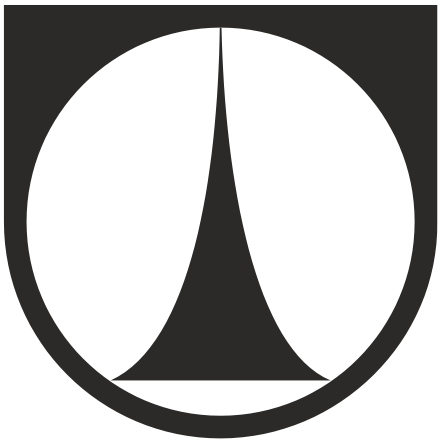
TECHNICKÁ UNIVERZITA V LIBERCI
Ekonomická fakulta



BAKALÁŘSKÁ PRÁCE

2013 **Ondřej Soukup**

TECHNICKÁ UNIVERZITA V LIBERCI
Ekonomická fakulta

Studijní program: **B 6209 Systémové inženýrství a informatika**
Studijní obor: **Manažerská informatika**

Využití nástroje Enterprise Architect pro reengineering informačních systému

The Use of Enterprise Architect Tool for Reengineering of Information Systems

BP – EF – KIN 2013 - 23
Ondřej Soukup

Vedoucí práce: Ing. Zádová Vladimíra, Ph.D., katedra informatiky
Konzultant: Ing. Novotný Rostislav, OR-CZ spol. s r.o.

Počet stran: Počet příloh:

Datum odevzdání: 10. května 2013

Prohlášení

Byl jsem seznámen s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé diplomové práce pro vnitřní potřebu TUL.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Bakalářskou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím diplomové práce a konzultantem.

V Liberci dne 10. května 2013 Ondřej Soukup

Poděkování

Rád bych poděkoval vedoucímu závěrečné práce Ing. Vladimíře Zádové, Ph.D. za trpělivost, připomínky a rady při vypracovávání bakalářské práce. Dále bych rád poděkoval společnosti OR-CZ, s.r.o. a jejímu celému kolektivu za přijetí a za pomoc při získávání nových zkušeností v oboru tvorby informačního systému a reengineeringu.

Anotace

Bakalářská práce se zabývá reengineeringem informačního systému, který byl prováděn pomocí case nástroje Enterprise Architect. V první kapitole je zde seznámení s case nástroji, konkrétně poté s nástrojem Enterprise Architect. V této kapitole je popisována historie case nástrojů, jejich stručný popis a poté nástroj Enterprise Architect, jeho charakteristika společně s jehofunkcemi. Dále je v práci popisován reengineering, jeho definice, osoby, které ho provádějí, a nejznámější metodiky používané při reengineeringu.

V praktické části je popisován reengineering procesu vývoje informačního systému ve společnosti OR-CZ pomocí nástroje Enterprise Architect od požadavků až k návrhu řešení. Jsou zde popisovány jednotlivé kroky, dle toho jak se při reengineeringu postupovalo. Na závěr je poté vše shrnuto a zhodnoceno.

Klíčová slova

case nástroj,Enterprise Architect, informační systém, proces, reengineering

Annotation

This bachelor thesis deals with reengineering of information system, which is implemented using case tool Enterprise Architect. There are introduced case tools, in particular following the tool of Enterprise Architect. The study also described reengineering, its definition, the persons who do reengineering, and the most popular methods used in reengineering.

In the practical part of the bachelor thesis is described reengineering the information system development process in company OR-CZ by using Enterprise Architect. Reengineering is described from requirements to design solutions. There are described all steps, depending on how reengineering progressed.

Key Words

case tool,Enterprise Architect, information system, process, reengineering

Obsah

Seznam zkratek

BPMN (Business Process Model and Notation) - standard pro modelování podnikových procesů

CASE Computer-Aided Software Engineering

EA Enterprise Architect

ERP (Enterprise Resource Planning) - druh informačního systému

IS Informační systém

MDA (Model Driven Architecture) - modelem řízená architektura

PP Podnikový proces

RTF **(**Rich Text Format) – formát ukládání v textovém editoru

TUL Technická univerzita v Liberci

Seznam tabulek

Seznam obrázků

Úvod

Bakalářská práce se zabývá reengineeringem informačního systému. Důvod proč jsem si toto téma zvolil, je jelikož mě zaujala možnost tvorby informačního systému (IS) na mé praxi a toto téma s tím úzce souviselo – byla zde potřeba provést reengineering procesu vývoje informačního systému.

Cílem práce je reengineering IS pomocí modelovacího nástroje Enterprise Architect (dále jen EA) ve firmě OR-CZ. V této práci se věnuji dané problematicce a řešením daných problémů firmě OR-CZ. Společnost OR-CZ se rozhodla pro radikální změnu ve vývoji jejich IS a s touto změnou vznikla potřeba reengineeringu procesu vývoje IS. V práci budu popisovat case nástroje všeobecně, protože jsem skoro vše dělal díky nim a konkrétní case nástroj, ve kterém jsem pracoval, jeho možnosti, funkce, atd.

Dále zde budu popisovat reengineering. Co se pod tímto pojmem skrývá, k čemu slouží, proč se využívá a jaký má přínos. Dále se poté věnovat reengineeringu konkrétního problému. Na daném procesu vývoje IS popíšu jednotlivé kroky procesu reengineeringu, problémy na které jsem narazil a výsledný návrh řešení – již „zreengineeringovaný“ IS. Tato práce by měla být pro OR-CZ přínosná v tom, že díky reengineeringu procesu vývoje IS budou analytici při vývoji využívat case nástroj EA a díky němu budou mnohem efektivnější.

1. Možnosti nástroje Enterprise Architect

V této kapitole se nejpve budu věnovat case nástrojům. V první části 1.1 bude popsána historie case nástrojů, jejich komponenty a klasifikace. V druhé části 1.2 se budu věnovat programu Enterprise Architect, konkrétně jeho obecné charakteristice a popisu funkcí, které nabízí.

1.1 Case nástroje obecně

1.1.1 Historie Case nástrojů

Case nástroje, angl. Computer-Aided Software Engineering tools, je sada nástrojů a metod softwarového inženýrství na počítačovou podporu a vývoj v rámci procesu vývoje softwaru. Viz. Také definice case nástrojů podle SEI (Software engineering institute): „A case tool is a computer-based product aimed at supporting one or more software engineering activities within a software development process.“

Se vznikem softwarového inženýrství, které začalo vznikat koncem 60. let, když začala tzv. softwarová krize, kdy výkon hardwaru předčil vývoj softwaru, začaly vznikat i nástroje na počítačovou podporu vývoje systému (case nástroje). Velmi důležitá byla konference NATO v r. 1998 ve Spolkové republice Německo, která popularizovala softwarové inženýrství. Do té doby se počítače využívaly pro vědeckotechnické výpočty, kde záleželo spíše na preciznosti řešení, než na efektivitě vývoje. case nástroje se začaly využívat ve vývoji softwaru koncem 70. a začátkem 80. let, kdy se především zaměřovaly na rané fáze vývoje softwaru, tj. na analýzu, návrh a požadavky v procesu vývoje softwaru. Case nástroje zaváděly textové editory, které zlepšily sledování dokumentace, která byla tvořena po celou dobu životního cyklu vývoje softwaru. Tyto nástroje byly použity pro zadání textu, manipulaci, zformátování pomocí vizuálního rozhraní, stejně jako sledování dokumentace dané verze. Tyto case nástroje výrazně posílily práci softwarových inženýrů a zvýšily jejich účinnost. Byla také vyvinuta počítačová podpora diagramů pro potřebu softwarových inženýrů a programátorů. Umožňovalo to rychle zpracovat a snadno upravit

diagramy a návrhy. Tyto diagramy byly používány v analýze a navrhování vývoje softwaru. Zlepšení case nástrojů vedlo k vývoji datových slovníků, které sloužily k uložení informací o všech datových typech souvisejících s procesem vývoje. Datový tok a strukturální vzory vytvořené grafickými nástroji dále podnítilo rozvoj datových slovníků. Nakonec grafické nástroje integrované s databází dat tvořily silný designový a vývojářský nástroj, které měl kompletní dokumentaci vývojového cyklu. case nástroje také prováděly testy softwaru v nepřetržitém procesu ověřování a schvalování. Toto významně snižovalo úkoly, které by měly být provedeny v průběhu testovací fáze softwaru. Spojení case nástrojů s grafickým uživatelským rozhraním vedlo k velkému rozmachu koncem 80. let. To byl prvopočátek case nadšenosti. Dosud byly dostupné pouze samostatné malé nástroje, nyní bylo možné vše integrovat na univerzální pracovní ploše. Case nástroje se postupem času vyvinuly z původního zaměření na podporu vývoje informačního systému do obecnější podoby nástrojů na modelování systémů.

1.1.2 Komponenty case nástrojů

Z hlediska funkcí, vlastností case nástrojů a požadavků na ně vyplývá také z jakých komponent se tyto systémy skládají. Mezi důležité funkce a vlastnosti case patří:

Konzistentní grafické ovládací prostředí (podle zásad tvorby GUI) – jednotný vzhled obrazovek, popisků, tlačítek, jednotné ovládání, použití symbolických ikon apod.

Centrální databáze (repository) pro uchování informací o všech objektech IS (tímto způsobem se zaručí, že informace je použitelná v libovolném dalším kroku projektování)

Prostředky verifikace konzistentnosti dat a podpora normalizace dat

Textový editor pro popis jednotlivých objektů – pro účely technické a uživatelské dokumentace systému, možnost jejího přímého generování ze systému

Možnost rychlého návrhu uživatelských obrazovek včetně simulace vstupů a výstupů (je vyžadováno pro prototyping)

Generátor zdrojových programů (pro případy častého znovupoužití daného kódu)

Export / import dat – pro práci s modely a dokumentací, které byly vytvořeny v jiných programech nebo jsou v jiných programech dále využívány a zpracovávány.

1.1.3 Klasifikace case nástrojů

Nejčastěji se používá rozdělení z hlediska životního cyklu projektu, tzn., v jaké fázi vývoje softwaru se case nástroj využije. Toto dělení je následující:

Pre CASE - Podpora činnosti předcházející vývoji IS

Upper CASE - Podpora analýzy.

Middle CASE - Podpora návrhu.

Lower CASE - Podpora implementace

Post CASE - Podpora fázi uvedení IS do provozu, provoz a údržbu

Integrovaný - také známé jako I-CASE podpory analýzy, návrhu a implementace.

Další rozdělení mohou být podle interaktivity, stupně integrace, toho, zda jsou využívány během celého životního cyklu software, ale tím se zde nebudu zabývat.

Díky case nástrojům dosahujeme nižších nákladů při vývoji softwaru, snížení doby potřebné na vývoj softwaru, snížení generace vad, snadnější identifikaci závad během vývoje, vyšších úspor během údržby, větší standardizace softwarových systémů a proto i zvýšení opětovných možnosti a snížení intenzity potřebné na rozvoj.

Důležitých aspektů pro výběr správného case nástroje je mnoho. Prvním aspektem, který bychom měli vědět je, zda nástroj podporuje strukturovanou, nebo objektovou metodiku. Další aspekty jsou: podpora integrace s ostatními nástroji potřebnými pro projekt. Jakou notaci nástroj podporuje. Jestli je nástroj otevřený a přístupný modifikacím. Podporuje

verzování a sdílení komponent. Podporuje kontrolu konzistence, úplnosti a dodržování metodiky. Zda nástroj generuje datový model a kód aplikační logiky. A zdali nástroj disponuje možností round trip engineeringu a reverse engineeringu.

