Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačního inženýrství

Bakalářská práce

Návrh modelu uživatelských požadavků na podnikovou webovou aplikaci

Miroslav Juráš

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Miroslav Juráš

Systémové inženýrství a informatika
Informatika

Název práce

Návrh modelu uživatelských požadavků na podnikovou webovou aplikaci

Název anglicky

Design of User Requirements' Model for Enterprise Web Application

---

Cíle práce

Cílem práce je analýza a vytvoření modelu uživatelských požadavků dle metodiky Unified Process pro podnikovou webovou aplikaci vykazování hodin, která bude sloužit vykazování hodin zaměstnanců do jednotlivých projektů.

Metodika

1. Na základě studia odborných zdrojů popište formou literární rešerše model uživatelských požadavků dle metodiky Unified Process

2. Proveďte analýzu uživatelských požadavků na podnikovou aplikaci a vytvořte pomocí vhodných nástrojů model požadavků

3. Pro vytipované funkční a nefunkční požadavky na zvolenou aplikaci vypracujte případy užití a scénáře dle metodiky Unified Process

Doporučený rozsah práce

30-40 stran

Klíčová slova

model požadavků, případy užití, vykazování hodin

Doporučené zdroje informací

KANISOVÁ, H. – MÜLLER, M. UML srozumitelně. Brno: Computer Press, 2007. ISBN 80-251-1083-4.

NEUSTADT, I. – ARLOW, J. *UML 2 a unifikovaný proces vývoje aplikací : objektově orientovaná analýza a návrh prakticky*. Brno: Computer Press, 2007. ISBN 978-80-251-1503-9.

Předběžný termín obhajoby

2019/20 LS – PEF

Vedoucí práce

Ing. David Buchtela, Ph.D.

Garantující pracoviště

Katedra informačního inženýrství

Elektronicky schváleno dne 14. 3. 2020

Ing. Martin Pelikán, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 14. 3. 2020

Ing. Martin Pelikán, Ph.D.

Děkan

V Praze dne 21. 03. 2020

Čestné prohlášení

Prohlašuji, že svou bakalářskou práci "Návrh modelu uživatelských požadavků na podnikovou webovou aplikaci" jsem vypracoval(a) samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autor(ka) uvedené bakalářské práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 18. 03. 2020 _____

Poděkování

Rád bych touto cestou poděkoval Ing. Davidu Buchtelovi, Ph.D., vedoucímu práce, za trpělivost a odborné vedení této práce.

Návrh modelu uživatelských požadavků na podnikovou webovou aplikaci

Abstrakt

Obsahem této bakalářské práce je analýza a vytvoření modelu uživatelských požadavků dle metodiky Unified Process. Tento model slouží pro následný vývoj podnikové webové aplikace na vykazování hodin zaměstnanců do jednotlivých projektů.

Praktická část mé bakalářské práce je tedy zaměřená na požadavky, tj. přání a potřeby klienta na danou podnikovou webovou aplikaci. Požadavky od klienta jsem si rozdělil do dvou kategorií, funkční a nefunkční. Funkční požadavky jsem dále zpracoval do jednotlivých případů užití, včetně možných alternativních scénářů.

Vytváření modelu uživatelských požadavků je pouze počáteční, ale zato důležitá fáze, bez které by se kvalitní finální aplikace nedala vytvořit.

Klíčová slova: analýza uživatelských požadavků, vytvoření modelu uživatelských požadavků, model požadavků, případy užití, vykazování hodin, Unified Process, UP, UML

Design of User Requirements' Model for Enterprise Web Application

Abstract

The content of this thesis is to analyze and create a model of user requirements according to the Unified Process methodology. This model is used for the subsequent development of an enterprise web application for reporting employee work hours to individual projects.

The practical part of my thesis is focused on the requirements, ie the wishes and needs of the client for the company web application. I divided the requirements from the client into two categories, functional and non-functional. Functional requirements were further elaborated into individual use cases, including possible alternative scenarios.

Creating a user request model is only an initial but very important phase without which a final application could not be created.

Keywords: user requirements analysis, user requirements model creation, requirements model, use cases, work hours reporting, Unified Process, UP, UML

Obsah

Seznam obrázků

Seznam tabulek

1 Úvod

V dnešním světě, kde je vše zaměřené na moderní technologie, se čím dál tím víc věcí převádí do digitální podoby. Mezi stěžejní důvody této digitalizace patří úspora času, snížení administrativní zátěže a v neposlední řadě zvýšení přehlednosti evidovaných dat. To jsou důvody, proč se vytvářejí různé procesy, informační systémy a řada dalších aplikací.

Jak řekl Bjarne Stroustrup, tvůrce jazyka C++: „Naše civilizace je řízená softwarem.“. Software se ve skutečnosti opravdu dotýká mnoha aspektů našeho života, od těch úplně obyčejných narozeninových přání, která po otevření hrají, přes ty všudypřítomné jako je mobilní telefon nebo počítač, až po složité systémy, jako jsou například letadla či jaderné elektrárny. (1, s. 16)

Většina podniků a organizací závisí právě na softwaru, ať již jde o běžně dostupný software, či software dělaný přímo na míru. Účelem těchto softwarů je usnadnění a zrychlení práce všech zúčastněných, jak zaměstnavatelů, tak zaměstnanců.

Pokud si chce podnik nechat vytvořit vlastní interní systém má dvě možnosti, buď si na jeho vývoj najme externí firmu, nebo daný vývoj realizuje vlastními zaměstnanci. Vždy záleží na konkrétních možnostech a potřebách daného podniku. Pokud si podnik vybere externího dodavatele, vývoj bývá levnější a rychlejší, ale veškeré požadavky se musí specifikovat ve smlouvě. Pozdější úpravy jsou složitější a časově náročnější.

Vývoj vlastními zaměstnanci bývá dražší a pomalejší, ale z dlouhodobějšího hlediska podstatně efektivnější. Poskytá možnosti pozdějších úprav a přizpůsobení se novým případným změnám daného podniku. Mezi další výhody vývoje vlastními zaměstnanci se řadí ochrana dat daného podniku, jelikož k těmto datům mají přístup pouze zaměstnanci podniku. Osobní údaje zaměstnanců se rovněž, v případě vývoje vlastními zaměstnanci, lépe ochrání.

Každý podnik si zvolí tu variantu, která mu více vyhovuje. Vytvoření modelu uživatelských požadavků se liší právě v závislosti na volbě dané varianty. V této práci se zabývám analýzou a vytvořením modelu uživatelských požadavků pro podnik pana Nováka, který se zabývá reklamou. Jeho podnik se rozrostl a potřebuje nový software na evidenci odpracovaných hodin svých zaměstnanců. Tento software by měl být otevřený, do budoucna snadno rozšiřitelný a popřípadě by měl umožnit změnu funkcionality, v závislosti na potřebách podniku. Pan Novák se z finančních a časových důvodů rozhodl pro externí firmu.

V praktické části mé práce je popsán proces analýzy a vytvoření modelu uživatelských požadavků, který vychází z potřeb tohoto podniku a bude základem pro tvorbu požadované aplikace.

2 Cíl práce a metodika

2.1 Cíl práce

Cílem práce je analýza a vytvoření modelu uživatelských požadavků dle metodiky Unified Process pro podnikovou webovou aplikaci vykazování hodin, která bude sloužit vykazování hodin zaměstnanců do jednotlivých projektů.

Jde o proces navrhování podnikové aplikace, jejíž hlavním účelem je vykazování hodin zaměstnanců do jednotlivých projektů, podprojektů a konkrétních úkolů. Tato aplikace by měla poskytnout přehled všech projektů podniku a zároveň vytíženost jednotlivých zaměstnanců. Zaměstnavatel by tak měl větší přehled o všech projektech, v jaké fázi rozpracování jsou a zda může plánovat nové projekty.

Návrh aplikace je velice obsáhlý a zahrnuje několik aktivit, jako jsou požadavky, analýza, návrh, implementace a testování. Ve své práci se proto zaměřuji pouze na první z nich, tj. požadavky. Je to nedůležitější část celého návrhu aplikace.

2.2 Metodika

V první kapitole teoretických východisek popisuji jazyk UML, který je universálním grafickým jazykem pro vizualizaci, specifikaci, navrhování a dokumentaci programových systémů. Popisuji nástroje CASE, které z tohoto modelovacího jazyka vycházejí. Dále se zde zabývám strukturou, která obsahuje stavební bloky, společné mechanismy a architekturu.

Ve druhé kapitole teoretických východisek se zabývám vývojem metodiky Unified Process. Tato metodika je dále rozpracována ve třetí kapitole, dle jednotlivých fází, a to zahájení, rozpracování, konstrukci a zavedení. Čtvrtá kapitola se věnuje aktivitám, na které se metodika UP dělí, a to požadavkům, analýze, návrhu, implementaci a testování. Ve své bakalářské práci se zabývám pouze první z nich, tj. požadavky.

Ve vlastní části mé bakalářské práce se tedy zaměřuji na požadavky, tj. přání a potřeby klienta na danou podnikovou webovou aplikaci.

V první kapitole vlastní části popisuji klienta a jeho zázemí. V druhé kapitole se věnuji požadavkům klienta, které jsem si rozdělil do dvou kategorií, funkční a nefunkční.

Funkční požadavky jsou dále zpracovány ve třetí kapitole v jednotlivých případech užití, včetně možných alternativních scénářů.

3 Teoretická východiska

V této práci se zabývám tvorbou webových aplikací. Tvorba aplikací je velice náročná. Aby byla daná aplikace dlouhodobě udržovatelná, je zvykem, aby se vývoj dané aplikace rozlišil do jednotlivých fází, které vytváří životní cyklus vývoje softwaru (software development life cycle). Toto rozlišení podstatně zlepšuje architekturu, rychlost vývoje a v neposlední řadě management projektu.

