VYSOKÁ ŠKOLA BÁŇSKÁ – TECHNICKÁ UNIVERZITA OSTRAVA

EKONOMICKÁ FAKULTA

KATEDRA SYSTÉMOVÉHO INŽENÝRSTVÍ

Návrh a implementace modulu softwarové aplikace pro firmu malého rozsahu

Design and Implementation of a Software Application Module for a Small Enterprise

Student: Bc. Václav Koběrský

Vedoucí bakalářské práce: Ing. Radek Němec, Ph.D.

Ostrava 2015

VŠB - Technická univerzita Ostrava
Ekonomická fakulta
Katedra systémového inženýrství

Zadání diplomové práce

Student: **Bc. Václav Koběrský**

Studijní program: N6209 Systémové inženýrství a informatika

Studijní obor: 6209T025 Systémové inženýrství a informatika

Téma: Návrh a implementace modulu softwarové aplikace pro firmu malého rozsahu
Design and Implementation of a Software Application Module for a Small Enterprise

Zásady pro vypracování:

1. Úvod
2. Teoreticko-metodické podklady pro návrh a implementaci softwarových aplikací
3. Analýza současného stavu
4. Návrh a implementace modulu softwarové aplikace
5. Závěr

Seznam použité literatury
Seznam zkratek
Prohlášení o využití výsledků diplomové práce
Přílohy

Seznam doporučené odborné literatury:

ARLOW, Jim a Illa NEUSTADT. *UML2 a unifikovaný proces vývoje aplikací*. Brno: Computer Press, 2007. 568 s. ISBN 978-80-251-1503-9.
CONOLLY, Thomas, Carolyn BEGG a Richard HOLOWCZAK. *Mistrovství databáze: profesionální průvodce tvorbou efektivních databází*. Brno: Computer Press, 2009. 584 s. ISBN 978-80-251-2328-7.
LARSON, Brian. *Delivering Business Intelligence with Microsoft SQL Server 2012*. 3rd ed. New York: McGraw-Hill, 2012. 656 p. ISBN 978-0-07-175938-0.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Radek Němec, Ph.D.**

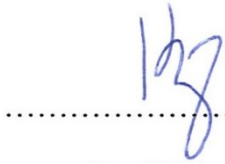
Datum zadání: 21.11.2014
Datum odevzdání: 25.04.2015

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA Ekonomická fakulta OSTRAVA 1

doc. Ing. Jana Hančlová, CSc.
vedoucí katedry

prof. Dr. Ing. Dana Dluhošová
děkanka fakulty

Prohlašuji, že jsem celou práci, včetně všech příloh, vypracoval samostatně.



.....

Bc. Václav Koběrský

V Ostravě dne 25.04.2015

Tímto bych chtěl poděkovat Ing. Radku Němcovi, Ph.D. za cenné rady a odbornou pomoc při vedení této diplomové práce. Děkuji také své rodině a přítelkyni za psychickou i materiální podporu po celou dobu mého studia.

Václav Koběrský

Obsah:

1 Úvod

Tato diplomová práce je zaměřena na řešení problému správy dat ve firmách, kterým se z důvodu malého počtu zaměstnanců, nižšího zisku nebo menšího administrativního zatížení nevyplatí zakoupit hotové softwarové řešení, ale zároveň již nelze kvůli časové náročnosti provádět celou agendu ručně.

Aby i tyto firmy dokázaly pružně reagovat na požadavky trhu, je pro ně důležité zejména snadno, rychle, přehledně a efektivně ukládat data pro potřeby budoucích analýz. Proto je pro tyto firmy výhodnější pořídit si jednoduchou softwarovou aplikaci, která bude přesně odpovídat jejich skromnějším potřebám a bude zahrnovat pouze to, co je opravdu využitelné, než zakoupit hotové softwarové řešení v podobě komplexního informačního systému s velkým množstvím funkcionalit.

Cílem této diplomové práce je navrhnout a implementovat databázi a na ni návaznou intuitivně ovládanou aplikaci, kterou bude moci využívat i méně zkušený uživatel počítače. Aplikace bude sloužit nejenom pro již zmíněné ukládání dat, ale také pro automatickou tvorbu výstupů jako jsou faktury, souhrny zakázek, finanční obrat firmy atd.

Následující kapitoly jsou zaměřeny na návrh, implementaci a realizaci části tohoto řešení. V teoreticko-metodické části této práce jsou nejprve popsány jednotlivé metody tvorby softwaru, jazyk UML, jeho struktura a základní diagramy. Dále jsou rozebrány teoretické podklady databází a použitých vývojových nástrojů. Další kapitola se zabývá analýzou požadavků firmy, srovnáním již existujících informačních systémů a výběrem nejlepšího řešení. Ve třetí části probíhá návrh softwaru pomocí jazyka UML a samotná implementace vybraného řešení.

Za účelem splnění cílů práce je nejprve k získání uživatelských požadavků použita metoda polostrukturovaného rozhovoru s majitelkou firmy. Následuje jejich popis, roztřídění do kategorií a vytvoření jednotlivých případů užití, jejichž konzistence je ověřena pomocí matice sledovatelnosti požadavků. Aplikačně-ověřovací část této práce je rozdělena na tři dílčí podkapitoly, které se při vývoji aplikace vzájemně doplňují. V rámci konceptuálního návrhu je vytvořen slovník dat, ER diagram databáze, komunikační a sekvenční diagram, které popisují důležité funkce aplikace. Dále v logickém návrhu jsou zkontrolovány tabulky pomocí normalizace, je provedena kontrola podpory uživatelských transakcí a kontrola integritních omezení. Nakonec je ve fyzickém návrhu provedena implementace v konkrétním databázovém systému.

2 Teoreticko-metodické podklady pro návrh a implementaci softwarových aplikací

V této kapitole budou uvedeny teoretické podklady pro správné pochopení práce a pro korektní a standardizovaný postup návrhu a implementace softwarové aplikace. Její přečtení by také mělo čtenáři přiblížit myšlenkové postupy, které autor použil při vývoji softwarové aplikace.

Nejprve zde budou obecně popsány postupy, myšlení a způsoby přístupu softwarového inženýrství, databáze, jejich komponenty, fáze životního cyklu jejich návrhu a normalizace. Dále následuje popis jazyka SQL, jeho základních kategorií a příkazů. Není opomenut ani jazyk UML, pomocí kterého je zajištěn unifikovaný proces vývoje aplikace. Nakonec následuje popis veškerého v této práci použitého softwaru.

2.1 Princip softwarového inženýrství

Fritz Bauer definoval softwarové inženýrství takto: *„Softwarové inženýrství je zavedení a používání řádných inženýrských principů tak, abychom dosáhli ekonomické tvorby softwaru, který je spolehlivý a pracuje účinně na dostupných výpočetních prostředcích.“*

Touto definicí je shrnuto vše podstatné, čeho se softwarové inženýrství snaží dosáhnout. Programátor nebo firma zabývající se vývojem softwaru by neměla pouze vyvinout program, dodat toto řešení a dále se o nic nestarat, ale jejich práce by měla zahrnovat i mnoho dalších důležitých aspektů, které jsou uvedeny níže.

Softwarové inženýrství se tedy nestará pouze o implementování inženýrských principů, ale pokouší se také podněcovat k jejich dodržování. Implementováno může být cokoliv, ovšem pokud to nebude řádně využíváno, výsledky nebudou ani z daleka dokonalé. Nejčastěji uváděným příkladem je implementace ISO certifikátu kvality. Tuto certifikaci lze získat po splnění mnoha podmínek, které firmě přinesou určité zlepšení a konkurenční výhodu. Nelze si ovšem myslet, že se firma stane okamžitě úspěšnější a efektivnější. Pokud se firma nebude skutečně starat o kvalitu a její udržování, tak jí k lepším výsledkům žádné certifikáty nepomohou (Kadlec, 2004).

Podmínky úspěšné, ekonomické tvorby softwaru

- Je potřeba vhodně sestavit vývojový tým, který by měl mít přiměřený počet členů v závislosti na složitosti řešeného problému. Měly by v něm být zastoupeny všechny potřebné role a klíčové dovednosti by neměly být v kompetenci jediného člena týmu, aby v případě jeho odchodu neskončil projekt nezdarem.
- Důležitá je volba správného vývojového nástroje, který je obvykle součástí zadání projektu, ovšem v určitých případech je nutné, aby firma navrhla zákazníkovi alternativu. Správná volba tohoto nástroje a vývojového prostředí podle znalostí, podmínek a typu projektu může ušetřit mnoho prostředků a času.
- Je nutné rozhodnout, zda vyvíjet nový systém, nebo zakoupit hotové řešení. V dnešní informační době je velmi pravděpodobné, že se již na trhu nachází komplexní systém, který dokáže uspokojit požadavky zákazníka. V případě rozsáhlých projektů nebo systémů se samozřejmě může jednat pouze o určitou implementovatelnou část nebo komponentu. Tímto rozhodnutím lze firmě ušetřit nemalé prostředky a také snížit riziko neúspěchu projektu.
- Velmi důležitá je komunikace se zákazníkem (zadavatelem projektu). Pokud se tyto dva subjekty nedokáží dohodnout nebo si nerozumí, existuje velká pravděpodobnost odmítnutí kontraktu, sporů při vývoji či dokonce odmítnutí převzetí hotového řešení.
- Je nezbytné uvažovat o budoucí údržbě a případném rozšíření systému již na začátku projektu, aby byly zvoleny správné technologie a postupy (Kadlec, 2004).

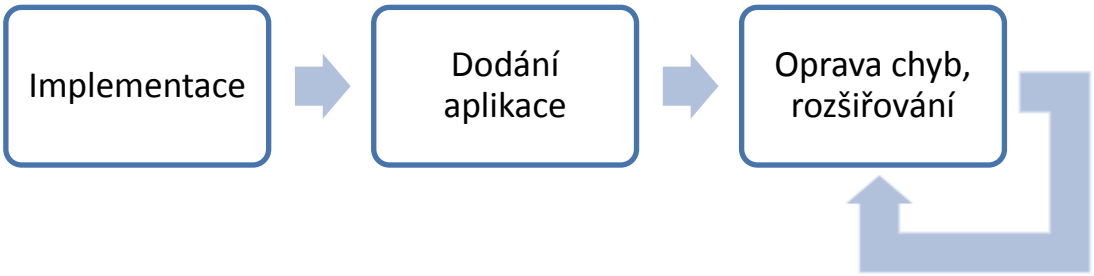
2.2 Metodiky, životní cykly a procesy vývoje softwaru

Metodiky a životní cykly vývoje aplikací jsou jedním z nejdůležitějších přínosů softwarového inženýrství. Stejně jako celý tento inženýrský obor i metodiky procházely určitým vývojem. Vždy bylo snahou jít s dobou a přizpůsobit vývoj softwaru aktuálním požadavkům, které byly na software kladeny v dané době. S tím, že s každým přizpůsobením bylo také žádoucí odstranit nedostatky, které se objevovaly v předchozích postupech (Kadlec, 2004).

2.2.1 Tradiční metody vývoje softwaru

Metoda „napiš a oprav“

V úplných začátcích vývoje programů se používal model „napiš a oprav“ neboli Build and Fix. Postup, který lze vidět na obrázku 2.1, byl založen na naprogramování aplikace a její implementaci, dodání řešení a následném rozšiřování nebo opravě zjištěných chyb. Jak lze již tušit, tento model nebyl příliš efektivní, nicméně v době kdy byl tento postup využíván, vznikaly jednoduché a ne příliš složité programové systémy (Kadlec, 2004).



Obrázek 2.1. Schéma metody „napiš a oprav“ (Zdroj: Kadlec, 2004).

Striktní posloupnost fází

S postupem času se přišlo na to, že vyvíjet software tímto způsobem není dlouhodobě možné. Proto byl již roku 1957 definován tzv. s*tagewise model* životní cyklus, který je založen na striktní posloupnosti jednotlivých fází vývoje a to: definice problému, specifikace požadavků, návrh a architektura, implementace a testování, provoz.

Po ukončení jedné fáze se pokračovalo na další, bez jakékoliv snahy ověřit její průběh, což bylo hlavním nedostatkem tohoto modelu. Nebyly vykonávány žádné revize, nehodnotily se dosavadní výsledky, nerevidovaly se požadavky ani neprobíhala žádná analýza možných rizik. Vývoj postupoval pouze dopředu a nebyl možný návrat nebo znovu vykonání předchozí fáze za účelem upřesnění požadavků a provedení analýzy. Jediná možnost návratu byla až v samotném závěru projektu a to po dodání softwaru, kdy mohla následovat fáze tzv. re-validace, ve které je možno rozhodnout o návratu zpět a opětovném vstoupení do procesu (Kadlec, 2004).

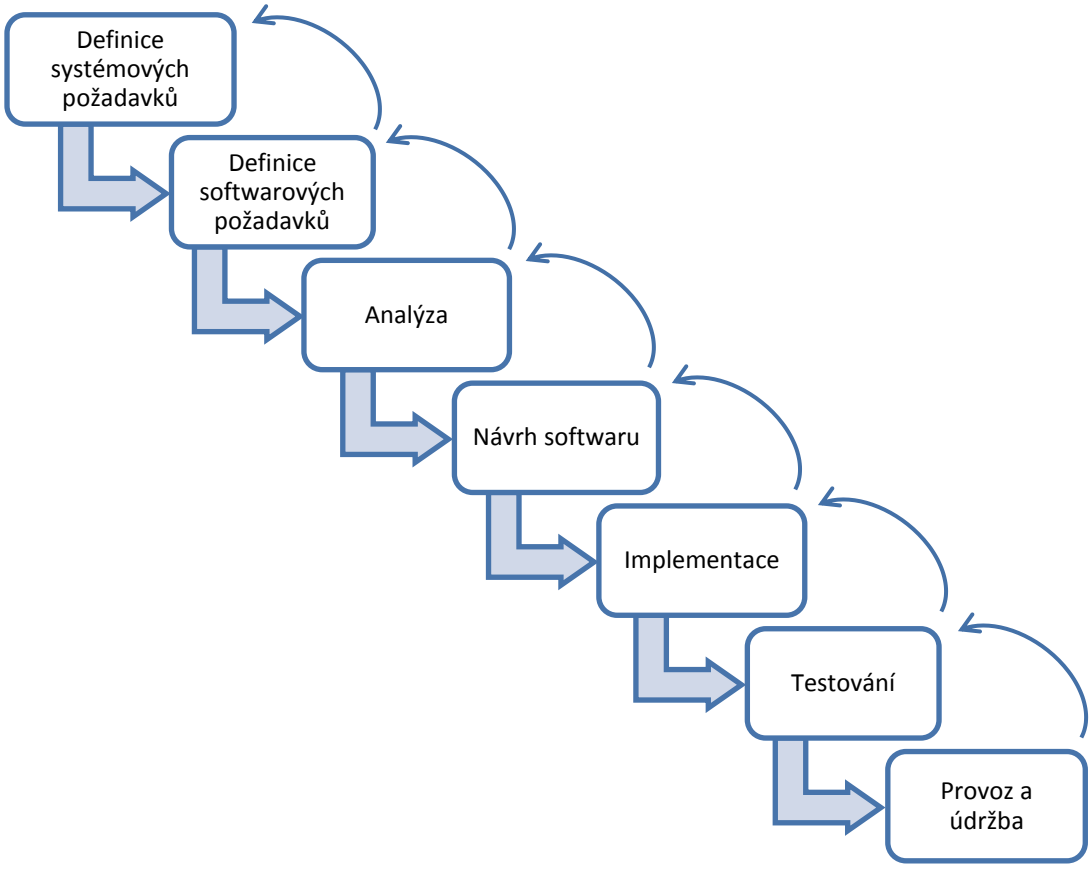
Vodopádový model

Pojmenování tohoto modelu vychází z posloupnosti jednotlivých fází. Poprvé jej v roce 1970 definoval Winston W. Royce. Jedná se v podstatě o vylepšení s*tagewise* modelu s implementací určité zpětné vazby. Do další fáze vývoje lze vstoupit až po dokončení,

uzavření a provedení kontroly fáze předchozí. Jednotlivé posloupnosti vodopádového modelu lze vidět na obrázku 2.2. Při zjištěni problému, např. když výsledek neodpovídá požadavkům nebo očekáváním, je daná fáze po úpravě specifikací provedena znovu.

Má-li být vývoj pomocí tohoto modelu úspěšný, musíme si být na konci každé fáze maximálně jistí její kompletností a správností. Jen tak lze bez problému zahájit následující fáze a dosáhnout tak úspěšného konce projektu (testovanisoftwaru.cz, 2015).

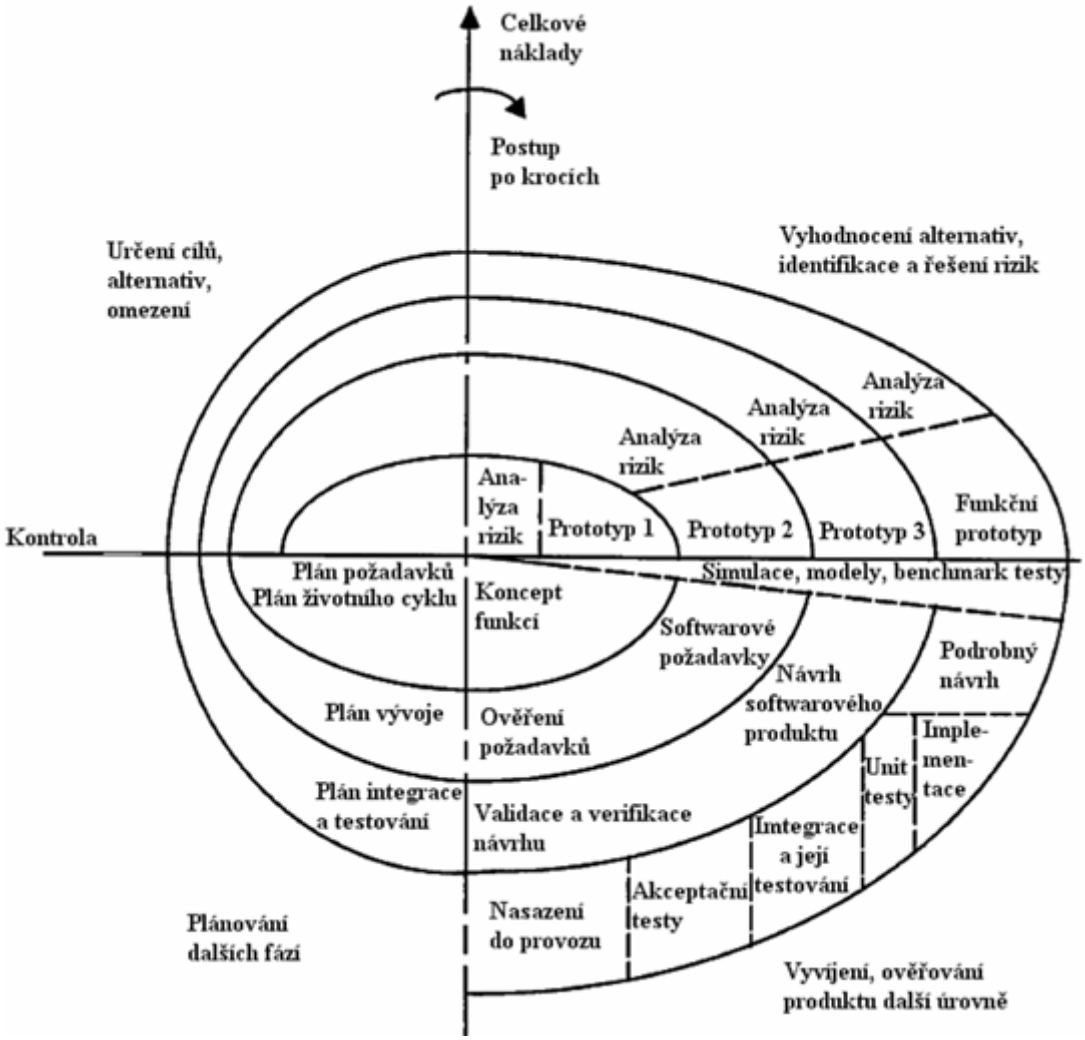
Největšími výhodami vodopádového modelu jsou jasně oddělené fáze, což umožňuje měřit a kontrolovat dosažený pokrok, jeho jednoduchost a možnost vykonat danou fázi znovu. Nevýhodou je, že finální verze produktu je zákazníkovi předána až po jeho úplném dokončení a nemá tedy možnost zasahovat do vývoje a průběžně kontrolovat, zda splňuje jeho očekávání. S tím souvisí také nepružnost při změně podmínek nebo požadavků. Z těchto důvodů je tento model vhodný spíše pro menší a střední projekty, u kterých je kratší i doba trvání (Kadlec, 2004).