1.2 Možnosti nástroje Enterprise Architect

1.2.1 Obecná charakteristika Enterprise Architect

Nástroj, který jsem použil pro vypracování své praktické části, a budu se mu věnovat více, se jmenuje Enterprise Architect EA. Tento case nástroj od australské firmy Sparx Systems je komplexním modelovacím nástrojem pro analýzu a návrh informačních systémů pomoci jazyka UML. Pokrývá všechny aspekty softwarového vývoje od sběru požadavků, analýzy, modelovaní, návrh, testovaní, řízení změn, údržbu až po implementaci, přičemž v rámci celého tohoto procesu je vždy zajištěna zpětná dohledatelnost. Tedy jeho zařazení z hlediska klasifikace je I-CASE. EA je mnohouživatelským, vizuálním nástrojem s velkým rozsahem funkcí, jehož současná verze podporuje všechny diagramy UML 2.4.1.

EA, který je dlouhodobě oceňován za podporu nejnovějších verzí UML, je týmově založený modelovací nástroj, podporující celý životní cyklus softwaru. Zaujme především vysoce výkonnými zobrazovacími nástroji pro plánování business procesů, podnikové architektury, managementu požadavků, testování apod.

EA usnadní fázi vývoje software, jak již bylo řečeno. Hlavním úkolem je modelování, přičemž nástroj podporuje diagramy UML, tedy diagramy chování a struktury.UML je standardním modelovacím jazykem s bohatou grafickou notací a komplexní sadou diagramů a prvků. Dalšími funkcemi jsou modelování obchodních procesů a datové modelování, jež umožňuje tvorbu logického a fyzického datového modelu. Do modelování je zařazeno i mapování struktury modelů.

U EA nechybí možnost code engineering. Umožňuje generovat zdrojový kód v celé řadě objektově orientovaných jazyků. Reverse code engineering umožňuje získat třídní diagram anebo fyzický datový model z již naimplementovaných tříd anebo z vytvořené databáze.

Vývoj software je v současné době časově a zdrojově náročnou situací. Na spolupráci při vývoji jednoho systému se podílí celé týmy vývojářů, z nichž každý je odpovědný za dílčí část celého projektu. Důležité je dodržení integrity mezi jednotlivými částmi, je tedy nutné zajistit version control (řízení verzí). K tomu je potřeba CVS - Concurrent Versions System a SVN - Subversion.Oba tyto systémy slouží ke správě verzí projektů a Enterprise Architect je podporuje jednak v lokální, jednak v serverové verzi.

Jednou z hlavních činností při návrhu a vývoji je zejména dokumentace. V EA se dá dokumentace psát ke všemu, co v něm vytvoříte, dá se také dodatečně nahrát z nějakého jiného zdroje k projektu, který je již vytvořený a samozřejmě také umí generovat dokumentaci z toho co je hotové. Dokumentace může být v PDF, RTF, HTML a XML.

EA podporuje hned několik různých typů testování, jako Unit testy, Integrační testy, Systémové testy či Akceptační testy.

1.2.2 Funkce nástroje EA

Podpora všech 14 diagramů UML verze 2.4.1. diagramy chování, aktivit, sekvenční, diagramy přehledu interakcí. Diagramy struktury: diagramy tříd, package diagramy, diagramy komponent, diagram nasazení. Dále podporuje Business proces modeling a i všechny starší diagramy verze UML 2.0

Jednoduše pochopitelné uživatelské prostředí, které dále nabízí rozsáhlé panely nástrojů, nastavitelná okna a zajímavé vizuální styly. Umožňuje vlastní nastavení rozložení oken, změnu a úpravu panelu nástrojů a tvorbu vlastních akcelerátorů. Úvodní stránka poskytuje okamžitý přístup k Learning Centeru, Nápovědu, příkladový model, Klávesové zkratky a možnosti přizpůsobení rozhraní. Prvky se dají najít ve schématech pomocí kontextového filtru a pole pro vyhledávání.

MDA - Model Driven Architecture (modelem řízená architektura) podpora pro transformaci jednoduchých prvků modelu do složitých cílů. Možnost řízení pomocí šablon pro transformaci, které se dají snad napsat a upravovat. Jsou podporované pro Java, C#, a

další. Také se dají vytvářet a synchronizovat specifické modely na jiných platformách od nezávislých vývojářů.

Jak jsem již zmínil v EA je řízený generátor RTF, který má v sobě šablony pro všechny modely, které podporují záhlaví, zápatí, obsah, vložené obrázky, titulní listy, složité vnořené tabulky atd. Dají se i vytvořit vlastní šablony pro pozdější opětovné použití. RTF dokumenty lze propojit s prvky modelu a upravovat pomocí vestavěného RTF editoru. Má možnost flexibilního výstupu s filtry a výběrovými kritérii. Také má HTML generátor pro vytváření detailních HTML zpráv, dá se také použít k vygenerování HTML sestav pro celý model a následné vyvěšení na internet nebo na lokální síť. Lze také ukládat přímo do PDF.

Reverse a Forward engineering poskytuje plně řízené šablony pro generování, které se dají upravit dle vlastních potřeb, nebo se dají napsat úplně nové dle potřeby. Dají se přidat i další cílové jazyky, kromě vestavěné podpory Java, VB.Net, atd..

Simulace je v EA podporována, dá se využít real-time model pro chování a analýzu, ověření syntaktické správnosti modelů chování. Jsou zde různé nástroje pro správu simulace jako triggery, zarážky, simulační proměnné atd. Pomocí simulace se dají identifikovat překážky, snížit rizika, odstranit redundanci a lépe pochopit změny.

Z databázového modelování je podporován zpětný engineering pro všechny známé DBMS jako Oracle, SQL Server, My SQL, Access, Postgre SQL a další. Model databáze poskytuje tabulky, sloupce, klíče, cizí klíče, a komplexní vztahy za použití UML a vestavěného profilu modelování dat. Dají se také generovat skripty DDL - data definition language (definuje datovou strukturu, obzvláště se používá u databází) k definici datové struktury databází.

EA poskytuje několik možností jak sdílet modely. Dá se buď sdílet .EAP (soubor ukládaný EA) soubor na síťovém disku. Toto se využívá u malých skupin. Dále se dá .EAP soubor replikovat, s tím že se určí hlavní soubor, ze kterého se udělají repliky a s kterým se poté repliky synchronizují. Nebo se dá využít databázové repository (úložiště), kde se dá využít i podpora spravování verzí. Z databázových repository (uložiště) jsou podporovány všechny známé databáze. Může se využít i XMI importů a exportů k řízení distribuce a

aktualizací v rámci projektu. Možnost exportu a importu základních referenčních údajů, aby se zabránilo vytvoření stejné informace vícekrát.

Pro správu verzí je zde integrace s Subversion, CVS a SCC kompatibilní s uložištěm pro správu verzí. V EA se verzuje na úrovni package. Porovnávací nástroj umožňuje prohlížení změn v aktuálním modelu s nejstaršími soubory na disku. K manipulaci s verzovanými soubory se dá využít formátu XMI.

K podpoře XML slouží vestavěný profil XSD, který zjednodušuje vývoj schématů XML pomocí UML. Díky tomu se zde dají tvořit komplexní schémata XML z UML modelů. Dají se provádět i opačně analýzou XML schémat do modelů UML.

Projekty v EA se dají zabezpečit. EA podporuje zabezpečení pro uživatele, stejně tak i pro skupiny a dá se vybrat z nesčetného množství práv. Tato práva mohou být nastavena jednotlivým prvkům, nebo i celým částem.

V EA je podpora řízení projektů – Project management jako Kalendář pro osobní plánování s přidělenými úkoly, týmové plánování a přidělování úkolů, poštovní klient pro rychlou komunikaci s členy týmu a skupinami, podpora pro zvýraznění rizikových položek, podpora Use Case Metrics , které umožňují na základě parametrů odhadnout čas a potřebné zdroje pro vypracování. Dále poskytuje informační podporu o stavu systému.

Dalšími funkcemi, kterými EA disponuje je Systémový Engineering konkrétně pokročilý model chování a spustitelná generace kódu.

EA je velmi složitý nástroj, ve kterém je nespočetné množství funkcí a žádný jeho uživatel nikdy nevyužije všechny jeho přednosti, protože je velice komplexním nástrojem.

2. Zásady reengineeringu a možnosti využití pro inovaci IS

Než v roce 1993 vydali Michael Hammer a James Champy knihu Reengineering the Corporation: A Manifest for Business Revolution (v češtině pod názvem „Reengineering: radikální proměna firmy“), nebyl pojem reengineering všeobecně známý. Rozmach nastal právě s touto knihou, která se stala výchozím zdrojem informací a vychází z ní většina metodik spojených s reengineeringem. A doposud není žádná jiná kniha, která by problematiku reengineeringu vysvětlovala, nebo definovala lépe.

2.1 Definice reengineeringu

„Reengineering v podstatě znamená zásadní přehodnocení a radikální rekonstrukci (redesign) podnikových procesů tak, aby mohlo být dosaženo dramatického zdokonalení z hlediska kritických měřítek výkonnosti, jako jsou náklady, kvalita, služby a rychlost.“[1 str. 38]

Kromě formální definice je v knize ještě uvedena neformální definice: *„Reengineering „představuje „nový začátek“. Nejde o vylepšování toho, co již existuje, nebo o provádění dílčích změn, které ponechávají základní struktury netknuté. Nejde o záplatování – pospravování existujících systémů, aby pracovaly lépe. Ve skutečnosti to znamená vzdát se zavedených postupů a nově pohlédnout na práce, jež jsou nezbytné k vytvoření výrobku nebo služby firmy, resp. poskytnutí hodnoty zákazníkovi. Znamená to položit si otázku: „Jak by vypadala tato firma, kdybychom ji dnes – nesoučasnými znalostmi a s využitím dnešních technologií – budovali znovu?“ Provést reengineering firmy znamená odhodit staré systémy a začít znovu. Jeho součástí je návrat k počátku a k nalezení lepších způsobů práce.“[1 str. 37]*

Jak je vidět v obou definicích, reengineering nespočívá v provádění dílčích změn na stávajícím systému. Reengineering je brán jako nový začátek, tak jako kdybychom vše stavěli od nova. Jde o vybudování nového „systému“ dle nejnovějších požadavků v souladu s cíli, které mají naplňovat. V ideálním případě by měly poskytovat přidanou hodnotu firmě, popř. zákazníkovi.

Ve formální definici jsou obsáhnuta čtyři klíčová slova, která definují reengineering: zásadní přehodnocení, radikální rekonstrukce (redesign), dramatické zdokonalení a podnikové procesy.

