 2016/3/28]. WiX Documentation. Dostupné z: `http://wixtoolset.org/documentation/manual/v3/overview/heat.html`.

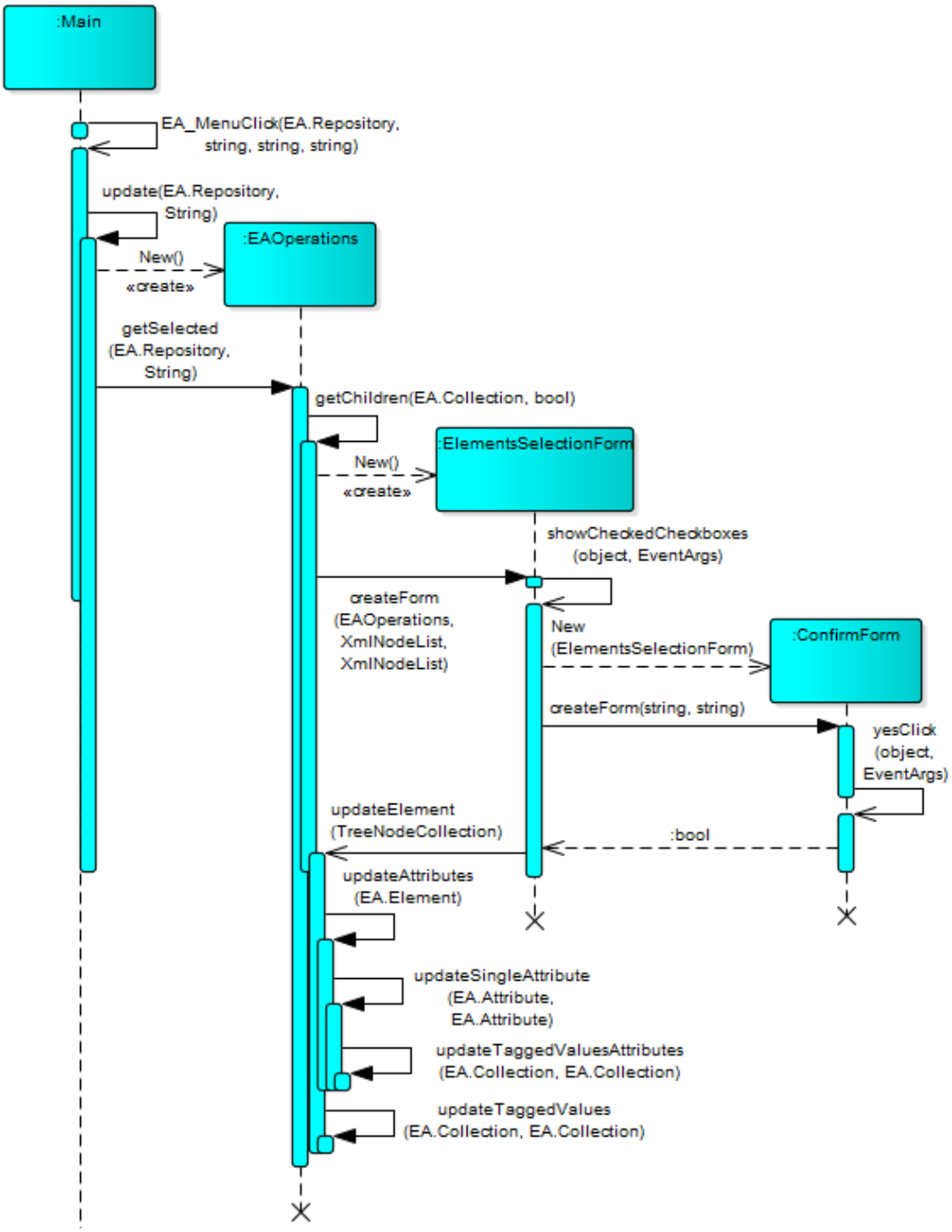
[13] *Registry Key Element* [online]. Outercurve Foundation, . [cit. 2016/4/11]. WiX Documentation. Dostupné z: `http://wixtoolset.org/documentation/manual/v3/xsd/wix/registrykey.html`.