Obrázek 2.2. Schéma vodopádového modelu (Zdroj: testovanisoftwaru.cz, 2015).

Spirálový model

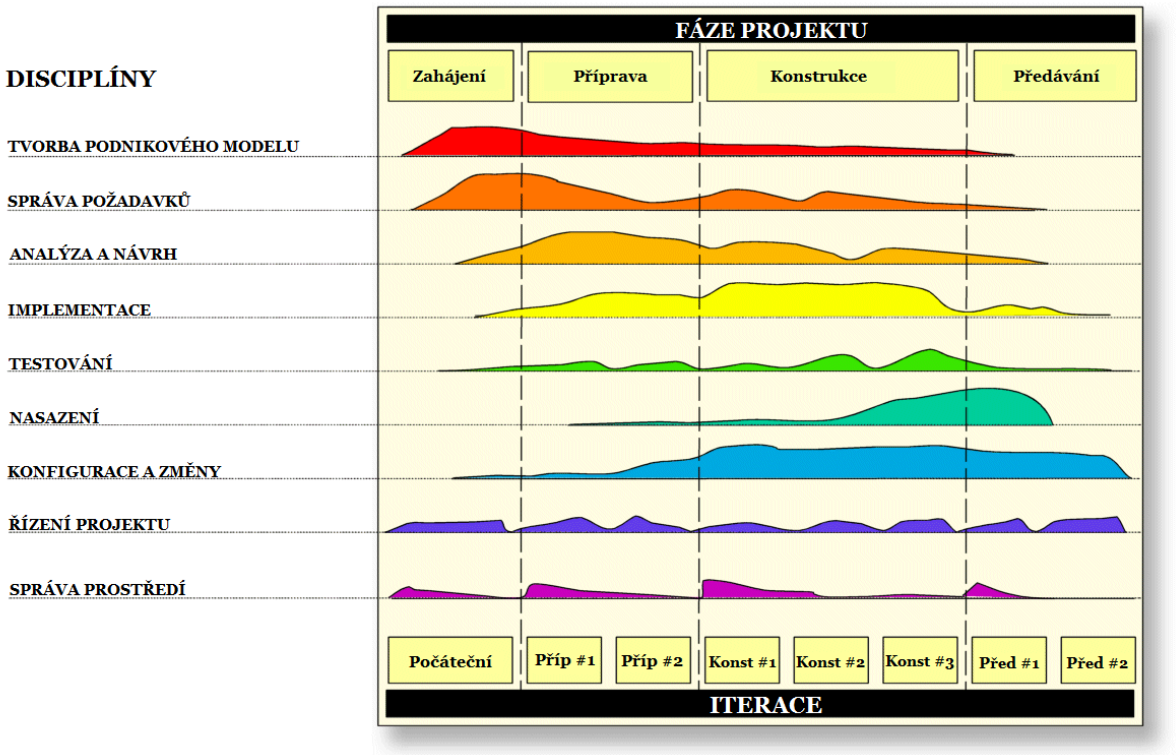
Spirálový model patří do skupiny tzv. přístupů řízených riziky a je založen na iterativním přístupu, bez kterého se žádný vývoj rozsáhlejšího projektu neobejde. Je tím myšleno, že postup do další fáze je umožněn až po důsledné analýze možných problémů a rizik. Integrací těchto kontrolních mechanizmů odstraňuje hlavní nedostatky vodopádového modelu, který byl založen na přímém postupu od definování požadavků, přes analýzu, návrh, implementaci, testovaní, až k dodání, provozu a údržbě. Spirálový model zavádí jiné pořadí provádění těchto fází. Z důvodu toho, že na začátku projektu je velmi obtížné přesně specifikovat všechny požadavky a funkce, stanoví se pouze obecný rámec (někdy ani sám zadavatel projektu není schopen v jeho počátku definovat konkrétně své požadavky). Dále se vyvine první verze aplikace nebo její části, tzv. prototyp, a předloží se zákazníkovi ke kontrole. Po konzultaci se pokračuje další iterací vývoje s upřesněnými požadavky. Celý cyklus je vyobrazen na obrázku 2.3 (Kadlec, 2004).



Obrázek 2.3. Spirálový model vývoje SW (Zdroj: Boehm, 2007 in testovanisoftwaru.cz, 2011).

Rational Unified Process (RUP)

Pro zajímavost je uvedena i tato komplexní komerční metodika vývoje SW, vyvinutá společností Rational, která je založena na objektově orientované iterativní koncepci (schéma je vyobrazeno na obrázku 2.4). Tato společnost prezentuje svůj produkt jako jakéhosi online instruktora poskytujícího metodické pokyny, instrukce, šablony a příklady pro všechny fáze vývoje. Svou propracovaností do hloubky i do šířky se hodí pro rozsáhlé projekty a větší vývojové týmy, které potřebují precizně stanovený a důsledně zdokumentovaný vývojový proces. Na druhou stranu je RUP přizpůsobivý i konkrétním specifickým potřebám projektu i týmu, takže ho lze také využít u projektů menšího rozsahu (Kadlec, 2004).



Obrázek 2.4. Rational Unified Process (Zdroj: projects.staffs.ac.uk, 2015).

Metodika Unified Software Development Process

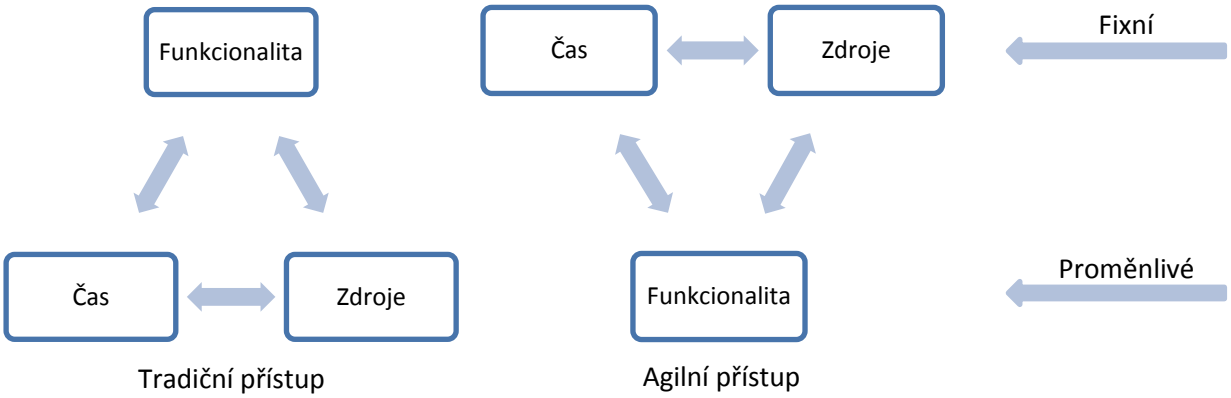
Tato metodika, neboli zkráceně Unified Process (UP), je méně robustní a bezplatná verze metodiky RUP. Unified Process je nejlépe adaptovanou metodikou pro jazyk UML. Proto je podrobněji popsána v kapitole 2.3 UML (Unified Modeling Language).

2.2.2 Agilní metodiky vývoje softwaru

„Tzv. agilní metodiky vývoje SW jsou nové inovativní přístupy, které na jednu stranu sice vychází z ortodoxního softwarového inženýrství, ale na stranu druhou boří mnohá

dogmata a zažité postupy. Jejich cílem je vytvářet SW svižněji, pružněji a efektivněji, nezakopávat se v dílčích vývojových fázích, ale co nejrychleji postupovat k cíli – tj. k dodání fungující aplikace – a lépe tak vyhovět požadavkům dnešní dynamické doby. Agilní metody na rozdíl od těch tradičních, kde se množina požadavků považuje za neměnnou, předpokládají měnící se požadavky na funkcionalitu" (Janků, 2015).

Rozdíl mezi tradičními přístupy a agilními metodikami je nejlépe vidět na následujícím obrázku 2.5. Je to tzv. trojimperativ, trojúhelník, jehož vrcholy vyjadřují tři omezené zdroje, které jsou využívány při realizaci projektu. V tradičních metodikách je kladen důraz na nezměněnou funkcionalitu - i za cenu vyšších nákladů a prodloužení doby trvání projektu musí být splněny všechny specifické požadavky. Při tomto přístupu jsou předem jasně dány všechny funkcionality vyvíjeného systému, avšak je velmi náročné odhadnout časovou a finanční náročnost takto rozsáhlého projektu. Naproti tomu agilní metodiky považují přidělené zdroje a dostupný čas za neměnný. Při náznaku problému proběhne konzultace se zákazníkem, přehodnotí se priority a vyřadí se některé funkcionality, aby byl dodržen časový harmonogram a postačily přidělené zdroje.



Obrázek 2.5. Rozdíl mezi tradičním a agilním přístupem (Zdroj: Kadlec, 2004).

Jak je to s agilním vývojem v praxi, dobře vystihuje následující citace Ondřeje Myslivečka, působícího ve společnosti LMC na pozici head of product: *„Často bývá z pohledu zákazníka výhodnější spokojit se s produktem pokrývajícím požadavky z osmdesáti procent, který vznikl za jeden den, než stoprocentní funkcí produktu získanou na základě pětinásobného úsilí. Důležité je uvést produkt na trh co nejdříve v použitelné podobě a implementovat pouze nezbytné funkce. Zadání pro další rozšíření získává tým díky zpětné vazbě od reálných uživatelů v reálném světě"* (Mysliveček, 2011).

Základní principy agilních metodik:

Iterativní a přírůstkový vývoj s velmi krátkými cykly - při každé iteraci probíhá konzultace se zákazníkem, ukázka nových funkcí a upřesnění jeho požadavků. Nové funkce jsou tedy s novou konfigurací produktu dodávány velmi často. Velkou výhodou tohoto přístupu je, že zákazník je stále informován o postupu vývoje a přesně ví, na čem se momentálně pracuje. Nemůže tedy nastat situace, že by zákazník na konci celého projektu odmítl produkt převzít z důvodu nesplnění zadávacích podmínek.

Důraz na přímou osobní komunikaci v týmu - nefungující komunikace v pracovním týmu způsobuje závažné problémy. Proto se agilní přístupy snaží komunikaci implementovat přímo do procesu vývoje (časté schůze – Scrum meeting, párové programování, apod.) a tím rychleji odhalovat problémy jak v projektu, tak v týmu vývojářů.

Stálá a častá komunikace se zákazníkem – ideálně by měl být zadavatel přímo členem vývojového týmu. Měl by komunikovat s týmem, podílet se na návrhu a také spolupracovat při testování výsledného produktu.

Opakované, přesné, automatizované a průběžné testování – vzhledem k častým vydáváním nových verzí produktu je velmi důležité průběžně ověřovat jejich správnost. Testy by měly být pokud možno napsány ještě před samotnou implementací dané části SW a z důvodu efektivnosti je nejlepší mít testy automatizované (Kadlec, 2004).

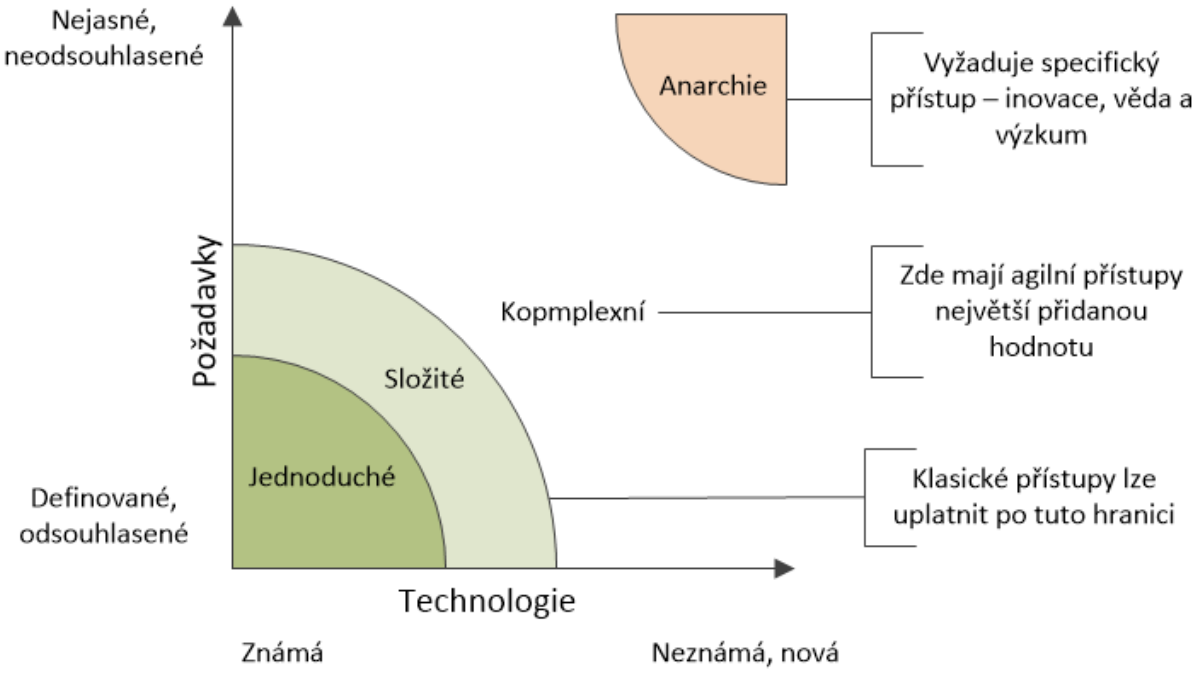
Nejvýznamnějšími agilními metodikami jsou:

- Extrémní programování,
- Scrum,
- Lean Development,
- Vlastnostmi řízený vývoj SW,
- Testy řízený vývoj SW,
- Metodiky Crystal,
- Adaptivní vývoj SW,
- Metoda dynamického vývoje SW.

„Pro agilní přístup se ale nehodí každý projekt. Agilní přístup má největší přidanou hodnotu u projektů kde je nejasné zadání, jedná se o nové technologie, novou oblast podnikání – prostě všude tam kde je velkou výhodou rychlost zpětné vazby od zákazníka, schopnost a ochota ke změně. Naopak v případech, kdy jsou jasné požadavky a požadovaný

výsledek je dobře definovaný, se nejlépe uplatní klasické metody řízení projektů“ (Zephoto, 2014).

Míru uplatnění agilních přístupů v projektech podle složitosti, technologie a požadavků dobře ilustruje následující obrázek 2.6:



Obrázek 2.6. Uplatnění agilních přístupů (Zdroj: Schwaber, 2004 in Zephoto, 2014).

2.3 Unified Modeling Language (UML)

Jazyk UML (Unified Modeling Language, unifikovaný modelovací jazyk) a jeho nástroje budou v této práci použity pro popis komponent návrhu databáze a aplikace.

Je to univerzální jazyk pro vizuální modelování. Nejčastěji se používá pro modelování objektově orientovaných softwarových systémů. Ovšem díky jeho zabudovaným rozšiřovacím mechanizmům má daleko širší využití (Arlow, 2011).

„Tento jazyk byl navržen proto, aby spojil nejlepší existující postupy modelovacích technik a softwarového inženýrství. Diagramy vytvořené v jazyku UML jsou srozumitelné pro lidi, ale navíc je mohou snadno interpretovat i programy CASE (computer-aided sotware engineering)“ (Arlow, 2011, s. 28).

Samotný jazyk UML poskytuje pouze vizuální syntaxi, která se využívá při sestavování modelu a není tedy metodikou, ani není na žádnou konkrétní vázán. Nejlépe adaptovanou metodikou pro jazyk UML je Unified Process.

„Záměrem jazyka UML a metodiky UP byla od jejich vzniku podpora nejlepších postupů používaných v softwarovém inženýrství, vycházejících z ověřených zkušeností posledního desetiletí. K tomuto účelu byly v jazyku UML a metodice UP unifikovány všechny předchozí pokusy o tvorbu jazyků pro vizuální modelování a proces softwarového inženýrství do nejčistšího řešení“ (Arlow, 2011, s. 29).

2.3.1 Objekty v jazyce UML

Základním předpokladem jazyka UML je skutečnost, že umožňuje modelování softwaru, stejně jako dalších systémů, jako kolekce spolupracujících objektů. Tento způsob koresponduje jak s objektově orientovanými SW a programovacími jazyky, tak i s obchodními a podnikatelskými procesy, kde funguje stejně spolehlivě (Arlow, 2011).

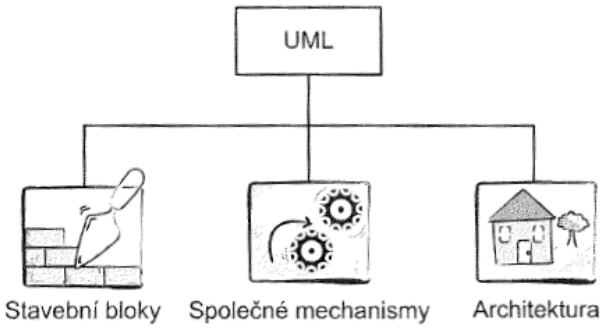
Tyto objekty mohou mít statickou nebo dynamickou strukturu:

- **Statická struktura** – popisuje, jaké typy objektů jsou pro modelování daného systému důležité a jak spolu tyto objekty souvisejí.
- **Dynamické chování** – popisuje životní cyklus daných objektů a způsob, jakým vzájemně spolupracují, s cílem dosažení požadované funkčnosti daného vyvíjeného systému (Arlow, 2011).

2.3.2 Struktura jazyka UML

Vizuální funkci tohoto jazyka nejlépe vyjadřuje následující obrázek 2.7, na kterém je zobrazena jeho struktura. *„Obsahuje tyto součásti:*

- ***Stavební bloky** – jsou to základní prvky modelu, relace a diagramy.*
- ***Společné mechanismy** – obecné způsoby, jimiž se jazyku UML dosáhne specifických cílů.*
- ***Architektura** – pohled v jazyku UML na architekturu navrhovaného systému“* (Arlow, 2011, s. 34).

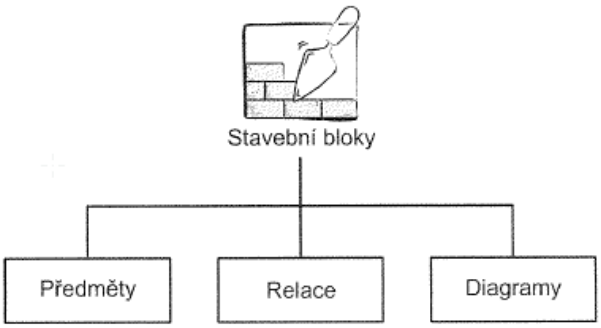


Obrázek 2.7. Struktura jazyka UML (Zdroj: Arlow, 2011, s. 34).

Stavební bloky jazyka UML

„*Tento jazyk je sestaven ze tří stavebních bloků (obrázek 2.8):*

- ***Předměty*** *(things), což jsou samotné prvky modelu,*
- ***Vztahy*** *(relationships), jež jsou pojítkem mezi předměty (relace určují, jak spolu dva nebo více předmětů významově souvisí),*
- ***Diagramy*** *(diagrams), což jsou pohledy na modely UML; ukazují kolekce předmětů, které „vyprávějí příběh“ o softwarovém systému a jsou naším způsobem vizualizace toho, co systém bude dělat (analytické diagramy), a toho, jak to bude dělat (návrhové diagramy)“* (Arlow, 2011, s. 35).



Obrázek 2.8. Stavební bloky jazyka UML (Zdroj: Arlow, 2011, s. 35).

