VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

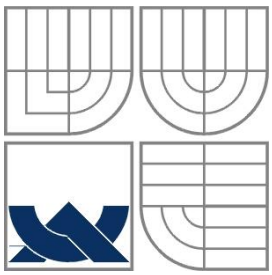
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

ROZŠÍŘENÍ NÁSTROJE PROCESS INSPECTOR O EXPORT DAT

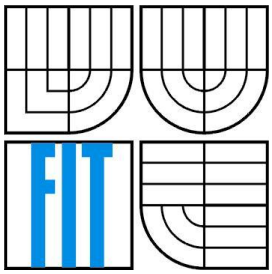
DIPLOMOVÁ PRÁCE

AUTOR PRÁCE Bc. Jan Brodzák

BRNO 2011



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

ROZŠÍŘENÍ NÁSTROJE PROCESS INSPECTOR O EXPORT DAT

PROCESS INSPECTOR TOOL EXTENSION FOR DATA EXPORT

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. Jan Brodzák
AUTHOR

VEDOUCÍ PRÁCE — doc. RNDr. Jitka Kreslíková, CSc.
SUPERVISOR

BRNO 2011

Abstrakt

Tato práce popisuje proces rozšíření nástroje Process Inspector o export dat. V teoretickém úvodu práce jsou vysvětleny pojmy týkající se managementu procesů, dále je popsán nástroj Process Inspector, platforma SharePoint, na níž je vyvíjeno vylepšení tohoto nástroje, a nástroj Enterprise Architect, pro který je rozšíření o export dat určeno. V další části jsou představeny jazyky UML a XML, které jsou využívány při exportu. Práce se zabývá procesy a dokumenty nacházejícími se v nástroji Process Inspector a uvádí postup, jak tyto procesy modelovat pomocí diagramů tříd. Na závěr práce popisuje, jakým způsobem je export dat implementován.

Abstract

This master‘s thesis describes the process of extending the Process Inspector for data export. The theoretical introduction deals with the theoretical background of process management. The following section describes the Process Inspector itself, SharePoint – a platform on which the extension will be developed – and Enterprise Architect – an instrument, which is the expansion of the data export intended for. It also describes the UML and XML languages, which are required for the export. The thesis also deals with the processes and documents in the Process Inspector and provides instructions on how to model these processes with the help of class diagrams. Finally, the method of data export implementation is discussed.

Klíčová slova

Process Inspector, Enterprise Architect, SharePoint, podnikové procesy, UML, XML, XMI

Keywords

Process Inspector, Enterprise Architect, SharePoint, business process, UML, XML, XMI

Citace

Jan Brodzák: Rozšíření nástroje Process Inspector o export dat, diplomová práce, Brno, FIT VUT v Brně, 2011

Rozšíření nástroje Process Inspector o export dat

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením doc. RNDr. Jitky Kreslíkové, CSc.
Další informace mi poskytli:

Ing. Zdeněk Fiedler, STI Software Technology Institut, a.s.
Bc. Martin Opršal.

Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Jan Brodzák
14.5. 2011

Poděkování

Děkuji vedoucí své diplomové práce doc. RNDr. Jitce Kreslíkové, CSc. za odbornou pomoc. Dále pak Ing. Zdeňkovi Fiedlerovi za odborné konzultace a Bc. Martinu Opršalovi za pomoc při řešení některých problémů.

Obsah

1 Úvod

Každá firma by se měla snažit o zefektivnění své činnosti. Pro dosažení tohoto cíle existuje mnoho různě složitých a také různě účinných způsobů. Jednou z metod je zavedení nového informačního systému. Ne každý informační systém však přináší stejné výhody a některé systémy mohou dokonce způsobit více škody než užitku.

Pro správný návrh nového informačního systému je nezbytné získat od zadavatele přesné zadání, co všechno by systém měl umět, k čemu by měl sloužit, a na základě těchto informací provést potřebné analýzy. Pro vývojáře je velmi užitečný každý nástroj, který tuto práci ulehčuje či jakkoliv zautomatizuje. Tato práce si klade za cíl rozšířit jeden z těchto nástrojů o další funkci. Jedná o nástroj Process Inspector sloužící k efektivnímu získávání informací o struktuře podnikových procesů. Cílem této práce je na základě dat z nástroje Process Inspector navrhnout vhodný model pro jejich reprezentaci a umožnit importování těchto dat v takové formě, která umožní následný export do nástroje Enterprise Architect. Enterprise Architect pak umožní další zpracování těchto dat. Nástroj Process Inspector je v současné době nově vyvíjen na platformě SharePoint a toto rozšíření bude součástí jeho nové verze.

Celá práce je rozdělena do několika kapitol. Následující kapitola se zabývá managementem procesů, poté následuje kapitola, ve které se seznámíme s použitými nástroji a technologiemi. Tato sekce detailně popisuje nástroj Process Inspector, vysvětluje jeho podstatu a možnosti využití k efektivnímu získání informací o struktuře podnikových procesů. V další sekci si představíme jazyk UML. Modelování za pomocí tohoto jazyka je základem nástroje Enterprise Architect, o němž pojednává další část. Process Inspector využívá platformu SharePoint, kterou se zabývá další část kapitoly, a na závěr se ve stručnosti zmíníme o jazyku XML, který bude využit při přenosu dat mezi nástroji Process Inspector a Enterprise Architect. Další kapitola s názvem Analýza se v úvodní podkapitole zabývá strukturou a obsahem dat uložených v databázi nástroje Process Inspector. Další část popisuje, jakým způsobem jsou daná data z databáze vybírána a upravována k dalšímu využití za účelem reprezentace pomocí UML modelu. Kapitola Implementace popisuje postup, kterým se export dat provádí. V úvodu této kapitoly jsou popsány technické prostředky, jež byly použity při implementaci, a dále následuje popis webové části a tříd. Následuje kapitola testování, v níž se ověřuje správnost vytvořeného nástroje.

Závěrečná kapitola hodnotí výsledky této práce, získané zkušenosti a možnosti dalšího rozšíření vytvořeného nástroje.

Kapitoly 3 a 4 jsou převzaty ze Semestrálního projektu a jen mírně upraveny, aby odpovídaly zvolené implementaci exportu dat.

2 Management procesů

Cíl práce spočívá v rozšíření nástroje Process Inspector o další funkce. Jak již bylo zmíněno, tento nástroj se zabývá efektivním získáváním informací o struktuře podnikových procesů. Tato kapitola se pokusí objasnit, co to vlastně podnikové procesy jsou, k čemu slouží a jak je lze využít ke zlepšení chodu firmy. Převážné část první poloviny této kapitoly je převzata z [1].

2.1 Motivace

Změny v ekonomickém prostředí nutí firmy neustále přehodnocovat svou konkurenceschopnost na trhu a hledat různé inovace a výhody, které jim pomohou uspět v konkurenčním boji. Společnosti obvykle rozdělují vnější pohled na svou firmu na tzv. „task environment“, v němž přijímají přímý vztah ke svým obchodním partnerům, a na „global market environment“, v němž hrají aktivity ostatních společností pouze minimální roli.

Task environment společnosti je tvořen nákupním chováním (buying behavior) spotřebitelů, strukturou trhu a dynamikou konkurence. Příkladem změny nákupního chování může být zvýšená poptávka po individualizaci produktu a služeb, což vede k větší fragmentaci trhu. S tím související nárůst změn v produkci vede k lepší koordinaci nákladů na zásobování, produkci, služby, distribuci a prodej. Snaha o zvládnutí zvyšující se náročnosti aplikací dalších koordinačních mechanizmů může vést ke vzniku tzv. „complexity trap“ (tendence přidávání dalších funkcí, za cenu větší složitosti a nižší spolehlivosti produktu), kdy režijní náklady na tuto dodatečnou kontrolu a koordinační systémy převyšují profit získaný množstvím změn. Kromě zvyšující se pozornosti na vnější pohled na společnost s ohledem na produktové řady, kvalitu služeb a spokojenost zákazníků se zvyšuje rovněž pozornost na vnitřní pohled (tj. účinné a inovativní provádění aktivit v rámci společnosti).
Je zdůrazňována nezbytnost soustředit se na podnikovou dynamiku: v době poměrně statické ekonomiky mohou být i strategie statické. Ve světě produktů s dlouhou životností, stabilních potřeb zákazníků, jasně vymezených národních a regionálních trhů a rozpoznatelné konkurence představovala konkurence „poziční boj“, kdy společnosti zaujímaly určité pozice na pomyslné šachovnici. Dnes představuje konkurence spíše „mobilní boj“, kdy úspěch závisí na očekávání trendů na trhu a rychlé reakci na měnící se potřeby zákazníků. Úspěšný konkurent rychle vyvíjí produkty, trhy a někdy dokonce celé odvětví s vidinou toho, že je opustí, co nejrychleji bude moci. Tento postup připomíná interaktivní videohru. V takovém prostředí spočívá jádro strategie nikoli ve struktuře produktů a trhů podniků, ale v dynamice jeho chování [7].

2.2 Vývoj v oblasti implementace procesního řízení

V uplynulých desetiletí se společnosti soustředily na účinné provádění jednotlivých funkcí, což vedlo k lokální optimalizaci a zdokonalení funkčních oblastí. Vývoj v oblasti technologie a organizace vedl k významnému zvýšení produktivity a kvality v oblastech logistiky, účetnictví a výroby, díky užívání nových informačních a komunikačních technologií jako např. standardizovaný software, telefonická poradenská centra (call centra), internet a intranet a rovněž díky využívání nejrůznějších organizačních konceptů, jako např. nákup služeb (outsourcing) funkčních oblastí. Užívání moderních informačních a komunikačních technologií však neumožňuje eliminovat tento strukturální problém. Interní elektronická komunikace firmy pouze odstraňuje symptomy, tj. délku koordinačních procesů.

Myšlenka uspořádat společnost tak, že bude orientována na procesy, není rozhodně nová. Tomuto konceptu je již od konce osmdesátých let minulého století věnována čím dál tím větší pozornost. Snaží se přilákat pozornost chytlavými frázemi jako „business process reengineering“ nebo „business proces management“. Ačkoli první úvahy o procesně orientovaném uspořádání společností se v akademické literatuře objevuje již od třicátých let, společnosti tento koncept začínají používat v praxi až koncem osmdesátých let.

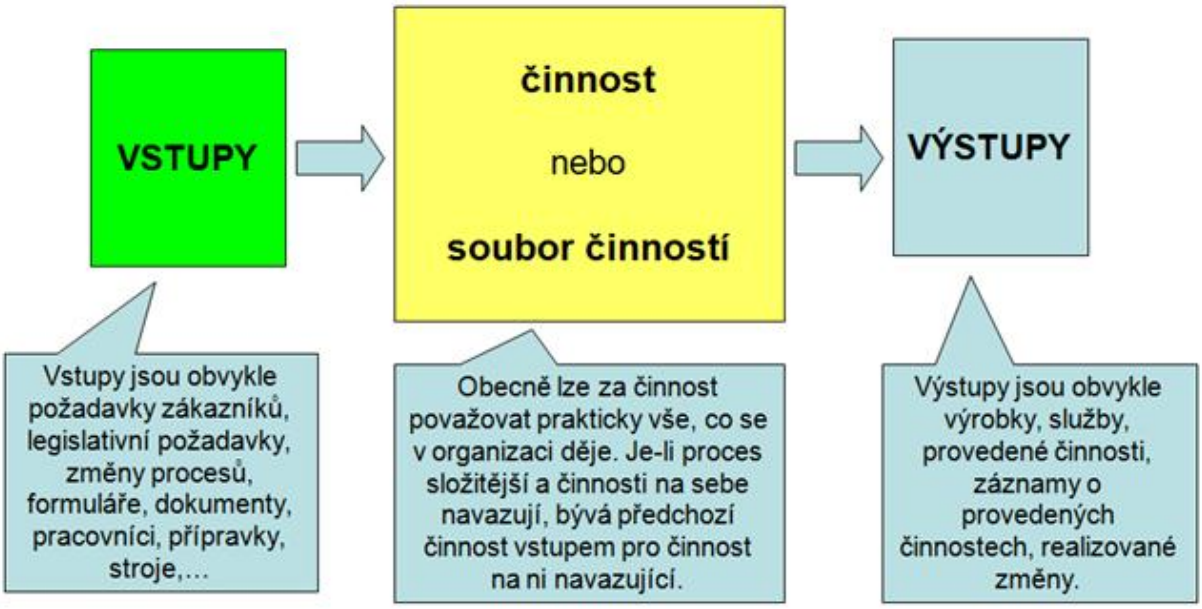
2.3 Vysvětlení důležitých pojmů

Hlavní předmět orientace

Klíčovým bodem uspořádání společnosti orientované na procesy jsou podnikové procesy (business process). Zatímco podle organizační struktury je společnost rozdělena na parciální systémy (např. oddělení, divize, jednotky), kdy každá z nich plní svůj úkol, obchodní procesy se zabývají prováděním těchto úkolů a koordinací dalších aspektů (kdo co dělá, jak a čím). Hlavní prvky úkolů představují činnosti, které tvoří základní části pracovního procesu. Činnost a/nebo funkce představují krok, který je nutný vykonat, aby bylo možné poskytnout službu.

Proces

Proces je naprosto uzavřená a logická sekvence činností, které jsou nutné pro práci s procesně orientovaným podnikovým objektem. Příkladem podnikově orientovaného objektu může být například faktura nebo objednávka. Podnikový proces je speciální druh procesu, který je řízený podnikovými cíli společnosti a podnikovým prostředím. Nezbytnými znaky podnikového procesu jsou vztahy k obchodním partnerům společnosti, např. k zákazníkům či dodavatelům. Příkladem podnikového procesu může být kupříkladu zpracování zakázky ve výrobě, vyřízení objednávky, nebo platební příkaz v bance. Pro lepší názornost je uveden obrázek 1, jenž může sloužit jako grafická definice procesu.



Obrázek 1: Definice procesu, převzato z [10]

Porterův hodnotový řetězec

Porter uveřejnil svůj model hodnotového řetězce v roce 1980, kdy rozděluje činnosti společnosti na primární a podpůrné. Primární činnosti jsou činnosti, které vytváří hodnoty, které mají přímý vztah k vyráběnému produktu, a tak přispívají k ekonomickému výsledku společnosti (tj. činnosti v oblasti zásobování, výroby, marketingu a prodeje, logistiky a zákaznických služeb). Podpůrné služby nemají přímý vztah k vyráběným produktům a službám, přesto by však bez těchto podpůrných služeb nebylo možné provádět žádné primární činnosti. Sem patří například řízení lidských zdrojů, účetnictví a zpracovávání dat.

Klíčové a podpůrné procesy

Klíčový proces je proces, jehož činnost přímo souvisí s produktem společnosti a tak přispívá k tvorbě hodnot společnosti. Podpůrné procesy jsou protiváhou klíčovým procesům. Jsou to procesy, jejichž činnosti nevytváří hodnoty z pohledu zákazníka, ale jsou nezbytné k vykonávání klíčových procesů. Hranice mezi klíčovými a podpůrnými procesy se často stírají, protože jeden a ten samý proces může být zároveň jak klíčovým, tak podpůrným procesem, v závislosti na kontextu nebo odvětví podnikání. Podpůrný proces nelze chápat jako méně významný proces, podpůrné procesy jsou naopak nezbytnou podmínkou pro vykonávání klíčových procesů, pouze nemají přímou spojitost s vyráběnými produkty anebo službami. Provádění klíčových procesů by bez pomocných procesů nebylo možné.

Management procesů

Management procesů sleduje, zda jsou procesy v dané organizaci prováděny tak, jak bylo naplánováno, zda jsou dodržovány, a případně činí nutná vylepšení pro dosažení maximální kvality výsledku. Pro management procesů jsou důležitá zejména aktuální data poskytovaná procesy a případně i historická data, která pomáhají vyvarovat se chybám při změnách procesů v souvislosti s jejich změnami [18].

Reengineering

Reengineering znamená zásadní přehodnocení a radikální rekonstrukci podnikových procesů tak, aby bylo dosaženo dramatických zdokonalení z hlediska kritických měřítek výkonnosti, jako jsou náklady, kvalita, služby a rychlost [6].

Výsledkem úspěšného reengineeringu je procesně řízená organizace, která se vyznačuje následujícími charakteristikami:

- Procesy mají přednost před organizační strukturou.
- Vykonávání činností v procesech je založeno na principu rolí, nikoli pracovních míst.
- Role jsou přiděleny těm pracovníkům, kteří pro jejich vykonávání mají odpovídající kompetence (znalosti, způsobilost a pravomoci).
- Role jsou pracovníkům přidělovány s přihlédnutím k jejich aktuálnímu pracovnímu vytížení.