Přílohy

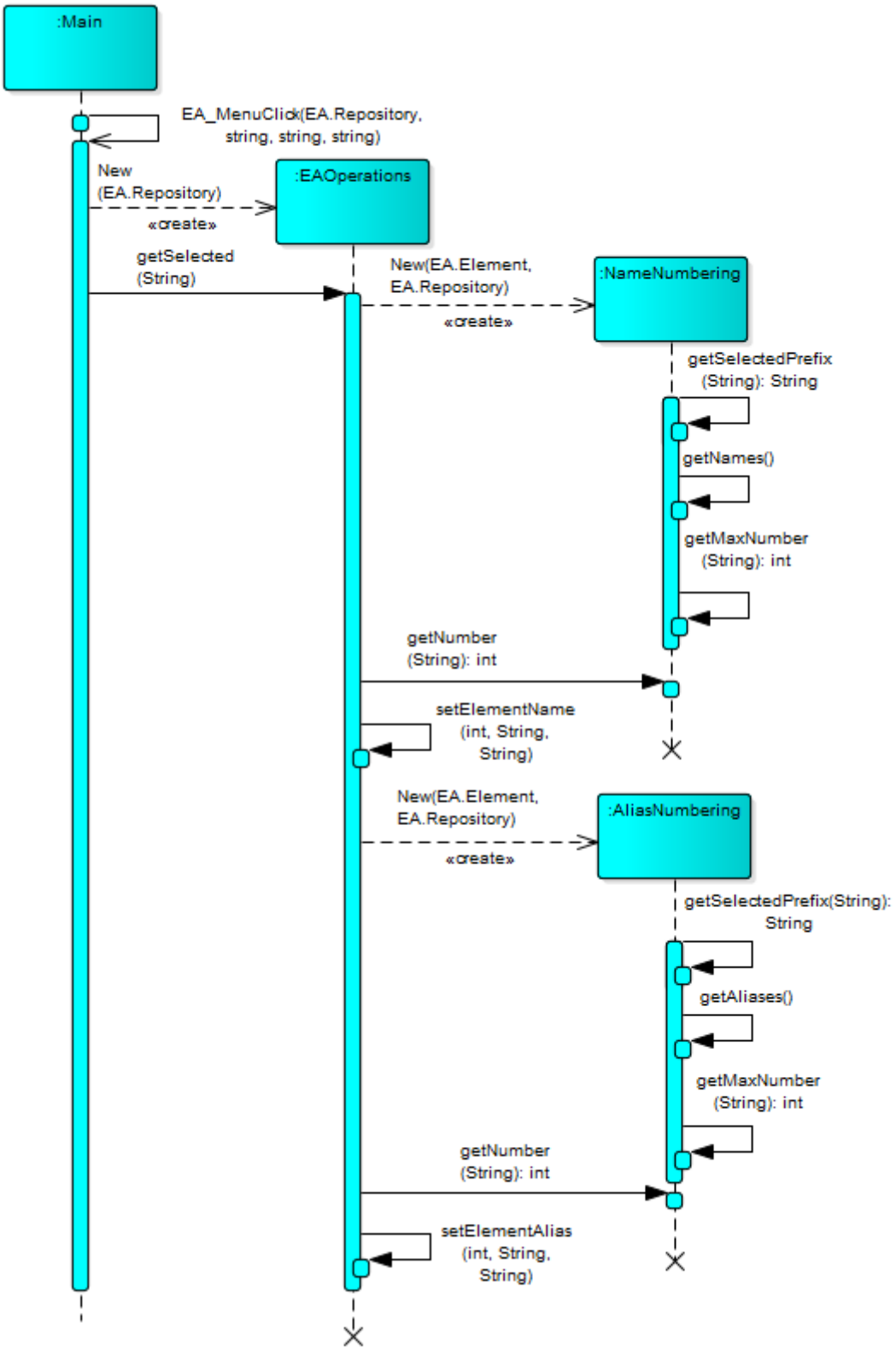
A Diagramy



Obrázek A.1: Sekvenční diagram volání tříd a důležitých metod komponenty „DragAndDropCopying“ pro případ potvrzení kopírování.