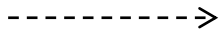
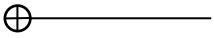
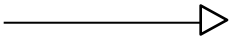
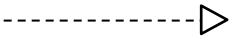
Předměty

Předmětům se říká také věci nebo abstrakce a lze je rozdělit na:

- **Strukturní abstrakce** - jsou podstatnými jmény modelu UML – třídy, rozhraní, spolupráce, případ užití, aktivní třída, komponenta, uzel.
- **Chování** – jsou slovesy modlu UML – interakce, stav.
- **Seskupení** – jsou balíčky používané ke sdružování významově souvisejících prvků modelu do spojitých jednotek.
- **Poznámky** – jsou anotacemi, které zvýrazňují informace sestavené pouze k tomuto účelu (Arlow, 2011).

Relace

Relace v modelu ukazují, jaký je vztah mezi dvěma předměty, a dokážou zachytit jejich významový vztah. Příkladem relace může být rodina a vztahy mezi jejími jednotlivými členy. Stručný přehled typů relací je možno vidět v následující tabulce 2.1.

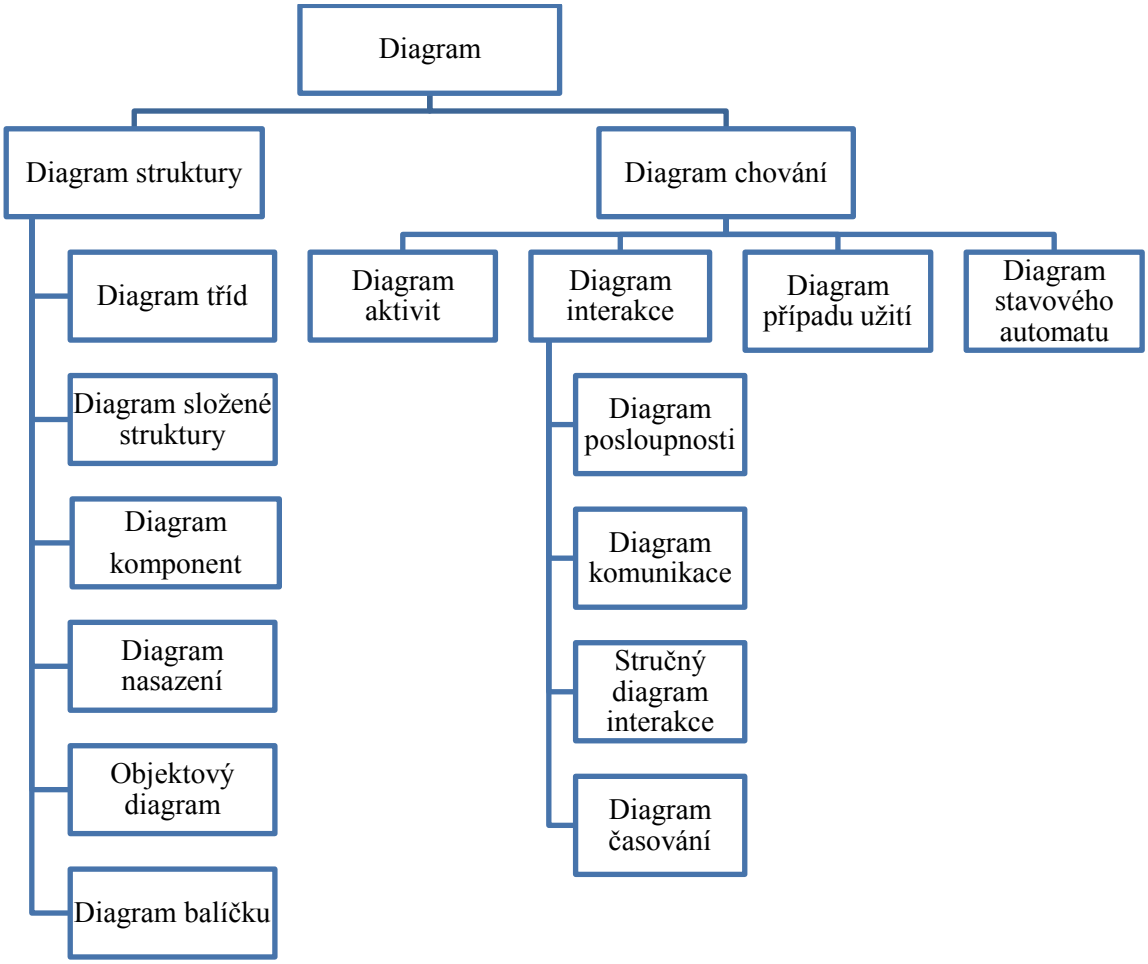
Typ relace	Syntaxe UML		Stručný popis
	zdroj	cíl	
Závislost			Změna v určitém předmětu ovlivňuje význam závislého předmětu.
Asociace			Popis množiny spojení mezi předměty.
Agregace			Cílový prvek je součástí zdrojového prvku.
Kompozice			Silnější forma agregace (má více omezení).
Ochranná nádoba			Zdrojový prvek obsahuje cílový prvek.
Zobecnění			Jeden prvek je specializací jiného prvku a lze jej nahradit obecnějším (univerzálnějším) prvkem.
Realizace			Asociace mezi klasifikátory, kde jeden klasifikátor určuje dohodu, jejíž uskutečnění zaručuje druhý klasifikátor.

Tabulka 2.1. Relace jazyka UML (Zdroj: Arlow, 2011, s. 36).

Diagramy

„Diagramy jsou okna nebo pohledy na model. Diagram není model! V tom je veliký rozdíl, protože předměty a relace lze z diagramu odstranit - lze je odstranit dokonce ze všech diagramů - ale v modelu mohou stále existovat. Ve skutečnosti v něm zůstanou až do té doby, dokud nebudou explicitně vymazány z modelu“ (Arlow, 2011, s. 37).

Celkem existuje třináct různých typů UML diagramů. Všechny jsou vyobrazeny na obrázku 2.9. Tuto množinu diagramů lze dále dělit na ty, které modelují statickou strukturu systému (statický model – zachycuje předměty a strukturní asociace mezi předměty), a na ty, které modelují dynamickou strukturu systému (dynamický model – zachycuje způsob, jakým na sebe na sebe navzájem působí jednotlivé předměty (Arlow, 2011).

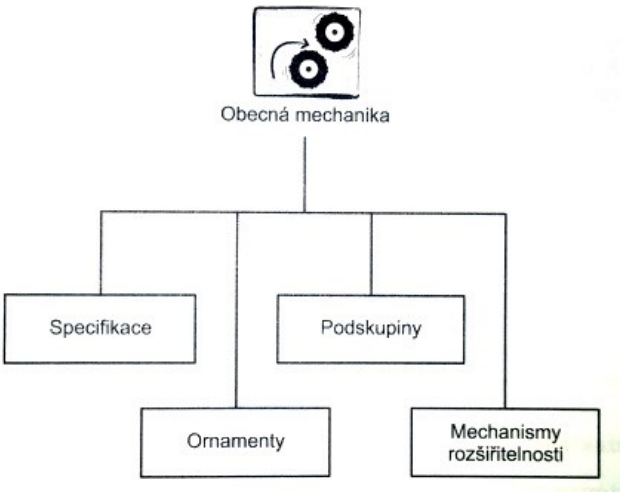


Obrázek 2.9. Diagramy UML (Zdroj: Arlow, 2011, s. 37).

I když neexistuje žádné pevně stanovené pořadí při tvorbě diagramu UML, tak se obecně začíná diagramem případu užití, který definuje rozsah platnosti navrhovaného systému. Často se souběžně pracuje na několika různých diagramech a postupně jsou při zjištění nového detailu vylepšovány (Arlow, 2011).

2.3.3 Obecná mechanika jazyka UML

Na obrázku 2.10 zle vidět čtyři společné mechanismy konzistentně používané v celém jazyku UML. Tyto mechanismy popisují čtyři strategie cesty k modelování objektů, které jsou v tomto jazyce opakovaně používány v různých kontextech (Arlow, 2011).



Obrázek 2.10. Čtyři společné mechanismy UML (Zdroj: Arlow, 2011, s. 39).

Specifikace

Specifikace jsou textové popisy jednotlivých prvků modelu. *„Množina specifikací je jádrem modelu a utváří sémantický podklad, který udržuje celý model pohromadě a dává mu smysl. Různé diagramy jsou různými pohledy nebo obrazovými projekcemi tohoto podkladu“* (Arlow, 2011, s. 40).

Ornamenty

Ornamenty jsou používány pro zahrnutí více informací do diagramu. Slouží tedy k vylepšování modelu. Zvyšují detailnost, ale také složitost, a proto by se měly používat s rozvahou.

„Všechny diagramy UML jsou pouze pohledem na daný model. Měli bychom v nich ornamenty používat pouze v případě, kdy zvyšují srozumitelnost a čitelnost diagramu, případně tehdy, když zdůrazňují určitou důležitou funkci modelu. V diagramu obvykle není potřeba zobrazovat všechny podrobnosti. Mnohem důležitější je, aby byl diagram srozumitelný, aby znázorňoval přesně ty cíle, jichž chceme dosáhnout, a aby byl snadno čitelný“ (Arlow, 2011, s. 41).

Podskupiny

Podskupin jsou v UML dva druhy. První je skupina klasifikátorů a instancí, druhá je skupina rozhraní a implementací. *„Klasifikátor je abstraktním vyjádřením typu předmětu. Instance je naproti tomu konkrétním výskytem abstraktní představy.“* UML odděluje to, co předmět vykonává (jeho rozhraní) od toho, jak to vykonává (jeho implementace). *„Rozhraní tedy definuje dohodu, která zaručuje, čím se budou jednotlivé implementace řídit“* (Arlow, 2011, s. 42).

Mechanismy rozšiřitelnosti

UML je univerzální jazyk navržený tak, aby uspokojil potřeby co nevětšího okruhu uživatelů. Z tohoto důvodu jsou do něj zabudovány tři jednoduché mechanismy rozšiřitelnosti (Tabulka 2.2).

Mechanismy rozšiřitelnosti	Popis
Omezení	Omezující podmínky rozšiřují sémantiku prvku tím, že umožňují přidávat k němu nová pravidla.
Stereotypy	Stereotyp umožňuje definovat nový prvek modelu UML, jenž je založen na existujícím prvku – definuje sémantiku stereotypu. Stereotypy přidávají do metamodelu jazyka UML nové prvky.
Označené hodnoty	Označené hodnoty umožňují rozšířit specifikaci prvku tím, že k takovému prvku přidáme informaci sestavenou jen k tomu to účelu (ad hoc.)

Tabulka 2.2. Mechanismy rozšiřitelnosti jazyka UML (Zdroj: Arlow, 2011, s. 43).

2.3.4 Modelování případů užití

Modelování případů užití je jednou z forem inženýrství požadavků. Umožňuje získávat, specifikovat a dokumentovat systémové požadavky. Je složeno z následujících činností:

- Nalezení hranic modelovaného systému, což je ohraničení kolem případů užití.
- Vyhledání a definování aktérů nebo tzv. rolí přidělených osobám a předmětům, které využívají systém.
- Nalezení případů užití a jejich specifikace. Neboli zaznamenání činností, které mohou aktéři vykonávat se systémem.
- Vytvoření relací mezi aktéry a jednotlivými případy užití.

RTM - Matice sledovatelnosti požadavků

Je tabulka, která mapuje jednotlivé systémové požadavky na případy užití. Pomocí tohoto nástroje je ověřována konzistence. Existuje-li nějaký požadavek, který není mapován na žádný případ užití, znamená to, že v modelu tento případ užití chybí. Toto platí samozřejmě i naopak (Conolly, 2009).

2.3.5 Sekvenční diagramy

„Sekvenční diagramy znázorňují interakce mezi čárami života jako časově uspořádanou posloupnost událostí. Tyto diagramy jsou nejbohatší a nejpružnější formou diagramů interakce“ (Arlow, 2011, s. 253).

Diagramy zachycují spolupráci a chování jednotlivých objektů v případu užití. Postupují shora dolů podle časové osy. Čáry života, které zastupují účastníky interakce, jdou zleva doprava a jsou vodorovné kvůli minimalizaci překřížení čar v diagramu. Jejich umístění znázorňuje okamžik vzniku. Tyto diagram nemají být doslovnými přepisy případu užití, ale mají ilustrovat jeho chování pomocí analytických tříd (Arlow, 2011).

2.3.6 Komunikační diagramy

Zachycují vzájemné vztahy a toky zpráv. Zdůrazňují strukturální aspekty interakce a zprávy mají pořadová čísla pro jasně danou posloupnost. Jsou velmi podobné sekvenčním diagramům, ale zachycují jiné - strukturální aspekty modelovaného systému (Arlow, 2011).

2.4 Databáze a jejich funkcionality

V současné počítačové době nelze navrhnout informační systém nebo různé aplikace bez databáze. Jsou jejich základní součástí. Jejich použití v této práci je tedy nezbytně nutné, a proto musí být uvedena potřebná teorie také k nim.

Výstižnou definici toho, co přesně je to vlastně databáze pronesl Andy Oppel (2008, s. 9): *„Databáze je kolekce vzájemně souvisejících datových položek, které jsou spravovány jako jediná jednotka.“*

Nebo také jak uvedl Thomas Conolly (2009, s. 57): *„Databáze je sdílená kolekce logicky souvisejících dat (a popisu těchto dat), navržená pro plnění informačních potřeb organizace.“*

Pro reprezentaci objektů a událostí reálného světa je používán model dat. Jeho účelem je uchovat data ve srozumitelné podobě.

Jak uvedl Conolly (2009, s. 62): **Model dat** je *„Integrovaná kolekce konceptů pro popis dat, relací mezi daty a omezení používaných dat.“*

„U datového modelu lze uvažovat tři složky:

1. *Strukturální část, která obsahuje množinu pravidel, jak je třeba konstruovat databázi.*

2. *Manipulační část, definující typy operací (transakcí), které jsou na datech přípustné (to zahrnuje operace pro aktualizaci nebo vyvolání dat a operace pro změnu struktury databáze).*
3. *Případně množinu pravidel integrity, která zajišťují, že data jsou přesná"* (Conolly, 2009, s. 63).

Nejpoužívanější je relační model dat, kde jsou data organizována logicky do struktury tabulek. Tento model je používán v mnoha databázových produktech jako např. Microsoft SQL Server, Oracle Database Management System, DB2 nebo také Microsoft Office Access, jehož funkcionality budou využity v této práci.

V systému řízení databáze je možno data organizovat pomocí modelu dat pěti různými způsoby a to lineárním, hierarchickým, síťovým, relačním a objektovým.

Dále jsou podrobněji popsány jednotlivé druhy modelů dat, které jsou využívány v systémech řízení, a také je vysvětlen pojem systém řízení databáze.

2.4.1 DBMS - Systém řízení databáze

„Systém řízení databáze umožňuje uživateli definovat, vytvářet a udržovat databázi a poskytuje řízený přístup k této databázi. Interaguje s uživateli, databázovými aplikacemi a s databází" (Conolly, 2009, s. 38, 39).

Tento systém dále také uživateli umožňuje změnu a přidání dat a také se na tato data pomocí dotazovacího jazyka (nejčastěji SQL) dotazovat. DBMS musí také zabezpečit řízení souběžného přístupu k datům pro více uživatelů (musí zamezit kolizím), také provádí opatření pro zálohu a obnovení databáze a zajišťuje ostatní bezpečnostní mechanismy. V této práci je použit relační systém řízení databáze, a proto bude popsán hlavně tento. Mimo podrobněji rozepsané, existují také hierarchické DBMS a síťové DBMS.

Hierarchický využívá pro ukládání dat invertovanou stromovou strukturu, což umožňuje ukládat na sériová paměťová zařízení (magnetické pásky).

Síťový DBMS *„reprezentuje data jako kolekce záznamů a relace jedna k více jsou reprezentovány jako množiny"* (Conolly, 2009, s. 48).

Relační systém řízení databáze (RDBMS) – Využívá relační model dat. *„Tento datový model je poměrně jednoduchý, umožňuje vysoký stupeň nezávislosti dat a všechna data reprezentuje ve formě tabulek. Klíčovým vývojovým krokem této doby byl SQL, standartní jazyk pro dotazování v relačních databázích. Označují se jako DBMS druhé generace"* (Conolly, 2009, s. 48).

Objektově orientované systémy řízení databáze (OODBMS) a **objektově relační systémy řízení databáze** (ORDBMS) byly vytvořeny z důvodu stále se zvyšující složitosti databázových aplikací a jejich požadavků. Oba se řadí do DBMS třetí generace. *„Základním principem objektové technologie je zásada, že veškerý software by se měl skládat pokud možno ze standartních, opakovaně použitelných komponent“* (Conolly, 2009, s. 429).

Složky prostředí DBMS

„Je možné identifikovat pět hlavních složek prostředí DBMS: hardware, software, data, procedury a osoby.

- *Hardware – počítačový systém (systémy), na němž běží databáze. Rozsah může být od jednoho PC k střediskovému počítači nebo počítačové síti.*
- *Software – software DBMS a databázových aplikací, společně s operačním systémem, včetně síťového softwaru, pokud se DBMS používá na síti.*
- *Data – data fungují jako můstek mezi hardwarovou a softwarovou složkou a lidskou složkou. Databáze obsahuje jak provozní data, tak metadata.*
- *Procedury – instrukce a pravidla, která řídí návrh a používání databáze. Mohou obsahovat instrukce o tom, jak se přihlásit k databázi, vytvářet záložní kopie databáze a zvládat selhání softwaru a hardwaru.*
- *Osoby - sem patří návrháři databáze, správci dat (DA), správci databáze (DBA), aplikační programátoři a koncoví uživatelé“* (Conolly, 2009, s. 41).

2.4.2 Relační databáze a její komponenty

Komponenty relační databáze jsou její dílčí části a řadí se mezi ně tabulky, entity, omezení, pohledy a relace. Pojem relace má obecně více významů, ve vztahu k databázím bude používán ve smyslu spojení mezi jednotlivými tabulkami - *„Množina smysluplných spojení mezi entitami“* (Conolly, 2009, s. 157).

Tabulka *„je hlavní jednotkou pro ukládání dat v relační databázi. Tvoří ji dvourozměrná struktura, která se skládá z řádků a sloupců. Každá tabulka představuje entitu, která se má v databázi znázornit“* (Oppel, 2008, s. 10).

Entita je *„množina objektů se shodnými vlastnostmi, které uživatel identifikuje jako nezávisle existující objekty“* (Conolly, 2009, s. 156).

Omezení *„je pravidlo aplikované na databázový objekt (většinou tabulku nebo sloupec), které nějakým způsobem omezuje přípustné hodnoty dat tohoto databázového objektu. Jakmile se jednou omezení nastaví DBMS je automaticky vynucuje a nelze je obejít“* (Oppel, 2008, s. 13).

- **NOT NULL** – toto omezení v daném sloupci znemožní použití hodnot null. Hodnotu null nelze srovnávat ani s jinou hodnotou null. *„Je to speciální kód, který v databázi nemá žádný jiný význam. Informuje o tom, že hodnota sloupce v daném řádku není známa. Hodnota null neodpovídá chybějícímu údaji, prázdnému řetězci ani nule“* (Oppel, 2008, s. 56).
- **Omezení primárního klíče** - tento klíč zajišťuje jedinečnost hodnot v rámci celé tabulky. Definovat lze pouze na sloupci, který má nastaveno omezení not null. Primární klíč je podrobněji vysvětlen a popsán v kapitole 2.4.5 Normalizace a postup její kontroly.
- **Omezení jedinečnosti** - je-li definován na sloupci, pak musí sloupec obsahovat pouze jedinečné hodnoty - žádné se nesmí opakovat. Rozdíl oproti omezení primárního klíče je ten, že zde může sloupec obsahovat hodnoty null.
- **Referenční omezení** *„vynucuje relaci mezi dvěma tabulkami v relační databázi. Tzn. RDBMS automaticky kontroluje, zda pro všechny hodnoty cizího klíče existuje odpovídající hodnota primárního klíče v nadřízené tabulce“* (Oppel, 2008, s. 14).
- **Omezení CHECK** – *„Ověřuje hodnotu sloupce pomocí jednoduchého logického příkazu (v jazyce SQL). Výsledkem příkazu musí být logická hodnota true nebo false. Výsledek true přitom umožní vložit hodnotu sloupce do tabulky, zatímco při výsledku false je hodnota sloupce odmítnuta s odpovídající chybovou zprávou“* (Oppel, 2008, s. 14).