2.1.1 Zásadní

První klíčové slovo je zásadní přehodnocení. Při reengineeringu je nutné klást si nejzákladnější otázky a to: proč děláme to, co děláme? Proč to děláme právě tímto způsobem? Je nutné nejprve si říci, co musíme udělat a až poté jak to máme udělat. „*Reengineering nepovažuje nic za předem dané. Ignoruje to, co je, a soustřeďuje se na to, co by být mělo.“[1 str. 38]*

2.1.2 Radikální

Druhým klíčovým slove je „radikální“. *„Je odvozeno z latinského slova „radix“, což znamená kořen. Radikální rekonstrukce (redesign) znamená jít ke kořenům věci: nedělat povrchní změny nebo dílčí úpravy toho, co již existuje, ale to, co je staré, odvrhnout. Radikální rekonstrukce v reengineeringu znamená, že se nerespektují žádné již existující struktury a postupy a vytvářejí se zcela nové.“[1 str. 39]* Reengineering je „zásadní obnova“, nejde o žádné vylepšování, propracování, nebo dílčí změny.

2.1.3 Dramatické

Reengineering by měl být zaváděn, když je zapotřebí výrazná změna. Dramatická zlepšení vyžadují rázně odbourat staré a nahradit je něčím novým.

Předně to jsou firmy, které jsou v hlubokých potížích a nemají jinou volbu. Za druhé jsou to firmy, které dosud potíže nemají, ale jejich vedení má dostatek předvídavosti a uvědomuje si, že se již blíží.

Třetí skupinu firem, jež přistupují k reengineeringu, tvoří ty, které jsou ve „vrcholné kondici“. Nemají žádné zřejmé potíže, ani teď, ani na obzoru, ale jejich vedení jsou ambiciózní a agresivní.

Reengineering tedy mohou provádět všechny firmy, nehledně na jejich ekonomickou situaci. Ovšem nejlepší výchozí pozici mají firmy, které jsou ve třetí skupině, jelikož nemají žádné problémy.

2.1.4 Procesy

„Definujme podnikový proces jako soubor činností, který vyžaduje jeden nebo více druhů vstupů a tvoří výstup, který má pro zákazníka hodnotu… Pod vlivem koncepce Adama Smítne, která rozčleňuje práci do nejjednodušších úkolů, jež přiděluje specializovaným pracovníkům, se moderní firmy a jejich manažeři zaměřují na individuální úkoly… a mají sklon ztrácet ze zřetele větší cíl… Individuální úkoly v rámci procesu jsou důležité, ale žádný z nich nemá pro zákazníka význam, pokud není v pořádku celý proces.“[1 str. 40]

Je tedy důležité, aby se provedl i reengineering manažerského pohledu na procesy. Je potřeba dívat se na věci i z jiného pohledu než jen jako na dílčí úkoly, členění z hlediska profesí, infrastruktur atd. Je nutné příjmout orientaci na procesy, na podřízení procesů zákazníkům a hledat i jiné způsoby jak dosáhnout kýženého cíle, který firma, nebo zákazník požaduje.

2.1.5 Co reengineering není

Mnoho lidí, když slyší pojem reengineering, si vybaví že jde v podstatě jen o zlepšení něčeho, co už vlastně znají. Mohou například říci že: *„Reengineering je jiný název pro redukci rozsahu činností. Nebo jej ztotožní s restrukturací nebo jiným, právě aktuálním způsobem zlepšování podnikových činností.“[1 str. 52]* Reengineering není totožný s restrukturací či se zmenšením rozsahu činností, protože redukce rozsahu činností a restrukturace pouze znamenají dělat méně s menšími kapacitami. Reengineering není nic, co je spojeného s reorganizací – změnou řídících úrovní a organizačních struktur, protože

problémy podniků nevyplývají z jejich organizačních struktur, nýbrž z jejich procesních struktur. Reengineering není ani zvyšování jakosti, úplným řízením, nebo nějaký dalším projevem. Programy zvyšování jakosti a reengineering mají dosti společného. Oba dva se zabývají významem procesů a oba dva začínají u potřeb zákazníka a od nich zpětně odvíjí své činnosti. Ale zásadně se odlišují v tom že, programy zvyšování jakosti se aplikují na stávající podnikové procesy a snaží se je neustále a postupně zdokonalovat. Namísto toho reengineering hledá změny ne zlepšováním stávajích procesů, nýbrž jejich odstraněním a nahrazením novými procesy. Také je zde jiný pohled na řízení změn, než v programu zvyšování jakosti.

Nakonec můžeme říci, že reengineering znamená dělat více s menšími vstupy. Proto se vrátím k původní definici reengineeringu, kterou vyjádřím pouze dvěma slovy: začít znovu. *„Reengineering začíná s čistým listem papíru. Reengineering znamená hledat nové přístupy k struktuře, která se jen málo podobá té z minulých období nebo je od nich dokonce naprosto odlišná. Reengineering je hledání nových modelů organizace. Reengineering je nový začátek.“[1 str. 53]*

2.1.6 Role informačních technologií

Podstatnou roli při reengineeringu hrají informační technologie, protože firma, která nedokáže změnit způsob svého pohledu na informační technologie, nemůže provádět reengineering. *„Firma, jež ztotožňuje informační technologii s automatizací, nemůže při reengineeringu uspět. Firma, která hledá problémy a pak způsoby jejich řešení na základě technologie, nemá při reengineeringu šanci.“[1 str. 84]*

Informační technologie mají jednu z hlavních rolí v reengineeringu, avšak velice jednoduše se může stát, že bude špatně „obsazena“. IT je součástí každého reengineeringového úsilí, je to nedílná součást spolutvorby, protože reengineering umožňuje provést. Ovšem špatné využití informačních technologií může docílit toho, že se reengineering zcela zastaví, nebo nám neumožní správně pokračovat. Stane se to, že se posílí stará schémata chování a staré způsoby pohledu myšlení.

K tomu aby si mohli analyzovat možnosti informačních technologií a představit si jejich následnou aplikaci, je nutné, aby si firmy osvojily způsob myšlení, kterému se moc neučí. Většina manažerů umí myslet deduktivně. To znamená, že dokáží identifikovat problémy a následně hledat a vyhodnocovat jejich různá řešení. Ale aplikace informačních technologií při reengineeringu potřebuje, aby byli schopni myslet induktivně – je to schopnost nejdříve rozpoznat, jaké možnosti informační technologie poskytuje a až poté hledat problémy, které by mohla vyřešit.

Jeden z velkých problémů, kterým se firmy při reengineeringu dopouštějí, je ten, že se dívají na svoje informační technologie ze současného pohledu a ne z pohledu budoucího. *„Jak můžeme užít těchto nových možností k zlepšení, k zdokonalení či zefektivnění toho, co již děláme.“* namísto toho *„Jak můžeme technologie využít, abychom mohli dělat věci, jež dosud neděláme.“[1 str. 86]* Při reengineeringu jde o to co nejefektivněji využít veškerý potenciál, který nám informační technologie poskytují. Jde o to jak již jsem řekl předtím, myslet o technologii induktivně.

2.2 Postup při reengineeringu

2.2.1 Tvůrci reengineeringu

Než se vůbec firma pustí do reengineeringu, je nutné, aby si stanovila, kteří lidé ho budou provádět. Jde o to tyto lidi správně vybrat a organizovat je. Protože nikoli firma, ale lidé budou reengineering provádět. Proto je výběr a organizování jedním z důležitých bodů úspěchu.

Michael Hammer, ve své knize Reengineering – radikální porměna firmy, zaznamenal nejčastěji tyto role: Vůdčí osobnost, Vlastník, Reengineeringový tým, Řídící tým, Řídící výbor a Reengineeringový car. V ideálním případě vůdčí osobnost určuje vlastníka, který vytváří tým k provedení reengineeringu a to s pomocí cara pod záštitou řídícího výboru.

Vůdčí osobnost

Jde o osobnost - lídra, která reengineering započne, je to vyšší vedoucí pracovník, který má dostatečný vliv na to utvrdit lidi v radikálních změnách, které reengineering obnáší. Bez tohoto typu člověka může firma mít vše na reengineering navržené, ale bez iniciativy této osoby k žádnému reengineeringu nedojde. A i když se reengineering rozběhne, tak bez vůdčí osobnosti zájem o něj postupem času uvadne a to povede k tomu, že bude odsunut na vedlejší kolej. Tímto typem se stává osoba, která má znalosti o reengineeringu, dobrou pozici ve firmě a dostatek energie k tomu, aby mohla provést tyto radikální změny a dotáhnout celý proces do zdárného konce. Jeho pozice ve firmě musí být taková, aby bylo jeho formální postavení výš než všech lidí, kterých se reengineering týká, nebo by měl mít alespoň vyšší pravomoce než zmíněné osoby. Jeho hlavním úkolem je předkládat vize a motivovat pracovníky, kteří se na tom podílejí. Také musí být přesvědčený a nadšený pro věc, což se pak přenese na další pracovníky. Vůdce určuje vlastníky jednotlivých procesů, kterým definuje vize a standardy, jichž se mají držet. Měl by být také dobrým motivátorem, ale na druhou stranu i tím kdo bude dohlížet na správné plnění povinností všech jeho podřízených. K tomu by měl využívat všechny prostředky, jimiž firma disponuje – např systém řízení, odměn, atd.

„Vlastník“ procesu

Vlastníkem procesu se myslí osoba odpovědná za konkrétní proces, která má svou pozicí ve firmě díky svému postavení, anebo přiřazenou vůdčí osobností. Jeho úkolem je zajistit, aby daný proces prošel reengineeringem. Jsou to lidé, kteří v daném procesu zastávají jednu z funkcí a zároveň celému procesu rozumějí. Avšak vlastník samostatný reengineering neprovádí, nýbrž zajišťuje jeho průběh a provedení. Určuje reengineeringový tým a dělá vše, aby tým mohl realizovat svou práci. Zároveň je také kritik, mluvčí, poradcce,styčný důstojník týmu.

Reengineeringový tým

Řekl bych, že po vůdčí osobnosti je toto nejdůležitější prvek z hlediska osob v reengineeringu. Tým je ta skupina lidí, se kterou je spojena veškerá práce a úsilí, jsou to ti, kteří reálně provádějí změny a jsou za ně odpovědní. Nutné je to, že žádný tým nemůže provádět reengineering více než jednoho procesu. Tým se skládá ze dvou typů lidí, z internistů – to jsou ti lidé, co jsou v daném procesu nějak zapojeni, nebo jej znají a

externistů, kteří s procesem, který prochází reengineeringem nejsou nijak obeznámeni. Jde o to, že každý typ má na daný proces nějaký pohled, který se poté může projevit při reengineeringu. Internisté se budou chtít poučit z toho, co o procesu vědí a externisté zase budou hledat úplně nová řešení. Kdyby byl tým složen jen z jedné skupiny lidí, nemohlo by to fungovat, protože jeden, nebo druhý typ lidí má na daný proces velmi omezený pohled. *„Reengineering předpokládá invenci a hledání, tvořivost a schopnost syntézy. Reengineeringový tým musí být schopen pracovat v podmínkách nejednoznačnosti. Členové týmu musí být připraveni na to, že budou chybovat, a musí se ze svých chyb umět učit.“[1 str.109]* Účast v týmu je dlouhodobou záležitostí, protože čim déle budou členové provádět reengineering, tím lépe budou vědět jak dané změny provést. Kdyby se v týmu členové dost často střídali, docházelo by ke ztrátě informací a celý proces by trval mnohem déle.

Řídící výbor

Je to skupina lidí, většinou řídících pracovníků, nebo vlastníků procesů, kteří utváří celkovou strategii reengeneeringu celé organizace. Lídr reengeneeringu by měl být v čele této skupiny. Zde se zabývají, úkoly které přesahují přes jednotlivé procesy a úkoly. Také je jeho úkolem určovat prioritu a kontrolovat jednotlivé reengeneeringové týmy. Výbor také řeší problémy, na které týmy během reengineeringu narazí a konflikty mezi týmy. Řídící výbor působí jako nejvyšší rozhodovací orgán, nad kterým už nikdo není a pomáhá tak ostatním při reengeneeringu.