2.4 Reengineering podnikových procesů

Jedná se o oblast managementu, která vyvolala velmi bouřlivé pozitivní ale i negativní reakce. Reengineering ovlivnil velké množství firem. Na jedné straně stojí firmy, které jej úspěšně zavedly a dokázaly tak radikálně zvýšit svou výkonnost a na straně druhé firmy, které jej nezavedly, a nebo bylo jejich zavedení neúspěšné, a tyto firmy tak byly vytlačeny konkurencí ze svých pozic. Reengineering tak paradoxně ovlivnil jak firmy, které jej přijaly a zavedly, tak firmy, které jej minuly bez povšimnutí.

Z toho logicky vyplývá, že jedna skupina organizací tento manažerský koncept chválí a druhá je jím zklamaná a odmítá jej. Reengineering podnikových procesů je však velmi dobrý nástroj, který – je-li správně použit – může zásadním způsobem pozitivně ovlivnit fungování jakékoliv organizace.

Řada manažerů stále není schopna se zaměřit na procesy. Důvod spočívá v tom, že podnikové organizační struktury byly více než 200 let založeny na specializaci práce a na jednoduchých opakovatelných činnostech. Procesy byly fragmentovány, a tak o nich manažeři nepřemýšleli, neměřili je a nesnažili se je vylepšit. Nikdo nebyl schopen překonat odpor ostatních funkčních manažerů, sledujících své zájmy, a jednotlivé kroky procesu spojit tím, že je povýší nad tradiční organizační strukturu. V dnešní době však procesní orientace není volbou a dokonce často ani konkurenční

výhodou. Jedná se o standard podnikatelského modelu a nutnost, bez které se firma, která chce uspět, neobejde [8].

Při reengineeringu procesů se využívá mnoho různých metodik, lišících se jak rozsahem, tak zaměřením a také poměrem teoretické orientace. Podrobný popis a porovnání jednotlivých metodik lze nalézt v řadě odborných publikací. V tabulce 1 je uveden reprezentativní přehled významných metodik – přístupů k reengineeringu procesů.

Metodika	Původ – specifické zaměření
Hammer, Champy	Konsultantský/akademický
Davenport	Akademický
Manganelli, Klein	Konsultantský
Kodak	Uživatelský
DoD	Státní správa
Aris Method (prof. Scheer)	Konsultantský/akademický Akcentuje sociálně-psychologické aspekty projektu
PPP Method (prof. Gappmaier)	Konsultantský/akademický Akcentuje sociálně-psychologické aspekty projektu
DEMO Method (prof. Dietz)	Konsultantský/akademický Akcentuje sociálně-psychologické aspekty projektu

Tabulka 1: Přehled metodik reengineeringu procesů, převzato z [6]

Jak se však uvádí v [6], platí, že: "*Jako všechny metodiky, i metodiky reengineeringu procesů mohou sloužit pouze jako základní přehled problematiky a vzor postupu. Použití metodiky potom nutně vyžaduje příslušnou znalost a dovednost, nikoliv pouze metodické informace. Žádná sebelepší metodika neudělá z neznalců znalce, nevyřeší sama problémy, které k reengineeringu vedou. Pro konkrétní projekt reengineeringu je vždy zapotřebí mít vlastní – v podstatě vlastní silou vytvořenou – metodiku, která zohledňuje jednak danou situaci, specifičnost potřeby a prostředí, včetně příslušné úrovně znalosti účastníků plánovaného reengineeringu. Není zkrátka možné efektivně použít něco, čemu ti, kdo to budou používat, nerozumí, co jim není vlastní*."

Základem při přípravě reengineeringového projektu je výběr metodiky, která může být pozměněna a která bude dále hrát roli metodického rámce, v němž posléze vznikne představa základního postupu – plán reengineeringového projektu.

Předpokladem úspěšného provedení reengineeringu je tvůrčí aplikace a zproduktivnění informační technologie. Cílem firmy je zprovoznit informační technologie a systémy, jež dokážou naplnit potřeby procesů a systémů a které budou efektivně přispívat k naplnění cílů firmy. Naplnit tento cíl však nebývá nijak snadné. Reengineering je manažerský problém, proto by měli mít manažeři

jasno v tom, co od informačních systémů a technologií (IS&IT) očekávat a co mohou požadovat. Je tedy velmi důležité, aby manažeři alespoň rámcově porozuměli tomu, jaké možnosti jim IS&IT nabízí. Správně použité informační technologie podporují vysokou výkonnost podnikových procesů, vedou ke zrychlení, zefektivnění a zabraňují vzniku chyb. Následující tabulka uvádí příklady některých technologií a způsob, jakým jejich používání vylepšilo fungování firem.

Zlomová technologie	Staré pravidlo	Nové pravidlo	Příklad
Sdílené databáze	Informace mohou být v jednu dobu přítomny pouze na jednom místě.	Informace mohou být v jednom okamžiku přítomny všude, kde jsou potřeba.	Vypočítávání pojistné sazby potenciálního klienta, zatímco jsou současně prověřovány jeho kreditní položky.
Expertní systémy	Komplexní práci mohou zastávat jen experti.	Univerzalista může v rámci jednoho procesu vykonávat práci více expertů.	U společnosti IBM Credit, podnikající v oblasti financování počítačů, software a služeb, zprostředkovává jeden člověk žádosti o financování místo pěti specialistů. Reengineering procesu vyřízení žádosti přinesl zkrácení doby cyklu o 90 % a stonásobné zvýšení produktivity.
Bezdrátový přenos dat a přenosné počítače	Provozní personál nemůže rychle a kvalitně rozhodovat, potřebuje pracoviště, kde může nakládat s informacemi.	Provozní personál může rychle a přesně rozhodovat a nakládat s informacemi kdekoli v terénu.	Půjčovna aut AVIS díky této technologii umožňuje zákazníkům vrátit vypůjčený vůz na kterékoli pobočce.
Internet a mezipodnikové databáze	Podniky nemohou účinně spolupracovat.	Podniky v rámci hodnotového řetězce mohou intenzivně spolupracovat.	Internet a mezipodnikové databáze, kde mohou být sdíleny informace pro všechny členy hodnotového řetězce, umožňují prodejcům získat přesné informace, vše lépe plánovat, minimalizovat zásoby a snižovat náklady.

Tabulka 2: Přehled metodik reengineeringu procesů, převzato z [8]

Tak jako existuje řada různých metodik procesního reengineeringu, existuje i mnoho metod a technik procesního modelování. Podrobný popis těchto metod a technik lze nalézt v odborné literatuře. V následující kapitole je popsán jazyk UML, jenž byl v této práci vybrán k modelování podnikových procesů v nástroji Process Inspector.

3 Použité nástroje a technologie

V současné době existuje široká škála nástrojů a technologií, které lze užít při analýze a návrhu nového programu, a stále nové možnosti přibývají. Tyto nástroje mají za cíl maximálně zjednodušit, zefektivnit a zautomatizovat celý proces tvorby nového systému. Vzhledem k obrovskému množství nástrojů a technologií na současném trhu není v silách běžného vývojáře všechny postihnout.

V následující kapitole jsou představeny základy nástrojů a technologií, které byly využity při realizaci tohoto projektu. Kompletní popis by byl velmi rozsáhlý, zaměříme se tedy jen na nejdůležitější části.

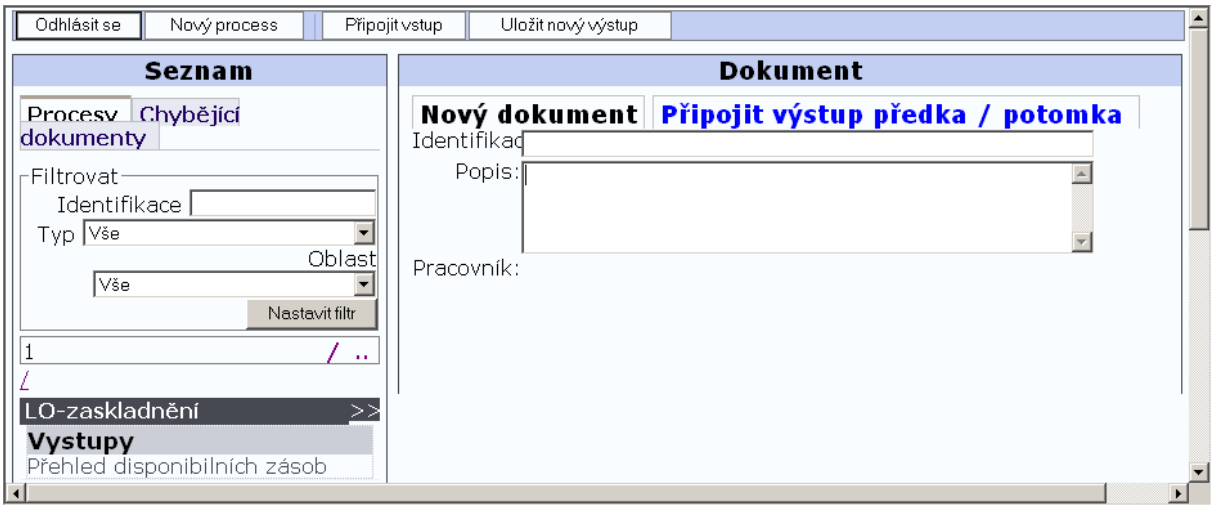
3.1 Process Inspector

Jedná se o nástroj vytvořený v roce 2001 firmou Allium s.r.o. k efektivnímu získání informací o struktuře podnikových procesů. Tato firma se mimo jiné zabývá implementací a podporou podnikových systémů. Při vytváření nového podnikového systému je naprosto nezbytné získat informace o struktuře podnikových procesů. Zákazníci však nebývají odborníci na tvorbu podnikových systémů a obvykle mají problém správně popsat všechny podnikové procesy. Většina společností navíc nemá přesně zmapovány všechny podnikové procesy, které v jejich firmě probíhají, a tak se často pro dva stejné procesy užívají různá pojmenování. Tento problém napomáhá řešit právě nástroj Process Inspector.

3.1.1 Neformální popis užití

Každý zaměstnanec firmy, pro kterou bude vytvářen nový podnikový systém, vytvoří seznam všech procesů a podprocesů, které ve firmě vykonává. Dále má za úkol popsat všechny vstupy do svých procesů, a všechny výstupy z nich. Tyto vstupy, výstupy a procesy jsou společně s podrobným popisem zaneseny do nástroje Process Inspector, kde je jim přiřazeno jedno z předdefinovaných pojmenování. Díky tomuto postupu se docílí sjednocení pojmů. Po zanesení všech procesů Process Inspector odhalí případné nesrovnalosti, např. když zaměstnanec ve svých procesech využívá vstupy, které nikdo neprodukuje, nebo naopak, když jiný zaměstnanec vytváří výstupy, které se k ničemu nevyužívají. Tyto nesrovnalosti mohou vzniknout v případě, že zaměstnanci zapomenou popsat některý z procesů, které vykonávají; ty jsou následně dodatečně zaznamenány do nástroje Process Inspector.

Poté, co jsou všechny procesy správně zaneseny, a je jim přiřazeno odpovídající označení, má uživatel Process Inspector k dispozici všechny potřebné a přesné informace. Tyto informace je dále možné využít při analýze a návrhu informačního systému a dosáhnout tak lepších výsledků. Na obrázku 2 je ukázka nástroje Process Inspector.



Obrázek 2: Ukázka původního nástroje Process Inspector

3.1.2 Použité technologie

Tento nástroj je implementován jako webová aplikace, což umožňuje snazší správu nasazené verze aplikace. Oproti klasickému řešení klient/server není třeba zákazníkovi poskytovat žádný instalační balíček ani řešit distribuci nových verzí. Uživateli aplikace stačí pouze prohlížeč internetových stránek a přístup k internetu [14]. Další výhodou je zkvalitnění servisu, protože technik nemusí jezdit ke klientovi, ale aplikaci opraví na jednom centrálním místě.

Tento nástroj byl původně vytvořen v programovacím jazyku PHP, a jak již bylo uvedeno v úvodní kapitole, v současné době se vytváří nová verze na platformě SharePoint 2010 s využitím technologií ASP.NET, která bude rozšířena o nové nástroje a funkce. Tato práce má za cíl vytvořit jedno z těchto rozšíření.

3.2 UML

Jazyk označovaný zkratkou UML (*Unified Modeling Language)* je, jak již název napovídá, unifikovaný modelovací jazyk, který má, na rozdíl od převážně textově orientovaných programovacích jazyků, vlastní grafickou syntaxi (tj. pravidla pro sestavování jednotlivých elementů jazyka do větších objektů) a sémantiku (tj. jednoznačná pravidla určující jednotlivým syntaktickým výrazům jejich význam).

V současné době má jazyk UML největší význam při návrhu softwarových systémů, protože objektově orientovaný návrh každé složitější aplikace je nezbytným předpokladem pro její úspěšnou a rychlou implementaci. Pro objektově orientovaný návrh je samozřejmě možné použít různé podpůrné prostředky, zejména další odlišné typy diagramů, UML je však významný také v tom ohledu, že přesně specifikuje, co má daný diagram obsahovat, což je velmi důležité zejména při sdílení informací mezi jednotlivými analytiky a vývojáři. Dále je již z principu UML nutné, aby vytvářené grafy měly vnitřní konzistenci a přesně danou sémantiku, což u jiných typů grafů nemusí být obecně zaručeno. Existuje několik typů UML diagramů, které se liší podle toho, jaké se pomocí nich plánují či zpracovávají úlohy. Tyto diagramy se od sebe odlišují především repertoárem použitých značek, způsobem vzájemného propojení a související sémantikou.

Mezi velké přednosti jazyka UML (a na něm postavených UML diagramů) patří existence otevřeného a rozšiřitelného standardu, podpora celého vývojového cyklu aplikace či jiného (ne nutně programového) systému a velká podpora pro různé aplikační oblasti. Pro širší využití jazyka UML v praxi mluví také významný fakt, že je podporován celou řadou vývojových nástrojů. Některé z těchto nástrojů jsou dokonce schopny z vývojářem zakresleného UML diagramu přímo vygenerovat spustitelný kód [17]. K vytvoření tohoto kódu je však zapotřebí velmi přesné vyjadřování při tvorbě UML diagramů.

3.2.1 Základní elementy jazyka UML

Jazyk UML se skládá ze tří typů elementů. Každý z těchto elementů je reprezentován pomocí grafických značek, které jsou rozmístěny v dvourozměrném grafu. Tyto tři základní elementy se podle své funkce nazývají:

- Předměty
- Relace
- Diagramy

V následujících podkapitolách budou jednotlivé grafické elementy jazyka UML blíže popsány a demonstrovány na schématech.

3.2.2 Předměty

Předměty (*Things*) jsou elementy zpracovávaného modelu, jež jsou následně členěny do několika rozdílných podkategorií. Prvním typem podkategorií jsou takzvané strukturní abstrakce (*Structural Things*) UML modelu, například programové třídy, aplikační či objektová rozhraní, případy užití, komponenty či uzly. V diagramu UML jsou strukturní abstrakce zobrazeny jako různé tvary, které jsou však vždy uzavřené. Jedná se kupříkladu o obdélníky, elipsy, kružnice či jednoduché, zdánlivě trojrozměrné tvary, například krychle či kvádry. Subjekt modelu nese statické informace o softwarovém artefaktu a je většinou vyjádřen podstatným jménem.

V podkategorii pro dynamické předměty (*Behavioral Things*) se nalézají předměty, které zachycují dynamické chování systému. Pomocí chování lze také modelovat stavový stroj, u něhož se stavy specifikují pomocí přechodů, událostí a aktivit. Pro zachycení chování se v UML diagramu používají různě zakončené šipky či propojovací čáry.

Ke sdružování sémanticky příbuzných předmětů se používají takzvané balíčky, pro něž je vytvořena samostatná kategorie předmětů vyjadřujících seskupení (*Grouping Things*). Balíčky mají tvar stylizovaný do kancelářské složky s popisem umístěným v levé horní části obdélníku. Zobrazení seskupení se může v různých prostředích mírně odlišovat.

Posledním a současně velmi jednoduchým typem předmětů jsou poznámky, které patří do kategorie anotačních předmětů (*Annotational Things*). Poznámky blíže specifikují vlastnosti a chování dalších elementů UML diagramu. Ke každé části modelu je možné zadat libovolné množství poznámek. Graficky se poznámky zaznamenávají pomocí vyplněného obdélníku s ohnutým rohem, často žluté barvy.

3.2.3 Relace

Předměty v grafech obvykle neleží chaoticky vedle sebe, ale jsou navzájem různě propojeny.
K tomuto propojení jsou v jazyce UML specifikovány relace. V UML jsou rozeznávány následující podtypy relací (z čistě jazykového hlediska patří relace mezi základní elementy UML):

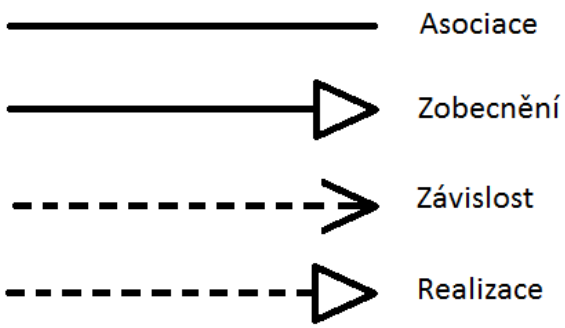
Asociace (*Association*). Pomocí asociací se modeluje obecná souvislost předmětů, která je však v diagramu UML přesným způsobem definovaná. Speciální variantou asociace jsou tzv. kompozice a agregace, které se často používají v objektově orientovaných jazycích a návrzích databází.