Pro návrh aplikací a systémů se používají různé procesy a metodiky, které se dělí na několik základních paradigmat. Mezi hlavní paradigmata se řadí: agilní, waterfall, prototypovací (prototyping), spirální (spiral development), extrémní (extreme programming), rapidní (rapid application development), a v neposlední řadě iterativní a inkrementální (iterative and incremental development).

Právě poslední zmíněný jsem si zvolil pro tuto práci, z důvodu jeho použitelnosti i v jiných oborech. Počátky má právě v návrhu softwaru, ale v dnešní době se používá i v hardwaru. SpaceX přinesl iterativní návrh do leteckého a vesmírného průmyslu, když jako první dokázala v březnu 2017 znovu použít nosnou raketu k vynesení satelitu SES-10, což je geostacionární telekomunikační družice, do vesmíru. (2)

Jedna z nejznámějších metodik iterativního a inkrementálního paradigmatu je **Unified Process** (**UP**), která je základem této práce.

3.1 UML

Název jazyka UML vychází z anglického názvu Unified Modeling Language, jde o univerzální grafický jazyk pro vizualizaci, specifikaci, navrhování a dokumentaci programových systémů. Má široké využití, a díky tomu se z něho stal standard, který poskytuje vizuální syntaxi při sestavování modelů.

Tento jazyk nenabízí žádný druh metodiky modelování, a tudíž není vázán na žádnou konkrétní metodiku, nebo životní cyklus. Lze jej použít s různými metodami, ale upřednostňovaná metoda je Unified proces, která je pro tento jazyk nevíce přizpůsobená. (3, s. 28)

Od roku 1994, kdy byl jazyk UML založen na sjednocení metodik OMT, Booch a CRC se postupně vyvíjel a v roce 1997 se stal otevřeným průmyslovým standardem. Jazyk UML spojuje mnoho nejlepších myšlenek vycházejících z původních metod. Společnost

OMG (Object management group), která tento jazyk neustále vylepšuje a rozvíjí vydala v roce 2017 verzi 2.5.1.

3.1.1 Struktura

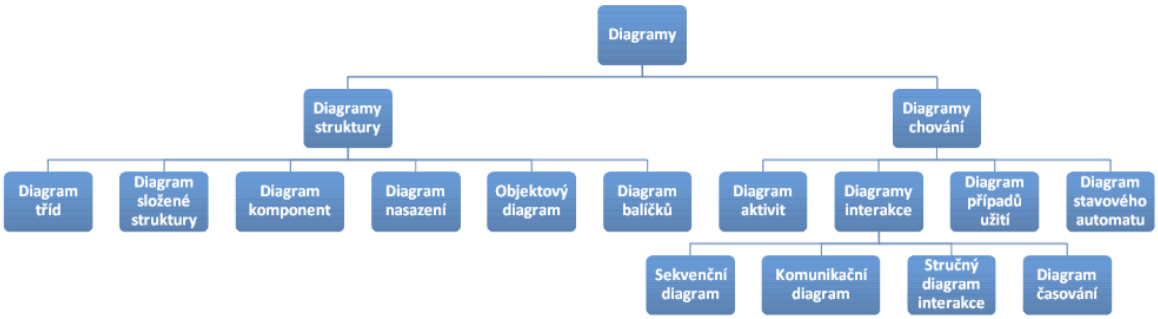
Struktura jazyka UML obsahuje následující součásti:

a. **Stavební bloky** – základní prvky modelu, relace, a diagramy,
   - Předměty – jsou prvky modelu
     - Strukturální abstrakce (Structural things) – podstatná jména modelu UML (třída, rozhraní, spolupráce, případ užití, aktivní třída, komponenta, uzel)
     - Chování (Behavioural things) – slovesa modelu UML (např. interakce, stav)
     - Seskupení (Grouping things) – seskupování významově souvisejících prvků modelu UML do balíčků
     - Poznámky (Annotational things) – anotace
   - Relace (Relationships) – určují jak spolu dva nebo více předmětů významově souvisí

Typ relace	Syntaxe UML zdroj cíl	Stručný popis
závislost		Změna v určitém předmětu ovlivňuje význam závislého předmětu
asociace		Popis množiny spojení mezi objekty
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 klasif.

Obrázek 1 Přehled UML relací (4)

- Diagramy (Diagrams) – pohledy na modely UML
  UML má celkem 13 různých typů diagramů, které můžeme vidět na následujícím obrázku.



Obrázek 2 Strom rozdělení UML diagramů (4)

b. **Společné mechanismy** – obecné způsoby, kterými dosáhneme v jazyce UML specifických cílů:
   - Specifikace – jsou jádrem modelu UML a textovým popisem sémantiky jednotlivých prvků
   - Ornamenty – každý prvek modelu lze vyjádřit symbolem, který můžeme doplnit několika ornamenty
   - Podskupiny – znázorňují různé pohledy na svět
     V jazyce UML rozlišujeme dvě takové podskupiny, jde o skupinu klasifikátorů a instancí a skupinu rozhraní a implementací.

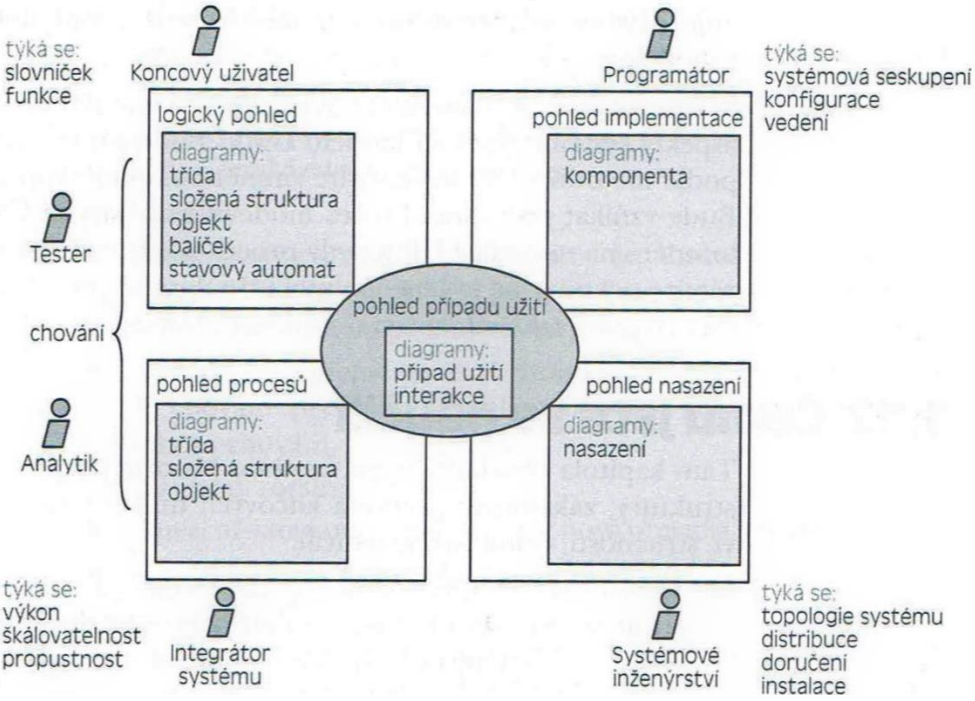
     Klasifikátor je abstraktní vyjádřením typu předmětu (např. bankovní účet), zatímco instance je již konkrétní výskytem abstraktní představy (např. můj či Váš bankovní účet). Je zvykem, že se instancím v UML modelu podtrhává název.

     Rozhraní definuje dohodu, která určuje, co musí obsahovat daná implementace.
   - Mechanismy rozšiřitelnosti – máme celkem 3 mechanismy:
     - Omezení (Constraints), jde o text, který specifikuje danou podmínku, která musí být vyhodnocena jako pravda,

- Stereotypy (Stereotypes), umožnují vytvářet nové prvky z jiných prvků daného modelu,
- Označené hodnoty (Tagged values), umožnují rozšiřovat prvky modelu o jejich vlastní vlastnosti

c. **Architektura** – nejvyšší úroveň koncepce systému v jeho vlastním prostředí Jazyk UML definuje čtyři různé pohledy na systém: logický pohled, pohled procesů, pohled implementace a pohled nasazení. Všechny tyto pohledy dále tvoří pohled pátý, znázorněný na následujícím obrázku.

(3, s. 34-47)



Obrázek 3 Pět základních pohledů na systém (3 str. 47)

Z modelovacího jazyka UML vychází nástroje CASE (Computer-aided software engineering), které umožnují modelování systémů pomocí diagramů, generování zdrojového kódu pomocí modelu, vytvoření modelu ze zdrojového kódu (reverse engineering) a v neposlední řadě vytvoření dokumentace. (4)

Mezi nejužívanější nástroje CASE se řadí Visual Paradigm, Enterprise Architect, MagicDraw, Oracle Designer, Rational Rose a MS Visio. (5, s. 12)

3.2 **Unified Process**

Unified Process (UP) je velice pragmatickou a obecnou metodou pro tvorbu aplikací, která vychází z metodiky Rational Unified Process (RUP). RUP vytvořila firma Rational v roce 1998, ze sloučení metod Ericsson a Rational. V následujícím roce 1999 Jacobson publikoval knihu Uniffied Softwate Development Process, v které uvádí právě metodiku Uniffied Process. V roce 1999 UP a RUP metodiky byly prakticky totožné, jejich největší rozdíl byl, že metodika UP nebyla komerční produkt, ale otevřený standard, který byl dostupný všem. V dnešní době RUP navíc ke všemu, co obsahuje UP, dále také obsahuje bohaté uživatelské rozhraní a podrobnou dokumentaci.