Reengeneeringový car

Car je tu proto, aby podporoval lídra, který nemusí mít čas na to, aby kontroloval všechny reengineeringové projekty. Jeho úkolem je podporovat vlastníky procesů a reengineeringové týmy, vytvářet pro ně dobré podmínky a koordinovat jednotlivé probíhající projekty. Pomáhá vybírat vhodné internisty, ale také externisty do týmu na základě svých zkušeností, protože je to typ člověka, který reengeneering začal dříve než ostatní. Dohlíží na to, aby se týmy neodchylovaly od „správné cesty“. Svolává a řídí diskuze mezi vlastníky procesů, pro získání informací. Stará se také o to, aby byla připravena infrastruktura, která nemusí odpovídat prvotním plánům, které byly vymezeny na začátku reengineeringového úsilí.

2.2.2 Metodiky reengineeringu

Existuje několik metodik, které se liší rozsahem, zaměřením a poměrem praktické a teoretické části. Záleží jen na problematice daného reengineeringu a rozhodnutí firmy, jakou metodiku si zvolí. V této kapitole se zaměřím na ty nejvýznamnější.

Tabulka 1, Seznam metodik
Zdroj Vlastní

Metodika	**Zaměření**
Hammer, Champy	Konzultantský/akademický
Davenport	Akademický
Manganelli, Klein	Konzultantský
Kodak	Uživatelský
DoD	Státní správa

2.2.2.1 Metodika Hammer, Champey

Ti svou metodiku definují jako *„fundamentální „přemýšlení“ a radikální rekonstrukci strategicky kritických podnikových procesů.“[2 str. 38]* Hlavní problémy vidí v nedostatečném managementu a nejasných cílech firmy. Zlepšení těchto dvou problémů je podle nich základní faktor úspěchu. Okrajově se potom zabývají možnou překážkou a to v odporu zainteresovaných lidí, který je ale dodnes považován jako jedna z největších překážek. Svou metodiku rozdělili do 6 hlavních kroků.

1. Uvedení do reengineeringu – Vrcholný management shrne současnou situaci v podniku a vylíčí ji zaměstnancům.

2. Identifikace podnikových procesů (PP) – Znázornění PP pomocí grafů.

3. Výběr PP k reengineeringu – Dochází k výběru nejvhodnějších PP, který zákazníkům přinesou zvýšený efekt. Vybírají se hlavně bezproblémové procesy.

4. Poznání vybraných PP – Jedná se o analýzu v porovnání stávajících procesů s očekáváním těch po reengineeringu.

5. Redesign vybraných PP – Hlavním cílem projektu, zde dochází k vytváření inovací

6. Implementace nových PP – Konec reengineeringu, končí na úrovni plánování projektu

2.2.2.2 Metodika T. Davenporta

V metodice T. Davenporta jsou hlavním prvkem reengineeringu informační technologie, protože v nich vidí potenciál inovace. Dále se jeho pozornost upíná na záležitosti organizační a personální, které podnikové procesy představují a potřebují. Podle něj by měly být staré metody reeengineeringu jednoduše propojitelné s novými. I jeho metodika má 6 kroků.

1. Vize a cíle – organizace si stanoví potřebné vize a cíle, na které se při procesech zaměřuje. Hlavním cílem je snížení nákladů. Podnik by se, ale neměl upínat pouze na snižování nákladů, protože jsou zde další důležité cíle jako např.: uspokojení zaměstnanců, zlepšení efektivity práce, snížení potřeby času na jednotlivé úkoly a další cíle, které zamezují redukci nákladů.

2. Identifikace PP – nejdříve zjistí, které PP procesy potřebují změnit. Davenport doporučuje vybrat max. 15 procesů, které tvoří jádro podniku.

3. Poznání a měření procesů - dochází k nastavení srovnávacích hodnot výkonu nových procesů *1+. Dochází zde k modelování procesů a zároveň k jejich měření

4. IT – zjišťují se možnosti implementace

5. Prototypování procesů – jde o vytvoření funkčního modelu procesu. Vytváří se pro zaměstnance podniku, aby se seznámili se změnami a zároveň se mohli vyjádřit k případným změnám prototypu.

6. Implementace procesů – uvádění nových procesů do podniku, kdy zároveň dochází k jejich testování. Davenport implementaci považuje za úspěch celého projektu a předpokládá, že potrvá minimálně jeden rok.

2.2.2.3 Metodika Manganelliho a Kleina

Tato metodika doporučuje zaměření se na procesy, které přímo podporují strategické cíle organizace a požadavky zákazníků. Cílem je vývoj produktu. Problémy pro ně představují organizaci, čas, náklady a rizika – což jsou de-facto faktory organizačních projevů. Jejich metodika se jmenuje „Rapid - Re“ a má 5 kroků.

1. Příprava projektu – stanovení cílů a připravení projektu

2. Identifikace – definice organizační struktury orientované na zákazníka, vyměření procesů, které se budou muset změnit či nově vytvořit

3. Vize – zvýšení efektivnosti procesů, závisí na přesném měření stávajících procesů

4. Redesign

5. Technický – řeší se design informačního systému a užití technologií

6. Personální – vytvoření nového pracovního prostředí

7. Transformace – jedná se o implementaci změněných procesů a pracovního prostředí v podniku

2.2.2.4 Metodika Kodak

Metodika mezinárodní organizace Kodak vyvinutá za účelem řešení typických problémů velkých nadnárodních firem. Je silně ovlivněna přístupem Hammera a Champeyho a má 5 kroků.

1. Iniciace projektu – jedná se o plánování projektu a definici všech potřebných administrativních projektových pravidel a procedur

2. Poznání procesů – nastavení společných cílů, vytvoření modelu procesů podniku a zajištění manažerů, kteří budou za projekt odpovědni

3. Design nových procesů – dochází k naplánování implementace

4. Transformace podniku – tento krok je zaměřen na implementaci a přizpůsobení infrastruktury podniku nově zkonstruovaných podnikových procesů

5. Řízení změny – řešení průběžných chyb v procesech

Je důležíté vybrat správnou metodiku, podle které se bude poté reengineering provádět, což není vůbec jednoduché, ale výsledný projekt by měl být ten který:

- je zaměřen na zákazníky;
- staví na nejlepších zkušenostech a respektuje ostatní;
- je vytvořen pro budoucnost;
- přináší významná a podstatná zlepšení činnosti celého podniku

V následující tabulce č. 2. můžete vidět srovnání jednotlivých metodik, tak jak to udělal Václav Řepa ve své knize „Podnikové procesy - procesní řízení a modelování“.

Tabulka 2, Porovnání metodik
Zdroj: [2 str. 41]

	Krok 1: Příprava projektu	**Krok 2: Rekonstrukce procesu**	**Krok 3: Implementace**
Hammer, Champy	1. Uvedení do reengineeringu 2. Identifikace 3. Výběr procesů	4. Poznání procesů 5. Redesign procesů	6. Implementace
Davenport	1. Vize a cíle 2. Identifikace procesů	3. Poznání a měření procesů 4. Informační technologie	5. Prototypování 6. Implementace
Manganelli, Klein	1. Příprava projektu 2. Identifikace	3. Vize 4a. Technický design 4b. Personální design	5. Transformace
Kodak	1. Iniciace projektu 5. Řízení změny	2. Poznání procesů 3. Design nových procesů 5. Řízení změny	4. Transformace podniku 5. Řízení změny

3. Současný stav

3.1 Nejznámější case nástroje na českém trhu

Známými case nástroji na českém trhu jsou: SELECT ARCHITECT, IDS SCHEER ARIS Design Platform, MS Visio 2010, Magic Draw UML a ENTERPRISE ARCHITECT. V následující části se o jednotlivých nástrojích dozvíte jejich stručný popis, přičemž ENTERPRISE ARCHITECT byl více rozepsán v kapitole č. 1.2.

Prvním nástrojem je SELECT ARCHITECT od společnosti Select business solutions. Jedná se o komerční software, kde jedna vývojářská licence stojí kolem 6000 amerických dolarů. Jeho účelem je objektově orientovaný vývoj aplikací ve vícevrstevné architektuře, hlavně podpora Upper Case – analýza. Nabízí širokou podporu sběru požadavků, procesního a datového modelování, objektově orientovaného modelování v notaci UML, generování kódu a generování relačních databázových schémat. Select Architect umožňuje rychlý vývoj aplikací pomocí objektově orientovaného modelování. K rychlosti a jednoduchosti vývoje přispívá i široká škála šablon a návrhových vzorů. Nabízí také možnost simulace firemních procesů a tím zajistí funkcionalitu systému shodnou se skutečnými požadavky uživatelů. Mezi možnosti tohoto produktu patří i generování databázových schémat včetně užitečné funkce jejich reverse engineeringu, což usnadní pochopení stávajících tabulek a pohledů. Nástroj je dobře škálovatelný – od jednoho uživatele až po velký tým. Podporuje celý životní cyklus aplikací.

Druhým nástrojem je produkt společnosti Software AG, která byla ještě donedávna známá pod názvem IDS SCHEER. Hlavním účelem je návrh a modelování business procesů. Tento software je nabízený v různých variantách přesně odpovídajících kriteriím zákazníků.

ARIS Business Architect & Designer umožňuje účinně modelovat procesy a následně optimalizovat činnosti.

ARIS Process Governance slouží k zlepšení kvality procesů, flexibility, přehlednosti definicí politik, rolí a odpovědností. ARIS IT Architect & Designer umožňuje harmonizaci a stadardizaci činností organizace s jejími IT systémy.

ARIS IT Inventory umožňuje udržovat informace o aplikacích a technologiích ve firmě ve strukturované formě. Informace se ukládají do centrálního úložiště ARIS Repository.

ARIS Business Simulator umožňuje realisticky simulovat a dynamicky analyzovat podnikové procesy.

ARIS Business Publisher zajišťuje informovanost zaměstnanců. Je to procesní nástroj, který s nízkými náklady garantuje flexibilní dostupnost informací o procesech nebo IT architekturách.

Další je Magic Draw UML, který je jedním z nejvybavenějších a funkčně nejsilnějších nástrojů na trhu. Tento nástroj nabízí všestrannou podporu pro analýzu, návrh a objektově orientované modelování systémů. Vyniká především v oblastech reverzního engineeringu a round-trip engineeringu, je integrovatelný s mnoha nástroji a podporuje nejrozšířenější programovací jazyky. Díky obsaženému nástroji Teamwork Server umožňuje jako jeden z mála programů týmovou práci na daném projektu na bázi klient-server architektury. Komponenty a části systému, se kterými uživatel pracuje, jsou zamykány, a tato informace je pak rychle šířena k ostatním připojeným uživatelům.

MS Visio 2010 Microsoft Office Visio 2010 je univerzální nástroj, který umožňuje pomocí řady diagramů vizuálně dokumentovat a navrhovat informační systémy. Kromě toho slouží jako nástroj tvorby obchodních a dokonce i technických modelů. Jedná se zejména o nástroj pro široké kancelářské využití s akcentem na informační technologie. Visio 2010 podporuje modelování UML diagramů a celou řadu dalších modelů: DFD, ROOM, Jacksonův diagram, model architektury aplikací, paměťové diagramy, COM a OLE, ExpressG, ORM, uživatelské rozhraní systému Windows a model databáze.

