VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
ÚSTAV AUTOMATIZACE A INFORMATIKY

FACULTY OF MECHANICAL ENGINEERING
INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE

NÁVRH ALGORITMŮ PRO MODUL INFORMAČNÍHO SYSTÉMU

DESIGN OF ALGORITHM FOR INFORMATION SYSTEM MODULE

DIPLOMOVÁ PRÁCE
DIPLOMA THESIS

AUTOR PRÁCE **BC. JANA WENLICHOVÁ**
AUTHOR

VEDOUCÍ PRÁCE **DOC. ING. BRANISLAV LACKO, CSC.**
SUPERVISOR

BRNO 2008

1.strana zadaní

2.strana zadaní

Abstrakt

Diplomová práce se zabývá návrhem algoritmů pro nový modul firemního informačního systému. V úvodní části se zabývá způsoby popisu informačních systémů. Pro upřesnění popisovaného systému stručně chrakterizuje prostředí IBM Lotus Notes. Následuje objektově orientovaná analýza a návrh modulu informačního systému pomocí diagramů jazyka UML v modelovacím prostředí Enterprise Architect. Ve třetí části je provedena analýza a návrh propojení navrženého modulu se stávajícími samostatnými systémy, konkrétně aktualizace dat ve formuláři. Uvádí realizaci navržených algoritmů v prostředí Lotus Domino Designer pomocí jazyka LotusScript a SQL a Lotus Domino Connectoru pro ODBC přístup k databázi. V poslední části práce navrhuje využití nástroje pro mapování ICT infrastruktury podporující využití procesu Change management podle metodiky ITIL k efektivnímu řízení změn ve vyvíjejícím se systému.

Abstract

Master´s thesis is considered with design of algorithms for new module of company information system. In the beginning of thesis there are characterized types of ways to describe an information systems. For specification of described system is briefly defined IBM Lotus Notes environment. Next chapter is about object-oriented analysis and design of a module of information system by using UML´s diagrams in modeling tool Enterprise Architect. In the third chapter is made analysis and design of module´s connection with current system, specificly update data in form. Thesis shows designed algorithms in environment of Lotus Domino Designer by using LotusScript and SQL languages and Lotus Domino Connector for access into the database by using ODBC. In last part of thesis is proposed to use a mapping tool to mapp the ITC infrastructure by using Change management process according to ITIL method, to manage all the changes in developing system effectively.

Klíčové slova: informační systém, objektová analýza a návrh, UML, aktualizace dat podle databáze, Lotus Domino Designer, LotusScript, ITIL, Change management

Keywords: information system, object analysis and design, UML, data update according to database, Lotus Domino Designer, LotusScript, ITIL, Change management

Poděkování

Děkuji všem, kteří jakýmkoliv způsobem pomohli k vytvoření této diplomové práce a kteří mi poskytovali po celou dobu studia podporu.

Zvláště bych chtěla poděkovat svému vedoucímu diplomové práce Doc. Ing. B. Lackovi, Csc., za metodickou pomoc a cenné rady při zpracování diplomové práce a panu T. Hlavičkovi, vedoucímu oddělení IT engeneeringové firmy, za trpělivé poskytování informací a zdrojů pro mou práci.

Obsah

Seznam zkratek

BTO	Business Technology Optimization
CASE	Computer-aided Software Engineering
DBF	Database File
DDL	Data Definition Language
EA	Enterprise Architect
EFQM	European Foundatíon for Quality Management
GUI	Graphical User Interface
HTML	HyperText Markup Language
ICT	Information and Communication Technology
IDE	Integrated Development Environment
IS	Informační systém
IT	Informační technologie
ITIL	Information Technology Infrastructure Library
ODBC	Open Database Connectivity
OGC	The Office of Government Commerce
SLA	Service Level Agreement
SQL	Structured Query Language
UML	Unified Modeling Language
XMI	eXtensible Metadata Interchange

Seznam obrázků

1 Úvod

1.1 Popis informačních systémů a jeho vztah k etapě návrhu a provozování IS

Chceme-li vytvořit jakýkoliv program, musíme si nejprve ujasnit, co od něj očekáváme. Chceme-li vytvořit informační systém, musí si tato očekávání ujasnit všichni lidé, kteří jej budou používat. Můžeme to samozřejmě nechat například jen na několika zodpovědných osobách, ale pak se může stát, že uživatelé systému budou nespokojeni, protože potřebují trochu jiné funkce, než se zadavatelům zdálo. Popsat tedy informační systém znamená zainteresovat mnoho lidí, kteří preferují různé způsoby popisu a tomu všemu pak dát jednotnou podobu. Vzniklý popis navíc musí být napsán srozumitelně jak pro dodavatele, tak i pro zákazníka. Mít dobře popsaný systém znamená v prvé řadě mít dobře specifikované požadavky. Odsouhlasí-li je všichni zainteresovaní lidé, snadněji se předejde pozdějším sporům o funkce systému. Shoda původních požadavků zadavatele s funkční specifikací vyvinuté aplikace je také základem pro její ohodnocení jako kvalitního softwaru. Kvalitu aplikace nelze nějak měřit, ale tato shoda je identifikovatelný požadavek kladený na aplikaci, na základě které budou aplikaci považovat za kvalitní jak zákazník, tak i dodavatel. Jasně a úplně specifikované požadavky mohou také zabránit tomu, aby v průběhu vývojového cyklu softwaru nedošlo k postupnému odklonu od těchto požadavků. Někdy se může stát, že některé požadavky splní dodavatel jen částečně, některé vůbec, a nebo naopak že si dodavatel uměle vytvoří nějaké požadavky, které pak mohou omezit využití systému pro zákazníka (např. nepožadované automatické kontroly dat, [10]).

Teprve z těchto specifikací vlastně vznikne konkrétní zadání softwarového projektu. Všechny další kroky při vývoji aplikace pak představují v podstatě jen transformace požadavků do určitého softwarového řešení. Ze specifikací se pak v etapě návrhu vytvoří model IS, čímž vznikne jeho přehledný a podrobný popis. Tento model slouží k vytvoření funkčních algoritmů. Pokud má model strukturu, která je srozumitelná pro všechny zainteresované, usnadňuje to komunikaci nejen mezi zákazníkem a dodavatelem, ale také mezi samotnými návrháři softwaru, kteří předejdou otázkám typu: „Jak to vlastně bylo myšleno?“. Jasná syntaxe a sémantika modelu je také dobrou pomůckou při správě IS nebo při hledání možných chyb ve struktuře. Potřebujeme-li např. seznámit nového zaměstnance s naším systémem, snadněji pochopí jeho principy z modelu, na kterém uvidí, jak jednotlivé moduly na sobě závisí a jak je celá struktura vymyšlena.

Mít IS popsaný je tedy nutné nejen při návrhu systému, ale velmi výhodné i při jeho provozování. Činnost organizací je dnes závislá na podpoře informačním systémem. Počet vazeb a závislostí mezi jednotlivými moduly informačního systému roste s množstvím podporovaných firemních aktivit. Firemní procesy (Business Proceses) jsou však velmi rychle se vyvíjející oblast, a tak je nezbytně nutné, aby byl informační systém také takto flexibilní. Rychlému promítání změn do systému velmi napomáhá právě dobrý popis tohoto informačního systému. Na modelu snadněji identifikujeme vhodná místa pro implementaci změněných požadavků, a také všechny navázané procesy, kterých se budou změny týkat. Tím se sníží riziko nevhodných zásahů do systému a zlepší se rychlost a efektivnost změn a v neposlední řadě také kontrola nákladů na informační technologie. Procesním řízením a modelováním obchodních procesů se zabývá lit. [28].

1.2 Řešená problematika

Firemní informační systém, kterým se zabývá tato práce, se nachází ve fázi postupného nasazování jednotlivých modulů. Některé jsou tedy již plně nasazeny, některé jsou ve fázi testování, některé ještě ve fázi návrhu a místo nich se používají moduly předchozího systému. Společnost má dokument s názvem Koncepce informačního systému (lit. [29]), kde je k určitému datu shrnut stav celého systému. Dokument obsahuje globální vizi struktury celého systému, jsou zde heslovitě shrnuty funkce jednotlivých modulů, u některých modulů ještě doplňující požadavky na drobné úpravy a podrobnější požadavky na funkce a strukturu několika modulů ve fázi návrhu. Vznikl zde však problém zpětné specifikace hotových modulů. Jednoduše a přehledně ukázat funkci hotového modulu není možné jinak než ukázkou funkcí na reálných datech. Také specifikace požadavků předávané programátorům se již špatně dohledávají a navíc se jedná o detailní slovní popisy, jejichž procházení není nejrychlejší. Textové specifikace požadavků na nové moduly také nejsou všechny ještě kompletní, navíc se v nich promíchávají funkční požadavky s doplňkovými požadavky na vzhled jednotlivých částí. Ve většině případů chybí základní specifikace, kdo bude modul používat. Chybí zde tedy jednoduchý a rychlý přehled o funkcích jednotlivých modulů, jejich vhodný popis. Práce se tedy snaží popsat jednotlivé moduly pomocí objektově orientovaného způsobu, a to diagramy UML (Unified Modeling Language). Dále tedy v práci zpracovávám objektově orientovanou analýzu a návrh vybraného modulu firemního informačního systému na základě unifikovaného procesu vývoje aplikací (Unified Process). Jednu s knihoven pro aktualizaci dat v modulu jsem realizovala v programu Lotus Domino Designer pro systém Lotus Notes, který společnost používá a umožnila mi ho využít pro diplomovou práci.

1.3 Cíle diplomové práce

Cíle diplomové práce jsem odvodila ze zadání diplomové práce:

1. Provést objektově orientovanou analýzu požadavků na nový modul informačního systému.

2. Navrhnout základní algoritmy a datový model v jazyku UML.

3. Analyzovat přínosy nasazení nového modulu a přístupu ITIL.

Kromě těchto uložených úkolů jsem se rozhodla provést analýzu a návrh propojení navrženého modulu se stávajícími samostatnými systémy, konkrétně aktualizaci dat v jednom z jeho formulářů a provést realizaci těchto navržených algoritmů.

2 Způsoby popisu informačních systémů

Důležitost vhodného popisu programového vybavení je již zřejmá z úvodní kapitoly. Způsob popisu systému je třeba volit podle úrovně znalostí zainteresovaných lidí a také podle zvoleného podpůrného softwarového nástroje. Různé systémy CASE využívají různé vyjadřovací prostředky, nejčastěji však jejich kombinaci. Dle [7] lze rozlišit tyto způsoby popisu softwaru:

- Verbální popis

Verbální popis systému by měl být ze své podstaty lehce srozumitelný zákazníkům i dodavatelům. Kvůli CASE nástrojům se však používá spíše formalizovaného způsobu vyjadřování (většinou Structured English). Tento způsob bývá velmi přesný, ovšem lidé neznalí programovacích technik mohou mít problémy s pochopením strukturovaných výrazů a obě strany musí znát a přesně dodržovat pravidla daného formalizovaného jazyka.

- Symbolický popis

Tento způsob popisu se podobá formalizovanému verbálnímu popisu, ale vychází spíše z aparátu matematických způsobů popisu světa. Využívá např. množin, matematické logiky nebo matematických formulí. Verbálnímu popisu se tedy podobá také svými nevýhodami pro nezběhlé v aparátu matematiky.

- Tabulkový popis

Tabulkový popis je velmi rozšířený způsob popisu softwaru. Je přehledný a tabulky jsou snadno srozumitelné jak pro zákazníky, tak pro dodavatele. Bývají standardizované, každý sloupec má svůj význam a podle nich se do řádků doplňují požadované vlastnosti. Tabulky se pak snadno zpracovávají pomocí relačních databází. Tento způsob popisu je vhodnější pro méně rozsáhlé projekty, protože pak se již může stát nepřehledným.

- Grafický popis

Grafy jsou již od počátků tvorby programového vybavení nejnázornějším způsobem vyjádření algoritmů. Stejně názorně však mohou vyjadřovat také požadavky kladené na navrhovaný systém. Pokud jsou správně nakresleny, pochopení problematiky je velmi rychlé a přehledné. Některé požadavky lze však graficky vyjádřit špatně, hlavně ty nefunkční a je lépe popisovat jinou metodou.

V praxi se uvedené způsoby kombinují, aby se využilo jejich výhod, kompenzovaly nevýhody jednotlivých způsobů popisu a co nejlépe se popsal celý analyzovaný a navrhovaný automatizovaný informační systém.

2.1 Charakteristika jazyka UML

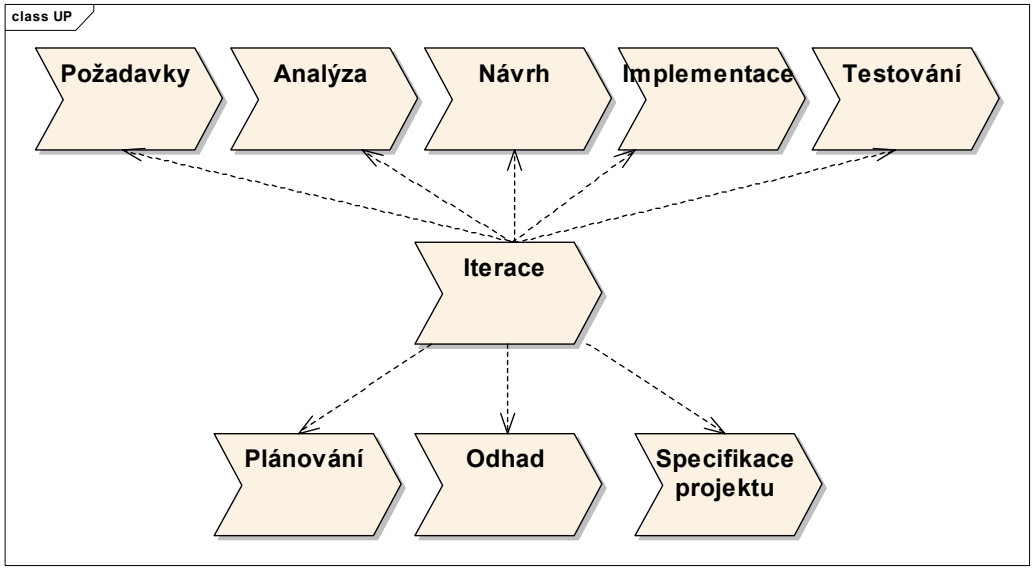
Jazyk UML (Unified Modeling Languauge) patří do kategorie grafických způsobů popisu softwarových produktů. Kombinuje však i verbální popis. Podrobný popis jazyka UML je obsažen v řadě příruček, např. v lit. [1], [5] nebo [11]. Uvedu zde proto jen jeho stručnou charakteristiku. Jedná se o objektově orientovaný přístup k vývoji systémů, podporuje všechny znaky objektově orientovaného programování:

- používá třídy a objekty
- dědičnost
- polymorfismus
- zapouzdření

Svými diagramy podporuje všechny fáze projektu, od specifikování požadavků a vstupních analýz, návrhu koncepce řešení, až po její implementaci a testování. Unifikovaný jazyk sjednotil používané výrazové prostředky pro modelování různých typů aplikací, takže vznikl jazyk s bohatou sémantikou a syntaxí. Výsledným modelům je snadnější porozumět, protože pro jejich popis již existují definovaná pravidla. Nedílnou součástí jazyka je také zabudovaný mechanismus rozšíření, který umožňuje podle potřeby definovat nové elementy jazyka. V současnosti se stal jazyk UML mezinárodním standardem pro popis analýzy a návrhu informačních systémů při použití objektově orientovaného přístupu. Proto jsem se rozhodla využít jeho prostředků v mé diplomové práci.

2.1.1 Metodika Rational Unified Process

Jazyk UML je jen jazykem, přímo však podporuje proces vývoje softwaru nazvaný Unified Process. V tomto procesu jsou uživatelské požadavky realizovány vytvořeným softwarem. Je to proces, který při vývoji definuje otázky KDO, CO, KDY a JAK. Metodika Rational Unified Process (vyvinutá firmou Rational, viz. lit. [30]) je založena na iterativním a přírůstkovém postupu, k tomu specifikuje pět hlavních aktivit:

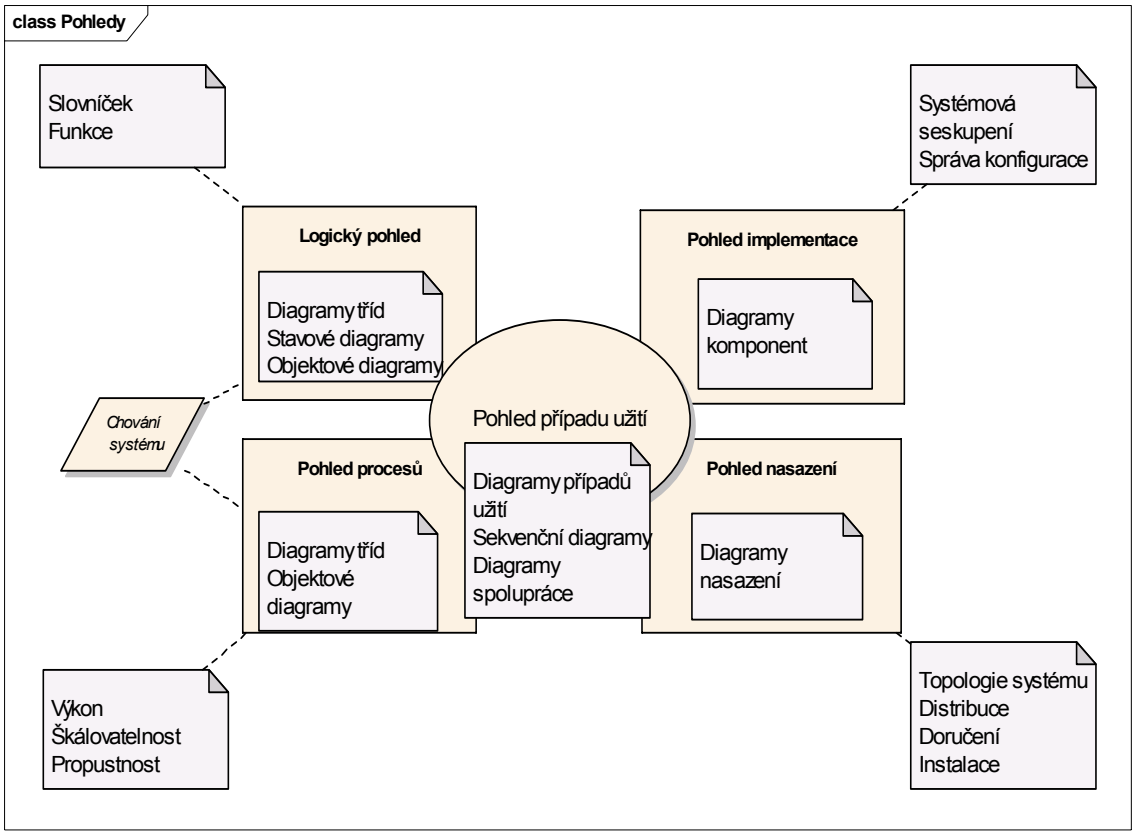


Obr. 1 Upravený obrázek aktivit Unified Process z lit. [1].

V důsledku masivní propagace metodiky RUP ze strany IBM, která zakoupila firmu Rational a rozšířila její používání z USA, se tato metodika používá v současné době i v Evropě jako standard v oblasti realizace projektů informačních systémů. Proto jsem ji využila i v mé diplomové práci.

2.1.2 UML pohledy na systém

Pohledy na systém v jazyce UML slouží k vyjádření všech strategických aspektů architektury daného systému, opět v závislosti na životním cyklu SW projektu.



Obr. 2 Upravený obrázek pohledů na systém z lit. [1].

- Pohled případů užití – jedná se o základní pohled na systém vyjadřující základní požadavky. Ostatní pohledy jsou odvozeny z případů užití.

- Logický pohled – zachycuje pojmy ze zkoumané oblasti jako množinu tříd a objektů, a zobrazením jejich chování zobrazuje chování systému.

- Pohled procesů – zobrazuje procesy, které systém vykonává jako aktivní třídy, které splňují omezení definované v případech užití. Je to procesová varianta logického modelu.

- Pohled implementace – modeluje systém jako komponenty a jejich závislosti, které pak utvářejí kódový základ systému.

- Pohled nasazení – modeluje nasazení komponent na hardwarové prostředky.

2.1.3 Stavební bloky jazyka UML

1. Předměty

   „Předměty jsou samotné elementy modelu“, jak uvádí lit. [1]. Elementy (obdélníky, šipky apod. tvořící diagram) mohou vyjadřovat:
   - Strukturní záležitosti modelu
     - jako jsou třídy, případy užití či komponenty.
   - Chování systému
     - formou zobrazení jednotlivých stavů nebo interakcí elementů a diagramů
   - Seskupovací prvky
     - balíčky sloužící k seskupování významově souvisejících elementů a diagramů
   - Poznámky
     - vhodné pro zobrazení zajímavých nebo důležitých vlastností elementu

2. Relace

   Relace vyjadřují propojení mezi předměty. Vyjadřují jejich vzájemný vztah, závislost, dědičnost a realizační postup. Základními typy relací jsou:
   - Asociace
   - Závislost
   - Zobecnění
   - Realizace

3. Diagramy

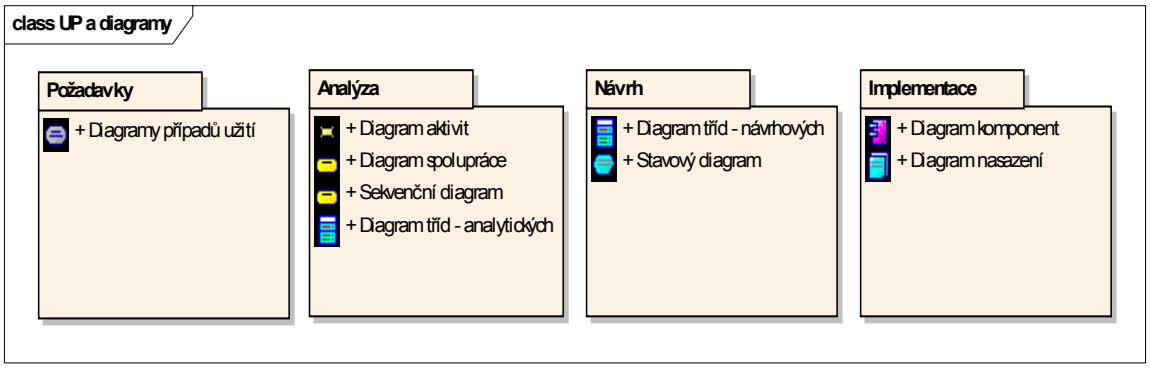
2.1.4 Diagramy UML

Z pohledů na systém vyplývá používání několika typů diagramů. Jedny jsou pro statický popis systému (systémová struktura), další pro vyjádření chování systému (dynamický model):

Statický model (systémová struktura)	**Dynamický model (chování systému)**
Diagram tříd	Objektový diagram
Diagram komponent	Diagram případu užití
Diagram nasazení	Sekvenční diagram
	Diagram spolupráce
	Stavový diagram
	Diagram aktivit

Tab. 1 Diagramy UML

V návaznosti na Unified Process se používají v jednotlivých fázích diagramy takto:



Obr. 3 Diagramy ve fázích Unified Process

2.1.5 Mechanismy UML

1. Specifikace

   – grafické diagramy poskytují model daného problému z různých pohledů, jeho sémantická specifikace jim však dává společný základ. Uvádějí se specifikace tříd, případů užití a závislostí.

2. Ozdoby

   – slouží ke zdůraznění důležitých vlastností elementů modelů, jsou to vlastně zajímavé podrobnosti (vlastnosti objektu apod.).