Metodiku UP, dle (3, s. 59), tvoří tři základní axiomy:

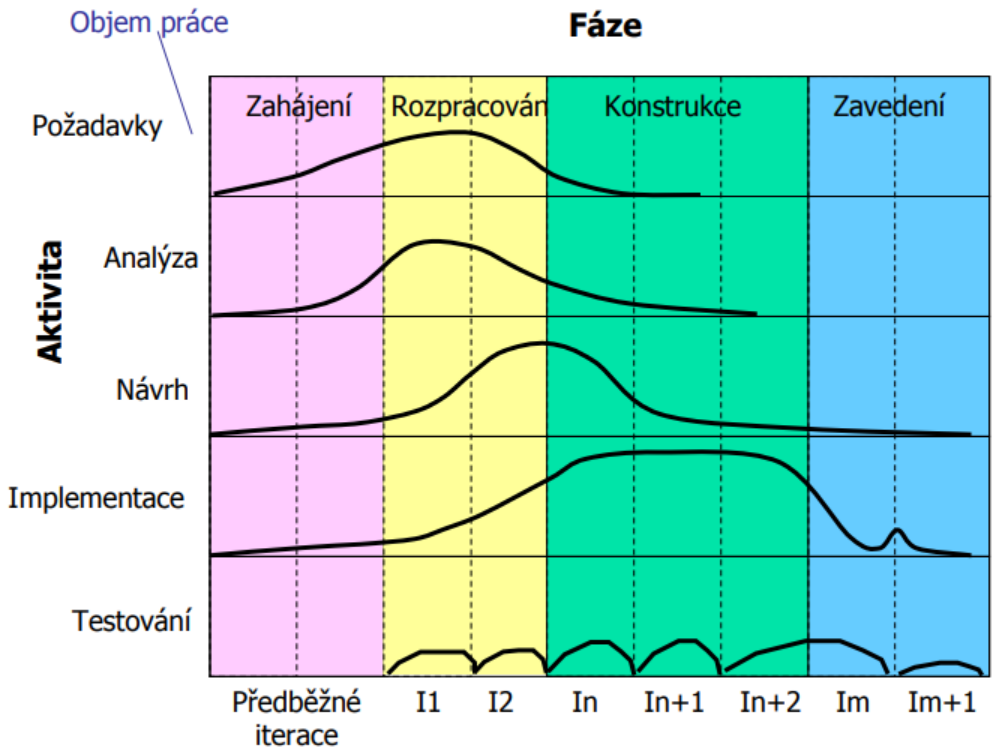
- zásada iterace a přírůstku (inkrementu),
- zásada soustředění se na architekturu,
- zásada řízení případem užití a rizikem.

Metodika UP stojí na iterativním postupu. Iteraci lze chápat jako rozdělení projektu na menší části, které se dále snadněji zpracovávají. Iterace tedy prochází celým procesem od plánování přes analýzu a návrh, tvorbu, integrací a testováním, až po interní nebo externí uvedení. Každá iterace obsahuje pět základních pracovních postupů (požadavky, analýza, návrh, implementace a testování) a další pracovní postupy (plánování, odhad a specifikace projektu), které nám upřesňují, co je třeba udělat a jak toho dosáhnout.

Zásada soustředění se na architekturu se zakládá na tvarování systému. Nelze vytvořit model, který by byl dostačující k pokrytí všech aspektů daného systému, a tudíž Unified Process podporuje více architektonických modelů a pohledů.

Zásada řízení případem užití a rizikem, vyžaduje, aby se projektový tým soustředil hlavně na nejkritičtější rizika, co nejdříve. Dochází tak k zabránění vzniku nových problémů, již v počátečních fázích.

Na následujícím obrázku vidíme čtyři fáze životního cyklu projektu, a to zahájení, rozpracování, konstrukce a zavedení. Tyto fáze se dělí na jednu nebo více iterací, jejíchž počet je závislý na velikosti projektu. Délka jedné iterace by nemá být delší než dva měsíce. Z následujícího obrázku vyplývá, kolik času je třeba vynaložit v jakékoliv fázi na jakou aktivitu. (3, s. 54-62)



Obrázek 4 Diagram metodiky UP (6)

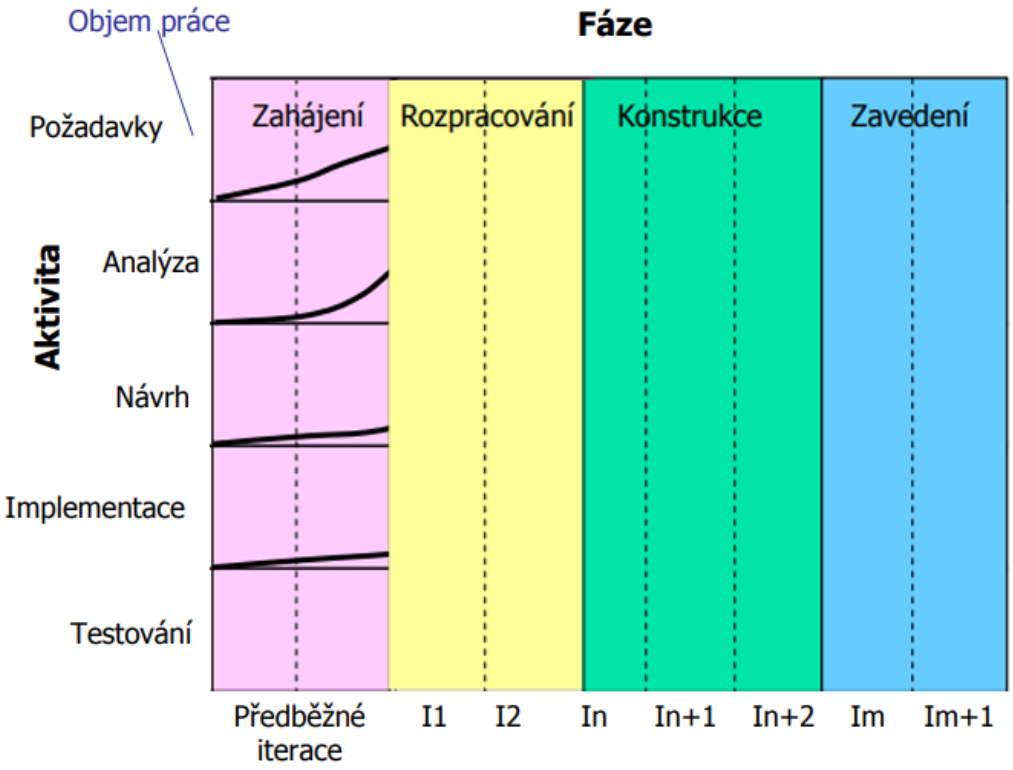
3.3 Fáze UP

Metodika UP se dělí na čtyři fáze: zahájení, rozpracování, konstrukci a zavedení.

3.3.1 Zahájení (Incepcion)

Fáze zahájení začíná v momentě, kdy klient přijde za projektovým manažerem s požadavky na tvorbu projektu. V této fázi se klade největší důraz na specifikaci požadavků a jejich následnou analýzu. Dále se zaměřujeme na tvorbu podmínek proveditelnosti (např. návrh prototypů a simulací) a možná rizika. V této fázi se neprovádí testování.

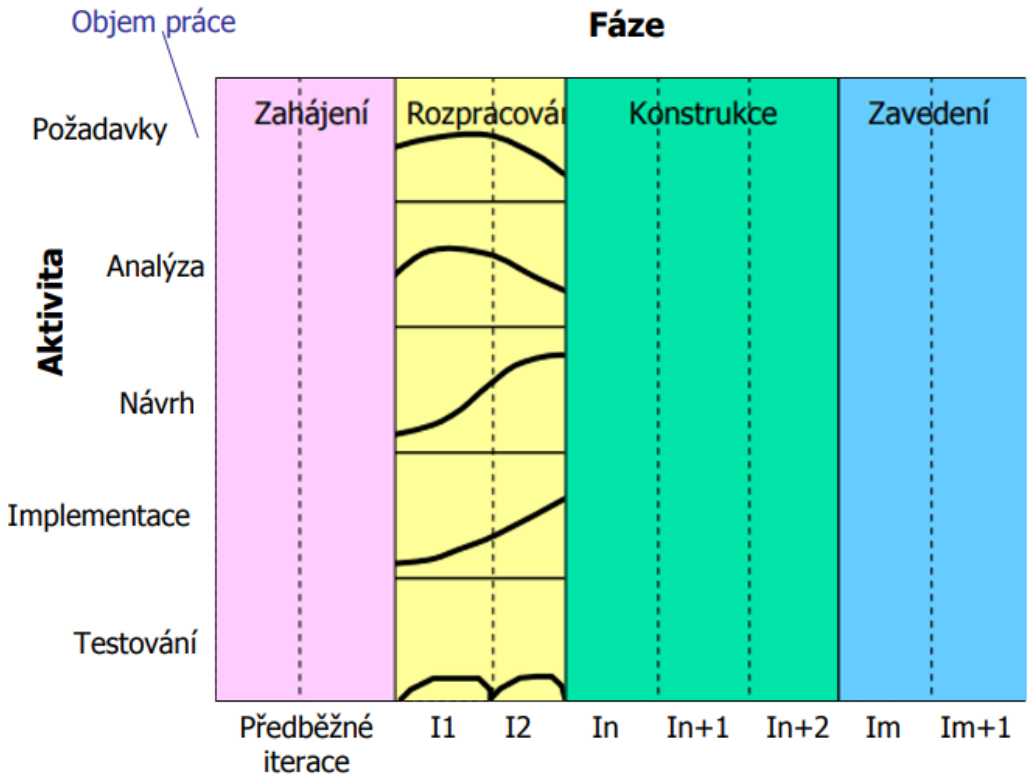
Tato fáze probíhá v několika konzultacích, kde klient neustále upřesňuje své požadavky a tím určuje počáteční směr projektu. (3, s. 63-64)