3.2 Současný pohled na reengineering

V současné době je reengineering, po uvedení tohoto pojmu v 90. letech, kdy reengineering byl velice zajímavé a ostře sledované téma, spíše ve stagnaci.

Je to z toho důvodu, že provádět kompletní reengineering celé organizace stojí veliké úsilí a to nejen finanční. Většinou se tedy společnosti, které se rozhodnou provést reengineering uchylují pouze k částečnému reengineeringu. Tedy provádějí reengineering pouze u stěžejních porcesů, které jsou pro chod organizace nezbytné a jejich zlepšení by pro organizaci mělo některé, nebo všechny z daných přínosů: zlepšení finanční náročnosti procesu, optimalizace dle stávajících potřeb, zrychlení, zjednodušení atd. Nebo organizace provádějí kompletní reengineering postupně po jednotlivých úsecích, v rámci rozdělení organizační struktury. Reengineering potom může probíhat i několik let.

Při vyhledávání této problematiky v databázi odborných článků ProQuest, jsem nejčastěji narážel na informace o tom, že v současné době reengineering nejvíce probíha v oblasti zdravotnictví a zdravotní péče. Dále jsem se pomocí databáze dověděl, že nejvyšší zájem o reengineering v porovnání s počtem vydaných článků v daném roce byl mezi lety 1990 až 1996. Od roku 1996 klesal každoročně počet vydaných článků v databázi, tak že každý následující rok byl počet článků nižší. V roce 2012 byl počet článků zabývající se tématikou reengineeringu nejnižší za celou dobu, co se články v databázi objevovaly. Z tohoto zjištění lze také vidět, jak se zájem o reengineering v současnoti oproti minulým létům snížil.

4. Reengineering procesu vývoje IS ve společnosti OR-CZ

4.1 O společnosti OR-CZ spol. s.r.o.

Společnost OR-CZ se zabývá informačními technologiemi, konkrétně v několika odvětvích: OR- SYSTEM – vývoj vlastního ERP systému, Business Intelligence – kompletní řešení podnikové výkonnosti, Marie PACS – komplexní digitalizace radiologických provozů, OR-Info – komunikační systémy pro podporu týmové spolupráce, Systémy personální identifikace – docházkové, přístupové a kamerové systémy.

4.1.1 OR-SYSTEM

OR-CZ se zabývá vývojem a implementací vlastního ERP systému, který se jmenuje OR-SYSTÉM. Je to nástroj pro plánování a řízení výroby v organizacích s výrobou kusovou, sériovou i hromadnou. Pokrývá celé spektrum informací potřebných pro efektivní řízení podniku bez ohledu na jeho velikost. Hlavní ambicí OR-SYSTEMu je podporovat uživatele při získávání a zvyšování jeho konkurenčních výhod zabezpečením aktuálních a relevantních informací pro komplexní řízení a včasné rozhodování - vždy ve vhodném okamžiku a na správném místě.

4.1.2 Stávající OR-SYSTEM

OR-SYSTEM má víceúrovňovou objektovou strukturu typu klient-server. Technologie klient/server vychází z filosofie tzv. „tenkého“ klienta. Klientská strana pouze prezentuje data a provádí formální kontroly správnosti vkládaných dat. Je programována pro operační systém MS Windows, využívá grafické uživatelské rozhraní. Aplikační část (Business logika), která je nositelkou algoritmů a zejména se stará o důsledné zabezpečení strategických dat je provozována na serveru a portována na operační systémy Linux, UNIX a MS Windows. Zpracovávaná uživatelská data ukládá OR-SYSTEM v relační databázi – podporovanými produkty jsou ORACLE a MS SQL, historicky pak podporoval DB2 a Informix.

Serverová část systému je napsána v programovacím jazyce Cobol, což je jazyk vyvinutý v 60. letech minulého století a má sloužit hlavně pro obchodní a databázové aplikace, a klientskou část je pomocí nástroje ISA Dialog Manager, který zajišťuje grafické rozhraní.

4.1.3 Koncepce OR-SYSTEMU

Systém je sestavován z jednotlivých modulů – Jádra systému, Obchodu, Konstrukce a technologie, Výroby, Kalkulací, Ekonomiky a Speciálních modulů. Jednotlivé programové moduly jsou chápány jako objekty, což umožňuje jednak jejich snadné skládání do celků, zabezpečujících zpracování vždy určitých oblastí, jednak vnořování na různých úrovních. Uživatel může získávat údaje agregované, ale i detaily v různých vrstvách podrobnosti. Tato modularita umožňuje postupné plnění datové základny a realizaci jednotlivých oblastí systému po krocích.

Duševní vlastnictví zdrojového kódu umožňuje OR-CZ spol. s r. o. trvalý vývoj OR-SYSTEMu návazně na změny legislativy, základního software a především podnětné požadavky široké základny uživatelů. Důsledně otevřená koncepce umožňuje snadné rozšiřování o další softwarové produkty a volbu nejvhodnějšího provozního prostředí s maximálním využitím stávající výpočetní techniky a základních softwarových nástrojů. Jednotná datová základna pro všechny úlohy eliminuje datovou redundanci a minimalizuje tak nároky na paměť. Jednotlivé části systému jsou datově i funkčně propojeny a dovolují tak přístup k veškerým aktuálním informacím ze všech modulů, které je možno provozovat na místně oddělených systémech. Funkce informačního systému je možno v širokých mezích snadno modifikovat, aby podporovaly dynamické podnikatelské strategie uživatele.

OR-SYSTEM plně respektuje právní předpisy. Jednotné obrazovky, sestavy, klávesy, ochrana dat a funkcí, jednoduchá instalace a národní prostředí zpřehledňují a usnadňují realizaci a provoz systému. OR-CZ spol. s r. o. podporuje trvale uživatele OR-SYSTEMu poradenskými službami a komplexním servisem. Poradenským a řešitelským servisem zajišťuje zákazníkovi nepřetržitý provoz a rozvoj instalovaného systému po celou dobu jeho používání. Do oblasti podpory náleží audit informačního systému a s ním související i

nabídka různých variant outsourcingu a provedení reengineeringu podnikových procesů, případně další poradenské služby z oblasti controllingu, logistiky, apod.

4.2 Nová technologie pro OR-SYSTEM v jazyce Java

4.2.1 Důvody ke změně

Společnost OR-CZ se rozhodla, po zvážení všech možných alternativ k tomu, že zakoupí nové jádro pro svůj systém v technologii Java. Důvodů k tomu měla hned několik.

První důvodem bylo, že stávající systém, tedy spíše jeho serverová část musí běžet na licencovaných Cobolovských serverech. Tedy nejen samotná společnost, která systém vyvíjí, musí mít placený licencovaný server, ale také každý uživatel samotného systému. Tedy přechod na technologii Java, kde se žádné licence platit nemusí, by přinesl menší finanční zátěž pro OR-CZ a jednotlivé uživatele.

Dalším důvodem proč přejít na novou technologii bylo zachování konkurenceschopnosti samotného systému a přechod na objektově orientovaný programovací jazyk. I když jazyk Cobol může být objektově orientovaný, společnost OR-CZ tuto jeho část nepoužívá a používá jen strukturovaný.

Nedostatek programátorů se znalostí jazyka Cobol, byl také jedním z dalších důvodů proč se společnost OR-CZ rozhodla pro jazyk Java. Na většině vysokých škol se jako základní programovací jazyky vyučují C# a právě Java. Sehnat na trhu práce programátora se znalostí jazyka Cobol je v dnešní době velký problém, ale programátorů se znalostí Java je dost, a jelikož se tento jazyk i vyučuje, tak s tím nebude v budoucnu problém.

4.2.2 Přechod na nové jádro

Proces výběru nového jádra probíhal v OR-CZ v horizontu několika let. Rozhodovalo se, zda nové jádro pro systém zakoupit, nebo zdali provádět vlastní vývoj. Vlastní vývoj nebyl z časového hlediska možný, proto se rozhodlo vybrat již hotové jádro. Nakonec bylo nové

jádro zakoupeno od partnerské společnosti, která zajišťuje vývoj ekonomické části OR-SYSTEMu a která měla kapacity toto jádro vyvinout. Toto řešení bylo pro obě dvě firmy výhodné, jelikož OR-SYSTEM by díky tomuto řešení mohl pracovat na jednom jádru a tudíž by bylo vše pro celý systém jednotné. Dříve tyto části byly izolované a probíhala mezi nimi pouze komunikace. Jak jsem se již zmínil, jádro bylo naprogramováno v jazyce Java, grafické uživatelské rozhraní zajišťuje Swing knihovna v Javě a komunikace s databází byla zajištěna pomocí Hibernate což je je framework napsaný v jazyce Java, který umožňuje tzv. objektově-relační mapování. Jak je patrné, celé nové jádro pracuje pouze s technologií Java namísto toho starého, které využívalo hned několik odlišných technologií pro každou ze zmíněných částí. Toto sjednocení v rámci jednoho programovacího jazyka umožňuje zlepšit efektivitu programování, jelikož nebude potřeba tvorby rozhraní pro komunikaci s jinými softwary, jak tomu je u stávajícího OR-SYSTEMu.

4.2.2.1 Stav Jádra

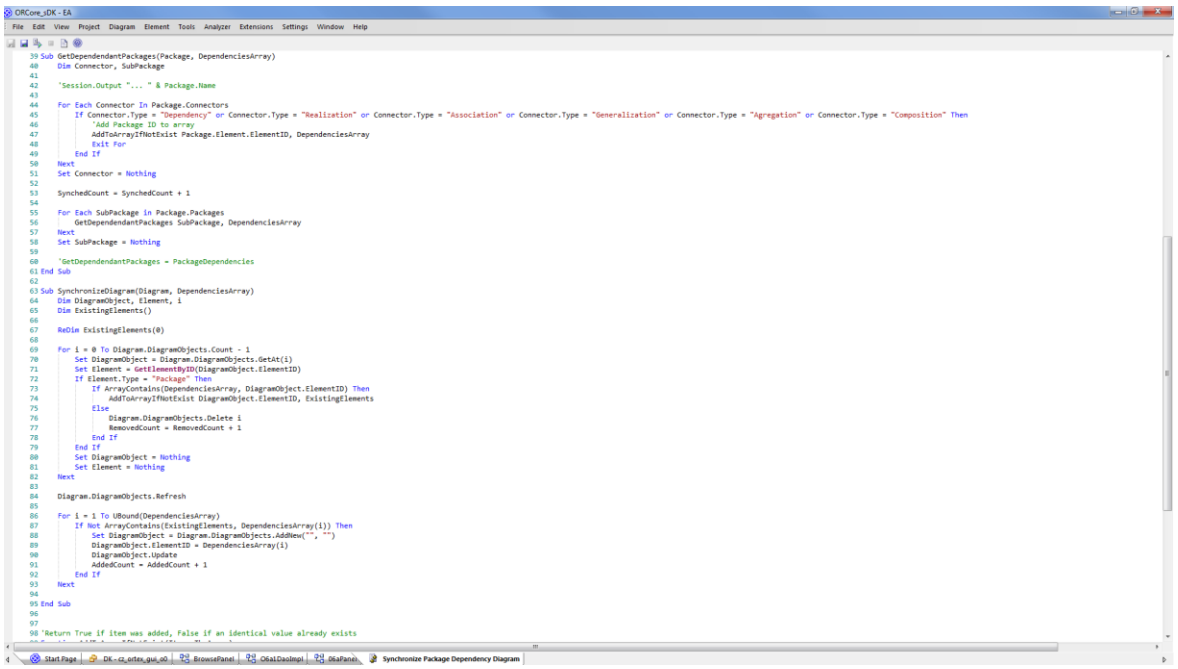
Po obdržení nového jádra od partnerské společnosti bylo potřeba nastudovat dané jádro. Dokumentace k jádru byla totiž jenom částečná s některými neúplnými vysvětleními. Mimo to byla dokumentace pouze přímo v Java kódu, tudíž nebylo možné si ji samostatně prohlédnout, ale muselo by se procházet kódem aby se dala vyčíst. Problém byl také v hledání vztahů mezi jednotlivými třídami. Bylo potřeba zjistit odkud je která třída zděděna, případně jaké další vztahy má s jinými třídami. Vymodelovat vztahy bylo tedy potřeba nejen mezi konkrétními třídami, ale i mezi abstraktnějšími package, ve kterých se dané třídy nacházejí.