3. Podskupiny

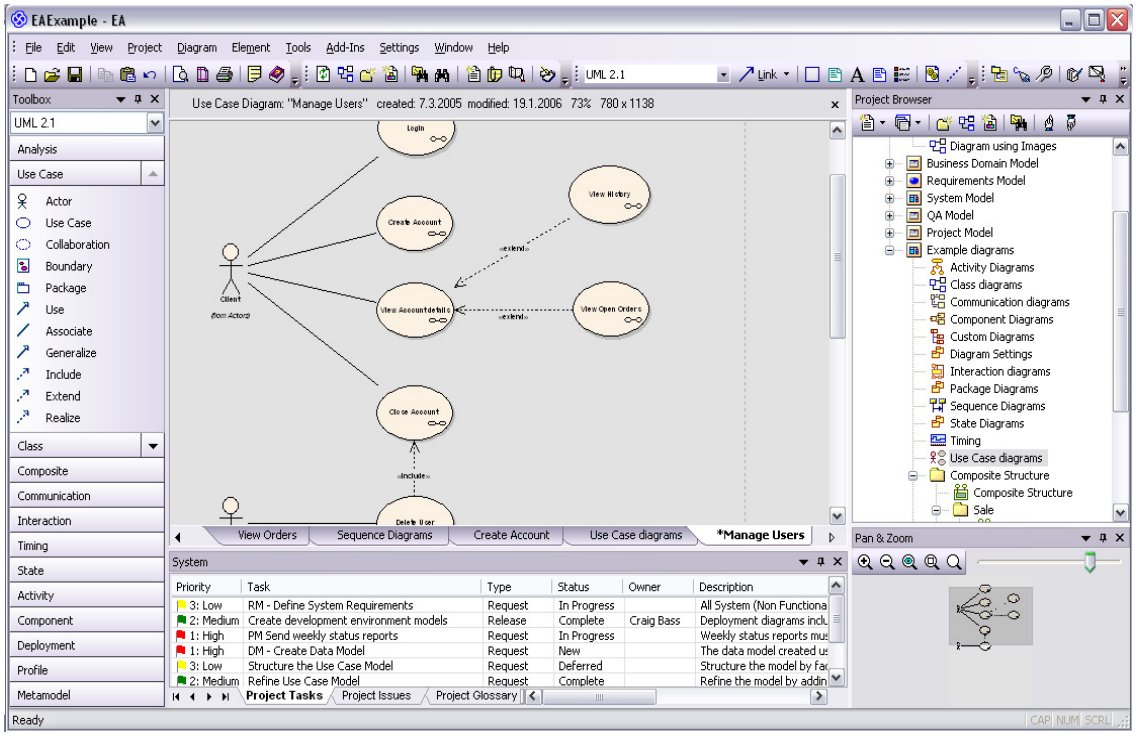
   – jazyk UML rozlišuje dvě podskupiny pohledů na systém:
   a. Klasifikátory a instance (typ předmětu a jeho konkrétní výskyt)
   b. Rozhraní a implementace (co předmět vykonává a jak to vykonává

4. Mechanismy rozšiřitelnosti

   a. Omezení (např.: {omezující_podmínka})
   b. Stereotypy (zastupují určité varianty existujícího elementu)
   c. Označené hodnoty (atributy objektů s definovanou hodnotou)

2.2 Popis použitého prostředku pro grafický popis IS v elektronické formě – Enterprise Architect

Enterprise Architect od firmy Sparx Systems je komplexní nástroj pro analýzu a návrh programového vybavení v jazyce UML. Pokrývá vývoj softwaru od shromáždění požadavků přes analytické fáze, navrhování modelů, testování a nasazení. Díky rozsáhlým doplňkům může vyzbrojit celý tým, analytiky, testery, projektové manažery, zaměstnance pro kontrolu kvality, nasazovací tým. Enterprise Architect je multi-uživatelský, na Windows založený grafický nástroj. Důležitým znakem je flexibilní dokumentační výstup. Enterprise Architect pomáhá zvládat složitost projektů pomocí nástrojů pro sledování závislostí, podporou pro velmi obsáhlé modely, kontroluje verze a zaměření na cíl v časových snímcích, obsahuje porovnávací funkce pro sledování změn v modelu, používá intuitivní rozhraní pro přehled o prvcích projektu podobné průzkumníku. Tento produkt jsem použila, protože nejlépe vyhovoval pro elektronický popis diagramů UML v mé diplomové práci.



Obr. 4 Pracovní prostředí Enterprise Architect

2.2.1 Hlavní znaky Enterprise Architect

Ze specifikací produktu uvedených na stránkách výrobce Enterprise Architect, lit. [13], a z mých zkušeností s tímto nástrojem se dají vysledovat tyto jeho rysy a vlastnosti, které uvádím podrobněji, protože zatím u nás není příliš rozšířeno používání takovýchto produktů.

- Základy EA jsou postaveny na specifikacích UML 2
  - Podporuje všech 13 diagramů a modelovacích prostředků UML:

Strukturální diagramy	Diagramy vyjadřující chování systému	Prostředky pro správu modulů
Diagram tříd	Diagram případů užití	balíčky (Packages)
Objektový diagram	Diagram spolupráce	subsystémy
Diagram komponent	Sekvenční diagram	modely
Diagram nasazení	Stavový diagram	přehled rozhraní
Diagram komponent	Diagram aktivit	
	Diagram časování	

Tab. 2 Diagramy UML podporované v Enterprise Architect

- Podporuje návrh a konstrukci případů užití, logických, dynamických a fyzických modelů

- Obsahuje zabudovanou podporu uživatelských rozšíření pro modelování procesů
  - Analytické diagramy (prosté aktivity)
  - Uživatelem definované diagramy (pro požadavky, změny)

- Umožňuje generovat kvalitní dokumentační výstup podle specifických potřeb, ať už pro vlastní tým nebo pro zákazníka
  - Kompletní MS Word kompatibilní dokumentaci se všemi diagramy podle navrhnutých propojení, jednotlivých specifikací případů užití, rozhraní atp.
  - Kompletní HTML report pro snadné a přehledné informování zákazníka o aktuálních postupech
  - Report složený pouze z vybraných diagramů pro jednoduchý základní přehled
  - Vlastní report vytvořený podle šablony upravené ve vestavěném editoru šablon

- Intuitivní uživatelské rozhraní obsahující kompletní ovládací prvky
  - Toolbox – nabízí výběr nástrojů podle zvoleného UML profilu k rozšíření modelovací domény. Kombinuje Business Procesy, informační tok a Work-flow v jednom:
    - UML 2.1
    - Business Analyst
    - Domain Analyst
    - Requirements Analyst
    - System Architect
    - Data Architect
    - SOA Architect

        - Developer
        - Real-time Engeneer
        - Tester
        - Deployment Architect
        - Project Manager
        - Meta-Modeler
        - ICONIX
    - Project Browser
    - Informace o projektu – ke snadnému sledování průběhu projektu, zadaných úkolů a problémů projektu. Umožňuje také rozdělení zdrojů a časové sledování prací. Podpora schvalování modelů zajišťuje integritu.
        - Project Tasks
        - Project Isues
        - Project Glossary

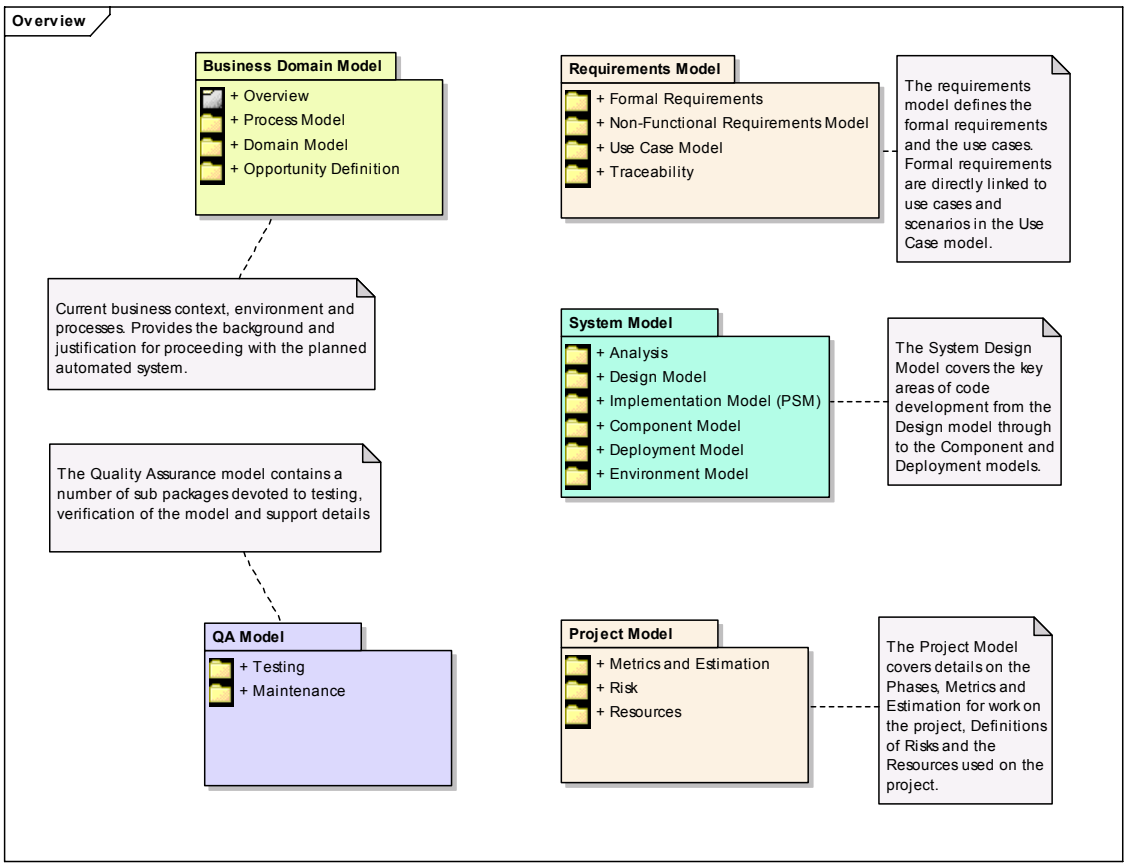
- Podporuje datové modelování
    - pokročilé databázové projektování pro DDL
    - reversní databázové projektování pro ODBC Multi-user

- Funkcemi pro generování a importování kódu podporuje přímé i reverzní programování. Díky vestavěnému editoru zdrojového kódu umožňuje rychle prozkoumat zdrojový kód modelu v jednom prostředí. Předlohy pro generování kódu umožňují upravit si uspořádání generovaného kódu podle potřeby. Podporuje:
    - ActionScript 2.0
    - Java
    - C#
    - C++
    - VB.Net
    - Delphi
    - Visual Basic
    - PHP
    - K dispozici jsou doplňky pro Python a CORBA

- Zařízení pro XMI import/export

2.2.2 Modelování v Enterprise Architect

Základem nového projektu je jeho založení. Při modelování v Enterprise Architect je třeba postupovat jako při vytváření standardního projektu. Při jeho vzniku je vhodné ujasnit si, které diagramy budou použity a nechat si od tohoto nástroje vložit do projektu předdefinované šablony. Během vývoje modelu je možné potřebné diagramy přidávat či ubírat, generovat reporty rovnou z okna Project Browseru či sledovat průběh prací v systémovém okně.

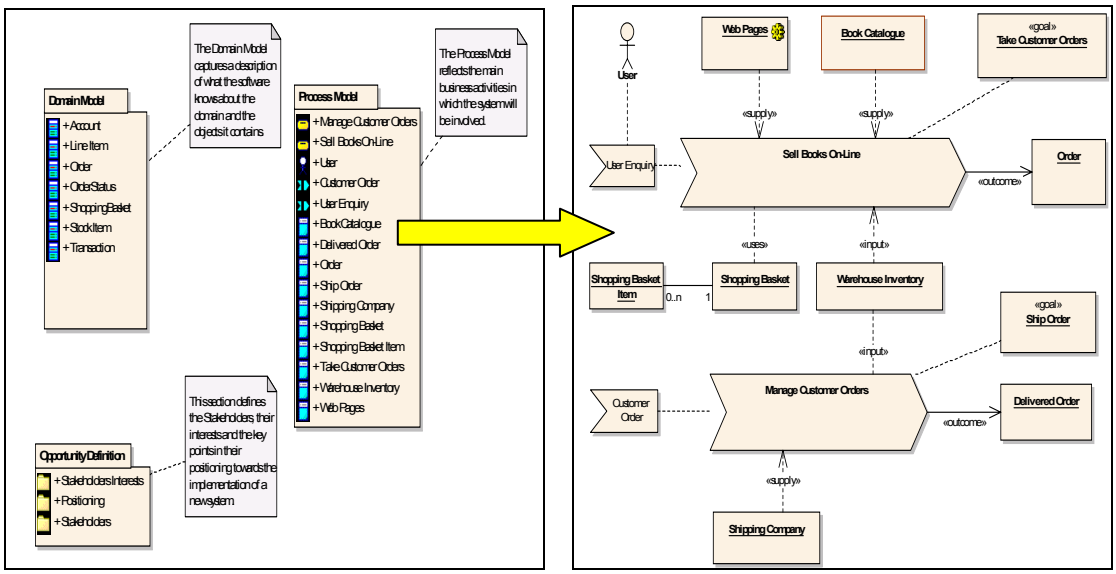


Obr. 5 Podpora různých fází vývoje softwaru v Enterprise Architect podle zabudovaného přehledu podporovaných funkcí

Model vznikajícího aplikace je složen z jednotlivých diagramů. Samotné vytváření diagramu je ve své podstatě jen vkládání elementů z Toolboxu. Přímo při vložení však program umožňuje určit konkrétní specifikace elementu, jejich vlastnosti, viditelnost v projektu nebo napojení na jiné diagramy. Také umožňuje v kterékoliv fázi vygenerovat kód (snadno přímo z okna Project Browseru) a jeho editaci.

Postupným upřesňováním nebo zobecňováním (záleží na zvoleném postupu dle konkrétního problému) vznikne komplexní model aplikace. Všechny vytvořené modely jsou uloženy v jednom projektu. Jednotlivé diagramy a pohledy na systém jsou zapouzdřeny v základních balíčcích zobrazených jako složky. Balíčky také zobrazují seznam položek, které jsou v nich obsaženy. Jednotlivé položky balíčků se zobrazí dvojklikem na balíček nebo pomocí Project Browseru. Takto si můžeme zvolit

podrobnost zobrazení jednotlivých modelů v projektu, což velmi zvyšuje přehlednost a rychlost orientace v problému.



Obr. 6 Ukázka zapouzdření diagramů do balíčků podle zabudovaného přehledu podporovaných funkcí

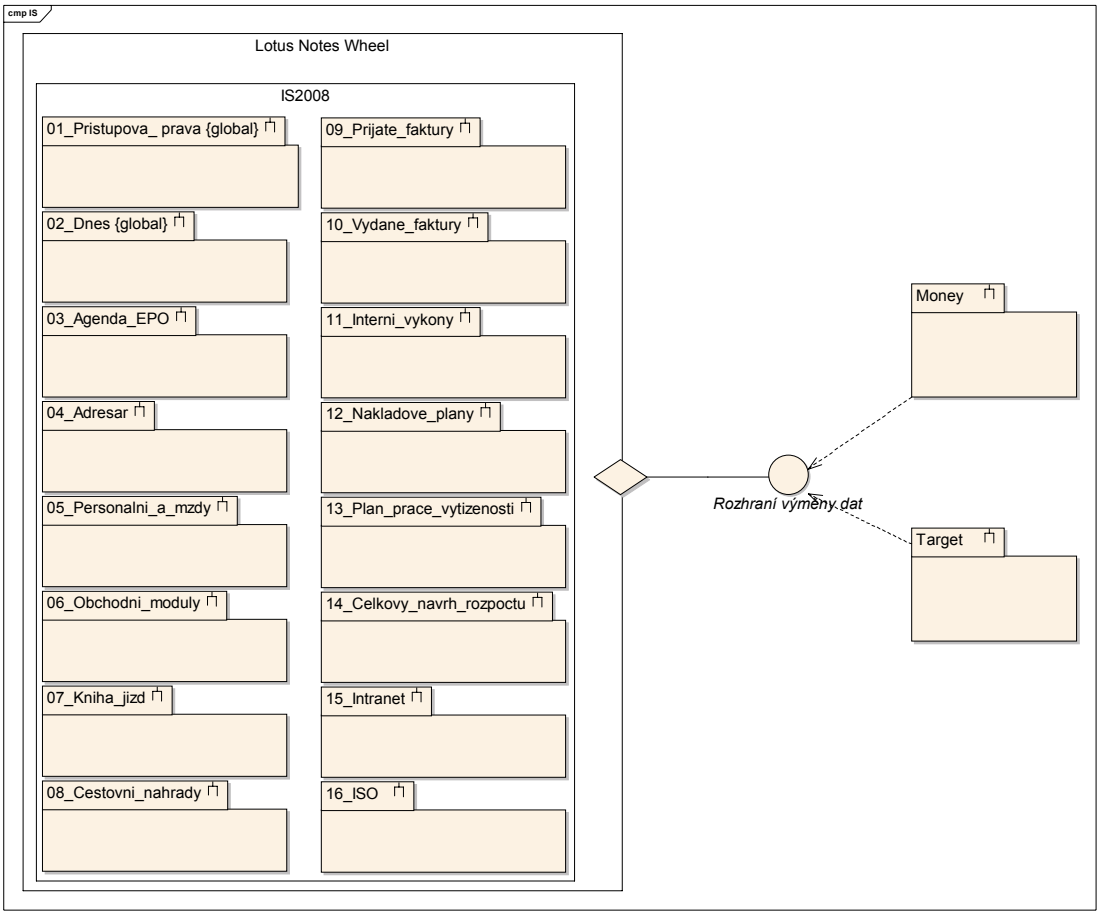
3 Popis informačního systému

3.1 Upřesnění konkrétního IS ve vybrané engineeringové firmě

Vybraná engineeringová společnost je středně velká firma zabývající se architektonickými návrhy, projektováním staveb, projektovým řízením zakázek a odbornými konzultacemi. Pro podporu svých činností využívá počítačové podpory různých specializovaných programů. Pro řízení lidských zdrojů a jednotlivých zakázek využívá dnes již starý informační systém IS2000 a další informační systémy.

Informační systém nazvaný IS2008 si klade za cíl snížit celkový počet nyní používaných informačních systémů, které běží na různých platformách, mají odlišné ovládání, grafické rozhraní atd.

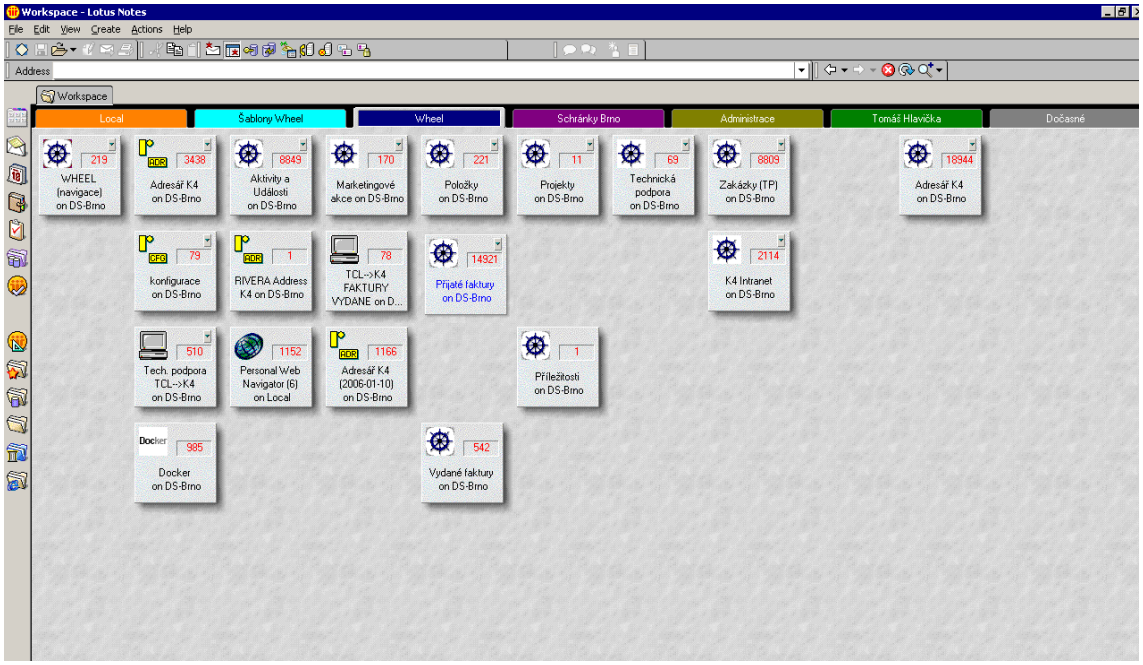
Pro konsolidaci stávajících systémů byl zvolen desktop klient IBM Lotus Notes 6.5.1. Informační systém se podle Koncepce informačního systému IS2008 má v konečné fázi skládat z 16-ti modulů. Moduly jsou vyvíjeny postupně. Nyní je na řadě modul č. 9 - Přijaté faktury. Celý informační systém využívá v maximální podobě silných stránek systému IBM Lotus. Využívá distribuovaný systém (replikace mezi servery i se vzdálenou kanceláří), poštovní schránky, kalendář, úkoly, systémový adresář, distribuční skupiny, skupiny přístupových práv, rezervace sdílených prostředků atd. IS dále využívá data s dalších samostatných systémů.



Obr. 7 Moduly IS2008

3.2 Lotus Notes

Podle lit. [14] je Lotus Notes určený pro zpracování zpráv a spolupráci na podnikové úrovni. Zabudované schopnosti Lotus Notes zahrnují elektronickou poštu s integrovanou podporou okamžitých zpráv, kalendářem a plánováním úkolů, referenční databází a nástroji personálního informačního řízení. Lotus Notes podporuje zároveň zabezpečení přístupu. Při použití funkce pro řízení provádění programů může zabránit spuštění neautorizovaných skriptů, kódů a vzorců na pracovní stanici. Víceúrovňové možnosti zabezpečení umožňují řídit přístupová práva od úrovně serveru až k jednotlivým polím na formulářích.



Obr. 8 Lotus Notes Workspace

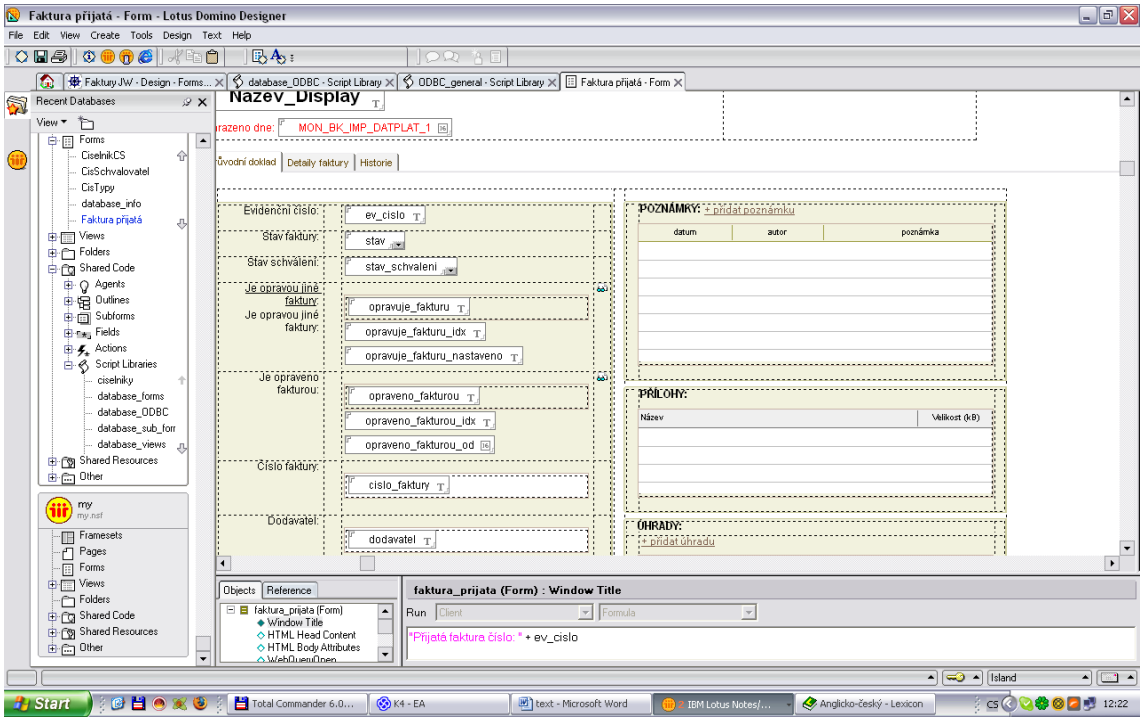
3.3 Lotus Domino Designer

Domino Designer 6.5.1 je integrované vývojové prostředí aplikací.