Obrázek A.2: Sekvenční diagram volání tříd a důležitých metod komponenty „ElementUpdate“ pro případ potvrzení aktualizace.



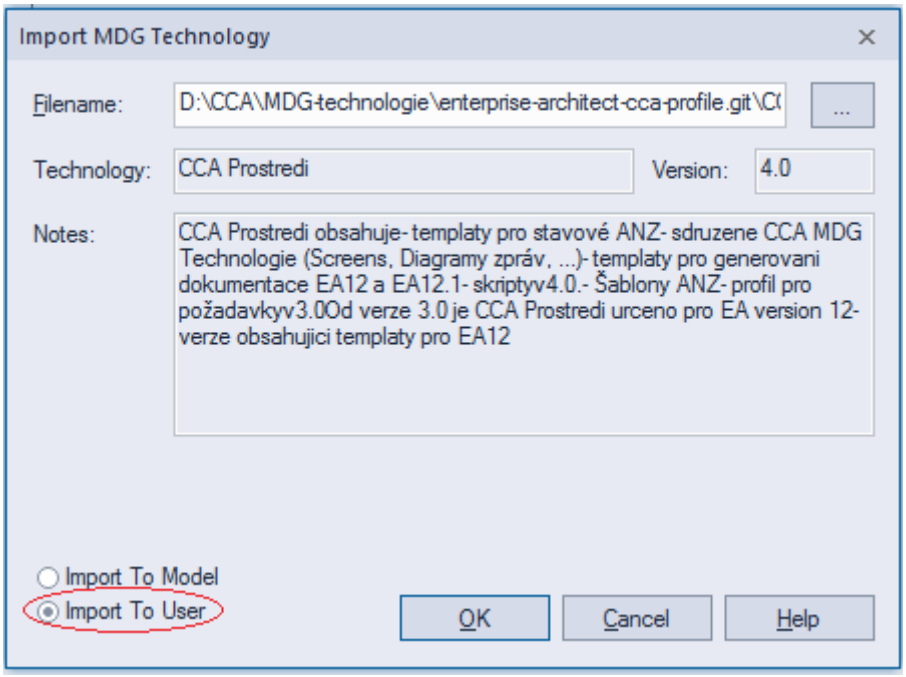
Obrázek A.3: Sekvenční diagram volání tříd a důležitých metod komponenty „Numbering“.

B Instalační příručka

Před instalací samotných komponent musí uživatel nainstalovat nástroj Enterprise Architect. Zkušební verzi tohoto nástroje je možné stáhnout z: http://www.sparxsystems.com/products/ea/trial/request.html

Po nainstalování nástroje Enterprise Architect musí uživatel nainstalovat do tohoto nástroje MDG technologii „CCA_Prostredi“ a nahrát definované typy. Všechny soubory potřebné pro instalaci, kromě nástroje Enterprise Architect, se nachází na přiloženém CD, včetně vzorového projektu (viz příloha D). Pro instalaci této MDG technologie a nahrání definovaných typů musí uživatel postupovat podle následujících kroků:

- Spustit Enterprise Architect a otevřít libovolný (vzorový) projekt.
- Otevřít **Project -> MDG Technology Import**
- Vybrat soubor **CCA_Prostredi.xml** a zaškrtnout „Import To User“. Výběr potvrdit tlačítkem **OK** (viz obr. B.1).