4.2.2.2 Tvorba dokumentace pomocí EA

V programu EA se nachází funkce Source Code Engineering, což znamená práce se zdrojovým kódem. Tato funkce patří do části zabývající se Reverse Engineeringem – zpětným reengineeringem. Jde o to, že se ze zvoleného adresáře kde je uložený Java kód vygeneruje pomocí funkcí a mechanismů v EA diagram tříd. Tento diagram tříd odpovídá danému kódu, tedy třída v jazyce Java odpovídá vygenerované třídě v EA, konkrétně v diagramu tříd. Navíc každá vygenerovaná třída obsahuje seznam veškerých proměnných

a metod se všemi funkčními závislostmi. Dále je také vygenerována dokumentace, která se zobrazuje jako příloha jednotlivých tříd, jelikož dokumentace v Javě se také píše pro jednotlivé třídy.

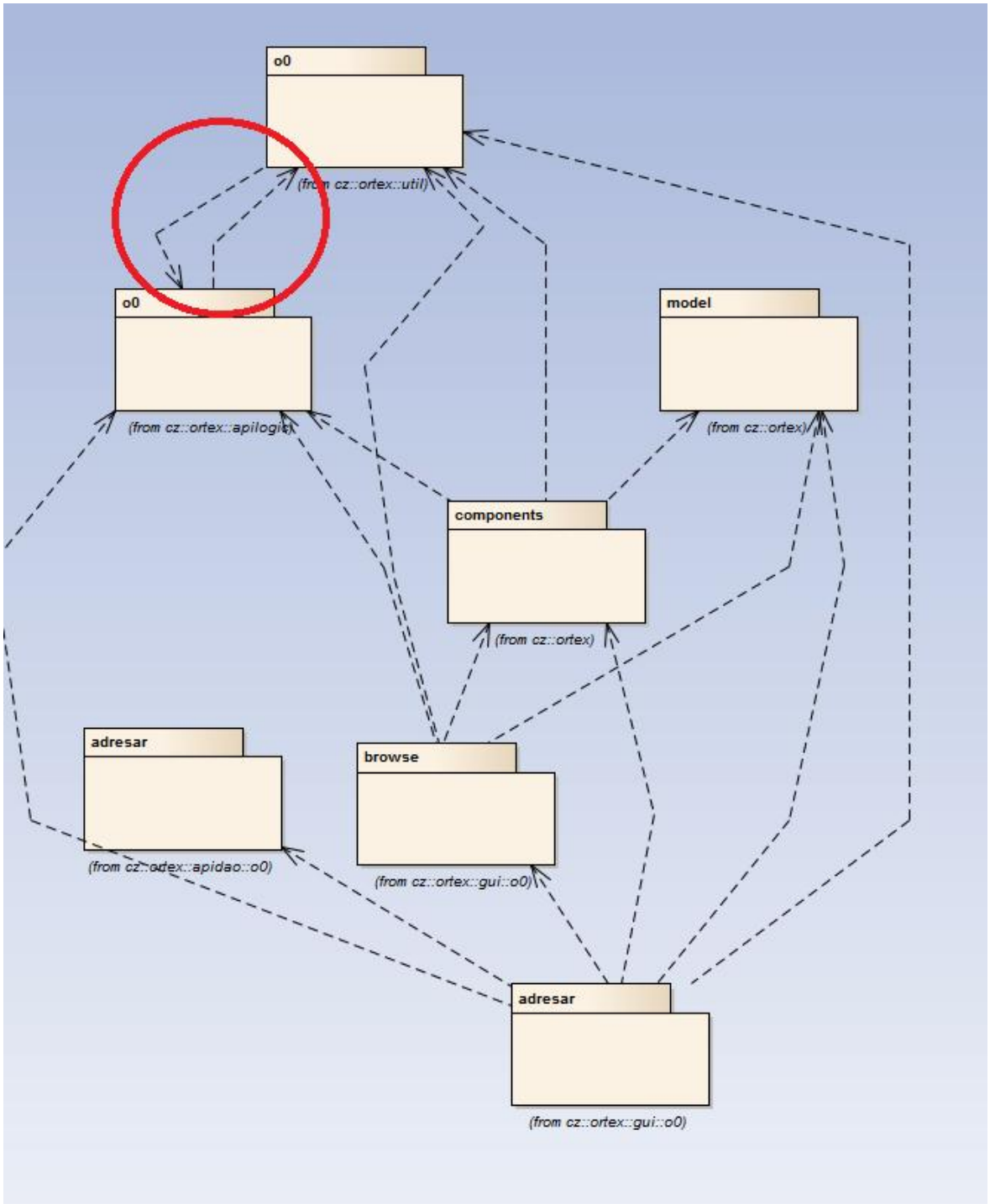
Na prvním obrázku je možné vidět package diagram, tedy diagram „balíčků“ vygenerovaný pomocí EA. EA sám o sobě umí vygenerovat pouze jednotlivé balíčky, ale nemá v sobě funkci, která by dokázala vygenerovat vztahy mezi nimi, tedy vztahy, které jsou mezi jednotlivými třídami v různých balíčcích. Ovšem to co EA nedokáže sám o sobě, se může dodatečně naprogramovat pomocí VisualBasic Scriptu.EA má přímo v sobě editor skriptů a mimo VBScript podporuje ještě JavaScript a JScript. Naprogramovala se tedy funkce, která na základě mnou zvolených vztahů, které se vyskytují mezi třídami v různých balíčcích, vygenerovala závislosti i mezi balíčky. Na obrázku číslo 1. je část VBScriptu, který byl použit pro generování závislostí mezi balíčky.



Obrázek 1: Ukázka VBScriptu
Zdroj: Vlastní

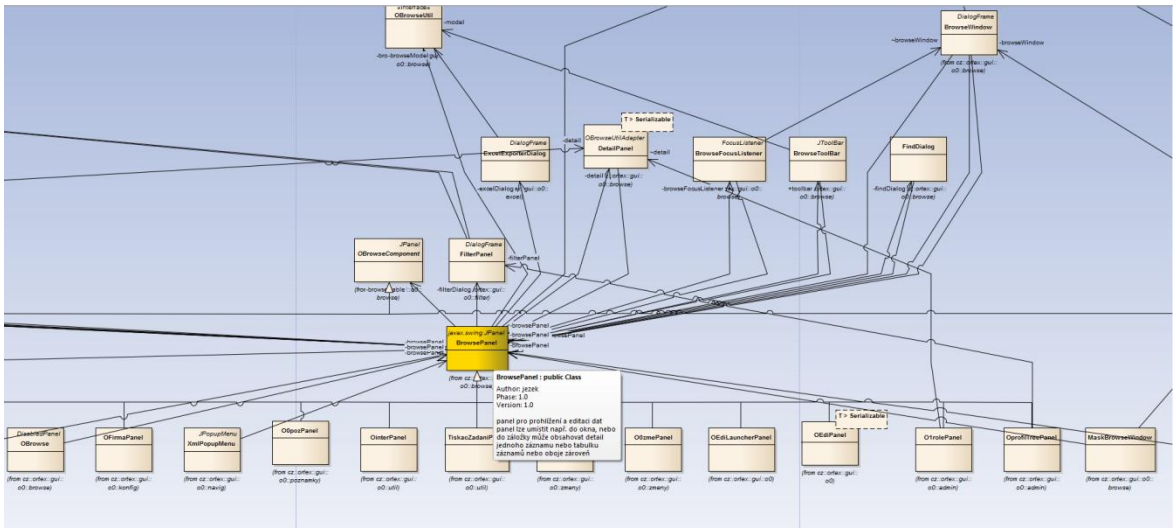
Na obrázku č. 2 je vidět výsledný diagram balíčků i s vygenerovanými vztahy. Konkrétně na tomto obrázku byla odhalena chyba, která by se jinak bez EA nezjistila. Tuto chybu pojmenoval ve své knize „Analytické modelování informačních systému pomocí UML v praxi“ RNDr. Ilja Kraval jako cirkulární závislost. Jde o to, že vtahy, nebo závislosti

mezi třídami jsou obousměrné, což by v ideálním diagramu tříd rozhodně být nemělo. Toto způsobuje v modelu, který je jinak dobře rozvrstven na jednotlivé balíčky značnou nepřehlednost.



Obrázek 2, Diagram balíčků s chybou
Zdroj: Vlastní

Na obrázku číslo 3. je zvýrazněná třída BrowsePanel v diagramu tříd, u které jsou vytvořeny vztahy a je zde také dokumentace, která byla získána z Java kódu. Jak je možné vidět, dokumentace je velice strohá – proto bylo nutné vytvořit si touto cestou vlastní.



Obrázek 3, Diagram tříd
Zdroj: Vlastní

Důvod proč se celý tento krok objevil v reengineeringu IS je ten, že v každé výše zmíněné metodice je jedním z prvních kroků analýza stávajícího systému. Daný systém (v našem případě pouze jádro) je nejprve nutné analyzovat, tak aby se zjistila celá jeho funkcionalita – veškeré možnosti, které poskytuje, jednotlivé vztahy, které se v něm vyskytují, tak aby při následném doplňování systému vlastními částmi nevznikaly duplicitní třídy. Protože by se mohlo stát, že z důvodu neúplné zmapovanosti systému by byla analytikem, nebo programátorem vytvořena třída, která se již ale v něm vyskytuje, ovšem nikdo o ní neví. Toto by mohlo vést k tomu, že by vznikaly duplicitní třídy a celková systémová integrace by byla zhoršena. Dále by bylo při reengineeringu dosti problematické v postupu pokračovat, když by ti, co by prováděli reengineering neměli kompletní přehled o tom co se reengineeringuje.

Navíc to jak jsem postupoval v tomto případě je z hlediska správného vývoje informačního systému špatně, ale z hlediska analýzy při reengineeringu je toto nezbytný krok.. Já jsem postupoval tak, že z již hotového kódu jsem zpětně generoval diagramy tříd a balíčků, pomocí funkce source code engineering v EA, a z nich jsem poté tvořil jednotlivé diagramy se závislostmi, jelikož už jsem měl k dispozici hotový systém, přesněji jádro systému Správně by se měl při vývoji IS udělat model v EA na základě požadavků, nebo kritérií a následně vygenerovat Java kód, který by tvořil základní kostru pro programátora, který by to poté jen dodělal. Důvod proč jsem tedy nemohl postupovat metodicky správně z hlediska vývoje IS, byl v tom, že jsem měl k dispozici již hotový zdrojový kód ze kterého jsem generoval dané diagramy.