- Umožňuje rychlé navrhování vzhledu aplikace pomocí položky Framesets v jednoduchém GUI.
- Pro zobrazování informací v aplikaci používá nastavení stránek (Pages), pro editování informací používá nastavení Forms. Změna dat ve formuláři způsobí také změnu dat v dokumentu z databáze, který Form zobrazuje.
- Jako vstupní bod do databáze se používají definované Views, které zobrazují uložení dokumentů v databázi. Většina databází má více pohledů.
- K uložení souvisejících dokumentů slouží Folders, které mají stejné položky jako Views.

Pro obsluhu navržených polí a pohledů se používají položky Shared Code:

- Agents – k obsluze uživatelem aktivovaných složitějších úkolů pro přístup a správu databáze
- Outlines – slouží k návrhu navigace pro uživatele, pro jeho snadný pohyb v aplikaci
- Subforms – jsou kolekce formulářů, které jsou uloženy jako samostatné objekty. Při změně položky v Subforms se změní všechny dané položky ve všech příslušných Forms.
- Fields – pole se vytvářejí na formulářích a slouží ke sběru dat. Každé pole ukládá jeden definovaný typ informace. Po uložení formuláře se data z polí uloží v samostatném dokumentu. Obsah polí je tak možno použít v dalších dokumentech a pohledech nebo upravovat pomocí formulí.
- Actions – představují tlačítka na formulářích, pohledech a stránkách a zajišťují obsluhu rutinních úkolů v pohledu nebo v dokumentu. Akce mohou být součástí návrhu elementu nebo sdílené uložené jako Shared resources v databázi.
- Script Libraries – jsou používány Agenty, kteří komunikují s databázemi. ScriptLibraries jsou součástí databáze. Mohou obsahovat LotusScripts, programy v Java nebo Java Sripty.



Obr. 9 Návrh formulářů a polí v prostředí Lotus Domino Designer

3.3.1 Lotus Domino Connectors

Lotus Domino Connectors poskytují přirozený přístup k různým DBMS systémům, ODBC, platformám pro správu souborů, k systémům pro Enterprise Resource Planning a Transaction Processing. Rozšíření LotusScriptu pro Lotus Connectors (LC LSX) poskytuje napojení těchto přístupů do psaných LotusScriptů. Zajišťuje interpretaci jejich příkazů mimo prostředí Notes. Programovací model je nezávislý na jednotlivých konektorech. To eliminuje nutnost učit se specifikace různých systémů a umožňuje přístup k jednotlivým položkám specifického systému. Např. díky Lotus Connectors mohou aplikace pracovat s externími daty formulářů bez použití definice agentů a zároveň je možno s dokumentem pracovat v jiné aplikaci.

Dostupná verze nástrojů rozšíření (LC LSX Toolkit) podporuje přístup k následujícím databázovým aplikacím:

- Notes
- DB2
- File System
- ODBC
- Oracle 7
- Oracle 8
- OLE DB
- Sybase

3.3.2 LotusScript

LotusScript je nedílnou součástí programovacího rozhraní do Lotus Notes v programu Lotus Domino Designer. LotusScript obecně a jeho nástroje poskytují běžné programovací prostředí pro Lotus aplikace na všech platformách podporovaných softwarem Lotus (Windows, AIX, Linux). LotusScript je založený na jazyku BASIC scripting language, obsahuje řadu rozšíření, které umožňují programování i objektově orientovaných aplikací.

Třída se definuje pomocí Class *className* [As *baseClass*], následuje tělo třídy a nakonec příkaz End Class. Integrované uživatelské prostředí (IDE) pro každou třídu většinou automaticky definuje specializované procedury (subs):

- Sub Initialize – umožňuje provést operace nastavení při zavolání třídy, je vždy typu Private
- Sub Terminate – umožňuje provést operace „úklidu“ při opouštění třídy
- Sub New – umožňuje provést definované operace při zavolání objektu třídy
- Sub Delete – umožňuje provést operace „úklidu“ při mazání objektu třídy

V těle funkce se mohou nacházet další funkce. Jsou to pojmenované procedury, které vrací jednu hodnotu. LotusScript poskytuje soubor zabudovaných funkcí a umožňuje také vytvořit funkce nové. Signatura funkce specifikuje jméno procedury, její prostor, typ hodnot, které očekává že bude aplikace používat, dobu existence těchto hodnot a typ návratové hodnoty. Definice tvořící tělo funkce může zahrnovat:

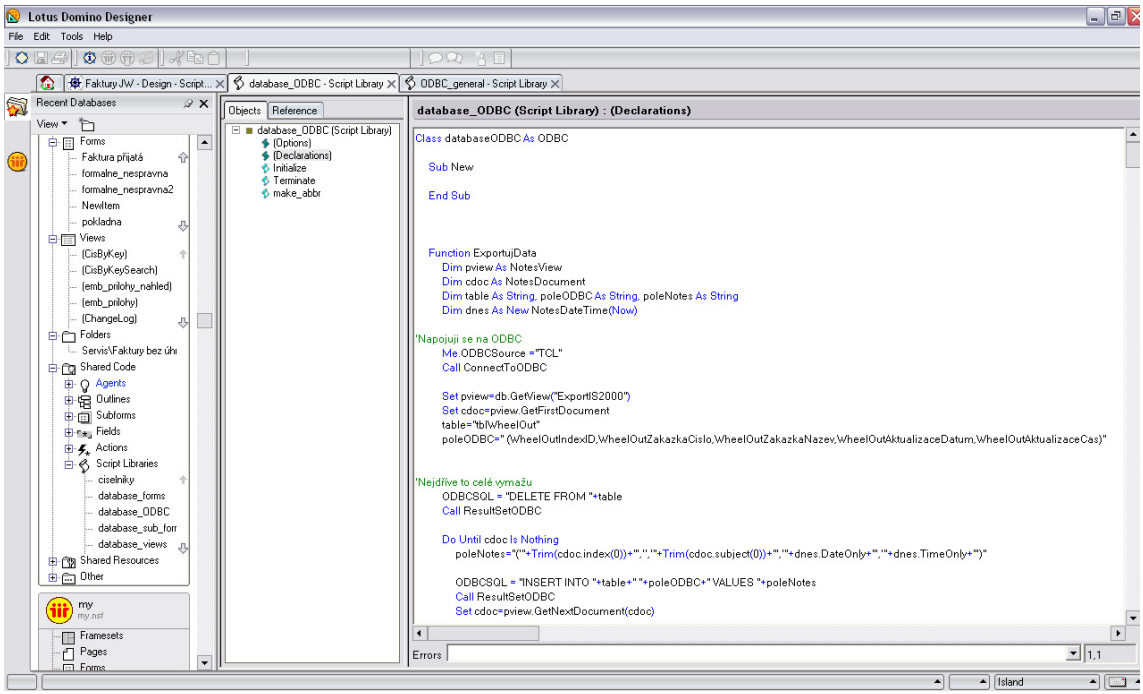
- deklaraci proměnných
  - Dim *variableDeclaration* As *type*

- přiřazovací příkazy
  - Set *var1* = *var2*
  - Set *var* = New *class* [*argList*]

- volání zabudovaných i nově definovaných funkcí (zahrnuje možnost volání sama sebe)
  - Call *subOrFunction* [([argList])]
  - *subOrFunction* (argPassedByVal)
  - *returnVal* = *function* [([argList])]

- funkce cyklů a větvení (zahrnuje příkaz Exit Function pro ukončení funkce dřív, než je dosaženo jejího konce)
  - cykly typu For, ForAll, Do, While

- funkce pro standardní práci se soubory a pro komunikaci s koncovým uživatelem
  - funkce Open, Close, Get, Kill



Obr. 10 LotusScript v Lotus Domino Designer

Více informací o Domino Designer, Lotus Connections a LotusScript je v lit. [15] a v nápovědě k programu.

4 Návrh modulu informačního systému

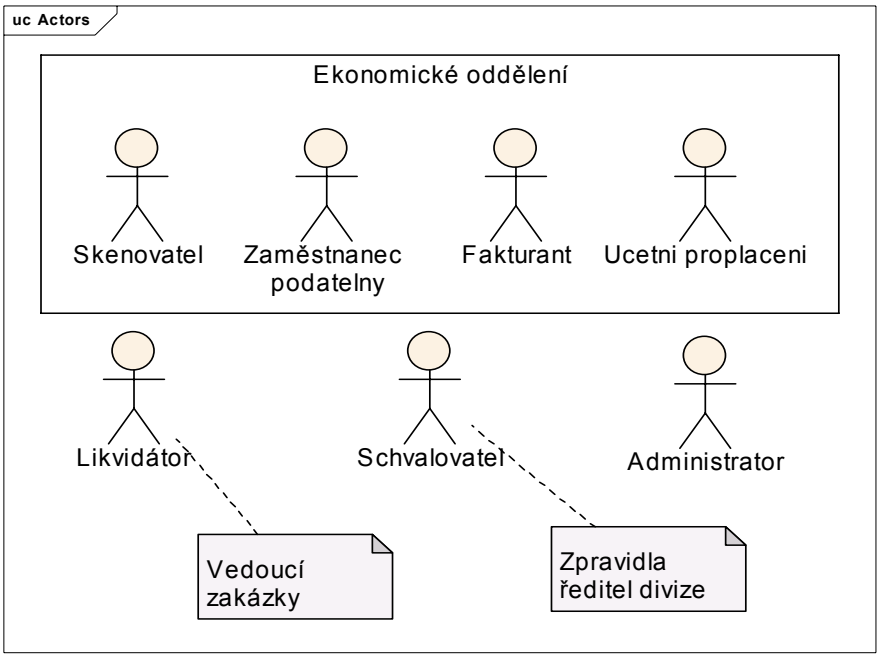
4.1 Charakteristika popisovaného modulu informačního systému

Nový modul informačního systému, který firma potřebuje doplnit do stávajícího informačního systému, se jmenuje Přijaté faktury a zajišťuje oběh přijatých faktur přes všechny zúčastněné osoby víceúrovňového schvalovacího systému. Fakturu musí schválit přiřazený likvidátor. Ten se stará o zakázku, které se faktura týká. Dále musí faktura projít přes fakturanta, který vloží fakturu do účetního systému Money. Po naplnění Detailů faktury podle Money ji musí schválit k úhradě přiřazený schvalovatel a pak teprve následují činnosti pro její uhrazení.

Pro přijatou fakturu je definováno 9 stavů:

Stav č. 0: Skenovatel
Stav č. 1: Podatelna
Stav č. 2: Odeslaná na likvidátora
Stav č. 3: Odeslaná na fakturanta
Stav č. 4: Čekající na import dat
Stav č. 5: Odesláno ke schválení úhrady
Stav č. 6. Schváleno
Stav č. 7. Odesláno k úhradě
Stav č. 8. Uhrazeno
Stav č. 9. Vyřazeno

Uživatelské rozhraní modulu ale bude podporovat jen akce uživatelů, import dat budou zajišťovat příkazy na serverech, přes které data procházejí. Pro proces uživatelské správy přijatých faktur je tedy možno definovat tyto aktory, zaměstnance firmy, kteří budou s novým modulem pracovat:



Obr. 11 Aktoři modulu Přijaté faktury

4.2 Přehled požadovaných funkcí

Požadavky kladené na systém, které jsem analyzovala průzkumem u pracovníků ve firmě, se dají rozdělit na funkční a nefunkční. Funkční požadavky určují pouze logické funkce systému, nefunkční požadavky zahrnují návrhy na vzhled obrazovky a uspořádání formulářů. Ve svém návrhu jsem se zaměřila na funkční požadavky na modul, ve verbálním popisu jsem se však kvůli vysvětlení návaznosti na ostatní akce kolem přijatých faktur nevyhnula popisu některých nefunkčních požadavků, hlavně některých pracovních postupů.

4.2.1 Verbální popis

Požadavky na funkce modulu Přijaté faktury vyplývají z možných stavů faktury. Jedná se o kontrolu a vyplňování položek formulářů pro fakturu, jejich kontrolu a editaci a přepínání mezi stavy. Proto jsem zvolila pro verbální popis funkcí přehled akcí v jednotlivých stavech. Po konzultacích s vybranými zainteresovanými zaměstnanci firmy, ale hlavně z firemního dokumentu Koncepce IS vyplynuly tyto požadavky:

- Stav č. 0: Skenovatel

Tento stav vyjadřuje úvodní proceduru s přijatou fakturou, kdy skenovatel na recepci v Brně nebo Praze naskenuje přijatou papírovou fakturu do elektronické podoby. Modul má podporovat založení nové přijaté faktury, přidání přílohy (naskenovaného originálu) a zobrazení přehledu naskenovaných faktur pro skenovatele. Skenovatel musí jednotlivé připravené faktury přepnout do následujícího stavu, tzn. postoupit na podatelnu.

- Stav č. 1: Podatelna

Podatelna je vlastně recepce v Brněnském sídle firmy. Na podatelně dostávají faktury, které se sem schází z různých pořizovacích míst jednotné evidenční číslo. V případě, že se jedná o zakázkovou fakturu, je třeba dohledat odpovídající zodpovědné osoby (likvidátora a schvalovatele). V případě, že se jedná o tzv. režijní fakturu má recepční vybrat likvidátora ze seznamu všech možných likvidátorů. Pokud se podařily předchozí akce, musí recepční postoupit fakturu na likvidátora. Je-li na faktuře něco špatně vyplněno (nesprávné ID, fakturační adresa), musí recepční fakturu označit jako formálně nesprávnou, pokud je faktura vůbec špatná nebo patří někam jinam, je třeba ji vyřadit z evidence. Podatelna si může zobrazit přehled všech přijatých faktur ve všech stavech, nemá však právo editace faktur, které nejsou ve stavu 1.

- Stav č. 2: Odeslaná na likvidátora

Přijatá faktura v tomto stavu čeká na likvidaci od likvidátora, tzn. na schválení její částky a účelu. Jakmile je zlikvidována, hned přeskočí do stavu č. 3. To proběhne akce likvidátora Schválení bez výhrad nebo Schválení s výhradou. Výhrady je možno zapsat do speciálního pole. Likvidátor má možnost také vrátit fakturu zpět do stavu č. 1 na podatelnu. Může si zobrazit seznam faktur, které jsou určeny k likvidaci právě jeho osobou.

- Stav č. 3: Odeslaná na fakturanta

Fakturant podle naskenovaného originálu vloží fakturu do účetního programu. Do vyhrazeného pole (v účetním programu) rovněž vloží párovací symbol, který

odkazuje na evidenční číslo (např. 08-0000237) z modulu Přijatých faktur. Modul však podporuje j

en tyto akce fakturanta: Odeslat zpět na likvidátora a Odeslat na schvalovatele. Když fakturant zvolí Odeslat na schvalovatele, přejde faktura do stavu č. 4, což ještě není stav pro samotného schvalovatele. Nejprve faktura čeká až proběhne import dat z účetního programu, čímž se doplní a případně aktualizují položky formulářů faktury.

- Stav č. 4: Čekající na import dat

V pravidelných intervalech probíhá jednosměrný export dat (mezi nimi i přijatých faktur) z účetního systému a rovněž tak probíhá v pravidelných intervalech import dat do systému Lotus (modulu Přijaté faktury). Párování položek probíhá na základě vloženého párovacího symbolu. V případě korektního importu dat dojde k naplnění karty „Detaily faktury“. Tímto importem se zabývá kapitola 6. Jakmile import proběhne, přepnou se faktury ve stavu č. 4 do stavu následujícího.

- Stav č. 5: Odesláno ke schválení úhrady

Přijatá faktura čeká na vyřízení od schvalovatele. Schvalovatel je obvykle vedoucí divize, šéf oddělení. Schvalovatel si může zobrazit seznam faktur, které jsou určeny ke schválení právě jeho osobou. Schvalovatel může schválit fakturu buď k plné úhradě nebo k částečné úhradě. V případě, že schvalovatel schválí fakturu k úhradě s částkou 0 Kč (ať se jedná o částečnou nebo plnou úhradou), pak je faktura pokládána za zaplacenou v hotovosti, šekem, zálohou atp. Příslušná informace se objeví zpravidla v poznámce. Takové faktury by se neměly nabízet jako Čekající na úhradu, nejdou přes proplatitele, ale jdou rovnou do stavu č. 7.

- Stav č. 6. Schváleno

Faktury v tomto stavu čekají na odeslání do banky osobou zodpovědnou za proplácení. Měly by se dát zobrazí v pohledu nastaveném na zobrazení informací o fakturách čekajících na úhradu. Osoba zodpovědná za proplácení má po provedení platby mít možnost přesunout vybrané faktury do „dnešní úhrady“. Faktury tak přechází do stavu č. 7.

- Stav č. 7. Odesláno k úhradě

Ve vedené Historii úhrad u aktuálního dne úhrady osoba zodpovědná za proplácení má mít možnost provést pomocí tlačítka Export vybraných faktur do Gemini, což vyexportuje faktury do formátu kompatibilního s softwarem pro internetové bankovnictví. Následně faktury čekají na to, až budou spárovány s platbou, která se objeví ve výpisu z banky a projde přes Money. Pak se faktura přepne do dalšího stavu.

- Stav č. 8. Uhrazeno

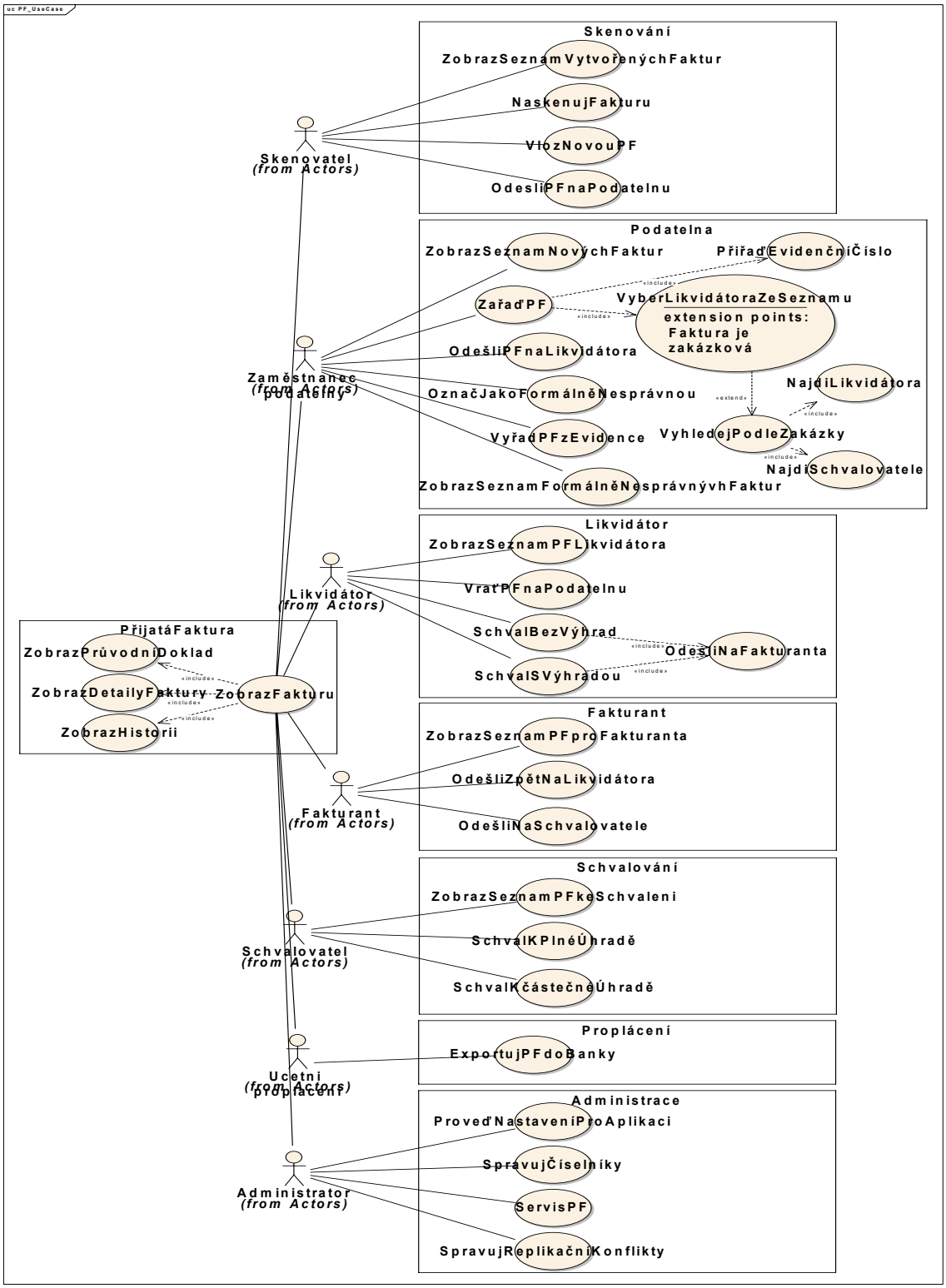
Faktury v tomto stavu jsou již uhrazeny. Faktury, které mají nastavenu částečnou úhradu by se měly opět vrátit schvalovateli do stavu Odesláno ke schválení úhrady. U takových faktur se liší celková částka a uhrazená, tj. zbývá uhradit je i po spárování s bankou nenulové.

- Stav č. 9. Vyřazeno

Tento stav slouží jako koš na faktury, které vůbec nepatří do systému a u kterých se nečeká na opravu (jako u Formálně nesprávných). Sem patří faktury, které vůbec nejsou adresovány této firmě.

4.2.2 Grafický popis Use Case diagramem v jazyce UML

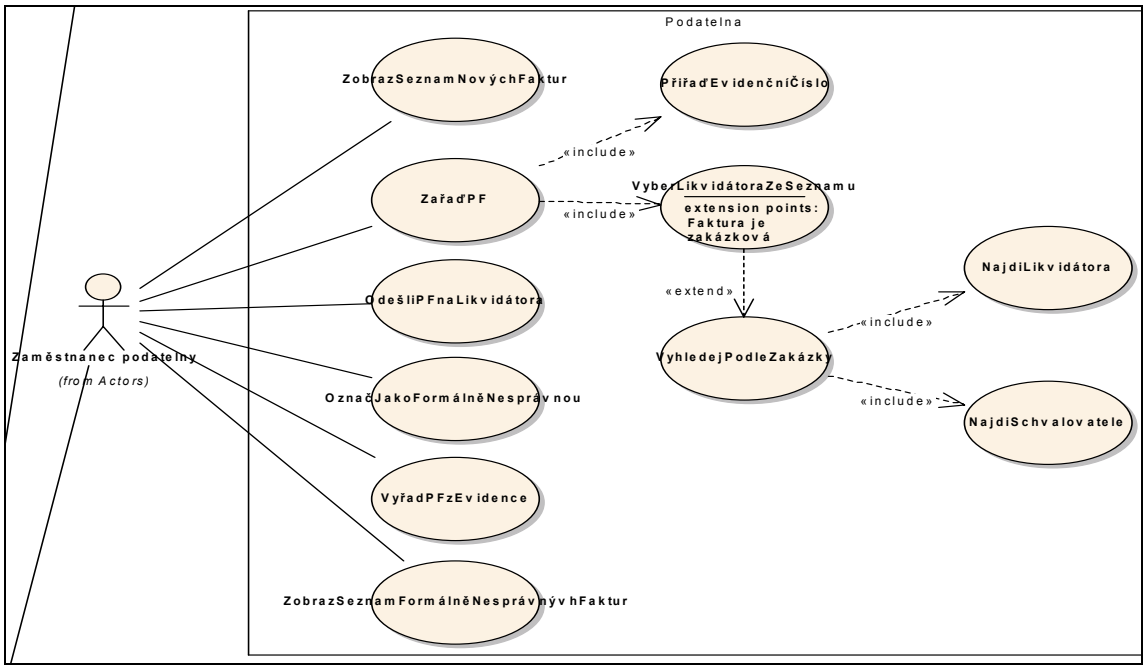
Popisované funkce jsou daleko přehledněji zobrazeny v Use Case diagramu.



Obr. 12 Upravený Use Case diagram modulu Přijaté faktury

Jedná se o diagram případů užití, tzn. kdy (v jakém případě) navrhovaný systém kdo (který aktor) potřebuje. Diagram přehledně zobrazuje jednotlivé případy v elipsách, které jsou spojeny s příslušnými aktory. Obr. 11 je upravený pro čitelné zobrazení, přehledný diagram požadavků na modul je zobrazen v příloze č. 1.