Obrázek 5 Diagram metodiky UP – zahájení (6)

3.3.2 Rozpracování (Elaboration)

Rozpracování je fáze, která přímo navazuje na fázi zahájení. V této fázi je cílem vytvořit neúplné ale funkční verzi systému – spustitelného architektonického základu. Tento základ se bude dále vyvíjet a rozšiřovat během následujících fází. Mezi další cíle této fáze patří: upřesnění odhadu rizik, odhalení případných nedostatků, zachycení případu užití pro 80% funkčních požadavků, naplánování konstrukční fáze a neposlední řadě vytvoření konkrétní finální nabídky. (3, s. 65)

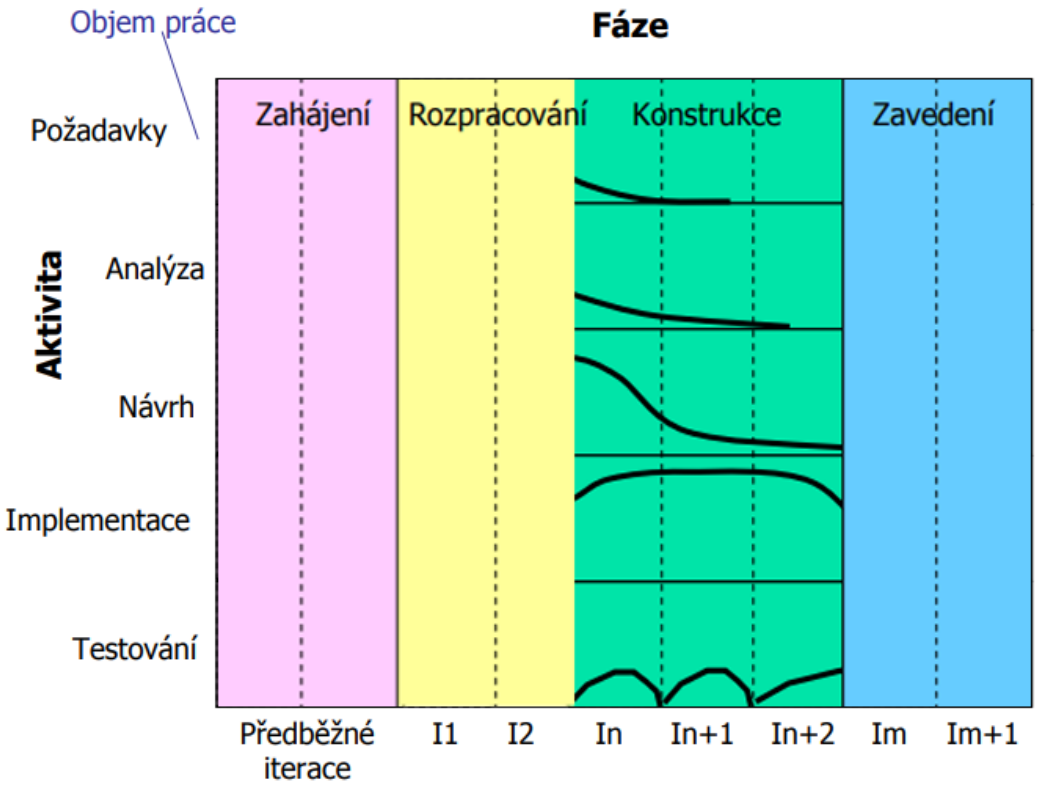


Obrázek 6 Diagram metodiky UP – rozpracování (6)

3.3.3 Konstrukce (Construction)

Cílem konstrukční fáze je postupně přeměnit spustitelný architektonický základ na kompletní a funkční systém. V této fázi je velice důležité zachování původního návrhu projektu. Často dochází k upuštění od původního plánu, vlivem nedostatku času, a to má za následek degradaci kvality finálního projektu.

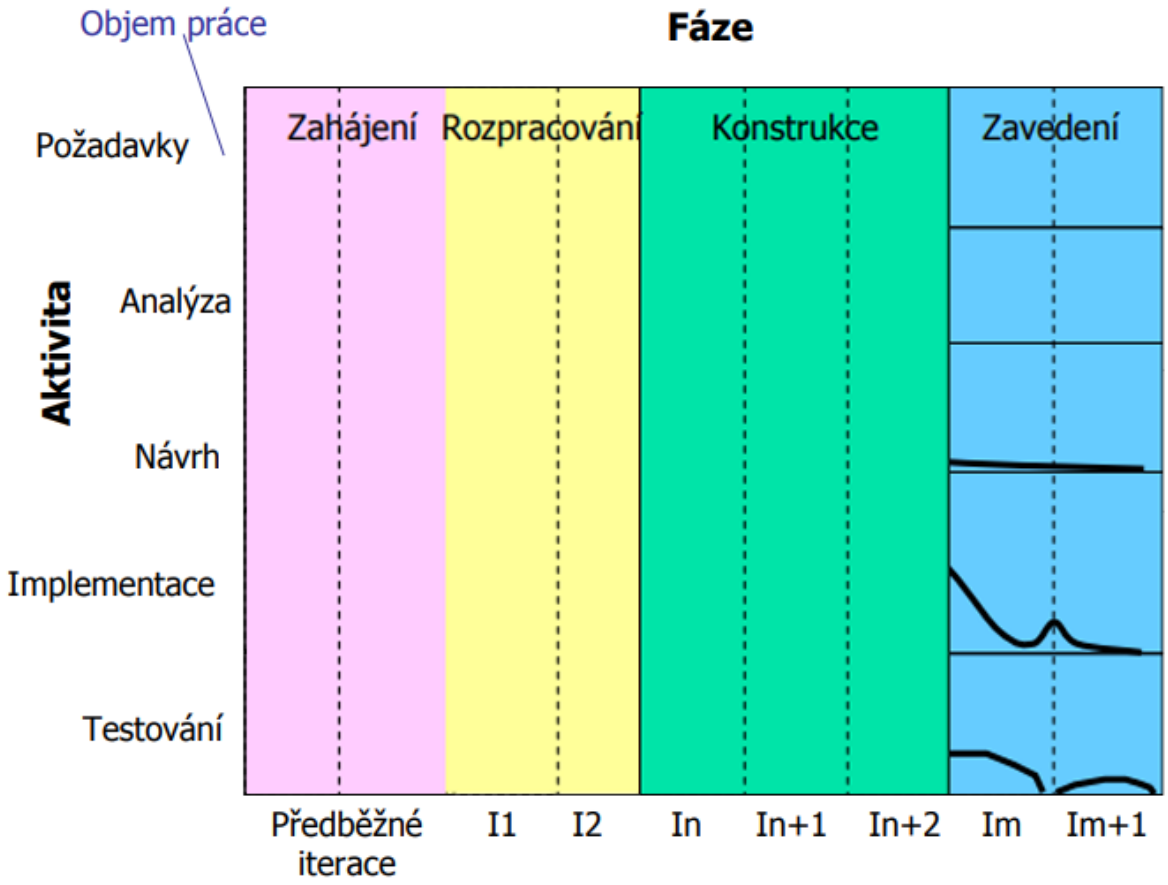
V této fázi se klade největší důraz na implementaci, požadavky a analýza se dokončují a testuje se počáteční funkční varianta. Na konci této fáze je projekt připraven pro beta testování uživateli. (3, s. 66)



Obrázek 7 Diagram metodiky UP – konstrukce (6)

3.3.4 Zavedení (Transition)

Fáze zavedení je nasazení hotového systému na pracoviště uživatele. Do této finální fáze patří: oprava chyb, příprava počítačů klienta na přijetí nového programu, nastavení daného programu, konzultace s uživateli a vytvoření dokumentace a dalších podkladů. Celá tato fáze je zakončená finální revizí. (3, s. 67-68)



Obrázek 8 Diagram metodiky UP – zavedení (6)

3.4 Aktivity UP

Metodika UP se dělí na pět aktivit: požadavky, analýza, návrh, implementace a testování.

3.4.1 Požadavky

Před tím, než se pustíme do tvoření aplikace, je třeba specifikovat požadavky na danou aplikaci, jde totiž o úplný začátek procesu tvorby softwaru.

Specifikace softwarových požadavků se skládá ze dvou modelů: modelu požadavků a modelu případu užití. Tyto modely zachycují různé informace, ale jako celek definují požadavky zadavatele na zadanou aplikaci.

3.4.1.1 Model požadavků

Model požadavků je soubor požadavků na danou aplikaci, vyjadřuje vizi aplikace, kterou zadavatel dané aplikaci potřebuje. Zadavatel by při zadání práce měl mít jasnou představu o tom co požaduje. Pokud by zpracovatel neměl dostatek informací, měl by si je zjistit ve vzájemné spolupráci se zadavatelem.

Jednou z nejčastějších příčin neúspěchu aplikací bývají právě nedostatečně specifikovány požadavky.

Požadavek je vlastnost či funkce, kterou zadavatel očekává ve výsledné aplikaci. Požadavky by tudíž měly vypovídat co by daná aplikace měla dělat, nikoliv jak by to měla dělat. Důvodem toho je, že mohou vzniknout dva požadavky, které se vzájemně vylučují, nebo v požadavku může být uvedená určitá technologie, která nemusí být aktuálně dostupná, či perspektivní pro danou aplikaci.

Forma zápisu požadavků není daná, většina firem je zapisuje pouze v přirozeném jazyce v různých podobách. Vhodné je stanovit si předem formu zápisu a důležité body, co musí obsahovat. Pro lepší a přehlednější zápis, můžeme jednotlivé požadavky sepsat do tabulky nebo do XML formátu, což usnadní práci s požadavky jak zadavateli, tak i zpracovateli. Tabulka může obsahovat údaje jako například: číselný identifikátor, popis požadavku, prioritu požadavku (např. nezbytný, možný a eventuální) a termín uvedení do provozu.