Závislost (*Dependency***)**. Změna v jednom předmětu způsobí změnu v jiném předmětu nebo mu poskytne požadovanou informaci.

Generalizace (*Generalisation*). Jeden předmět je specializací jiného předmětu. Například třída „osobní auto“ je specializací obecné třídy „vozidlo“. Tato relace je velmi často používána v objektově orientovaných jazycích, implementuje se většinou pomocí dědičnosti (*Inheritance*).

Realizace (*Realization*). Realizace je druh vztahu, při němž jeden předmět představuje dohodu, za jejíž plnění je odpovědný jiný předmět. V objektově orientovaných jazycích se realizace vytváří pomocí rozhraní (*Interface*).

Grafické znázornění relací je zaznamenáno na obrázku 3.



Obrázek 3: Příklady jednotlivých relací

3.2.4 Diagramy

Diagramy jsou nejznámější a nejpoužívanější částí standardu. Zachycují různé aspekty modelovaného systému, který může být vyjádřen i pomocí několika UML diagramů. To je možné například pomocí balíčků (*Packages*) sdružujících více diagramů do jednoho hierarchicky organizovaného celku. Existuje velké množství nejrůznějších typů diagramů, např. diagram případů použití, diagram tříd, diagram objektů, diagram komponent, diagram nasazení, diagram aktivit, stavový diagram, diagram spolupráce a sekvenční diagram [15]. Každý z výše vyjmenovaných typů diagramů obsahuje poněkud odlišný repertoár dostupných grafických značek a záleží na vývojáři, který z diagramů si vybere.

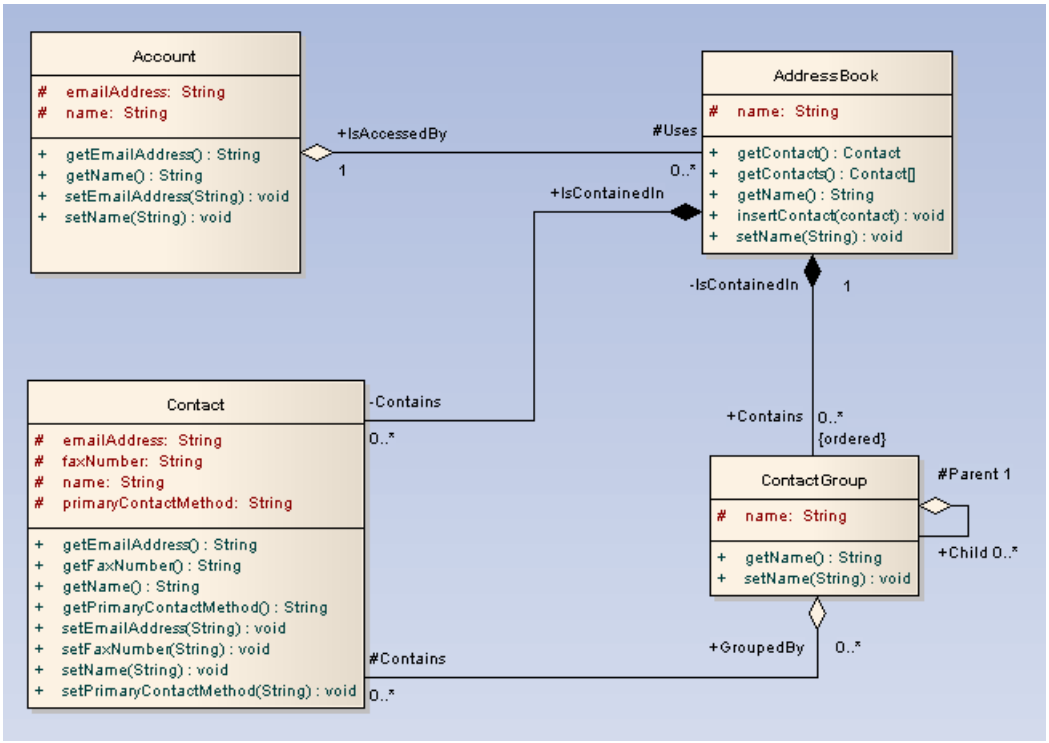
Při realizaci diplomové práce byl použit diagram tříd a z tohoto důvodu je jako jediný diagram blíže popsán.

Diagram tříd zobrazuje statický pohled na systém, tj. zejména třídy (*Class*) jako typy objektů, obsah tříd a statické vztahy, které mezi nimi existují. Může obsahovat také *Packages*. Dále může obsahovat elementy chování (*Behavioral*), např. operace, ale jejich dynamika je vyjádřena jinými

diagramy, jako jsou např. stavový diagram nebo diagram komunikací. Tento diagram se využívá v následujících fázích:

- fáze analýzy (konceptuální model)
- fáze návrhu (návrh atributů a operací)
- fáze implementace (návrh a tvorba programového kódu) projektovaného systému

Na obrázku 4 je ukázka diagramu tříd zobrazující adresář.



Obrázek 4 :Příklad diagramu tříd – adresář, převzato z [20]

3.3 Enterprise Architect

Na současném trhu lze nalézt celou řadu kvalitních produktů pro práci s jazykem UML, speciálně s UML diagramy. Většinou se jedná o velmi rozsáhlé a profesionální produkty, pro jejichž úspěšné a efektivní používání je zapotřebí absolvovat školení či poměrně dlouhé studium. Výsledkem nasazení těchto systémů však bývá rapidní urychlení práce, zejména prvotního návrhu systému. Jedním z těchto nástrojů je Enterprise Architect. Jak již bylo uvedeno dříve, cílem této práce je vytvořit nástroj, který umožní import UML diagramů z nástroje Process Inspector do Enterprise Architect.

3.3.1 Obecné informace

Jedná se o nástroj používaný pro modelování za pomocí UML, který je vyvíjen australskou společností Sparx Systems. Program podporuje platformy Windows i Linux. Enterprise Architect umožňuje podpořit a usnadnit celou fázi vývoje software, od definice požadavků na systém, přes design až po přípravu testování.
Program se dodává v několika verzích, jejich odlišnosti spočívají zejména v míře podpory týmové spolupráce.

Enterprise Architect se vyznačuje podporou modelů, mezi něž patří Business Process Model, Use Case model, Class model, Activity model, Sequence model a Component model. Nástroj vytváří výstupy jak ve formátu RTF pro dokumentaci, tak ve formátu XMI, který je určen pro budoucí spolupráci s ostatními programy.

3.3.2 Popis funkcí

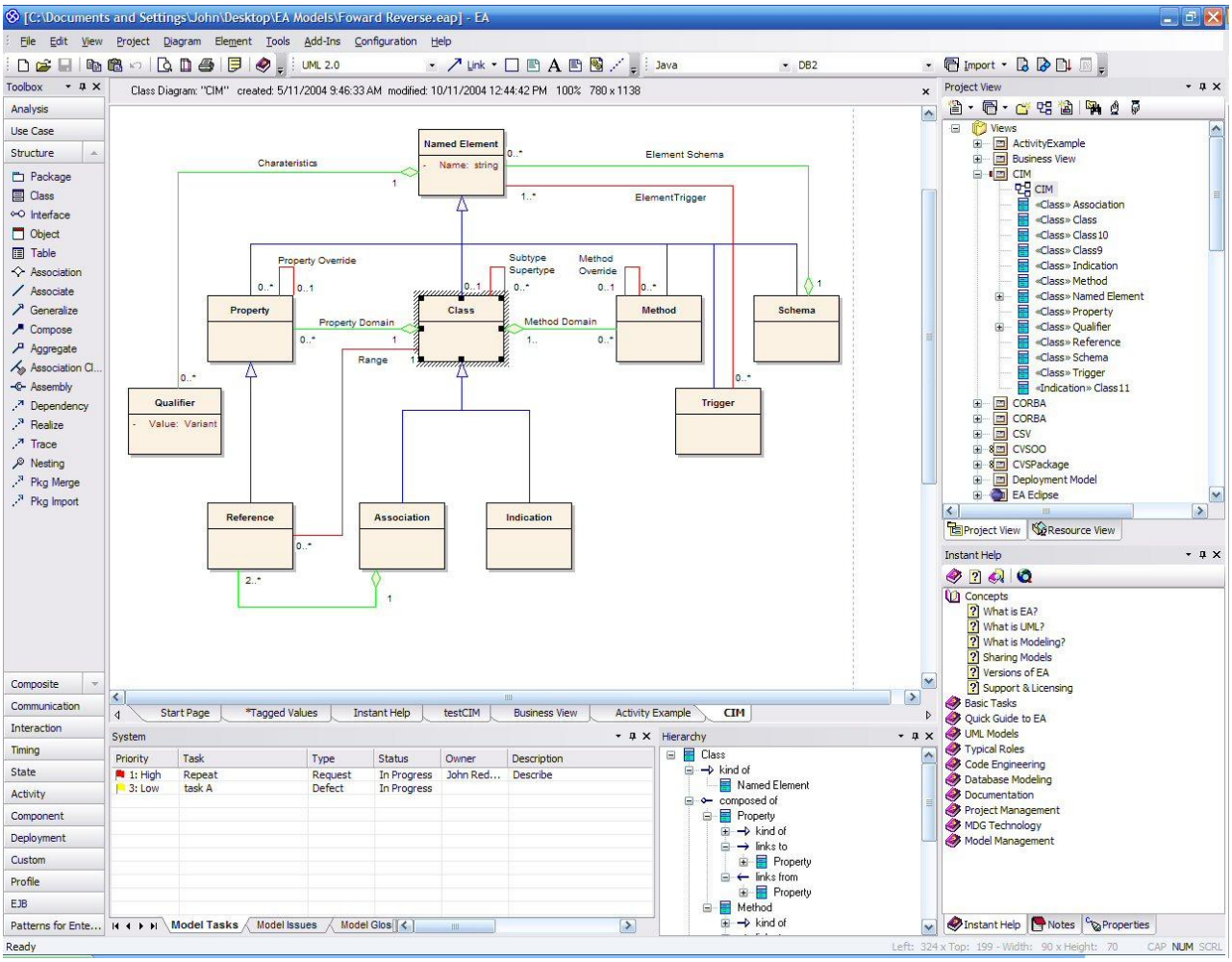
Jak již bylo řečeno, Enterprise Architect má za úkol usnadnit fázi vývoje software. Nyní se zaměříme na podrobnější popis jednotlivých funkcí a nástrojů. Hlavním úkolem je modelování, přičemž nástroj podporuje diagramy UML, tedy diagramy chování a strukturní diagramy. Dalšími funkcemi jsou modelování obchodních procesů a datové modelování, jenž umožňuje tvorbu logického a fyzického datového modelu. Do modelování je zařazeno i mapování struktury modelů.

Enterprise Architect nabízí možnost automatického generování kódu (*Code engineering)*, do níž spadá dopředné inženýrství (*Forward code engineering)* a zpětné inženýrství (*Reverse code engineering)*. *Forward code engineering* umožňuje generovat zdrojový kód z již vytvořených modelů. Bohužel při vývoji se *forward code engineering* příliš nevyužívá, protože jeho použití je podmíněno synchronizací modelů, což je příliš náročné na zdroje. Enterprise Architect umožňuje generovat zdrojový kód v celé řadě objektově orientovaných jazyků, konkrétně Java, C++, C #, Visual Basic, Delphi a PHP. *Reverse code engineering* umožňuje získat třídní diagram nebo fyzický datový model z již naimplementovaných tříd anebo z vytvořené databáze. Používají se stejné programovací jazyky jako u *forward code engineeringu.*

Nástroj Enterprise Architect podporuje hned několik různých typů testování, jako Unit testy, integrační testy, systémové testy či akceptační testy [21].

Pro tuto práci je také velmi důležité, že Enterprise Architect podporuje celou řadu různých mechanizmů pro export a import modelů reprezentovaných standardem XMI. Tato vlastnost umožňuje uživateli zpracovávat informace vytvořené v jiných nástrojích a přenášet informace o modelech mezi Enterprise Architect a dalšími nástroji.

Pro lepší představu je nástroj Enterprise Architect zobrazen na obrázku 5.



Obrázek 5: Ukázka nástroje Enterprise Architect, obrázek převzat z [20]

3.4 SharePoint

Nová, právě vyvíjená verze nástroje Process Inspector využívá nejnovějších technologií, jednou z nichž je i platforma SharePoint. Rovněž vyvíjený nástroj na import dat je vytvářen za pomocí této platformy, a proto zde bude blíže představena.
Microsoft SharePoint 2010 je profesionální komunikační centrum, které nabízí možnost on-line sdílení veškerých informací mezi spolupracovníky nebo projektovými týmy. Technologie SharePoint si klade za cíl především:

- usnadnit spolupráci mezi zaměstnanci a pracovními týmy
- zajistit sdílení znalostí
- poskytnout nástroje pro správu dokumentů a webového obsahu
- umožnit uživatelům přístup k informacím, které potřebují pro svou práci
- umožnit další vývoj aplikací využívajících funkčnosti zabudované v technologii SharePoint (například obousměrná komunikace s podnikovými aplikacemi, nastavení automatizovaných procesů, vyhledávání informací nebo nástroje pro reportování).

SharePoint umožňuje propojení jednotlivých nástrojů vytvořených společností Microsoft v jeden celek obohacený o další užitečné funkce. Hlavním požadavkem při tvorbě SharePoint bylo, aby uživatel mohl pracovat s nástroji od Microsoftu, které zná, a nemusel si zvykat na nové aplikace a příkazy. Proto je technologie SharePoint 2010 velmi provázaná s aplikacemi balíku Office 2010 a tak uživatel často může využívat serverové možnosti, aniž by musel opustit již známé aplikace. V rámci jednotlivých aplikací balíku Office v záložce Soubor je k dispozici technologie *Backstage*, která umožní pracovat s dokumenty nacházejícími se v lokalitě SharePoint, jako by byly uloženy na harddisku [13].

3.4.1 Oblasti SharePoint

Možnosti technologie SharePoint lze rozdělit do šesti oblastí tak, jak ukazuje následující obrázek (obrázek 6):

Obrázek 6: Oblasti technologie SharePoint, převzato z [12]

V následující části se jednotlivými oblastmi a možnostmi SharePoint budeme zabývat podrobněji, informace pro tuto část byly čerpány z [13].

Stránky

Pod pojmem stránky jsou myšleny internetové stránky, které je možno pomocí nástroje SharePoint snadno vytvářet či upravovat. Uživatel ovládá systém pomocí pásu karet (*Ribbon*), podobně jako u balíku Office od společnosti Microsoft. Pás karet lze přizpůsobovat podle potřeby. Při editaci stránek je ovládání podobné jako u balíčku Office (například se využívá systém drag-and-drop, systém vkládání objektů, formátování písma apod.). SharePoint nabízí zabezpečení stránek, jejich položek a objektů podle jednotlivých přístupových práv uživatelů. Další výhodou je možnost pracovat s dokumenty přímo v prostředí SharePoint pomocí aplikace Office Web Applications.

Komunita

V této oblasti je patrná inspirace vývojářů sociální sítí Facebook a tak má uživatel SharePoint možnost vytvářet si vlastní profil, přidávat komentáře ke každému dokumentu, hodnotit jej nebo ho dokonce označit (systém zvaný tagging). Uživatelé mohou využívat pro sdílení znalostí systém interních blogů, který je upraven tak, aby odpovídal potřebám organizace z hlediska správy obsahu i zabezpečení.

Obsah

SharePoint nabízí mnoho nástrojů a funkcí pro práci s dokumenty. Každý dokument má svoje unikátní ID, což je užitečné při odkazování na tento dokument. I při změně názvu dokumentu nebo jeho přesunu do jiné lokality zůstane použitý odkaz platný. Další užitečnou funkcí je možnost uzamčení určité verze dokumentu (např. odsouhlasený plán investic), přestože se na dokumentu bude i nadále

pracovat. SharePoint také nabízí zjednodušený systém vytváření a používání metadat nad dokumenty. Tato metadata lze zadávat a editovat přímo v aplikacích typu Word, Excel a PowerPoint a technologie SharePoint 2010 nabízí nástroje k automatickému vytváření metadat z nového dokumentu.

Vyhledávání

SharePoint nabízí vyhledávání strukturovaného i nestrukturovaného obsahu, a to nejenom v lokalitách SharePoint, ale i v souborech na discích, webových stránkách, ve složkách Exchange a v databázích. Systém se učí na základě předcházejících hledání daného uživatele, jeho týmu apod. Vyhledávání lze upřesnit například podle formátu a stáří dokumentu, autorů, lokality dokumentu, nebo je možné vyhledávat kolegy a experty na určité téma.