Dalšími částmi relační databáze jsou **pohledy.** *„Ty poskytují uživatelům databáze mnoho výhod, protože dovolují přizpůsobit data individuálním požadavkům a zpřehledňují zobrazení. Pohled je v zásadě uložený dotaz SQL, na který se lze dotazovat stejně, jako by se jednalo o skutečnou tabulku. Pohledy lze přirovnat k virtuálním tabulkám, protože se chovají jako tabulky i když fyzicky ve formě tabulek neexistují“* (Oppel, 2008, s. 63).

*„**Relace** představují souvislosti mezi tabulkami relační databáze. Zatímco každá relační tabulka může existovat samostatně, databáze jsou především o ukládání souvisejících dat. Pomocí relací lze provázat související tabulky formálním způsobem, který je snadno použitelný, chcete-li ve stejném dotazu spojit data z více tabulek, ale flexibilně přitom zahrnout pouze informace, které vás zajímají. To umožnuje přizpůsobit informace v databázi konkrétním potřebám jednotlivce nebo aplikace, kteří budou databázi využívat.“* (Oppel, 2008, s. 11)

„Relace mezi entitami jsou reprezentovány pomocí mechanizmu primárního a cizího klíče. Při rozhodování zda umístit atribut cizího klíče musíme nejprve identifikovat „rodičovskou“ a „dceřinou“ entitu, kterých se relace týká. Rodičovská entita posílá kopii svého primárního klíče do tabulky, která reprezentuje dceřinou entitu, kde tato kopie funguje jako cizí klíč.“ (Conolly, 2009, s. 233)

Obecně v databázích existuje možnost, vytvořit tři druhy relací. Relace mohou být jedna k jedné (1:1), jedna k více (1:N) nebo více k více (N:M). V databázi založené na relačním modelu dat, která je použita v této práci, lze vytvořit pouze vazby jedna k jedné a jedna k více. Vazby více k více je nutno řešit spojením přes další tabulku. Dále jsou vysvětleny pouze vazby relačního modelu.

Relaci 1:N lze vytvořit, když právě jedna entita v tabulce odpovídá jedné nebo více entitám v jiné tabulce. Pokud právě jedna entita v tabulce odpovídá přesně jedné entitě v jiné tabulce, pak je možno vytvořit relaci 1:1. Např. notebook a jeho výrobní číslo.

2.4.3 Transakce

Podle Thomas Connoly (2009, s. 116) je transakce *„akce nebo řada akcí, prováděná jedním uživatelem nebo databázovou aplikací, která přistupuje k obsahu databáze nebo mění obsah databáze.“*

Transakce jsou tedy základní operací v databázích a jako takové by měly splňovat určité požadavky. Transakce může skončit dvěma způsoby. Buď je v pořádku celá provedena, pak je tzv. *potvrzená*, nebo nebude úspěšně dokončena a tedy bude *zrušena*. V případě zrušení musí být proveden tzv. rollback celé transakce a databáze je navrácena do původního konzistentního stavu. Existují čtyři základní vlastnosti, které jsou od transakcí vyžadovány (tzv. ACID):

- **Atomárnost** (Atomicity) – znamená, že transakce představuje dále nedělitelnou jednotku. Buď je vykonána celá, nebo se neprovede vůbec. Není tedy možné provést pouze její určitou část.

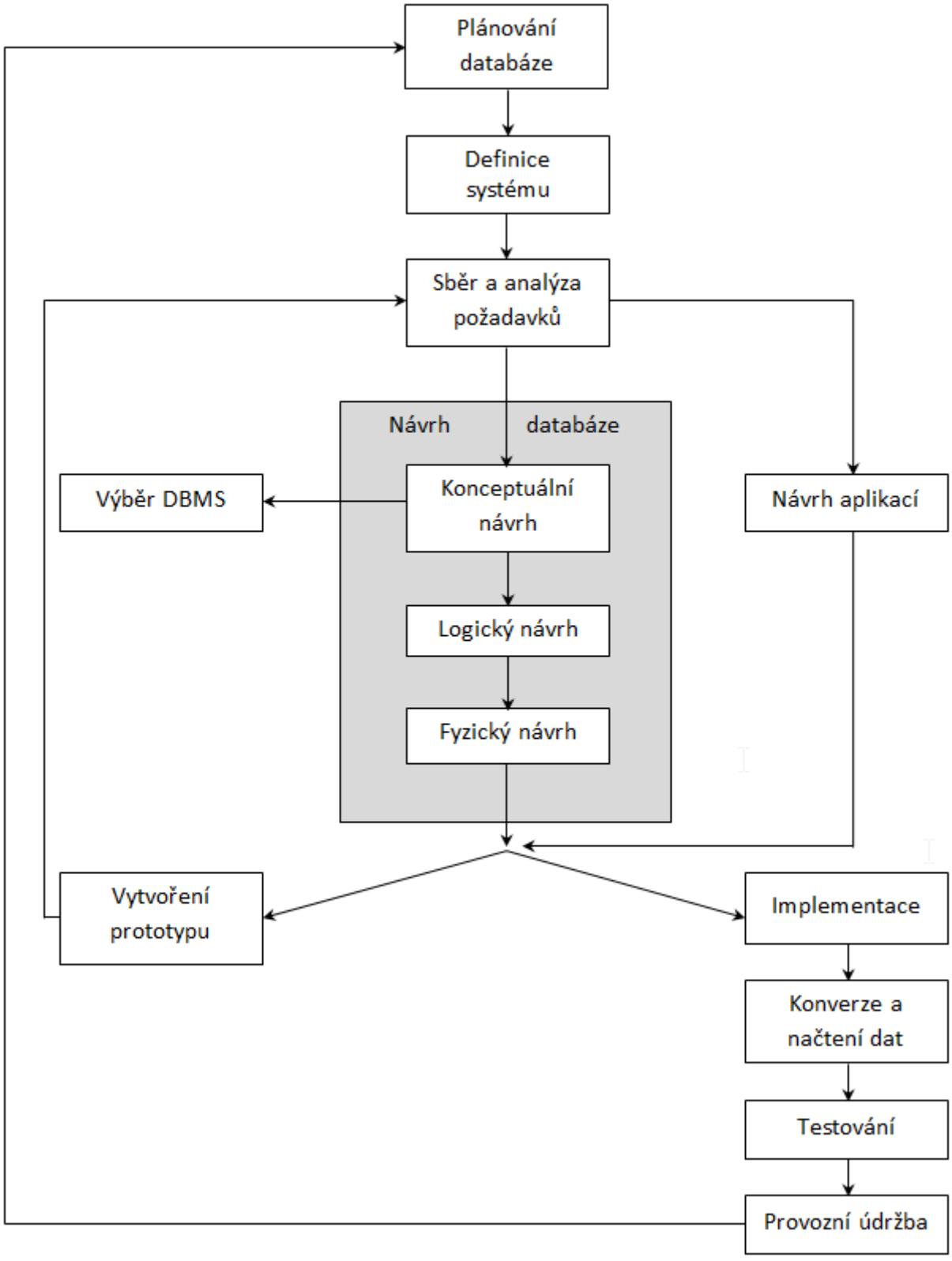
- **Konzistence** (Consistency) – transakce musí převést databázi z jednoho konzistentního stavu do jiného konzistentního stavu. O to se stará DBMS pomocí vynucování integritních omezení, ale také vývojáři při vývoji aplikací.
- **Izolace** (Isolation) – znamená, že jednotlivé transakce musí být na sobě nezávislé. Jedna transakce nemůže mít vliv na průběh jiné transakce.
- **Trvalost** (Durability) – v případě úspěšného provedení transakce musí být tyto změny trvale zapsány do databáze a již nesmí být ztraceny. (Conolly, 2009)

2.4.4 Metodologie vývoje databáze a databázového systému

Postup, který je při vývoji tohoto databázového systému dobré dodržet popisují fáze životního cyklu vývoje. Tyto jednotlivé fáze nemusí být uskutečňovány ve stanoveném pořadí, pokud například vznikne při vývoji nějaký problém, je příhodné využít zpětnou vazbu a daný krok, ve kterém se vyskytl problém vykonat znovu nebo úplně přeskočit. Na obrázku 2.11, lze vidět vývojový diagram životního cyklu vývoje. Nejdůležitější částí tohoto vývoje databázového systému je samotný návrh databáze, ve kterém je vytvořen ER diagram (a další potřebné diagramy návrhu), který zobrazuje entity spolu s relacemi mezi nimi a popis tabulek. Tento diagram rozděluje návrh na tři navazující části: konceptuální návrh, logický návrh a fyzický návrh. (Connoly, 2009)

V rámci **konceptuálního návrhu**, jsou identifikovány důležité entity a relace mezi nimi, které je nutno reprezentovat v databázi. Tento model dat, je vytvářen bez jakýchkoliv úvah o konkrétních podrobnostech relačního modelu dat nebo fyzické implementace. Konceptuální fáze návrhu je zdrojem informací pro **logický návrh**, v němž je mapován ER model na množinu relačních tabulek. Dále jsou definována integritní omezení a struktura každé tabulky je kontrolována pomocí normalizace z důvodu minimalizace redundance. Ve **fyzickém návrhu** je převeden logický popis relačních tabulek do fyzické podoby tabulek podle funkcionalit konkrétního DBMS a jsou implementována integritní omezení. (Connoly, 2009)

Návrh databázových aplikací pro zpracování dat probíhá souběžně s vývojem databáze. Aby nevznikly případné pozdější problémy, např. při získávání dat, je důležité mít již v této fázi na paměti některé aspekty a požadavky aplikace a databázi jim přizpůsobit.



Obrázek 2.11. Fáze životního cyklu vývoje databázového systému (Zdroj: Conolly, 2009, s. 110)

2.4.5 Normalizace a postup její kontroly

„Normalizace je technika pro vytvoření sady tabulek s minimální redundancí, která podporuje datové požadavky organizace“ (Conolly, 2009, s. 188).

Tuto techniku vyvinul již v roce 1972 E. F. Codd. Normalizace optimalizuje převod struktury z ER diagramu, tedy strukturu entit a relací na tabulkové a relační uspořádání v databázi. Jejím cílem je odstranit možné problémy, které mohou nastat při tzv. anomáliích aktualizace. Ta vzniká v případě chybně strukturovaných tabulek při aktualizaci nebo odstraňování dat. Produktem normalizace je také výkonnější a přehlednější databáze.

Normalizace využívá jedinečné identifikátory, které jednoznačně identifikují všechny jednotlivé záznamy v tabulce (žádný řádek tabulky tedy nikdy není stejný). Tento identifikátor je nazýván primárním klíčem a je vybírán z množiny kandidátních klíčů. Může být tvořen jedním sloupcem nebo kombinací několika sloupců tabulky.

Dle Thomas Conolly (2009, s. 66) je **kandidátním klíčem** *„sloupec nebo množina sloupců, které jedinečně identifikují záznam v relaci.*

Tento klíč pro tabulku má dvě vlastnosti:

- *Jedinečnost – v každém záznamu určuje hodnota kandidátního klíče výlučně daný záznam.*
- *Neredukovatelnost – žádná vlastní podmnožina kandidátního klíče nezajišťuje jedinečné určení záznamů.*

Primárním klíčem *je tedy kandidátní klíč, který je vybrán, aby jedinečně určoval záznamy v tabulce.“*

První normální forma

Prvním krokem normalizace je odstranění vícehodnotových atributů, tzn. takových, kde se na průsečíku řádku a sloupce nachází více hodnot. Tím je databáze uvedena do **1NF**.

„První normální forma je tabulka, v níž každý průsečík sloupce a záznamu obsahuje jen jedinou hodnotu“ (Conolly, 2009, s. 191).

Druhá normální forma

„Tabulka, která je v 1NF a ve které jsou hodnoty každého sloupce, který není součástí primárního klíče, determinovány všemi hodnotami sloupců, které tvoří primární klíč“ (Conolly, 2009, s. 192).

Tato normální forma se týká pouze tabulek, ve kterých je primární klíč složen z více sloupců. Tabulka, která je v 1NF a její primární klíč je tvořen pouze jedním sloupcem, je automaticky také v 2NF (Conolly, 2009).

Pro splnění této musí být zajištěna úplná závislost všech neklíčových atributů na primárním klíči (částečné závislosti musí být odstraněny).

Plnou funkční závislostí se rozumí: *„Pokud jsou* a *a* b *sloupce tabulky, pak je* b *plně determinován* a*, pokud* b *není determinován žádnou podmnožinou z* a*. Pokud je sloupec* b *determinován podmnožinou z* a*, mluvíme o částečné závislosti“* (Conolly, 2009, s. 194).

Třetí normální forma

„Tabulka, která již je v 1NF a 2NF a ve které všechny hodnoty ve sloupcích, které nepatří k primárnímu klíči, jsou determinovány pouze sloupci primárního a nejsou determinovány žádnými jinými sloupci“ (Conolly, 2009, s. 195).

3NF je tedy splněna pokud je odstraněna tranzitivní závislost mezi atributy a primárním klíčem. Žádný neklíčový atribut nesmí být tranzitivně závislý na primárním klíči.

„Tranzitivní závislost popisuje vztah mezi sloupci a, b, *a* c*. Pokud* a determinuje b (a → b) *a* b *determinuje* c (b → c), *pak sloupec* c *je tranzitivně závislý na* a *prostřednictvím* b *(pokud* b *nebo* c *nedeterminuje* a*)“* (Conolly, 2009, s. 196).

Boyce-Coddova normální forma

„Poslední prakticky užívanou formou je tzv. ***Boyce-Coddova normální forma*** *(BCNF). Tabulka splňuje BCNF, právě když pro dvě množiny atributů A a B platí: A→B a současně B není podmnožinou A, pak množina A obsahuje primární klíč tabulky. Tato forma zjednodušuje práci s tabulkami. Ve většině případů, pokud dobře postupujeme při tvorbě tabulek, aby splňovaly postupně 1NF, 2NF a 3NF, forma BCNF je splněna“* (Interval.cz, 2000).

Další normální formy

Existují ještě vyšší normální formy než BCNF - 4NF a 5NF, nicméně tyto pokročilé normální formy nejsou obvykle využívány, protože identifikují a řeší málo frekventované problémy (Conolly, 2009).

2.4.6 SQL - Strukturovaný dotazovací jazyk

SQL jazyk je nejrozšířenějším komerčním databázovým jazykem, který je navržen a určen k používání jak odborníky, tak i neodborníky.

„Jazyk SQL připomíná klasický přirozený jazyk (samozřejmě anglický), má však přesně definovaná syntaktická a lexikální pravidla“ (Kiszka, 2009, s. 64).

„SQL je základním jazykem, který slouží ke komunikaci s relačními databázemi. Často se říká, že matematika je jazykem vědy. Obdobně platí, že SQL je jazykem databází“ (Oppel, 2008, s. 7).

SQL je neprocedurálním jazykem. Tzn., že není třeba určovat způsob, jakým je nutno data získat ale stačí uvést, která data jsou potřeba. Toto zajišťuje modul SQL v databázovém systému. Velmi často je využíván v kombinaci s procedurálními nebo objektově orientovanými jazyky. Stejně tak i v této práci, kde je použit pro vytvoření a správu databáze. O zobrazení dat uživateli a jejich přenos do programu Microsoft Access se po propojení s SQL Serverem postarají jejich vnitřní mechanizmy přes ODBC driver.

SQL příkazy jsou rozděleny do několika kategorií podle úkonů, jaké daný příkaz provádí:

- DML - příkazy pro manipulaci s daty,
- DDL - příkazy pro definici dat,
- DCL - příkazy pro řízení dat.

Tyto příkazy jsou složeny z předem definovaných a z uživatelsky volitelných slov. Předem definovaná neboli rezervovaná slova, jsou tvořena jednoduchými anglickými slovíčky. Zvyklostí programátorů je psát tato slova velkým písmem. SQL ovšem velikost písmen nerozlišuje, takže to není striktně vyžadováno. Je to spíše kvůli přehlednosti programového kódu. Tato uživatelsky definovaná slova musí samozřejmě plnit určitá pravidla syntaxe. Konkrétně to mohou být názvy různých databázových objektů jako např. sloupců a tabulek. Rezervovanými slovy nejpoužívanější kategorie **DML** jsou:

- SELECT - pro dotazy na data v databázi,
- INSERT - pro vkládání dat do tabulky,
- UPDATE - pro aktualizaci dat v tabulce,
- DELETE - pro vymazání dat z tabulky. (Oppel, 2008)

K příkazu SELECT se dále vážou klauzule FROM, WHERE, GROUP BY, HAVING, ORDER BY, jejichž pořadí je striktně stanoveno a nelze jej měnit. Tyto klauzule zajišťují specifikaci výběru dat a jejich řazení a třídění.

V kategorii **DDL** jsou používány tyto příkazy:

- CREATE - pro vytvoření nového objektu,
- ALTER - pro změnu existujícího objektu,
- DROP - pro odstranění existujícího objektu.

Poslední zmíněnou kategorii jazyka SQL je **DCL** s těmito příkazy:

- GRANT - příkaz slouží pro přidělení oprávnění uživateli k přístupu k databázovým objektům,
- REVOKE - příkaz slouží pro odebrání oprávnění uživateli k přístupu k databázovým objektům,
- START TRANSACTION - zahajuje transakci,
- COMMIT - potvrzuje transakci,
- ROLLBACK - ruší transakci a vrací databázi do konzistentního stavu, ve kterém byla před neúspěšným provedením transakce.

2.5 Použitý software

Tato podkapitola obsahuje popis veškerého softwaru, který je použit v této práci při návrhu a realizaci vybraného řešení.

2.5.1 Microsoft SQL Server Express Edition

Microsoft SQL Server v Express edici je relační databázový systém, který je dostupný zdarma a slouží pro správu většího množství dat, ať už přes internet, podnikovou síť, nebo na lokálním počítači. S tím, že Microsoft poskytuje tento software zdarma, se vážou určitá omezení např. velikosti databáze a počtu využitelných procesorů. Pro menší objem dat ovšem bohatě postačuje.

Tato edice je vhodná pro vývoj aplikací a jejich nasazení do klientských počítačů, webů a malých serverů. Hlavními funkcemi a možnostmi využití této verze jsou:

- Podporuje 10 GB úložného prostoru pro data v každé databázi.
- Je snadno zálohovatelná a obnovitelná.
- Je k dispozici i s grafickým nástrojem pro správu.
- Umožňuje rychlou tvorbu a nasazení databázového řešení.
- Podporuje vytváření webů nebo webových aplikací.
- Také umožňuje vytváření menších business aplikací (microsoft.com, 2013).

2.5.2 Microsoft Access

Microsoft Access je aplikace pro správu databází. Je alternativou k ostatním databázovým produktům, jako jsou Microsoft SQL Server, Oracle nebo MySQL. Oproti nim ovšem umožňuje vytvořit vlastní grafické uživatelské prostředí (formou tlačítek a formulářů), což je pro méně zkušené uživatele databází obrovskou výhodou. Koncový uživatel tedy bude odstíněn od klasických databázových tabulek. Budou mu zobrazeny pouze formuláře a tlačítka, pomocí kterých bude přistupovat k potřebným datům v databázi. Právě této funkce bude při realizaci využito.

Visual Basic for Applications

Visual Basic for Application neboli VBA je programovací jazyk vyvinutý společností Microsoft, určený pro produkty Microsoft Ofiice. Vychází z jazyka VB a v této práci je použit pro zajištění funkční logiky aplikace MS Access, konkrétně přechodu mezi jednotlivými formuláři, kontrolu datové kvality apod. (Zbíral, 2011).

2.5.3 Google Draw

Aplikace Google Draw (dostupná na adrese draw.io) je všestranná bezplatná webová aplikace pro tvorbu nejrůznějších diagramů. Podporuje komponenty vývojových a síťových diagramů, myšlenkových map a také UML a BPMN. Mimoto umožňuje vytvářet také vlastní komponenty (seibert-media.net, 2015).