Tento proces přinesl firmě nejen potřebnou dokumentaci, ale i celkový přehled o jádře. O jeho funkčních závislostech, všech komponentech, které obsahuje, a chybách. Díky této analýze se poté mohla společnost OR-CZ rozhodnout zda dané jádro zakoupí, nebo nezakoupí. Následně by se poté těchto znalostí využívalo při programování nových komponentů v rámci vývoje nového OR-SYSTÉMU. Navíc tato analýza systému byla prospěšná i pro firmu, která jádro vyvíjí, neboť jsme touto analýzou poukázali na několik chyb, viz výše zmíněná chyba, o kterých ani firma, která software vyvíjela, nevěděla.

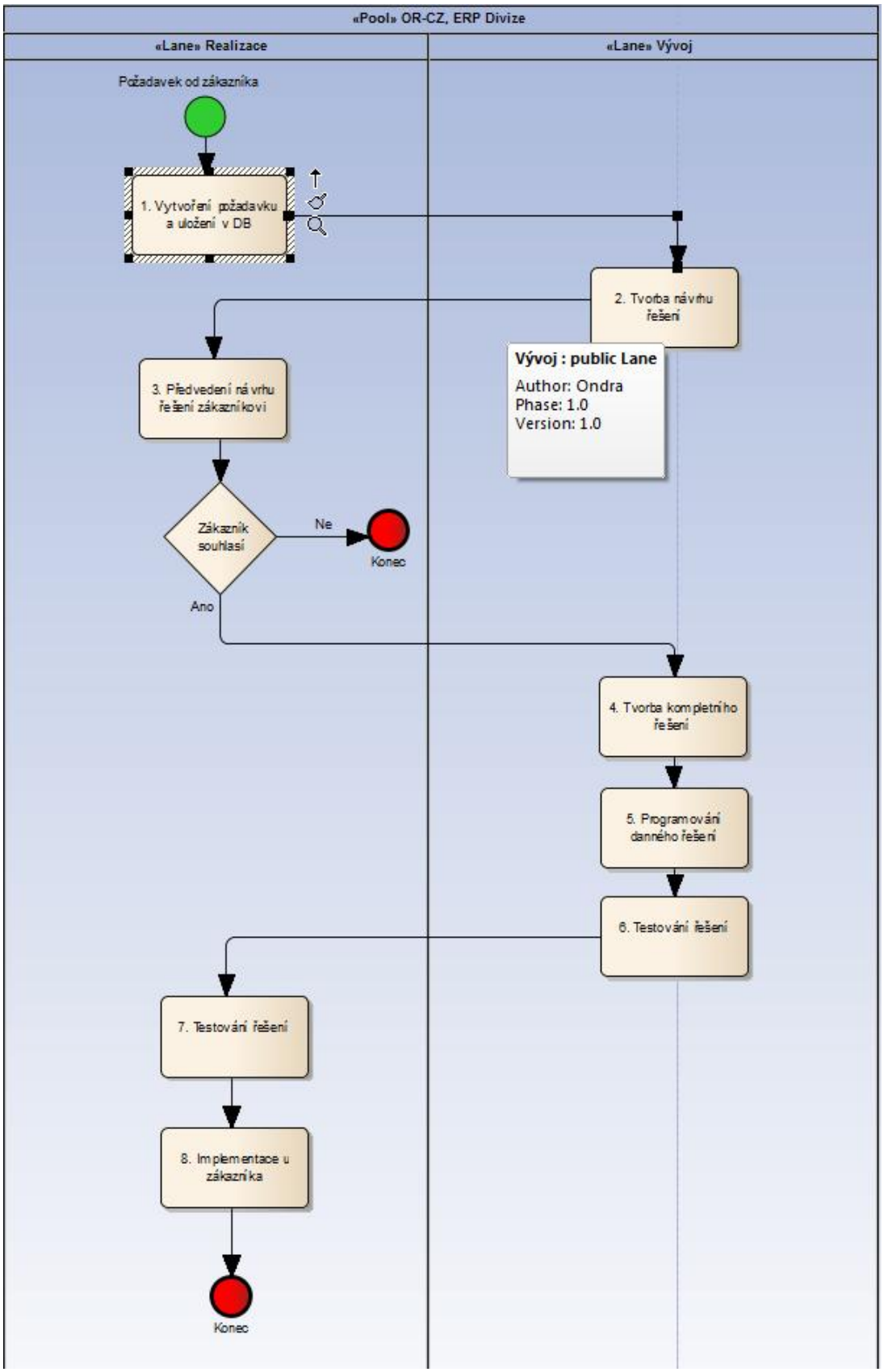
4.3 Reengineering procesu vývoje IS v ORCZ

Společnost OR-CZ vyvíjí vlastní IS. Důsledkem nákupu nového jádra, tím i přechodem na novou technologii vznikla potřeba na změnu související s procesem vývoje IS. Tato potřeba v sobě zahrnuje změnu v některých krocích v rámci procesu vývoje IS. Tato změna by se měla týkat case nástroje Enterprise Architect a jeho implementaci v některých krocích procesu.

4.3.1 Analýza stávajícího proces vývoje IS

Stávající IS OR-SYSTEM je vyvíjen divizí ERP, přesněji dvěma podúseky – podúseky realizace a vývoj. Stávající proces vývoje můžete vidět znázorněný na obrázku č. 4, na procesním diagramu.

Po přijetí požadavku od zákazníka, který je následně zaznamenán programem pro řízení životního cyklu požadavků v databázi v oddělení realizace, postupuje v rámci procesu po schválení a vytvoření úkolu na daný požadavek podúseku vývoj, kde analytik podle požadavků zákazníka zpracovává návrh řešení (č. 1. a 2. v diagramu). Tento návrh řešení se vypracovává pouze textovou formou v textovém editoru. Případná schémata a obrázky jsou vytvořeny též pouze v daném editoru. Po vyhotovení návrhu řešení je představeno úsekem realizace, u zákazníka (č. 3. v diagramu). Pokud zákazník s návrhem souhlasí, přesouvá se proces opět do úseku vývoj, kde je dané řešení podrobně rozpracováno na základě návrhu, tato část procesu se dělá opět pouze v textovém editoru (č. 4.). Poté se již podle daného řešení programují dané komponenty systému (č. 5.). Dále se již jen hotové naprogramované komponenty testují, nejdříve v úseku vývoj a poté v úseku realizace (č.6., 7.). Po úspěšném testování a odladění veškerých chyb je poté dané řešení implementováno realizátory u zákazníka, čímž celý proces vývoje končí.



Obrázek 4, Proces vývoje IS v OR-CZ
Zdroj: Vlastní

4.3.2 Důvod k reengineeringu procesu

Rozhodnutí provést reengineering stávajícího procesu, bylo hlavně z důvodu dokumentace, která se v rámci stávajícího procesu vytváří. Současná dokumentace, která se tvoří při vývoji nových částí systému, je zpracovávána pouze v textovém editoru (Word, nebo OpenOffice), a zároveň i popis nových procesů, které vznikají při vývoji nových částí systému, se popisují taktéž pouze pomocí textového editoru. Dalším problémem je, že daná dokumentace není nijak centralizovaně ukládaná. Je uložena pouze u analytika, který daný požadavek řešil a je také uložena jako přiložený dokument v programu, který slouží k řízení životního cyklu požadavků od zákazníka. Ovšem dohledatelnost v tomto programu je velice problematická, tedy je zde i problém pokud by se dané řešení muselo modifikovat. Pokud se provádí nějaká modifikace již hotového požadavku, vytváří se nová dokumentace, ve které se popisují provedené změny a je zde pouze odkaz na původní dokumentaci k danému požadavku.

Bylo tedy potřeba provést reengineering dílčích kroků v rámci procesu vývoje IS. Píši dílčích, protože některé kroky v stávajícím procesu reengineering nepotřebují, jelikož s daným problémem vůbec nesouvisí. Těmito kroky myslím kroky č. 1., 5., 6., 7. a 8., které jsou znázorněny na obrázku č. 4. Proces vývoje IS v OR-CZ. Tyto kroky totiž nesouvisí s problematikou dokumentace a jsou to spíše jen kroky, kde už se provádí „mechanická“ práce na základě vytvořené dokumentace. V kroku 5. se pouze dle vytvořené dokumentace programuje – daná dokumentace se nijak neupravuje, pouze se z ní čerpá. V krocích 6. a 7. se testuje a porovnává, zdali vše pracuje tak jak je to definováno v dokumentaci, tedy opět se dokumentace pouze čte a nijak se neupravuje.

4.3.3 Reengineering dílčích částí procesu

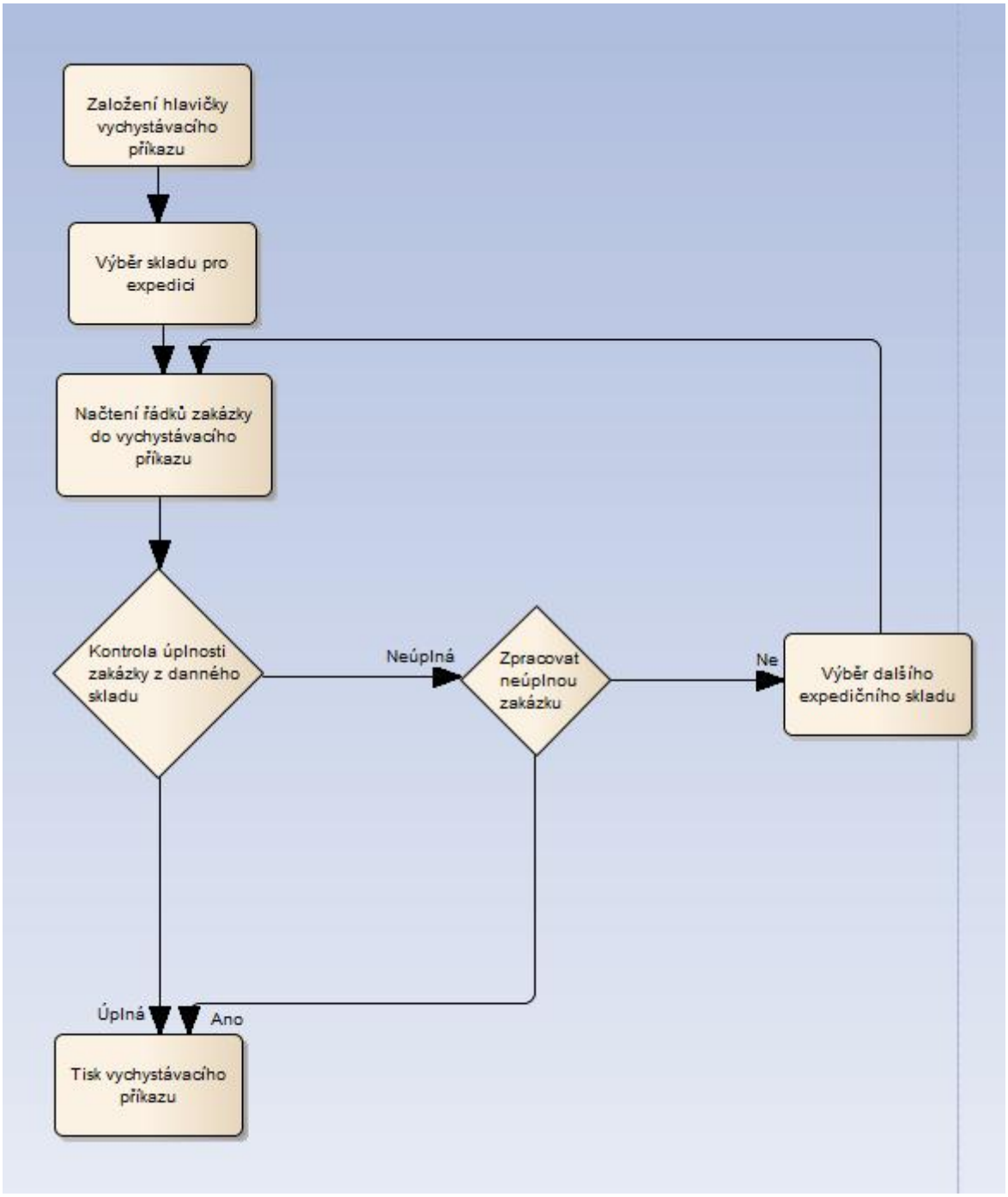
Samotného reengineeringu se tedy týkají výhradně kroky 2., 3. a 4., kde se k stávající tvorbě návrhu řešení bude používat nástroj EA.