Rozlišujeme dva druhy požadavků: funkční a nefunkční.

3.4.1.1.1 Funkční požadavky

Pod pojmem funkční požadavek si můžeme představit konkrétní funkci, kterou by daná aplikace měla dělat. Například pokud navrhujeme výpůjční systém pro knihovnu, je naprosto důležité, aby systém umožnoval svým zákazníkům provést výpůjčky a samozřejmě vypůjčené knihy zase vracet.

3.4.1.1.2 Nefunkční požadavky

Nefunkční požadavky jsou nejčastěji podmínky, které nějakým způsobem omezují danou aplikaci. Většinou se jedná o preferovaný jazyk nebo komunikační interface.

Na pojem omezení se můžeme dívat jak z hlediska obchodního, či podnikového (např. zákonná omezení či omezení zdrojů), tak i technického (např. předepsané technologické standardy, či předepsané prvky řešení). (1, s. 119-120)

3.4.1.2 Model případů užití

Modelování případu užití je jeden ze způsobů, jak získávat a zaznamenat požadavky klienta. Model případu užití zachycuje funkce dané aplikace z pohledu aktérů. Čím více aktérů, tím je tento model případu užití přínosnější a efektivnější, a naopak čím méně aktérů, tím více tento model postrádá svůj smysl.

Tento způsob se skládá ze čtyř základních částí: hranic systémů, aktérů, případů užití a relací.

3.4.1.2.1 Hranice systému

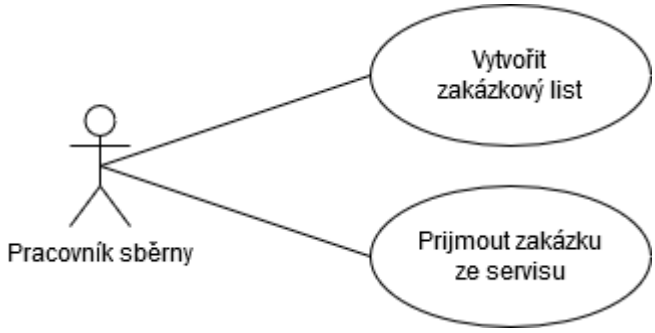
Hranice systému neboli subjekt je stanovení hranic, které určují, co by mělo být součástí dané aplikace, a co by naopak nemělo být její součástí. I když to zní jako samozřejmost, často se to v dnešní době opomíjí. V důsledku tohoto opomíjení může dojít až k neúspěchu daného projektu.

Hranice systému jsou specifikované klientem a vycházejí z jeho potřeb.

3.4.1.2.2 Aktéři

Aktér je určitá role, jejíž reprezentant, pracuje přímo s danou aplikací. Pro určení aktéra je nezbytné specifikovat kdo nebo co danou aplikaci používá a jakou roli ve vztahu k ní zastupuje.

Pojmem aktér můžeme chápat jako určitou roli, kde uživatel vystupuje v rámci své komunikace se systémem. Aktéři spouštějí případy užití. Jeden aktér může realizovat řadu případů užití, a naopak jeden případ užití může být realizován více aktéry. Na následujícím obrázku lze vidět jednoduchý diagram případu užití jednoho aktéra, pracovníka sběrny. Přestože je ve sběrně zaměstnáno více pracovníků, vůči informačnímu systému mají všichni stejnou roli. (5, s. 38)



Obrázek 9 Diagram případu užití pracovníka sběrny

3.4.1.2.3 Případy užití

Případ užití je požadavek aktéra na danou aplikaci, tj. jeho očekávání a potřeby. K hledání případů užití se využívá metoda postupného upřesňování. Na jednotlivé kostry se dále nabalují další a další podrobnosti, které nakonec vystihují celý případ užití.

Případ užití obsahuje přesně tu funkčnost, kterou bude daný sytém zvládat, a tím vymezuje rozsah celé práce. To znamená, že by se mělo programovat pouze to, co soubor užití popisuje, tj. žádnou jinou funkčnost. (5, s. 37)

Vzhledem k tomu, že případy užití zachycují funkci systému z pohledu aktérů, jsou málo efektivní, má-li systém jednoho aktéra, nebo dokonce nemá-li žádného. Případy užití zachycují funkční požadavky. Nejsou tedy účinné v systémech, nichž převládají požadavky nefunkční.

Ideální pro vytvoření případu užití, je si předem vytvořit šablonu a podle ní poté zapisovat dané případy užití. To nám zapříčiní, že všechny záznamy budou unifikovány. V šabloně by se měly nacházet určitě alespoň tyto body: unikátní identifikátor, název, popis, aktéři a scénář.

Správně sestavený scénář je základ správného případu užití. Scénář neboli tok událostí, je sekvence kroků, která popisuje interakce mezi aktéry a systémem. Tyto scénáře je vhodné zapisovat do tabulky, nebo pomocí odrážek, aby bylo zřejmé kde daný krok začíná a kde končí.

3.4.1.2.4 Relace

Relace jsou vztahy mezi případy užití, rozlišujeme dva typy: include a extend. Relace include je vztah mezi případy užití, kdy jeden případ užití začleňuje chování jiného případu užití. Zatímco relace extend je vztah mezi případy užití, kdy jeden případ užití rozšiřuje své

chování (svou funkci) pomocí jednoho nebo více fragmentů chování jiného případu užití. (3, s. 74-132)

3.4.2 **Analýza**

Analýza nám předurčuje základní chování aplikace. Jejím záměrem je vytvoření analytického modelu, který se zaměřuje na to, co sytém musí udělat, ale neřeší detaily způsobu, jakým to udělá. To se následně řeší v návrhu. Hranice mezi analýzou a návrhem je totiž velice nejasná a závisí na individuálním cítění daného analytika.

Analýza se skládá ze dvou částí:

- analytických tříd,
- realizaci případů užití.

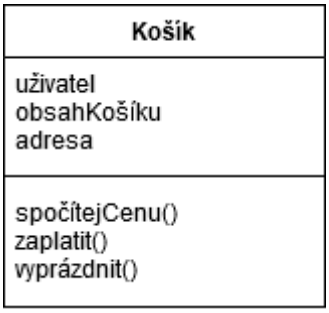
Analytické třídy jsou souborem pojmů v obchodní doméně. Realizace případů užití ukazují, jak mohou instance analytických tříd vzájemně komunikovat s cílem realizovat chování systému specifikovaném případu užití. (3)

Ve fázi analýzy bychom se měli vyvarovat veškerých implementačních rozhodnutí. Tato rozhodnutí kvůli zachování určité hladiny abstrakce patří do návrhu a implementace.

3.4.2.1 Analytické třídy

Analytické třídy jsou abstrakcí skutečných věcí ve skutečném světě, které modelují zásadní části problémové domény. Problémová doména je prvotní požadavek na vznik softwaru. Analytické třídy mapují skutečné obchodní pojmy, jako například „produkt“, „objednávka“ nebo „košík“.

Každá dobře navržená abstraktní třída musí mít název, který definuje účel dané třídy, musí být soudržná a počet vazeb na jiné třídy, by měl být co nejmenší.



Obrázek 10 Příklad analytické třídy

Analytické třídy by měly obsahat jen ty atributy, které jsou významné pro pochopení základního smyslu. Podle syntaxe jazyka UML se třída značí obdélníkem, rozděleným na tři části. Do první části se píše název, který je vždy povinný. Další část obsahuje atributy, což jsou vlastnosti dané třídy. Každý atribut musí obsahovat název a volitelně i typ, ale příliš se nedoporučuje jeho používání v analytických třídách. V poslední části se nacházejí operace, jde o funkce dané třídy.

3.4.3 **Návrh**

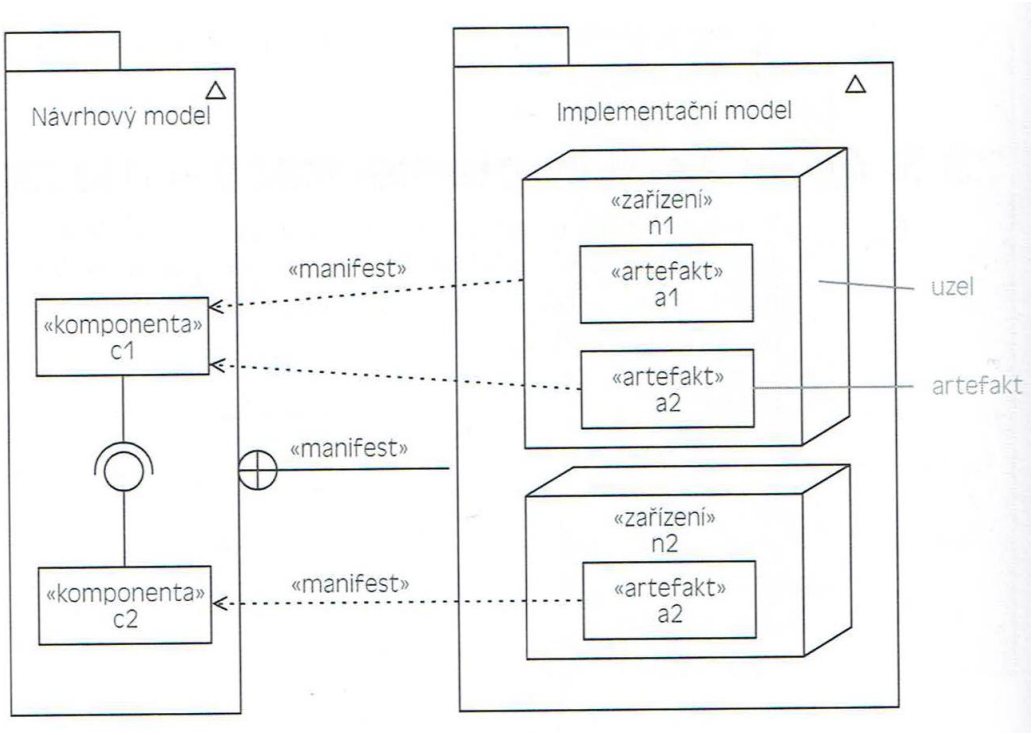
Zatímco analýza je zaměřena hlavně na tvorbu logického modelu připravovaného systému, který zachycuje funkce, jež tento systém musí poskytovat, aby uspokojil požadavky uživatelů, smyslem návrhu je přesný specifikace způsobu, jak takové funkce implementovat. Pro zefektivnění modelování se používají návrhové třídy, které jsou stavebními bloky daného modelu. (3)

3.4.4 **Implementace**

Jak je možné vidět na obrázku 11 implementace začíná již ve fázi rozpracování, ale jeho největší část je tvořena ve fázi konstrukce. Primární účel implementace je vytvoření spustitelného kódu pomocí návrhu. Tato transformace je v režii programátorů.