Analýza

Technologie SharePoint obsahuje zabudované nástroje tzv. Business Intelligence, do které patří: reporty, scorecards, dashboards, možnost detailní analýzy dat, podpora datových kostek a kontingenčních tabulek. Uživatel má možnost práce s daty z podnikových systémů (funguje zde dokonce i obousměrná synchronizace), takže lze například s daty z ERP (*Enterprise Resource Planning*) nebo CRM (*Customer Relationship Management*) systémů pracovat na stránkách SharePoint, nebo zprostředkovaně dokonce v aplikacích balíku Office. Uživatelé mohou využít i speciální nástroje pro prezentaci grafů nebo propojení s Visio.

Kompozice

Uživatel má možnost vytvářet objekty a automatizovat procesy pomocí workflows, a to bez znalosti kódování. Systém podporuje webové databáze a aplikace vytvořené v programu Access. SharePoint nám nabízí možnost nastavit oprávnění, sledovat, spravovat a případně izolovat vytvořená uživatelská řešení. Technologie SharePoint má také zabudovanou podporu technologie Silverlight pro tvorbu interaktivního obsahu.

V této podkapitole byla platforma SharePoint představena v obecné rovině. Konkrétnější popis částí užitých v této práci, např. struktury databáze v této platformě, bude uveden v dalších kapitolách.

3.5 XML

Jazyk XML je jedním z nejvýznamnějších počinů ve vývoji syntaxe pro popis dokumentů v historii. Používá se v nejrůznějších oblastech, jako jsou např. justice, finančnictví, zdravotnictví, zemědělství ap. XML lze považovat za nejrobustnější, nejspolehlivější a nejflexibilnější syntaxi pro dokumenty, jaká byla doposud vyvinuta [4]. Hlubší popis tohoto jazyka by byl nad rámec této práce, proto je tato kapitola zestručněna jen na nezbytné minimum. Další doplňující informace je možné najít v mnoha publikacích zabývajících se jazykem XML. Tento jazyk, a na něm založený standard XMI, je v diplomové práci použit k popisu reprezentace UML modelů a výměně dat reprezentujících tyto modely.

3.5.1 Původ XML

Jazyk XML (v překladu „rozšiřitelný značkovací jazyk“) je následníkem jazyka SGML. Standard SGML vznikl již v roce 1986. SGML byl vytvořen jako sémantický a strukturovaný značkovací jazyk určený pro textové dokumenty. Největším úspěchem SGML je jeho aplikace jazyk HTML, který je specializovaný čistě pro popis webových stránek a přirozeně nenabízí celou množinu schopností SGML. Pro jiné využití SGML, například jako prostředku pro správu velkého množství dokumentů nebo jako formát pro výměnu dat, se nabízejí jiné aplikace. Přenositelnosti a kompatibilitě jednotlivých aplikací SGML však stála v cestě překážka v podobě velice složité a obsáhlé specifikace SGML, která pokrývá mimo jiné i velké množství zvláštních a nepravděpodobných situací. Dá se říci, že neexistuje aplikace, která plně splňuje specifikaci SGML a rozdíly mezi aplikacemi bývají často významné - vlastnost, kterou jedna aplikace považuje za podstatnou, druhá vůbec takto vidět nemusí a neimplementuje ji.

V roce 1996 započaly práce na vývoji odlehčené verze SGML - takové verze, která by zachovala většinu užitečných vlastností SGML a vynechala rysy, které byly mnoha uživatelům nejasné, byly příliš komplikované na implementaci nebo se neprokázala jejich užitečnost. Na tomto projektu pracoval jedenáctičlenný tým s podporou mnoha desítek dalších specialistů. Výsledkem se stala v roce 1998 první verze XML 1.0, která slavila okamžitý úspěch. Standard XML definuje obecnou syntaxi, která se používá pro označování dat jednoduchými, člověku srozumitelnými značkami. Ustanovuje flexibilní formát pro počítačové dokumenty.

Následovaly další standardy, určitým způsobem rozšiřující či doplňující XML, jako například XML Namespaces, jazyk pro stylové šablony XSL nebo odkazovací jazyk XLL. Tyto standardy pak prošly dalším vlastním vývojem. K samotné základní specifikaci XML přibyla časem řada dalších rozšíření jako například XML Schemas nebo XHTML [4].

3.5.2 Struktura XML dokumentu

Dokument XML se skládá z logické a fyzické struktury. Logická struktura má za úkol rozdělit dokument na pojmenované jednotky a podjednotky nazývané elementy, zatímco fyzická struktura umožňuje vkládat do dokumentů samostatně pojmenované části dokumentu zvané entity. Entity mohou být i v samostatných datových souborech, aby mohly být opakovaně použity v různých dokumentech, a také aby bylo možné odkazovat na data, která nejsou ve standardu XML (například obrázky či binární soubory) [5].

Logická struktura dokumentu

Jak již bylo uvedeno, logická struktura dokumentu je tvořena elementy. Tyto elementy se v textu vyznačují pomocí tzv. značek (tagů). Většina elementů má dvě značky: počáteční a ukončovací. Názvy značek se zapisují mezi lomené závorky (tedy znaky < a >), aby se tak odlišily od zbytku textu. Pokud je v textu třeba tyto znaky použít, je nutné je nahradit zástupnými textovými entitami < a >. Ukončovací značka má před svým názvem ještě znak /, aby se tak odlišila od počáteční.
Ve jménech elementů se rozlišují malá a velká písmena, takže například <nazev>, <Nazev> a <NAZEV> jsou tři různé elementy. Elementy mohou obsahovat další vnořené elementy, a tím dle potřeby zachycovat strukturu informací uložených v dokumentu [2]. Celý dokument je vložen do jediného kořenového elementu dokumentu.

Elementy mohou kromě svého jména obsahovat další informace, které dále upřesňují jeho obsah. Tyto informace se nazývají atributy. Každý atribut má své jméno a element může obsahovat i více než jeden atribut. Obsah atributu musí být ohraničen v uvozovkách nebo apostrofech. Následuje ukázka s jednoduchým příkladem.

```
<osoba>
    <jmena>Jan</jmena>
    <prijmeni>Novák</prijmeni>
    <kontakty>
      <kontakt type="telefon">777123321</kontakt>
      <kontakt type="email">jan.novak@email.cz</kontakt>
    </kontakty>
</osoba>
```

Fyzická struktura dokumentu

Tak jako musí mít každý XML dokument logickou strukturu složenou z elementů, musí mít i fyzickou strukturu, která je tvořena entitami. Entity reprezentují paměťové jednotky (např. soubor v počítači), ve kterých jsou uloženy jednotlivé dokumenty, z nichž se skládá fyzická struktura dokumentu. Každá entita má obsah a identifikátor (např. URI). Jedna entita má mezi ostatními výjimečné postavení.

Jedná se o tzv. entitu dokumentu (*Document entity*), která je výchozím bodem při fyzickém rozčleňování dokumentu. Entity jsou vzájemně propojeny pomocí odkazů.

Existují dva základní typy fyzických entit - entity analyzované (*Parsed entities*) a entity neanalyzované (*Unparsed entities*). Je-li entita analyzovaná, pak se na ní můžeme dívat jako na nedílnou součást entity dokumentu. Odkaz na analyzovanou entitu je nahrazen jejím obsahem. Analyzovaná entita musí být správně vytvořeným XML dokumentem (*well-formed*). Oproti tomu neanalyzované entity obsahují data, která nejsou přímou součástí XML dokumentu. Jedná se např. o různé obrázky, textové soubory, hudbu, atp. Tyto entity nemusí splňovat požadavky na správně vytvořený XML dokument.

3.5.3 DTD

Definice typu dokumentu (DTD) určuje, které elementy a atributy mohou být v dokumentu použity, a v jakých vzájemných vztazích. DTD je tedy nástroj, který kontroluje, zda mají dokumenty správnou strukturu. Ve světě se používá mnoho DTD, které vyhovují různým požadavkům. Jako příklad lze uvést DTD DocBook, sloužící k definici elementu a atributů vhodných pro značkování technické dokumentace.

Hlavní výhodou při používání DTD je možnost využití parseru (programu, který kontroluje, zda je dokument správně strukturovaný) pro kontrolu, zda má dokument strukturu odpovídající danému DTD. Další výhodou je možnost použití různých jednoúčelových nástrojů navržených pro konkrétní DTD [5].

3.5.4 XMI

XMI je zkratkou výrazu XML Metadata Interchange. XMI je standard pro výměnu informací ve formě metadat založený na XML. Můžeme ho použít pro jakákoliv metadata, která popisují metamodel vyjadřující Meta-Object Facility (MOF). Nejčastější využití nachází jako formát pro výměnu dat UML modelů. Za vytvořením jazyka stojí skupina OMG a několik předních softwarových firem. XMI umožňuje přenášet data UML popisující návrhy informačních systémů, softwarové komponenty, schémata databází mezi různými systémy, což dříve nebylo možné z důvodu velké rozlišnosti formátů jednotlivých systémů [19].

Nabízí se zde otázka, jaké výhody přináší XMI oproti XML. Pomocí XML lze vytvářet strukturovaná data a jejich syntaktické specifikace, zatímco XMI představuje mechanizmus pro kódování MOF do jediné syntaktické struktury XML.

4 Analýza

4.1 Struktura dat v nástroji Process Inspector

Jak již bylo uvedeno, Process Inspector je nástroj pro získávání informací o struktuře podnikových procesů. Tato kapitola shrnuje, jaká data a v jaké formě se v tomto nástroji shromažďují za účelem dalších analýz.

Process Inspector ukládá data v databázi, která je součástí nástroje SharePoint, další podkapitola se proto věnuje podrobnějšímu popisu této databáze.

4.1.1 SharePoint databáze

Data v SharePoint Services jsou uložena v položkách listů (*lists*), nejedná se teda o klasickou relační databázi. Struktura položek v listu je tvořena sloupci (*columns*). Sloupce mohou být odvozeny od předdefinovaného obsahového typu (*content type*) nebo nadefinovány přímo – vytvořením nového sloupce v listu.

Sloupce mohou být předdefinovány nezávisle na listech a jejich definice tak může být společná pro celou stránku (*site*) – to znamená, že takto definovaný sloupec lze využít jako vzor pro definování struktury listu. Následující tabulka uvádí přehled základních datových typů sloupců.

Název datového typu	Popis datových typů
Single line of text	Textový řetězec
Multiple line of text	Text s podporou formátování
Choice	Výběr z dané množiny řetězců
Number	Číslo s možností specifikace desetinných míst a rozsahu
Currency	Měna s možností specifikace desetinných míst, rozsahu a symbol měny
Date and Time	Datum a čas
Lookup	Vazba na pole listu obsaženém v rámci jedné site.
Yes/No	Dvouhodnotová proměnná typu ANO/NE (boolean)
People or Group	Uživatelé a skupiny site
Hyperlink or Picture	Hyperlink
Calculated	Kalkulované pole z existujících polí položky (řádku) vyjma polí vazby (Lookup)

Tabulka 3: Základní datové typy sloupců

Obsahové typy se mohou skládat z již existujících sloupců v rámci stránky nebo obsahovat (stejně jako list) přímo nadefinované sloupce. Každý obsahový typ je odvozen od nějakého předka – elementárně od výchozího obsahového typu –„Item“, který je definován systémem a od něhož jsou odvozeny všechny dále deklarované obsahové typy.
Každý list (kromě obsahových typů) může mít definovány další přidružené sloupce – metadata.
V následující tabulce je přehled jednotlivých metadat [16]. V této práci je využita pouze položka ID.

Položka metadat	Popis položky
ID	Jednoznačný automaticky inkrementovaný identifikátor
Modified	Datum poslední změny položky
Created	Datum vytvoření položky
Created By	Uživatel, který vytvořil danou položku
Modified By	Uživatel portálu (site), který změnil položku listu
Version	Verze položky listu
Attachements	Souborové přílohy k položce listu
Approval Status	Stav schválení provedených změn
Content Type	Sloupec, který identifikuje obsahový typ položky listu

Tabulka 4: Přehled přidružitelných sloupců

4.1.2 Porovnání tradiční relační databáze s SharePoint databází

SharePoint databáze nepoužívá klasické databázové tabulky, ale nahrazuje je takzvanými listy.
SharePoint listy se skládají z řádků a sloupců, ve kterých se ukládají data, stejně jako v tradiční relační databázi, např. v SQL Serveru. Výhoda listů spočívá v tom, že jsou propojeny s webovou částí, která umožňuje jednoduchou práci s těmito listy. Hlavní výhodou těchto listů je jejich snadné a rychlé použití. Při větším množství položek v listu (tým produktu SharePoint doporučuje počet položek omezit na maximálně 2 000) se projeví, že listy jsou obecně méně výkonné než relační databáze.
Relační databáze je oproti listům komplexnější a podporuje složitější vztahy.
SharePoint databáze nepodporuje jazyk SQL, ale používá jazyk CAML.

Schéma dotazů Collaborative Application Markup Language (CAML) se používá různými způsoby v kontextu k platformě Microsoft SharePoint pro definici nad daty v listech. CAML dotazy se používají v kontextu CAML náhledů k získání specifikovaných údajů [22].

V následující tabulce jsou prezentovány rozdíly při použití databázových tabulek a SharePoint listů. U položky „snadnost ovládání“ je třeba vycházet z toho, že uživatel neovládá ani SharePoint listy, ani relační databázi.

Výhody	Databáze	SharePoint listy
Zpracovávání komplexních vztahů mezi daty	Ano	Ne
Zpracovávání velkého množství položek	Ano	Ne
Zpracovávání transakcí	Ano	Ne
Snadnost ovládání	Ne	Ano
Obsahuje standardní rozhraní	Ne	Ano
Snadná možnost přidání binárních dat	Ne	Ano

Tabulka 5: Shrnutí výhod databáze a SharePoint listů

4.1.3 Listy nástroje Process Inspector

Hlavní úkol nástroje Process Inspector spočívá v získávání informací o podnikových procesech. V každé firmě zaměstnanci provádí mnoho procesů. Příkladem takového procesu může být například proces „prodej zboží“, který se může dále skládat z dalších podprocesů, jako např. „určení ceny“, „přijetí platby“ nebo „vystavení faktury“, které se mohou větvit na další podprocesy atd. Každý proces se v nástroji Process Inspector ukládá do SharePoint databáze ve formě listu. U každého listu procesu se v databázi zaznamenávají následující údaje:

List Procesu

- **Name** – určuje jméno daného procesu.
- **Description** – zde se nachází podrobnější popis daného procesu a uvádí se zde důležité informace, které není možno zaznamenat v jiných částech tabulky.
- **Frequency** – určuje četnost provádění daného procesu. Uživatel má na výběr z možností denně, týdně, měsíčně, čtvrtletně a ročně.
- **Area** – určuje, pod jaké oddělení v podniku tento proces spadá. Uživatel má na výběr z mnoha možností, jako jsou například marketing, prodej, IT, projektový management, logistika ap.
- **Type** – určuje typ procesu. Na výběr je zde jedna ze čtyř možností: administrativa (zdroje, lidé), opravování (námitky), vylepšení (znalost) a anonymní.
- **Worker** – určuje, kdo daný proces provádí, např. Jan Novák.
- **Unfinished** – na výběr jsou pouze dvě možnosti, určující, zda je proces dokončen či nikoli.

- **Details** – zde se ukládají podobné informace jako u Description s tím rozdílem, že Description je určen ke stručnému popisu na několik řádků, zatímco u Details se zaznamenává i několikastránkový podrobný popis daného procesu.
- **File Particulars** – ke každému procesu lze přiložit jakákoliv další doplňující data, která se nacházejí v podobě dalších souborů. Například k procesu „instalace programu“ můžeme přidat soubor s podrobným návodem ve formátu PDF ap.
- **File Description** – zatímco File Particulars obsahuje soubory, které slouží k objasnění podrobností o procesu, File Description obsahuje soubory popisující tento proces.
- **Parent** – tato informace je velmi důležitá k zachování struktury jednotlivých procesů. Každý proces je umístěn ve stromě procesů. Pokud proces nemá žádného rodiče, jedná se o kořen, pokud rodiče má, je jeho ID zaznamenáno v položce Parent pomocí datového typu Lookup.
- **Inputs** – každý proces může obsahovat různé vstupy, ID těchto vstupních dokumentů je zaznamenáno pod položkou Inputs typu Lookup.
- **Outputs** – zaznamenává ID výstupních dokumentů z tohoto procesu typu Lookup.

Ke každému procesu se mohou připojit různé vstupy či výstupy. Tyto vstupy a výstupy jsou reprezentovány pomocí dokumentů. Každý dokument musí být připojen k některému procesu a nemůže existovat osamoceně. U každého dokumentu se ukládají následující údaje (údaje, odpovídající údajům u procesů, jsou popsány jen stručně).