Obrázek B.1: Okno pro import MDG technologie.

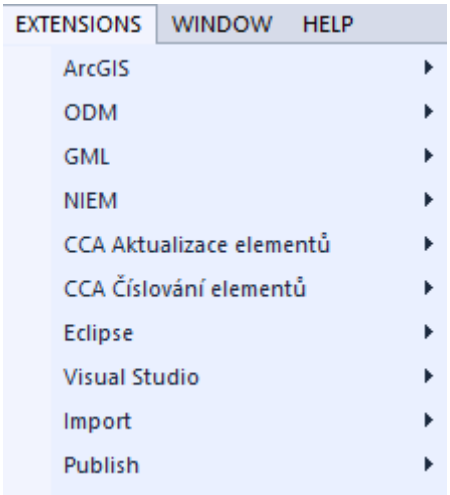
- Restartovat Enterprise Architect.

- Otevřít **Extensions -> MDG Technologies**.
  - Zkontrolovat, že v seznamu je „CCA Prostředí“ uvedeno pouze jednou.
  - Zkontrolovat, že je „CCA Prostředí“ zaškrtnuto.
- Otevřít projekt, pokud již není otevřen.
- Otevřít **Project -> Data management -> Import Reference Data**.
- Vybrat soubor **ref_tagged_values.xml**. V „Select Datasets to Import“ vybrat **Tagged Value Types** a kliknout na tlačítko **Import**.

Toolboxy definované v MDG technologii „CCA_Prostredi“ je možné otevřít kliknutím na položku **More tools** v okně **Toolbox**. Toolboxy se nachází pod položkou **CCA Prostredi**.

Po provedení výše uvedených kroků již stačí nainstalovat komponenty. Komponenty je možné nainstalovat pomocí přiloženého instalačního balíčku s názvem **`CCA_EA_ReusableComponents`** (tento balíček je přiložen na CD, viz příloha D). Před spuštěním instalačního balíčku je vhodné vypnout Enterprise Architect. Instalační balíček stačí pouze spustit a dále v něm vybrat složku, do které budou umístěny knihovny.

Po úspěšné instalaci komponent se v menu **`Extensions`** objeví dvě nové volby („CCA Aktualizace elementů“ a „CCA Číslování elementů“). Výsledné menu **`Extensions`** po úspěšné instalaci komponent je zobrazeno na obrázku B.2.



Obrázek B.2: Menu „Extensions“ po nainstalování komponent.