Na rozdíl od předchozích aktivit, implementace je závislá na výběru programovacího jazyka. Z tohoto důvodu, při změně programovacího jazyka, můžeme s implementací začít téměř od začátku.

Implementační model je část návrhového modelu a upřesňuje implementaci. Návrhový model obsahuje komponenty, které se v implementačním modelu dále převádí na uzly a artefakty. Uzly jsou konkrétní kusy hardwaru nebo prostředí, na které jsou nasazovány artefakty (např. zdrojové soubory).



Obrázek 11 Relace mezi návrhovými a implementačními podsystémy (3, s. 464)

3.4.5 Testování

Při vývoji webových aplikací se na testování často zapomíná, většinou je realizována až v pozdějších fázích vývoje.

Dokonce se stává že část softwaru je testována velmi málo, nebo vůbec, a tudíž některé chyby objeví až koncový uživatel. Aby se tomuto zabránilo, hodně společností nařizují testování již v průběhu vývoje daného softwaru. (7, s. 1)

Testování hraje ve vývoji softwaru velice důležitou roli, čím dříve se na chybu přijde, tím jednodušší je náprava, jak z časového, tak z finančního hlediska. Paradoxně právě z nedostatku času není často testování realizováno.

4 Vlastní práce

Návrh podnikové webové aplikace je velice obsáhlý a zahrnuje několik aktivit, jako jsou požadavky, analýza, návrh, implementace a testování.

Ve své práci se zaměřuji pouze na první z nich, tj. požadavky. Zabývám se analýzou a tvorbou modelu uživatelských požadavků pomocí metodiky Unified Process, jde to nedůležitější část celého návrhu. Požadavky se zpracovávají na základě spolupráce s klientem, tak aby co nejvíce vyhovovaly konkrétním potřebám daného podniku. Jsou základním pilířem pro následnou realizaci podnikové webové aplikace.

4.1 Popis klienta

Podnik pana Nováka, zabývající se reklamou, se rozrostl a potřebuje nový software na evidenci odpracovaných hodin svých zaměstnanců. Tento software by měl být otevřený, do budoucna snadno rozšiřitelný a popřípadě by měl umožnit změnu funkcionality, v závislosti na potřebách podniku.

Tento podnik byl založen před dvěma roky, v současné době má celkem 42 zaměstnanců. Tito zaměstnanci pracují na různých pozicích, jako účetní, personalista, designer, tiskař, programátor, prodejce, operátor a uklízeč.

Podnik má své webové stránky, na kterých nabízí prodej svých výrobků. Mezi nejprodávanější výrobky patří potisk triček, propisek a hrníčků. Dále podnik nabízí tisk a potisk různých reklamních materiálů.

Do současné doby podnik používal pouze tabulkový editor, který s rostoucím počtem zaměstnanců a rostoucími požadavky pro evidenční správu není dostatečný. V minulosti si podnik nechal externě vytvořit několik scriptů do tabulkového editoru, které slouží k vytvoření čtvrtletních reportů. Nově požadovaný software proto musí umět exportovat data do formátu CSV.

4.2 Sběr požadavků

Od podniku pana Nováka jsme dostali požadavky na vytvoření podnikové webové aplikace, které ale byly nestrukturované a nepřehledné. Mezi požadavky klienta bylo i několik věcí, které byly zbytečné, duplicitní, a dva požadavky byly dokonce protichůdné.

Daným požadavkům jsem moc nerozuměl, proto jsem si sjednal konzultační schůzku. Na této schůzce jsme si dané požadavky upřesnili, přepsali do bodů a dále seskupili dle funkčních a nefunkčních kategorií. Funkční kategorie jsme si dále roztřídili do podkategorií, a to evidence hodin, statistiky, kalendář a zprávy.

4.2.1 **Funkční požadavky**

4.2.1.1 Vykazování hodin

1. Aplikace bude umožnovat zaměstnanci zažádat správce o vytvoření projektu.
2. Aplikace bude umožnovat zaměstnanci vytvořit podprojekt v projektu, ke kterému má oprávnění.
3. Aplikace bude umožnovat zaměstnanci vytvořit úkol v projektu, ke kterému má oprávnění.
4. Aplikace bude umožnovat zaměstnanci evidovat si pracovní dobu v daném úkolu.

4.2.1.2 Statistiky

5. Aplikace bude vytvářet statistiky jednotlivých zaměstnanců.
6. Aplikace bude vytvářet statistiky projektů, podprojektů a úkolu.
7. Aplikace bude umožnovat zaměstnanci zobrazit si pouze svoje statistky.
8. Aplikace bude umožnovat správci zobrazit si statistky všech zaměstnanců.

4.2.1.3 Kalendář

9. Aplikace bude umožnovat přidávat nové události.
10. Aplikace bude umožnovat zobrazovat události, ke kterému má oprávnění.
11. Aplikace bude umožnovat upravovat události, ke kterému má oprávnění.
12. Aplikace bude umožnovat upomínání na blížící se události zainteresovaným osobám.
13. Aplikace bude umožnovat mazat události, ke kterému má oprávnění.

4.2.1.4 Zprávy

14. Aplikace bude umožnovat posílání zpráv.
15. Aplikace bude umožnovat upozornění na příchozí zprávy.
16. Aplikace bude umožnovat přečtení příchozích zpráv.
17. Aplikace bude umožnovat mazání zpráv.
18. Aplikace bude umožnovat nástěnku, na kterou můžou všichni napsat příspěvek.
19. Aplikace bude umožnovat správci mazat příspěvky z nástěnky.

4.2.2 **Nefunkční požadavky**

Na konzultační schůzi jsme přišli na to, že nejvhodnějším řešením by bylo, nahrát novou aplikaci na jejich interní server, který je velmi málo vytížený. Z toho plynou další nefunkční požadavky na aplikaci, která musí běžet na Windows serveru.

20. Aplikace bude běžet na Windows serveru.
21. Aplikace bude využívat šifrovanou komunikaci.
22. Aplikace nepovolí přístup nepřihlášenému uživateli.
23. Aplikace bude přístupná z jakéhokoliv prohlížeče.
24. Aplikace bude hierarchicky dělit projekty na podprojekty a dále na úkoly.
25. Aplikace bude responzivní.

4.2.3 **Role**

Následně jsme si stanovili role, které budou implementovány do aplikace. Klient zatím požaduje pouze dvě role: zaměstnance a správce. Správce by měl mít všechny funkcionality jako zaměstnanec, z tohoto důvodu implementujeme roli třetí – uživatele, ze které budou oba, zaměstnanec i správce, dědit veškerou funkcionalitu. Toto umožnuje snazší úpravu kompetencí a do budoucna i přidávání dalších rolí.

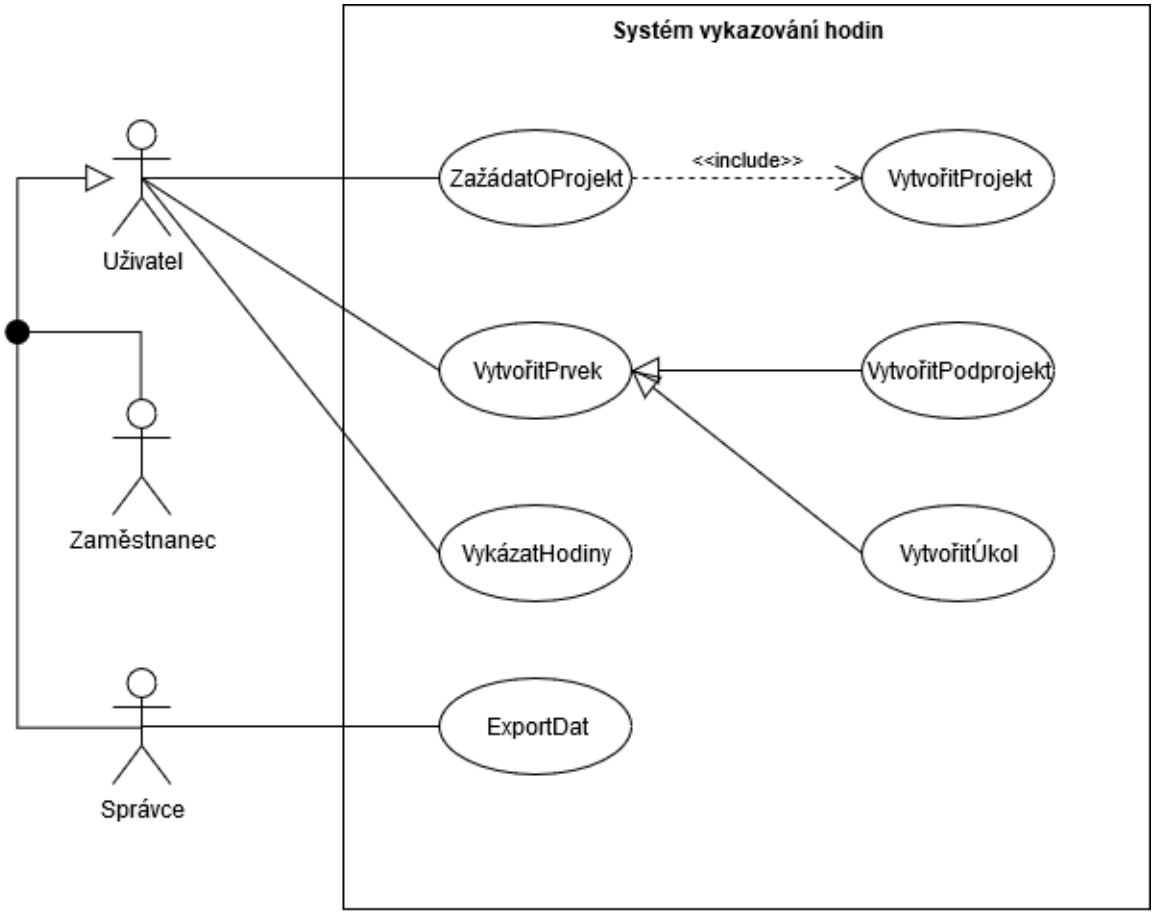
NÁZEV ROLE	POPIS
Uživatel	Uživatel je pouze abstraktní role, určená pro dědění základních funkcionalit.
Zaměstnanec	Uživatel, který pracuje s danou aplikací.
Správce	Uživatel, který je zodpovědný za správu aplikace.