3 Analýza současného stavu

3.1 Současný stav firmy

V posledních několika letech firma zaznamenala nárůst počtu zakázek. Z tohoto důvodu byli najímáni další zaměstnanci a brigádníci na dohodu o provedení práce. Tím vzrostlo množství běžných administrativních činností (jako např. tvorby plánu práce, ale také faktur pro zákazníky) a to se negativně projevilo v čase potřebném pro tvorbu těchto výstupů.

Veškeré výstupy jsou zdlouhavě ručně tvořeny a upravovány přímo majitelkou firmy v softwaru Microsoft Office. Např. faktury jsou vytvářeny ve formátu Microsoft Word a kniha vydaných faktur zase ve formátu Microsoft Excel ve formě samostatných souborů. Některé vybrané výstupy jsou umístěny v příloze: Příloha 1 - Faktura, Příloha 2 - Docházka jednotlivých zaměstnanců, Příloha 3 - Kniha vydaných faktur, Příloha 4 - Plán práce. Tyto soubory nemají žádnou vzájemnou vazbu a proto je jejich tvorba i správa časově náročná a nepřehledná. Tímto je samozřejmě také zapříčiněno zdlouhavé vyhledávání vydaných faktur v případě vzniku problému nebo neuhrazení některé z těchto faktur.

Na základě výše uvedených zjištění a po konzultaci s majitelkou firmy bylo rozhodnuto o vytvoření návrhu řešení zmíněných problémů. Základním požadavkem firmy je tedy sjednotit data, tzn., aby veškeré ukládané informace měly jednotný charakter pro lepší vyhledávání a přehlednost. S tímto také souvisí požadavek na zjednodušení celého procesu ukládání dat a tvorby výstupů. Jelikož již firma vlastní licenci balíku produktů Microsoft Office a i majitelka již má s tímto softwarem určité zkušenosti, bude k tomuto přihlédnuto při návrhu řešení.

Nejprve budou tedy shrnuty požadavky firmy, dále budou krátce představena a analyzována již hotová softwarová řešení dostupná na trhu. Poté bude následovat vyhodnocení a budou provedeny další kroky ke správnému návrhu řešení zadaného problému.

3.2 Shrnutí a analýza požadavků firmy

Nejprve jsou jednotlivé požadavky firmy na softwarovou aplikaci rozděleny podle jejich funkčnosti a priorit (Tabulka 3.1). Dále jsou definovány hranice systému pomocí profilů jednotlivých aktérů (Tabulka 3.2) a je vytvořen diagram případu užití, neboli Use Case diagram, který přehledně zobrazuje funkcionalitu systému. Dále jsou podrobně rozebrány jednotlivé případy užití.

3.2.1 Rozdělení a popis požadavků

ID	Název a popis požadavku	Priorita	Kategorie
R1	Evidence zákazníků	1	Funkční
R2	Evidence zaměstnanců	1	Funkční
R3	Evidence docházky	1	Funkční
R4	Evidence zakázek	1	Funkční
R5	Tvorba a tisk faktur	1	Funkční
R6	Intuitivní ovladatelnost a přehlednost	1	Nefunkční
R7	Vyhledávání údajů v evidencích	1	Funkční
R8	Tiskové sestavy	2	Funkční
R9	Analýza dat	3	Funkční
R10	Ochrana dat	3	Nefunkční
R11	Přijatelné systémové nároky	3	Nefunkční
R12	Rychlá odezva	3	Nefunkční

Tabulka 3.1. Shrnutí požadavků na software (Zdroj: vlastní zpracování).

R1 - Evidence zákazníků - V aplikaci bude možnost evidovat údaje o všech zákaznících, kteří kdy objednávali zakázku. Zejména jejich adresy a kontakty pro možnost budoucího využití.

R2 - Evidence zaměstnanců - V aplikaci bude možnost evidovat údaje o všech zaměstnancích a pracovnících na dohodu o provedení práce, a to nejen současných, ale také těch, se kterými již byl rozvázán pracovní poměr. Zejména jejich adresy a kontakty.

R3 - Evidence docházky - V aplikaci bude možnost evidovat docházku, pro potřeby fakturování zákazníkům, vedení účetnictví a tvorby výstupů pro externí účetní.

R4 - Evidence zakázek - V aplikaci budou evidovány jednotlivé zakázky pro potřeby fakturování zákazníkům a dalších analýz.

R5 - Tvorba a tisk faktur - Aplikace bude podporovat možnost generování a tisku faktur.

R6 - Intuitivní ovladatelnost a přehlednost - Aplikace bude přehledná a intuitivně ovladatelná tak, aby s ní mohl pracovat i méně zkušený uživatel počítače.

R7 - Vyhledávání údajů v evidencích - Aplikace bude umožňovat vyhledávat různé údaje v evidencích a zobrazovat souvislosti k výsledům hledání.

R8 - Tiskové sestavy - V aplikaci bude možnost vytvářet tiskové sestavy pro rychlé zobrazení a tisk různých přehledů.

R9 - Analýza dat - V aplikaci bude možnost provádět jednoduché analýzy.

R10 - Ochrana dat - Bude zajištěna ochrana před neoprávněným přístupem k datům.

R11 - Přijatelné systémové nároky - Aplikace nebude mít přehnané hardwarové a systémové nároky. V kanceláři je umístěn notebook středního výkonu.

R12 - Rychlá odezva - Aplikace nebude zamrzat, produkovat chyby atd.

3.2.2 Definování hranic systému

Aktér	Profil
Majitel	Majitelka firmy, která obsluhuje celou aplikaci a provádí veškeré administrativní činnosti, tvorbu výstupů pro účetní a pro zaměstnance. Je poučena o funkcích softwaru a má k dispozici přístupové údaje.
Zaměstnanec	Zaměstnanec firmy nebo člověk provádějící práci na dohodu o provedení práce. Nemá přístup do aplikace, ale poskytuje vstupní data a využívá jejich výstupů.
Účetní	Externě najatá účetní osoba, která vede účetnictví a kalkuluje mzdy. Nemá přístup do aplikace, ale využívá jejich výstupů a evidenčních dat.
Administrátor	Administrátor průběžně analyzuje chod systému, optimalizuje procesy a přidává nové funkce nebo upravuje ty stávající.

Tabulka 3.2. Profily aktérů (Zdroj: vlastní zpracování).

3.2.3 Návrh případů užití

Nyní, po rozdělení jednotlivých požadavků, je vytvořena tabulka 3.3 s konkrétními případy užití a nakonec samotný Use Case diagram.

ID	Název případu užití
UC1	Přihlášení do systému
UC2	Evidování zákazníků
UC3	Evidování zaměstnanců
UC4	Vytváření faktur
UC5	Tvorba knihy vydaných faktur

UC6	Tisknutí faktur
UC7	Tvorba výstupů pro účetní
UC8	Evidování zakázek
UC9	Evidování docházky
UC10	Vyhledávání údajů v evidencích
UC11	Průběžná analýza požadavků, optimalizace, vylepšování

Tabulka 3.3. Jednotlivé případy užití (Zdroj: vlastní zpracování).

3.2.4 RTM matice

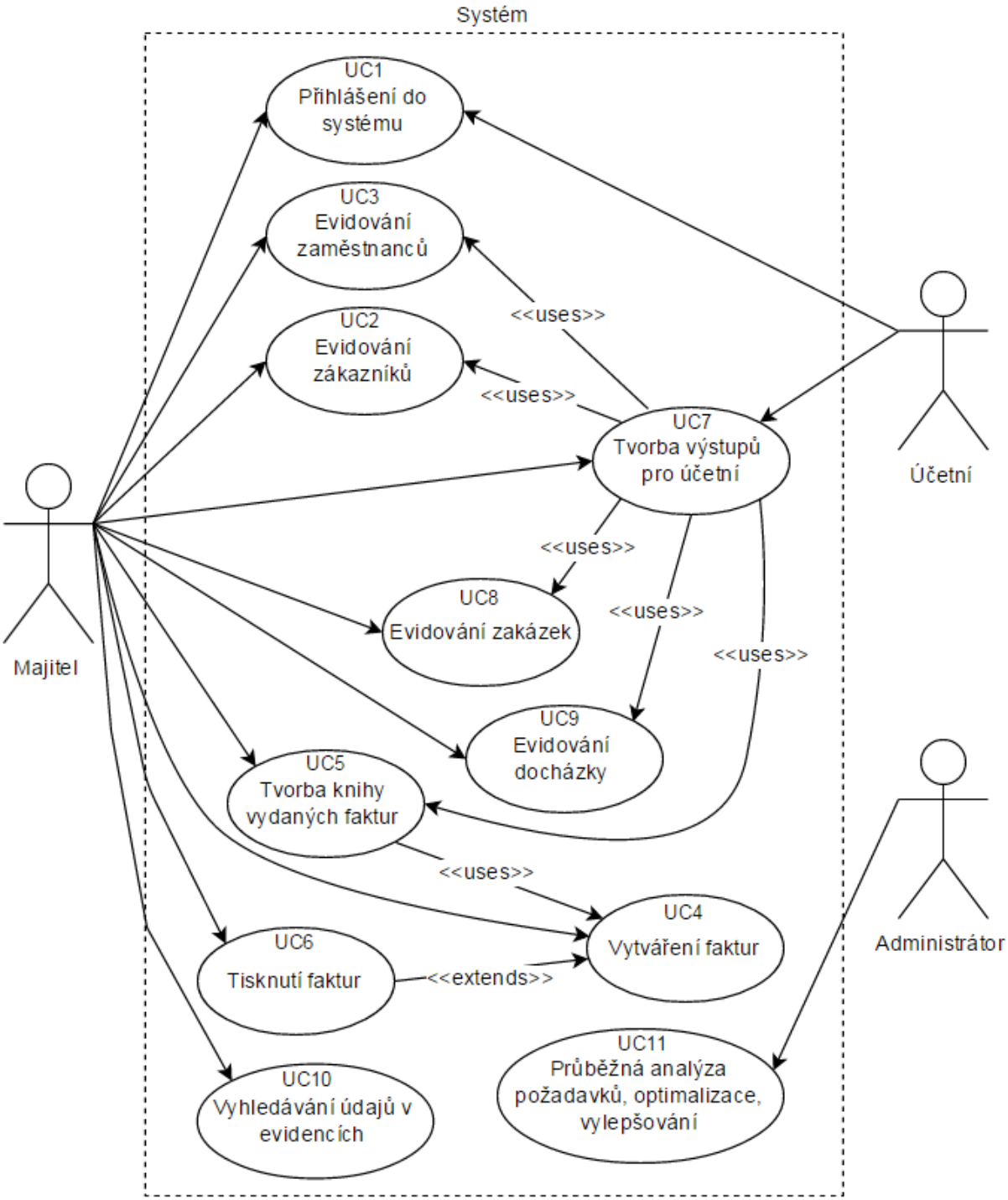
Nyní je nutno udělat poslední krok před vytvořením samotného diagramu a to zaznamenat do RTM matice (Tabulka 3.4) mapování požadavků na jednotlivé případy užití. Pomocí matice sledovanosti požadavků je ověřeno, zda jsou v modelu pokryty všechny funkční uživatelské požadavky.

	Funkcionality											
Uživatelské požadavky		UC1	UC2	UC3	UC4	UC5	UC6	UC7	UC8	UC9	UC10	UC11
	R1		X									
	R2			X								
	R3									X		
	R4								X			
	R5				X		X					
	R6											X
	R7										X	
	R8					X		X				
	R9							X				
	R10	X										
	R11											X
	R12											X

Tabulka 3.4. RTM matice (Zdroj: vlastní zpracování).

3.2.5 Use Case diagram

Nyní již lze podle předešlých analýz a tabulek vytvořit Use Case diagram (Obrázek 3.1).



Obrázek 3.1. Diagram případů užití (Zdroj: vlastní zpracování).

3.3 Analýza existujících softwarových řešení

Na českém trhu je k dispozici velké množství informačních systému pro správu firemních dat a tvorbu výstupů. Lze získat velké a komplexní IS, ale také méně robustní, určené pro menší a střední firmy. Většina těchto řešení je nabízena v několika variantách podle velikosti dané firmy nebo podle funkcionalit. Pokud se ovšem vezmou v úvahu požadavky majitele firmy a výběr se zúží na oblast služeb, konkrétně na software použitelný pro sektor služeb úklidových, je takových k dispozici pouze několik. V této podkapitole jsou tyto SW blíže představeny a nakonec je také zhodnocena vhodnost jejich použití jako řešení daného problému.

Ekonomický systém POHODA

„Ekonomický systém POHODA je jednoduchý, ale přitom nabízí nespočet užitečných funkcí, které doplňují stovky praktických tiskových sestav. Můžete v něm vést účetnictví i daňovou evidenci a vyhoví plátcům i neplátcům DPH. Každá z variant programu POHODA obsahuje takový rozsah a kombinaci funkcí, aby sedla uživateli na míru a nikdo nemusel platit za funkce, které nepoužívá“ (Pohoda.cz, 2015).

Tento systém (viz Příloha 5) je opravdu propracovaný a robustní. Nabízí osm verzí podle obsažených funkcí. Ve verzi POHODA Mini, což je rozsahem nejmenší verze, by mohl být vhodný pro daného podnikatele, a to i když obsahuje některé nadbytečné funkce. Nicméně ani v rozšířenějších verzích, které již obsahují stovky v tomto případě zbytečných funkcí, nenabízí evidenci docházky zaměstnanců (umožňuje samozřejmě výpočet mezd na základě zadaných údajů, ale neeviduje docházku jako takovou).

Informační systém ABRA

„Informační systém ABRA je ideálním řešením pro podporu řízení firmy a evidenci podnikových procesů pokrývajících oblasti obchodu, ekonomických a účetních agend a manažerské řízení a plánování“ (Abra.eu, 2015).

I tento IS (viz Příloha 6) je složen z mnoha modulů a lze tak složit pouze z těch vhodných. Při kompletaci z modulů CRM, Prodej a Docházka, které odpovídají požadavkům dané firmy, obsahuje desítky nadbytečných funkcí.

Helios

Tento software (Příloha 7) je rozdělen na pět produktových řad podle velikosti a oblasti působení firmy, ke kterým lze dokupovat jednotlivé rozšiřující moduly. Pro živnostníky lze vybírat z produktu Helios Red nebo Helios Easy.

„Systém pokrývá tu nejpotřebnější agendu pro podnikatele: kompletní účetnictví, fakturace, banka, evidence majetku, sklady a obchod, sledování a vyhodnocování zakázek, obchodních „procesů“, CRM, mzdy, evidence pošty, kniha jízd“ (Helios.eu, 2015).

Jak lze vyvodit i z tohoto popisu, opět se i v základní verzi jedná o propracovaný a komplexní systém, za který si tato firma také účtuje náležitou částku. Za jednu licenci pro koncového uživatele firma Asseco Solutions, a. s. požaduje přes 33 tisíc korun, což je nejvíce ze všech ostatních zmíněných společností. Také z tohoto důvodu není tento SW vhodným řešením zadaného problému.

Vario Express

Vario Express je software společnosti Altus Vario přizpůsobený živnostníkům a malým firmám. Jeho vzhled vychází z programu Microsoft Office, viz Příloha 8. V této konfiguraci je využitelný spíše jako účetní software, protože obsahuje moduly: Nákup a prodej zboží, Skladová evidence, Prodej služeb a Vedení účetnictví (Vario.cz, 2015).

Jelikož nejsou obsaženy požadované funkce, ani tento software není vhodný.

Informační systém Karat

Tento systém od společnosti KARAT Software, a. s. je opět komplexním řešením, které se skládá z několika funkcionálních modulů. V tomto případě lze také evidovat docházku ve výkazech práce (viz Příloha 9). Z hlediska požadovaných funkcionalit je systém vyhovující. Na druhou stranu kvůli své robustnosti a přílišné složitosti není v tomto případě vhodný.

Informační systém IZIO a **Informační systém QI**

Tyto dva informační systémy (Izio.cz, Master.cz/qi/) nebyly blíže analyzovány z důvodu jejich základu postaveného na cloudovém řešení.

Majitelka firmy si nepřeje ukládat citlivá data mimo lokální počítač a to jednak kvůli možnému zneužití těchto dat, a také kvůli nedostupnosti dostatečně rychlého připojení k internetu na cestách.

3.4 Zhodnocení a návrh řešení

Z dostupných softwarových řešení nelze vybrat takové, které by přesně odpovídalo specifickým požadavkům firmy. V některých základních konfiguracích, které vyhovují z hlediska jednoduchosti, chybí požadované funkcionality a s přechodem na vyšší verzi jsou nabalovány další, v tomto případě nevyužitelné funkce (konektivita na e-shop, vedení celého účetnictví, propojení s datovými schránkami atd.). Jiné softwary jsou opravdu komplexními systémy, které ani nenabízí možnost individualizace pro menšího podnikatele a celkově tak nejsou vhodné. Přehledné zobrazení srovnání nabízí následující tabulka 3.5.

Název softwaru	Nepokrývá
Ekonomický systém POHODA	Evidence docházky (R3)
Informační systém ABRA	Přehlednost (R6)
Helios	Ovladatelnost, přehlednost + cena (R6)
Vario Express	Evidence docházky, přehlednost (R3, R6)
Informační systém Karat	Ovladatelnost a přehlednost (R6)

Tabulka 3.5. Srovnání dostupných SW produktů (Zdroj: vlastní zpracování).

Vzhledem k výše uvedeným zjištěním je nejvhodnějším řešením problémů uvedené firmy vývoj vlastní softwarové aplikace s na míru implementovanými funkcemi.

4 Návrh a implementace modulu softwarové aplikace

V této kapitole již probíhá samotný návrh a implementace softwarové aplikace a to podle požadavků zjištěných v kapitole předcházející. Návrh probíhá podle teoretických východisek uvedených v metodologické části této práce. Nejprve je tedy vytvořen konceptuální návrh se všemi náležitostmi. Dále logický návrh, kde probíhá kontrola normalizace a úprava modelu. Ve fyzickém návrhu probíhá samotná implementace databáze v konkrétním databázovém systému.

4.1 Konceptuální návrh

Nejprve je vytvořen slovník dat (Tabulka 4.1), dále Tabulka 4.2, která zobrazuje omezení multiplicity (kardinalitu vztahů mezi entitami) a poté ER diagram konceptuálního návrhu databáze (Obrázek 4.1), tedy první obecný pohled na zúčastněné entity a vztahy mezi nimi.