C Uživatelská příručka

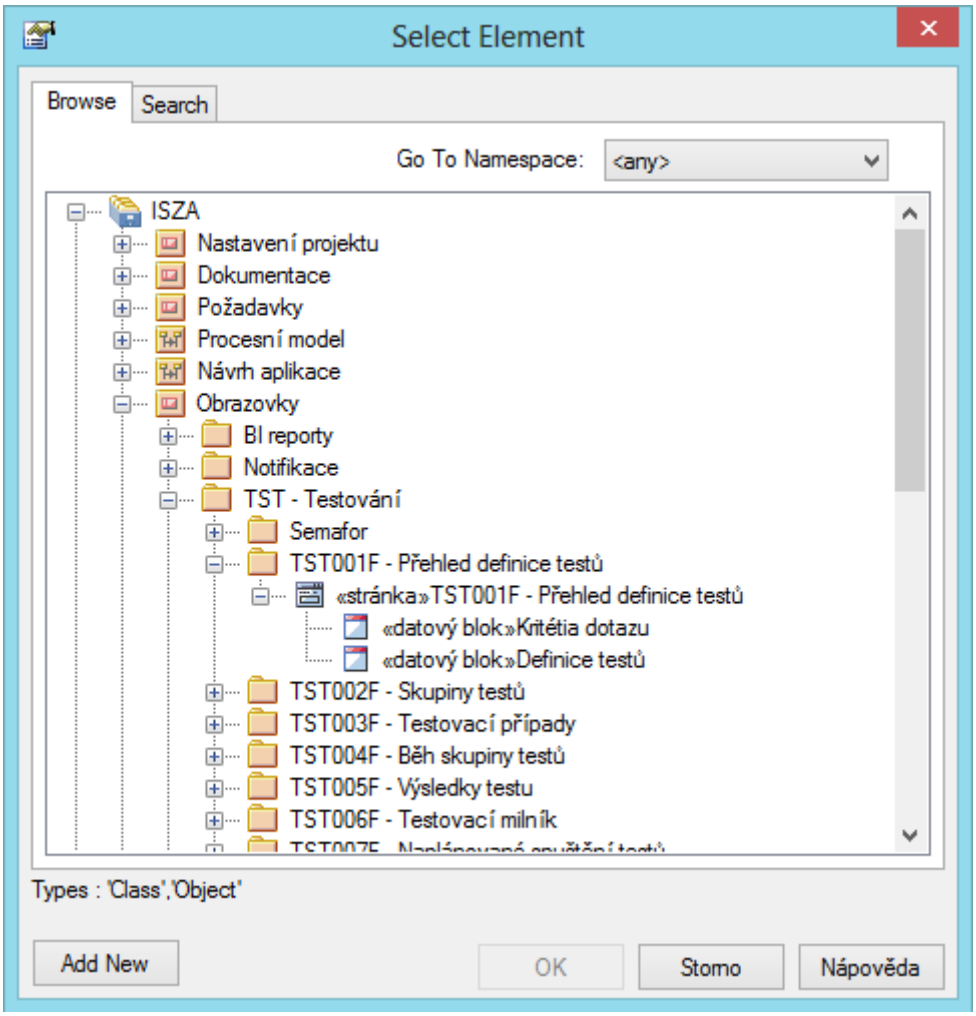
Uživatelská příručka pro používání komponent **DragAndDropCopying**, **ElementUpdate** a **Numbering**. Komponenty jsou určeny pro Enterprise Architect verze 11 a vyšší. Ve starších verzích Enterprise Architectu nemusí komponenty fungovat správně. Pro fungování komponent **DragAndDropCopying** a **ElementUpdate** musí být v Enterprise Architectu nainstalována MDG technologie „CCA_Prostredi“.

C.1 DragAndDropCopying

Tato komponenta slouží ke kopírování elementů typu **Class** a **Object** do vybraného diagramu.

Komponenta reaguje na „drag and drop“ událost. Pro spuštění komponenty stačí v Toolbox zvolit položku **More tools -> CCA Prostredi -> Screens** a přetáhnout na diagram element **datový blok**. Po přetažení datového bloku na příslušný diagram se objeví dotazovací okno, zda chce uživatel element zkopírovat. Pokud uživatel klikne na tlačítko **Ne**, dojde v diagramu k vytvoření nového prázdného elementu. Pokud uživatel klikne na tlačítko **Ano**, zobrazí kompomponenta nové okno se stromovou strukturou (viz obrázek C.1). V tomto okně ve stromové struktuře vybere uživatel element, který chce zkopírovat. Výběr zobrazuje pouze elementy typu **Class** a **Object**.

Po výběru elementu uživatel klikne na tlačítko **OK**. Po potvrzení výběru komponenta zkopíruje vybraný element do diagramu, na který uživatel přetáhl **datový blok**. Komponenta při kopírování také přidá do tagovaných hodnot elementu novou hodnotu s názvem **RefGUID** odkazující na původní element. Ke každému atributu komponenta přidá **attribute tag** s názvem **RefGUID**, který pomocí GUID odkazuje na „rodičovské“ atributy. Pokud v okně s výběrem elementu klikne uživatel na tlačítko **Storno**, vytvoří se v diagramu nový prázdný element.



Obrázek C.1: Okno pro výběr elementů a atributů ke kopírování.