Pro vysvětlení jeho principu jsem zvolila část diagramu zahrnující akce pro podatelnu. Aktor Zaměstnanec podatelny použije systém pro případ ZobrazeníNovýchFaktur. Případ ZařaďPF vždy při svém provádění použije jiné případy užití, jako je PřiřaďEvidenčníČíslo nebo VyberLikvidátora ze seznamu. Znamená to že zahrnuje další případy užití, použila jsem tedy relaci <<include>>. Pokud se jedná o zakázkovou fakturu, bod rozšíření Faktura je zakázková nabude platnosti. Pak případ užití použije ještě jeden případ, a to VyhledejPodleZakázky.



Obr. 13 Část Use Case diagramu pro Přijaté faktury – akce podatelny

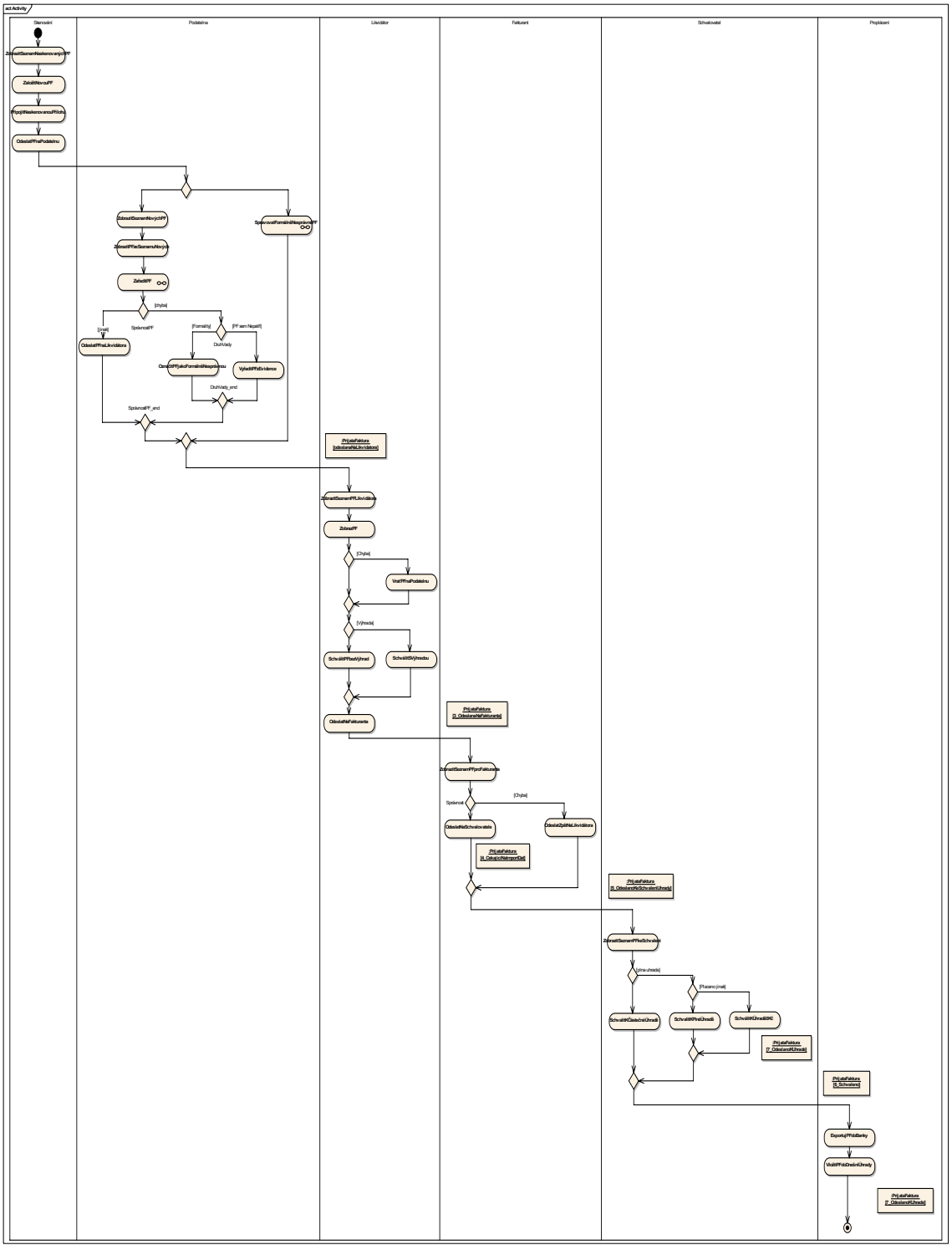
4.3 Návrh koncepce řešení modulu

4.3.1 Diagram aktivit

Podle lit. [1] jsou diagramy aktivit objektově orientovanými diagramy toků. Díky nim lze modelovat proces jako seznam aktivit a přechodů mezi nimi. Aktivity jsou zobrazeny jako obdélníky se zaoblenými hranami a jsou řazeny pod sebe podle časového sledu. Tok událostí je v určitých místech rozdělen rozhodovacími body, aby se ve slučovacím bodě zase sešel. V diagramu jsou také pro úplnost zobrazeny některé stavy objektu Přijatá faktura (obdélníky s podtrženým názvem). Určité soubory aktivit lze logicky ohraničit, v diagramu tak vznikly jednotlivé zóny zobrazené jako pojmenované sloupce.

Diagram aktivit je podle metodiky Unified Process ještě součástí analytické části vývoje aplikace. Vzhledem k tomu, že je ale vlastně speciálním případem stavového diagramu návrhové části procesu a protože je již navržen dostatečně podrobně, rozhodla jsem se jej použít i pro část návrhu. Výsledný diagram aktivit vznikl postupným upřesňováním požadavků definovaných v diagramu případů užití a následnými

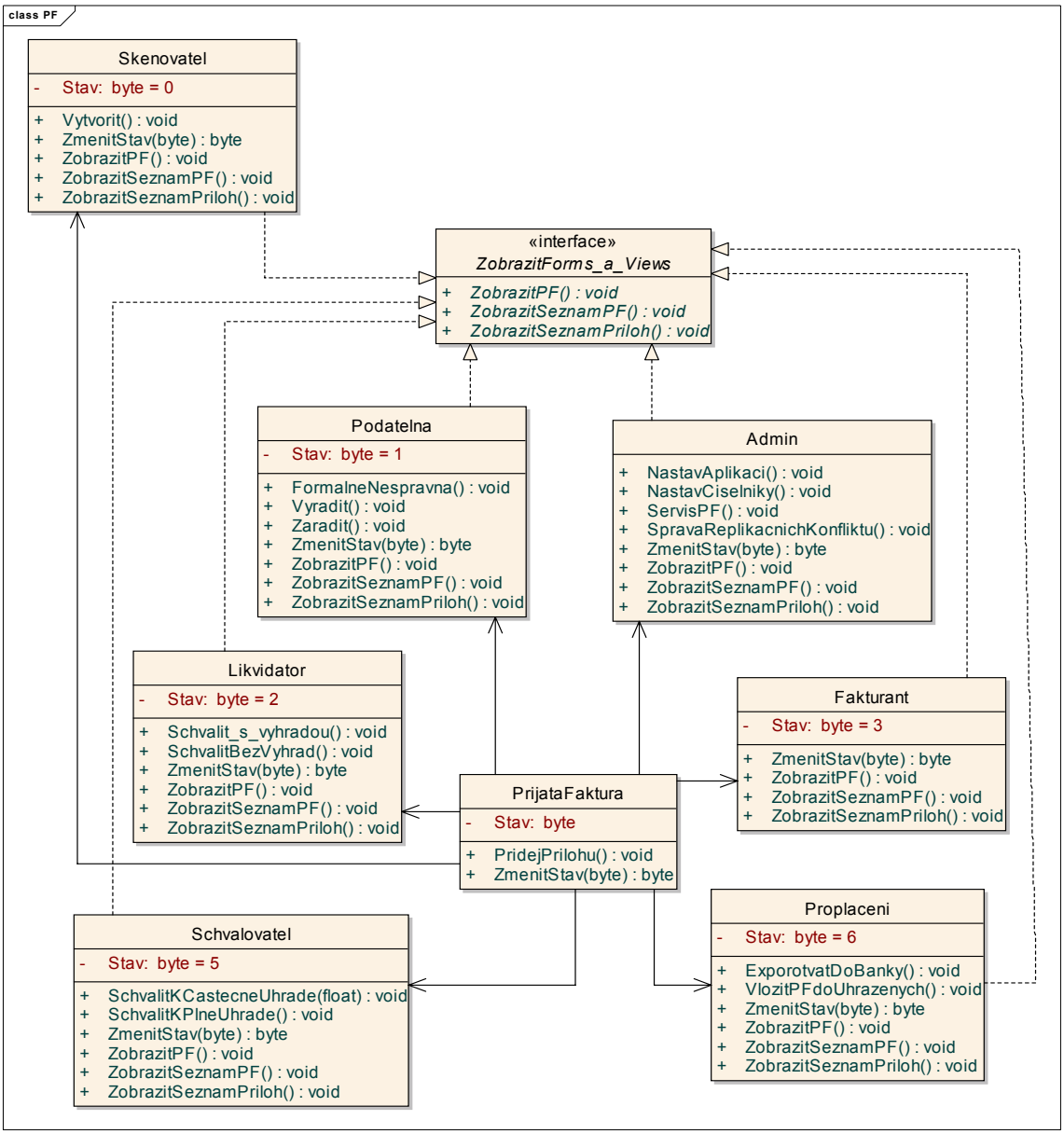
konzultacemi se zástupci firmy na funkčnost modulu. Celý diagram je velmi rozsáhlý, proto je zobrazen samostatně v příloze č. 2. Zde uvádím jen náhled na celý diagram.



Obr. 14 Náhled na diagram aktivit modulu Přijaté faktury

4.3.2 Diagram tříd

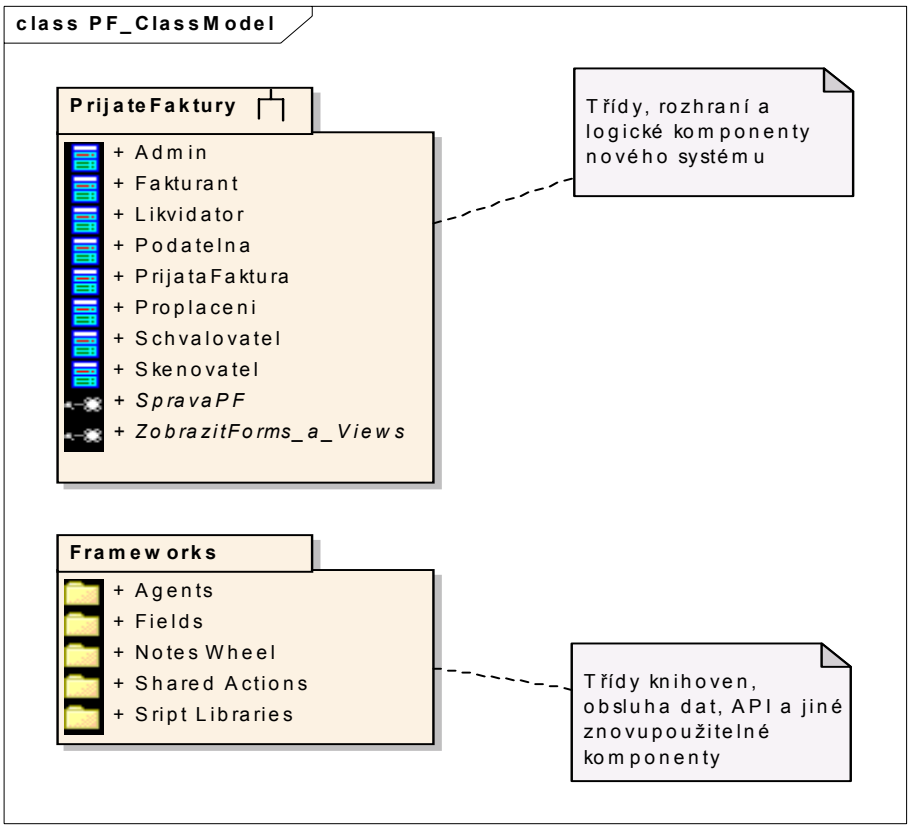
Diagram tříd zobrazuje statický pohled na systém, jeho strukturu. Zobrazuje třídy jako typy objektů, obsahy tříd a statické vztahy, které mezi nimi existují. Z verbálního popisu požadavků na systém lze analýzou podstatných jmen a sloves rozpoznat základní návrhové třídy pro systém. Třídy jsem zvolila s ohledem na použité programovací a uživatelské prostředí, systém Lotus Domino Designer a Lotus Notes a jejich specifické vlastnosti. Základní třídou je PrijataFaktura s atributem Stav a hlavní metodou ZmenitStav. Ta je používána všemi ostatními třídami. Všechny třídy také používají metodu ZobrazPF a metody zobrazení seznamů. Kvůli nastavení přístupových práv bude vytvořen pro každého aktora vlastní formulář faktury. Obsluhu tohoto zobrazení a naplnění obecných dat ve formulářích zajišťuje rozhraní ZobrazitForms_a_Views, tak jak předpokládá postup tvorby programu v Lotus Domino Designer.



Obr. 15 Diagram tříd modulu Přijaté faktury

4.3.3 Rozhraní systému

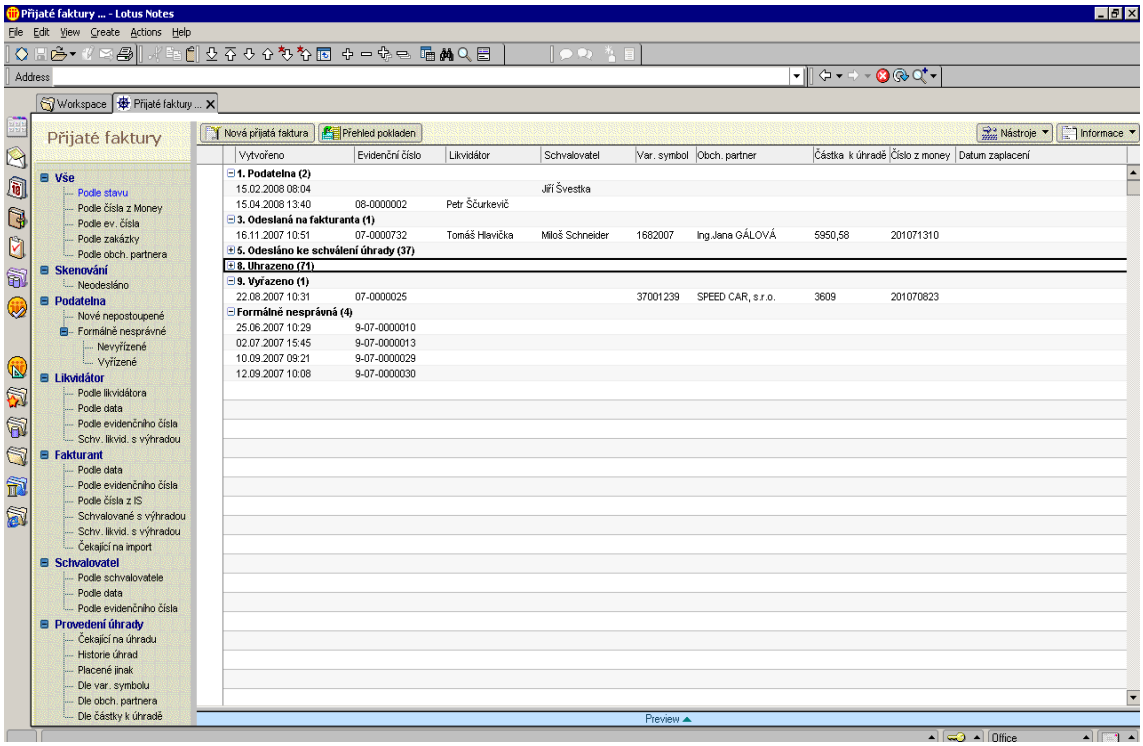
Zobrazení pohledů na faktury a správu dat zajišťují rozhraní systému, frameworks, jak specifikuje návrhové prostředí Lotus Domino Designer (viz kap. 4.3). Agenti obsluhují pomocí akcí jednotlivá pole. Akce se píšou pomocí formulí jazyka LotusScript, ale to není předmětem této práce. Tyto jednoduché formule si totiž píší administrátoři systému podle diagramu aktivit a také podle aktuálních změn v business a pracovních postupech. Já jsem se v následující kapitole zaměřila na import dat do Přijatých faktur, který je zajišťován pomocí Scipt Libraries.



Obr. 16 Model návrhu modulu Přijatých faktur

4.3.4 Návrh vzhledu obrazovek modulu

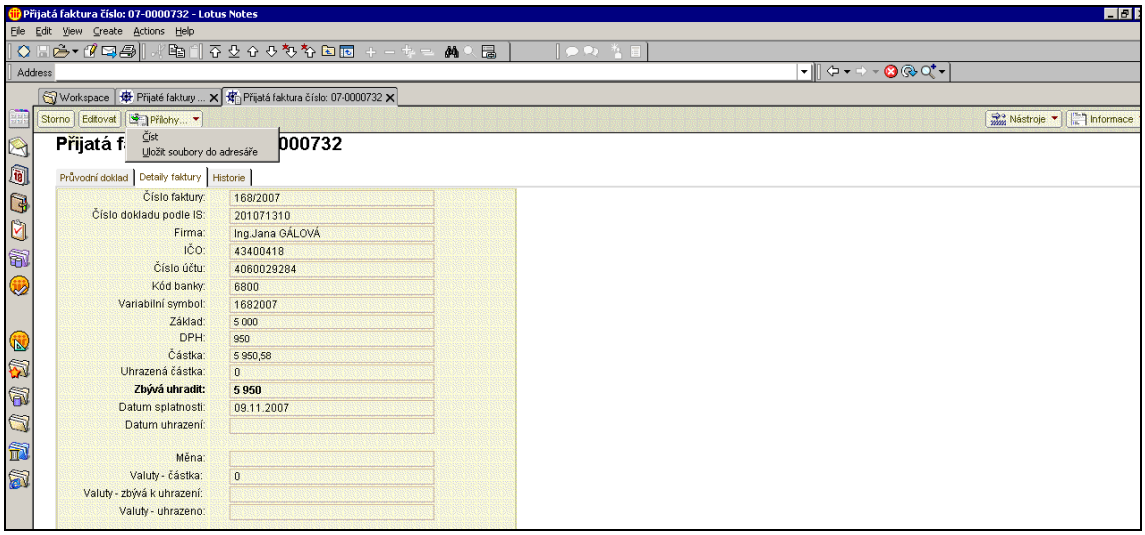
Jak jsem již uvedla na začátku analýzy, zaměřila jsem se na základní návrh modulu, tedy hlavně funkční požadavky. Vzhled výsledné aplikace je totiž dán vzhledem a uspořádáním již hotových modulů, aby byla zachována jejich jednotnost. V levé části obrazovky musí být kaskádová nabídka. Prostým kliknutím na požadovanou položku se zobrazí daný seznam přijatých faktur, roztříděný podle definovaných položek. Na obr. 15 můžeme vidět všechny položky modulu, je to pohled administrátora aplikace. Díky tomu také v pravé části obrazovky můžeme vidět přehled o všech stavech všech faktur.



Obr. 17 Návrh vzhledu modulu

5 Návrh aktualizace dat v modulu Přijaté faktury

Aktualizace dat v navrhovaném modulu je součástí rozhraní systému, jak ukazuje model na obr. 15. Když je faktura ve stavu č. 3, tzn. je založena jako nová v systému a má připojen jako přílohu svůj naskenovaný originál, fakturant podle něj přepíše její údaje do samostatného účetního programu Money. Fakturu pak přepne do stavu č. 3, čekající na import. Aby se data nepřepisovala dvakrát, využije nový modul schopnosti programu Money exportovat svá data. V pravidelných intervalech probíhá jednosměrný export dat z účetního systému a nyní probíhá také export přijatých faktur. Rovněž tak bude probíhat v pravidelných intervalech import dat do systému Lotus a jeho modulu Přijaté faktury. Párování bude probíhat na základě párovacího symbolu, který vkládá fakturant. V případě korektního importu dat dojde k naplnění karty „Detaily faktury“ a faktura se přepne do následujícího stavu, stavu č. 5.



Obr. 18 Detaily faktury

5.1 Výměna dat mezi systémy

Synchronizaci dat, která byla navrhnuta již dříve pro celý informační systém, lze rozložit do jednotlivých kroků:

1. Na serveru pro účetní a personální systémy se každou hodinu spouští naplánovaná úloha Export z Money do IS. Volaný skript vyexportuje všechna data z Money do zvoleného adresáře. Ve složce 2008 lze ověřit aktuálnost dat podle data, v souboru PFaktury.dbf lze ve sloupci AG (obsahující párovací symboly) zkontrolovat, jestli obsahuje nově do Money zadané přijaté faktury.

2. Na serveru, který obsahuje mimo jiné nový informační systém se se zpožděním také každou hodinu spustí naplánovaná úloha Automat_Money, která vytvoří z DBF odpovídající XLS soubory. U vzniklého souboru PFaktury.xls lze zkontrolovat jak datum, tak jeho obsah.

3. Na stejném serveru, opět se zpožděním každou hodinu se spustí naplánovaná úloha Automat_LN, která aktualizuje soubor IS_WheelIn.mdb (u kterého lze opět ověřit datum). Přímo v tomto souboru lze v tabulce tblUcto_MON_PF zkontrolovat ve sloupci MON_PF_IMP_PARSYMBOL, zda obsahuje nové faktury.

4. Na zálohovacím serveru se každých 5 minut spustí Export z Wheelu do IS, který jen přenáší data z IS_WheelIn.mdb do zálohovacího adresáře, kde jsou nachystána pro zpracování Lotus Notes agentem.

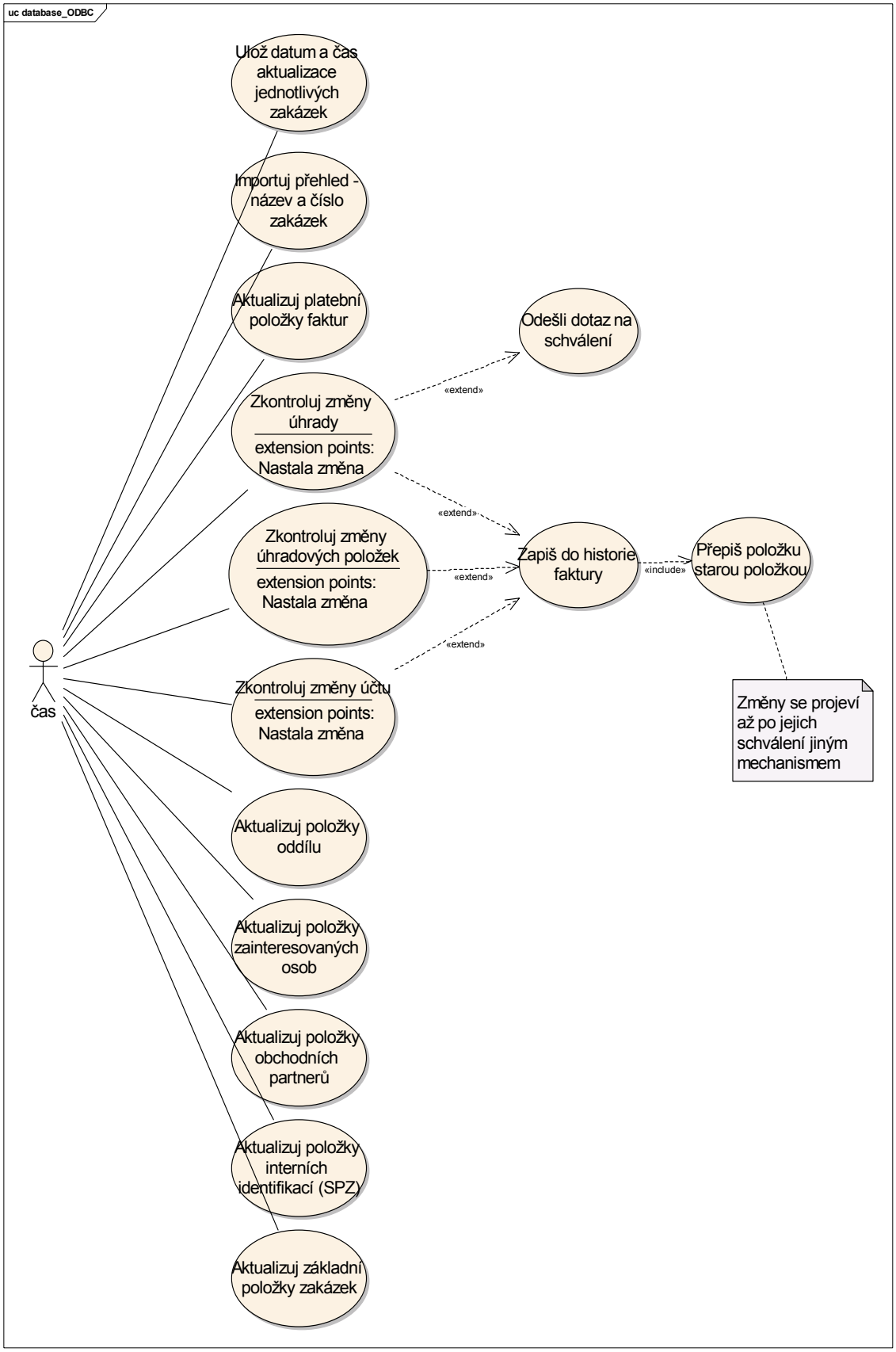
5. Každých 15 min. od 6 do 18 hodin na zálohovacím serveru zpracuje připravená data z příslušného adresáře Lotus Notes agent, který mimo jiné provede aktualizaci dat v prijate_faktury.nsf, což je soubor databáze Lotus Notes obsahující modul informačního systému Přijaté faktury.