4.1.1 Slovník dat

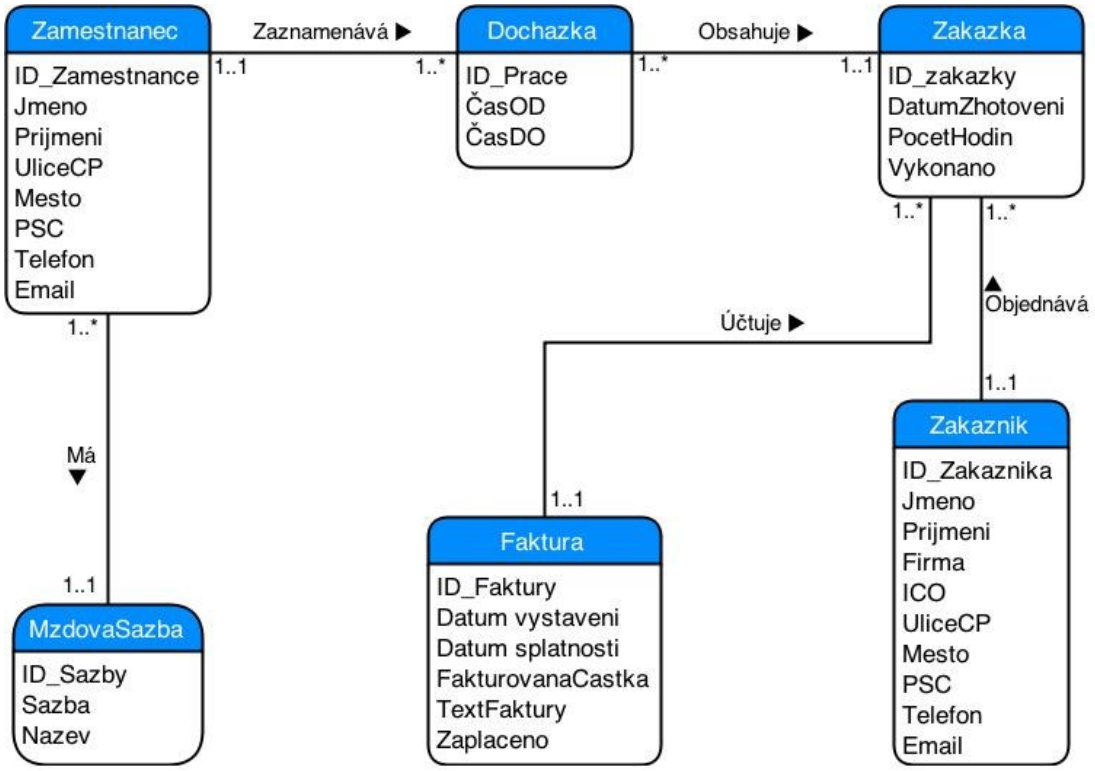
Entita	Jméno atributu	Popis
Zakázka	ID_Zakazky	Jedinečně identifikuje každou zakázku
	DatumZhotovení	Termín zhotovení zakázky
	PocetHodin	Počet odpracovaných hodin na zakázce
	Vykonano	Odlišuje zhotovenou zakázku od plánované
Docházka	ID_Prace	Jedinečně identifikuje každý záznam docházky
	CasOD	Datum a čas začátku práce
	CasDO	Datum a čas konce práce
Zaměstnanec	ID_Zamestnance	Jedinečně identifikuje každého zaměstnance
	Jmeno	Jméno zaměstnance
	Prijmeni	Příjmení zaměstnance
	UliceCP	Ulice a číslo popisné bydliště zaměstnance
	Mesto	Město bydliště zaměstnance
	PSC	PSČ bydliště zaměstnance

	Telefon	Kontaktní telefon na zaměstnance
	Email	Kontaktní email na zaměstnance
Zákazník	ID_Zakaznika	Jednoznačně identifikuje zákazníka
	Jmeno	Jméno zákazníka
	Prijmeni	Příjmení zákazníka
	Firma	Jméno firemního zákazníka
	ICO	Identifikační číslo firemního zákazníka
	UliceCP	Ulice a číslo popisné bydliště zákazníka
	Město	Město bydliště zákazníka
	PSC	PSČ bydliště zákazníka
	Telefon	Kontaktní telefon na zákazníka
	Email	Kontaktní email na zákazníka
Faktura	ID_Faktury	Jednoznačně identifikuje každou fakturu
	DatumVystaveni	Datum vystavení faktury
	DatumSplatnosti	Datum splatnosti faktury
	FakturovanaCastka	Fakturovaná částka za zakázku
	TextFaktury	Text uvedený na faktuře
	Zaplaceno	Odlišuje nezaplacenou fakturu od zaplacené
Mzdová sazba	ID_Sazby	Jedinečně identifikuje mzdovou sazbu
	Sazba	Hodinová mzdová sazba zaměstnance
	Nazev	Název mzdové sazby

Tabulka 4.1. Slovník dat konceptuálního modelu (Zdroj: vlastní zpracování).

V databázové aplikaci je dále třeba vytvořit další integritní omezení. Jedná se například o vstupní masky jednotlivých sloupců (sloupec telefon musí obsahovat pouze číslice, směrovací číslo nesmí mít více než 5 znaků atd.).

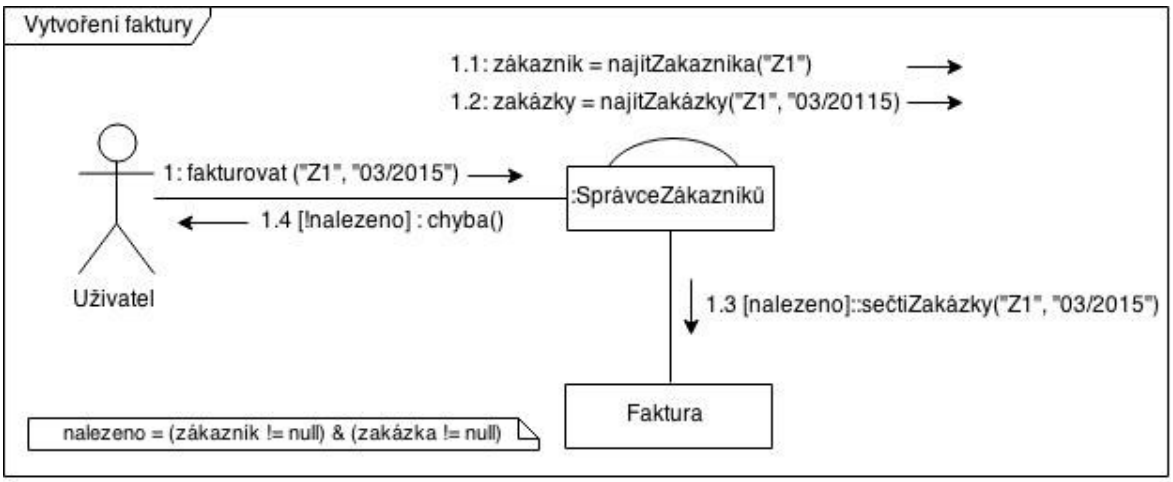
4.1.2 ER Diagram konceptuálního modelu



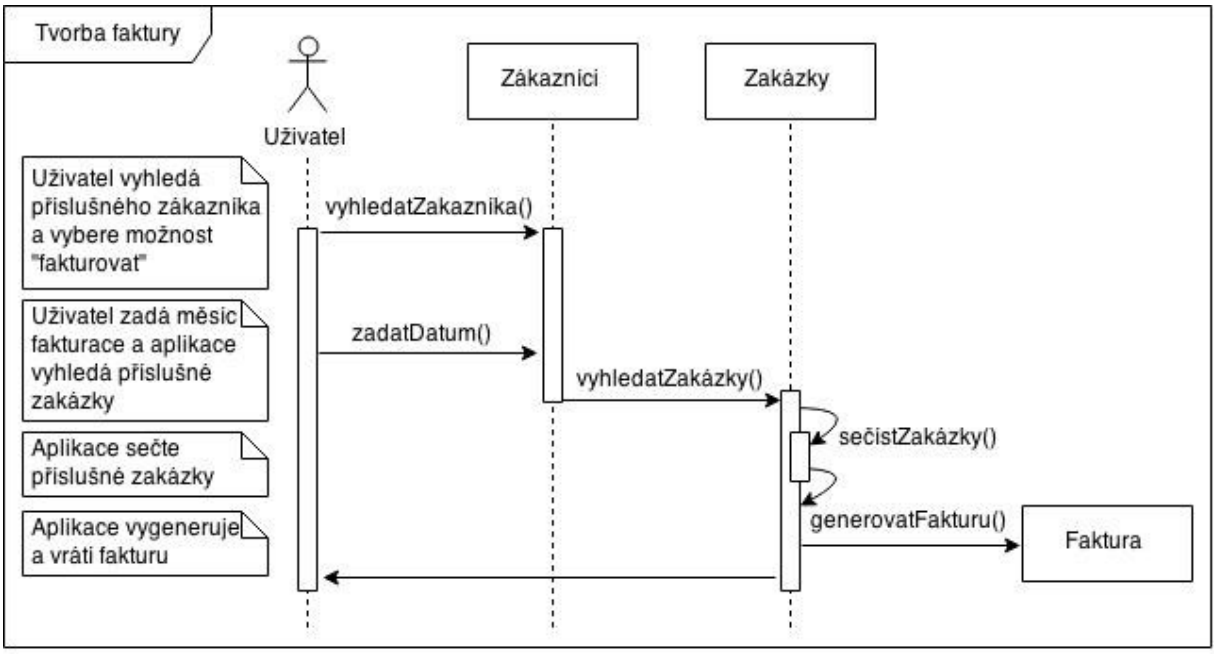
Obrázek 4.1. Konceptuální návrh databáze (Zdroj: vlastní zpracování).

4.1.1 Modelování funkcí aplikace pomocí UML

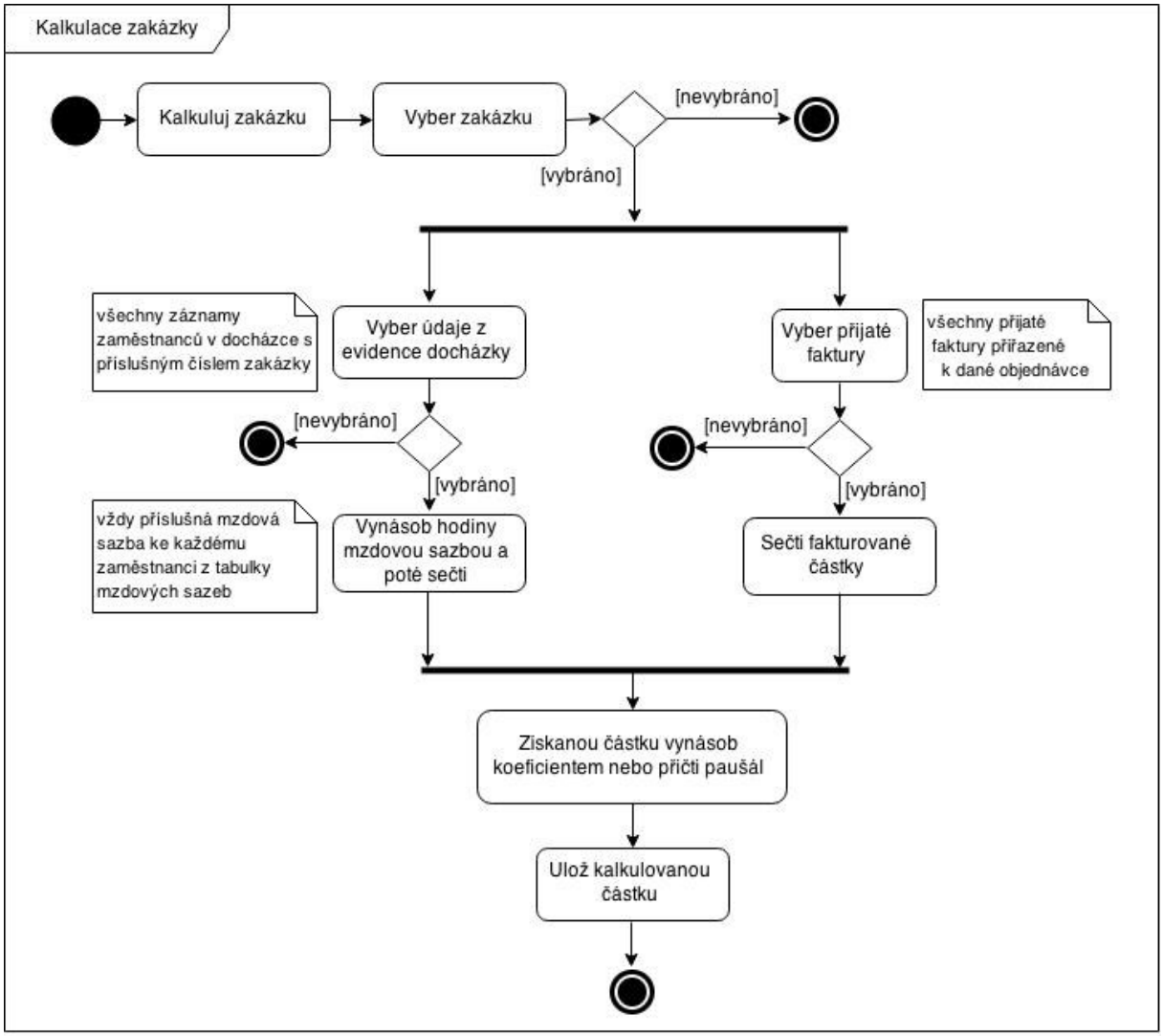
Vzhledem k tomu, že vytváření faktur je složeno z více funkcí, je k tomuto případu užití (UC4) vytvořen komunikační diagram 4.2 a sekvenční diagram 4.3, který je vhodný k popisu tohoto procesu. Jako příklad je uvedeno vytvoření březnové faktury zákazníka se jménem „Z1“. Samotnému vytvoření faktury musí předcházet zaevidování zakázky a získání dat z evidence docházky, jinak nelze fakturu vytvořit. Obrázek 4.4 zobrazuje proces kalkulace ceny zakázky.



Obrázek 4.2. Komunikační diagram UC4 (Zdroj: vlastní zpracování).



Obrázek 4.3. Sekvenční diagram UC4 (Zdroj: vlastní zpracování).



Obrázek 4.4. Diagram aktivit - kalkulace ceny zakázky (Zdroj: vlastní zpracování).

4.2 Logický návrh

Základem tohoto návrhu je ER diagram vytvořený v předchozí podkapitole. Zde je převeden do logického návrhu, který je založen na relačním modelu dat. Výsledkem je logický návrh, který prošel normalizací, má implementovány integritní omezení a je tedy připraven pro fyzickou realizaci.

V tabulce 4.2 jsou definovány a přiřazeny primární a cizí klíče. Dále je specifikováno, jaká akce nastane při aktualizaci nebo smazání záznamu. Některé sloupce v tabulkách jsou přejmenovány, aby byla splněna konvence pojmenování objektů databáze.

tblZamestnanci (ID_Zamestnanec, Jmeno, Prijmeni, UliceCP, Mesto, PSC, Telefon, Email, ID_MzdovaSazba) **Primární klíč** ID_Zamestnanec **Cizí klíč** ID_MzdovaSazba reference tblMzdovaSazba (ID_Sazby) při aktualizaci kaskáda, při vymazání nastavení na výchozí hodnotu
tblMzdovaSazba (ID_Sazby, MzdovaSazba, NazevSazby) **Primární klíč** ID_Sazby
tblDochazka (ID_Prace, ID_Zamestnanec, ID_Zakazka, OdpracovaneHodiny, CasOD, CasDO, Splneno) **Primární klíč** ID_Prace **Cizí klíč** ID_Zamestnanec reference tblZamestnanci (ID_Zamestnanec) při aktualizaci kaskáda, při vymazání kaskáda **Cizí klíč** ID_Zakazka reference tblZakazky (ID_Zakazka) při aktualizaci kaskáda, při vymazání kaskáda
tblZakazky (ID_Zakazka, ID_Zakaznik, Datum, PocetHodin, Vykonano) **Primární klíč** ID_Zakazka **Cizí klíč** ID_Zakaznik reference tblZakaznici (ID_Zakaznik) při aktualizaci kaskáda, při vymazání kaskáda **Cizí klíč** ID_Faktura reference tblFaktura (ID_Faktura) při aktualizaci kaskáda, při vymazání nastavit null
tblFaktura (ID_Faktura, ID_Zakazka, DatumVystaveni, DatumSplatnosti, FakturovanaCastka, TextFaktury, Zaplaceno)

Primární klíč ID_Faktura
tblZakaznici (ID_Zakaznik, Jmeno, Prijmeni, Firma, ICO, UliceCP, Mesto, PSC, Telefon, Email) **Primární klíč** ID_Zakaznik

Tabulka 4.2. Popis referenčních omezení (Zdroj: vlastní zpracování).

4.2.1 Kontrola struktury tabulek pomocí normalizace

Z entitně-relačního diagramu a z tabulky popisu referenčních omezení lze vidět, že návrh splňuje první tři normální formy, které je doporučeno vždy vyžadovat. Není tedy potřeba provádět dekompozici tabulek nebo jiné změny.

Sloupec UliceCP v tabulce Zamestnanec a Zakaznik sice obsahuje dva údaje, ale v tomto případě se nejedná o porušení 1NF. Jak říká Conolly (2009, s. 191) *„V tomto případě jsme se prostě rozhodli údaje o jedné adrese uložit jako jedinou hodnotu. Tato reprezentace adresy neporušuje 1NF.“* Nicméně i přes to se autor práce rozhodl tento jediný vícehodnotový atribut rozdělit do jednotlivých sloupců.

4.2.2 Kontrola podpory uživatelských transakcí

V tomto kroku probíhá kontrola, zda tabulky podporují požadované uživatelské transakce a to pomocí případů užití vytvořených v předcházející kapitole. Zde je navržena Tabulka 4.5, která obsahuje vždy daný případ užití a k němu potřebné tabulky a sloupce.

Název případu užití	Potřebné tabulky	Potřebné sloupce
Přihlášení do systému	Bude implementováno přímo v softwarové aplikaci.	
Evidování zákazníků	tblZakaznici	(ID_Zakaznik, Jmeno, Prijmeni, Firma, ICO, UliceCP, Mesto, PSC, Telefon, Email)
Evidování zaměstnanců	tblZamestnanci	(ID_Zamestnanec, Jmeno, Prijmeni, UliceCP, Mesto, PSC, Telefon, Email, ID_MzdovaSazba)
Vytváření faktur	tblZakazky	(ID_Zakazka, ID_Zakaznik, Datum, PocetHodin, Vykonano)
	tblDochazka	(ID_Zakazka, OdpracovaneHodiny, Splneno)

	tblZakaznici	(ID_Zakaznik, Jmeno, Prijmeni, Firma, ICO, UliceCP, Mesto, PSC, Telefon, Email)
Tvorba knihy vydaných faktur	tblFaktury	ID_Faktura, ID_Zakazka, DatumVystaveni, DatumSplatnosti, FakturovanaCastka
Tisknutí faktur	tblFaktury	ID_Faktura, ID_Zakazka, DatumVystaveni, DatumSplatnosti, FakturovanaCastka
Tvorba výstupů pro účetní	tblZakaznici	Všechny dostupné
	tblZamestnanci	Všechny dostupné
	tblZakazky	Všechny dostupné
	tblDochazka	Všechny dostupné
	tblMzdovaSazba	Všechny dostupné
Evidování zakázek	tblZakazky	(ID_Zakazka, ID_Zakaznik, Datum, PocetHodin, Vykonano)
Evidování docházky	tblDochazka	(ID_Prace, ID_Zamestnanec, ID_Zakazka, OdpracovaneHodiny, CasOD, CasDO, Splneno)
Vyhledávání údajů v evidencích	tblZakaznici	Všechny dostupné
	tblZamestnanci	Všechny dostupné
	tblZakazky	Všechny dostupné
	tblDochazka	Všechny dostupné

Tabulka 4.3. Kontrola podpory uživatelských transakcí (Zdroj: vlastní zpracování).

Kontrolou v Tabulce 4.3 bylo zjištěno, že všechny uživatelské požadavky a transakce jsou databázovými tabulkami podporovány a lze tedy přistoupit k dalšímu kroku.

4.2.3 Kontrola integritních omezení

Reprezentace integritních omezení je kontrolována v Tabulce 4.2. Je potřeba ověřit, zda jsou navržena všechna potřebná omezení a také zda jsou správně vytvořena. To se týká jak jednotlivých primárních a cizích klíčů, tak také kardinality relací z Obrázku 4.5 a omezení „NULL“ hodnot ve slovníku dat v Tabulce 4.4.