Tabulka 1 Přehled rolí

4.3 Případy užití

Z funkčních požadavků klienta jsme vytvořili případy užití (use cases), které přibližují cílovou aplikaci a práci s ní. Ve své práci se, z důvodu rozsahu tohoto tématu, věnuji pouze těm funkčním požadavkům, které jsou uvedené v kapitole 4.2.1.1 a týkají se vykazování hodin. Tyto případy užití jsme zakreslili do následujícího obrázku – diagramu případů užití.

Na následujícím obrázku vidíme subjekt – Systém vykazování hodin, který ukazuje jednotlivé aktéry a funkce, které mají.



Obrázek 12 Diagram případu užití subjektu – Systém vykazování hodin

Případy užití zobrazeny na předchozím obrázku jsou dále detailně rozpracovány do následujících tabulek. Obsah těchto tabulek je jednoduchý, tak aby byl srozumitelný pro všechny zúčastněné, včetně klienta. Zároveň jsou tyto případy užití záměrně repetitivní, což usnadňuje práci analytikům a programátorům.

Každá tabulka obsahuje název případu užití, jedinečný identifikátor (Id), stručný popis případu užití, primární a vedlejší aktéry, vstupní a výstupní podmínky a také hlavní a alternativní scénář. Alternativní scénáře jednotlivých případu užití jsou dále podrobně rozpracovány pod stejným Id a zahrnují všechny možné scénáře.

Na následujícím případu užití, v tabulce 2 Zažádat o projekt, vidíme použití klíčových slov KDYŽ a ZAHRNOUT.

Klíčové slovo KDYŽ (někdo používá anglické IF), představuje rozvětvení hlavního scénáře. Mnoho analytiků toto větvení neuznává z důvodu, že větvení uvnitř scénáře zvyšuje jeho složitost a raději by vytvořili nový alternativní scénář. Ale z pragmatického hlediska,

zde toto snižuje celkový počet případů užití, tudíž výsledný model je kompaktnější, a to pouze na úkor přidání jen jedné jednoduché větve do případu užití.

Pokud nastane daná podmínka a správce opravdu odsouhlasí vytvoření daného projektu, pak pomocí relace include a klíčového slova ZAHRNOUT, zde zahrneme chování případu užití Vytvořit projekt z tabulky 6. Nejčastěji se tohoto využívá, pokud daný případ užití využívá více dceřiných případů užití, což v tomto případě není pravda. Ale v této situaci se tedy myslí na budoucí možné rozšíření a samozřejmě i na lepší možnost otestování.

Případ užití:	ZažádatOProjekt
Id:	1
Stručný popis:	Uživatel si zažádá o vytvoření projektu. Daný projekt musí být schválen správcem.
Primární aktéři:	Uživatel
Vedlejší aktéři:	Správce
Vstupní podmínky:	1. Uživatel je přihlášen
Hlavní scénář:	1. Případ užití začíná, když uživatel vybere volbu "Zažádat o projekt". 2. Uživatel zadá do formuláře název, popis, odhadovaný čas a uživatele, odpovědného za daný projekt. 3. Systém upozorní správce na žádost o založení nového projektu. 4. KDYŽ správce odsouhlasí vytvoření daného projektu, 4.1. ZAHRNOUT (VytvořitProjekt)
Výstupní podmínky:	Byl vytvořen nový projekt.
Alternativní scénář:	NeplatnéVyplněníFormuláře Storno Zamítnutí

Tabulka 2 Případ užití – Zažádat o projekt

Alternativní scénář:	ZažádatOProjekt:NeplatnéVyplněníFormuláře
Id:	1.1
Stručný popis:	Systém informuje zaměstnance, že zadal nesprávné údaje.
Primární aktéři:	Uživatel
Vedlejší aktéři:	Žádní
Vstupní podmínky:	1. Zaměstnanec zadal nesprávné údaje.
Hlavní scénář:	1. Alternativní scénář začíná krokem 2 hlavního scénáře. 2. Systém informuje zaměstnance, že zadal nesprávné údaje.
Výstupní podmínky:	Žádné

Tabulka 3 Případ užití – Zažádat o projekt: Neplatné vyplnění formuláře

Alternativní scénář:	ZažádatOProjekt:Storno
Id:	1.2
Stručný popis:	Uživatel zruší proces zažádání o projekt.
Primární aktéři:	Uživatel
Vedlejší aktéři:	Žádní
Vstupní podmínky:	Žádné
Hlavní scénář:	1. Alternativní scénář může začít kdykoliv. 2. Uživatel zruší proces zažádání o projekt.
Výstupní podmínky:	Zrušení žádosti o projekt.

Tabulka 4 Případ užití – Zažádat o projekt: Storno

Alternativní scénář:	ZažádatOProjekt:Zamítnutí
Id:	1.3
Stručný popis:	Správce zamítne žádost o projekt.
Primární aktéři:	Správce
Vedlejší aktéři:	Žádní
Vstupní podmínky:	1. Uživatel zažádal o projekt.
Hlavní scénář:	1. Alternativní scénář začíná krokem 4 hlavního scénáře. 2. Správce zamítne zažádání o projekt.
Výstupní podmínky:	Zrušení žádosti o projekt.

Tabulka 5 Případ užití – Zažádat o projekt: Zamítnutí

Případ užití:	VytvořitProjekt
Id:	2
Stručný popis:	Správce vytvoří nový prvek.
Primární aktéři:	Správce
Vedlejší aktéři:	Žádní
Vstupní podmínky:	1. Správce je přihlášen
Hlavní scénář:	1. Systém vytvoří daný projekt.
Výstupní podmínky:	Byl vytvořen nový projekt.
Alternativní scénář:	Žádné

Tabulka 6 Případ užití – Vytvořit projekt

Případ užití v tabulce 7 – Vytvořit prvek, je obecný případ užití, ze kterého dědí funkce a vlastnosti případy užití s id 4 (Vytvořit podprojekt) a 5 (Vytvořit úkol).

Případ užití:	VytvořitPrvek
Id:	3
Stručný popis:	Uživatel vytvoří nový prvek.
Primární aktéři:	Uživatel
Vedlejší aktéři:	Žádní
Vstupní podmínky:	1. Uživatel je přihlášen
Hlavní scénář:	1. Případ užití začíná, když uživatel vybere volbu "Zažádat o nový prvek". 2. Uživatel zadá do formuláře požadované údaje. 3. Systém vytvoří daný prvek.
Výstupní podmínky:	Byl vytvořen nový projekt.
Alternativní scénář:	NeplatnéVyplněníFormuláře Storno

Tabulka 7 Případ užití – Vytvořit prvek

Alternativní scénář:	VytvořitPrvek: NeplatnéVyplněníFormuláře
Id:	3.1
Stručný popis:	Systém informuje uživatele, že zadal neplatné údaje.
Primární aktéři:	Uživatel
Vedlejší aktéři:	Žádní
Vstupní podmínky:	Žádné
Hlavní scénář:	1. Alternativní scénář začíná, krokem 2 hlavního scénáře. 2. Systém vyzve uživatele o nové zadání údajů.
Výstupní podmínky:	Uživatel nevytvoří nový prvek.

Tabulka 8 Případ užití – Vytvořit prvek: Neplatné vyplnění formuláře

Alternativní scénář:	VytvořitPrvek:Storno
Id:	3.2
Stručný popis:	Uživatel zruší proces vytváření nového prvku.
Primární aktéři:	Uživatel
Vedlejší aktéři:	Žádní
Vstupní podmínky:	Žádné
Hlavní scénář:	1. Alternativní scénář může začít kdykoliv. 2. Uživatel zruší proces vytváření nového prvku.
Výstupní podmínky:	Uživatel nevytvoří nový prvek.