5.2 Aktualizace dat v kartě Detaily faktury

Aktualizace dat v modulu Přijatých faktur se týká především detailů faktury. Celá aktualizace je dána konceptem výměny dat, konkrétně aktualizace dat detailů faktury se bude provádět pomocí Lotus Domino Connector pro přístup k ODBC databázím definovaného v knihovně LSXODBC. Agent definující přístup k ODBC databázi je již napsán v knihovně ODBC_general, kterou budu požívat pro aktualizaci konkrétních dat v kartě Detaily faktury.

5.2.1 Požadavky na knihovnu

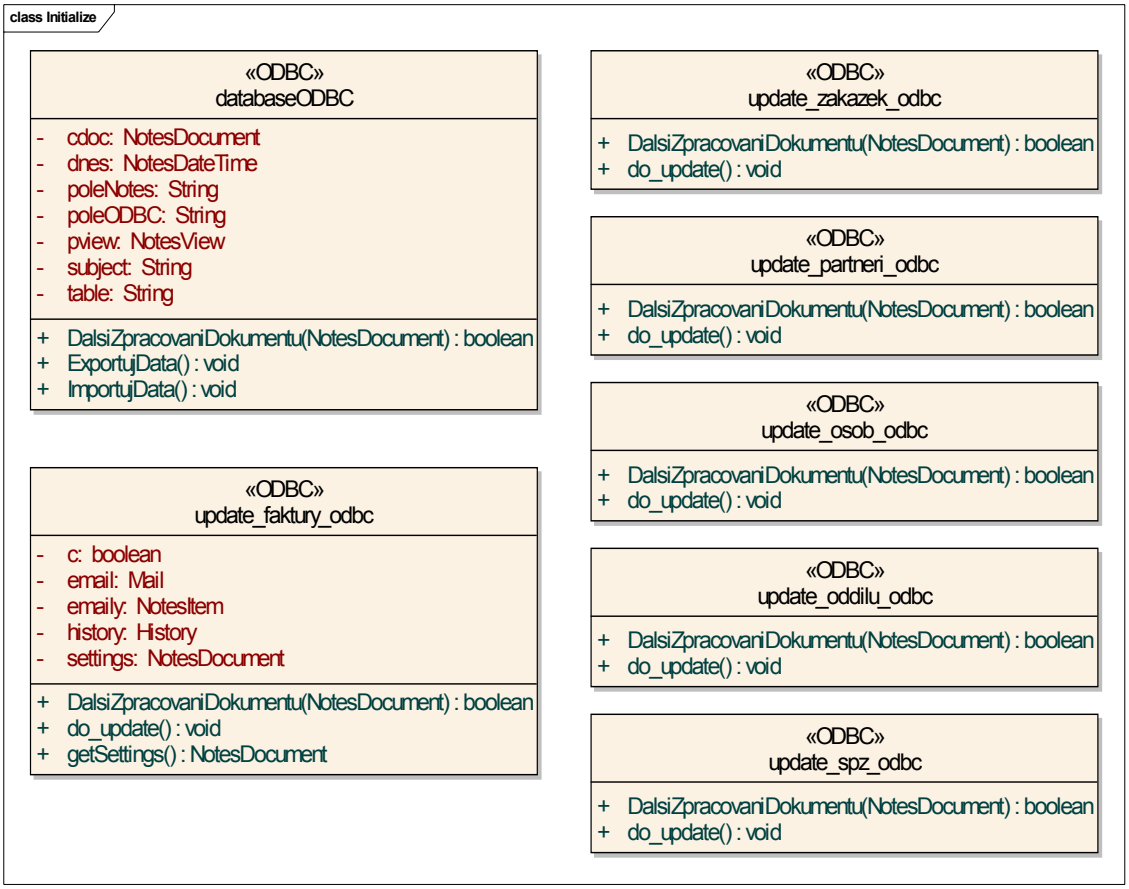
Pro popis požadavků na knihovnu jsem zvolila grafický popis, diagram případů užití. Diagram ukazuje čím se bude knihovna zabývat. Jedná se zálohu přehledu aktualizovaných zakázek, import přehledu zakázek podle účetního systému a u úhradových položek je třeba pohlídat případné změny a zapsat je do historie. V případě změny schválené částky k úhradě je třeba vyvolat nové schválení položky.



Obr. 19 Use Case diagram aktualizace Detailů faktury

5.3 Diagram tříd podle analýzy požadavků knihovny

Podle případů užití lze rozeznat několik tříd a sestavit diagram tříd. Spolupráce tříd není znázorněna, neboť o použití každé třídy bude rozhodovat funkce DalsiZpracovaniDokumentu z knihovny ODBC_general podle toho, jestli proběhne daná část aktualizace dat v pořádku. Uložením data a času jednotlivých zakázek a importem nového přehledu zakázek se zabývá třída databaseODBC. K aktualizací základních položek zakázek slouží třída update_faktury_odbc, která má také za úkol kontrolu změn účtů a úhradových položek a případné vyvolání jejich schválení. Ostatní třídy jsou určena k aktualizaci dat dalších položek Detailu faktury.



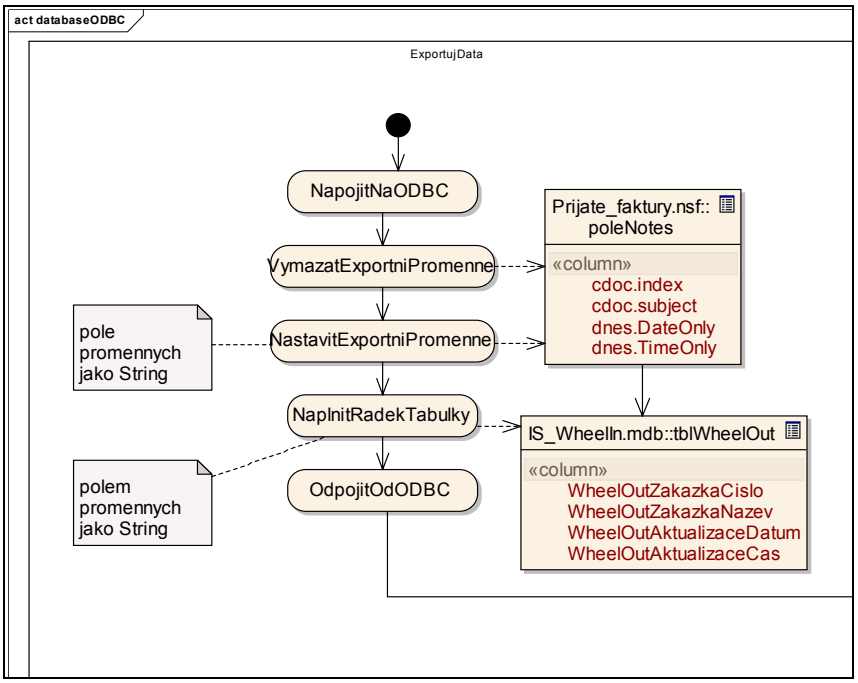
Obr. 20 Class Diagram knihovny database_ODBC

5.4 Analýza algoritmů knihovny pomocí diagramů aktivit

Funkce jednotlivých tříd a konkrétní data, která aktualizují jsou popsány diagramy aktivit.

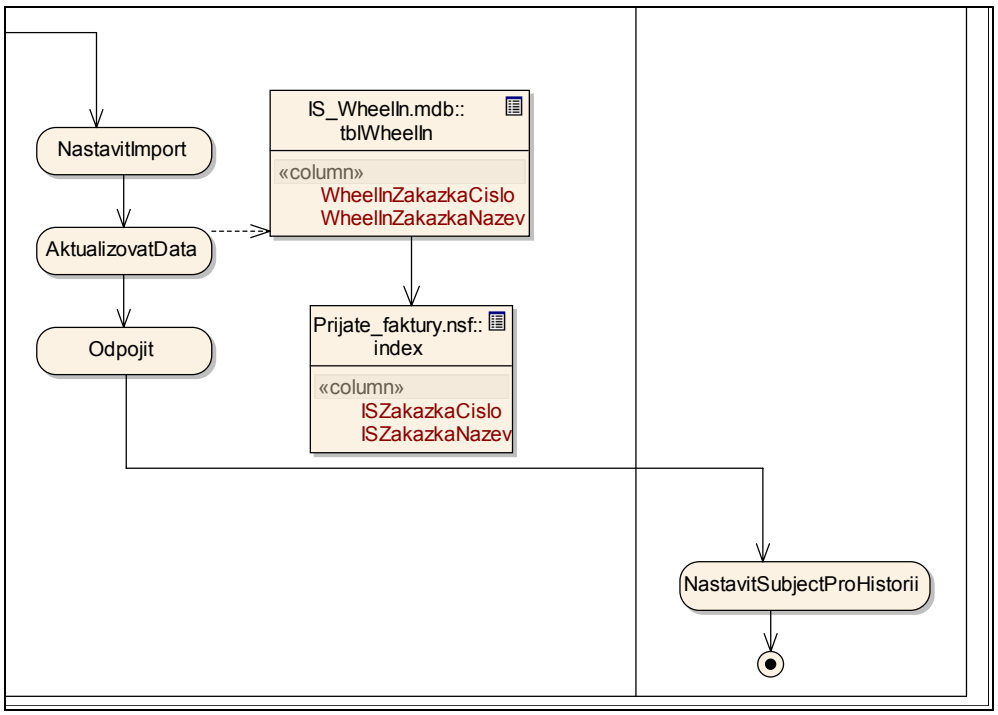
5.4.1 Návrh třídy databaseODBC

Aktivity diagram této třídy je zobrazen v příloze č. 3. Zde je zobrazen ve dvou jeho hlavních částech. Nejprve se zabývá zálohou přehledu aktualizovaných zakázek do tabulky tblWheelOut.



Obr. 21 Upravený diagram aktivit třídy databaseODBC – 1.část

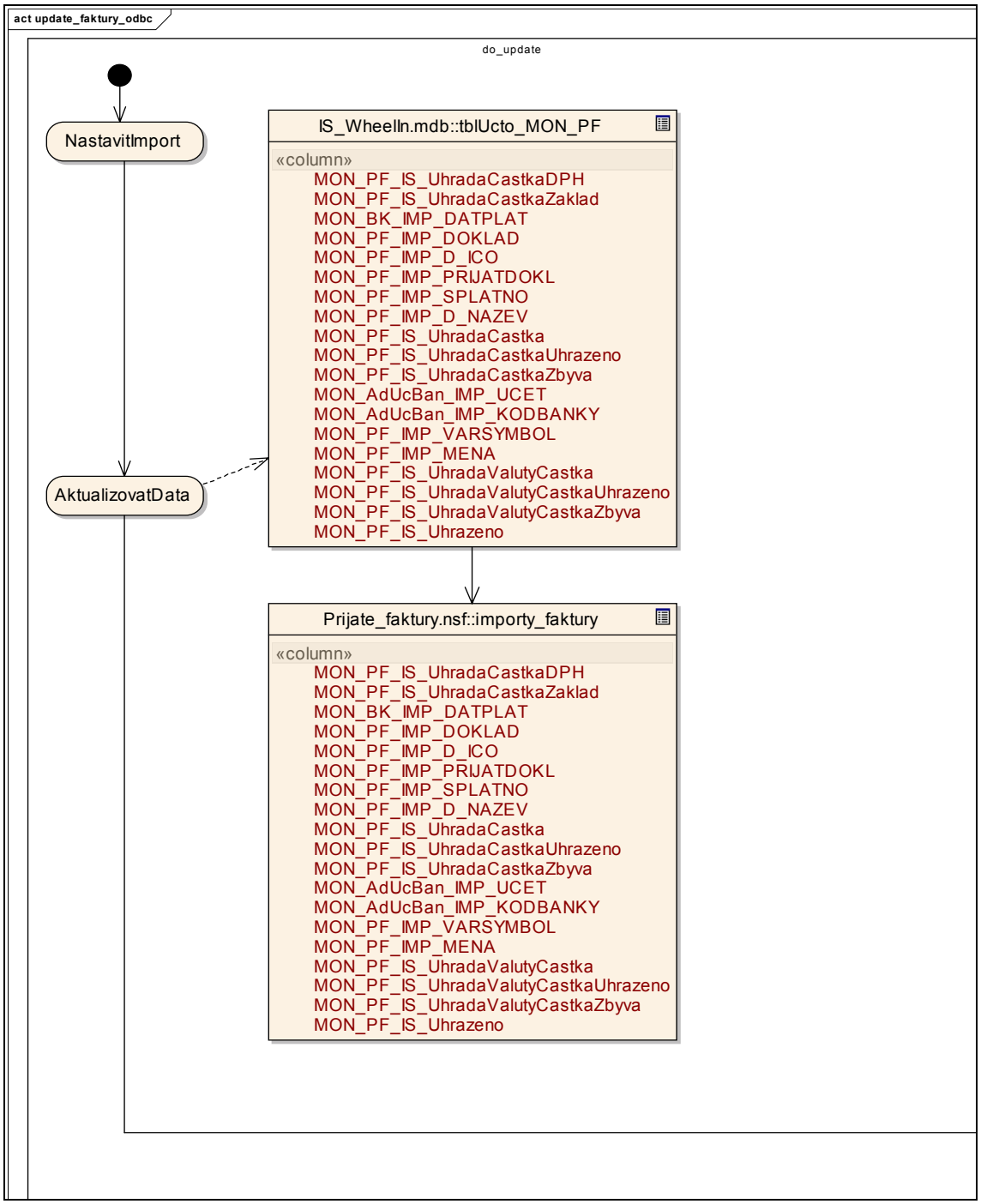
Dalším krokem je import přehledu zakázek podle účetního systému z tabulky tblWheelIn do tabulky nazvané index v databázi Lotus Notes Přijaté faktury (Prijate_faktury.nsf).



Obr. 22 Upravený diagram aktivit třídy databaseODBC – 2.část

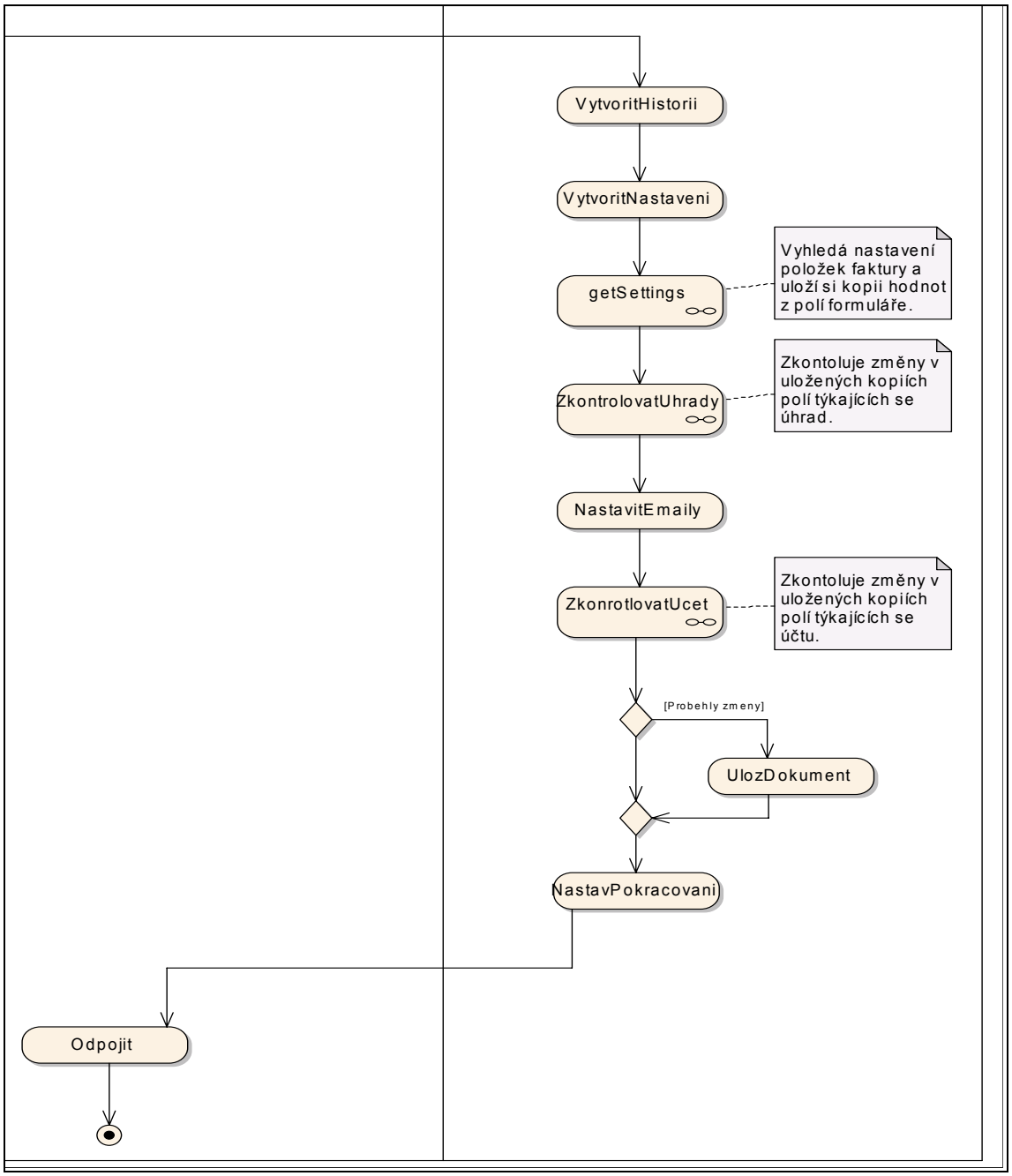
5.4.2 Návrh třídy Update_faktury_odbc

Aktivity diagram třídy Update_faktury_odbc je zobrazen v příloze č. 4. Zde je zobrazen ve dvou jeho hlavních částech. Nejprve aktualizuje základní položky zakázky v pohledu importy_faktury podle tabulky tblUcto_MON_PF.



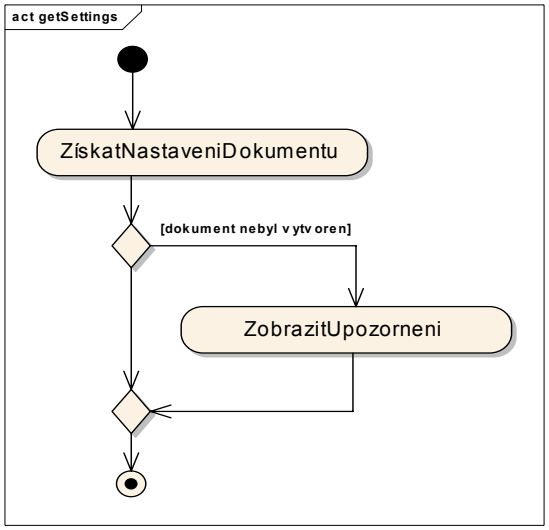
Obr. 23 Upravený diagram aktivit třídy Update_faktury_odbc – 1.část

Potom zkontroluje změny účtů a úhradových položek, které je třeba zapsat do vytvořené historie zakázky. Aby mohly být změny zkontrolovány, vyhledá se nastavení daného formuláře obsahující předmětné položky. Postup je znázorněn jako Structured Activity, tzn. že je rozebrán v dalším diagramu. Nastavení dokumentu také obsahuje zvlášť kopie jeho položek, což zajišťuje prostředí Lotus Notes. V případě změny schválené částky k úhradě je třeba vyvolat nové schválení položky. Pro kontrolu úhrad jsem navrhla další Structured Activity ZkontrolovatUhrady a pro kontrolu položek účtu Structured Activity ZkontrolovatUcet. Pokud proběhly v aktuálním dokumentu faktury změny, musí se uložit. Pak se již jen nastaví pokračování pro aktualizaci dalších položek Detailu faktury.



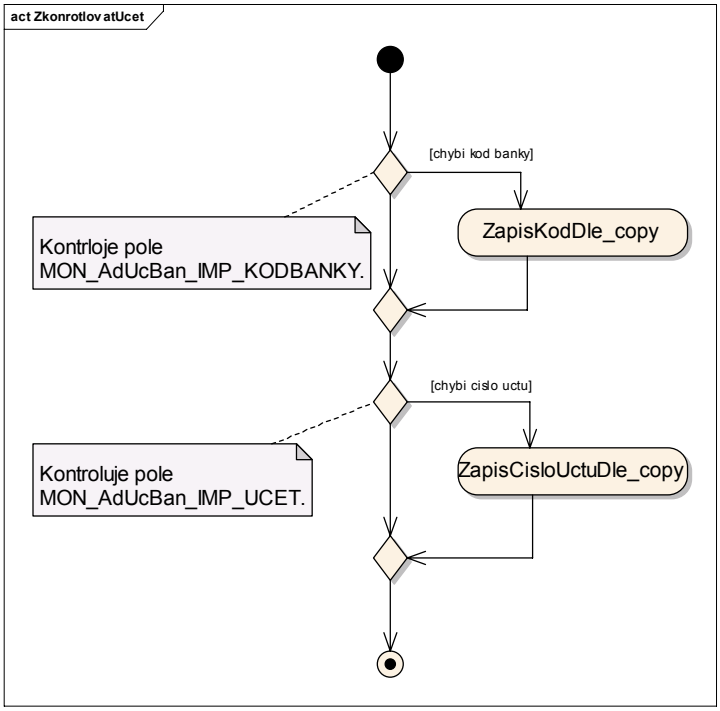
Obr. 24 Upravený diagram aktivit třídy Update_faktury_odbc – 2.část

Structured Activity getSettings vyhledá profil dokumentu nastavený v Lotus Notes a převezme jeho nastavení. Pokud tento dokument nenalezne, nemůže aktualizovat jeho položky, proto musí zobrazit upozornění.