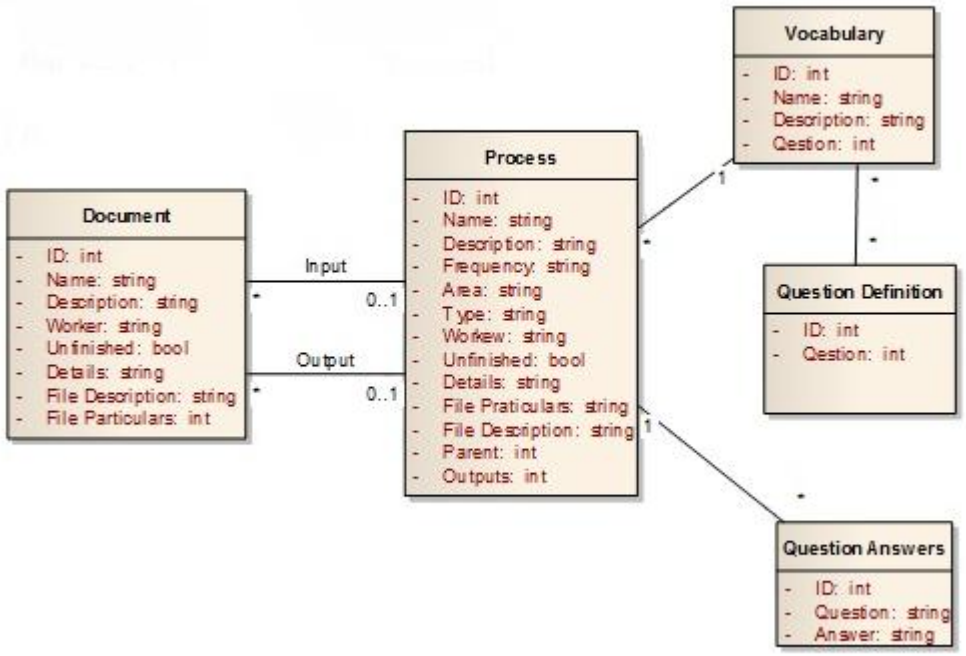
Dokument

- **Name** – jméno dokumentu.
- **Description** – popis daného dokumentu, obsahuje důležité informace, které není možno zaznamenat v jiných částech tabulky.
- **Worker** – autor dokumentu.
- **Unfinished** – stav dokončení.
- **Details** – podrobné detaily.
- **File Particulars** – soubory s podrobnostmi.
- **File Description** – soubory s popisem.

V databázi se dále nachází tyto listy:

- **Vocabulary** – obsahující slovník jmen jednotlivých procesů a dokumentů. Tato jména se přiřazují zadaným dokumentům a procesům pro sjednocení terminologie.
- **Question Definition** – na základě typu procesu se k němu přiřadí jednotlivé otázky sloužící k upřesnění informací o daném procesu.
- **Question Answer** – k přiděleným otázkám jsou přiděleny i odpovědi, ze kterých uživatel vybírá tu nejvíce odpovídající.

Pro účel této práce jsou důležité pouze listy Process a Document. Ostatní listy nebudou mít žádný vliv na grafickou reprezentaci procesů v jazyce UML, a tak se jimi nebudeme hlouběji zabývat. Přehled jednotlivých listů a vztahů mezi nimi je zobrazen na obrázku 7.



Obrázek 7: Listy databáze nástroje Process Inspector

4.2 Grafický model procesů

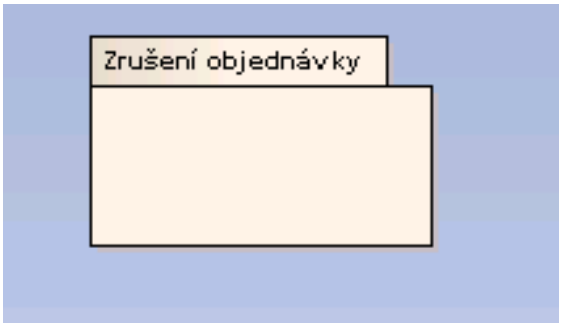
V současné době nástroj Process Inspector nenabízí možnost zobrazení procesů pomocí některého z mnoha UML diagramů. V této části si ukážeme, jak a které procesy zobrazovat. Na úvod je třeba určit, která data by bylo vhodné v UML diagramu zobrazovat. Vyjdeme z předchozí podkapitoly, popisující strukturu databáze a v ní obsažené listy.

List Vocabulary obsahující slovník pojmenování ke sjednocení terminologie sice umožňuje správně pojmenovat jednotlivé procesy, ale při grafickém zobrazení procesů by k ničemu neposloužil a tak jej nebudeme zobrazovat. Stejně tak i listy Question Definition a Question Answer není třeba zahrnout do UML diagramů.

Zbývají listy Process a Document, což budou jediné listy zobrazené v našem UML diagramu. Jak již bylo zmíněno dříve, jazyk UML nabízí řadu různých diagramů. Pro modelování procesů byl zvolen diagram tříd, který poskytuje všechny potřebné nástroje pro náš model. Při modelování budou využívány pouze takové elementy, které u diagramu tříd nabízí nástroj Enterprise Architect verze 8.0.864, a to z důvodu budoucího přenesení navrženého diagramu na tento nástroj, na němž bude tento UML diagram dále zpracováván. Všechna jména jednotlivých elementů v následující části vychází z pojmenování použitých v nástroji Enterprise Architect.

Nejjednodušší je namodelovat pouze jeden proces nebo jeden dokument. Jednotlivé procesy budou v UML diagramu reprezentovány za pomocí elementu třída (*Class*) a vlastnosti těchto procesů budou reprezentovat jednotlivé atributy těchto tříd. Pro list Document (reprezentující různá vstupní a výstupní data procesů) bude v našem diagramu tříd užíván element s názvem Document, a analogicky s procesy i zde budou jednotlivé vlastnosti vyjádřeny atributy dokumentu.

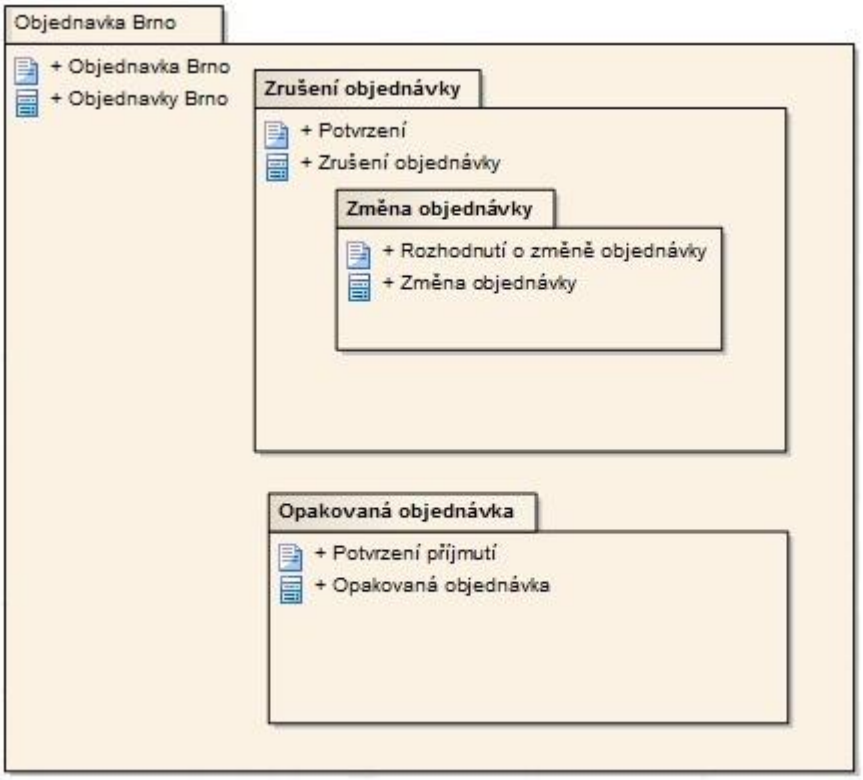
Jednotlivé dokumenty však nebývají nezávislé entity, ale jsou někým vytvořeny, přesněji nějakým procesem. V našem třídním diagramu bude tento vztah znázorňován pomocí elementu s názvem Dependency. Na obrázku 8 je ukázka velmi jednoduchého třídního diagramu vytvořeného v nástroji Enterprise Architect, jenž znázorňuje jeden proces.



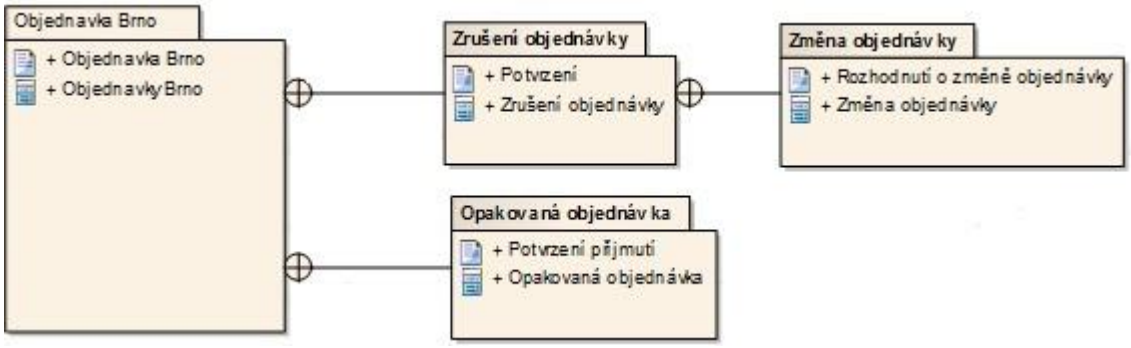
Obrázek 8: Diagram reprezentující jeden proces

Je však zřejmé, že takto jednoduchých třídních diagramů bude v nástroji Process Inspector jen minimum. Většina procesů se skládá z dalších podprocesů, jež se větví na další podprocesy, a mohou tak vytvářet dosti rozsáhlý strom procesů.

K zaznamenání hierarchické struktury se v diagramu tříd používají takzvané balíčky, v nástroji Enterprise Architect jsou reprezentovány elementy s názvem Package. Při použití těchto balíčků znázorníme jednotlivé procesy pomocí elementu Package, který může obsahovat element Class s podrobnějším popisem tohoto procesu, dále může obsahovat dokumenty a vztahy mezi entitami, ale hlavně může obsahovat další balíčky, které mohou obsahovat další balíčky, a tak je možno zachytit jakkoli rozsáhlý strom procesů. Na obrázcích 9 a 10 jsou zobrazeny dva ekvivalentní třídní diagramy zachycující stromovou strukturu procesů.



Obrázek 9: Zachycení hierarchie procesů



Obrázek 10: Zachycení hierarchie procesů (jiné grafické znázornění)

Výše popsaným způsobem je možné vytvořit UML diagram pro jakékoli procesy nacházející se v nástroji Process Inspector.

5 Implementace

5.1 Technické prostředky k implementaci

Process Inspector je vyvíjen na platformě SharePoint, rovněž vyvíjený nástroj tedy bude využívat tuto platformu. SharePoint podporuje dva hlavní nástroje k vytváření SharePoint aplikací. Prvním z nich je SharePoint Designer.

5.1.1 SharePoint Designer

Jedná se o aplikaci, která je určena pro navrhování webů a aplikací, používá se k sestavování a přizpůsobování aplikací a webů služby SharePoint. Jak uvádí prodejce tohoto programu „*Pomocí aplikace SharePoint Designer 2010 můžete vytvářet stránky s bohatým obsahem, sestavovat výkonná řešení s podporou pracovních postupů a navrhovat vzhled a chování svého webu. Můžete vytvářet různě rozsáhlé weby, od malých webů pro řízení projektů až po portálová řešení s řídicími panely pro velké podniky. Aplikace SharePoint Designer 2010 poskytuje jedinečné možnosti pro vytváření webů, neboť v ní lze na jednom místě vytvořit web, přizpůsobit jeho součásti, navrhnout logiku webu na základě firemního procesu a nasadit web jako balíček řešení. Všechny uvedené činnosti můžete provést, aniž byste napsali jediný řádek kódu*“ [11]. Tento nástroj je při implementaci v této práci využíván jen příležitostně, a to pro vkládání prvků do aplikace, a nebo při práci s databází, tyto činnosti se v tomto nástroji provádí pouhým kliknutím na některou z ikon, což uživateli ušetří hodně času. K vytváření vlastního zdrojového kódu je však mnohem vhodnější druhý podporovaný nástroj - Visual Studio 2010.

5.1.2 Microsoft Visual Studio 2010

Visual Studio 2010 je integrované vývojové prostředí pro vývoj aplikací, a to jak formulářových, tak i webových aplikací a stránek, webových služeb a dalších. V současné době zahrnuje 14 projektových šablon pro vývoj služby SharePoint. Obsahuje také vizuální návrháře webových částí a balíčků.

Toto prostředí zahrnuje textový editor s podporou systému IntelliSence, který programátorovi nabízí interaktivní výpisy vlastností a metod objektů. Nabízí pokročilý nástroj na opravu chyb ve zdrojovém kódu [22]. Dále je součástí překladač s možností překladu do binárního kódu nebo mezikódu. Obsahuje také nástroj pro návrh formulářů pro aplikace s grafickým uživatelským rozhraním, pro návrh webových aplikací, schématu databáze a mnoho dalších, které napomáhají efektivnímu a rychlému vývoji aplikací.

Visual Studio podporuje řadu různých programovacích jazyků. SharePoint projekty však mohou být vytvářeny pouze v jazyce Visual Basic a nebo Visual C#. Vzhledem k tomu, že Process Inspector je psán v jazyce Visual C# , je k tvorbě nástroje na import dat v této práci užit tento jazyk.

5.1.3 Visual C#

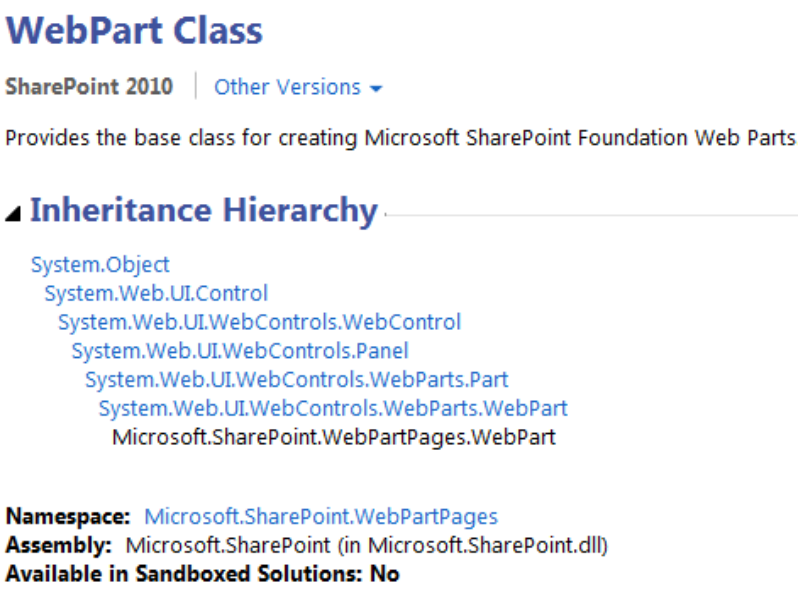
Jedná se o poměrně nový jazyk vytvořený společností Microsoft, jehož začátky se datují do roku 1999, kdy Anders Hejlsberg sestavil tým, který vytvořil jazyk Cool, jenž byl v roce 2000 přejmenován na C#. C# je jednoduchý, moderní, objektově-orientovaný jazyk. Inspiruje se jazyky C++ a Java a snaží se využít jejich nejlepších vlastností a naopak odstranit ty, které programátorům znepříjemňují tvorbu aplikací. Pro usnadnění vývoje v tomto jazyku slouží například detekce překročení hranic pole, detekce použití neinicializovaných proměnných nebo automatická správa paměti a rušení objektů.

Program v jazyku C# je překládán téměř ve všech případech do mezijazyka CIL (Common Intermediate Language). Tento kód se poté spouští v běhovém prostředí, což bývá nejčastěji .NET Framework, který zajišťuje běh programů v operačním systému Windows [14]. Nezávisle na .NET Framework jsou vyvíjena prostředí Mono a DotGNU, implementující běhové prostředí pro Unixové Systémy.

5.2 Webová část

Implementovaný nástroj na export dat bude realizován jako webová část (Web Part), kterou bude možno vložit na jakoukoliv stránku v nástroji Process Inspector. Webové části umožňují uživatelům přímo upravovat obsah, vzhled a chování stránek a aplikací vytvářených v nástroji SharePoint. Webové části jsou serverovými ovládacími prvky. Každá webová část přebírá kontext SharePoint stránky, na které se nachází. Uživatel má možnost vytvořit si vlastní webové části, ty musí být závislé (dependency) na Microsoft.SharePoint.dll a musí dědit z třídy WebPart ve jmenném prostoru Microsoft.SharePoint.WebPartPages [22]. Druhou možností je využít některou z již předpřipravených webových částí nacházejících se ve Visual Studiu 2010. V této práci byla využita předpřipravená webová část Visual Web Part.