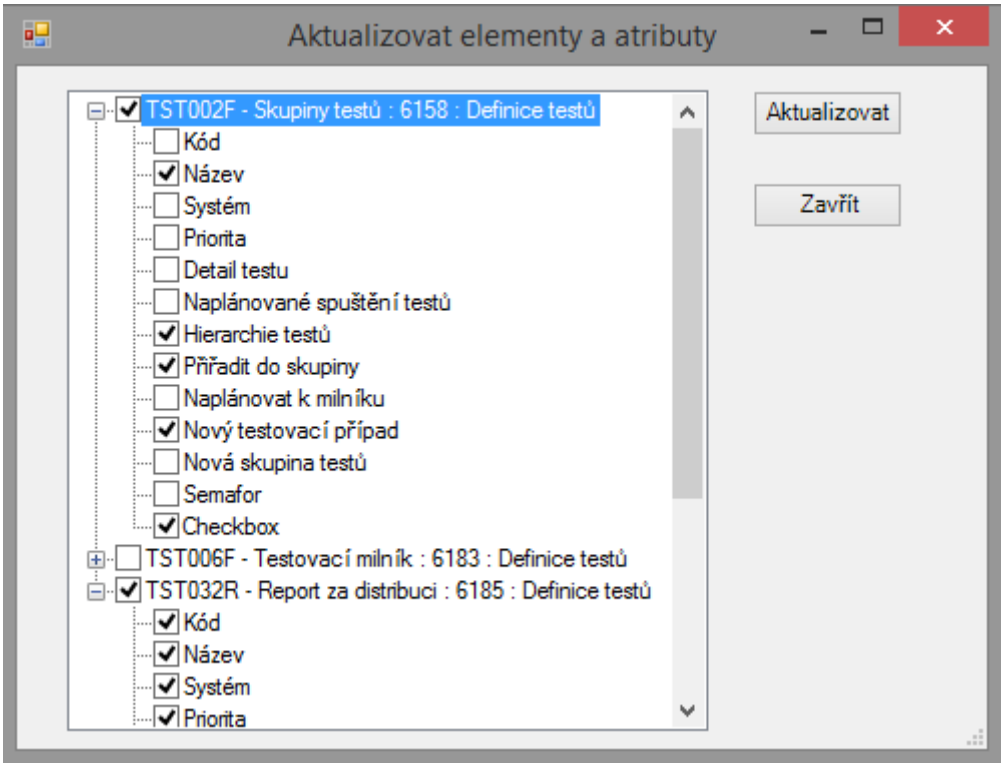
C.2 ElementUpdate

Komponenta slouží k propagaci změn provedených u původního elementu na vybrané zkopírované elementy.

Pro spuštění komponenty musí uživatel označit původní element. Po označení původního elementu existují dvě možnosti jakými lze komponentu spustit. První možností je pravým tlačítkem myši vyvolat menu a zvolit položku **`Extensions -> CCA Aktualizace elementů -> Aktualizovat`**. Druhou možností je zvolit **`Extensions`** na horní liště Enterprise Architectu a opět zvolit **`CCA Aktualizace elementů -> Aktualizovat`**. Elementy lze označit přímo v okně **`Diagram View`** i v okně **`Project Browser`**. Komponentu lze spustit pouze pro elementy, které mají stereotyp **`table`** nebo **`datový blok`**, případně pokud jsou stereotypy **`table`** nebo **`datový blok`**

součástí složeného stereotypu. Pokud z vybraného elementu byly zkopírovány elementy pomocí komponenty pro kopírování (viz kapitola C.1), otevře komponenta okno pro výběr elementů a atributů k aktualizaci (viz obr. C.2). Elementy a atributy jsou zobrazeny ve stromové struktuře. Uzly na nejvyšší úrovni reprezentují elementy zkopírované z označeného elementu. U těchto uzlů je zobrazen název diagramu či balíku, ve kterém se „potomek“ nachází, celočíselný identifikátor elementu a název elementu. Uzly, které jsou na nižší úrovni (tj. jsou viditelné až po rozbalení uzlu reprezentujícího element), reprezentují atributy daného elementu. Pokud neexistuje žádný „potomek“, zobrazí komponenta informační zprávu, že neexistuje žádný element, který by bylo možné aktualizovat.