Obr. 25 Subdiagram aktivit getSettings třídy Update_faktury_odbc

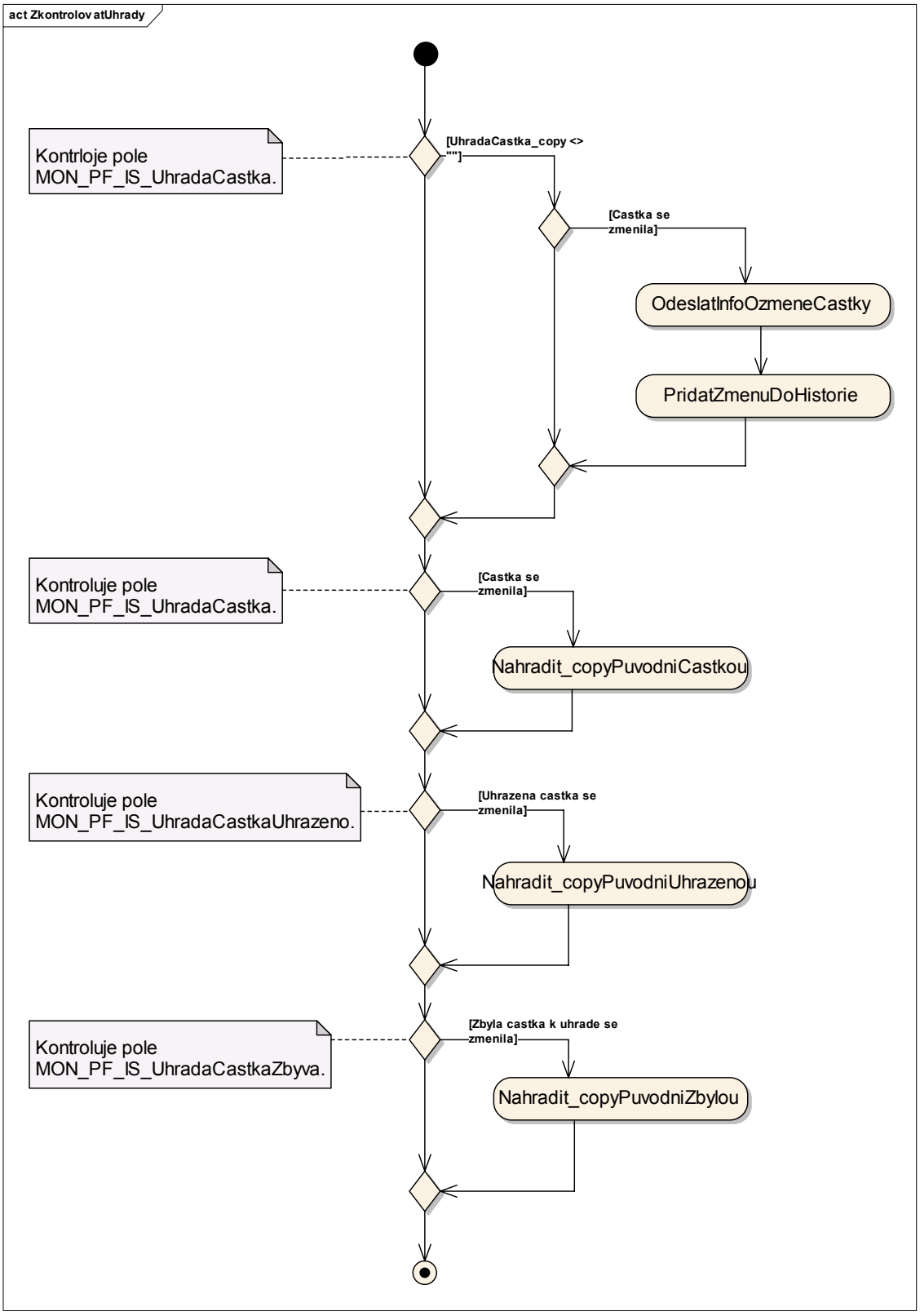
Structured Activity ZkontrolovatUcet má na starost zapsat údaje bankovního účtu podle zálohovaných předchozích dat, pokud v importovaných datech údaje z jakéhokoliv důvodu chybí.



Obr. 26 Subdiagram aktivit ZkontrolovatUcet třídy Update_faktury_odbc

Structured Activity ZkontrolovatUhrady nejprve zkontroluje, je-li položka částka k úhradě v impotovaných datech vyplněna. Pokud ne, zapíše údaj podle

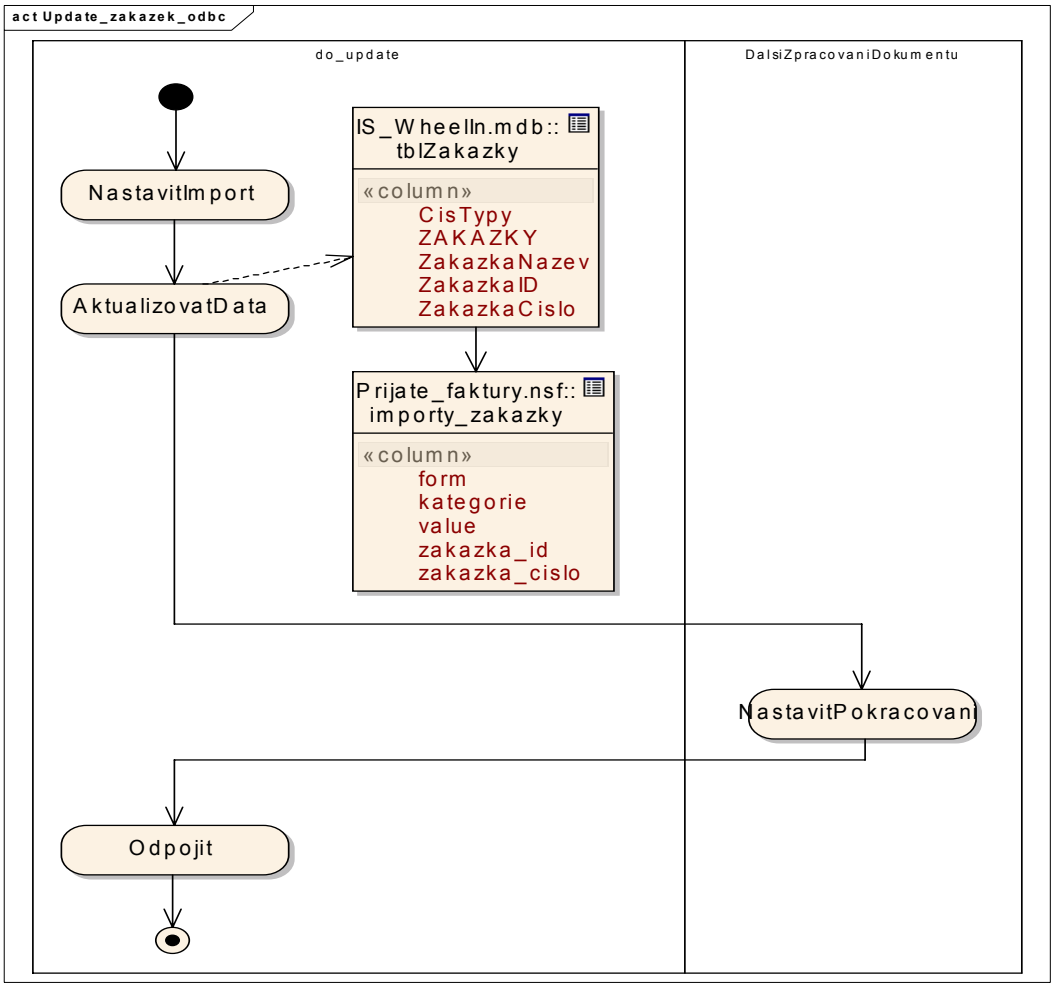
zálohovaných předchozích dat. Jestliže částka vyplněna je, odešle email o změně této částky příslušnému schvalovateli. Další aktivity kontrolují vyplnění údajů o uhrazené částce a zbylé částce k úhradě.



Obr. 27 Subdiagram aktivit ZkontrolovatUhrady třídy Update_faktury_odbc

5.4.3 Návrh třídy Update_zakazek_odbc

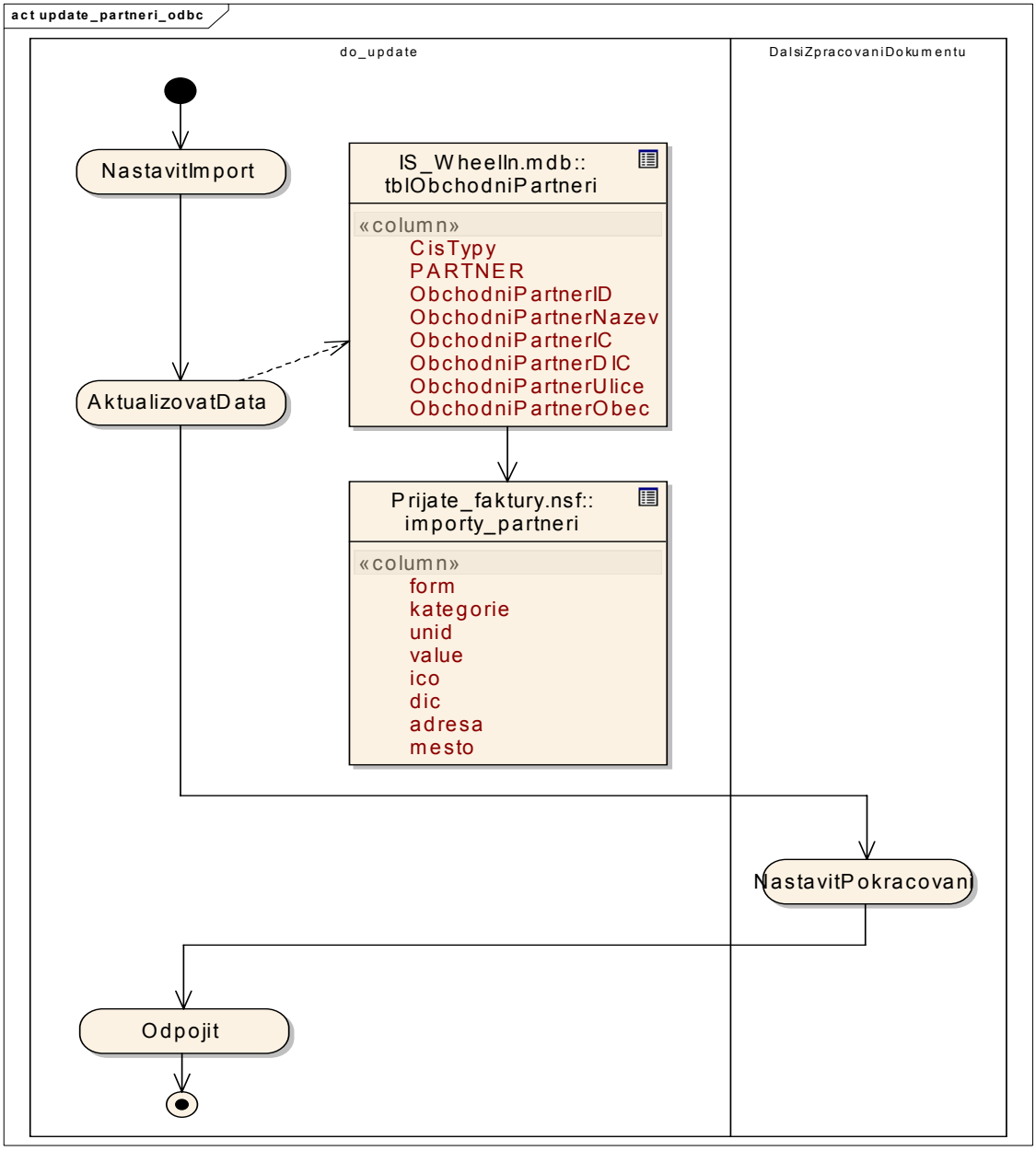
Třída je určená k aktualizaci polí týkajících se zakázky, ke které faktura patří. Nejprve nastaví příkazy pro daný import dat a pak provede update příslušných polí formuláře zobrazených v diagramu aktivit.



Obr. 28 Diagram aktivit třídy Update_zakazek_odbc

5.4.4 Návrh třídy Update_partneri_odbc

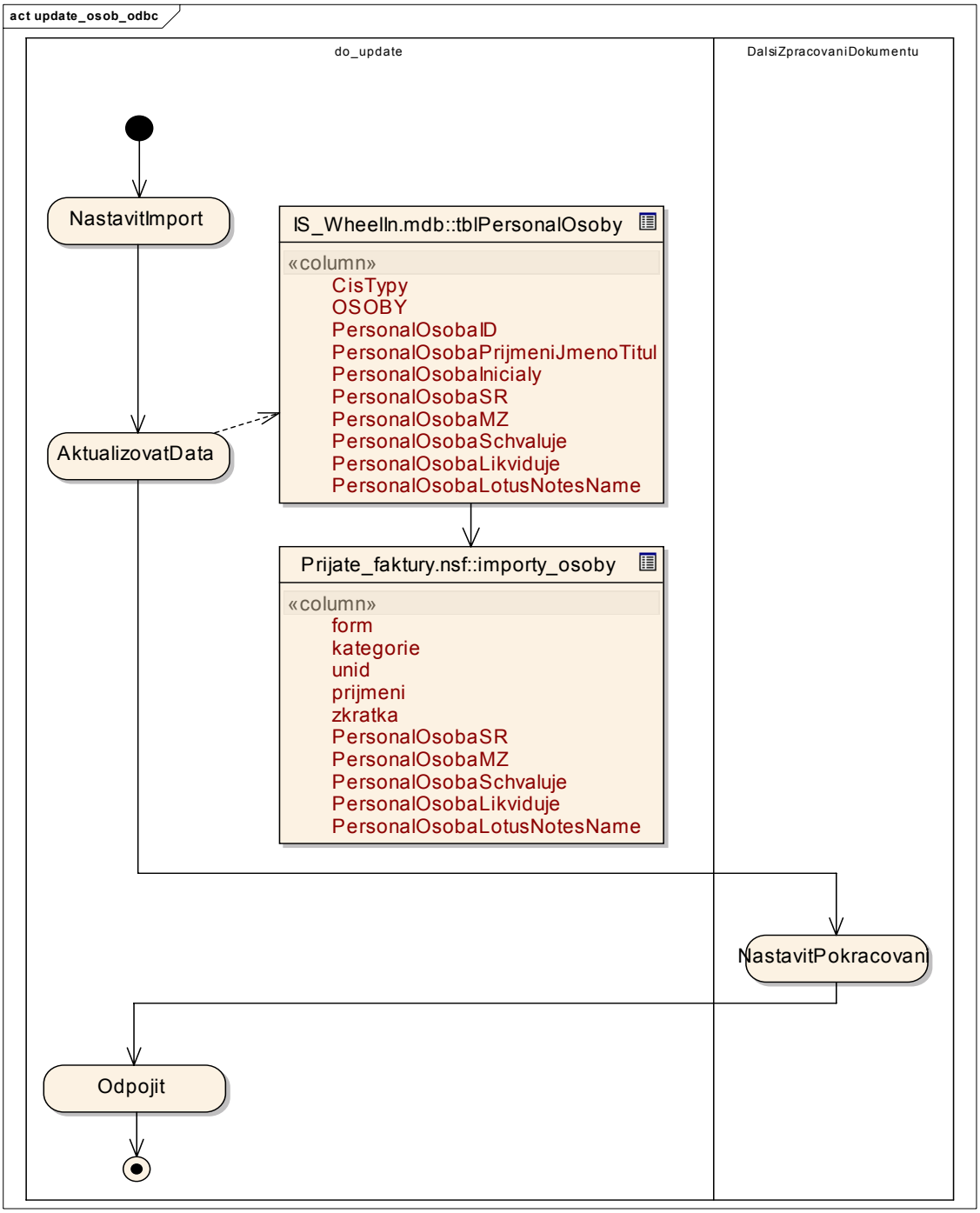
Třída je určena k aktualizací polí týkajících se údajů o obchodních partnerech pro konkrétní fakturu dané zakázky. Nejprve nastaví příkazy pro daný import dat a pak provede update příslušných polí formuláře zobrazených v diagramu aktivit.



Obr. 29 Diagram aktivit třídy Update_partneri_odbc

5.4.5 Návrh třídy Update_osob_odbc

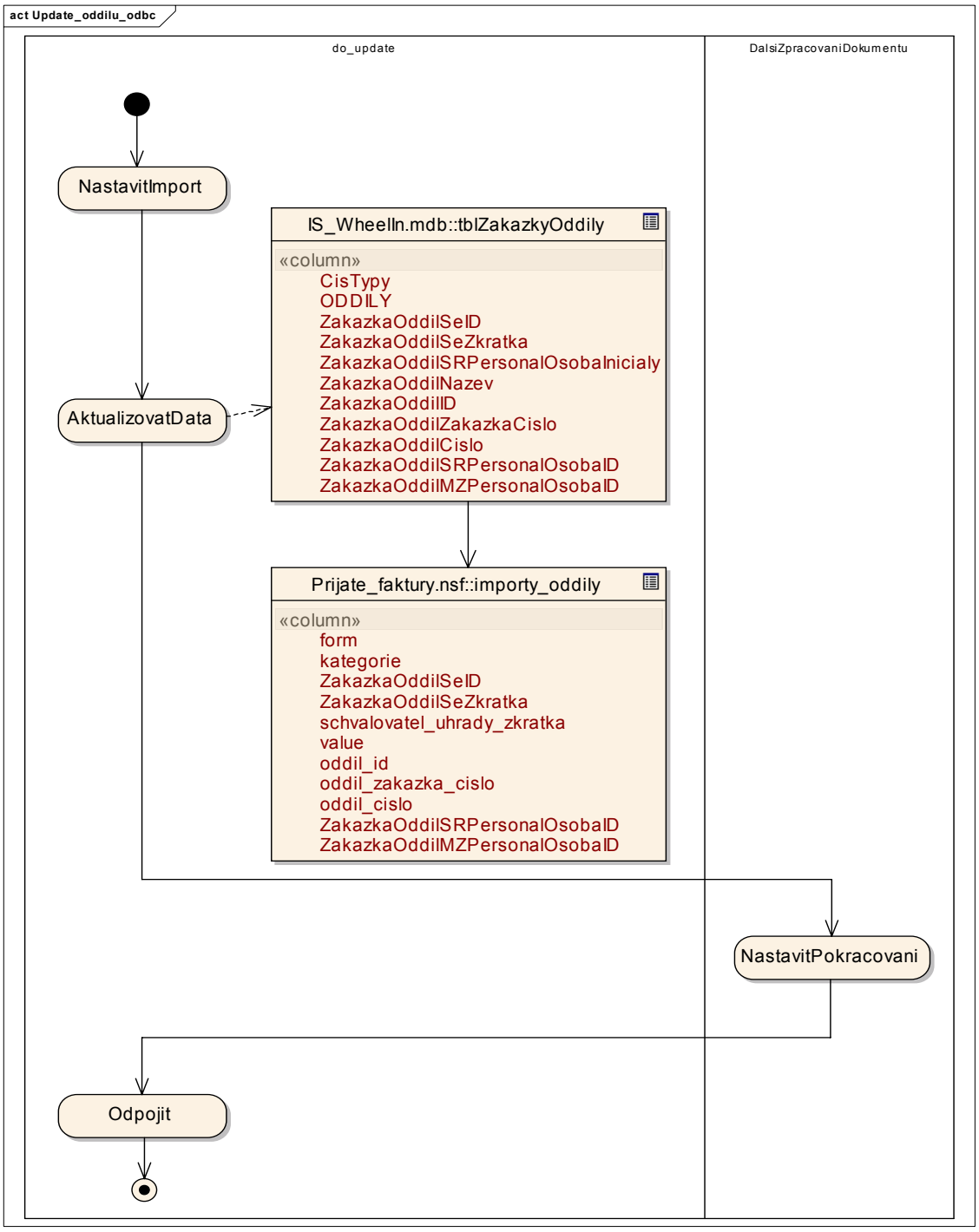
Tato třída se zabývá aktualizací polí týkajících se údajů o vnitropodnikových osobách zainteresovaných ve víceúrovňovém schvalovacím procesu pro přijaté faktury. Nejprve nastaví příkazy pro daný import dat a pak provede update příslušných polí formuláře jak ukazuje diagram aktivit.



Obr. 30 Diagram aktivit třídy Update_osob_odbc

5.4.6 Návrh třídy Update_oddilu_odbc

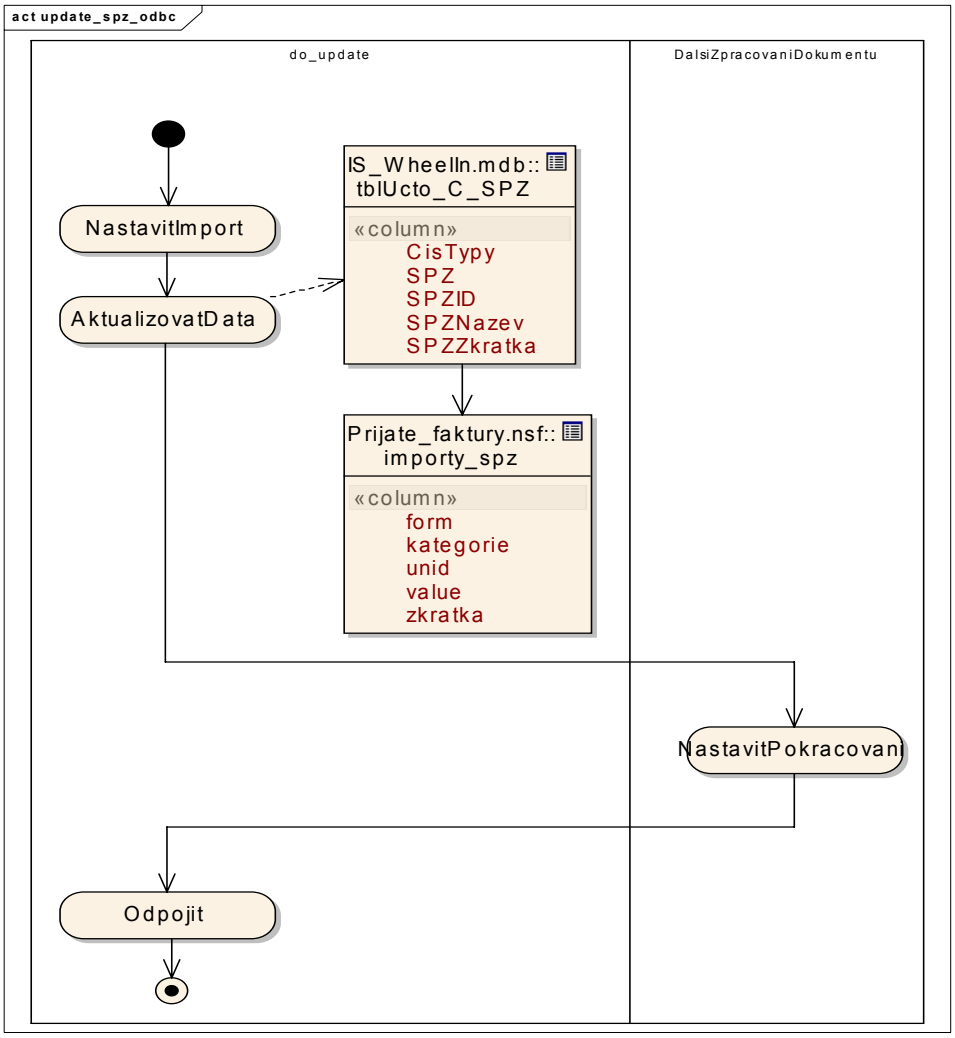
Tato třída se zabývá aktualizací polí s údaji o vnitropodnikovém oddílu, který zpracovává danou zakázku, ke které přijatá faktura patří. Nejprve nastaví příkazy pro daný import dat a pak provede update příslušných polí formuláře jak zobrazuje diagram aktivit.



Obr. 31 Diagram aktivit třídy Update_oddílu_odbc

5.4.7 Návrh třídy Update_spz_odbc

Tato třída se zabývá aktualizací polí vnitropodnikové identifikace zakázky, ke které přijatá faktura patří. Nejprve nastaví příkazy pro daný import dat a pak provede update příslušných polí formuláře jak podle diagramu aktivit.



Obr. 32 Diagram aktivit třídy Update_spz_odbc

6 Realizace aktualizace dat v Detailu faktury

Aktualizaci dat v pohledu Detaily faktury jsem realizovala v programovacím prostředí Lotus Domino Designer, v jazyce LotusScript. Tento nástroj a programovací jazyk je popsán v kapitole 4.3. Všechny navržené algoritmy používají knihovnu ODBC_general, dědí z ní některé metody třídy ODBC a používají v ní definované funkce pro přístup k databázi IS_WheelIn.mdb. Třída ODBC_general používá Lotus Domino Connector pro přístup k ODBC databázím pomocí knihovny LSXODBC. Kromě LotusScriptu bylo třeba použít SQL dotazů pro výběr dat z tabulek databáze.

Algoritmy jsem navrhovala podle diagramů aktivit pro jednotlivé třídy knihovny popsané v předchozí kapitole.

6.1 Třída databaseODBC

Podle diagramu aktivit v příloze č.2 a rozebraného v kapitole 5.3.1 vznikl tento algoritmus. Pro práci s daty je nejprve třeba vytvořit připojení s externí databází. K tomu třída používá funkci z knihovny ODBC ConnectToODBC, pro kterou je třeba nastavit název zdrojové knihovny pomocí ODBCSource. Protože LotusNotes používá k zobrazení formulářů tzv. pohledy, nastaví se první pohled na databázi podle jednoho z jejích pohledů ExportIS2000 a použije se první pohled jeho dokumentu.