Tabulka 9 Případ užití – Vytvořit prvek: Storno

Případ užití:	VytvořitPodprojekt
Id:	4
Id předka:	3
Stručný popis:	Uživatel vytvoří nový podprojekt.
Primární aktéři:	Uživatel
Vedlejší aktéři:	Žádní
Vstupní podmínky:	1. Uživatel je přihlášen
Hlavní scénář:	1. (o.1) Případ užití začíná, když uživatel vybere volbu "Zažádat o nový podprojekt". 2. (o.2) Uživatel zadá do formuláře název, popis, odhadovaný čas a uživatele, odpovědného za daný podprojekt. 3. (o.3) Systém vytvoří daný podprojekt.
Výstupní podmínky:	Byl vytvořen nový podprojekt.
Alternativní scénář:	NeplatnéVyplněníFormuláře Storno

Tabulka 10 Případ užití – Vytvořit podprojekt

Alternativní scénář:	VytvořitPodprojekt: NeplatnéVyplněníFormuláře
Id:	4.1
Stručný popis:	Systém informuje uživatele, že zadal neplatné údaje.
Primární aktéři:	Uživatel
Vedlejší aktéři:	Žádní
Vstupní podmínky:	Žádné
Hlavní scénář:	1. Alternativní scénář začíná, krokem 2 hlavního scénáře. 2. Systém vyzve uživatele o nové zadání údajů.
Výstupní podmínky:	Uživatel nevytvoří nový prvek.

Tabulka 11 Případ užití – Vytvořit podprojekt: Neplatné vyplnění formuláře

Alternativní scénář:	VytvořitPodprojekt:Storno
Id:	4.2
Stručný popis:	Uživatel zruší proces vytváření nového prvku.
Primární aktéři:	Uživatel
Vedlejší aktéři:	Žádní
Vstupní podmínky:	Žádné
Hlavní scénář:	1. Alternativní scénář může začít kdykoliv. 2. Uživatel zruší proces vytváření nového prvku.
Výstupní podmínky:	Uživatel nevytvoří nový prvek.

Tabulka 12 Případ užití – Vytvořit podprojekt: Storno

Případ užití:	VytvořitUkol
Id:	5
Id předka:	3
Stručný popis:	Uživatel vytvoří nový úkol.
Primární aktéři:	Uživatel
Vedlejší aktéři:	Žádní
Vstupní podmínky:	1. Uživatel je přihlášen
Hlavní scénář:	1. (o.1) Případ užití začíná, když uživatel vybere volbu "Zažádat o nový úkol". 2. (o.2) Uživatel zadá do formuláře název, popis, odhadovaný čas a uživatele, odpovědného za daný úkol. 3. (o.3) Systém vytvoří daný úkol.
Výstupní podmínky:	Byl vytvořen nový úkol.
Alternativní scénář:	NeplatnéVyplněníFormuláře Storno

Tabulka 13 Případ užití – Vytvořit úkol

Alternativní scénář:	VytvořitUkol: NeplatnéVyplněníFormuláře
Id:	5.1
Stručný popis:	Systém informuje uživatele, že zadal neplatné údaje.
Primární aktéři:	Uživatel
Vedlejší aktéři:	Žádní
Vstupní podmínky:	Žádné
Hlavní scénář:	1. Alternativní scénář začíná, krokem 2 hlavního scénáře. 2. Systém vyzve uživatele o nové zadání údajů.
Výstupní podmínky:	Uživatel nevytvoří nový prvek.

Tabulka 14 Případ užití – Vytvořit úkol: Neplatné vyplnění formuláře

Alternativní scénář:	VytvořitUkol:Storno
Id:	5.2
Stručný popis:	Uživatel zruší proces vytváření nového prvku.
Primární aktéři:	Uživatel
Vedlejší aktéři:	Žádní
Vstupní podmínky:	Žádné
Hlavní scénář:	1. Alternativní scénář může začít kdykoliv. 2. Uživatel zruší proces vytváření nového prvku.
Výstupní podmínky:	Uživatel nevytvoří nový prvek.

Tabulka 15 Případ užití – Vytvořit úkol: Storno

Případ užití:	VykázatHodiny
Id:	6
Stručný popis:	Uživatel si hned po odpracování, vykáže hodiny.
Primární aktéři:	Uživatel
Vedlejší aktéři:	Žádní
Vstupní podmínky:	1. Uživatel je přihlášen
Hlavní scénář:	1. Případ užití začíná, když uživatel vybere volbu "Vykázat hodiny". 2. Uživatel zadá do formuláře datum a počet hodin.
Výstupní podmínky:	Hodiny jsou vykázány.
Alternativní scénář:	NeplatnýDatum NeplatnýPočetHodin Storno

Tabulka 16 Případ užití – Vykázat hodiny

Alternativní scénář:	VykázatHodiny:NeplatnýDatum
Id:	6.1
Stručný popis:	Systém informuje zaměstnance, že zadal neplatný datum.
Primární aktéři:	Zaměstnanec
Vedlejší aktéři:	Žádní
Vstupní podmínky:	1. Zaměstnanec zadal neplatný datum
Hlavní scénář:	1. Alternativní scénář začíná, krokem 2 hlavního scénáře. 2. Systém informuje zaměstnance, že zadal neplatný datum.
Výstupní podmínky:	Žádné

Tabulka 17 Případ užití – Vykázat hodiny: Neplatný datum

Alternativní scénář:	VykázatHodiny:NeplatnýPočetHodin
Id:	6.2
Stručný popis:	Systém informuje zaměstnance, že zadal neplatný počet hodin.
Primární aktéři:	Zaměstnanec
Vedlejší aktéři:	Žádní
Vstupní podmínky:	1. Zaměstnanec zadal neplatný počet hodin
Hlavní scénář:	1. Alternativní scénář začíná, krokem 2 hlavního scénáře. 2. Systém informuje zaměstnance, že zadal neplatný počet hodin.
Výstupní podmínky:	Žádné

Tabulka 18 Případ užití – Vykázat hodiny: Neplatný počet hodin

Alternativní scénář:	VykázatHodiny:Storno
Id:	6.3
Stručný popis:	Zaměstnanec zruší proces vykazování hodin.
Primární aktéři:	Zaměstnanec
Vedlejší aktéři:	Žádní
Vstupní podmínky:	Žádné
Hlavní scénář:	1. Alternativní scénář může začít kdykoliv. 2. Zaměstnanec zruší proces vykazování hodin.
Výstupní podmínky:	Zaměstnanec nevykáže žádné hodiny.

Tabulka 19 Případ užití – Vykázat hodiny: Storno

Případ užití:	ExportDat
Id:	7
Stručný popis:	Správce exportuje data do formátu XML.
Primární aktéři:	Správce
Vedlejší aktéři:	Žádní
Vstupní podmínky:	1. Správce je přihlášen
Hlavní scénář:	1. Případ užití začíná, když správce vybere volbu "Export dat". 2. Správce dále specifikuje data, která chce exportovat. (období, zaměstnanci)
Výstupní podmínky:	Data byly exportována.
Alternativní scénář:	Žádný

Tabulka 20 Případ užití – Export dat

5 Výsledky a diskuse

V této práci jsem se zabýval analýzou a tvorbou modelu uživatelských požadavků dle metodiky Unified Process pro podnikovou webovou aplikaci vykazování hodin. Tato aplikace by měla sloužit k vykazování hodin zaměstnanců do jednotlivých projektů.

Výsledkem této práce je model uživatelských požadavků, včetně odpovídajících případů užití. Tento výsledek je základ pro tvorbu výše uvedené aplikace.

Pro dokončení této aplikace je zapotřebí pokračovat ve vývoji pomocí Unified Process. Je nutné pokračovat analýzou, implementací a celý proces uzavřít nasazením, což ale není předmětem této bakalářské práce.

6 Závěr

Cílem bakalářské práce byla analýza a vytvoření modelu uživatelských požadavků, týkající se vykazování hodin zaměstnanců v rámci podnikové webové aplikace a jejich následné zpracování do jednotlivých případů užití.

Klient je majitelem středně velkého podniku na výrobu reklamních předmětů a z důvodu expanze podniku požadoval vytvoření této aplikace.

Požadavky na vytvoření této aplikace vycházely z konkrétních podmínek, potřeb a možností tohoto podniku. Na jednotlivých konzultacích s klientem jsme tyto požadavky neustále upřesňovali, konzultovali, tak aby jim maximálně vyhovovaly.

Na základě těchto požadavků jsem následně vypracoval jednotlivé případy užití, což je první krok k vytvoření aplikace.

K dokončení této aplikace je zapotřebí pokračovat analýzou, implementací a celý proces uzavřít nasazením, což není předmětem této bakalářské práce.

7 Bibliografie

(1) CRIPPS, Peter a Peter EELES. *Architektura softwaru*. Brno: Computer Press, 2010.

(2) FERNHOLZ, Tim. What it took for Elon Musk's SpaceX to disrupt Boeing, leapfrog NASA, and become a serious space company. *Quartz* [online]. 2014 [cit. 2020]. Dostupné z: https://qz.com/281619/what-it-took-for-elon-musks-spacex-to-disrupt-boeing-leapfrog-nasa-and-become-a-serious-space-company/

(3) ARLOW, Jim a Ila NEUSTADT. *UML 2 a unifikovaný proces vývoje aplikací*. Brno: COMPUTER PRESS, 2008.

(4) VALASEK, Stanislav . CASE modelovacie nástroje. *Stanislav Valasek* [online]. 2012 [cit. 2020]. Dostupné z: https://valasek.wordpress.com/2012/05/02/case-modelovacie-nastroje/

(5) KANISOVÁ, Hana a Miroslav MÜLLER. *UML srozumitelně*. Brno: COMPUTER PRESS, 2007.

(6) Úvod do Unified Process (UP). *Materiály by Trtkal* [online]. 2015 [cit. 2020]. Dostupné z: http://fei.mtrakal.cz/materialy_public/7.semestr/%5b2010-2011%5dINPSW_Simerda/prednasky/02_UPIntroduction.pdf

(7) PETE , Ness a Thomas LEE. The Rational Unified Process for testers. *Ibm* [online]. 2005 [cit. 2020]. Dostupné z: https://www.ibm.com/developerworks/rational/library/04/r-3239/r-3239-pdf.pdf