Obrázek 11 uvádí bližší popis třídy WebPart.

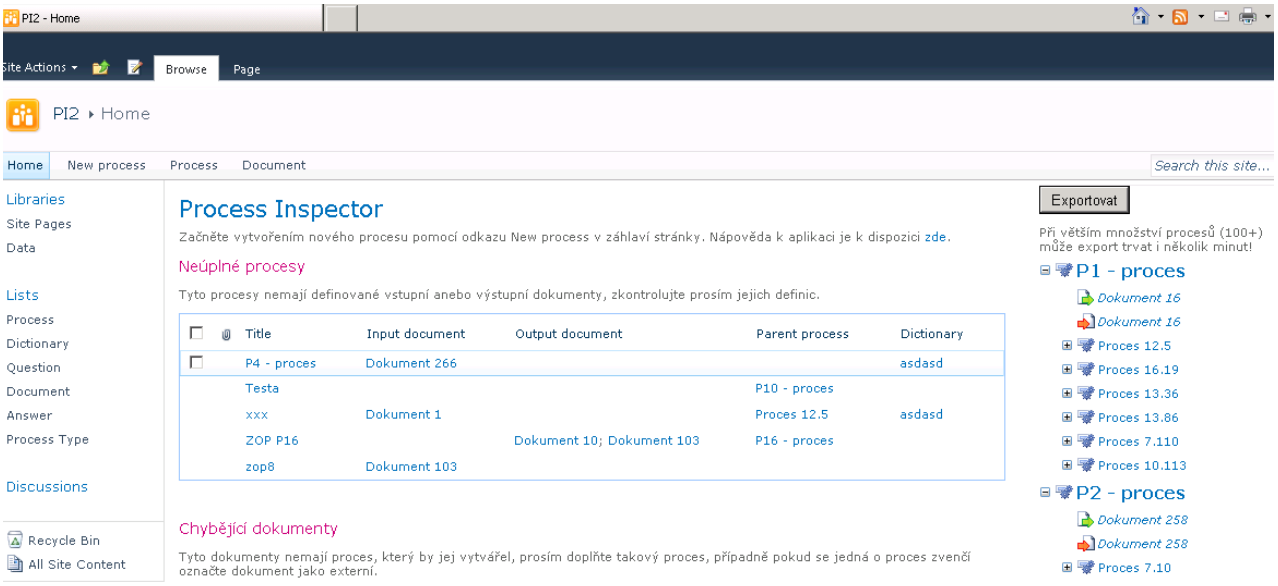


Obrázek 11: Hierarchie třídy WebPart, obrázek převzat z [22]

5.3 Koordinační třída

Každá webová část obsahuje tlačítko s nápisem Export. Po kliknutí na toto tlačítko se začne vytvářet soubor ve formátu XML. Tento soubor reprezentuje zvolené procesy, všechny jejich podprocesy, vstupy, výstupy z procesů a vztahy mezi nimi. Soubor je určen k následnému importu do nástroje Enterprise Architect. Po zmáčknutí tohoto tlačítka se vyvolá událost, jež zavolá třídu **ExportProcess**.

Na následujícím obrázku (obr. 12) je uveden příklad vloženého exportu dat na jednu ze stránek nástroje Process Inspector. Vzhled webové části je velmi jednoduchý a lze jej snadno upravit podle vzhledu stránky, na kterou je umístěn.



Obrázek 12: Vložení webové části k exportu dat do nástroje Process Inspector

Třída **ExportProcess** představuje hlavní třídu tohoto projektu, která má za úkol koordinovat činnost ostatních tříd. Poté, co je tato třída zavolána stisknutím tlačítka, z databáze získá všechny procesy, které se nacházejí na stránce, na které je toto tlačítko umístěno, a postupuje následujícím způsobem: pomocí třídy **XmlFile** vytvoří „prázdný“ XML soubor. Pro každý ze získaných procesů provede následující - přes třídu **XmlFile** přidá do prázdného XML souboru zvolený proces. Dále z databáze získá seznam všech potomků zvoleného procesu a seznam všech vstupních a výstupních dokumentů. Opět za pomoci třídy **XmlFile** vše zapíše do XML souboru, přičemž zapíše i vztahy mezi jednotlivými entitami. Výše popsaný proces (vyjma tvorby prvotního prázdného XML souboru) se postupně opakuje i pro všechny potomky zvoleného procesu.

Tato třída obsahuje metodu **export**, jenž načte data z databáze a vybere všechny procesy jenž nemají žádného předchůdce (kořenové procesy). Pro každý tento proces se za pomocí třídy **XmlFile** vytvoří balíček v kořenovém balíčku, do těchto balíčků se vloží grafická plocha k zobrazování elementů a třída reprezentující daný proces. Pro každý z podprocesů jednotlivých procesů se zavolá metoda **addChild**, která vloží všechny požadované prvky do balíčku, a tímto způsobem dále postupuje, dokud nezapíše všechny podprocesy ve stromové hierarchii.

5.4 Práce s daty

SharePoint umožňuje snadné získávání dat na základě toho, která stránka je právě aktuální. Následující příklad ukazuje, jak získat data nacházející se v listu Process na aktuální stránce.

```
SPWeb mySite = SPContext.Current.Web;
SPList myList = mySite.Lists["Process"];
```

Databáze v SharePoint nepodporuje SQL jazyk, ale má vlastní způsob na dotazování, tzv. CAML Query.
Následující kód ukazuje způsob, jak získat kolekci všech položek procesu s požadovaným ID.

```
SPQuery query = new SPQuery();
query.Query = "<Where><Eq><FieldRef Name=\"ID\"/>" +
              "<Value Type=\"Counter\">" + id + "</Value></Eq></Where>";
SPListItemCollection items = myList.GetItems(query);
```

V SharePoint databázi má každá položka automaticky přiděleno své unikátní ID. Jedná se o číselnou hodnotu, která je však uložena jako datový typ string, toto ID lze využít jako primární klíč. Cizí klíče v SharePoint databázi jsou simulovány pomocí náhledů, tzv. „Lookup“. Ty umožňují přístup k sloupcům jiných tabulek, které se však musí nacházet pouze na úrovni aktuálního webu (není zde možnost propojení sloupců umístěných v různých webech či kolekcích webů).
Náhled (lookup) je tvořen jednou hodnotu typu string. Tato hodnota se dále skládá z části ID, odkazující na další sloupec, tato hodnota je oddělena středníkem, za nímž následuje jméno sloupce začínající znakem křížek. Pro názornost následující příklad:

```
ID;#String
```

Ke zpracování náhledu třída **ExportProcess** obsahuje metodu **getId**, této metodě se předá náhled (lookup) a vrátí hodnotu ID, která odpovídá ID prvního odkazovaného sloupce v náhledu. Dále pak metodu **getIdList**, která z předaného náhledu vrátí list jednotlivých ID odkazovaných sloupců. Metodě **childList** se zadá ID zvoleného procesu a vrátí list obsahující ID všech podprocesů tohoto procesu.

5.5 Tvorba výsledného souboru k exportu

Třída **XmlFile** představuje hlavní třídu, díky níž se vytváří výsledný soubor, jenž prezentuje procesy popsané v nástroji Process Inspector a vztahy mezi nimi.

K vytváření toho souboru je využívána třída **List**. Tato třída se nachází ve jmenném prostoru System.Collections.Generic, jenž je součástí .NET Framework od verze 2.0. Třída **List** reprezentuje list objektů, které mají stejný datový typ a je možné k nim přistupovat pomocí indexu. Tato třída poskytuje řadu metod k prohledávání, řazení, vkládání a manipulování s listem.

Díky těmto vlastnostem je vhodné ji použít jako reprezentanta výsledného souboru, ke kterému se bude často přistupovat a do nějž budou vkládána nová data. Každý řádek výsledného souboru je reprezentován jednou položkou v listu, což umožňuje snadnou manipulaci s jednotlivými řádky. Následuje ukázka vytvoření instance třídy **List**.

```
List<string> myXmlFile = new List<string>();
```

Tento kód se nachází v první popisované metodě třídy **XmlFile** - metodě s názvem **emptyFile.** Jedná se o první metodu, která je použita při vytváření nového souboru k exportu dat. Při zavolání této metody je vytvořena instance třídy **List**, do níž se postupně přidají položky (řádky výsledného souboru), které vytváří „prázdný“ soubor k importu do nástroje Enterprise Architect. Pod označením „prázdný“ soubor je myšlen soubor, v němž sice nejsou vloženy žádné procesy nebo dokumenty, ale je plně připraven ke vkládání nových procesů. Tento soubor lze importovat do nástroje Enterprise Architect a dále sním pracovat. Jako příklad je uvedeno vložení jednoho prvku (řádku) do listu.

```
myXmlFile.Add("<XMI.header>");
```

Tento počáteční soubor se skládá z deklarace XML souboru, jež určuje verzi a kódování tohoto souboru. Následuje ukázka:

```
<?xml version="1.0" encoding="windows-1252"?>
```

Hned za deklarací XML následuje Deklarace typu dokumentu, je zde použita deklarace určená pro nástroj Enterprise Architect, jak lze vidět z následující ukázky:

```
<!DOCTYPE XMI SYSTEM "UML_EA.dtd">
```

Pak již následuje samotný XML soubor. Enterprise Architect je schopen importovat UML modely jen ze souboru, který dodržuje XMI standard, a tak i vytvářený soubor tento standard dodržuje. Následující příklad uvádí deklaraci XMI standardu a hlavičky XMI souboru. Tato hlavička obsahuje dokumentaci k XMI souboru. V tomto příkladu se jedná o označení exportéra souboru a jeho verze.

```
<XMI.header>
	<XMI.documentation>
		<XMI.exporter>Enterprise Architect</XMI.exporter>
		<XMI.exporterVersion>2.5</XMI.exporterVersion>
	</XMI.documentation>
</XMI.header>
```

Poté již následuje samotný obsah tohoto souboru. Všechna data, která jsou určena k importu do nástroje Enterprise Architect, jsou umístěna v sekci content. Tato sekce je vyznačena značkami (tagy), jak je ukázáno na následujícím příkladu.

```
<XMI.content>
…
Data určená k importu
…
</XMI.content>
```

Sekce content je dále rozdělena na dvě části, a to na část Model, a na část, která obsahuje položky s označením Diagram. Část Model dále obsahuje podčást Namespce.onwnedElement, do této části se vkládají všechny elementy, které se nacházejí ve vytvářeném souboru. Metoda **emptyFile** do této části vkládá jeden balíček, do kterého jsou následně přidávány všechny procesy. Proces vkládání balíčků bude podrobněji popsán v dalších částech práce. Následuje ukázka, která zobrazuje zdrojový kód Modelu.

```
<XMI.content>
	<UML:Model name="EA Model" xmi.id="MX_EAID_001">
		<UML:Namespace.ownedElement>
			…
		</UML:Namespace.ownedElement>
	</UML:Model>
```

Za touto částí se mohou nacházet části s názvem Diagram, jež slouží k zobrazení jednotlivých elementů.

Metoda **emptyFile** vrací vytvořený list, se kterým se může dále pracovat.

Metodě **newPackage** se předá vytvářený soubor (list), jméno přidávaného balíčku, jeho ID a jméno a ID jeho předchůdce a vrátí nám soubor, který má přidán element Package. Jak bylo popsáno v kapitole Grafický model procesů, hierarchie procesů se v nástroji Enterprise Architect vytváří pomocí elementů balíčků (Package). Tyto balíčky se jmenují stejně jako proces, který reprezentují a vkládají se s každým přidaným procesem.

ID balíčků vždy začíná znaky „EAPK_“, za nimi mohou následovat další vybrané znaky. V této implementaci za tyto znaky volíme ID procesu. Díky tomu nenastane případ, kdy by dva balíčky sdílely stejné ID. Jméno a ID předchůdce odpovídá balíčku, do kterého chceme aktuální balíček vkládat a tyto informace známe z předchozí práce s tímto balíčkem, nebo se může jednat o kořenový balíček vložený metodou **emptyFile**, jehož jméno i ID také známe. Jednotlivá ID jsou velmi důležitá, protože se využívají při propojování jednotlivých elementů.

Metoda **newPackage** postupně prochází řádky souboru (položky listu), dokud nenarazí na řádek, na kterém je zapsáno jméno a ID balíčku odpovídající zadanému balíčku předchůdce. Následně do prvního vyskytujícího jmenného prostoru zapíše balíček.

Následuje zjednodušený příklad jednoho přidaného balíčku, který bude následně podrobněji popsán.

```
<UML:Package name="Process3" xmi.id="EAPK_10" isRoot="false" isLeaf="false">

	<UML:ModelElement.taggedValue>
	</UML:ModelElement.taggedValue>

	<UML:Namespace.ownedElement>
	<UML:Collaboration xmi.id="EAID_10_Collaboration" name="Collaborations">
		<UML:Namespace.ownedElement>
		</UML:Namespace.ownedElement>
	</UML:Collaboration>
	</UML:Namespace.ownedElement>
</UML:Package>
```

Příklad uvádí přidání balíčku s názvem Process3. Proces, který tento balíček zastává, má v databázi nástroje SharePoint hodnotu ID rovnu 10, což lze vyčíst z položky xmi.id. Po deklaraci balíčku se nachází sekce <UML:ModelElement.taggedValue>, kam se vkládají různé nepovinné značkovací hodnoty (Tagged Value). Tyto hodnoty slouží k přidávání dalších informací, kromě těch, které jsou již podporovány v UML, ke zvolenému elementu. Tyto hodnoty jsou používány k importu do různých nástrojů, jako je například Enterprise Architect. I když neuvedeme žádnou z těchto

hodnot, je možné po importu do nástroje Enterprise Architect tyto hodnoty různě upravovat anebo vkládat nové hodnoty. Následující tabulka uvádí pro příklad některé z mnoha hodnot.

Jméno	Ukázka hodnoty	Popis značkovací hodnoty
created	2011-04-13 12:43:19	Datum vytvoření balíčku
modified	2011-04-13 12:59:15	Datum poslední modifikace balíčku
lastloaddate	2011-04-13 13:01:59	Datum posledního přečtení balíčku
version	1.0	Určuje verzi k importu pro nástroj Enterprise Architect
ea_stype	Public	Upřesnění typu elementu
parent	EAPK_02	ID nadřazeného balíčku
author	Jan	Jméno autora balíčku

Tabulka 6: Přehled některých značkovacích hodnot

Za sekcí značkovacích hodnot následuje jmenný prostor balíčku. Jak již bylo zmíněno dříve, ve jmenném prostoru se nacházejí všechny elementy ve zvoleném elementu. Při vytvoření nového balíčku se do něj automaticky vkládá i element spolupráce (Collaboration). Do tohoto elementu se vkládá element ClassifierRole, který bude popsán v dalších částech této práce. ID spolupráce začíná znaky „EAID_“, za nimiž následují další znaky. Je zde opět použito číslo procesu, které je zakončeno znaky „_Collaboration“.

Další metoda **addClassfierRole** vloží do zvoleného elementu spolupráce element ClassfierRole.

Metoda **newLogicDiagram** vytvoří nový diagram. Jak již bylo zmíněno, všechny diagramy se nacházejí v sekci content, za sekcí model. Tento diagram se vytváří pro zvolený balíček, reprezentuje grafickou plochu (tj. plošný graf), na které se zobrazují elementy v daném balíčku a vztahy mezi nimi. Pro vkládání těchto elementů slouží další metoda **addDiagramElement**, jež vloží element do zadaného Diagramu. Následuje ukázkový příklad, který bude následně podrobněji vysvětlen.

```
<UML:Diagram name="Proces1" xmi.id="EAID_1_DIAGRAM" diagramType="ClassDiagram"
owner="EAPK_1" toolName="Enterprise Architect 2.5">
       <UML:ModelElement.taggedValue>
       </UML:ModelElement.taggedValue>
       <UML:Diagram.element>

<UML:DiagramElement geometry="Left=172;Top=234;Right=302;Bottom=314;"
subject="EAID_13_Doc" seqno="1" />
<UML:DiagramElement geometry="Left=10;Top=10;Right=100;Bottom=80;" subject="EAID_1_Class"
seqno="1" />
<UML:DiagramElement geometry="SX=0;SY=0;EX=0;EY=0;Path=;0 " subject="EAID_1_13" />
       </UML:Diagram.element>
</UML:Diagram>
```

Z prvního řádku příkladu lze určit jméno a ID diagramu, jeho typ, skutečnost, že náleží k balíčku EAPK_1 a že byl vytvořen nástrojem Enterprise Architect. Následuje sekce značkovacích hodnot, která má stejný význam jako u již zmíněného balíčku. Následuje sekce Diagram.element, do níž se vkládají elementy, které se mají v tomto diagramu zobrazovat. V uvedeném příkladu se nachází jeden element třídy a jeden dokument, poslední položka značí vztah mezi elementy. Hodnota geometry určuje pozici, kde se má element nacházet.