Do tabulky tblWheelOut pak budou zapsána data nastavená v proměnné poleODBC. Před zápisem se nastaví SQL dotaz pro smazání předchozích položek tabulky, což se vykoná zavoláním funkce ResultSetODBC z knihovny ODBC. Pak se podle nastaveného pohledu dokumentu naplní proměnná poleNotes, jejíž položky se připojí k položkám poleODBC a pomocí ResultSetODBC se vykoná záloha přehledu aktualizovaných položek. Plnění řádků tabulky se opakuje, dokud není splněna podmínka, že v dokumentu již nejsou žádné položky k naplnění proměnné poleNotes:

```
Class databaseODBC As ODBC

Sub New
End Sub

Function ExportujData
      Dim pview As NotesView
      Dim cdoc As NotesDocument
      Dim table As String, poleODBC As String, poleNotes As
            String
      Dim dnes As New NotesDateTime(Now)
'Napojení na ODBC
      Me.ODBCSource ="TCL"
      Call ConnectToODBC
'Nastavení exportní proměnné
      Set pview=db.GetView("ExportIS2000")
      Set cdoc=pview.GetFirstDocument
      table="tblWheelOut"
      poleODBC="(WheelOutIndexID,WheelOutZakazkaCislo,WheelOutZak
            azkaNazev,WheelOutAktualizaceDatum,WheelOutAktualizac
            eCas)"
```

```
'Vymazání exportních proměnných
     ODBCSQL = "DELETE FROM "+table
     Call ResultSetODBC
'Naplnění exportní proměnné
     Do Until cdoc Is Nothing
          poleNotes="('"+Trim(cdoc.index(0))+"','','"+Trim(cdoc
          .subject(0))+"','"+dnes.DateOnly+"','"+dnes.TimeOnly+
          "')"
'Naplnění řádku tabulky
     ODBCSQL = "INSERT INTO "+table+" "+poleODBC+" VALUES
          "+poleNotes
     Call ResultSetODBC

     Set cdoc=pview.GetNextDocument(cdoc)
     Loop
     Call con.Disconnect
End Function
```

Pro import dalších dat se bude používat zdroj K4. Dále se nastaví vyhledávací dotaz pro vyhledání položek z tabulky tblWheelIn, pohled pro vyhledání dokumentu keyViewName jako „index“ a klíčem pro hledání unikátního záznamu bude pole WheelInIndexID. V proměnné seznampoli se nastavují pole, která se budou přenášet z tblWheelIn do nastavených polí pohledu „index“ databáze v Lotus Notes. Třída ODBC definuje ve funkci ConnectToContact, kterou používá funkce RunUpdate význam posloupnosti znaků „->“ jako „přemísti položku z prvního pole do druhého pole“. Rovněž definuje znak „|“ jako oddělovač pro dvojice polí. Nakonec se ve funkci ImportujData nastaví pro RunUpdate informace, že chci provést pouze update dat (ne další funkce jako je smazání přečtených položek atp.):

```
Function ImportujData
'Nastavení importu
     Me.ODBCSource ="K4"
     Me.ODBCSQL="SELECT * from tblWheelIn"
     Me.keyViewName="index"
     Me.SearchKey="WheelInIndexID"
     Me.seznampoli="WheelInZakazkaCislo-
          >ISZakazkaCislo|WheelInZakazkaNazev->ISZakazkaNazev"
     Me.justUpdate=True
'Aktualizovat data
     Call RunUpdate
'Odpojit
     Call con.Disconnect
End Function
```

Z funkce RunUpdate knihovny ODBC je volána funkce DalsiZpracovaniDokumentu, která slouží k nastavení, jestli chci se současným dokumentem ještě manipulovat. V této třídě je přepsána pro nastavení předmětu do zápisu do historie dokumentu:

```
Function DalsiZpracovaniDokumentu(zakazka As NotesDocument) As
      Boolean
      Dim subject As String
      subject="Založená zakázka v IS č. " &
            zakazka.ISZakazkaCislo(0) & " s názvem " &
            zakazka.ISZakazkaNazev(0)
End Function

End Class
```

6.2 Třída update_faktury_odbc

Hlavní funkcí této třídy je funkce do_update(). Ta nejprve nastaví dotaz pro vyhledání položek z tabulky tblUcto_MON_PF, pohled pro vyhledání dokumentu keyViewName jako „importy_faktury“ a klíčovým polem bude MON_PF_IMP_PARSYMBOL. V proměnné seznampoli se nastavují pole, která se budou přenášet z tabulky do nastavených polí pohledu „importy_faktury“ databáze v Lotus Notes podle diagramu na obr. 23. Jak jsem již napsala dříve, třída ODBC definuje ve funkci ConnectToContact, kterou používá funkce RunUpdate význam posloupnosti znaků „->“ jako „přemísti položku z prvního pole do druhého pole“. Rovněž definuje znak „|“ jako oddělovač pro dvojice polí:

```
Class update_faktury_odbc As ODBC
      Sub New
      End Sub

Function do_update()
'Nastavení importu
      Me.ODBCSource ="K4"
      Me.ODBCSQL="SELECT * from tblUcto_MON_PF"
      Me.keyViewName="importy_faktury"
      Me.SearchKey="MON_PF_IMP_PARSYMBOL"
      Me.seznampoli="MON_PF_IS_UhradaCastkaDPH-
            >MON_PF_IS_UhradaCastkaDPH|MON_PF_IS_UhradaCastkaZakl
            ad->MON_PF_IS_UhradaCastkaZaklad|MON_BK_IMP_DATPLAT-
            >MON_BK_IMP_DATPLAT|MON_PF_IMP_DOKLAD-
            >MON_PF_IMP_DOKLAD|MON_PF_IMP_D_ICO-
            >MON_PF_IMP_D_ICO|MON_PF_IMP_PRIJATDOKL-
            >MON_PF_IMP_PRIJATDOKL|MON_PF_IMP_SPLATNO-
            >MON_PF_IMP_SPLATNO|MON_PF_IMP_D_NAZEV-
            >MON_PF_IMP_D_NAZEV|MON_PF_IS_UhradaCastka-
            >MON_PF_IS_UhradaCastka|MON_PF_IS_UhradaCastkaUhrazen
            o-
            >MON_PF_IS_UhradaCastkaUhrazeno|MON_PF_IS_UhradaCastk
            aZbyva-
            >MON_PF_IS_UhradaCastkaZbyva|MON_AdUcBan_IMP_UCET-
            >MON_AdUcBan_IMP_UCET|MON_AdUcBan_IMP_KODBANKY-
            >MON_AdUcBan_IMP_KODBANKY|MON_PF_IMP_VARSYMBOL-
            >MON_PF_IMP_VARSYMBOL"
      Me.seznampoli=Me.seznampoli+"|MON_PF_IMP_MENA-
            >MON_PF_IMP_MENA|MON_PF_IS_UhradaValutyCastka-
            >MON_PF_IS_UhradaValutyCastka|MON_PF_IS_UhradaValutyC
            astkaUhrazeno-
```

```
            >MON_PF_IS_UhradaValutyCastkaUhrazeno|MON_PF_IS_Uhrad
            aValutyCastkaZbyva-
            >MON_PF_IS_UhradaValutyCastkaZbyva|MON_PF_IS_Uhrazeno
            ->MON_PF_IS_Uhrazeno"
      Me.justUpdate=True
      Me.removeDocs=False
      Me.Searchformula=""
      Me.compute_with_form=False

      Call RunUpdate
      Call con.Disconnect

End Function
```

Následující funkce getSettings() je nachystána pro použití funkcí DalsiZpracovaniDokumentu. Funkce getSettings() vyhledá profil dokumentu nastavený v Lotus Notes a převezme jeho nastavení. Pokud tento dokument nenalezne, nemůže aktualizovat jeho položky, proto musí zobrazit upozornění:

```
Function getSettings() As NotesDocument
      Dim settings As NotesDocument
      Set settings=db.GetProfileDocument("settings")
      If settings Is Nothing Then
            Msgbox "Nebyl vytvořen dokument nastavení. Požádejte
            administrátora."
      End If
      Set getSettings= settings
End Function
```

Z funkce RunUpdate knihovny ODBC je volána funkce DalsiZpracovaniDokumentu, která slouží k nastavení, jestli chci se současným dokumentem ještě manipulovat. V této třídě je použita pro kontrolu některých položek na změnu. Nejprve vytvoří novou položku historie dokumentu a pak převezme pomocí funkce getSettings() profil nastavení dokumentu. Před samotnou kontrolou položek ještě nastaví emaily pro zasílání schvalovacích požadavků při změně v kontrolovaných polích:

```
Function DalsiZpracovaniDokumentu(doc As NotesDocument) As
      Boolean
      Dim email As Mail
      Dim history As History
      Dim settings As NotesDocument
      Dim emaily As NotesItem

      Set history=New History()

      Dim c As Boolean

      Set settings=getSettings()
      Set emaily=settings.GetfirstItem("fakturanti")
      c=False
```

Kontrola změn v uložených kopiích polí týkajících se úhrad probíhá podle diagramu na obr. 27. Pokud byla původní částka vyplněna (nyní je v kopii) a pokud v importované částka je hodnota jiná, pak je třeba poslat email na schválení nové částky a zapsat změnu do historie faktury. Pokud se částka k úhradě změnila, je třeba nahradit položku kopie v nastavení dokumentu za novou částku. Stejně tak u položky Uhrazená částka a Zbývá uhradit. Funkce Cstr zajistí vrácení její proměnné jako řetězce pro snadné porovnávání hodnot polí:

```
'Kontrola změn v uložených kopiích polí týkajících se úhrad
     If Cstr(doc.MON_PF_IS_UhradaCastka_copy(0))<>""
     Then If
     doc.MON_PF_IS_UhradaCastka(0)<>doc.MON_PF_IS_UhradaCastka_
     copy(0)
          Then
          'odešli info o změně částky
          Set email=New Mail(doc,emaily.values,"U faktury číslo
               "+Cstr(doc.ev_cislo(0))+" byla změněna celková
               výše úhrady z
               "+Cstr(doc.MON_PF_IS_UhradaCastka_copy(0))+" na
               "+Cstr(doc.MON_PF_IS_UhradaCastka(0)),"Prosím o
               schválení úhrady faktury viz. přiložený odkaz")
          Call email.send()
          'a proveď zápis do historie
           Call history.AddHistoryMessageDirect(doc,"Byla
                změněna celková výše úhrady z
                "+Cstr(doc.MON_PF_IS_UhradaCastka_copy(0))+" na
                "+Cstr(doc.MON_PF_IS_UhradaCastka(0)))
          End If
     End If

'pokud se částka k úhradě změnila, je třeba nahradit položku
     kopie v nastavení dokumentu za novou částku
     If
          Cstr(doc.MON_PF_IS_UhradaCastka_copy(0))<>Cstr(doc.MO
          N_PF_IS_UhradaCastka(0))
          Then
          doc.MON_PF_IS_UhradaCastka_copy=doc.MON_PF_IS_UhradaC
          astka
          c=True
     End If

'pokud se uhrazená částka změnila, je třeba nahradit položku
     kopie v nastavení dokumentu za novou částku
     If
          Cstr(doc.MON_PF_IS_UhradaCastkaUhrazeno_copy(0))<>Cst
          r(doc.MON_PF_IS_UhradaCastkaUhrazeno(0))
          Then
          doc.MON_PF_IS_UhradaCastkaUhrazeno_copy=doc.MON_PF_IS
          _UhradaCastkaUhrazeno
          c=True
     End If
```

```
'pokud se zbylá částka k úhradě změnila, je třeba nahradit
     položku kopie v nastavení dokumentu za novou částku
     If
          Cstr(doc.MON_PF_IS_UhradaCastkaZbyva_copy(0))<>Cstr(d
          oc.MON_PF_IS_UhradaCastkaZbyva(0))
          Then
          doc.MON_PF_IS_UhradaCastkaZbyva_copy=doc.MON_PF_IS_Uh
          radaCastkaZbyva
          c=True
End If
```

Kontrola změn v uložených kopiích polí týkajících se bankovního účtu probíhá podle diagramu na obr. 26. Funkce Trim zajistí vrácení upravené hodnoty její proměnné bez případných mezer. Pokud po importu dat chybí údaj o kódu banky nebo samotném čísle účtu, je třeba použít údaje nastavené před importem (předpokládá, že byly vyplněny, pokud ne, pole budou nadále prázdná):

```
'Kontrola změn v uložených kopiích polí týkajících se účtu
     If Trim(Cstr(doc.MON_AdUcBan_IMP_KODBANKY(0)))=""
          Then
          doc.MON_AdUcBan_IMP_KODBANKY=doc.MON_AdUcBan_IMP_KODB
          ANKY_copy
          c=True
     End If

     If Trim(Cstr(doc.MON_AdUcBan_IMP_UCET(0)))=""
          Then
          doc.MON_AdUcBan_IMP_UCET=doc.MON_AdUcBan_IMP_UCET_cop
          y
          c=True
     End If
```

Pokud v dokumentu nastavujícím pole formuláře proběhly změny, bude příznak „c“ nastaven na hodnotu True a změněný dokument se uloží:

```
     If c=True Then
          Call doc.save(True,False)
     End If

     DalsiZpracovaniDokumentu=True
End Function

End Class
```

6.3 Třída Update_zakazek_odbc

Hlavní funkcí této třídy je opět funkce do_update(). Ta nastaví dotaz pro vyhledání položek z tabulky tblZakazky a pohled pro vyhledání dokumentu na „importy_zakazky“. Klíčovým polem bude ZakazkaID. V proměnné seznampoli nastaví pole, která se budou přenášet z tabulky do nastavených polí pohledu „importy_zakazky“ databáze v Lotus Notes podle diagramu na obr. 28.

Použité symboly „->“ a „|“ definuje třída ODBC ve funkci ConnectToContact, kterou používá funkce RunUpdate. Definuje význam posloupnosti znaků „->“ jako „přemísti položku z prvního pole do druhého pole“. Rovněž definuje znak „|“ jako oddělovač pro dvojice polí. Zároveň znak „$“ určuje, podle kterých počátečních položek se budou načítat ty ostatní. Pak nastaví proměnnou justUpdate na False a removeDocs na True, takže původní dokument se podle třídy ODBC celý přepíše a jeho kopie se smaže. Pokud bude removeDocs na True a zároveň Searchformula nastavená jako prázdný řetězec, formula pro výběr dokumentů, které se budou aktualizovat se podle ODBC převezme z nastavení klíčového pohledu. Proměnná compute_with_form slouží k nastavení, aby se následně provedla kontrola, jestli data umístěná do formuláře jsou podle jeho nastavených požadavků na vyplňování polí:

```
Class update_zakazek_odbc As ODBC
      Sub New
      End Sub

Function do_update()
      Me.ODBCSource ="K4"
      Me.ODBCSQL="SELECT * from tblZakazky"
      Me.keyViewName="importy_zakazky"
      Me.SearchKey="ZakazkaID"
      Me.seznampoli="$CisTypy->form|$ZAKAZKY-
            >kategorie|ZakazkaNazev->value|ZakazkaID-
            >zakazka_id|ZakazkaCislo->zakazka_cislo"
      Me.justUpdate=False
      Me.removeDocs=True
      Me.Searchformula=""
      Me.compute_with_form=True
      Call RunUpdate
      Call con.Disconnect
End Function

Function DalsiZpracovaniDokumentu(doc As NotesDocument) As
      Boolean
      DalsiZpracovaniDokumentu=True
End Function

End Class
```

Ostatní popisované třídy jsou založany na principu popsaném v této kapitole. Používájí ale jiné tabulky a jiná pole těchto tabulek, proto uvádím popis každé další třídy zvlášť.

6.4 Třída Update_partneri_odbc

Třída nejprve nastaví dotaz pro vyhledání položek z tabulky tblObchodniPartneri a pohled pro vyhledání dokumentu na „importy_partneri“. Klíčovým polem bude ObchodniPartnerID. V proměnné seznampoli nastaví pole, která se budou přenášet z tabulky do nastavených polí pohledu „importy_partneri“ databáze v Lotus Notes podle diagramu na obr. 29. Pak nastaví ostatní proměnné pro funkci RunUpdate třídy ODBC tak, jak je popsáno v podkapitole 6.3:

```
Class update_partneri_odbc As ODBC
      Sub New
      End Sub

Function do_update()
      Me.ODBCSource ="K4"
      Me.ODBCSQL="SELECT * from tblObchodniPartneri"
      Me.keyViewName="importy_partneri"
      Me.SearchKey="ObchodniPartnerID"
      Me.seznampoli="$CisTypy->form|$PARTNER-
            >kategorie|ObchodniPartnerID-
            >unid|ObchodniPartnerNazev->value|ObchodniPartnerIC-
            >ico|ObchodniPartnerDIC->dic|ObchodniPartnerUlice-
            >adresa|ObchodniPartnerObec->mesto"
      Me.justUpdate=False
      Me.removeDocs=True
      Me.Searchformula=""
      Me.compute_with_form=True
      Call RunUpdate
      Call con.Disconnect
End Function

Function DalsiZpracovaniDokumentu(doc As NotesDocument) As
      Boolean
      DalsiZpracovaniDokumentu=True
End Function

End Class
```

6.5 Třída Update_osob_odbc

Třída nejprve nastaví dotaz pro vyhledání položek z tabulky tblPersonalOsoby a pohled pro vyhledání dokumentu na „importy_osoby“. Klíčovým polem bude PersonalOsobaID. V proměnné seznampoli nastaví pole, která se budou přenášet z tabulky do nastavených polí pohledu „importy_osoby“ databáze v Lotus Notes podle diagramu na obr. 30. Pak nastaví ostatní proměnné pro funkci RunUpdate třídy ODBC tak, jak je popsáno v podkapitole 6.3:

```
Class update_osob_odbc As ODBC
      Sub New
      End Sub

Function do_update()
      Me.ODBCSource ="K4"
      Me.ODBCSQL="SELECT * from tblPersonalOsoby"
      Me.keyViewName="importy_osoby"
      Me.SearchKey="PersonalOsobaID"
      Me.seznampoli="$CisTypy->form|$OSOBY-
            >kategorie|PersonalOsobaID-
            >unid|PersonalOsobaPrijmeniJmenoTitul-
            >prijmeni|PersonalOsobaInicialy-
            >zkratka|PersonalOsobaSR-
            >PersonalOsobaSR|PersonalOsobaMZ-
            >PersonalOsobaMZ|PersonalOsobaSchvaluje-
            >PersonalOsobaSchvaluje|PersonalOsobaLikviduje-
            >PersonalOsobaLikviduje|PersonalOsobaLotusNotesName-
            >PersonalOsobaLotusNotesName"
      Me.justUpdate=False
      Me.removeDocs=True
      Me.Searchformula=""
      Me.compute_with_form=True
      Call RunUpdate
      Call con.Disconnect
End Function

Function DalsiZpracovaniDokumentu(doc As NotesDocument) As
      Boolean
      DalsiZpracovaniDokumentu=True
End Function

End Class
```

6.6 Třída Update_oddilu_odbc

Třída nejprve nastaví dotaz pro vyhledání položek z tabulky tblZakazkyOddily a pohled pro vyhledání dokumentu na „importy_oddily“. Klíčovým polem bude ZakazkaOddilID. V proměnné seznampoli nastaví pole, která se budou přenášet z tabulky do nastavených polí pohledu „importy_oddily“ databáze v Lotus Notes podle diagramu na obr. 31. Pak nastaví ostatní proměnné pro funkci RunUpdate třídy ODBC tak, jak je popsáno v podkapitole 6.3:

```
Class update_oddilu_odbc As ODBC
      Sub New
      End Sub

Function do_update()
      Me.ODBCSource ="K4"
      Me.ODBCSQL="SELECT * from tblZakazkyOddily"
      Me.keyViewName="importy_oddily"
      Me.SearchKey="ZakazkaOddilID"
      Me.seznampoli="$CisTypy->form|$ODDILY-
            >kategorie|ZakazkaOddilSeID-
            >ZakazkaOddilSeID|ZakazkaOddilSeZkratka-
            >ZakazkaOddilSeZkratka|ZakazkaOddilSRPersonalOsobaIni
            cialy->schvalovatel_uhrady_zkratka|ZakazkaOddilNazev-
            >value|ZakazkaOddilID-
            >oddil_id|ZakazkaOddilZakazkaCislo-
            >oddil_zakazka_cislo|ZakazkaOddilCislo-
            >oddil_cislo|ZakazkaOddilSRPersonalOsobaID-
            >ZakazkaOddilSRPersonalOsobaID|ZakazkaOddilMZPersonal
            OsobaID->ZakazkaOddilMZPersonalOsobaID"
      Me.justUpdate=False
      Me.removeDocs=True
      Me.Searchformula=""
      Me.compute_with_form=True
      Call RunUpdate
      Call con.Disconnect
End Function

Function DalsiZpracovaniDokumentu(doc As NotesDocument) As
      Boolean
      DalsiZpracovaniDokumentu=True
End Function

End Class
```

6.7 Třída Update_spz_odbc

Třída nejprve nastaví dotaz pro vyhledání položek z tabulky tblUcto_C_SPZ a pohled pro vyhledání dokumentu na „importy_spz“. Klíčovým polem bude SPZID. V proměnné seznampoli nastaví pole, která se budou přenášet z tabulky do nastavených polí pohledu „importy_spz“ databáze v Lotus Notes podle diagramu na obr. 32. Pak nastaví ostatní proměnné pro funkci RunUpdate třídy ODBC tak, jak je popsáno v podkapitole 6.3:

```
Class update_spz_odbc As ODBC
      Sub New
      End Sub

Function do_update()
      Me.ODBCSource ="K4"
      Me.ODBCSQL="SELECT * from tblUcto_C_SPZ"
      Me.keyViewName="importy_spz"
      Me.SearchKey="SPZID"
      Me.seznampoli="$CisTypy->form|$SPZ->kategorie|SPZID-
            >unid|SPZNazev->value|SPZZkratka->zkratka"
      Me.justUpdate=False
      Me.removeDocs=True
      Me.Searchformula=""
      Me.compute_with_form=True
      Call RunUpdate
      Call con.Disconnect
End Function

Function DalsiZpracovaniDokumentu(doc As NotesDocument) As
      Boolean
      DalsiZpracovaniDokumentu=True
End Function

End Class
```

7 Zajištění podkladů pro údržbu informačního systému s využitím ITIL

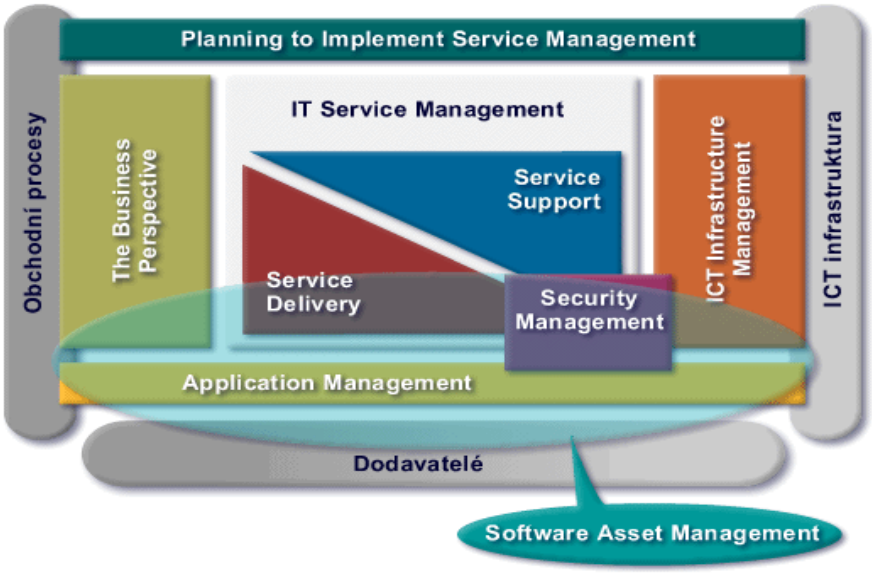
7.1 Návrh údržby informačního systému

Firemní informační systém, kterým se zabývá tato práce, se nachází ve fázi postupného nasazování jednotlivých modulů. Některé jsou tedy již plně nasazeny, některé jsou ve fázi testování, některé ještě ve fázi návrhu a místo nich se používají moduly předchozího systému. V informačním systému tedy neustále probíhají změny, ať ty vývojové, nebo ty vztahující se k modulům, které již jsou nasazeny v plném provozu. Změny v systému nejsou dokumentovány a tak nastávají incidenty z důvodu nedostupnosti některé služby informačního systému. Změna jedné části totiž může způsobit změnu v části jiné, ale tyto možné důsledky snad dokáže odhalit jen dodavatel celého systému.