K provedení aktualizace stačí, aby uživatel zvolil atributy, které chce aktualizovat. Poté musí uživatel kliknout na tlačítko **`Aktualizovat`**. Pokud uživatel označí element, pak se automaticky označí i všechny jeho atributy. Po kliknutí na tlačítko **`Aktualizovat`** ještě komponenta zobrazí potvrzovací okno se seznamem vybraných elementů a atributů. Po potvrzení provede komponenta aktualizaci informací ve všech vybraných elementech a atributech podle informací uložených v původním elementu.



Obrázek C.2: Výběr elementů ke kopírování.

Pokud chce uživatel aktualizovat pouze jediný atribut, stačí aby v okně

Project Browser označil u zkopírovaného elementu atribut, který chce aktualizovat, a opět zvolil položku **Extensions -> CCA Aktualizace elementů -> Aktualizovat**. Po spuštění add-inu nad atributem zobrazí komponenta zprávu, zda chce uživatel atribut opravdu aktualizovat. Po kladné odpovědi provede komponenta aktualizaci atributu. Po záporné odpovědi komponenta žádnou akci neprovede.

C.3 Numbering

Při vytváření nového elementu, který má být očíslován, vyplní uživatel do názvu a případně do aliasu pouze prefix a případný popis. Prefix a popis musí být odděleny oddělovacím znakem. Oddělovací znak může být pouze pomlčka nebo dvojtečka. V případě vyplnění popisu musí být mezi prefixem a oddělovacím znakem mezera, např. „REQ - Jednotné šablony pro všechny typy dokumentů“. Po vyplnění prefixu stačí, aby uživatel vyvolal menu pravým tlačítkem myši a zvolil položku **Extensions -> CCA Číslování elementů -> Číslovat**. Element lze označit v okně **Project Browser** i v okně **Diagram View**. Komponenta po svém spuštění nalezne všechny elementy v projektu, které mají stejný prefix. Mezi elementy se stejným prefixem nalezne prefix s největším číslem, toto číslo zvýší o jedna a uloží ho za prefix nového elementu. Pokud bude uživatel chtít touto komponentou očíslovat již očíslovaný element, neprovede komponenta žádnou akci. V případě, že uživatel bude chtít očíslovat jiný objekt než element, zobrazí komponenta chybovou hlášku a neprovede žádnou další akci. Číslování probíhá v rámci celého projektu.

D Obsah CD ROM

- Soubor „README.txt“ - Tento soubor obsahuje základní informace o instalaci a obsahu CD ROM.
  README.txt
- Kompletní text práce.
  Text práce\BP_A13B0355P_2016.pdf
- Zdrojové soubory textu práce.
  Text práce\Zdrojové soubory
- Vzorový projekt „ISZA.eap“ pro otestování komponent.
  Vzorový projekt\ISZA.eap
- Kompletní solution „CCA_EA_ReusableComponents“ nástroje Visual Studio se všemi projekty vytvořenými v rámci této práce.
  Reusable components
- Soubor „CCA_Prostredi.xml“ potřebný pro import MDG technologie CCA Prostředí.
  MDG technologie\CCA_Prostredi.xml
- Soubor „tagged_values.xml“ potřebný pro import definovaných typů hodnot.
  Definované typy\tagged_values.xml
- Instalační balíček „CCA_EA_ReusableComponents.msi“ potřebný pro instalaci výsledných komponent.
  Výsledné komponenty\CCA_EA_ReusableComponents.msi