Metoda **addNewPack** má za úkol provést postupně metody **newPackge**, **newLogicDiagram**, **addDiagramElement** a **addClassifierRole,** což jsou všechny potřebné metody k vytvoření nového balíčku, vložení do něj grafické plochy k zobrazování elementů a zaznamenání tohoto balíčku do grafické plochy nadřazeného balíčku.

Tak jako metoda **addNewPack** vloží nový balíček a vše potřebné do zadaného souboru, tak metoda **addNewClass** provede to stejné pro třídu. Tato metoda provádí jen dvě další metody - již zmíněnou **addDiagramElement**, která vloží umístění vytvářené třídy do zvoleného diagramu, a metodu **addClass,** jež do jmenného prostoru zadaného balíčku vloží novou třídu. Do každého balíčku je vložena jedna třída, stejného jména jako je balíček, ve kterém se nachází. Tato třída reprezentuje proces stejného jména, jako má balíček, ve kterém se nachází. Následuje ukázka třídy.

```
<UML:Class name="Proces1" xmi.id="EAID_1_Class" namespace="EAPK_1" >
       <UML:ModelElement.taggedValue>
              …
       </UML:ModelElement.taggedValue>
</UML:Class>
```

Další metoda **addNewDocument** vloží nový dokument do zadaného souboru stejným způsobem jako metoda **addNewClass**. Jediná odlišnost v zápisu mezi třídou a dokumentem spočívá v tom, že dokument navíc obsahuje sekci stereotype, ve které se definuje, že se jedná o dokument. Následuje názorná ukázka:

```
<UML:ModelElement.stereotype>
       <UML:Stereotype name="document"/>
</UML:ModelElement.stereotype>
<UML:ModelElement.taggedValue>
```

Výše popsané metody umožňují vytvářet balíčky a do nich vkládat další balíčky, třídy a dokumenty a ty graficky rozmisťovat na ploše. Další skupina metod umožňuje graficky zobrazovat vztahy mezi těmito elementy. Tyto vztahy se zapisují pomocí závislostí (dependency). Jako příklad je uvedena metoda **addDependencyClassPack**. Této metodě se předá právě vytvářený soubor, informace o zdrojové třídě, cílovém balíčku a balíčku, ve kterém se tyto elementy nacházejí, a vytvoří mezi těmito dvěma elementy orientovanou hranu (šipku). To provádí vložením závislosti do jmenného prostoru balíčku, ve kterém se má element šipky zobrazovat. Obdobné metody existují i pro jiné kombinace elementů, mezi nimiž se má vytvořit šipka.

Následuje zkrácená ukázka zápisu závislosti, která reprezentuje orientovanou hranu mezi třídou a balíčkem.

```
<UML:Dependency client="EAID_1_Class" supplier="EAPK_13" xmi.id="EAID_1_13" >
      <UML:ModelElement.taggedValue>
                  …
            <UML:TaggedValue tag="ea_sourceName" value="Proces1"/>
            <UML:TaggedValue tag="ea_targetName" value="Process3"/>
            <UML:TaggedValue tag="ea_sourceType" value="Class"/>
            <UML:TaggedValue tag="ea_targetType" value="Package"/>
                  …
      </UML:ModelElement.taggedValue>
</UML:Dependency>
```

5.6 Rozmístění elementů na grafické ploše

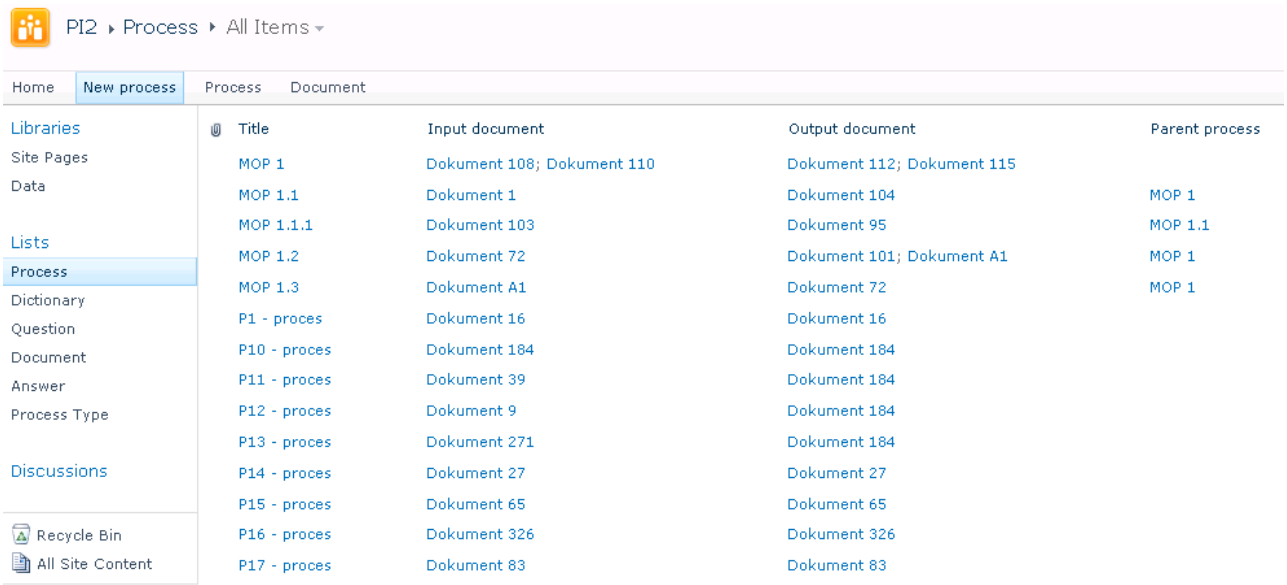
Zatím bylo popsáno, jak vytvářet jednotlivé elementy a vazby mezi nimi. Výše popsaným způsobem jsme schopni zaznamenat všechny potřebné procesy a vztahy s dalšími elementy nacházejícími se v nástroji Process Inspector. Po importu do nástroje Enterprise Architect bychom však na grafické ploše viděli pouze jeden element. To je zapříčiněné tím, že pokud nástroji Enterprise Architect nezadáme přesné souřadnice jednotlivých elementů (s výjimkou šipek), zobrazí je všechny přes sebe. Uživatel si sice tyto elementy může libovolně rozprostřít na dané ploše, tento postup je však dosti nepraktický.

Až na element závislosti jsme u každého přidávaného elementu používali metodu **addDiagramElement**, která vloží umístění vytvářeného elementu do zvoleného diagramu. Této metodě je možné zadat přesné souřadnice, kde se má tento element zobrazit. K vytváření těchto souřadnic slouží metoda **getGeometry** ve třídě **XmlFile.** Této metodě se zadá číslo značící, o kolikátý prvek na grafické ploše se jedná, a ta vrátí vhodné souřadnice umístění toho prvku. Způsob rozmístění jednotlivých prvků na ploše byl zvolen po konzultaci s pracovníky společnosti Allium, kteří s tímto modelem budou pracovat, a po přihlédnutí k očekávanému počtu prvků na ploše.

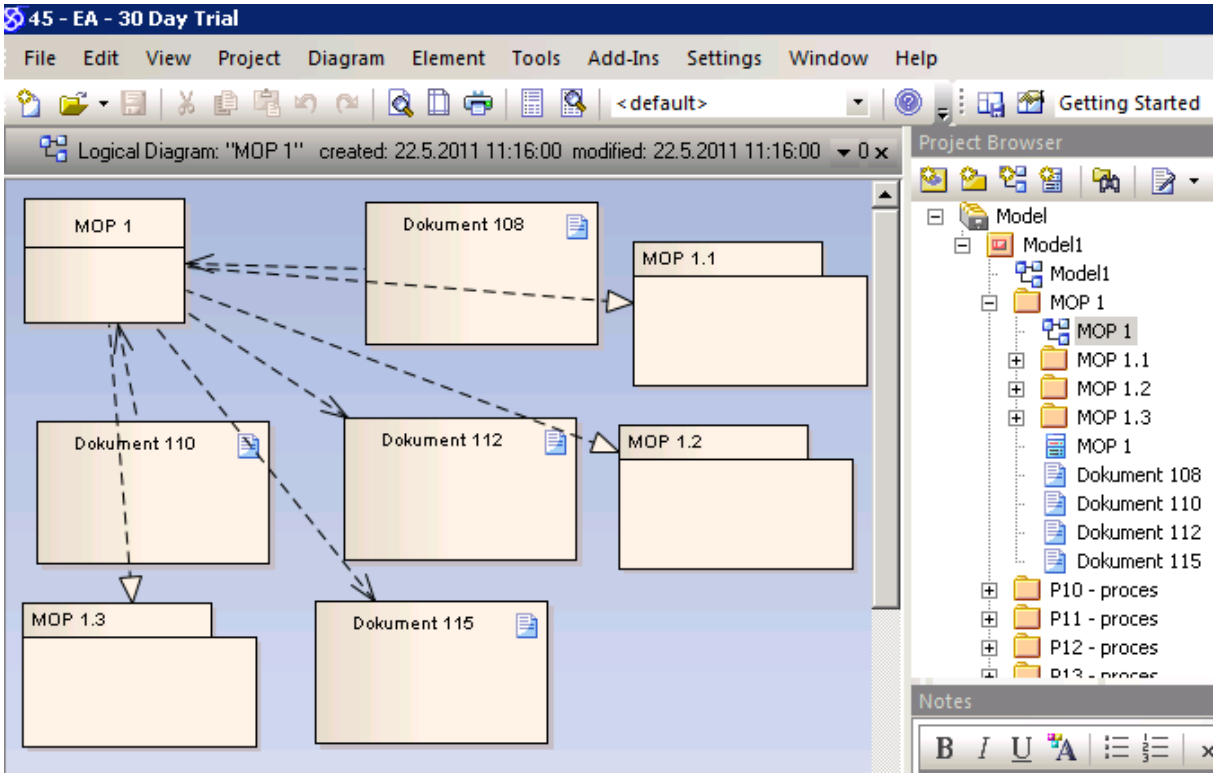
Prvním prvkem na ploše je vždy třída reprezentující právě vytvořený nový balíček. Nabízí se zde možnost umístit tuto třídu doprostřed plochy. V nástroji Enterprise Architect je však grafická plocha k zobrazování prvků dosti velká a na menších monitorech se stává, že po importu dat uživatel vidí stále prázdnou plochu, což je zapříčiněno tím, že prvky se nacházejí mimo právě zobrazovanou část plochy. Při zapnutí nástroje Enterprise Architect se jako první vždy zobrazuje levá horní část plochy. Z tohoto důvodu bylo zvoleno zobrazení, ve kterém se také první prvek umísťuje do levého horního rohu. Další souřadnice jsou vypočítávány tak, aby postupně plnily řady napravo a dolů od prvního prvku a vytvářely čtverec. Pokud chceme mít nejblíže u hlavní třídy prvky vstupních dokumentů a následně prvky výstupních dokumentů a až nakonec prvky balíčků, stačí zvolit správné pořadí, ve kterém se prvky na plochu umisťují. Vhodnost takto zvoleného rozmístění byla ověřena testováním.

5.7 Import do nástroje Enterprise Architect

Import do nástroje Enterprise Architect je velmi snadný. V tomto nástroji založíme nový projekt a z nabízených možností zvolíme, že chceme vytvořit třídní model (Class model). Vytvoří se nám vzorový projekt, který následně odstraníme, a zůstane pouze prázdný balíček Model. Kliknutím pravého tlačítka myši na tento balíček získáme možnost importovat model ze souboru XMI – zvolíme soubor vytvořený nástrojem Process Inspector a zahájíme importování. Obrázek 13 zobrazuje procesy zaznamenané v nástroji Process Inspector a obrázek 14 zobrazuje tyto procesy v nástroji Enterprise Architect po jejich převodu.



Obrázek 13: Zaznamenaná struktura procesů v nástroji Process Inspector



Obrázek 14: Zobrazení procesů v nástroji Enterprise Architect

6 Testování

V průběhu vývoje aplikace byla její funkčnost pravidelně testována na vhodně zvoleném příkladu přibližně třiceti procesů a pěti dokumentů. Správnost vytvořeného soboru byla vždy ověřena následným importem do nástroje Enterprise Architect. Tento nástroj provádí před každým zahájením importování kontrolu, zda má soubor strukturu odpovídající definovanému DTD dokumentu. Správnost vytvořeného modelu byla následně ověřena i vizuálně.

6.1 Zavedení na server

V závěrečné části vývoje poskytla společnost Allium prostřednictvím služby vzdálená plocha přístup k jednomu ze svých serverů, kde byla vyvíjená aplikace zprovozněna a ověřeno její správné fungování. Tento server má dvojjádrový procesor (v minimálních požadavcích ke správnému chodu platformy SharePoint se uvádí potřeba čtyřjádrového procesoru) a 6 GB RAM paměti (minimální uváděný požadavek je 8 GB RAM). Tento server byl v plném provozu, byl využíván současně několika dalšími uživateli, a probíhala na něm častá instalace a aktualizace různých programů. V závislosti na momentálním zatížení serveru se stávalo, že i některé jednoduché operace v nástroji Process Inspector (jako například přechod mezi záložkami) trvaly velmi dlouhou dobu. Celkově bylo znát, že hardware neodpovídá minimálním požadavkům k provozu platformy SharePoint. Na tomto serveru byl nainstalován nástroj Process Inspector obsahující vzorový projekt s asi 300 procesy a 180 dokumenty.

Ve společnosti Allium mají dva největší projekty asi 300 a 500 procesů, většina ostatních projektů je svým rozsahem mnohem menší. Vložení aplikace do nástroje Process Inspector proběhlo bez komplikací. Hned první pokus o export všech procesů však byl přerušen asi po 15 minutách. Při dalším pokusu se tvorba exportu omezila jen na 5 procesů a všechny jejich podprocesy a export byl úspěšně dokončen v čase asi 5 minut. Uvádět délku trvání v sekundách by bylo dosti zavádějící. Doba provedení dvou stejných operací, kdy jedna je provedena bezprostředně po dokončení druhé, se značně liší v závislosti na momentálním zatížení serveru. Z tohoto důvodu zde budou uváděny orientační časy v minutách.

6.2 Optimalizace rychlosti

Prvotní aplikace nebyla schopna provést export 300 procesů a 180 dokumentů v rozumném čase, bylo ji tedy nutné upravit. Byla vytvořena nová třída sloužící k vytváření exportů všech procesů nalézajících se v nástroji Process Inspector (což byl upřesněný požadavek firmy Allium). Tato třída, na rozdíl od původního návrhu, začíná tvorbu výsledného souboru tím, že si z databáze načte všechny informace o procesech a dokumentech do paměti a dále již k databázi nepřistupuje. Nad těmito daty se provádí tvorba souboru podobně, jako v původním návrhu. Při tvorbě souboru jsou však některé

činnosti spojeny v jednu. Kupříkladu pokaždé, když původní návrh vytvářel nový balíček, vždy do něj vložil třídu stejného jména. Tato činnost byla vytvářena ve dvou krocích. V novém návrhu se balíček vytváří automaticky i s novou třídou. Podobných spojení několika činností v jednu je v novém návrhu více. Dalšího zrychlení bylo docíleno odstraněním některých nadbytečných značkovacích hodnot. Tímto opatřením se počet řádků vygenerovaného souboru snížil asi o třetinu. Soubor zaznamenávající strukturu 430 procesů a 350 dokumentů má však stále přibližně 24 000 řádků zdrojového kódu. Po všech uvedených změnách byla doba k vytvoření potřebného souboru zkrácena asi na 4 minuty pro 300 procesů a 180 dokumentů a na 7 minut pro 430 procesů a 350 dokumentů. Tyto časy byly konzultovány s odpovědným pracovníkem ve společnosti Allium a schváleny jako vyhovující.

6.3 Testy a výsledky testování

6.3.1 Import do nástroje Enterprise Architect

Správnost všech vytvořených souborů byla ověřena pomocí nástroje Enterprise Architect. Pomocí parseru byla provedena kontrola, zda je soubor správně strukturovaný a zda odpovídá danému DTD. Následně byl proveden import do tohoto nástroje. Import souboru zachycujícího strukturu 430 procesů a 350 dokumentů trval 58 sekund, zatímco import souboru zachycujícího strukturu 30 procesů a 10 dokumentů trval pouze 5 sekund.

6.3.2 Uživatelské testování

Na testování se podíleli dva nezávislí uživatelé, kteří měli za úkol převést procesy z nástroje Process Inspector do nástroje Enterprise Architect. Tito uživatelé neměli žádné předešlé zkušenosti s prací s těmito nástroji, měli však k dispozici podrobný uživatelský manuál, který je součástí této práce. Oba uživatelé zvládli tento úkol bez problémů. Na základě jejich připomínek byl na webovou část přidán jednoduchý popisek s doplňujícími informacemi.