Tabulka	Jméno atributu	Datový typ	Null hodnoty	Klíč
tblZakazka	ID_Zakazky	int	Ne	PK
	ID_Zakaznik	int	Ne	FK
	ID_Faktura	int	Ano	FK

	Datum	datetime	Ano	
	PocetHodin	float	Ano	
	Cena	money	Ano	
tblDochazka	ID_Prace	int	Ne	PK
	ID_Zamestnanec	int	Ne	FK
	ID_Zakazka	int	Ne	FK
	OdpracovaneHodiny	float	Ano	
	CasOD	Datum a čas	Ano	
	CasDO	Datum a čas	Ano	
tblZamestnanec	ID_Zamestnance	int	Ne	PK
	Jmeno	nvarchar(30)	Ne	
	Prijmeni	nvarchar(30)	Ne	
	Ulice	nvarchar(30)	Ano	
	CP	varchar(10)	Ano	
	Mesto	nvarchar(30)	Ano	
	PSC	nchar(6)	Ano	
	Telefon	char(16)	Ano	
	Email	nchar(30)	Ano	
	ID_MzdovaSazba	int	Ne	FK
tblZakaznik	ID_Zakaznika	int	Ne	PK
	Jmeno	nvarchar(30)	Ano	
	Prijmeni	nvarchar(30)	Ano	
	Firma	nvarchar(30)	Ano	
	ICO	char(8)	Ano	
	Ulice	nvarchar(30)	Ne	
	CP	varchar(10)	Ne	
	Mesto	nvarchar(30)	Ne	
	PSC	char(6)	Ne	
	Telefon	char(16)	Ano	
	Email	nchar(30)	Ano	
tblFaktura	ID_Faktury	int	Ne	PK
	DatumVystaveni	date	Ne	
	DatumSplatnosti	date	Ne	
	FakturovanaCastka	money	Ne	
	TextFaktury	nvarchar(MAX)	Ano	
	Zaplaceno	bit	Ne	

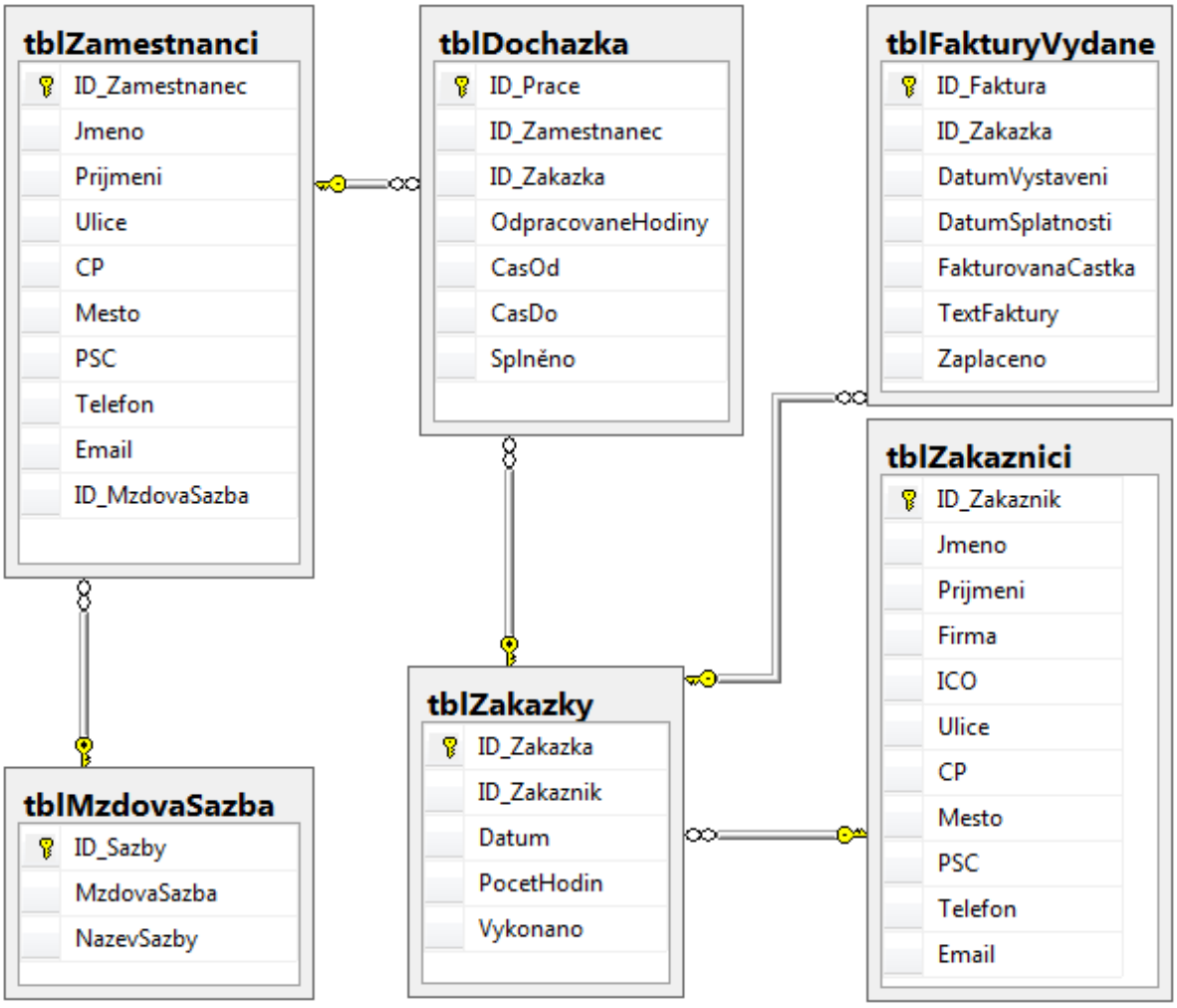
tblMzdovaSazba	ID_Sazby	int	Ne	PK
	Sazba	money	Ne	
	Nazev	nvarchar(20)	Ne	

Tabulka 4.4. Slovník dat logického modelu (Zdroj: vlastní zpracování)

4.3 Fyzický návrh

V této podkapitole probíhá samotná implementace předcházejícího návrhu v konkrétním databázovém systému. Nejprve je vytvořen ER diagram fyzického modelu 4.5 a jsou realizovány jednotlivé relační tabulky dle slovníku dat 4.4. Poté je pomocí cizích klíčů zajištěna návaznost tabulek a integrita dat.

4.3.1 ER diagram fyzického modelu



Obrázek 4.5. ER diagram fyzického modelu (Zdroj: vlastní zpracování).

4.3.2 Návrh tabulek

Následuje fyzický návrh jednotlivých tabulek v Microsoft SQL Express Server 2012.

tblDochazka

Column Name	Data Type	Allow Nulls
ID_Prace	int	☐
ID_Zamestnanec	int	☐
ID_Zakazka	int	☐
OdpracovaneHodiny	float	☑
CasOd	datetime	☑
CasDo	datetime	☑

Obrázek 4.6. Tabulka docházky

(Zdroj: vlastní zpracování).

tblFaktury

Column Name	Data Type	Allow Nulls
ID_Faktura	int	☐
DatumVystaveni	date	☐
DatumSplatnosti	date	☐
FakturovanaCastka	money	☐
TextFaktury	nvarchar(MAX)	☑
Zaplaceno	bit	☐

Obrázek 4.7. Tabulka faktur

(Zdroj: vlastní zpracování).

tblMzdovaSazba

Column Name	Data Type	Allow Nulls
ID_Sazby	int	☐
MzdovaSazba	money	☐
NazevSazby	nvarchar(20)	☐

Obrázek 4.8. Tabulka mzdových sazeb

(Zdroj: vlastní zpracování).

tblZakazky

Column Name	Data Type	Allow Nulls
ID_Zakazka	int	☐
ID_Zakaznik	int	☐
ID_Faktura	int	☑
Datum	datetime	☐
PocetHodin	float	☑
Cena	money	☑

Obrázek 4.9. Tabulka zakázek

(Zdroj: vlastní zpracování).

tblZamestnanci

Column Name	Data Type	Allow Nulls
ID_Zamestnanec	int	☐
Jmeno	nvarchar(30)	☐
Prijmeni	nvarchar(30)	☐
Ulice	nvarchar(30)	☑
CP	varchar(10)	☑
Mesto	nvarchar(30)	☑
PSC	nchar(6)	☑
Telefon	char(16)	☑
Email	nchar(30)	☑
ID_MzdovaSazba	int	☐

Obrázek 4.10. Tabulka zaměstnanců

(Zdroj: vlastní zpracování).

tblZakaznici

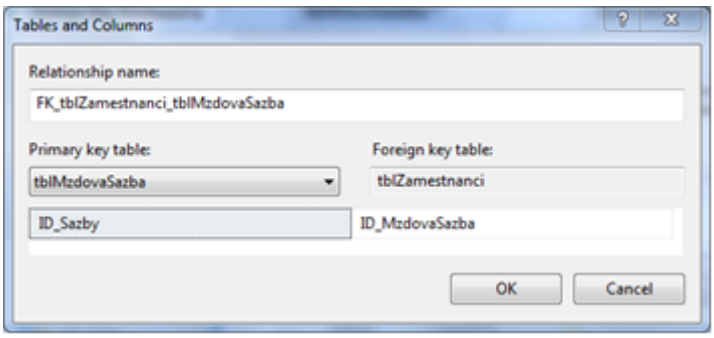
Column Name	Data Type	Allow Nulls
ID_Zakaznik	int	☐
Jmeno	nvarchar(30)	☑
Prijmeni	nvarchar(30)	☑
Firma	nvarchar(30)	☑
ICO	char(8)	☑
Ulice	nvarchar(30)	☐
CP	varchar(10)	☐
Mesto	nvarchar(30)	☐
PSC	char(6)	☐
Telefon	char(16)	☑
Email	nchar(30)	☑

Obrázek 4.11. Tabulka zákazníků

(Zdroj: vlastní zpracování).

4.3.3 Vytvoření integritních omezení a vazeb mezi tabulkami

Vazby mezi tabulkami jsou vytvářeny pomocí cizích klíčů, které reprezentují primární klíč z návazné tabulky. Těmito vazbami je také zajištěna integrita dat při změně nebo vymazání záznamů. Jak lze vidět na obrázku 4.12, při vytváření cizího klíče je nutno zadat název, který by měl plnit konvenci pojmenování (naming convention). FK - jako cizí klíč (foreign key) a názvy obou zúčastněných tabulek. Dále je potřeba vybrat tabulku s primárním klíčem, tabulku s cizím klíčem a také příslušné klíčové sloupce.



Obrázek 4.12. Vytvoření vazby mezi tabulkami (Zdroj: vlastní zpracování).

V dalším kroku je zajištěna referenční integrita. Na následujícím obrázku 4.13 je nastaveno v řádku „INSERT And UPDATE Specification“ co se stane při aktualizaci nebo smazání některého ze záznamu v tabulce. V tomto případě je při změně záznamu aktualizován také návazný záznam v tabulce Zamestnanci a při smazání je nastavena výchozí hodnota.

(General)	
Check Existing Data On Creation Or I	Yes
Tables And Columns Specification	
Database Designer	
Enforce For Replication	Yes
Enforce Foreign Key Constraint	Yes
INSERT And UPDATE Specification	
Delete Rule	Set Default
Update Rule	Cascade
Identity	
(Name)	FK_tblZamestnanci_tblMzdovaSazba
Description	

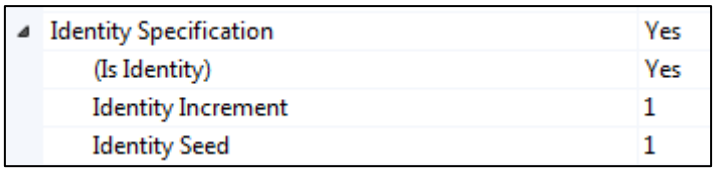
Obrázek 4.13. Zajištění ref. integrity dat - tblZamestnanci (Zdroj: vlastní zpracování).

Dále je popsána tvorba ostatních integritních omezení a pravidel u jednotlivých tabulek. U některých je uveden také obrázek jejich realizace.

tblMzdovaSazba

Sloupec ID_Sazby má nastaveno *„identity specification“* na hodnotu *„yes“* s parametry 1, 1. To znamená, že přidávané záznamy jsou automaticky číslovány od jedničky s jednotkovým přírůstkem (Obrázek 4.13).

Zajištění integrity tabulky s tblZamestnanci je zachyceno výše na obrázku 4.12.



Obrázek 4.13. Identity specifikace (Zdroj: vlastní zpracování).

tblZakaznici

Sloupec ID_Zakaznik má nastaveno *„identity specification“* na hodnotu *„yes“* s parametry 1, 1.

tblDochazka

Sloupec ID_Prace má nastaveno *„identity specification“* na hodnotu *„yes“* s parametry 1, 1.

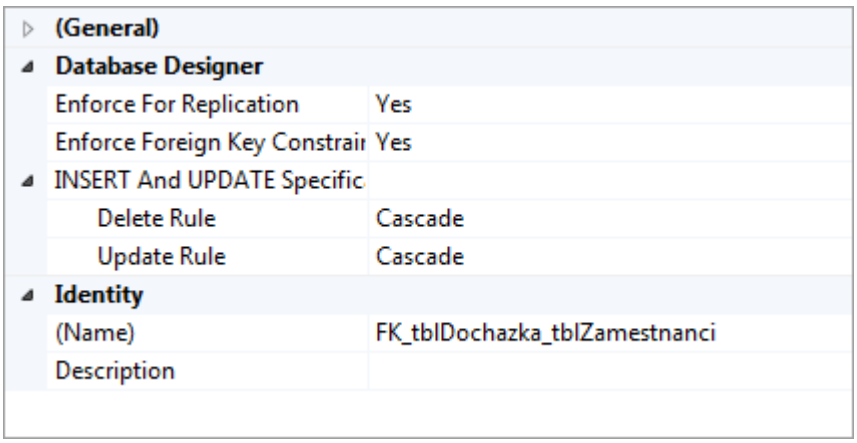
Sloupec OdpracovanéHodiny je počítán automaticky podle vzorce ([tblDochazka.CasDo]-[tblDochazka.CasOd]).

tblZamestnanci

Sloupec ID_Zamestnanec má nastaveno *„identity specification“* na hodnotu *„yes“* s parametry 1, 1.

Implementace vazby na tabulku tblMzdovaSazba je na obrázku 4.12.

Ukázku implementace vazby a zajištění integrity s tabulkou tblDochazka zachycuje Obrázek 4.14.

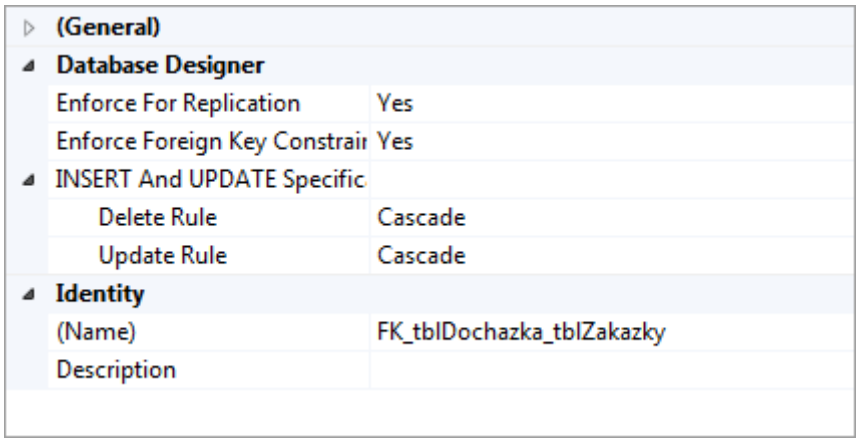


Obrázek 4.14. Zajištění integrity tblDochazka a tblZamestnanci (Zdroj: vlastní zpracování).

tblZakazky

Sloupec ID_Zakazka má nastaveno *„identity specification“* na hodnotu *„yes“* s parametry 1, 1.

Vlastnosti vazby tblDochazka na tblZakazky jsou zobrazeny na obrázku 4.15. Lze vidět, že v tomto případě při aktualizace i smazání záznamů probíhá kaskádou.



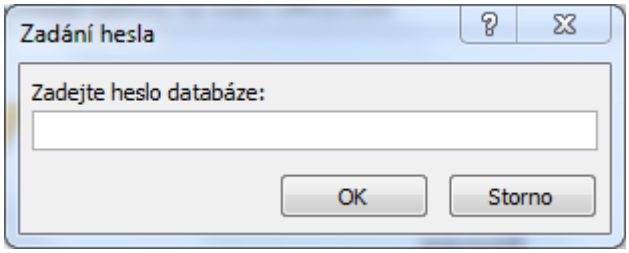
Obrázek 4.15. Zajištění integrity tblDochazka a tblZakazky(Zdroj: vlastní zpracování).

Po všech předchozích krocích je nyní databáze fyzicky implementována se všemi integritními pravidly a je připravena k využívání. Následně může docházet k vylepšování a změnám podle aktuálních požadavků zákazníka.

4.3.4 Návrh rozhraní aplikace

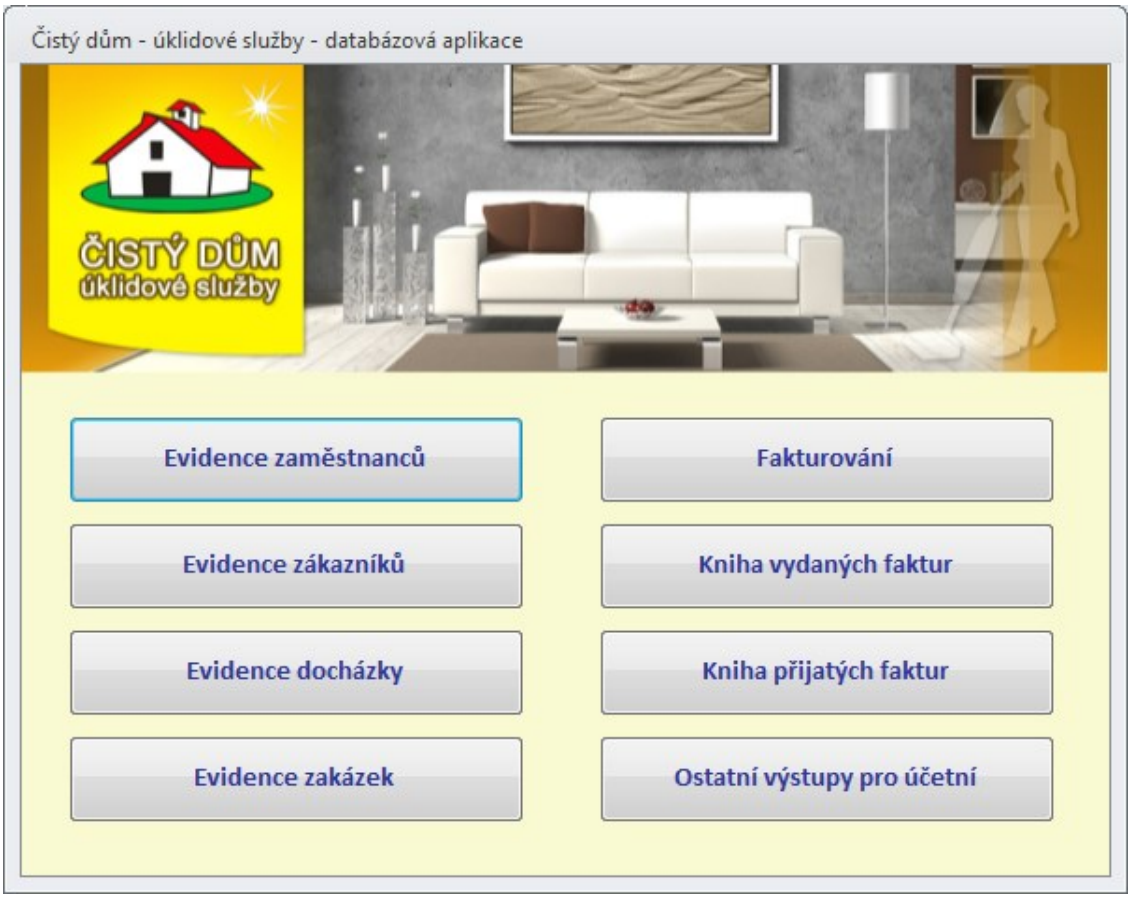
V této podkapitole je proveden návrh rozhraní databázové aplikace v programu Microsoft Access. Je zde vždy uveden popis určité části aplikace spolu s uvedením případů užití, které jsou v dané částí rozhraní realizovány.