Proto navrhuji pro řízení změn v informačním systému použít proces Change management metodiky ITIL zaměřené na správu služeb v oblasti informačních technologií. Change Management je proces používající standardizované metody a procedury k efektivnímu a rychlému vyřízení změn. Jeho účelem je minimalizovat vznik incidentů z důvodu změny. Proces managementu změn se zabývá zajištěním, aby byly prováděny jen takové změny, které mají smysl pro celou firmu a přinášejí nějaké výhody – vyšší výnosy, úspory nákladů, eliminaci rizik. Pokud změny nejdou zdůvodnit nebo jsou důvody typu nevyhnutelný trend v IT nebo že finančně to sice vyčíslit nelze, ale kvalita bude podstatně vyšší, nejsou to podle Change managementu důvody ke změně. IT oddělení jako dodavatel služeb a management podniku musí najít ve změně kvantifikovatelný přínos pro celou organizaci.

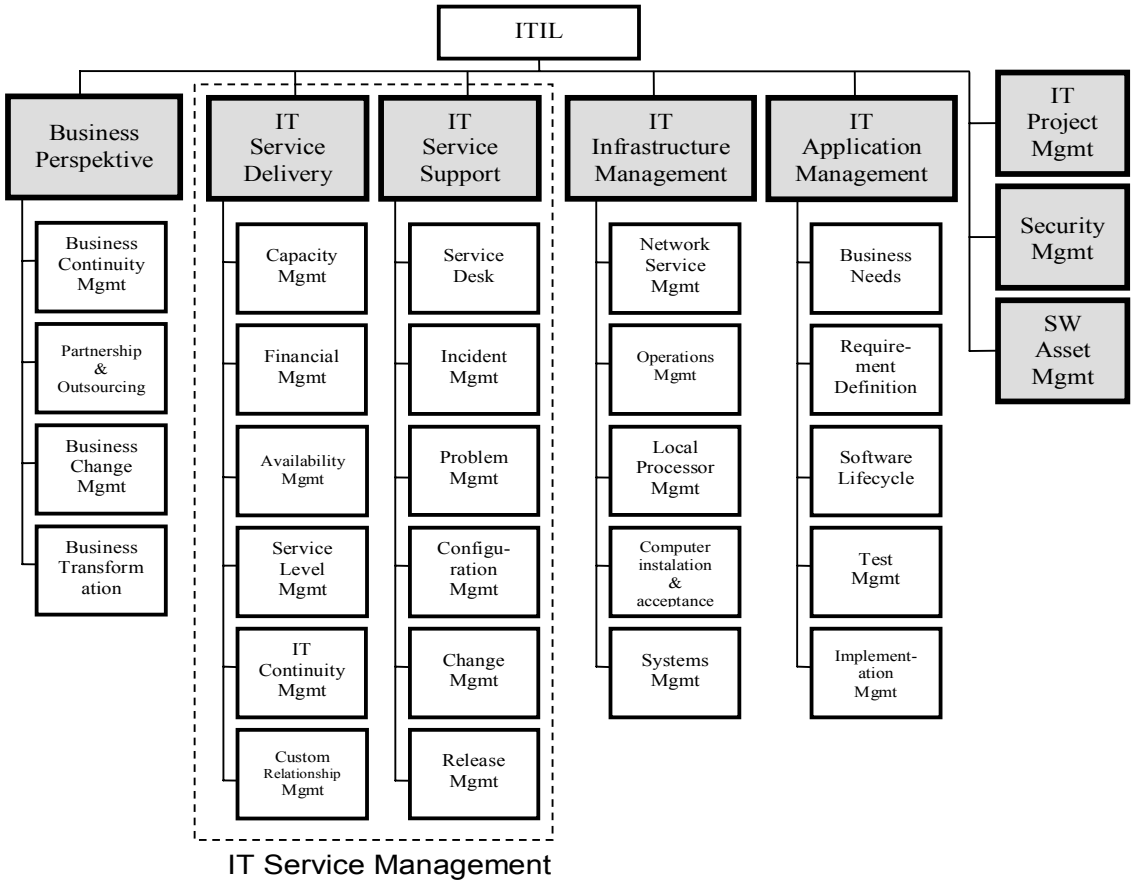
7.2 Charakteristika metody ITIL

ITIL je zkratka anglického názvu *Information Technology Infrastructure Library*. V překladu to znamená knihovna infrastruktury informačních technologií. Podle lit. [16], ze které byla převzata tato podkapitola, je knihovna ITIL je dnes spravována organizací The Office of Government Commerce (OGC) a šířena formou knih, CD, školení, konzultací a certifikací. Je podporována celou řadou školicích firem (např. Pink Elephant, Quint Wellington Redwood a celá řada dalších) a softwarových nástrojů. Podle internetových zdrojů (www stránek uvedených školících firem a v česku např. podle lit. [17], [18], [19]) je mezinárodně uznávaným standardem založeným na „Best Practices“, nejlepších zkušenostech z praxe řízení IT služeb. Hlavním cílem ITIL je dosáhnout požadovanou kvalitu IT služeb za odpovídající cenu. Vychází z britských norem BS 15000, ale opírá se o další kvalitativní standardy jako jsou normy ISO 9001 a European Foundatíon for Quality Management (EFQM). Jednotlivé knihy ITIL popisují CO se má udělat. Způsob provedení, tj. JAK se to má dělat, je potom závislý již na vlastní organizaci implementující ITIL, na její velikosti, interní kultuře atd. Vzájemné vztahy všech osmi publikací ITIL a jednotlivých jejích publikací k obchodním procesům na jedné straně a ICT infrastruktuře na straně druhé ukazuje schéma na obr. 33.



Obr. 33 Záběr metody ITIL

ITIL obsahuje osm základních oblastí, které jsou úzce propojeny (viz Obr. 33). Pod těmito oblastmi jsou pak definovány jednotlivé procesy:



Obr. 34 Procesy ITIL

7.3 Návaznost dokumentace na ITIL

Metoda ITIL je založená na kdykoliv dostupném a dobře zdokumentovaném přehledu funkcí všeho, čeho se týká. Potřebuje tedy také dokumentaci IT infrastruktury. Zvládnout rozsáhlou IT infrastrukturu ve firmě vyžaduje dokonalé zmapování jejích jednotlivých prvků. Znamená to mít informace o kterémkoliv článku tohoto řetězce v kterémkoliv okamžiku. Výsledkem dobré dokumentace a zmapování IT infrastruktury jsou snadno zvládnutelné dříve těžké situace, jako např. řešení krizových situací, rozšiřování IT infrastruktury nebo její audit. Uplatnění ITIL principů v IT oddělení velmi usnadňuje jeho audit, díky důraznému uplatňování procesního přístupu. To umožňuje definování základních IS/IT procesů, jejich náležitostí, vazeb, úrovní vyspělosti atd., jak uvádí lit. [22]. Dokumentace podle ITIL by měla např. obsahovat:

- cíle a přínosy procesů
- role lidí v procesech, jejich odpovědnosti a kontakty na ně
- rozhodovací pravomoci
- priority
- rizika, jejich pravděpodobnost a návrh řešení
- záruky a úrovně kvality
- technické specifikace
- nároky na zdroje - externí a interní

Prořízení změn je třeba mít také definováno:

- zodpovědné osoby
- postupy pro provádění
- zohlednění změn v ceně a čase
- termíny změn
- kritické cesty

O dokumentaci pro ITIL pojednává lit. [23] a [25].

7.4 Návrh opatření pro implementaci Change managementu

Základem pro řízení změn v IT je zmapování podnikové IT infrastruktury. K tomu jsou již vyvinuty mapovací nástroje, které mohou automaticky rozpoznat, z jakých objektů se dané podnikové IT skládá. Zmapují přitom nejen infrastrukturu, sítě, servery a aplikace, ale také uživatele, databáze, databázové instance, otevřené porty na firewallech atd. Zjištěné podrobné údaje uloží do konfigurační databáze a poté sledují používání a chování jednotlivých objektů. Cílem nástrojů pro mapování aplikací je přitom zjistit, jakým způsobem spolu jednotlivé objekty komunikují, jaké jsou prováděny transakce, jak fungují aplikace a které z nich se provádění transakcí účastní. Zjišťují také vzájemné vazby jednotlivých objektů při vykonávání podnikových procesů, jejich hierarchickou provázanost. Výsledkem tohoto monitoringu je databáze s přesným popisem jednotlivých IT objektů, jejich činností a vazeb mezi nimi ve všech vrstvách IT infrastruktury, od síťové po aplikační. V databázi jsou navíc jednotlivé infrastrukturní objekty přiřazeny konkrétním business objektům (podnikovým funkcím, podnikovým procesům atd.). Podnik pak může shora vytvořit a nastavit priority využívání a provozu jednotlivých IT objektů podle stupně důležitosti aplikací a

transakcí, které tyto objekty používají. Při špatné funkci nebo selhání určité transakce se dá snadno zjišťovat, které objekty jsou za tuto situaci odpovědné, a nasadit na ně odpovídající diagnostické nástroje. Zároveň může ještě před výměnou nebo upgradem hardwaru zjistit, jaké aplikace a transakce daný hardware využívají, jaké úkoly hardware plní a co bude znamenat jeho vypnutí a opětné zapnutí z hlediska všech provozních vrstev. Nástroje pro mapování aplikací nejen vytvoří statický snímek stavu podnikového IT, ale poté nepřetržitě v reálném čase udržují obraz dění v IT infrastruktuře. Podnik tak může průběžně registrovat změny, k nimž v jeho IT dochází, a reagovat na ně. Z rozdílu provozních snímků před a po změně lze vysledovat, jaké objekty v IT přibyly či ubyly, jak se změnilo jejich nastavení a zároveň jak se změnily vazby mezi nimi, způsob jejich vzájemné komunikace, jejich hierarchie atd. Tyto údaje slouží pro vyhodnocení nákladů na změnu a posouzení jejího dopadu na bezpečnost.

Nad technologickou částí BTO nástrojů funguje jejich procesní část umožňující realizovat procesní workflow změn. Tato část určuje, jak má změna probíhat, kdo ji má iniciovat, kdo ji má schválit, kdo ji má provést či kdo ji má zkontrolovat. Konsolidují a normalizují požadavky na změny a dávají jim jednotnou tvář odpovídající pravidlům ITIL.

Další schopností těchto nástrojů je sledování životního cyklu výkonnosti aplikací jak z technologického, tak podnikatelského hlediska. Tyto nástroje varují uživatele nejen v případě selhání nebo úplné nedostupnosti aplikace, ale i při její zhoršené nedostupnosti nebo špatné kvalitě. Vyskytne-li se nějaký problém nebo pokud je třeba aplikovat změnu, postupuje se od žádosti o testování aplikace přes její změnu a testování této změny k testování celé aplikace z hlediska výkonu.

Nástroje podporují tvorbu plánu, který umožňuje přesně stanovit náklady například na zkrácení odezvy systému na dvacet, deset nebo pět vteřin. Umožňuje také vyhodnotit situaci, kdy daný požadavek není možno pomocí současné aplikace splnit a je třeba napsat novou, či použít úplně jiné řešení. Účelem sledování životního cyklu výkonnosti aplikací je umožnit pomocí kvalitních simulačních nástrojů kvalifikované rozhodování o tom, jaké řešení použít, a po jeho zavedení do provozu prokázat monitoringem reálnost předpokladů a stabilitu aplikace. Poté lze na konkrétním základě realizovat vazby mezi IT a podnikáním, které se dají zakotvit ve formě SLA (Service Level Agreement), dalšího procesu ITIL.

Příkladem popisovaného nástroje může být např. produkt Mercury Application Relationship Mapping. Popisem nástrojů, kterým byla podstatně inspirována tato podkapitola se zabývá lit. [26].

8 Závěr

Grafický popis softwaru je čím dál více používán pro efektivní návrh aplikací. Je dobrým komunikačním prostředkem mezi zákazníkem a dodavatel, rovněž tak je přehledným podkladem pro programátory. Ve své práci jsem se pokusila shrnout všechny používané formy popisu informačních systémů. Grafický popis pomocí unifikovaného modelovacího jazyka UML jsem popsala podrobněji v jeho hlavních charakteristikách. Rovněž jsem se pokusila nastínit jeho použití v metodice Rational Uniffied Process pro unifikovaný návrh aplikací. Tento proces podporuje také použitý modelovací nástroj Enterprise Architect od firmy Sparx Systems.

Pro upřesnění popisovaného firemního informačního systému jsem stručně charakterizovala prostředí IBM Lotus Notes, které bylo zvoleno pro konsolidaci stávajících systémů. Dále jsem charakterizovala jeho návrhové prostředí Lotus Domino Designer rozšířené o Lotus Domino Connectors a jeho programovací jazyk LotusScript.

První cíl této práce, provést objektově orientovanou analýzu požadavků na nový modul informačního systému jsem splnila v kapitole 4. Zde jsem pro popis požadavků použila jak verbální popis, tak popis grafický diagramem případů užití podle notace jazyka UML a modelovacího nástroje Enterprise Architect. Tato kombinace popisu informačních systémů je asi nejběžnější.

V kapitole 4 a 5 jsem splnila také druhý cíl diplomové práce, navrhnout základní algoritmy a datový model v jazyce UML. Navrhla jsem koncepci řešení nového modulu Přijaté faktury pomocí diagramu aktivit a návrhu diagramu tříd. Diagram aktivit ukazuje procesy probíhají v modulu jako seznam aktivit a přechodů mezi nimi. Diagram tříd ukazuje strukturu systému. Zobrazuje 8 navržených tříd jako 8 typů objektů, obsahy těchto tříd a statické vztahy, které mezi nimi existují.

V návaznosti na návrh nového modulu jsem se rozhodla provést analýzu a návrh propojení navrženého modulu se stávajícími samostatnými systémy, konkrétně aktualizaci dat ve formuláři Detaily faktury a provést realizaci těchto navržených algoritmů. Tímto se zabývá kapitola č. 5. Pomocí Use Case diagramu ukazuje požadavky na knihovnu v rámci automatizované výměny dat mezi systémy. Z požadavků jsem navrhla jednotlivé třídy znázorněné pomocí diagramu tříd a činnosti tříd jsem znázornila v jednotlivých diagramech aktivit. V 6. kapitole se pak zabývám realizací těchto navržených algoritmů v prostředí Lotus Domino Designer. Použila jsem jazyk LotusScript a pro výběr dat z ODBC databáze SQL dotazy. Přístup k databázi zajišťuje již hotová knihovna pomocí Lotus Domino Connectoru pro ODBC.

Nový modul Přijaté faktury informačního systému umožní přehlednější správu přijatých faktur pro jednotlivé zakázky a jejich schvalovatele a likvidátory. Automatická aktualizace položek formulářů zajistí snadnou synchronizaci dat mezi modulem informačního systému a účetním softwarem Money. Přehled funkcí systému popsaný pomocí UML umožňuje přenositelnost informací o programovém vybavení. Jasná syntaxe a sémantika modelu bude pomůckou při správě IS nebo při hledání možných chyb ve struktuře. Bude využívána k seznamování zaměstnanců s novým modulem. Z obrazových podkladů snadněji pochopí jeho principy, uvidí jak jednotlivé moduly na sobě závisí a jak je celá struktura vymyšlena. Díky modelu bude systém také flexibilnější. Bude snadněji reagovat na změny vyplývající z rychle se měnících obchodních procesů, které jsou na podpoře informačního systému závislé. Na modelu manažeři snadněji identifikují vhodná místa pro implementaci změněných požadavků, a také všechny navázané procesy, kterých se budou změny týkat. Tím se sníží riziko

nevhodných zásahů do systému a zlepší se rychlost a efektivnost změn a v neposlední řadě také kontrola nákladů na informační technologie. K tomu navrhuji v kapitole 7 využití procesu řízení změn podle metodiky ITIL. Základem pro řízení změn v oblasti IT je zmapování podnikové IT infrastruktury. K tomu navrhuji použití podpůrného mapovacího nástroje, který dokáže automaticky rozpoznat, z jakých objektů se podnikové informační technologie skládají. Jako výsledek monitoringu vznikne databáze s přesným popisem jednotlivých IT objektů, jejich činností a vazeb mezi nimi ve všech vrstvách IT infrastruktury, od síťové po aplikační. Nástroj bude nepřetržitě sledovat a v reálném čase udržovat obraz dění v IT infrastruktuře a jeho procesní část umožňující realizovat procesní workflow změn. Bude konsolidovat a normalizovat požadavky na změny a dávat jim jednotnou tvář odpovídající pravidlům ITIL. Firemní informační systém, kterým se zabývá tato práce, se nachází ve fázi postupného nasazování jednotlivých modulů. Některé jsou tedy již plně nasazeny, některé jsou ve fázi testování, některé ještě ve fázi návrhu a místo nich se používají moduly předchozího systému. Change management umožní udržovat přehled o změnách a jejich dopadu na celou infrastrukturu co se týče dostupnosti aplikací i potřebných finančních prostředků. Tímto je možno splnit poslední cíl, který ukládá analyzovat přínosy nasazení nového modulu a přístupu ITIL.

Přehled funkcí systému popsaný pomocí UML umožní přenositelnost informací o programovém vybavení, nový modul realizovaný podle takového návrhu umožní přehlednější správu přijatých faktur pro zakázky a osoby, které je budou spravovat. Automatická aktualizace položek formulářů zajistí snadnou synchronizaci dat mezi modulem informačního systému a účetním softwarem. Domnívám se že, práce ukázala význam modelování uživatelských požadavků a návrhu algoritmů pro optimální objektově orientovaný návrh informačního systému. Doporučuji, aby byl model dále využíván pro správu vyvíjeného informačního systému a celý systém byl spravován pomocí nástroje pro popsaný proces řízení změn.

Literatura

[1] ARLOW, J., NEUSTAD, I. *UML a unifikovaný vývoj aplikací*. Brno : Computer Press, 2003. 387 s.

[2] VOŘÍŠEK, J. *Strategické řízení informačního systému a systémová integrace*. Praha : Management Press, 1997. 125 s.

[3] BASL, J. *Podnikové informační systémy*. Praha : Grada Publishing , 2002. 200 s.

[4] STEIN, René. Návrh aplikací v jazyce UML. *Interval.cz : Vývoj aplikací, Analýzy* [online]. 2003 [cit. 2008-05-16]. Dostupný z WWW: <http://interval.cz/clanky/navrh-aplikaci-v-jazyce-uml-unified-modeling-language/>

[5] ŠVEŘEPA, Jaromír. Modelování webových služeb v UML. *LBMS, s.r.o.*. 2006, č. 4, s. 7.

[6] MERUNKA, Vojtěch. *Učební text pro univerzitu třetího věku : Metody tvorby informačních systémů*. [s.l.] : [s.n.], 2005. 31 s.

[7] LACKO, Branislav. *Navrhování systémů řízení*. [s.l.] : [s.n.], 2006. 208 s.

[8] PŘIBYSLAVSKÝ, Petr. Jak na IT služby : SLM – service level management. *IT SYSTEMS : IT Governance* [online]. 2007 [cit. 2008-05-11]. Dostupný z WWW: <http://www.systemonline.cz/sprava-it/jak-na-it-sluzby-service-level-management.htm>.

[9] ŠVEŘEPA, Jaromír. Praktický způsob modelování podnikových IS. *IT SYSTEM* [online]. 2003 [cit. 2008-05-11]. Dostupný z WWW: <http://www.systemonline.cz/clanky/prakticky-zpusob-modelovani-podnikovych-is.htm>.

[10] ČÍHA, Milan. Katalog uživatelských požadavků : Užitečná pomůcka pro vývojáře. *IT SYSTEMS* [online]. 2007 [cit. 2008-05-11]. Dostupný z WWW: <http://www.systemonline.cz/sprava-it/katalog-uzivatelskych-pozadavku.htm>.

[11] Unified Modeling Language. *Wikipedie* [online]. 2008 [cit. 2008-05-11]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/Unified_Modeling_Language>.

[12] KOVAL, Filip. Unifikovaný modelovací jazyk UML. *Owebu.cz : seriály* [online]. 2007 [cit. 2008-05-12]. Dostupný z WWW: <http://www.owebu.cz/filozofie/vypis.php?clanek=1770>.

[13] *Sparx Systems : Modeling & Design Tools for your Enterprise* [online]. 2000 [cit. 2008-05-13]. Dostupný z WWW: <http://www.sparxsystems.com.au/>.

[14] *IBM Lotus Notes 7*. [s.l.] : Admira Studio, 2005. 5 s.

[15] *IBM Lotus Domino 7*. [s.l.] : Admira Studio., 2005. 5 s.

[16] WEINLICHOVÁ, Jana. *Návrh zajištění provozu automatizovaných systémů*. [s.l.], 2005. 53 s. Vysoké učení technické v Brně. Bakalářská práce.

[17] AIT Aplikace informačních technologií [online]. 2005 [cit. 2008-05-13]. Dostupné z: < http://www.ait.cz >.

[18] LBMS Let our brains make your success [online]. 2005[cit. 2008-05-13]. Dostupné z: < http://www.lbms.cz >.

[19] Omnicom [online]. 2004 [cit. 2008-05-13]. Dostupné z: <http://www.omnicom.cz >.

[20] VÁŇA, Vladimír . Požadavky na kvalitu IT služeb z pohledu zákazníka. *IT SYSTEMS* [online]. 2006, roč. 2006, č. 5 [cit. 2008-05-16]. Dostupný z WWW: <http://www.systemonline.cz/outsourcing-ict/pozadavky-na-kvalitu-it-sluzeb-z-pohledu-zakaznika.htm>.

[21] BĚHÁVKA, Petr. Svítání ve světě ITIL. *Business World : Implementace* [online]. 2007, roč. 2007, č. 7 [cit. 2008-05-13]. Dostupný z WWW: <http://www.businessworld.cz/bw.nsf/id/vyvoj-oblasti-ITIL>.

[22] Audit informačního systému. *AKA-monitor : ITIL - ITSM - COBIT* [online]. 2006, roč. 2006, č. 9 [cit. 2008-05-13]. Dostupný z WWW: <http://www.akamonitor.cz/ITIL/>.

[23] *Orbit* [online]. 2005 [cit. 2008-05-14]. Dostupný z WWW: <http://www.orbit.cz/>.

[24] *ITIL : IT Service Management* [online]. 2007 [cit. 2008-05-15]. Dostupný z WWW: <http://www.itil.cz/>.

[25] *Cirkva : Wiki* [online]. 2007 [cit. 2008-05-15]. Dostupný z WWW: <http:// wiki.cirkva.net/>.

[26] AMBLER, Martin. Nové metody řízení podnikové informatiky. *IT SYSTEMS* [online]. 2006, roč. 2006, č. 9 [cit. 2008-05-15]. Dostupný z WWW: <http://www.systemonline.cz/clanky/nove-metody-rizeni-podnikove-informatiky.htm>.

[27] *Gratex* [online]. 2000 [cit. 2008-05-15]. Dostupný z WWW: <http://www.gratex.com/cz/pages/services/service.asp?id=15>.

[28] ŘEPA, Václav. *Podnikové procesy – procesní řízení a modelování*. [s.l.] : Grada, 2007. 186 s.

[29] *Koncepce informačního systému*. Interní materiál firmy. [s.l.] : [s.n.], 2008. 18 s.

[30] *IBM : Rational Unified Process* [online]. 2008 [cit. 2008-05-15]. Dostupný z WWW: <http://www-306.ibm.com/software/awdtools/rup/>.

[31] *Microsoft : ODBC-Open Database Connectivity* [online]. 2008 [cit. 2008-05-15]. Dostupný z WWW: <http://support.microsoft.com/kb/110093>.

[32] HUMPOLEC, Martin. Načítání dat z Active Directory. *Blog Martina Humpolce* [online]. 2006, roč. 2006, č. 8 [cit. 2008-05-15]. Dostupný z WWW: <http://martinhumpolec.cz/A55BE2/Blog.nsf/dx/1155206132-nacitani-dat-z-active-directory.html>.

[33] *IBM Developer Works : Application Performance Tuning* [online]. 2008 [cit. 2008-05-15]. Dostupný z WWW: <http://www.ibm.com/developerworks/lotus/library/ls-AppPerfpt1/index.html>.

[34] ŠVEŘEPA, J. Jak na ITIL?. *IT SYSTEMS*. 2004, roč. 6, č. 5, s. 62-63.

[35] *Information Technology Service Management* [online]. 2004 [cit. 2008-05-15]. Dostupné z: < http://www.itsm.cz >.

[36] *The ITIL and ITSM Directory* [online]. [cit. 2008-05-15]. Dostupné z: < http://www.itil-itsm-world.com >.

Seznam příloh