Jak jsem se již zmínil, celý návrh řešení se tvoří pouze pomocí textového editoru. Po analýze stávajících dílčích podprocesů, při které se zjistilo, že současný způsob tvorby dokumentace je z hlediska zachování integrity popisů procesů nedostačující. Využití nástroje Enterprise Architect

4.3.3.1 Tvorba návrhu řešení a prezentace u zákazníka

Enterprise Architect se bude při tvorbě návrhu používat k tvorbě Business Process Model and Notation (BPMN) diagramů a diagramů užití.

BPMN diagramy jsou diagramy, které slouží pro grafické znázorňování procesů pomocí procesních diagramů. Jejich zapojení do tvorby návrhu řešení pomocí nástroje EA by mělo přinést standardizovaný zápis ve snadno srozumitelné podobě popisu dokumentace návrhu řešení pro všechny zainteresované osoby. Pomocí těchto diagramů se budou popisovat procesy, které v daném návrhu probíhají a následně budou všechny procesy ukládány v centralizované repository (centrální datové úložiště). Všechny diagramy, které se pomocí EA vytvoří, budou jednoduše dohledatelné pro všechny, kdo se na tvorbě návrhu podílí. Jelikož jsou jednotlivé diagramy propojené, budou se moci provádět dílčí změny, které se poté projeví ve všech dílčích závislostech, tudíž bude zajištěna aktuálnost dat. Dále se bude z diagramů generovat dokumentace ve formě PDF, nebo HTML, která bude následně použita při prezentování návrhu řešení u zákazníka (krok č. 3. na obrázku 4.). Příklad BPMN diagramu na kterém je popsaná část procesu, který se zabývá tvorbou vychystacího příkazu, můžete vidět na obrázku č. 5. Na tomto obrázku je vidět tisk vychystacího příkazu.



Obrázek 5, Ukázka BPMN - Vychystací příkaz
Zdroj: Vlastní

Další diagramy, které se budou pomocí programu EA při tvorbě návrhu dělat, jsou diagramy užití. Hlavním účelem diagramu užití je zachycení aktérů, kteří se systémem komunikují, vztahů mezi službami a těmi, kterým jsou poskytovány, a to vizuální i textovou podobou, která je srozumitelná vývojářům systému i zákazníkům. Diagramy užití

se budou tvořit až v návaznosti na hotové BPMN diagramy a budou z nich částečně vycházet, protože budou popisovat užití jednotlivých procesů, které byly popsány v BPMN. Platí u nich stejné vlastnosti jako u BPMN, tedy jednoduchá dohledatelnost, zajištěnost aktuálnosti dat a možnost tvorby dokumentace a její následné prezentování u zákazníka.

4.3.3.2 Tvorba kompletního řešení

Poslední krok, kterého se reengineering týká je tvorba kompletního řešení (krok č. 4 na obrázku č. 4.). V tomto kroku se vytvořený návrh řešení dopracuje tak aby byl kompletní a bylo možné podle něj programovat části IS. V tomto kroku se namodelují všechny BPMN diagramy a všechny diagramy užití, které patří do daného řešení a doposud nebyly namodelovány v předchozích krocích.

Dále se v tomto kroku budou tvořit diagramy tříd, které bude tvořit analytik. Tyto diagramy tříd budou sloužit k namodelování statické struktury systému a jejich vztahům. Budou obsahovat atributy, modely a jednotlivé vztahy. Tato změna souvisí se změnou programovacího jazyka a přechodem na objektově orientovaný jazyk. Kde díky struktuře, která bude vytvořena pomocí diagramu tříd, bude poté mnohem jednoduší programovat nové části systému. Zároveň bude také díky těmto diagramům celý systém zmapovaný a jakékoli změny, které se budou systému týkat, budou zaznamenány i v tomto diagramu.

Jakmile je hotové kompletní řešení, měli by být všechny vytvořené diagramy v EA uloženy na centrální repository. Tyto diagramy by poté měli být dostupné pro následné modifikace, které by se týkaly daného řešení, ale zároveň by se v nich měly projevit i změny které by vznikly při tvorbě nějakého jiného řešení. Tedy měla by být zachována konzistence dat.

4.4 Současný stav reengineeringu v OR-CZ

V současné době je v OR-CZ implementována pouze část reengineeringovaného procesu. Zatím se pomocí nástroje EA tvoří pouze BPMN diagramy oddělením realizace. Ostatní diagramy se netvoří z důvodu nedostatku lidských zdrojů a proškolenosti stávajících

zaměstnanců Zároveň se provádí analýza nového jádra, aby bylo poté možno podle hotové analýzy tvořit diagramy tříd.

4.5 Zhodnocení reengineeringu v OR-CZ

Reengineering procesu vývoje IS bude pro společnost OR-CZ výrazným přínosem. Zatím se využívá EA pouze pro tvorbu BPMN diagramů, ale postupem času bude využíván i pro tvorbu diagramů tříd a diagramů užití. Vše je limitováno časovou náročností a nedostatkem lidských zdrojů.

Tvorba BPMN diagramů společnosti OR-CZ usnadní celkovou tvorbu dokumentace k danému projektu. S pomocí těchto diagramů nebudou muset realizátoři veškeré procesy detailně popisovat v písemné dokumentaci, ale budou ji tvořit za pomoci nástroje EA. Toto zavedení tvorby BPMN by mělo zrychlit a zjednodušit tvorbu daného návrhu. Zároveň se bude tvořit „knihovna“ obecných procesů, které probíhají v OR-SYSTEMu a z které bude moci poté realizátor, který tvoří dokumentaci a BPMN diagramy k projektu čerpat. Toto opět vede k zjednodušení a zrychlení celého kroku vývoje IS. Dále se bude využívat exportovaných diagramů do HTML podoby pro jednoduchou prezentaci projektu ve firmě. Tedy opět zjednodušení a zrychlení celého kroku.

Vytvořené diagramy jsou v EA dynamicky propojené, tedy pokud se některý proces změní, projeví se změny u všech jeho závislých prvků.

Z ekonomického hlediska se ještě přesně nedá říci jak moc je zavedení EA v OR-CZ přínosné, protože nyní je vše ve fázi zavádění a nebyl ještě vytvořen žádný projekt pouze pomocí EA, jak je to v plánu do budoucnosti. Toto tedy bude známo až v blízké budoucnosti, kde bude možné porovnat časovou náročnost tvorby dokumentace projektu s a bez EA, ale nyní můžeme říci, že s EA je tvorba rychlejší, takže by se to mělo projevit ve snížení nákladů na tvorbu projektu.

Závěr

Podařilo se mi úspěšně pomocí získaných informací a znalostí o reengineeringu a case nástroji Enterprise Architect aplikovat dané vědomosti a provést reengineerig zadaného procesu.

Stávající proces vývoje byl nejprve analyzován a s ním i nově zakoupené jádro v jazyce Java, tak aby bylo možné dle zjištěných informací, o stávajícím procesu a novém jádře, provést reengineering procesu vývoje IS. Nový proces vývoje IS je přímo koncipován na míru nové technologii Java a využití nástroje EA při jeho tvorbě.

K tomu, abych docílil daného výsledku, jsem nejprve zmapoval na základě firemních informací stávající proces vývoje IS v EA. Také jsem pomocí EA získával informace o novém jádře a dále je prezentoval pomocí vytvořené dokumentace v EA. Z takto zjištěných informací jsem na základě požadavků vedení firmy provedl reengineering procesu vývoje IS, tak aby v rámci tohoto procesu byl využíván, v některých krocích, case nástroj EA. Toto zavedení docílilo k zjednodušení tvorby návrhu IS a zjednodušení tvorby dokumentace při vývoji nového projektu v oddělení realizace.

Dále by měla být v budoucnu zavedena tvorba diagramů tříd a užití pomocí EA v dalších krocích procesu vývoje v oddělení vývoj, což by mělo dále zlepšit a zkvalitnit celý proces vývoje.

Zavedení takto zreengineeringovaného procesu je náročné, nejen časově, ale i z hlediska lidských zdrojů. Proto jsem to dělal já v rámci své praxe a mohu říci, že jsem udělal jenom první krok z několika dalších kroků, které budou následovat v budoucnu. Doufám, že reengineering procesu vývoje bude pro společnost OR-CZ přínosný a další studenti, nebo zaměstnanci v OR-CZ budou moci na mou práci v budoucnosti navázat.

Seznam použité literatury

CITACE

[1] HAMMER M. A CHAMPEY J., RADIKÁLNÍ PROMĚNA FIRMY, MANIFEST REVOLUCE V PODNIKÁNÍ, 2.VYD. PRAHA: MANAGEMENT PRESS, 1996. ISBN 80-85943-30-1. 212 STRAN

[2] ŘEPA, V. PODNIKOVÉ PROCESY, PROCESNÍ RÍZENÍ A MODELOVÁNÍ. 2. VYD. PRAHA: GRADA, 2007. ISBN 978-80-247-2252-8. 281 STRAN

BIBLIOGRAFIE

ARLOW, J. A NEUSTADT, I. UML A UNIFIKOVANÝ PROCES VÝVOJE APLIKACÍ: PRŮVODCE ANALÝZOU A NÁVRHEM OBJEKTOVE ORIENTOVANÉHO SOFTWARU. BRNO: COMPUTER PRESS, 2003. ISBN 80-7226-947-X. 387 STRAN

KRAVAL, ILJA ANALYTICKÉ MODELOVÁNÍ INFORMACNÍCH SYSTÉMU POMOCÍ UML V PRAXI. 1. VYD. LIPINA: OBJECT CONSULTING, 2010. ISBN 978-80-254-6986-6. 140 STRAN

DATABÁZE ČLÁNKŮ PROQUEST

FIREMNÍ MATERIÁLY A DOKUMENTACE SPOLEČNOSTI OR-CZ, SPOL. S.R.O.