Při spuštění aplikace musí uživatel zadat správné heslo (viz obrázek 4.16). Tím je realizován případ užití **UC1 - Přihlášení do systému.**



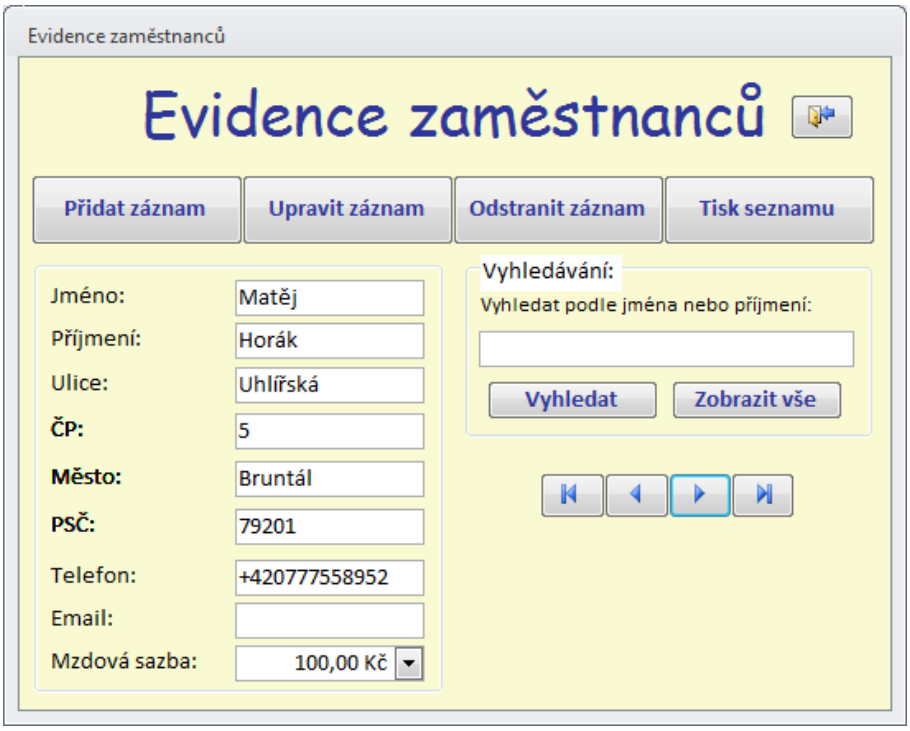
Obrázek 4.16. Přihlášení do systému (Zdroj: vlastní zpracování).

Po zadání správného hesla se uživateli zobrazí hlavní formulářové okno (viz Obrázek 4.17). Zde jsou tlačítka pro otevření všech evidencí, knihy vydaných faktur, jejich tisku atd. Zkrátka je rozcestníkem pro všechny funkce aplikace.



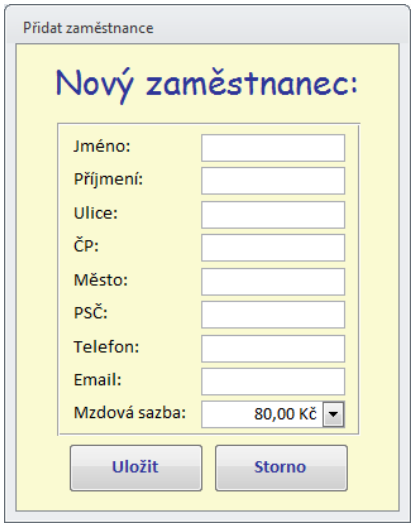
Obrázek 4.17. Hlavní okno aplikace (Zdroj: vlastní zpracování).

Klikem na tlačítko *Evidence zaměstnanců* je otevřen nový formulář (Obrázek 4.18), čímž je realizován případ užití **UC2 - Evidování zaměstnanců**.

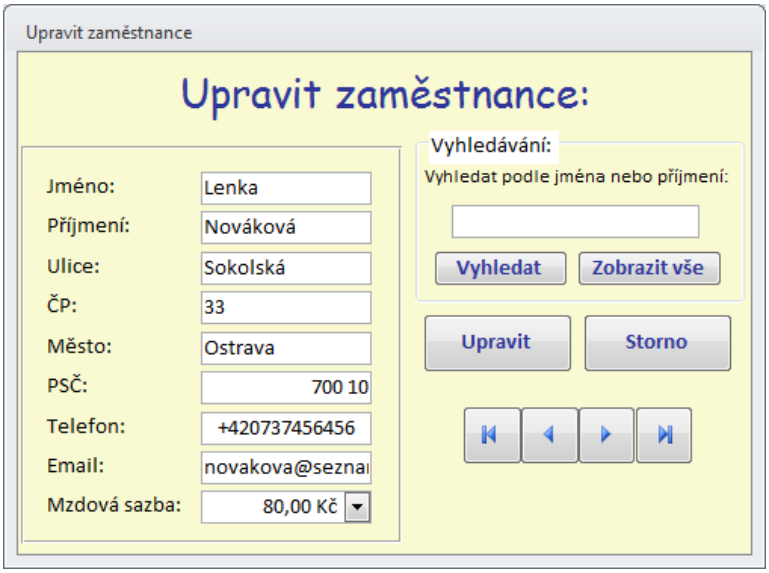


Obrázek 4.18. Evidence zaměstnanců (Zdroj: vlastní zpracování).

V tomto okně, lze přecházet mezi jednotlivými záznamy tlačítky s šipkami nebo lze využít vyhledávání. Tím je naplněn případ užití **UC10 - vyhledávání údajů v evidencích**. Dále je zde možnost přidávat další záznamy. Klikem na příslušné tlačítko je otevřen další formulář, pro zadání údajů (Obrázek 4.19). Klikem na tlačítko *Upravit záznam*, je otevřen podobný formulář, ale s předvyplněnými údaji pro úpravu dat a s vyhledáváním (Obrázek 4.20).



Obrázek 4.19. Přidání nového zaměstnance (Zdroj: vlastní)



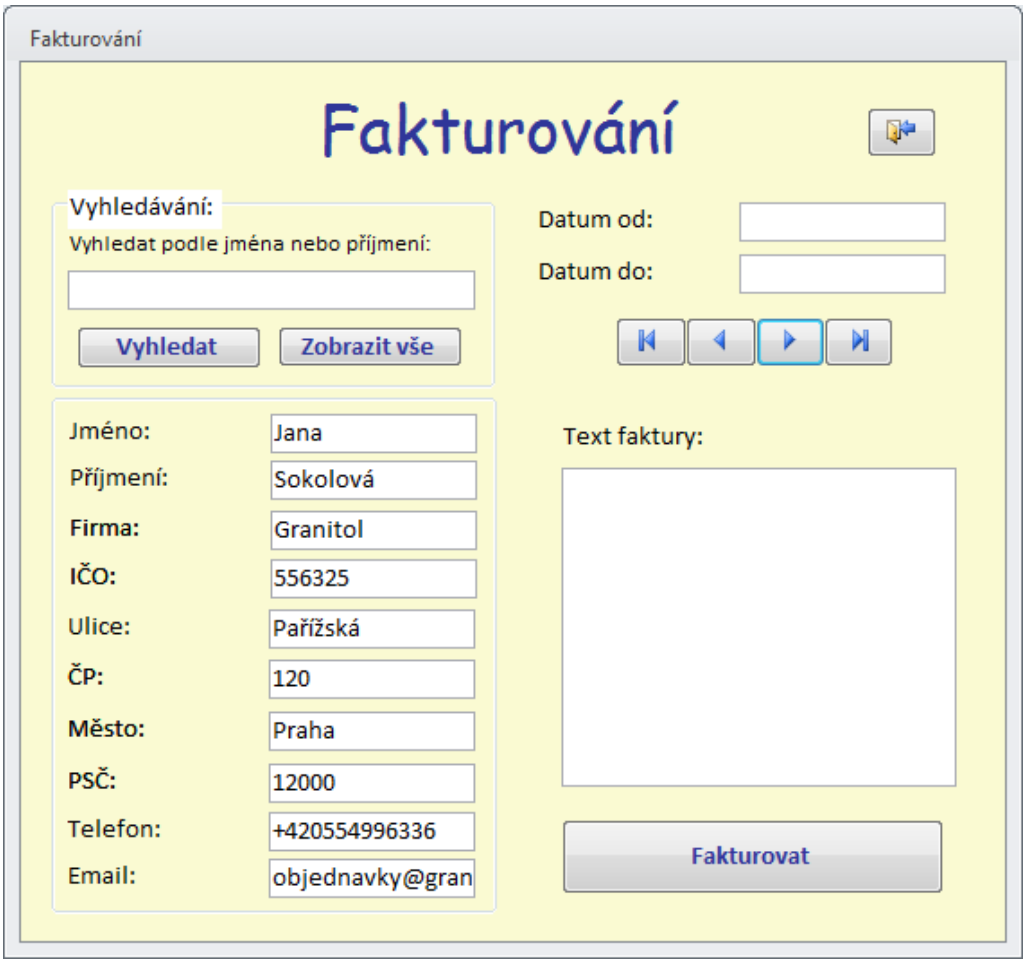
Obrázek 4.20. Úprava zaměstnanců (Zdroj: vlastní zpracování).

Další funkcí je odstranění příslušného záznamu. Klikem na tlačítko *Odstranit záznam* je zobrazeno výstražné okno a po potvrzení je záznam smazán z databáze (Obrázek 4.21). Klikem na tlačítko *Tisk seznamu*, je zobrazena tisková sestava se všemi zaměstnanci s možností tisku.

Velmi podobný průběh mají funkce Evidence zákazníků, Evidence docházky, evidence zakázek a knihy vydaných a přijatých faktur. Tím jsou realizovány případy užití **UC2 - Evidování zákazníků, UC8 - Evidování zakázek, UC9 - Evidování docházky, UC5 - Tvorba knihy vydaných faktur, UC6 - Tisknutí faktur.**

Pod tlačítkem Ostatní výstupy pro účetní jsou ukryty různé sestavy a uživatelské pohledy, které externí účetní využívá pro potřeby vedení účetnictví. Tím je splněn případ užití **UC7 - Tvorba výstupů pro účetní**.

Klikem na tlačítko *Fakturování*, v hlavním okně aplikace, je zobrazen formulář pro generování faktur zákazníkům (Obrázek 4.21). Uživatel vyhledá zákazníka, zadá dva parametry, což je rozmezí dat, mezi kterými se mají vyhledat všechny příslušné zakázky a po kliku na tlačítko *Fakturovat* je vygenerována sestava s fakturou. Tímto je realizován případ užití **UC4 - Vytváření faktur**.



Obrázek 4.21. Vytváření faktur (Zdroj: vlastní zpracování).

5 Závěr

Cílem této diplomové práce bylo vyřešit problém správy dat a tvorby výstupů z podnikání v menší soukromé firmě. Po analýze uživatelských požadavků a průzkumu na trhu dostupných informačních systémů a aplikací bylo rozhodnuto, že nejlepší variantou řešení tohoto problému je vývoj vlastní softwarové aplikace s na míru implementovanými funkcemi.

Aby bylo možno dosáhnout tohoto cíle, nejprve musely být v teoreticko-metodické části popsány jednotlivé metody tvorby softwaru, možnosti využití unifikovaného modelovacího jazyka a objasněna teorie databází a používaných vývojových nástrojů.

Uživatelské požadavky byly získány metodou polostrukturovaného rozhovoru s majitelkou firmy. Na základě jejich analýzy byly dále vytvořeny diagramy případů užití, aktivit, komunikační a sekvenční diagramy nutné pro korektní průběh celého procesu vývoje softwarové aplikace. Proběhla také kontrola tabulek pomocí normalizace, podpory uživatelských transakcí a nakonec kontrola integritních omezení pro ověření správné funkce celé databáze. Na závěr byl vytvořen také finální ER diagram a slovník dat, který sloužil jako základ pro fyzickou implementaci tohoto návrhu v Microsoft SQL Serveru a Microsoft Accessu.

Výsledným produktem této práce je databáze a na ni návazná intuitivně ovládaná aplikace zjednodušující správu dat, kterou může využívat i méně zkušený uživatel počítače. Aplikace slouží nejen pro ukládání firemních dat, ale také pro automatickou tvorbu výstupů jako jsou faktury nebo souhrny zakázek.

Základem aplikace byla zvolena serverová databáze z důvodu možného budoucího rozšíření o webovou aplikaci, přes kterou budou moci zaměstnanci nebo účetní zadávat data z jakéhokoliv počítače připojeného k internetu.

Nyní je tato aplikace nasazena v pilotním provozu na firemním počítači, kde probíhá testování a hodnocení spokojenosti. Postupně bude aplikace optimalizována a vylepšována novými funkcemi podle aktuálních požadavků majitelky. Návrh i implementace modulu této aplikace byla úspěšná, tudíž lze cíle práce považovat za splněné.

Seznam použité literatury

Odborná literatura

ARLOW, Jim a Illa NEUSTADT. UML2 a unifikovaný proces vývoje aplikací. Brno: Computer Press, 2007. 568 s. ISBN 978-80-251-1503-9.

BOEHM, Barry W. Software engineering: Barry W. Boehm's lifetime contributions to software. Wiley-IEEE Computer Society Press. 2007. 832 s. ISBN: 978-0-470-14873-0.

CONOLLY, T., C. BEGG a R. HOLOWCZAK. Mistrovství databáze: profesionální průvodce tvorbou efektivních databází. Brno: Computer Press, 2009. 584 s. ISBN 978-80-251-2328-7.

KADLEC, Václav. Agilní programování: Metodiky efektivního vývoje softwaru. Brno: Computer Press, 2004. ISBN 80-251-0342-0.

KISZKA, Bogdan. 1001 tipů a triků pro jazyk Java. Brno: Computer press, 2009. ISBN 978-80-251-2467-3.

KRUCZEK, Aleš. Microsoft Access 2010: podrobná uživatelská příručka. Brno: Computer Press, 2010. ISBN 978-80-251-3289-0.

LARSON, Brian. Delivering Business Intelligence with Microsoft SQL Server 2012. 3rd ed. New York: McGraw-Hill, 2012, 656 p. ISBN 978-0-07-175938-0.

OPPEL, Andy. SQL bez předchozích znalostí. Brno: Computer press, 2008. ISBN 978-80-251-1707-1.

PÍSEK, Slavoj. *Access 2010: podrobný průvodce.* Praha: Grada, 2011. ISBN 978-80-247-3653-2.

Elektronické dokumenty a ostatní

Draw.io: The Leading Diagramming Solution since 2005. Seibert-media.net [online]. 2015 [cit. 2015-04-21]. Dostupné z: https://info.seibert-media.net/display/landingpage/draw.io+for+Confluence+and+JIRA+-+Test%2C+Consultation%2C+Licensing%2C+Training%2C+Adaptation

Edice Microsoft SQL Server Express 2012. Microsoft.com [online]. 2013 [cit. 2013-05-02]. Dostupné z: http://microsoft.com/sqlserver/cs/cz/editions/express.aspx

HELIOS Easy. Helios.eu [online]. 2015 [cit. 2015-04-07]. Dostupné z: http://www.helios.eu/produkty/helios-orange/helios-easy-space/helios-easy/

Informační systém IZIO. Izio.cz [online]. 2015 [cit. 2015-04-07]. Dostupné z: https://www.izio.cz/

Informační systém QI. Master.cz [online]. 2015 [cit. 2015-04-07]. Dostupné z: www.master.cz/qi/

JANKŮ, Gábina. Kadlec, Václav: Agilní programování. [online]. 2015 [cit. 2015-04-06]. Dostupné z: http://precteno.com/kadlec-vaclav-agilni-programovani-1472

Komplexní informační systém ABRA G3. Abra.cz [online]. 2015 [cit. 2015-04-10]. Dostupné z: http://www.abra.eu/informacni-systemy/informacni-system-abra-g3

MYSLIVEČEK, Ondřej. Agilní vývoj v praxi. Systemonline.cz [online]. 2011 [cit. 2015-04-06]. Dostupné z: http://systemonline.cz/sprava-it/agilni-vyvoj-v-praxi.htm

RUP – Rational Unified Process. Testovanisoftwaru.cz [online]. 2011 [cit. 2015-04-07]. Dostupné z: http://testovanisoftwaru.cz/manualni-testovani/modely-zivotniho-cyklu-softwaru/rup/

RUP Summary. Projects.staffs.ac.uk [online]. 2008 [cit. 2015-04-07]. Dostupné z: http://projects.staffs.ac.uk/suniwe/project/rup.html

Řešení pro firmy poskytující služby. Karatsoftware.cz [online]. 2015 [cit. 2015-04-10]. Dostupné z: http://www.karatsoftware.cz/oborova-reseni/sluzby/

Spirálový model. Testovanisoftwaru.cz [online]. 2011 [cit. 2015-04-07]. Dostupné z: http://testovanisoftwaru.cz/manualni-testovani/modely-zivotniho-cyklu-softwaru/spiralovy-model/

Vario Express. Vario.cz [online]. 2014 [cit. 2015-04-10]. Dostupné z: http://www.vario.cz/vario-express/

Vodopádový model. Testovanisoftwaru.cz [online]. 2011 [cit. 2015-04-07]. Dostupné z: http://testovanisoftwaru.cz/manualni-testovani/modely-zivotniho-cyklu-softwaru/vodopadovy-model/

Vyberte si účetní program POHODA. Pohoda.cz [online]. 2015 [cit. 2015-04-10]. Dostupné z: http://www.pohoda.cz/vyberte-si-variantu/

ZBÍRAL, David. Visual Basic for Applications, VBScript. [online]. 2011 [cit. 2015-04-21]. Dostupné z: http://www.david-zbiral.cz/vb.htm

ZEPHOTO. Budoucnost bude agilní. I pro projektové manažery. Zework.wordpress.com [online]. 2014 [cit. 2015-04-07]. Dostupné z: https://zework.wordpress.com/2014/06/20/budoucnost-bude-agilni-i-pro-projektove-manazery

Seznam zkratek

1NF - První normální forma

2NF - Druhá normální forma

3NF - Třetí normální forma

A pod. - A podobně

A.s. - Akciová společnost

Atd. - A tak dále

BCNF - Boyce-Coddova normální forma

BPMN - Notace určená k popisu podnikových procesů

CASE - Počítačem podpořené softwarové inženýrství

CRM - Řízení vztahů se zákazníky

DA - Správce dat

DBA - Správce databáze

DBMS - Systém řízení báze dat

DPH - Daň z přidané hodnoty

ER diagram - Entitně-relační diagram

FK - Cizí klíč

IČO - Identifikační číslo

IS - Informační systém

MS - Microsoft

Např. - Například

ODBC - Otevřené databázové propojení

OODBMS - Objektově orientované systémy řízení databáze

ORDBMS - Objektově relační systémy řízení databáze

PC - Počítač

RDBMS - Relační systém řízení databáze

Ref. - Referenční

RTM - Matice sledovatelnosti požadavků

RUP - Metodika vývoje softwaru Rational Unified Process

SQL - Strukturovaný dotazovací jazyk

SW - Software

Tj. - To je

Tzn. - To znamená

Tzv. - Takzvaně, takzvaný

UC - Diagram případu užití

UML - Unifikovaný modelovací jazyk

UP - Metodika Unified Process

VB - Visual Basic

VBA - Visual Basic for Application

Seznam obrázků

Seznam tabulek

Prohlášení o využití výsledků diplomové práce

Prohlašuji, že

- Jsem byl seznámen s tím, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 Sb. – autorský zákon, zejména § 35 – užití díla v rámci občanských a náboženských obřadů, v rámci školních představení a užití díla školního a § 60 – školní dílo;
- beru na vědomí, že Vysoká škola báňská – Technická univerzita Ostrava (dále jen VŠB-TUO) má právo nevýdělečně, ke své vnitřní potřebě, diplomovou práci užít (§ 35 odst. 3);
- souhlasím s tím, že diplomová práce bude v elektronické podobě archivována v Ústřední knihovně VŠB-TUO a jeden výtisk bude uložen u vedoucího diplomové práce. Souhlasím s tím, že bibliografické údaje o diplomové práci budou zveřejněny v informačním systému VŠB-TUO;
- bylo sjednáno, že s VŠB-TUO, v případě zájmu z její strany, uzavřu licenční smlouvu s oprávněním užít dílo v rozsahu § 12 odst. 4 autorského zákona;
- bylo sjednáno, že užít své dílo, diplomovou práci, nebo poskytnout licenci k jejímu využití mohu jen se souhlasem VŠB-TUO, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly VŠB-TUO na vytvoření díla vynaloženy (až do jejich skutečné výše).

V Ostravě dne 25.04.2015

.....

Bc. Václav Koběrský