Následující tabulka uvádí výsledky provedených testů nad různými daty.

Počet provedených testů	Počet zachycených procesů	Počet vstupních a výstupních dokumentů	Doba potřebná k exportu z nástroje Process Inspector	Doba potřebná k importu do nástroje Enterprise Architect	Úspěšnost kontroly souboru oproti DTD	Úspěšnost vizuální kontroly grafického modelu procesů
30	30	10	1 min	5 s	100 %	100 %
10	300	180	4 min	33 s	100 %	100 %
5	430	350	7 min	58 s	100 %	100 %

Tabulka 7: Výsledky provedených testů

7 Závěr

7.1 Vlastní zkušenosti s vývojem na platformě Microsoft SharePoint

První komplikace nastala při instalaci SharePoint Server 2010. Po nainstalování z neobjasněného důvodu nebylo možné se připojit k službě SharePoint. Tento problém jsem konzultoval s dalším diplomantem pracujícím s platformou SharePoint, ale problém se nepovedlo odhalit. Řešením byla až kompletní reinstalace SharePoint Server 2010. Problém s instalací SharePoint měl i další letošní diplomant, který mě žádal o radu a mé zkušenosti s touto platformu. Obecně lze říci, že pokud se při vývoji na platformě SharePoint objeví nějaká neobvyklá chyba, může být její vyřešení velmi náročné. Z důvodu novosti této platformy není k dispozici takové množství informací o této platformě, jako je tomu u jiných již zavedených nástrojů. Klíčovým a téměř jediným zdrojem pro získávání informací a řešení problémů s platformou SharePoint je oficiální portál pro vývojáře a softwarové inženýry společnosti Microsoft (MSDN).

Microsoft SharePoint 2010 je komplexní řešení, které zahrnuje velkou řadu různých technologií, což vede k tomu, že vývoj aplikací na této platformě vyžaduje výrazně vyšší nároky na vývojáře např. oproti technologii PHP. Vývojář potřebuje mít ke správnému zvládnutí této platformy znalosti o principech programování v .NET, tvorbě webových aplikací pomocí této technologie a obecně o objektově orientovaném programování.

Většina problémů s touto platformou byla způsobena jevem nazývaným Blackbox, který značí to, že používáme objekty, u nichž nevíme, jak fungují, což však většinou nepotřebujeme vědět. Stačí nám znát, jaký vstup tomuto objektu máme předat, a on nám vydá očekávaný výstup. Když však nastane nějaká nečekaná chyba, bývá velmi obtížné ji odstranit.

Obrázek 15: Schéma blackbox

Pokud se však vývojář naučí pracovat s touto platformou a nabude potřebné zkušenosti, získá komplexní a velmi efektivní nástroj, s nímž bude schopen velmi rychle a snadno vytvářet opravdu komplexní informační systém dostupný z jednoho místa – jak napovídá název platformy.

Minimální uváděné požadavky k instalaci a provozu SharePoint Server 2010 jsou 64-bitový čtyřjádrový procesor a 8 GB RAM. Při implementaci jsem použil svůj osobní počítač s odpovídající pamětí, avšak jen dvoujádrovým procesorem. SharePoint byl dosti pomalý, stále však použitelný. Hardwarová náročnost jen potvrzuje, že SharePoint Server 2010 je určen pro výkonné servery.

Data v platformě SharePoint

Při práci s daty jsem se obával velmi omezeného propojení listů, pouze pomocí náhledů na sloupce jiného listu. Tato obava se nakonec nepotvrdila a propojení pomocí náhledů bylo pro mou práci dostačující. Naopak za poměrně nepohodlné považuji pohledy nad daty pomocí jazyka CAML. Tento jazyk, ač významem podobný jazyku SQL, nenabízí zdaleka takové možnosti jako SQL (který je navíc standardizován a častěji používán).

7.2 Zhodnocení implementace

Cílem implementace bylo vytvořit nástroj na import dat z nástroje Process Inspector, jejich zpracování a export jejich struktury pro nástroj Enterprise Architect.

Způsob, jakým jsem importoval data z nástroje Process Inspector, je podrobně popsán v kapitole Implementace. V této kapitole také podrobně popisuji, jak ze získaných dat vytvářet XML soubor. Tento soubor, určený k importu do nástroje Enterprise Architect, reprezentuje strukturu jednotlivých procesů, jejich podprocesů, vstupních a výstupních dokumentů, a dále pak rozmístění prvků reprezentující tyto procesy a dokumenty na ploše, a grafické znázornění vztahů mezi těmito prvky. Správnost mé implementace potvrzuje kapitola Testování. V této kapitole jsem ověřil správnou funkci zvoleného řešení a jeho rychlost.

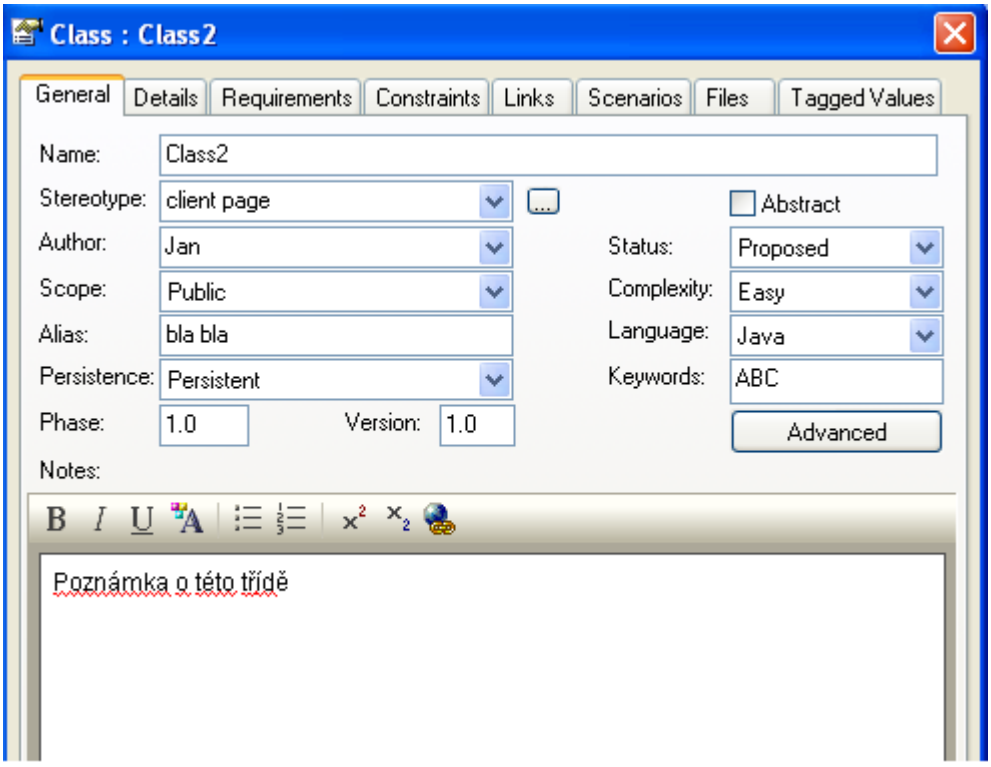
Celkově implementaci hodnotím pozitivně. Vytvořený nástroj je schopen vytvořit soubor, který po importování do nástroje Enterprise Architect velmi přehledně zobrazí všechny potřebné informace z nástroje Process Inspector a umožní jejich další efektivní zpracovávání. Struktura implementovaného nástroje je také přehledná a snadno udržovatelná či modifikovatelná.

7.3 Další možnosti rozvoje vytvořeného nástroje

Prostor pro další rozšíření vytvořeného nástroje považuji za dosti omezený. V následujících podkapitolách uvádím dva možné směry, kterými by bylo možné se ubírat při rozšiřování vytvořeného nástroje.

7.3.1 Přidání dalších hodnot

Jednou z možností jsou již zmíněné nepovinné hodnoty taggedValue. Tyto hodnoty se používají k přidání extra informací pro nástroj Enterprise Architect, i když tyto hodnoty nepoužijeme. Po následném importu do nástroje Enterprise Architect máme možnost je upravovat, měnit a přidávat. Následující obrázek ukazuje možnosti úpravy těchto hodnot v nástroji Enterprise Architect.



Obrázek 16: Možnosti nastavení tříd v nástroji Enterprise Architect

Vytvořený nástroj by mohl být rozšířen o další webovou část (Web Part), v níž by uživatel dostal možnost úpravy těchto hodnot, podobně jako na obrázku 16. Možnost tohoto rozšíření byla konzultována s budoucími uživateli vytvořeného nástroje ve společnosti Allium, a byla zamítnuta. Důvodem k zamítnutí tohoto rozšíření je jeho nadbytečnost, tyto hodnoty lze upravovat v nástroji Enterprise Architect. Další důvod je, že většina těchto hodnot se nepoužívá a ty, které jsou používány,

již vytvořená aplikace obsahuje. Dále by přidání dalších, pro uživatele většinou nepotřebných hodnot, vedlo k zpomalení a znepřehlednění struktury vytvořeného souboru a vytvořeného nástroje.

7.3.2 Export do dalších nástrojů pro práci s UML diagramy

Další možností rozvoje vytvořeného nástroje je rozšířit tento nástroj o export dat i do jiného nástroje, než je Enterprise Architect. Seznam dalších takovýchto nástrojů jsem čerpal z článku o UML nástrojích [17]. Tento článek uvádí dalších devět nástrojů. Celá má práce byla úzce spjata s produkty společnosti Microsoft, přirozeně jsem tedy vynechal ty nástroje, které nepodporují operační systém Windows.

Částečně jsem se seznámil s volně šiřitelnými nástroji, jedná se o tyto nástroje:
Violet UML Editor Version 0.21.1. Tento nástroj nepodporuje export dat formátu XMI, logicky tedy není možné rozšířit vytvořený nástroj o import do tohoto nástroje pomocí jazyka XMI.
ArgoUMLVersion 0.32.2. Tento nástroj podporuje import dat ve formátu XMI. Jím vytvářený XMI soubor má mnohem jednodušší strukturu a výrazně méně různých doplňujících hodnot. Také má odlišný systém volby ID pro jednotlivé elementy, což však lze očekávat u všech dalších nástrojů pro práci s UML diagramy. Upravit vytvořený nástroj tak, aby byl schopen exportovat data i do toho nástroje, by nemělo být moc náročné.

Následuje jeden z komerčních nástrojů:
IBM Rational Rose, tento velmi rozšířený nástroj podporuje import z XMI souboru. Je však třeba nainstalovat „XMI add-in for Rose“ doplněk. Struktura XMI souboru vytvářeného tímto nástrojem je dosti komplikovaná a úprava vytvořeného nástroje o import do toho nástroje by byla o poznání těžší než do nástroje ArgoUMLVersion.

Rozšíření vytvořeného nástroje o import do dalších nástrojů k práci s UML modely je možný, pokud tento nástroj podporuje import z XMI souboru. Obecně platí, že složitost rozšíření odpovídá komplexnosti nástroje, pro který je export vytvářen. Všech 9 nástrojů, které jsem vzal v potaz k rozšíření importu, nepodporovalo programovací jazyk C#. Podpora tohoto jazyka je však jedním z požadavků. V současné době postačují funkce, které nabízí nástroj Enterprise Architect, a tak není nutné vytvářet import i do jiných nástrojů.

7.4 Celkové zhodnocení

Při realizaci této práce jsem získal cenné zkušenosti, a to jak teoretické, týkající se problematiky managementu procesů, tak převážně praktické. V průběhu práce jsem se seznámil s řadou různých nástrojů, technologií a postupů. Jsem přesvědčen, že tyto zkušenosti budu moci uplatnit i v dalších projektech, kterých se budu účastnit.

Vytvořený nástroj splnil všechna očekávání zadavatelů a doufám, že jim usnadní jejich další práci při návrhu nových aplikací.

Literatura

[1] Becker, J; Kugler, M; Rosemann, M. Process Management: A Guide for the Design of Business Processes. Germany : Springer-Verlag, 2003. 337 s. ISBN 3-540-43499-2.

[2] Bradley, N. XML: kompletní průvodce. Praha : Grada Publishing s.r.o., 2000. 537 s. ISBN 80-7169-949-7.

[3] Hammer, M; Champy , J. Reengineering – manifest revoluce v podnikání, radikální proměna firmy. Praha : Management Press, 2000. ISBN 80-7261-0287.

[4] Harold, E. R; Means, W. S. XML v kostce. Praha : Computer Press, 2002. 440 s. ISBN 80-7226-712-4.

[5] Kosek, Jiří. XML pro každého. Praha : Grada Publishing, 2000. 164 s. ISBN 80-7169-860-14.

[6] Řepa, Václav. Podnikové procesy: Procesní řízení a modelování. 2., aktualizované a rozšířené vydání. Praha : Grada Publishing, 2007. 281 s. ISBN 978-80-247-2252-8.

[7] Stalk, G; Evans, P; Shulman, L.E. Competing on Capabilities. Harvard : Business Review, 1992. 70 s.

[8] Šmída, Filip. Zavádění a rozvoj procesního řízení ve firmě. Praha : Grada Publishing, 2007. 293 s. ISBN 978-80-247-1679-4.

[9] Business Process Modeling Notation *1.2* [online]. [s.l.] : [s.n.], 2009 [cit. 2011-05-22]. Dostupné z WWW: <http://www.visual-paradigm.com/support/documents/bpvabpmnspec.jsp>.

[10] Lévay, Radek. Ikvalita.cz [online]. 28.4.2007 [cit. 2011-05-04]. Management procesů virtuální přednáška. Dostupné z WWW: <ikvalita.cz/download/virtualni_prednaska_01.pps>.

[11] Microsoft. Microsoft Office [online]. 2011 [cit. 2011-05-04]. Úvod do aplikace SharePoint Designer 2010. Dostupné z WWW: <office.microsoft.com/cs-cz/sharepoint-designer-help/uvod-do-aplikace-sharepoint-designer-2010-HA101782482.aspx>.

[12] Microsoft SharePoint 2010 [online]. 2010 [cit. 2011-01-02]. Sharepoint.microsoft.com. Dostupné z WWW: <sharepoint.microsoft.com/cs-cz/Pages/default.aspx>.

[13] Microsoft SharePoint 2010. Zive.cz [online]. 2010, 1, [cit. 2011-01-04]. Dostupné z WWW: <zive.cz/clanky/microsoft-sharepoint-2010/sc-3-a-152097/default.aspx>.

[14] Rozsypálek, Michal. Rozvoj nástroje na podporu vývoje interaktivních aplikací. Brno, 2008. 51 s. Diplomová práce. VUT Brno. Dostupné z WWW: <fit.vutbr.cz/study/DP/rpfile.php?id=5458>.

[15] Stein, Rene. Návrh aplikace v jazyce UML. Interval.cz [online]. 2004. 1, [cit. 2010-12-12]. Dostupné z WWW: <interval.cz/clanky/navrh-aplikaci-v-jazyce-uml-zakladni-pojmy-a-pravidla>.

[16] Školoud, Otakar. IS pro oddělení technické podpory [online]. Praha, 2007. 125 s. Diplomová práce. ČVUT FEL. Dostupné z WWW: <dip.felk.cvut.cz/browse/pdfcache/skoloo1_2007dipl.pdf>.

[17] Tišnovský, Pavel. Nástroje pro tvorbu UML diagramů. Root.cz [online]. 2005, 1, [cit. 2010-12-11]. Dostupné z WWW: <root.cz/clanky/nastroje-pro-tvorbu-uml-diagramu>.

[18] Verner, Jan. Metriky procesů vývoje softwaru. Brno, 2009. 58 s. Diplomová práce. VUT Brno FIT. Dostupné z WWW: <http://www.fit.vutbr.cz/study/DP/rpfile.php?id=8948>.

[19] Wrightson, Ann M. XML study notes [online]. 1999 [cit. 2011-01-08]. XML.coverpages.org. Dostupné z WWW: <xml.coverpages.org/wrightson-xminotes.html>.

[20] Enterprise Architect [online]. 2010 [cit. 2010-12-12]. Sparxsystems.com. Dostupné z WWW: <sparxsystems.com>.

[21] Informační web o nástroji Enterprise Architect [online]. 2010 [cit. 2011-01-02]. Enterprisearchitect.cz. Dostupné z WWW: <enterprisearchitect.cz>.

[22] Microsoft. Oficiální portál pro vývojáře a softwarové inženýry [online]. 2011 [cit. 2011-05-04]. Dostupné z WWW: <msdn.microsoft.com>.

Seznam příloh

Příloha A: Obsah přiloženého DVD

- Adresář Webpart se zdrojovými kódy potřebnými k vložení webové části do nástroje Process Inspector
- Uživatelská příručka ve formátu PDF
- Programátorská příručka ve formátu PDF
- Vzorový soubor ve formátu XML vytvořený v nástroji Process Inspector a určený k následnému importu do nástroje Enterprise Architect
- Text diplomové práce ve formátu PDF
- Dokument aplikace MS Office s textem diplomové práce